Utrecht University

MASTER'S THESIS

# Active Selection of Classification Features


*Author:*
Thomas KOK

*Supervisors:*
Dr. habil. Georg KREMPL
Dr. Ad FEELDERS

*External Supervisors:*
Dr. Hugo SCHNACK
Dr. Rachel BROUWER
Dr. René MANDL


July 20, 2020

# Contents

# List of Figures

# List of Tables

# Part I

# Introduction

# Chapter 1

# Problem Statement

Building a classification model can require data points that are costly to obtain, related either to time, money or personal comfort. To save on these fronts, it would be optimal if the number of data points that is needed to build a model could be minimized with regards to its performance. An important example of such a model is constructing a classification model for schizophrenia prediction [46]. Demographic information is given for training, as well as the true label, but an MRI (Magnetic Resonance Imaging) scan can be done at a relatively high cost. The final classifier predicts the label, only from the MRI features. These scans are expensive, and any reduction in scans needed is a notable improvement.

This problem, that we will explore in this thesis is one that is both theoretically and practically motivated. The theoretical motivation is an academic one, expanding on previous research on active learning, active feature acquisition, instance completion and active class selection. The practical motivation is from a medical source, where we aim to minimize the number of MRI scans needed to more economically build a model.

The practical motivations for this problem indicate its social relevance, as this is a problem with relevance in its own field. Solving this problem also allows it to be generalized to other settings. Any setting with costly features, and more cheaply available features that can not be used for prediction can make use of a potential solution to this problem. Other potential use cases include problems such as risk assessment, customer modelling, and personalized recommendation systems.

## 1.1 Problem Definition

Given dataset $\mathcal{D}$ consisting of known *selection features* $x$, unknown *classification features* $z$ and known binary labels $y$. Use active learning to find the next subject, whose classification features are estimated to be most useful to create classifier $\Gamma$. $\Gamma$ takes the classification features as input, and has the binary labels as output. Repeat this process until a certain stopping condition is found. This method can be evaluated on both the performance of $\Gamma$ using known performance measures, and the number of subjects queried which should be low. The goal is maximizing and minimizing these respectively.

## 1.2 Research Questions

The following research questions have been defined for the problem:

- What approaches work best to actively select classification features?

- Does actively selecting data points with classification features improve the performance, as opposed to passive learning or randomly selecting data points?
- Can we reduce the number of MRI scans (known data points) needed, while retaining a similar performance on classifying schizophrenia patients (unknown data points)?
- Can we improve the performance on classifying schizophrenia patients, while retaining the same number of MRI scans needed?
- Is it attainable to improve the general performance of active selection of classification features, using known performance measures such as AUC?
- Can we only improve performance when there is observable correlation between the two sets of features?

## 1.3 Notation

In this section, we will define the notation used in the thesis. The notation in the literature study (chapters 2, 3) might differ, as previous literature discusses differing problems.

TABLE 1.1: Most of the notation used in this thesis.

| Symbol | Meaning | Symbol | Meaning |
|---|---|---|---|
| $\mathcal{D}$ | Dataset. | $i^*$ | The optimal instance to query. |
| $\mathcal{I}$ | Incomplete instances. | $\mathcal{U}()$ | Utility function. |
| $\mathcal{C}$ | Complete instances. | $P()$ | Probability function. |
| x | Selection features. | $\theta$ | Imputation model. |
| z | Classification features. | $\lambda$ | Score balance parameter. |
| $i$ | Input vector (x, z). | $\Gamma$ | Classifier. |
| $y$ | Label. | $d$ | Density. |
| $F$ | All input vectors. | $\beta$ | Batch size. |
| $Y$ | All labels. | $\phi$ | Active learning method. |

## 1.4 Implementation

The implementation used for this thesis, and all relevant experiments, has been developed using `Python 3` [63]. Several packages were used for relevant components of the project, such as:

1. `scikit-learn` [50] for many useful machine learning functions, such as classification models and cross validation.
2. `matplotlib` [29] and `seaborn` [66] for visualization of results from experiments, as seen in this thesis.
3. `pandas` [59, 41] for data organization.
4. `numpy` [47, 62] and `scipy` [65] for scientific programming.

The entire project and source code, as well as the results from the experiments can be downloaded from `https://github.com/thomastkok/active-selection-of-classification-features`.

# Chapter 2

# Background

In this chapter, we will give a background of the general research areas which this thesis relates to.

## 2.1 Machine Learning

Machine learning is a subfield of computer science and artificial intelligence, which studies the idea of learning from examples, finding interesting patterns in data, and being able to perform tasks without specific instructions.

Data is collected in many different fields, and machine learning allows us to analyze data in all of them, without human intervention. This can save time and effort, while also finding patterns that a human might not notice.

Machine learning is generally split up into *supervised* and *unsupervised* learning. Supervised learning focuses on predictive information, where the aim is to construct an input-output mapping based on existing input-output pairs. Unsupervised learning focuses on descriptive information, where no set output is given. Instead, information is learned from the data itself. For example, *clustering* divides a dataset into several groups or clusters, based on the features of each data point.

As our problem relates to predicting a label from a given input, it is a supervised learning problem: it learns from previous examples with this given label to find a trend within these examples.

Additional forms of machine learning are *semi-supervised* learning, which lies in-between supervised and unsupervised learning, and *reinforcement learning*, where the problem is interactive and the learning process is part of the results.

For an extensive overview of machine learning, refer to [45].

## 2.2 Supervised Learning

Supervised learning models aim to predict an output $y$ from a set of input values $x$. This output value $y$ can either be a categorical value or a numerical value. In the former case it is called a *classification* problem, while in the latter case it is called a *regression* problem.

The input vector $x$ can consist of any number of variables, while the output variable $y$ generally consists of only one value (although it can be multiple). The dataset consisting of previous known examples consists of any number of these vectors: $D = \{(x_i, y_i)\}_{i=1}^{N}$.

The aim is to construct a model or function which maps $x$ to $y$: $f(x) = y$, with minimal error. This error is defined by a specific performance measure, such as accuracy for a classification problem or the mean squared error for a regression problem. The accuracy is defined as the ratio of correctly predicted examples, while the mean

squared error sums the squared difference between each prediction and the true value.

To accomodate finding this error value, the dataset is split up into a *training set* and a *test set*. The training set is used to train the model, approximating the function. The test set is then used to evaluate the performance of the function with new data, which is sampled from the same true distribution. The reason for this split is to avoid inherent bias to specific examples or data points, which are simply coincidental within the sample. The unknown test set is still approximated with the same distribution, but learning noise is avoided. This problem of learning coincidental information from the sample is known as *overfitting*.

### 2.2.1 Classification

Classification problems aim to classify given instances: given a picture of an animal, we might want to define this as either a cat or a dog. In this case, $y \in \{1, .., C\}$ with $C$ being the number of classes.

A special, common case is where $y$ only has two potential values: 0 and 1. This is known as *binary classification*, where $C = 2$. Real-world cases are ones such as illness diagnosis, or fraud detection. The problem we consider in this thesis is a binary classification problem, as it relates to our practical motivations and can often be generalized to multiclass classification.

Classification models are widespread in real-world applications. Document classification can be used to filter spam emails by either classifying them as spam (1) or not spam (1). Handwriting recognition classifies each input letter as $a$, $b$, .., $z$.

### 2.2.2 Regression

Regression problems aim to approximate an output $y \in \mathcal{R}$, which is a numerical value: given a lot size, location and number of bedrooms we might want to estimate the value of a house.

Other real-world examples of regression models are predicting the stock market price given current conditions, predicting the age of a website's visitor, and predicting the temperature at a given location given the known weather conditions.

## 2.3 Active Learning

Active learning is a variant of machine learning. The resulting machine learning model is the same as a supervised learning model, but the process is different. An active learning problem has available data, where the majority or even all data is unlabeled. Active learning then aims to select the most informative instances to be labeled. A key aspect of the active learning problem is that labeling is costly, and the goal is aiming to more economically train a model.

There are many applications for active learning, as any setting where obtaining labels is associated with some sort of cost can help decrease these costs or increase performance with selection. Examples are speech recognition—where noting the desired output requires a lot of time, personalized recommendation— where the dataset is constructed by querying the user, and document classification.

The active learning process is shown in figure 2.1. An unlabeled dataset is known, but obtaining the labels is costly. The problem then lies in selecting the data points that return the most useful information for their cost.

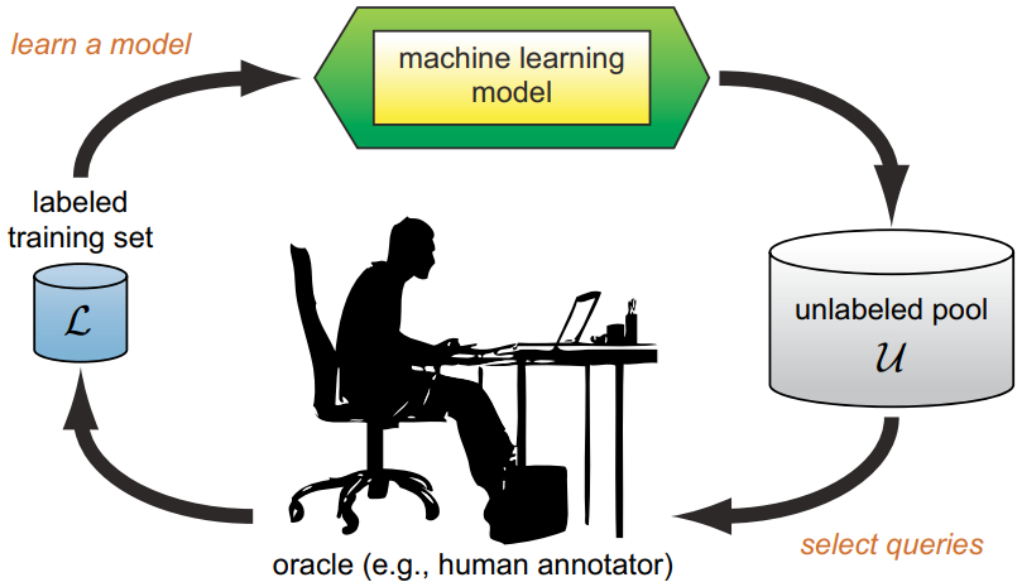



FIGURE 2.1: The pool-based active learning cycle. [57]

The problem setting for this thesis is slightly different, but the process holds the same principles. For each instance, its classification features z are missing. These can be queried, where the oracle will return its true values. The training set and current instantiation of the model can then be updated, repeating the process.

For an overview of active learning research, see [57].

## 2.4 Data Imputation

*Data imputation* is the problem of estimating, or otherwise dealing with, missing values within a dataset. This is commonly used in situations where the dataset is missing values, but does not have enough instances to simply drop all instances with missing values. Imputing the missing values allows an estimation of what those instances would be with complete information, and can then still be used for training purposes.

An overview of data imputation for missing data is provided in [24]—although this is missing specific feature-value pairs. It suggests general methods which include dataset reduction and other infeasible methods for our problem. For data imputation, it suggests methods such as selecting the mean, selecting the most similar known data point, and training a regression model.

Older statistically-based models are used to estimate specific values in [72, 71]. Bootstrapping can then be used to estimate variance or probabilities of specific values. The relevant papers mentioned are [1, 37, 55].

A frequency-based probability estimation, purely based on the class label, is used in [53].

## 2.5 Subgroup Partioning

*Subgroup partioning* aims to construct a partioning of the dataset, where each division considered its own subgroup with semantic meaning. Essentially, a custom set of

classes is constructed. For example, one might construct the class of *women with positive diagnosis of age 50 and higher*, from the medical diagnosis problem setting.

Similar research has been done in the area of *conceptual clustering* [51]. This research area considers clustering of datasets, but with each cluster being explainable.

Alternatively, clusters do need to be conceptually-based, but be able to be described in a certain way. This can be done by any grid-like partition of the feature space. This can be enforced by clustering via decision tree construction, as in [12, 38].

Constructing a partitional clustering via decision tree construction is considered in [12], replacing the split heuristic (as there is no classification to be optimized). The two heuristics are *box volume* and *graph closeness*. The box volume heuristic defines the impurity of a split by constructing a box for each partition, containing 0.95 of samples along each feature axis. The volume of this box is then the impurity. Alternatively, the graph closeness constructs a graph for each partition, where each data point is a vertex, having edges to its $k$ nearest neighbors, with each edge weighted with its Euclidean distance. The impurity is then defined as the inverse of the sum of weights.

As the decision tree is constructed, the clusters are all very small. This step is followed by a cluster agglomeration step, where the smaller clusters are combined. This is done by one of several similarity metrics between clusters (prototype: where the mean is compared, box distance: similar to the box volume heuristic, graph connectivity: similar to the graph closeness heuristic).

The results are then compared, with graph closeness performing better than box volume, and prototyping performing better than the two other agglomeration methods. However, all methods do not perform as well as a $k$-means clustering followed by a decision tree step which uses the clusters as class labels.

## 2.6 Mixture Models

Data which is empirically obtained, can often be represented with Gaussian distributions [42]. We can assume the same for our hidden classification features in our specific use case, as well as a lot of other real-world scenarios. In practice, data might actually consist of a *mixture* of many of these normal distributions. Using *Gaussian Mixture Models* allows us to model a distribution, which consists of multiple normal distributions.

It is easy to think of examples of this in real-world scenarios. For example, modeling a distribution of the height of a set of people. Looking at the entire distribution, it more than likely is not normally distributed. However, when considering the men and women in this dataset, it might show that within each of these genders there is in fact a normal distribution.

This section shows a very succinct overview of mixture models, for a more in-depth overview, refer to [42].

### 2.6.1 Gaussian Mixture Models

A Gaussian or normal distribution can be defined with a mean $\mu$ and a standard deviation $\sigma$. The distribution is then symmetrically centered around $\mu$, with $\sigma$ defining the spread.

Let $f_i(x|\mu, \sigma)$ be normal distribution for the $i$th component. The mixture model consisting of two normal distributions can be defined as:

$$f(x) = \pi_0 f_0(x|\mu_0, \sigma_0) + \pi_1 f_1(x|\mu_1, \sigma_1) \tag{2.1}$$

$\pi_i$ represents the probability of a random data point belonging to component $i$. So, the complete distribution is determined by adding the proportions of each component multiplied by their Gaussian distribution. Sampling a new instance from this distribution can be done by first sampling from the distribution of proportions, then sampling from the distribution of the selected component.

### 2.6.2 Expectation Maximization

The most commonly used method to finding the components and their proportions and distributions, is by using the *Expectation Maximization* algorithm. It repeatedly performs an expectation step and a maximization step until it converges.

The method is based on *maximum likelihood*. The maximum likelihood represents the set of parameter values $(\pi, \mu, \sigma)$ for which the observed data is more probable than for any other values of $(\pi, \mu, \sigma)$.

We find this likelihood with our expectation-maximization procedure.

Consider that for each data point $x$, there are $K$ possible Gaussian models it can belong to. Combining each of these $K$ Gaussian models, the probability of a value occuring can be defined as $P(X = x) = \sum_{k=1}^{K} \pi_k P(X = x|Z = k)$, where Z represents the latent or unseen variable noting which Gaussian model or cluster the data point belongs to, where $P(X = x|Z = k)$ is the normal distribution $(N(\mu_k, \sigma_k))$.

Therefore, the probability for each data point being the value that it is can be defined as:

$$P(X_1 = x_1, ..., X_n = x_n) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k N(x_i; \mu_k, \sigma_k) \tag{2.2}$$

With maximum likelihood, we aim to optimize this probability. The highest value of this probability given a set of instantations for all $(\pi, \mu, \sigma)$ values is the most likely and thus best option.

For mathematical purposes, we use the loglikelihood, which is defined as follows:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log(\sum_{k=1}^{K} \pi_k N(x_i; \mu_k, \sigma_k)) \tag{2.3}$$

The expectation maximization algorithm then tries to find this optimum value with the iterative process of repeating the *Expectation step (E-step)* and the *Maximization step (M-step)*.

1. Initialize $\theta$: all values for $\pi_k, \mu_k, \sigma_k$. This can be done randomly, or heuristically.
2. Repeat until local optimum is found:
   (a) **E-step:** Evaluate the distributions of each of the $K$ clusters using the current value for $\theta$.
   (b) **M-step:** Estimate new values for $\theta$ using the found distributions.
   (c) Evaluate the new likelihood given the new values for $\theta$. If it has not improved (significantly), the optimum is found.

Because the EM process returns a local maximum or saddle point, it can be wise to repeat the process if the exact result is important.

# Chapter 3

# Related Work

In this chapter, we will give an overview of previous work done in related areas. Some sections will give context for the other work, while other sections will be used as reference material later in this thesis.

## 3.1 Active Feature Acquisition

*Active feature acquisition* aims to query missing feature-value pairs in the dataset, to improve the classifier using this data. The exact problem statement can differ somewhat, as for some problems all feature values can be queried at once, while for other problems all instance values can be queried at once. The most common problem instantiations are emphactive feature-value acquisition, where each feature-value pair is queried individually, and *instance completion*, where the features of an instance are queried all at once.

In this section, the most relevant papers are referenced and summarized. An overview of relevant papers is also provided in [4].

### 3.1.1 Active Feature-Value Acquisition

The active feature-value acquisition problem is introduced in [53]. It queries missing feature-value, and considers the problem where an incomplete feature matrix $F$ is given, a complete label set and a cost matrix $C$ corresponding to the feature matrix. The aim is to construct the best-performing classifier given that queries for $F_{i,j}$ can be placed at cost $C_{i,j}$. The cost matrix $C$ is optional, and can be defaulted to a matrix of all 1's.

The method builds on [43, 44] which introduce the *Sampled Expected Utility* approach. The approach of selecting the next queried feature-value is as follows:

1. Compute a score for each potential query, indicating the increase in performance of the classifier given the new information
2. Select the query (or subset of queries with step-size $\beta$) with the highest score
3. Acquire the relevant feature-values, and repeat

The score for each potential query is calculated by multiplying the probability of each potential value for the feature, with its utility. This utility is equal to the change in value of the classifier, divided by the cost of this query.

$$score(x) = \sum_{v \in V} P(v|x) \cdot U(v|x) \tag{3.1}$$

where $x$ is a feature-value pair, and $V$ is all potential values for $x$.

One thing to note is the slight restriction of this method for numerical variables. As there an infinite possible amount of values for these variables, a discretization method must be used. A simple method, is using buckets for the values. Each bucket represents a subset of the numerical space, such as $0 \leq x < 10$. Ten buckets are created, and each numerical value is considered to be in exactly one of the buckets and thus to be that category.

Applications of the *expected utility* framework are shown in [52] by applying it to e-commerce datasets, and in [60] by applying it to predict protein interaction. Active feature acquisition is applied to the problem of *matrix completion* in multiple papers [11, 14, 28], where all values in a partially complete matrix are to be found.

The general pseudocode is straightforward, but the steps themselves require some additional considerations. Computing the increase in performance needs to be defined properly, as well as the distribution of expected probabilities of possible values. These are used to estimate the score for each query.

#### Estimating contribution to induction

The contribution to induction—where induction is the ability of the model to construct a generalized machine learning model—is defined by simulating the values of the query. Higher contribution should give a higher utility value. Using the expected or possible values in addition to the known values, we retrain the classifier. This classifier is then compared to the existing classifier to determine the utility value of the query.

The utility value depends on the chosen performance measure to quantify the performance of the classifier. The most intuitive option, accuracy, presents itself with some issues. For example, consider a binary classifier. It will either classify as 0 or 1, which makes the accuracy very black and white. The paper suggests instead using the cross-entropy measure. This measure takes into account not just the prediction, but also the certainty of the prediction.

A different heuristic is suggested in [60]: using the difference in probability that the classifier predicts the label correctly. Determine the probability of label $L$, where $L$ is the correct label; add the feature-value information to the classifier, and determine the new probability. The increase in probability is the utility value.

#### Estimating value distributions

The other missing value is the probability for each value. As this is needed to calculate the score for each potential query, a probability is necessary for each feature-value pair. The paper suggests using only the class label to determine the probability, as it is the only certain variable to be present. This is done by frequency analysis: if for half of the instances in the dataset with label 1 have a specific value for a specific feature, the conditional probability for that feature-value pair given $y = 1$ is 0.5.

$$P(v|x) = \frac{N(v|y)}{N(y)} \tag{3.2}$$

where $N()$ is the count of the value within the dataset, and $y$ is the known label for $x$.

#### Reducing the consideration set

This method can be computationally complex for larger datasets, as for each missing feature-value pair a classifier needs to be trained for each potential feature-value combination. To circumvent this, the query set can be sampled.

The first suggested approach for sampling is sampling from the query set uniformly. As this is sampling is at random, the best queries can potentially be sampled out just as often as the worst queries. The second suggested approach is using error sampling. This approach aims to select the most informative instances. Instances that are considered informative are instances which are misclassified with the current data, and instances where the classifier is very uncertain.

### 3.1.2 Instance Completion

The first introduction of the active feature acquisition problem [72], is now known more specifically as *instance completion*. For this subproblem, all unknown features of instance are queried at once.

The problem considers available labels, and data points with missing features. The goal is to build a classifier using $K$ data points with all available features, and improve the performance when compared to using only features available in all $N$ data points. It mentions active learning as the main option for solving this problem, and considers two specific algorithms: *AVID* and *GODA*.

Both approaches are based on *imputation models*. These models estimate the values of the missing data, based on the existing data. The paper does not elaborate on these imputation models, but refers to previous papers [1, 37, 55] as imputation is a long-studied problem within statistics.

#### Algorithm AVID

The AVID (Acquisition based on Variance of Imputed Data) algorithm determines the data point with the lowest likelihood of the missing feature values being estimated from the available data (the highest imputation uncertainty).

This algorithm does not take into account the labels, nor does it consider the performance of a resulting classifier.

The pseudocode is as follows (notation has been changed to correspond with 1.3):

1. Sample an initial batch of $\beta$ instances to initialize $\mathcal{C}$
2. Repeat until $k$ instances are sampled:
   (a) Build $B$ bootstrapped imputation models $\theta$ using $\mathcal{C}$
   (b) For each instance $x$ in $\mathcal{I}$, determine the score:

$$\text{score}(x) = \sum_{i \in I} \sqrt{\sum_{j=1}^{B} (x_{ij} - \mu_i)^2 / B}$$

   (c) Select $\min(\beta, k - |\,\mathcal{C}\,|)$ instances:

$$x^* = \max(score(x))$$

Where $B$ is the number of bootstrap samples, $\beta$ is the step size, $k$ is the desired number of instances sampled, $\theta$ is the imputation model, $\mu_i$ is the mean of $x_i$ within all $B$ imputation models, and $I$ is all missing features.

The algorithm needs an initial set of instances to function, which are sampled in step 1. Step 2 is then repeated until the necessary amount of instances is sampled. It builds $B$ imputation models on the known data (2a), and uses these imputation models to determine a score for each instance (2b). This score function essentially determines the variance for each instance, by summing the imputation variance for each missing feature value. The instance (or instances) with the highest score is then selected (2c), as this instance is hardest to estimate its values and thus considered interesting to sample.

**Algorithm GODA**

The GODA (Goal-Oriented Data Acquisition) algorithm chooses data points to maximize the performance of the model using a given classifier. Its next point chosen is the one with the highest expected improvement of the model built on the data of the data points with all features available.

It estimates the expected improvement by guessing the data if the features of a data point are expanded, and building the classifier with this data. It guesses this data with the same imputation models as used in the AVID algorithm.

The pseudocode is as follows (notation has been changed to correspond with 1.3):

1. Sample an initial $\beta$ instances to initialize $\mathcal{C}$
2. Repeat until $k$ instances are sampled:
   (a) Build an imputation model $\theta$ from $\mathcal{C}$
   (b) For each instance $x$ in $\mathcal{I}$:
      i. Build classifier $\Gamma$ with $\mathcal{C} + x$
      ii. score$(x) = p(\Gamma)$ where $p$ is a performance measure
   (c) Select min$(\beta, k - |\mathcal{C}|)$ instances:

$$x^* = \max(score(x))$$

The algorithm needs an initial set of instances to function, which are sampled in step 1. In step 2, it samples new instances until the necessary amount of samples is reached. This is done by estimating the missing values with an imputation model (2a), and constructing a classifier with these new values (2b). The score is then the performance of this new classifier, where the instance (or instances) with the highest score is selected. This instance is estimated to construct the best performing classifier when sampled, and as such is selected.

**Dual objective**

The authors continue in another paper [71] with the same concepts, expanding on them by combining the score into a dual objective: both the score of the classifier and the score of the imputation model are used to evaluate each data point after each step. The imputation model is what is used to estimate the potential values for each unseen feature-value combination, and improving the score of the imputation

model intuitively implies better estimation of the score of the classifier in the next steps. These two scores are combined by a weighted addition.

The score of the imputation model is such that points which cannot be imputed well are more likely to be selected. This is determined by using multiple bootstrap samples to build imputation models, and finding the variance between them. If it is low, the data can be imputed well.

The score used in [71] is:

$$\text{score}(x) = \lambda \cdot \text{score}_{\text{AIVD}}(x) + (1 - \lambda) \cdot \text{score}_{\text{GODA}}(x) \tag{3.3}$$

In their experiments, $\lambda = 0.5$.

#### Joint Instance Selection

A similar, but different, goal-oriented method is introduced in [54]. It works with batches of size $k$, completing all instances in each batch. For each step, it determines which instances are misclassified with the current available information. For each of these misclassified instances, it determines what values for the unknown data would change the prediction of the classifier for this instance to the correct prediction. It then estimates the probability of these values being present in this data point. This algorithm is a more extensive version of the expected utility method, as this paper shows by simplifying its own method to expected utility.

This method is extended on in a later paper [19]. It considers the theoretically optimal classifier, and works towards approximating this classifier. All previous methods do not take into account this optimal classifier, but simply aim to improve its current classifier.

It aims to move towards this optimal classifier by defining a distance function between two classifiers based on the dataset. This distance is defined such that binary classifiers with similar predictions have low distance, while binary classifiers with different predictions have high distance. For multi-class classifiers, correctness of the prediction is considered instead of the predicted class. Aiming to diverge from the previous classifier with as large a distance as possible, by aiming to change only misclassified instances thus leads to approximating the optimal classifier.

### 3.1.3 Budgeted Learning

A similar problem is mentioned in [39]. A research project for a classifier for cancer subtypes has a fixed budget to perform tests on patients to create a dataset. This dataset is then used to train the classifier with maximized performance.

It considers the problem of selecting the next patient-test pair as a Markov Decision Process, where its experimentally best policy of selecting new information is *single feature lookahead*. This policy estimates the expected loss for each feature-value pair from the current belief state, and selects the lowest one. It continues in a greedy manner until the budget is exceeded.

An overview of Budgeted Learning research up to that point is provided in [17]. It also mentions active feature acquisition and its similarities to budgeted learning. The main differences it considers is the set budget for budgeted learning: active feature acquisition might have any other stopping condition, or not have costs associated to the acquisition of features.

Fundamentally, the difference between active feature acquisition and budgeted learning is that budgeted learning is in the realm of reinforcement learning. It considers each feature an action, and aims to select any feature with the highest reward.

Which data point is then selected is not necessarily part of the problem, as they are just samples from a distribution. This does not hold for our problem, as all features are obtained at once and selecting the correct data point is the exact problem. Another example of this is shown in [3], as it shows the application of active learning to the classic budgeted learning problem of the multi-armed bandit. This problem considers $n$ slot machines with different distributions, mapping slot machines to features and instances to a distribution resulting from the slot machine.

### 3.1.4 Active feature selection

There exist several papers [7, 64] which focus on active feature selection. Active feature selection problems consider adding all missing values for one feature. This is in a way the opposite of instance completion, where all missing features are added for one instance.

### 3.1.5 Active classification

Several papers [25, 30, 36, 70] also focus on acquiring feature values in an efficient way, but do so during the classification step. This changes the problem, as sampling now focuses on the performance of the selection. This leads to different approaches from the two-step problem of active feature acquisition.

## 3.2 Active Class Selection

Instead of considering querying for a feature set of all classification features, we can also consider our problem to be a instance of the *active class selection* approach. This approach can be seen as the opposite of the regular active learning approach. Instead of being given instances and querying for labels, the labels are given and instances can be queried. For our problem, we might consider the set of classification features as the instances, as labels are already given.

The problem of active class selection is introduced in [40]. It focuses on the problem where if $n$ training instances can be acquired from any class, what class distribution is optimal for maximizing learning performance.

This paper proposes five different methods, and compares their results. The best performing method is *redistricting*, where instances are selected from volatile class labels. This is defined as class labels from instances where their prediction from the last round is different from the round before. This prediction is from a classifier trained on the instances obtained by that round. The idea behind this method is that instances that lie close to a decision boundary can still be reasonably improved. This performs better than looking at accuracy, precisely because it does not take into account whether the classification made is actually correct: some classes are considered unlearnable or harder to learn than others. All methods can be seen in table 3.1.

TABLE 3.1: All methods for Active Class Selection introduced in [40].

| Method | Description |
|---|---|
| Uniform | Sample uniformly from all classes. |
| Inverse | Sample in proportion inverse to their accuracy on the last round. |
| Original Proportion | Sample in proportion to the original sample, relying on domain knowledge. |
| Accuracy Improvement | Sample in proportion to each classes' change in accuracy from the last round. |
| Redistricting | Sample in proportion to each classes' volatility. |

A later paper [69] applies the active class selection to the arousal classification problem. The results between methods differ, indicating that results for each method might be application-specific.

### 3.2.1 Probabilistic Active Learning for Active Class Selection

An alternative approach to the active class selection problem is introduced in [33], which adjusts and applies the Probabilistic Active Learning method to the active class selection problem. The probabilistic active learning method is introduced in [35], and extended upon in [34, 32].

Probabilistic active learning considers the *pgain* (probabilistic gain) for each potential query. This probabilistic gain depends on the instances in the neighbourhood: many known labels decrease the value and vice versa; an equal ratio of positive and negative labels increases the value and vice versa.

This method is mapped to the active class selection problem, by transforming the active class selection problem to a regular active learning problem. It samples multiple instances from each class, calculating the performance gain for each instance. For each class, a weighted sum of all samples determines the predicted gain; the class with the highest value is selected.

1. $n_p \leftarrow 25, M \leftarrow 3$
2. Repeat until stopping condition is reached:
   (a) For each class, model a distribution from the known instances
   (b) Sample $n_p$ pseudo-instances $X^p$ from this distribution
   (c) For each pseudo-instance $i$ in $X^p$:

$$\mathrm{pg}_i = \mathrm{pgain}(k, M)$$
$$\mathrm{pgain(k, M)} = \max_{m \leq M}\left(\frac{1}{m}(f(k,m) - f(k,0))\right)$$
where $f(k,m)$ is the expected performance given $k$ and $m$
and $k$ is the local label statistics

   (d) For each class $y$:

$$g_y = \sum_{i=1}^{X^p} \mathrm{pg}_i \cdot k_{i,y} / \left(\sum_{j=1}^{|X^p|} k_{j,y}\right)$$

(e) Select class: $y^* = \text{argmax}_y(g_y)$

(f) Request an instance $x^*$ from $y^*$, and add $(x^*, y^*)$ to $\mathcal{C}$

Step 1 initializes the parameters of the algorithm. The second step is repeated until the algorithm is finished: it simulates each class by creating its own distribution from the known data (2a), and samples pseudo-instances from this distribution (2b). This pseudo-instances are then used to estimate the increase in performance (2c), and these estimations are combined for each class (2d). The class with the highest score is selected (2e) and sampled (2f).

# Part II

# Approaches

# Chapter 4

# Active Classification Feature Acquisition

The first approach to be considered is the approach that is based on previous research on Active Feature Acquisition [72, 71, 53, 43, 44]. The Active Feature Acquisition is the most similar known problem to our problem, and thus provides a good base of solutions.

Most solutions to the Active Feature Acquisition problem are as follows: estimate the values of the missing features, then estimate the utility of the estimated features. There are some significant differences between the multiple options here, but the method boils down to the same. If the features are discrete, $\sum P(x|i) \cdot U(x,i)$ can be used. However, for continuous values (which we assume for our problem), it is not possible to concretely sum up *individual* probabilities. Instead, we can either estimate the mean or expected value for the distribution, or estimate an entire distribution and sample a number of instances to represent the distribution.

We need to keep the computational expenses in mind: as we are repeating this process for every instance, for every sample, this will be repeated a lot. Depending on the use case, this may or may not be a problem and we can adjust our method of utility estimation based on that. For example, a real-world method that samples a few times each week and wants accurate results does not mind waiting an hour for knowing the next sample. However, a real-time suggestion system or evaluation system might want the sampling to be a lot faster.

The basic algorithm is shown in 4.1.

LISTING 4.1: Active Classification Feature Acquisition

```
def ActiveClassificationFeatureAcquisition(x, z, Y):
    for each unsampled instance i:
        estimated_features = EstimateFeatures(i)
        estimated_utility[i] = EstimateUtility(estimated_features)
    return argmax(estimated_utility)
```

Where the method can then vary is within the two substeps: the estimation of the features and the estimation of the utility. The estimation of the missing classification features is usually considered an imputation problem, which is the same as a missing data problem. However, this does not apply to our problem. For there are some features that are always available (the selection features x), and some features that are always missing (the classification features z). This means that the input is always the same set of features, and the output is always the same set of features. This means the problem reduces to a regression problem, instead of an imputation problem. For this, we can use known methods such as linear regression.

## 4.1 Utility estimation

For the utility estimation, we dive more into the relevant theory. We base the possible methods on the ones found in existing literature. Not all methods proposed for Active Feature Acquisition can be mapped to our problem. This difference mainly originates from methods focusing on the already known features and its inductive value to the classification or the likeliness of predicting the label with the currently known features [43, 54, 19]. This is not applicable to our case, as all classification features z are sampled simultaneously and never known partially beforehand. We have investigated the following four methods:

1. *GODA*: Goal Oriented Data Acquisition, or Expected Utility Estimation [72, 44, 53]
2. *AVID*: Acquisition based on Variance of Imputed Data [72]
3. *Dual objective*: Weighted average of GODA and AVID [71]
4. *Probability-based*: Based on the probability of flipping the instance [60, 19]

In the following subsections, these four methods will be discussed and extended on. All methods assume an input of estimated classification features z and binary label $y$.

### 4.1.1 GODA

This method most intuitively focuses on the result of the increased utility. It considers the performance of the classifier as it currently is, and the performance of the classifier if the selected instance would be added. The difference is the increase in performance and thus the utility of sampling the instance [72].

$$U(\mathbf{z}, y, \mathcal{C}) = M(\mathcal{C} + \langle \mathbf{z}, y \rangle, \Gamma) - M(\mathcal{C}, \Gamma) \tag{4.1}$$

where:

$\mathcal{C}$ is the sampled data,

$M$ is the relevant performance measure,

$\Gamma$ is the relevant classifier model.

This models the expected improvement of the classifier model when adding the new instance (given the expected classification features).

As noted in [72], the performance measure used here is relevant for the end result and is an important consideration. Most measures used for evaluation purposes simply look at the how good the model is at predicting labels correctly. However, this is not beneficial for noting the actual improvement of the model: for the prediction of each instance it can then only factually increase performance once. For example, consider the well-known accuracy performance measure. It increases *only* when the prediction of an instance is changed from 0 to 1, or 1 to 0. As such, this means that only instances that are close to the decision boundary are affected.

The alternative is using a more informative metric that considers the distance to the decision boundary, as suggested in [53]. For Logistic Regression, we use *log loss*, and for Support Vector Machines, we use *hinge loss*.

Obtaining these predictions must be done with $\mathcal{C}$, as we can not use the test set and there is no validation set available. To do this, we use cross-validation with the leave-one-out strategy until enough instances are sampled to use 5-fold cross-validation.

### 4.1.2 AVID

This method focuses on the concept of obtaining the most information by sample, with the reasoning that the most new information will lead to the most useful classifier model in the end. To do this, we consider how well the missing classification features can be estimated. If these classification features can be properly esimated, then the information already known must be high. This is defined by the *variance* of the imputation of the missing data.

To do this, define $B$ regression models (for AFA these are imputation models) that predict the classification features from the selection features. The predictions of each of these models are then aggregated: the variance determines the amount of information known or unknown. A high variance implies that it is at the moment still difficult to estimate the classification features from what is known. On the other hand, a low variance implies a certain amount of consistency within these predictions and thus some level of information known. This implies that the instances with the highest variance for the estimations of their classification features are most informative and best to sample, having the highest utility score.

The utility is defined as follows [72]:

$$U(\mathrm{x}, \Theta) = var(\{\forall \theta \in \Theta : \theta(\mathrm{x})\}) \tag{4.2}$$

where:

$\theta$ is a regression model trained on a bootstrapped version of $\mathcal{C}$,

$\Theta$ is the set of $B$ imputation models.

### 4.1.3 Dual objective

The dual objective function combines the previous two methods—GODA and AVID. It is simply a weighted average of the two, aiming to balance the improvement of the model of the estimation of the classification features and the highest utility. It should be taken into account that the former is just as important as the latter. Although it seems logical that simply selecting the sample with the highest expected utility is best, it helps with later queries to improve the feature estimation model as well. With better estimations of the features, the estimated utility will also be closer to the truth.

The utility is defined as follows [71]:

$$U(\mathrm{x}, \Theta, \mathrm{z}, y, \mathcal{C}) = \lambda \cdot U_{AVID}(\mathrm{x}, \Theta) + (1 - \lambda) \cdot U_{GODA}(\mathrm{z}, y, \mathcal{C}) \tag{4.3}$$

where:

$\lambda$ is the hyperparameter to balance the dual objective.

As proposed in [71], we use $\lambda = 0.5$. Potential improvements could be made here by considering a simulated annealing-like approach, where $\lambda$ is initially high and is gradually lowered.

### 4.1.4 Probability-based

The fourth method works only with probability-based classifiers, as it requires a probability estimation of the labels. It uses the estimated classification features to check the probability of predicting the true label correctly, given those estimated features. Using the probability of the classifier's prediction allows us more information than just the prediction, and is significantly more efficient than the GODA method—as it only has to fit a classifier once, instead of $|\mathcal{I}| + 1$ times.

We can derive a method of selecting the probability from previous research in Active Feature Acquisition [60]. Their method maximizes the change in probability when sampling the specific feature-value. Consider that in the case of feature-value acquisition, the features used for classification are partially known. A probability can thus already be retrieved from the current classifier. Then, the estimated classification features are added, and the new probability—from the same classifier, so no need to retrain—is noted as well. The increase should optimally be as high as possible.

Mapping this to our problem setting results in a lack of previously known probability, as we do not have any known feature-values that can be used for classification. This can be adjusted for by setting the previously known probability to a constant $c$ for all instances:

$$U(\Gamma, \mathbf{z}, y) = P(\Gamma(\mathbf{z}) = y) - c \tag{4.4}$$

where:

$\Gamma$ is the classifier fit on $\mathcal{C}$,

$c$ is the determined constant.

To simplify this and remove the unknown $c$, we can simply drop this constant:

$$\underset{i \in \mathcal{I}}{\operatorname{argmax}}\, U(\Gamma, \mathbf{z}_i^*, y_i) = \tag{4.5}$$

$$\underset{i \in \mathcal{I}}{\operatorname{argmax}}\, P(\Gamma(\mathbf{z}_i^*) = y_i) - c = \tag{4.6}$$

$$\underset{i \in \mathcal{I}}{\operatorname{argmax}}\, P(\Gamma(\mathbf{z}_i^*) = y_i) - c + c = \tag{4.7}$$

$$\underset{i \in \mathcal{I}}{\operatorname{argmax}}\, P(\Gamma(\mathbf{z}_i^*) = y_i) \tag{4.8}$$

as $\operatorname{argmax}_X = \operatorname{argmax}_{X+c}$, so:

$$U(\Gamma, \mathbf{z}, y) = P(\Gamma(\mathbf{z}) = y) \tag{4.9}$$

where:

$\Gamma$ is the classifier fit on $\mathcal{C}$.

This might not make sense intuitively, as you are selecting instances where the likely classification features are already predicted well and might not need more information. In fact, this shows in the results in the next section as the results biases towards obtaining unnecessary information. We would in fact prefer to sample instances where the probability is closer to 0.5, or perhaps even lower.

We consider this by generalizing this utility method as described. [19] proposes a score function which depends on a desired probability $b$ for a perfect sample instance, and $p$ which is the expected probability of misclassification. Using this score

function, we can obtain a more generalized version of the Probability-based estimation (and show that the above is a special case where $b = 0$). Note that the probability of misclassification is considered, so not the probability of correct classification.

The score function is given as:

$$score_x = \frac{p(1-p)}{(-2b+1)p + b^2} \tag{4.10}$$

where:

$b$ is the asymmetry parameter,

$p$ is the probability of misclassification.

The $k$ instances with the highest score are then queried. We can use this score function as base for our probablity-based utility function, with the probability of misclassification using the classifier fit on $\mathcal{C}$ and the estimated classification features $\mathbf{z}^*$:

$$U(\Gamma, \mathbf{z}, y) = \frac{P(\Gamma(\mathbf{z}) \neq y) \cdot (1 - P(\Gamma(\mathbf{z}) \neq y))}{(-2b+1) \cdot P(\Gamma(\mathbf{z}) \neq y) + b^2} \tag{4.11}$$

which is the final probability-based utility function.

The previous version is indeed a special case of this function, where $b = 0$:

$$U(\Gamma, \mathbf{z}, y) = \tag{4.12}$$

$$\frac{P(\Gamma(\mathbf{z}) \neq y) \cdot (1 - P(\Gamma(\mathbf{z}) \neq y))}{(-2b+1) \cdot P(\Gamma(\mathbf{z}) \neq y) + b^2} = \tag{4.13}$$

$$\frac{P(\Gamma(\mathbf{z}) \neq y) \cdot (1 - P(\Gamma(\mathbf{z}) \neq y))}{(-2 \cdot 0 + 1) \cdot P(\Gamma(\mathbf{z}) \neq y) + 0^2} = \tag{4.14}$$

$$\frac{P(\Gamma(\mathbf{z}) \neq y) \cdot (1 - P(\Gamma(\mathbf{z}) \neq y))}{P(\Gamma(\mathbf{z}) \neq y)} = \tag{4.15}$$

$$\frac{P(\Gamma(\mathbf{z}) \neq y)}{P(\Gamma(\mathbf{z}) \neq y)} \cdot (1 - P(\Gamma(\mathbf{z}) \neq y)) = \tag{4.16}$$

$$(1 - P(\Gamma(\mathbf{z}) \neq y)) = \tag{4.17}$$

$$P(\Gamma(\mathbf{z}) = y) \tag{4.18}$$

However, as mentioned in [19], this is not an optimal value for $b$. More optimal values for $b$ should be close to but above 0.5. This accompanies the intuition that the most useful instances to query are those that are likely to be misclassified, but can potentially be correctly classified in the next iteration. This allows us to most efficiently approach the optimal classifier. As such, the value for $b$ proposed when sampling individual instances is suggested to be equal to $0.5 + \frac{1}{2 \cdot |\mathcal{C}|}$. This value is somewhat above 0.5, and decreases when the number of queried instances increases. This means the utility function can be rewritten to:

$$U(\Gamma, \mathbf{z}, y) = \frac{P(\Gamma(\mathbf{z}) \neq y) \cdot (1 - P(\Gamma(\mathbf{z}) \neq y))}{(-2(0.5 + \frac{1}{2 \cdot |\mathcal{C}|}) + 1) \cdot P(\Gamma(\mathbf{z}) \neq y) + (0.5 + \frac{1}{2 \cdot |\mathcal{C}|})^2} = \tag{4.19}$$

$$\frac{P(\Gamma(\mathbf{z}) \neq y) \cdot (1 - P(\Gamma(\mathbf{z}) \neq y))}{(-\frac{1}{|\mathcal{C}|}) \cdot P(\Gamma(\mathbf{z}) \neq y) + (0.5 + \frac{1}{2 \cdot |\mathcal{C}|})^2} \tag{4.20}$$

# Chapter 5

# Active Pseudo-Class Selection

In this chapter, we will look at the second approach. This approach is based on the existing research on Active Class Selection [40, 33], by constructing *pseudo-classes*. These pseudo-classes are then used to map the problem to an Active Class Selection problem, allowing us to use any existing Active Class Selection approach. This idea can be defined more concretely, as in listing 5.1.

LISTING 5.1: Active Pseudo-Class Selection

```
def ActivePseudoClassSelection(x, z, Y):
    pseudoclasses = PseudoClassConstruction(x + Y)
    sampled_pclass = ActiveClassSelection(pseudoclasses)
    instance = select one instance from sampled_pclass
    return instance
```

The main problem we need to solve, now that we have defined the idea behind this approach, is how to construct these pseudo-classes to perform the mapping from the Active Classification Feature Selection problem to the Active Class Selection problem. In addition, it is useful to consider which Active Class Selection method performs the best with this problem. Presumably, these are the ones which also work well in the regular Active Class Selection setting (3.2).

## 5.1 Pseudo-class construction

The pseudo-classes should be well-constructed: if these pseudo-classes are random or nonsensical partitions of the data, then the resulting mapping leads to a useless problem. The pseudo-classes should make some sense: they should have as high as possible intraclass correlation and low interclass correlation. In addition, these pseudo-classes should be semantically meaningful. In any real setting, classes are a description of the data points. Any random selection of data points is not describable as a group, while a meaningful selection can be a combination of existing attributes: "female patient of over 45", "low-alcohol wine (<=10%) with high color intensity (>=7)", "married and self-employed" are all examples of meaningful partitions of a dataset.

Related work on similar problems is mentioned in 2.5, with the optimal [12] and most straightforward method resulting from previous experiments being a concatenation of $k$-means clustering and decision tree construction.

The $k$-means clustering step finds $k$ clusters within the data, which is a partition of the data points with high intraclass correlation and low interclass correlation. Because this is a partition of data points, it is not semantically meaningful or describable. To achieve this, we use the resulting cluster labels as input labels for the construction of a decision tree 5.2. The resulting fitted decision tree can then predict

these labels as accurately as possible. The results from this decision tree are the final labels which we will use. The use of this decision tree is that the final partition is describable, as with any decision tree. The nodes in the tree all consist of rules, and so the resulting partition (if $k$ is not too large) is easily describable.

LISTING 5.2: Construction of the pseudo-classes.

```
def PseudoClassConstruction(data):
    clusters = kMeans(data)
    labels = DecisionTree(data, clusters)
    return labels
```

For illustrational purposes, figure 5.1 shows the construction of these pseudo-classes on the selection features and labels of the *Breast Cancer* [49] and *Iris* [21] as mentioned in 7.2. The labels for the Iris dataset are converted to binary labels by combining the two smallest classes into one class. The number of classification features of the Breast Cancer dataset are reduced to 2 using Principal Component Analysis, and not actual features (thus the convoluted results). The maximum depth for the decision tree construction is set to 4.

## 5.2 Bootstrap aggregating

To improve performance, and to combat the randomness of $k$-means clustering, we can repeat the pseudo-class construction step multiple times. To do this, we use bagging and our bootstrap sample is then repeatedly used for the new pseudo-classes.

When selecting the pseudo-class to sample from, we aggregate these results. This is done with a voting process, where each of the $n$ partitions adds a vote to each instance within the selected pseudo-class. The instance with the most votes is then sampled. The updated approach is seen in 5.3.

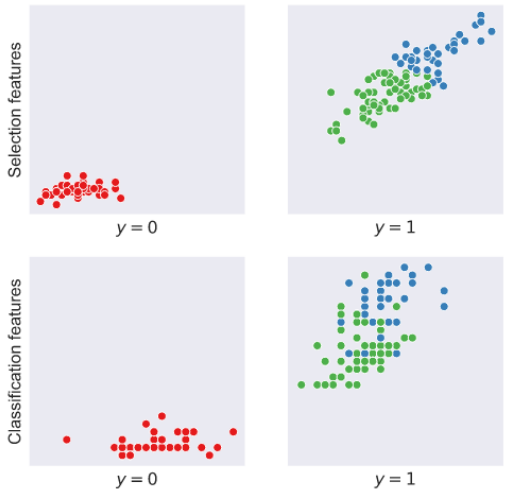
LISTING 5.3: Active Pseudo-Class Selection with bagging

```
def ActivePseudoClassSelection(x, z, Y):
    for i in (0..n):
        pseudoclasses[i] = PseudoClassConstruction(x + Y)
    for i in (0..n):
        pclass = ActiveClassSelection(pseudoclasses[i])
        add one vote to each instance in pclass
    instance = select the instance with the most votes
    return instance
```

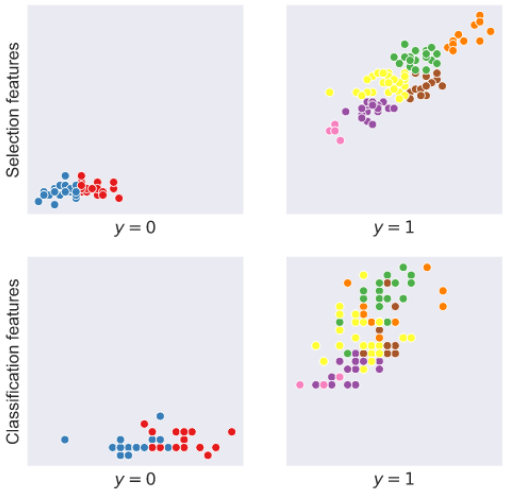
Using this voting process aims to improve results, in a similar manner to *Random Forest* models [10]. This method allows finding the most interesting instances, which are selected often regardless of the specific partioning. When not using bootstrap aggregating, one pseudo-class is selected and one instance needs to be sampled from this pseudo-class (randomly or heuristically). Using bootstrap aggregating allows a better separation of specific instances.

## 5.3 Active Class Selection method

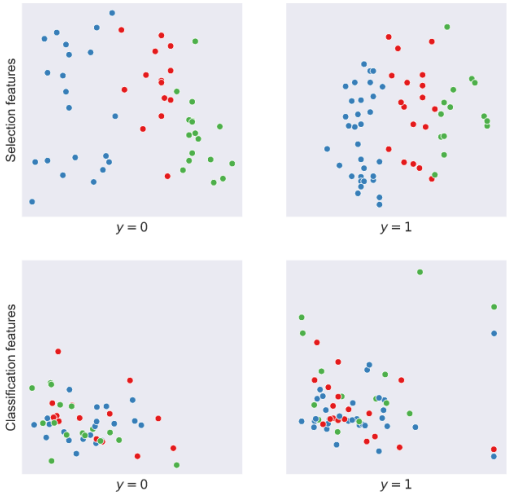
For this method, as we map the problem to an Active Class Selection problem, we then use an existing approach to solve this problem. Considering previous research on this subject [40, 69, 33], we will consider the following three methods: *redistricting*, *accuracy improvement*, [40] and *PAL-ACS* [33]. These methods showed good



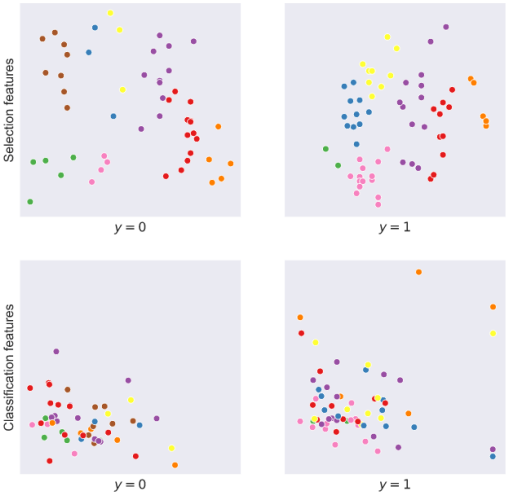

(A) Constructed pseudo-classes with $k = 3$ on the Iris dataset.




(B) Constructed pseudo-classes with $k = 8$ on the Iris dataset.




(C) Constructed pseudo-classes with $k = 3$ on the Breast Cancer dataset.




(D) Constructed pseudo-classes with $k = 8$ on the Breast Cancer dataset.

FIGURE 5.1: Examples of constructed pseudo-classes.

and promising results on regular Active Class Selection problems, so if this problem mapping is representative and informative we should expect these methods to perform well.

### Redistricting

The redistricting algorithm is described in 3.2, and pseudocode is given in 5.4. It focuses on selecting instances from the most volatile class labels, as the instances that are close to a decision boundary are most interesting to sample. In a way, this follows a similar philosophy to the uncertainty sampling method used in regular label-based active learning.

LISTING 5.4: Redistricting

```
def Redistricting(pseudoclasses):
    foreach instance:
        pclass = pseudoclasses[instance]
        count[pclass]++
        if prediction changed for instance:
            redistricted[pclass]++
    return argmax(redistricted / count)
```

### Accuracy Improvement

The accuracy improvement method is referenced and described in section 3.2. Pseudocode for this method is given in 5.5. The method aims to find the class or pseudo-class for which the accuracy has improved the most from the last iteration. This pseudo-class has potential to improve its score, and thus there is still information obtainable.

LISTING 5.5: Accuracy Improvement

```
def AccuracyImprovement(pseudoclasses):
    foreach instance:
        pclass = pseudoclasses[instance]
        count[pclass]++
        if prediction is correct for instance:
            correct[pclass]++
    for each pclass:
        improvement[pclass] = correct / count − previous_accuracy
        previous_accuracy[pclass] = correct / count
    return argmax(improvement)
```

### Probabilistic Active Learning for Active Class Selection (PAL-ACS)

The PAL-ACS method is referenced and explained in section 3.2.1, and pseudocode for the implementation is given in 5.6. The method aims to estimate a distribution for each pseudo-class. It then samples $n_p$ pseudo-instances from these distributions and estimates the best pseudo-class to select.

LISTING 5.6: PAL-ACS

```
def PAL–ACS(pseudoclasses):
    n_p = 25, M = 3, L = []
```

```
foreach pclass in pseudoclasses:
    density = estimate_density(pclass)
    sample n_p p-instances from density
    foreach p-instance:
        estimate_p-gain(p-instance)
    avg_p-gain = average(p-gain) for all p-instances
return argmax(avg_p-gain)
```

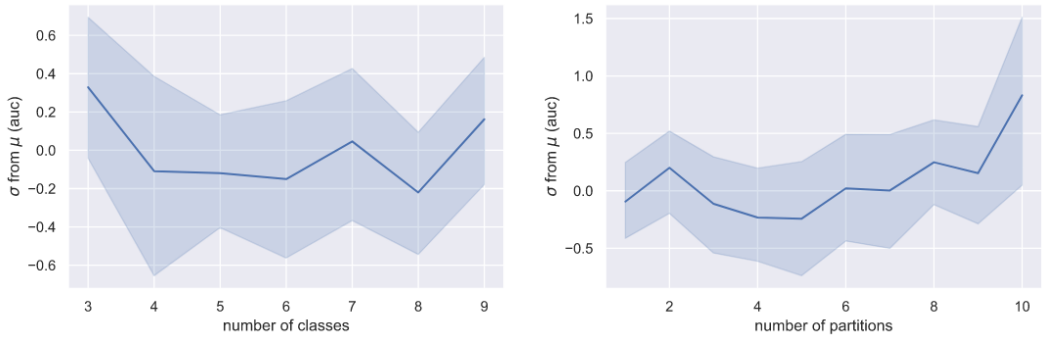
## 5.4 Hyperparameters

For this approach, several parameters are required as shown in table 5.1.

TABLE 5.1: The parameters required for the Active Pseudo-Class Selection approach.

| Parameter | Usage |
|---|---|
| $k$ | The number of pseudo-classes within each partition. |
| $n$ | The number of partitions. |
| `ACS` | The Active Class Selection method used. |

The first two parameters—$k$ and $n$—are numerical and tuning can show reasonable choices. $k$ must be at least 3, otherwise the problem is mapped from a binary class problem to a binary class problem. $n$ must be at least 1, but should be reasonably higher. It is expected that increasing $n$ should increase the performance, or at least never decrease. With an ensemble predictor, one does not expect the performance to decrease when more of the same predictors are added. Figure 5.2 confirms this fact, combining preliminary experiments with the redistricting method and comparing their results to the mean.




(A) The results by number of classes.

(B) The results by number of partitions.

FIGURE 5.2: Results from parameter tuning experiments for redistricting, combining all classifier models and datasets.

Broadly looking at the results, it seems that the number of pseudo-classes does not have a large influence on the results. Instead, the number of partitions indeed seems to increase performance when increased. This makes sense, as for example random forests also perform better with a larger number of trees. Perhaps even more partitions make sense, if the computational power allows it.

The last parameter—`ACS`—is the selection of the method used for the resulting Active Class Selection problem, and is discussed in the previous section.

# Chapter 6

# Expectation Maximization

In this chapter, the third and final approach is considered. This approach is based on the known Expectation Maximization method, used for creating Gaussian Mixture Models 2.6.

Using Expectation Maximization, *soft* clusters are created using the complete data. That is: the features of all instances with known classification features combined. These clusters are then used as a way of modeling the true distribution of the data. For all incomplete data—the instances of which the classification features are not yet known and which we aim to sample—the probabilities of belonging to each soft clusters are then estimated. With this, we can use the expected utility of the clusters to determine the most useful instance or instances.

The method is defined in 6.1.

LISTING 6.1: EM-based ACFS

```
def EM_based_sampler(x, z, Y):
    gmm = ExpectationMaximization(x+z)
    for each cluster c in gmm:
        utility[c] = estimateClusterUtility(c)
    for each instance:
        estimated_utility =
            ∑_{c∈gmm} utility[c] · gmm.prob(instance)[c]
    return instance with the highest estimated_utility
```

The method of Expectation Maximization and the estimation of probabilities of instances is well known, and does not need redefining. We can simply use the established methods here. Each clusters is then defined with its mean $\mu$ and its covariances. An example of a Gaussian mixture model on a 2D plane can be seen in figure 6.1.

## 6.1 Cluster Utility Estimation

An important element of this approach is the definition of the estimation of the cluster utility. We can use the approaches of the estimation of utility in Chapter 4, but instead focus on an expected return for a distribution. The EM-based approach is potentially more efficient than the AFA-based approach. This is because for the AFA-based approach, each instance must estimate its classification features and then determine its utility. On the other hand, the EM-based approach does not estimate any classification features, and estimating the utility is only done for each *cluster*, not for each *instance*. If estimating the utility of a cluster is then less expensive than the combined estimation of the utility of all instances, it is computationally less expensive.

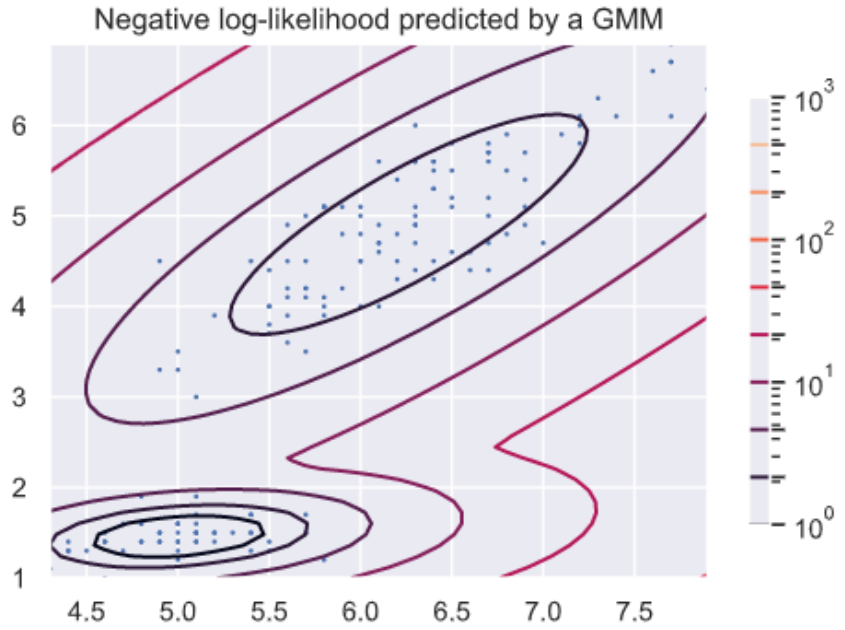



FIGURE 6.1: Gaussian Mixture Model fit on two attributes of the Iris dataset [21].

The utility for each instance given this method can be defined as follows:

$$\mathbb{E}\,U(i) = \sum_{c \in C} P(c|i) \cdot \mathbb{E}\,U(c) \tag{6.1}$$

The cluster utility can be defined as follows:

$$\mathbb{E}\,U(c) = \int_{-\infty}^{\infty} U(x)dx \tag{6.2}$$

where:

$U(x)$ is the defined utility method.

$U(x)$ unfortunately is not easily definable, as it depends on retraining of the classifier model or imputation models. As such, a numerical or algebraic way of defining the expected utility is not feasible as we can see. The best option is a stochastic Monte Carlo method of estimating the true mean.

Consider the function $U$ as an unknown or black box function. The distribution is known with $\mu$ and $\sigma$. We can then estimate the expected value of $U(x)$ by sampling instances from the distribution. For this instance, we 'query' the black box function to return us an output. This is repeated $n$ times, until the expected value converges. For obvious reasons, the higher value for $n$, the better, result wise. For computational reasons though, a limit for $n$ must be chosen. This depends on the capabilities of the machine. Keeping true that $n < |c|$ ensures that this method is computationally more efficient than the AFA-based method.

This method is more efficient than the alternative method of proportional seleciton. This would mean that the instances are split *fairly* along the distribution, with each instance representing $1/n$ of the cumulative distribution. When the dimensionality of the features is high, this causes a great increase in computational complexity. Whereas the stochastic process just samples many points and the law

of large numbers implicate some convergence, the manual selection is more complicated with many features. If there are eight features, splitting each up into four sections requires $4^8 = 65,536$ samples. Stochastic sampling should converge by itself without requiring a complicated instance generation and selection.

# Part III

# Experiments and Results

# Chapter 7

# Methodology

In this chapter, we will give an overview of the methodology used for comparing, executing and evaluating multiple approaches to the problem.

## 7.1 Experimental Setup

The experiments consist of three parts: setup, execution, and evaluation. In the setup step, we prepare the given dataset $\mathcal{D}$, concealing a subset of features to create known x and unknown z. Additionally, we select all parameters, such as the classifier model, the approach and the approach-specific hyperparameters.

In the execution step, we try to simulate a real-world scenario by allowing the algorithm to sample instances of z. It can then update its beliefs and sample another value.

Finally, the evaluation step then evaluates the performance of each approach in the experiments. For example, by plotting a comparison of a performance measure to the number of sampled instances. The same should be done for a baseline to compare to, such as random sampling. Because this exact problem is not found yet in literature, there are no other approaches that we can compare to.

It should be noted that evaluating any Active Learning approach is no simple task and we should aim to optimize this. Pointers are provided in [31] as to how to approach this problem, which we will take into account. The three critical aspects of the evaluation are *reliable evaluation*, *realistic evaluation*, and *comparable evaluation*.

It concludes that the following guidelines must be implemented to attain these three aspects:

- Use at least 50 repetitions
- Use the same selector and consumer classifier when comparing different active learning methods
- Show the learning curves of performance measures, and show pairwise differences
- Start with an unlabeled set, or sample $n$ instances randomly
- Use a clearly defined stopping criterion

### 7.1.1 Dataset Setup

An important issue regarding realistic evaluation (among others) is that most experimental setups do not consider a real-world scenario, but create one using existing or synthetic datasets meant for another use. For our problem, there is one dataset specific to this experiment, which we will use.

Initializing the dataset, all classification features z must be hidden. Some approaches do not work well with an initially empty set, but the initial sample of $n$ features must be included within the performance of the algorithm and shown within the learning curve [31].

The features within the dataset should be normalized.

### 7.1.2 Algorithm Execution

To create a more reliable evaluation, it is a good idea to perform multiple runs (preferably, 50 or more [31]) for each approach. These multiple runs can then in the evaluation section be pooled, plotting means and standard deviations or quartiles.

These runs are then evaluated with k-fold cross validation during the execution. After the sampling of each instance, the evaluation is saved so that it can be used within the evaluation step.

As for the final classifier trained on the classification features and the labels, it must be the exact same classifier for each approach to facilitate comparable evaluation. In addition, the same classifier must be used for the selection method as the final classifier. In our case, we are using *logistic regression* [6] and *support vector machine* (SVM) [16]. The relevant parameters must all be chosen reasonable and kept consistent among the different approaches. Changing this either induces bias or an unfair comparison between algorithms. These default values for the parameters ([50]) are shown in table 7.1.

TABLE 7.1: The hyperparameters used for both classifier models.

| Classifier | Parameters |
|---|---|
| Logistic Regression | $C = 1.0, L2$ |
| Support Vector Machine (RBF) | $C = 1.0, \gamma = \frac{1}{\texttt{n\_features} \cdot \texttt{var(X)}}$ |

No stopping condition is implemented, and the experiment will run until all instances are selected. The learning curve can then be plotted in its entirety.

The setup and execution is implemented as follows:

1. Define $\mathcal{D}$.
2. Define x, z, $Y$ as subsets from $\mathcal{D}$.
3. Repeat the following $n = 10$ times:
   (a) Generate a $k = 5$-fold split of the data.
   (b) For each fold, perform the following:
      i. Define $\mathcal{D}_{train}$, with $\mathrm{x}_{train}$ and $Y_{train}$.
      ii. Define the oracle with $\mathcal{D}_{train}$ and $\mathrm{z}_{train}$.
      iii. For each possible combination of classifier model and sampling method, define the classifier model $\Gamma$, referencing $\mathcal{D}_{train}$, and define the sampling method, referencing $\mathcal{D}_{train}$ and $\Gamma$.
      iv. For each combination, repeat until every instance is sampled:
         A. Use the defined sampling method to sample the next instance.
         B. Query the classification features of this instance.
         C. Retrain $\Gamma$ with the new instance added to $\mathcal{D}_{train}$.

D. Evaluate the model, using the defined metric (F1-score for the benchmark datasets, accuracy for the schizophrenia dataset).

(c) Average the evaluation at each step (number of instances sampled).

4. Save the results, which is a DataFrame of shape ($n$, `number of instances sampled`).

### 7.1.3 Evaluation

The multiple algorithms need to be evaluated in such a way, that they can be compared properly to each other. To evaluate the performance of the algorithm, it needs to be quantified. This is done with a *performance measure*. A common example is accuracy, which maps the performance to (in most cases) the range of 0 to 1 based on the proportion of correctly classified instances.

For active learning problems, not only is the quality of the classifier important, we also want to learn as economically as possible. One option to approach this problem is by plotting a standard performance measure (such as accuracy). This performance measure is than plotted over time (the number of instances sampled), which is known as the *learning curve*.

As we will be running each experimental configuration 50 times, we will want to represent these many runs appropriately. The learning curve will plot the mean of these 50 runs, as this is the expected value for each point of the learning curve. To aid in showing the distribution, we will add a *confidence interval*. This confidence interval will display the range in which the majority of the runs are contained. We determine $\alpha = 0.10$, meaning that the likelihood of any run being contained in this confidence interval is $1 - \alpha = 0.90$.

Determining the performance measure for a binary classifier is described in [48], where multiple measures are compared with each other. Performance measures which often disagree with the other measures, are considered to be inconsistent. The results show the *h-measure*, *Matthews Correlation Coefficient* (MCC), and *F1-score* to give the best indications. The h-measure uses relative costs of misclassification, which might not be available for all datasets; it is also not as commonly used as the other two performance measures. The latter is true as well for the MCC/$\phi$-coefficient. As such, we will use the F1-score for the evaluation and the learning curve: the F1-score is a combination of precision ($P$) and recall ($R$), and is defined as $\frac{2 \cdot P \cdot R}{P+R}$.

Learning curves are often not strictly better or worse than the other. For example, one algorithm might perform better in the earlier stages, while another algorithm might perform better in the later stages. Comparing learning curves requires contextual information. Depending on the use case, the earlier or latter stages of the learning curve might be more important.

Additionally, we might use performance measures specifically developed for active learning. These measures aim to summarize the shape of the learning curve. Although we can never fully represent the learning curve and the advantages and disadvantages of each approach within one number, we can use several methods to allow for a more clear and statistically significant comparison.

The main measure we will use is *Data Utilization Rate (DUR)*. DUR uses a target accuracy, defined as the mean of the random selection strategy. The DUR is the minimum number of samples needed to reach this accuracy, divided by the number of samples needed to reach this accuracy with random selection.

To get a better overview of the results, we will use an additional number of measures that represent what we aim to achieve. Ultimately, the initial few samples are not relevant, as we will never reduce the training set size ten-fold. Instead, we aim to obtain a method where we can obtain a performance close to the complete one, with a significant decrease in training set size.

To represent this, we will obtain the performance measure at two points in the curve: 0.8 and 0.9 of the total number of instances. In addition, we measure the area under the curve.

The numerical measures we will use for evaluation in addition to the learning curve are noted in table 7.2.

TABLE 7.2: The numerical summaries of the learning curve.

| Measure | Explanation |
|---|---|
| DUR | Data Utilization Rate. Represents how many instances are needed to reach a target F1 score, derived from the random selection strategy. |
| F1@90% | The F1-score after having sampled 0.9 of the total number of instances. |
| F1@80% | The F1-score after having sampled 0.8 of the total number of instances. |
| AUC | The area under the learning curve. |

Now that we have numerical results, we can compare two algorithms using pairwise difference, using the Wilcoxon signed rank test [67], or the Friedman test [23].

## 7.2 Datasets Used

As the problem directly ties to a real-world scenario and dataset, this dataset [46] is the most relevant. It is analyzed in chapter 12. The aim is to achieve good results with this dataset, reducing the number of MRI scans needed for similar results.

### 7.2.1 Real-world datasets

For benchmark results, some datasets from the UCI Machine Learning Repository [20] are used. The datasets should be binary classification datasets, with a relatively large number of numerical features. For these datasets, there are no classification features and they have to be selected. These features can be chosen randomly or hand-picked. Realistically, the selection features are simpler features and the classification features are more expensive to obtain. Therefore, the feature split is hand-picked and can be seen in table 7.3 for the used datasets.

TABLE 7.3: The datasets used and their feature splits.

| $\mathcal{D}$ | x | z |
|---|---|---|
| Breast Cancer (Coimbra) [49] | Age | Glucose |
| | BMI | Insulin |
| | | HOMA |
| | | Leptin |
| | | Adiponectin |
| | | Resistin |
| | | MCP-1 |
| Heart Disease [18] | Age | Resting Blood Pressure |
| | Sex | Cholesterol |
| | Chest Pain Type | Fasting Blood Sugar |
| | | Resting ECG |
| | | Max Heart Rate |
| | | Exercise Induced Angina |
| | | Exercise Induced ST Depression |
| | | Slope of Peak ST |
| | | Number of Major Vessels |
| | | Thalassemia |
| Wine [22] | Alcohol | Malic acid |
| | Color intensity | Ash |
| | Hue | Alcalinity of ash |
| | | Magnesium |
| | | Total phenols |
| | | Flavanoids |
| | | Nonflavanoid phenols |
| | | Proanthocyanins |
| | | OD280/OD315 of diluted wines |
| | | Proline |

### 7.2.2 Synthetic datasets

In addition to these real datasets, we will use synthetic datasets for proof of concept. These datasets are generated from existing distribution, with an expected noise: $y = f(x) + \epsilon$.

$N(\mu, \sigma)$ represents the normal distribution. The reasoning behind the generation of the datasets is to determine what correlations are necessary for approaches to function—or for this problem to be solvable.

The synthetic datasets can more generally be defined, by defining each instance as in equations 7.1, 7.2, 7.3. Constructing a dataset of size $n$ is done by concatenating $n$ of these generated instances.

$$\mathrm{x} = \mathrm{x}_0, \mathrm{x}_1, .., \mathrm{x}_{k-1} = N(\mu, \sigma_\mathrm{x}), N(\mu, \sigma_\mathrm{x}), .., N(\mu, \sigma_\mathrm{x}) \tag{7.1}$$

$$\mathrm{z} = \mathrm{z}_0, \mathrm{z}_1, .., \mathrm{z}_{k-1} = N(f(\mathrm{x}_0), \sigma_\mathrm{z}), N(f(\mathrm{x}_1), \sigma_\mathrm{z}), .., N(f(\mathrm{x}_{k-1}), \sigma_\mathrm{z}) \tag{7.2}$$

$$y = \begin{cases} 1 & \text{if } \sum_{\mathrm{x}_i \in \mathrm{x}} g(\mathrm{x}_i) + \sum_{\mathrm{z}_i \in \mathrm{z}} h(\mathrm{z}_i) + N(0, \sigma_Y) > c \\ 0 & \text{if } \sum_{\mathrm{x}_i \in \mathrm{x}} g(\mathrm{x}_i) + \sum_{\mathrm{z}_i \in \mathrm{z}} h(\mathrm{z}_i) + N(0, \sigma_Y) \leq c \end{cases} \tag{7.3}$$

where:

$f(x), g(x), h(x)$ are definable functions,

$\sigma_{\mathrm{x}}, \sigma_{\mathrm{z}}, \sigma_{\mathrm{Y}}, c$ are definable constants.

The synthetic datasets can then be defined as seen in table 7.4. This also allows for more complexity and more realistic datasets, by defining $f$, $g$ or $h$ as non-linear functions, for example.

TABLE 7.4: The synthetic datasets.

| $\mathcal{D}$ | $f(x)$ | $g(x)$ | $h(x)$ | $\sigma_{\mathrm{x}}$ | $\sigma_{\mathrm{z}}$ | $\sigma_{\mathrm{Y}}$ | $c$ |
|---|---|---|---|---|---|---|---|
| Direct correlation | $\sin(x)$ | 0 | $\sin(x)$ | 1 | 1 | 0.2 | 0 |
| Indirect correlation | 0 | $\sin(x)$ | $\sin(x)$ | 1 | 1 | 0.2 | 0 |
| No correlation | 0 | 0 | $\sin(x)$ | 1 | 1 | 0.2 | 0 |

The first synthetic dataset represents the datasets with correlations between the selection features and the classification features, which is the case where the approaches are expected to perform best. The second synthetic dataset shows an indirect correlation via the label, which is influenced by both the selection features and the classification features. Finally, the third synthetic dataset shows a case where the selection features are irrelevant. This allows us to consider what correlations are necessary for our methods to function.

## 7.3 Baseline

To evaluate and compare the results of the approaches, we need a baseline method to compare to. [13] notes a set of potential baseline methods, but these are mostly not applicable to our problem set as our setup is significantly different. For example, the uncertainty sampling method is not usable for our problem as there is no input available and no unknown binary output.

As such, we will use *Random sampling* as the baseline, as it is essentially not using any specific approach. We would expect to improve upon this baseline, to motivate using any potential approach. Thus we will include this as our baseline.

This is implemented by creating a query space (of all data points), and randomly selecting one.

# Chapter 8

# Active Classification Feature Acquisition

In this chapter, we will show an overview of the results for the Active Classification Acquisition approach and each of the utility methods, comparing them to each to other as well as the random baseline (where each instance is randomly selected).

For each method, we need an initial sample of instances. These instances are then randomly selected from each label alternately. We aim to minimize this number of initial instances so that the method is used as much as possible, this minimum number of instances rqeuired is 4. This allows the leave-one-out cross-validation methods as needed for a classifier to fit, with a minimum of 2 labels for each class (a class with 1 label cannot be trained when this singular label is the separate fold).

For the estimation of the features, we use a relatively simple and reliable method: linear regression. This allows us to properly compare the methods.

The results are averaged over 50 runs as mentioned in chapter 7, using the F1-score.

## 8.1 GODA

The results for the GODA utility method are shown in figure 8.1. The results show an obvious, and significant decrease in the performance for a major portion of the learning curve. Up until to the very last few samples, the F1-score for the GODA utility method is significantly lower than the Random sampling baseline.

One thing that should be noted is the slight peak at the very end of the learning curve, as this seems to indicate that when given more instances the performance of the GODA utility method might increase. However, there is not enough information to indicate this. Although the datasets have a differing number of instances, the peak in the learning curve happens near the end for all of them. This implies the peak in the learning curve coming from the *data pool* setting, as only a very small possible set of samples is left near the end of the experiment. When nearing the end, the set of samples is near-identical for each method, and the difference in performance is not significant for most of the dataset/classifier combinations.

The results for the synthetic datasets are shown in figure 8.2. These synthetic datasets allows us some more information about the effectiveness of the method, as the correlations between features are clear. These results confirm the results for the UCI datasets, as even with a direct correlation these results are worse than random sampling. It can be noted that although these results are subpar, they are *less worse* for the direct correlation synthetic dataset than the absent correlation (with the Logistic Regression classifier model). This means that there is some benefit in the analysis, but it is not practical and does not compensate for the huge error.

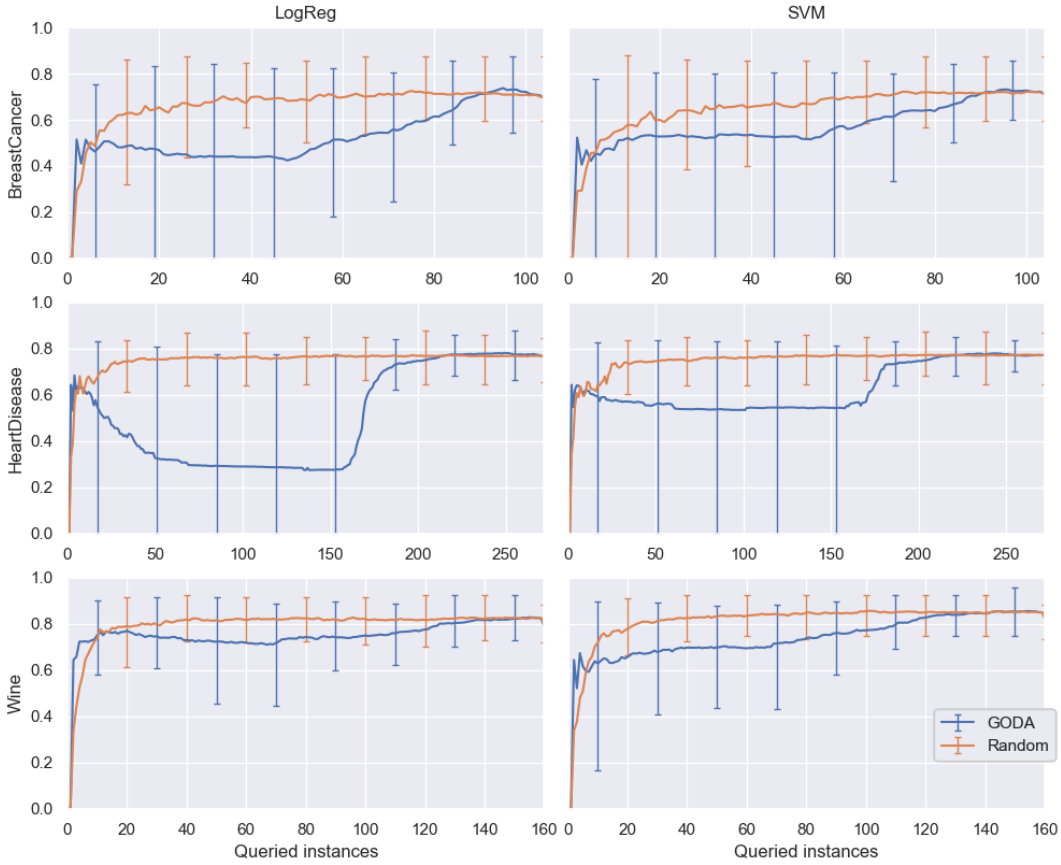



FIGURE 8.1: Comparison of the F1-score of the GODA utility method and the random baseline, on the UCI datasets.

From these results, we can generally conclude that the GODA utility method is ineffective, and recommend to not use this utility method for the Active Selection of Classification Features problem setting.

## 8.2 AVID

The results for the AVID utility method are shown in figure 8.3. In this case, $B$ is set to 10. The results are differing, depending on the dataset (but perhaps surprisingly, indepedent of the classifier model). For the Heart Disease dataset, we can clearly say that the AVID utility method gives us an approach that performs strictly better than the random sampling baseline. At all points in the learning curve, the performance exceeds or is at least as good as the random sampling performance. For the Breast Cancer and Wine datasets, there is a trade off in the learning curve. Interestingly enough, this is the opposite for the two datasets. For the Breast Cancer dataset, there is a significant decrease in the early phases of the experiment for an increase in the middle stages. For the Wine dataset, the opposite is true: the performance increases in the early phases and decreases in the later stages.

It should be noted that for the Wine dataset, a very small set of instances found with the AVID utility method allows us a higher performance on the test set than the complete dataset.

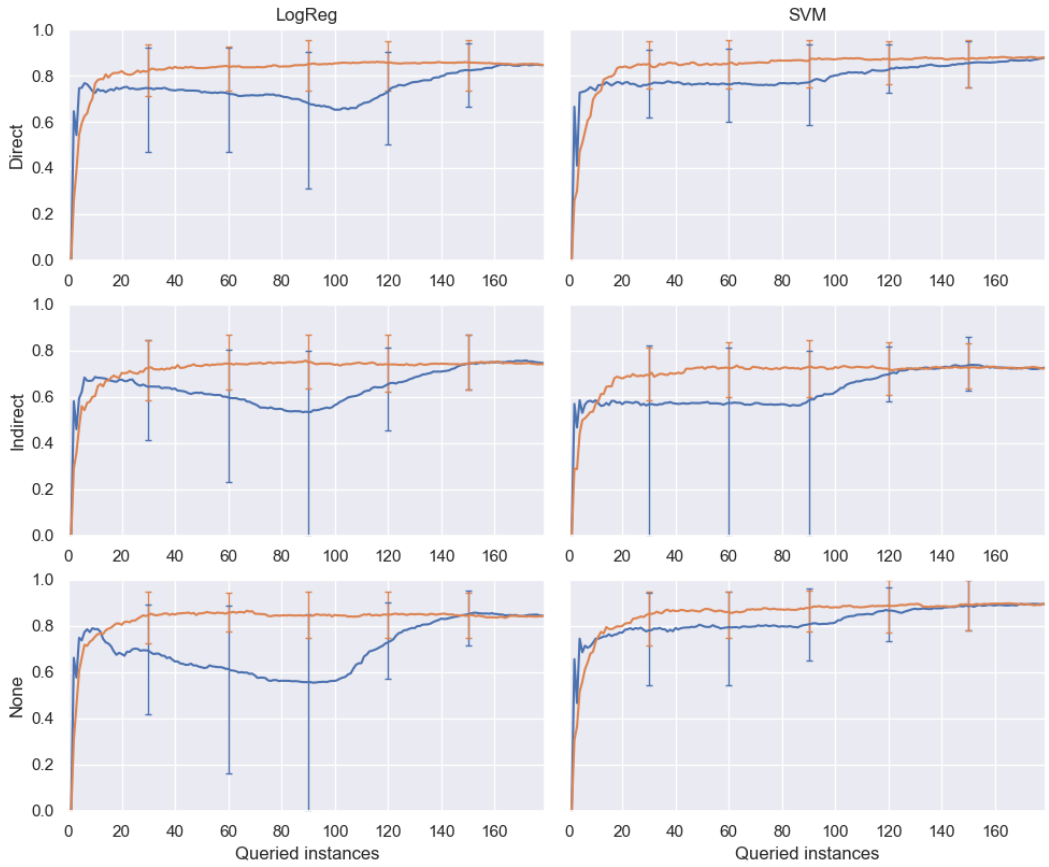



FIGURE 8.2: Comparison of the F1-score of the GODA utility method and the random baseline, on the synthetic datasets.

The results for the synthetic datasets are shown in figure 8.4. These results are quite consistent, regardless of the dataset correlations. This means that it is somewhat reliable in its results and does not provide subpar results for inappropriate datasets.

Although we can not say that using the AVID utility method with the ACFA approach is always effective or a strict improvement, it seems that in general it does provide good results more often than not and we can recommend using this method.

## 8.3 Dual objective

The results for the Dual objective utility method are shown in figure 8.5. As suggested in [71], $\lambda$ is set to 0.5.

There is not much to say about these results, considering this method is a combination of the two previous already discussed methods. As the GODA utility method was shown to be ineffective, and the AVID utility method to provide promising results, one would expect the combination of these two to have mediocre results: this is mostly true. The results are simply worse than AVID's results, while at least being better than GODA's results. This method is not recommended to use in place of the AVID utility method.

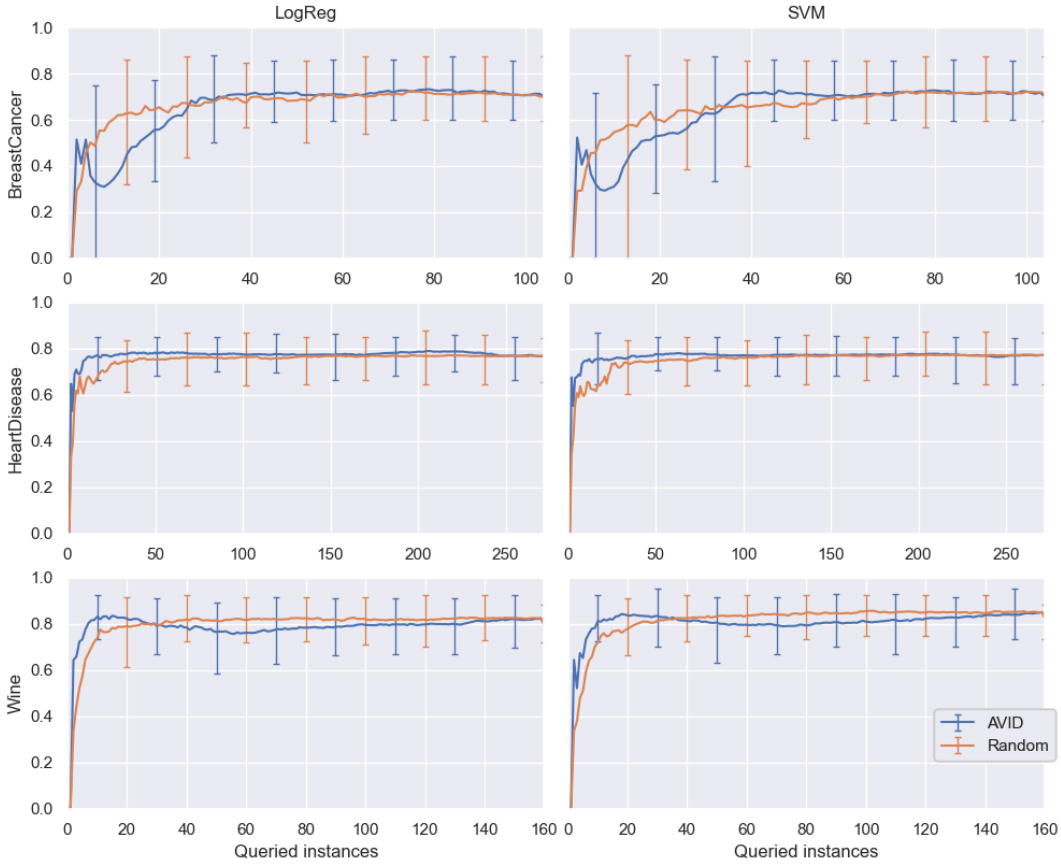



FIGURE 8.3: Comparison of the F1-score of the AVID utility method and the random baseline, on the UCI datasets.

## 8.4 Probability-based

The results for the Probability-based utility method are shown in figure 8.6. As this method only works with probability-based classifiers, only the Logistic Regression model is used.

For comparison, we considered two options for $b$: the score function as noted previously, and $b = 0$.

As seen in figure 8.6, $b = 0$ does not work well in practice. This matches with the previous intuitions that selecting a method which is already expected to perform well does not add much new information, but instead introduces a bias to non-informative instances and away from instances that are close to the decision boundary.

On the other hand, the automatic scoring function provides acceptable results, albeit not exceptional. For the Wine dataset, the results are strictly better than the random sampling baseline, while on the other hand for the Breast Cancer and Heart Disease one might be more inclined to prefer the results from the random sampling baseline. As with previous utility methods, the results are dataset-dependent and this is unfortunately unknown beforehand.

The results with the automatic scoring function, on the synthetic datasets are shown in figure 8.7.

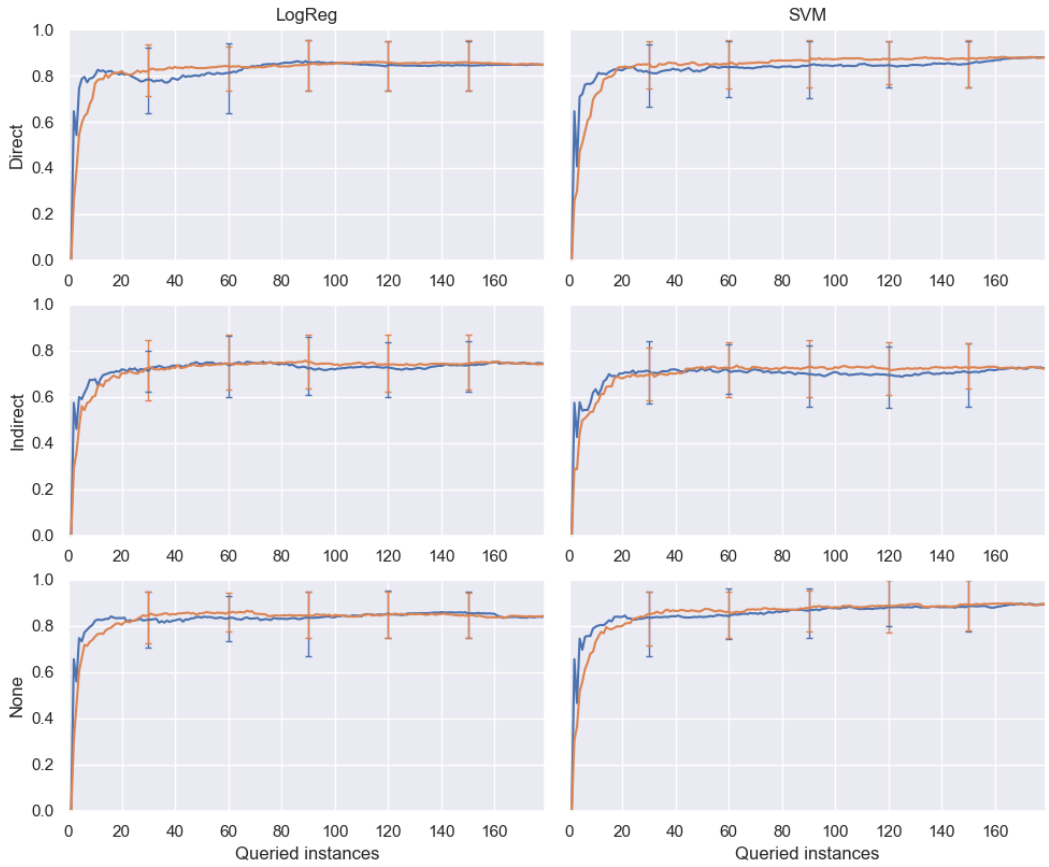



FIGURE 8.4: Comparison of the F1-score of the AVID utility method and the random baseline, on the synthetic datasets.

## 8.5 Comparing the utility methods

To be able to compare the utility methods, we can use the Data Utilization Rate to represent each averaged run as one value properly. For each experiment, we determine the mean of the random selection strategy as the target score. The number of instances queried to reach this score, divided by the number needed by the random selection strategy, is then set as the Data Utilization Rate value.

TABLE 8.1: The Data Utilization Rate results for all ACFA methods, using the F1-score.

| | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|
| | BreastCancer | HeartDisease | Wine | BreastCancer | HeartDisease | Wine |
| GODA | 3.44 | 5.26 | 5.21 | 2.70 | 4.85 | 4.60 |
| AVID | 1.08 | **0.26** | **0.29** | 1.13 | **0.24** | **0.40** |
| Dual | 1.24 | **0.26** | **0.38** | 1.13 | **0.49** | **0.48** |
| Prob | 2.36 | **0.34** | **0.46** | | | |
| Random | 1 | 1 | 1 | 1 | 1 | 1 |

The results of the Data Utilization Rate are shown in table 8.1. These results show several trends. The GODA approach is for every classifier, and for every dataset, simply inferior to the other methods, as well as the random method. Intuitively, this would imply that the task of predicting the classification features from the selection

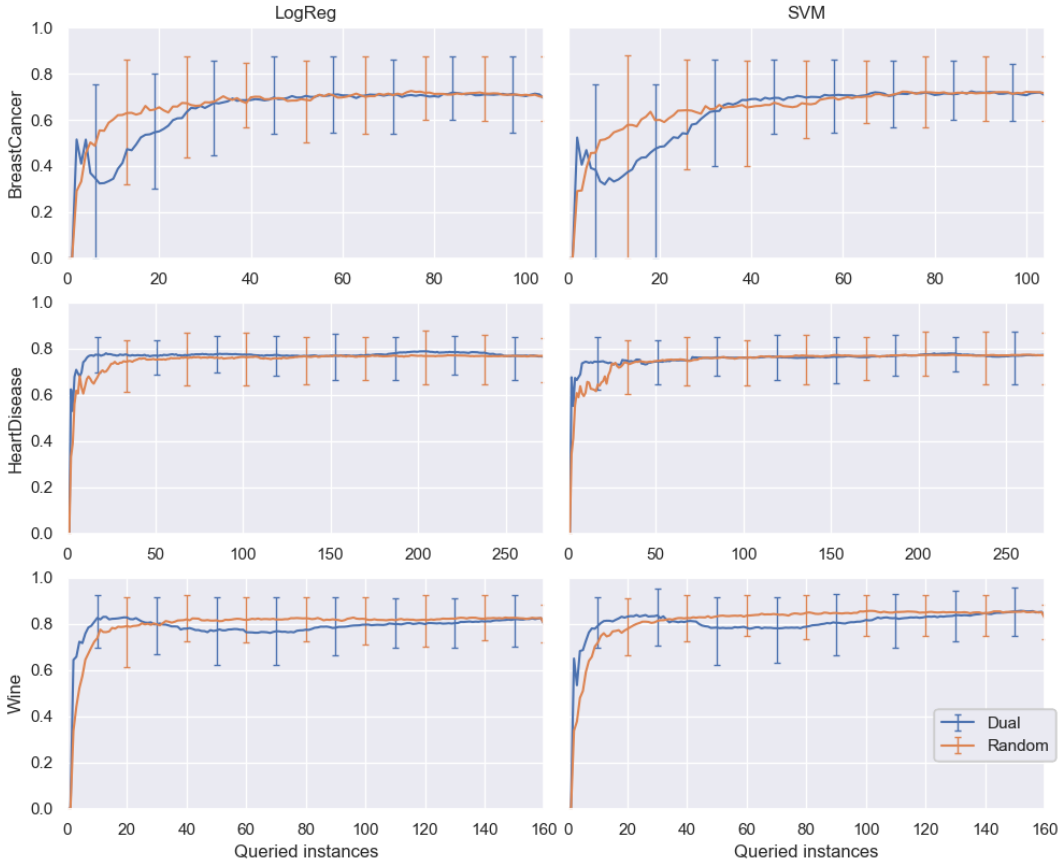



FIGURE 8.5: Comparison of the F1-score of the Dual objective utility method and the random baseline, on the UCI datasets.

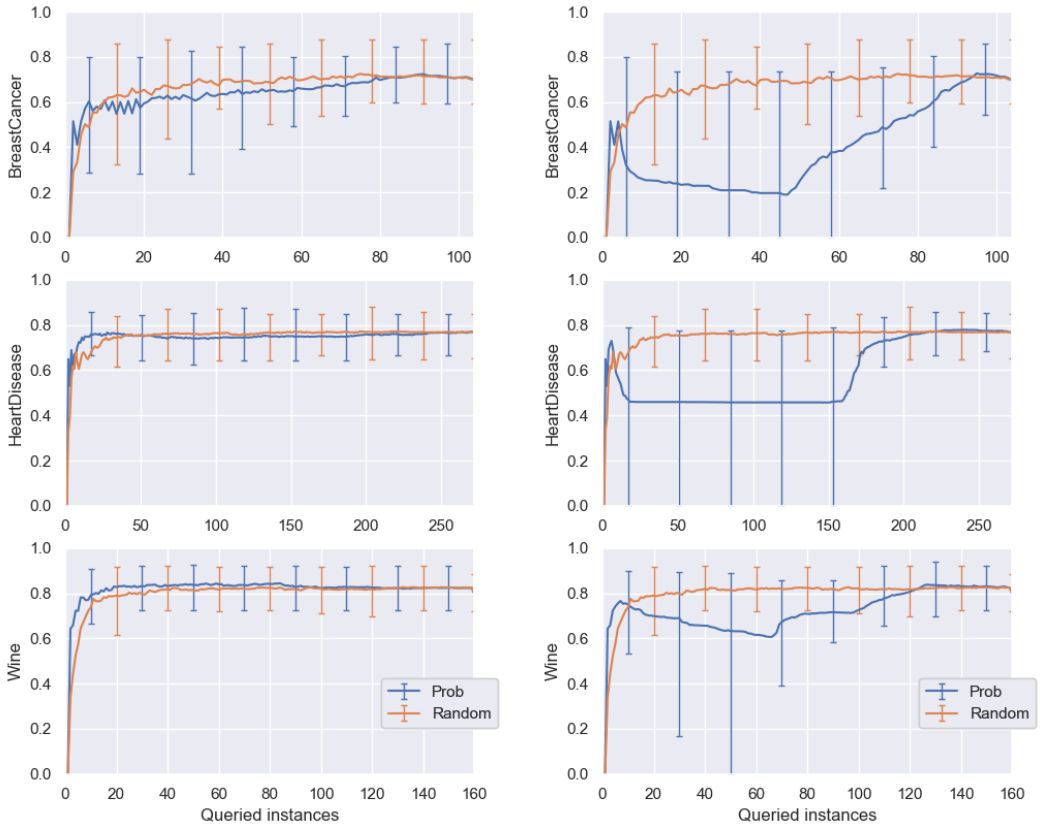
features plus using it for classification is computationally very complex, meaning it does not improve the results. Instead, it introduces a bias which means the results are worse than an unbiased random selection.

Additionally, it seems to imply that there is a difference in feasibility for each dataset. There is not a single method that is able to improve upon the random selection method for the Breast Cancer dataset. Opposed to this, the other three methods are able to improve upon the random selection strategy for each other dataset and classifier combination, implying some usefulness. The most likely implication is either a more complicated or non-existing correlation between the selection features and classification feature. It should be noted that this also depends on the use-case of the method, as an entire learning curve is hard to summarize with just one number. Looking at the learning curves for the Breast Cancer dataset, improvement can be made in the intermediary period at the cost of reduction in performance in the early period.

The best method seems to be AVID, based on the variance of the estimation of the classification features. This method bases its results on the instances about which essentially the least is known, and thus selects the most informative instances. It seems this method works well in practice.

To see if there is a statistically significant difference between any of these methods and the random selection strategy, we can use the Wilcoxon signed rank test. This statistical test can be used to compare two matched samples.

For each method, we compare it to the random baseline, where the null hypothesis is that that the random selection strategy and the selected method are from the




(A) $b$ is determined by the score function.

(B) $b$ is constant, set to 0.

FIGURE 8.6: Comparison of the F1-score of the Probability-based utility method and the random baseline, on the UCI datasets.

same distribution: implying no difference in results. If this hypothesis can be rejected, there is a significant difference and as such a expected better method. We use $p = 0.05$, giving a critical value $W \geq 0$.

TABLE 8.2: The results for the Wilcoxon signed rank test, where critical value for $W = 0$.

| Method | $W$ |
|---|---|
| GODA | 0 |
| AVID | **3** |
| Dual | **3** |

The results for each method are shown in table 8.2. The probability-based method is not included, as its sample size is too small. The AVID and Dual objective methods are shown to be an expected improvement on the random selection baseline, whereas the GODA method is not. This confirms the suspicions from the initial look at table 8.1.

To give a more extensive comparison, table 8.3 shows an overview of all numerical measures mentioned in chapter 7.

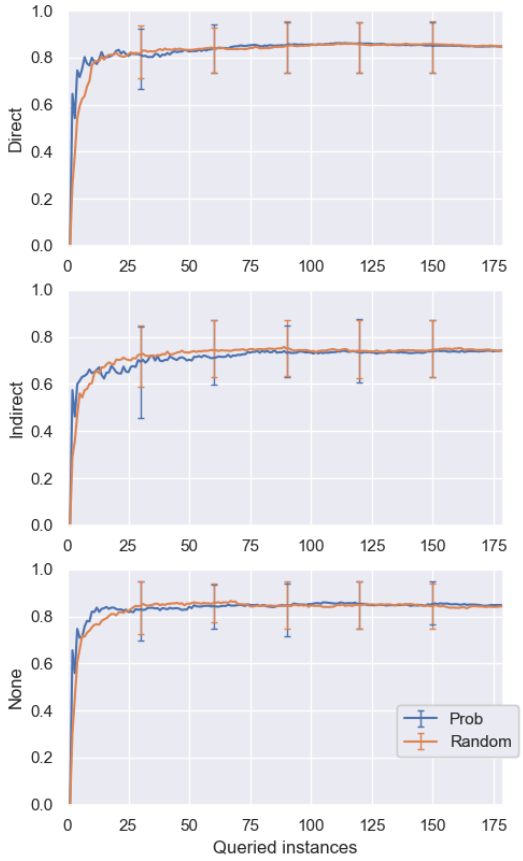



FIGURE 8.7: Comparison of the F1-score of the Probability-based utility method and the random baseline, on the synthetic datasets.

TABLE 8.3: All numerical results for all Active Classification Feature Acquisition approaches. Improvements upon random selections are bolded.

| | | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|---|
| | | Breast Cancer | Heart Disease | Wine | Breast Cancer | Heart Disease | Wine |
| GODA | DUR | 3.44 | 5.26 | 5.21 | 2.70 | 4.85 | 4.60 |
| | AUC | 0.52 | 0.50 | 0.75 | 0.57 | 0.63 | 0.74 |
| | F1@80% | 0.64 | 0.77 | 0.80 | 0.67 | 0.77 | 0.84 |
| | F1@90% | **0.73** | **0.78** | 0.82 | **0.73** | **0.78** | 0.85 |
| AVID | DUR | 1.08 | **0.26** | **0.29** | 1.13 | **0.24** | **0.40** |
| | AUC | 0.64 | **0.77** | 0.78 | 0.63 | **0.76** | 0.80 |
| | F1@80% | **0.73** | **0.79** | 0.80 | **0.72** | 0.77 | 0.83 |
| | F1@90% | 0.71 | 0.77 | 0.81 | 0.72 | 0.77 | 0.84 |
| Dual | DUR | 1.24 | **0.26** | **0.38** | 1.13 | **0.49** | **0.48** |
| | AUC | 0.63 | **0.77** | 0.78 | 0.62 | 0.75 | 0.80 |
| | F1@80% | **0.72** | **0.79** | 0.80 | 0.71 | **0.78** | 0.83 |
| | F1@90% | 0.71 | 0.77 | 0.82 | 0.71 | 0.77 | 0.85 |
| Prob | DUR | 2.36 | **0.34** | **0.46** | | | |
| | AUC | 0.64 | 0.74 | **0.81** | | | |
| | F1@80% | 0.71 | 0.76 | 0.82 | | | |
| | F1@90% | **0.72** | 0.77 | **0.83** | | | |
| Random | DUR | 1 | 1 | 1 | 1 | 1 | 1 |
| | AUC | 0.66 | 0.75 | 0.79 | 0.64 | 0.75 | 0.81 |
| | F1@80% | 0.71 | 0.77 | 0.82 | 0.71 | 0.77 | 0.85 |
| | F1@90% | 0.71 | 0.77 | 0.82 | 0.72 | 0.77 | 0.85 |

## 8.6 Regression method

For the previous experiments, we have considered linear regression as the baseline regression (or imputation) method. This provides a good baseline to compare the multiple methods. It might be possible to improve the results with a different regression mode though. In this section we will look at possible improvements by using Lasso [61] or Ridge [27] regression.

For comparing these regression methods, we use the AVID utility method. This method performs the best on the existing datasets, and directly uses the regression method, allowing a good overview of the changes. The results can be seen in figure 8.8.

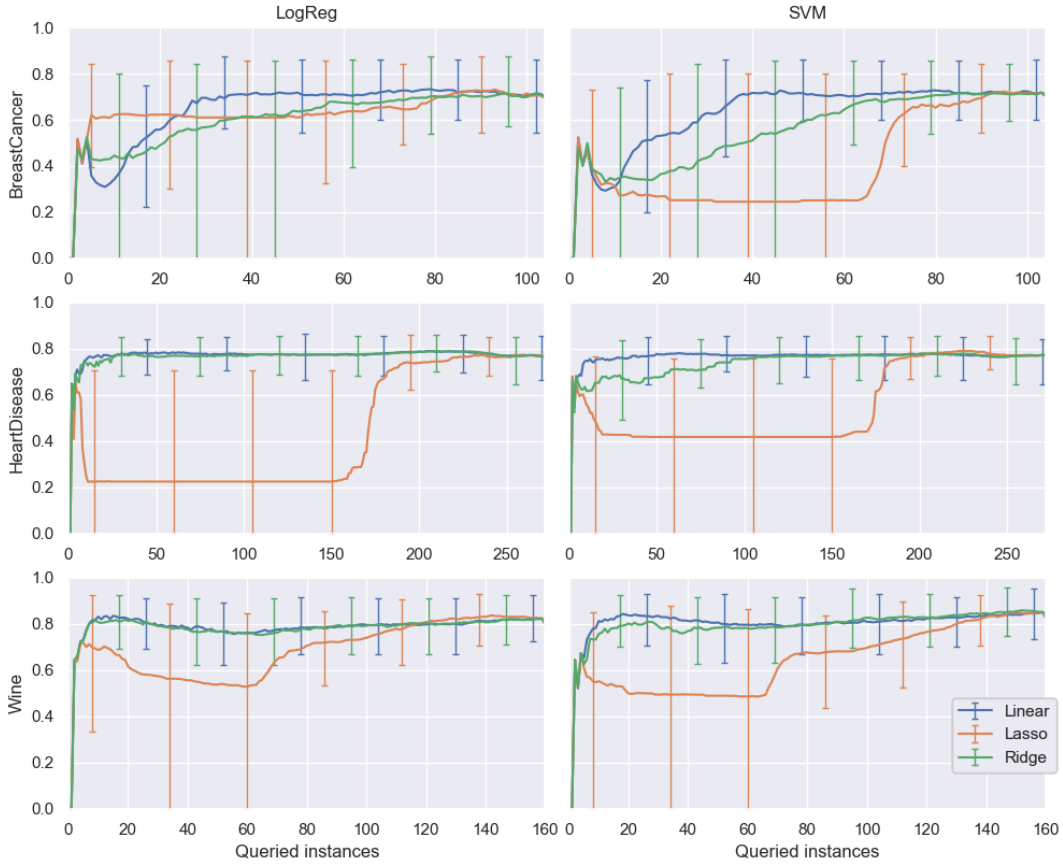



FIGURE 8.8: Comparison of the F1-score when using the three different regression methods, on the UCI datasets.

These results show that linear regression performs the best on the known dataset. For each dataset, lasso regression gives especially mediocre results, where it drops off to below-random guessing for significant amounts of time. This is perhaps to lasso being able to 'drop' parameters more frequently with regularization, which might decrease performance in complex models such as these. In addition, it might have an effect on the variance, where it removes the informativeness of those instances with high variance of the imputed features.

# Chapter 9

# Active Pseudo-Class Selection

In this chapter, we will show an overview of the results: for each of the ACS methods, comparing them to each to other as well as the Random baseline (where each instance is randomly selected).

For each method, we need an initial sample of instances. These instances are then randomly selected from each label alternately. We aim to minimize this number of initial instances so that the method is used as much as possible. The number of instances selected varies: for a proper application of the method we need to have at least one instance for each pseudoclass, in each partition. Selecting $n \cdot k$ as initial sample can be unproductive though: for example with 8 pseudoclasses and 10 partitions, the number of initial random samples could be nearly as high as the entire training set. To counter this, we have implemented a greedy method that selects the most beneficial instance repeatedly until this requirement is fulfilled. The pseudocode is shown in 9.1.

LISTING 9.1: Greedy Initial Sample

```
def InitSample(I, partitions, n=1):
    for each instance ∈ I:
        score[instance] = 0
        score[instance] += 1
            for each pseudoclass in partitions where
            samples for pseudoclass < n
            and partition[instance] == pseudoclass
    return argmax(score)
```

The results are averaged over 50 runs as mentioned in 7, using the F1-score.

For each Active Class Selection method, we first considered all numbers of pseudoclasses from 3 to 8, with a single partition. This allows us to compare the methods to each other, as well as the number of needed pseudoclasses.

Increasing the number of partitions is expected to increase the performance, but only according to the effectiveness of the method itself. Thus, we can properly compare the methods and pseudoclasses hyperparameter with only a single partition. However, for comparing to the baseline we will then increase the number of partitions to get a more reasonable performance.

## 9.1 Redistricting

The results for the Redistricting method are shown in figure 9.1.

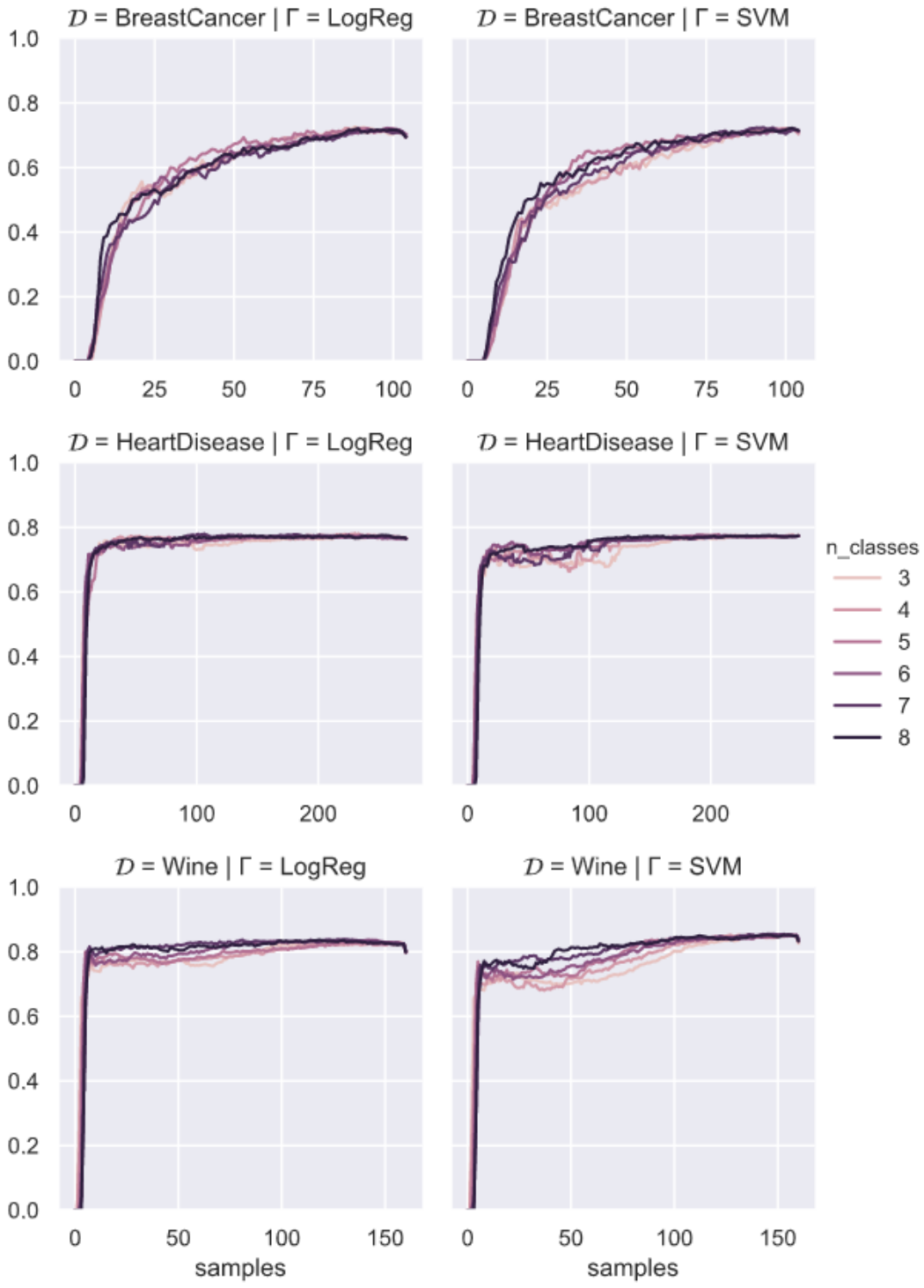



FIGURE 9.1: Comparison of the F1-score of the Redistricting method and the random baseline, on the UCI datasets.

There does not seem to be a significant difference in the number of pseudoclasses, as they all follow a similar trajectory. The most significant difference is the trajectory of the Wine dataset, where a lower number of pseudoclasses results in a dip in performance in the learning curve: here, a higher number of pseudoclasses seems to give a better performance. For the other two datasets—the Breast Cancer and the Heart Disease dataset—the number of 5 pseudoclasses seems to give better results.

However, this does not seem to be significant, and the performance is not generally better or worse with a higher or lower number of pseudoclasses. Thus, it does not seem to be groundbreaking what value we select for this variable.

Table 9.1 shows the Data Utilization Rate values for each number of pseudoclasses (with number of partitions = 1, as such a lower performance is expected).

TABLE 9.1: The Data Utilization Rate results for Redistricting, using the F1-score.

| | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|
| $n$ | BreastCancer | HeartDisease | Wine | BreastCancer | HeartDisease | Wine |
| 3 | 2.28 | **0.53** | 3.42 | 2.17 | 3.68 | 4.20 |
| 4 | 2.28 | 0.68 | 3.45 | 1.93 | 2.70 | 3.84 |
| 5 | **1.92** | 0.66 | **0.21** | **1.23** | 2.68 | 3.16 |
| 6 | 2.12 | 1.63 | 0.25 | 1.63 | **0.51** | 3.40 |
| 7 | 2.72 | 0.66 | 0.25 | 1.73 | 2.41 | 2.76 |
| 8 | 2.24 | 0.74 | 0.29 | 1.63 | 2.41 | **2.00** |

As seen in the figures, for the Wine dataset a higher number seems to perform better; for the other two datasets, the sweet spot seems to be around 5 or 6 pseudoclasses. This is confirmed by the AUC statistic as seen in 9.2. Note: the F1@80% and F1@90% metrics are excluded, as their values are too similar (as can be seen in the learning curve).

TABLE 9.2: The Area Under the Curve results for Redistricting, using the F1-score.

| | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|
| $n$ | BreastCancer | HeartDisease | Wine | BreastCancer | HeartDisease | Wine |
| 3 | 0.57 | 0.73 | 0.77 | 0.54 | 0.71 | 0.75 |
| 4 | 0.57 | **0.74** | 0.78 | 0.54 | 0.72 | 0.76 |
| 5 | **0.58** | 0.74 | 0.78 | 0.57 | 0.73 | 0.78 |
| 6 | 0.57 | 0.74 | 0.79 | 0.56 | **0.73** | 0.77 |
| 7 | 0.56 | 0.74 | **0.80** | 0.55 | 0.72 | 0.79 |
| 8 | 0.57 | 0.74 | 0.80 | **0.58** | 0.73 | **0.79** |

## 9.2 Accuracy Improvement

The results for the Accuracy Improvement method are shown in figure 9.2.

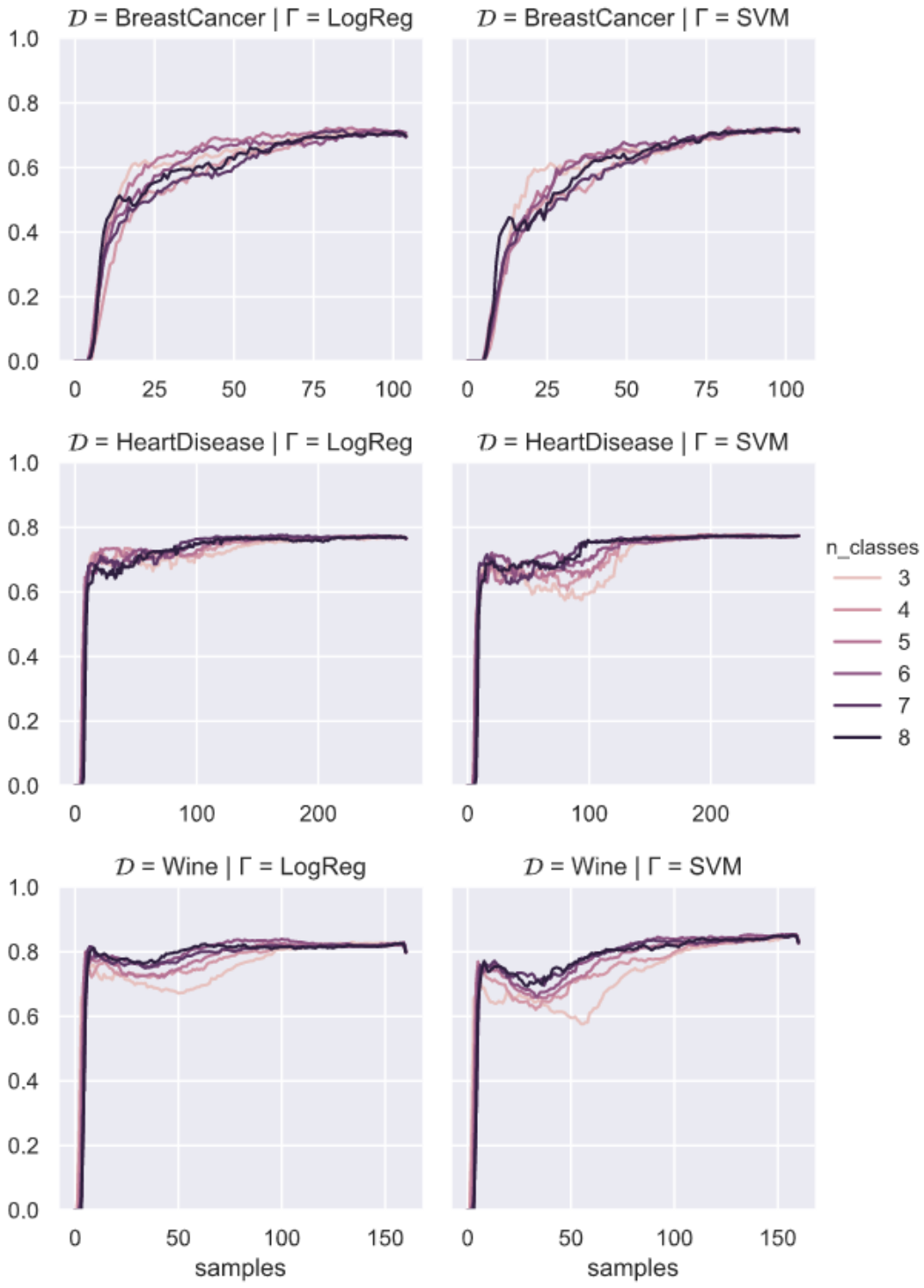



FIGURE 9.2: Comparison of the F1-score of the Accuracy Improvement method and the random baseline, on the UCI datasets.

When compared to the results with the Redistricting method, there is a bit more variance albeit with the same general results: with the Wine dataset, selecting a higher number of pseudoclasses gives the better results—for the other two datasets such a thing cannot generally be said and selecting a more inbetween value of 5 or 6 seems appropriate. Although, it should be noted that this does not seem to be true for the combination of the Heart Disease dataset and the Support Vector Machine

classifier.

Table 9.3 shows the Data Utilization Rate values for each number of pseudo-classes (with number of partitions = 1, as such a lower performance is expected).

TABLE 9.3: The Data Utilization Rate results for AccuracyImprovement, using the F1-score.

| | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|
| *n* | BreastCancer | HeartDisease | Wine | BreastCancer | HeartDisease | Wine |
| 3 | 2.32 | 3.84 | 4.00 | 1.57 | 3.31 | 4.36 |
| 4 | 2.64 | 3.24 | 3.29 | 1.93 | 3.05 | 4.20 |
| 5 | **1.52** | 2.89 | **0.21** | **1.37** | 2.90 | 3.36 |
| 6 | 1.68 | 2.58 | 0.25 | 1.47 | 3.05 | 3.08 |
| 7 | 2.48 | **2.29** | 0.25 | 1.93 | **2.24** | **2.92** |
| 8 | 2.48 | 2.47 | 0.29 | 1.83 | 2.29 | 3.12 |

These values seem to indicate as previously with the Redistricting results. The value of 5 or 6 pseudoclasses seems *in general* like a good value. This is confirmed by the Area Under the Curve score, as shown in table 9.4. Although increasing the number of pseudoclasses to 7 increases the results on same configurations very slightly, the drop in results in other configurations is more significant. Selecting 5 (or 6) pseudoclasses gives the best overall results, and is never far off from optimal.

TABLE 9.4: The Area Under the Curve results for AccuracyImprovement, using the F1-score.

| | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|
| *n* | BreastCancer | HeartDisease | Wine | BreastCancer | HeartDisease | Wine |
| 3 | 0.60 | 0.71 | 0.75 | 0.58 | 0.69 | 0.72 |
| 4 | 0.56 | 0.72 | 0.77 | 0.55 | 0.70 | 0.74 |
| 5 | **0.61** | **0.73** | 0.77 | 0.57 | 0.71 | 0.76 |
| 6 | 0.59 | 0.72 | **0.79** | **0.57** | 0.72 | 0.77 |
| 7 | 0.56 | 0.73 | 0.78 | 0.55 | **0.72** | **0.77** |
| 8 | 0.58 | 0.72 | 0.78 | 0.57 | 0.72 | 0.77 |

## 9.3 PAL-ACS

The results for the PAL-ACS method are shown in figure 9.3.

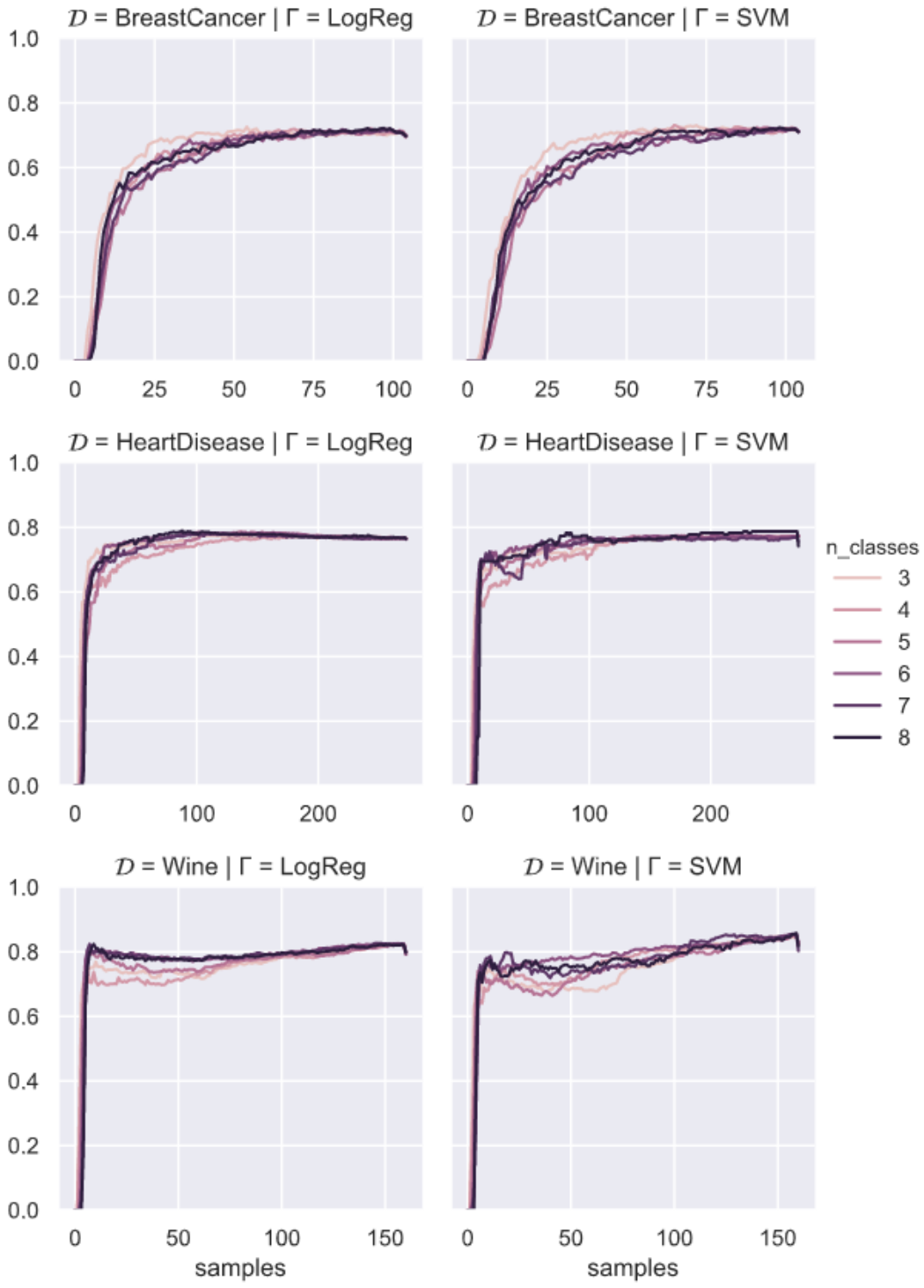



FIGURE 9.3: Comparison of the F1-score of the PAL-ACS method and the random baseline, on the UCI datasets.

The results are mostly similar with regards to the optimal number of pseudo-classes, to the Accuracy Improvement and the Redistricting results. For the Wine dataset, a higher number seems to be better, while for the Breast Cancer dataset a somewhat lower number gives better results.

Although the differences are not always major, we will have different optimal values depending on the dataset.

Table 9.5 shows the Data Utilization Rate values for each number of pseudoclasses (with number of partitions = 1, as such a lower performance is expected).

TABLE 9.5: The Data Utilization Rate results for PAL-ACS, using the F1-score.

| | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|
| $n$ | BreastCancer | HeartDisease | Wine | BreastCancer | HeartDisease | Wine |
| 3 | **0.92** | **0.76** | 4.67 | **0.80** | 2.68 | 4.40 |
| 4 | 1.56 | 2.66 | 5.08 | 1.37 | 2.61 | 4.08 |
| 5 | 1.68 | 1.68 | 0.29 | 1.40 | 1.51 | 4.96 |
| 6 | 1.36 | 0.95 | **0.25** | 1.10 | 1.44 | **3.44** |
| 7 | 1.76 | 1.58 | **0.25** | 1.57 | 1.41 | 4.12 |
| 8 | 1.60 | 1.03 | 0.29 | 1.23 | **1.39** | 4.36 |

These Data Utilization Rate values indicate that the optimal setting indeed differs by dataset. For the Breast Cancer and Wine datasets, it is either lower or higher, for the Heart Disease dataset it differs by classifier even and the best option seems to be an intermediate value.

Additionally, we can look at the Area Under the Curve scores as displayed in table 9.6 and we will be able to draw the same conclusions for each dataset. The Breast Cancer dataset gets better results with a lower amount of pseudoclasses, the Wine dataset gets better results with a higher number of pseudoclasses, and the Heart Disease datasets gets the most consistent results with an intermediate number of pseudoclasses.

TABLE 9.6: The Area Under the Curve results for PAL-ACS, using the F1-score.

| | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|
| $n$ | BreastCancer | HeartDisease | Wine | BreastCancer | HeartDisease | Wine |
| 3 | **0.63** | **0.74** | 0.75 | **0.63** | 0.73 | 0.74 |
| 4 | 0.61 | 0.72 | 0.74 | 0.60 | 0.71 | 0.76 |
| 5 | 0.59 | 0.73 | 0.76 | 0.58 | 0.73 | 0.74 |
| 6 | 0.61 | 0.74 | **0.78** | 0.59 | **0.74** | **0.78** |
| 7 | 0.60 | 0.73 | 0.77 | 0.58 | 0.72 | 0.77 |
| 8 | 0.61 | 0.74 | 0.77 | 0.59 | 0.73 | 0.76 |

## 9.4 The number of pseudoclasses

It is hard to make a concrete statement about the optimal number of pseudoclasses. The results show a variance in the results, implying no certainty about whether one is better than the other. It is most likely also highly dependent on the dataset: all three methods show that using only 3 or 4 pseudoclasses performs poorly on the Wine dataset, while using 5 or more provides good results (even with $k = 1$).

In general, selecting $n = 5$ seems to give the most balanced results, which are never far from the optimal results. As such, to compare the multiple Active Class Selection methods to each other—and to the baseline—we will use $n = 5$.

Unfortunately, this problem does not allow hyperparameter tuning of this sort, as discovering the dataset is done at the same time as the execution of the algorithm.

This implies also the problem with this method as it is, and perhaps for this approach to function properly, the number of pseudoclasses must be adjusted automatically.

## 9.5 Comparing Active Class Selection methods

For comparison, we use $n = 5, k = 1$. Figure 9.4 shows the results on each dataset for each method.

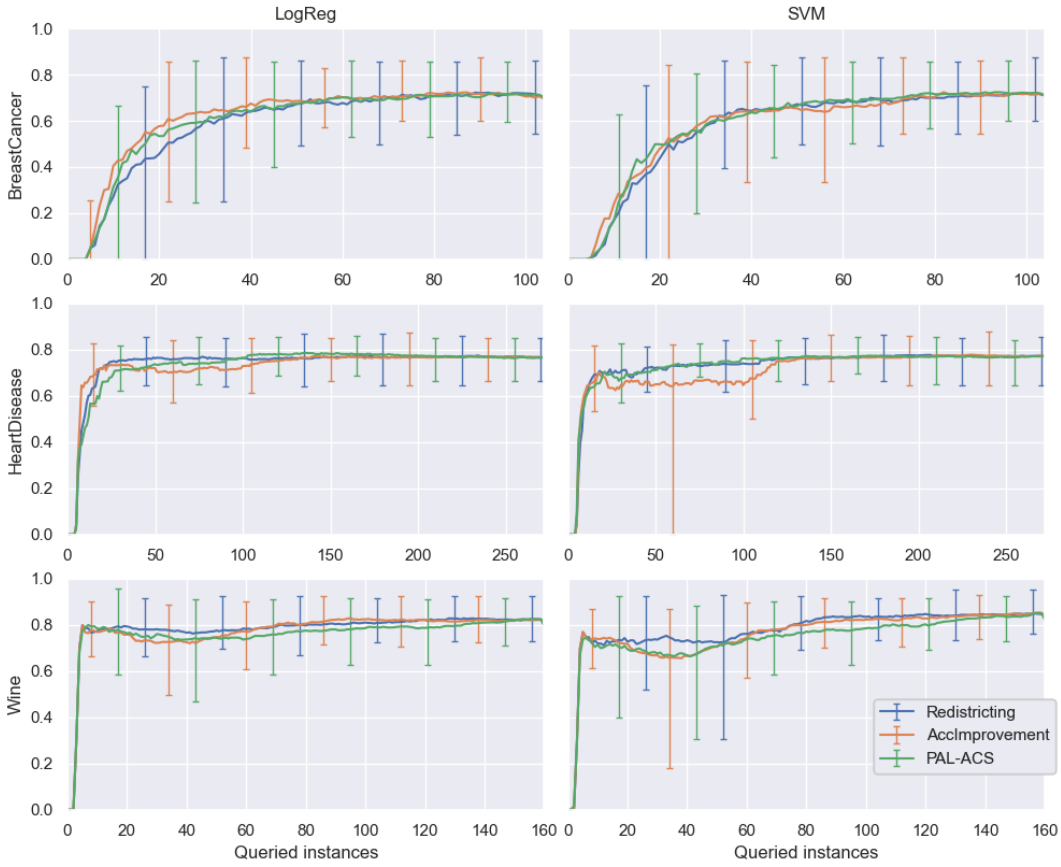



FIGURE 9.4: Comparison of the three ACS methods on the UCI datasets.

These results (9.4) seem to indicate a volatility in the results of Accuracy Improvement. In the last two datasets—which we have previously seen are the most reasonable to expect better performance—show a dip in the results of Accuracy Improvement.

The Redistricting method seems to give the most consistent results: the Accuracy Improvement has volatile results as mentioned, while the PAL-ACS method gives worse results for the Wine dataset. The only decrease in results of the Redistricting method is during the early phases of the Breast Cancer dataset, which is less relevant.

## 9.6 Increasing the number of partitions

In the previous sections, we have determined that the selection of parameters with the most consistently good results is the combination of Redistricting as the Active Class Selection method and $n = 5$. So far, we have been using $k = 1$ for comparison

sake (and to reduce computation complexity), but to compare to the random baseline and get true expected results, we will increase $k$ to 10. Now, we will compare Redistricting with $n = 5, k = 10$ to the random selecction baseline.

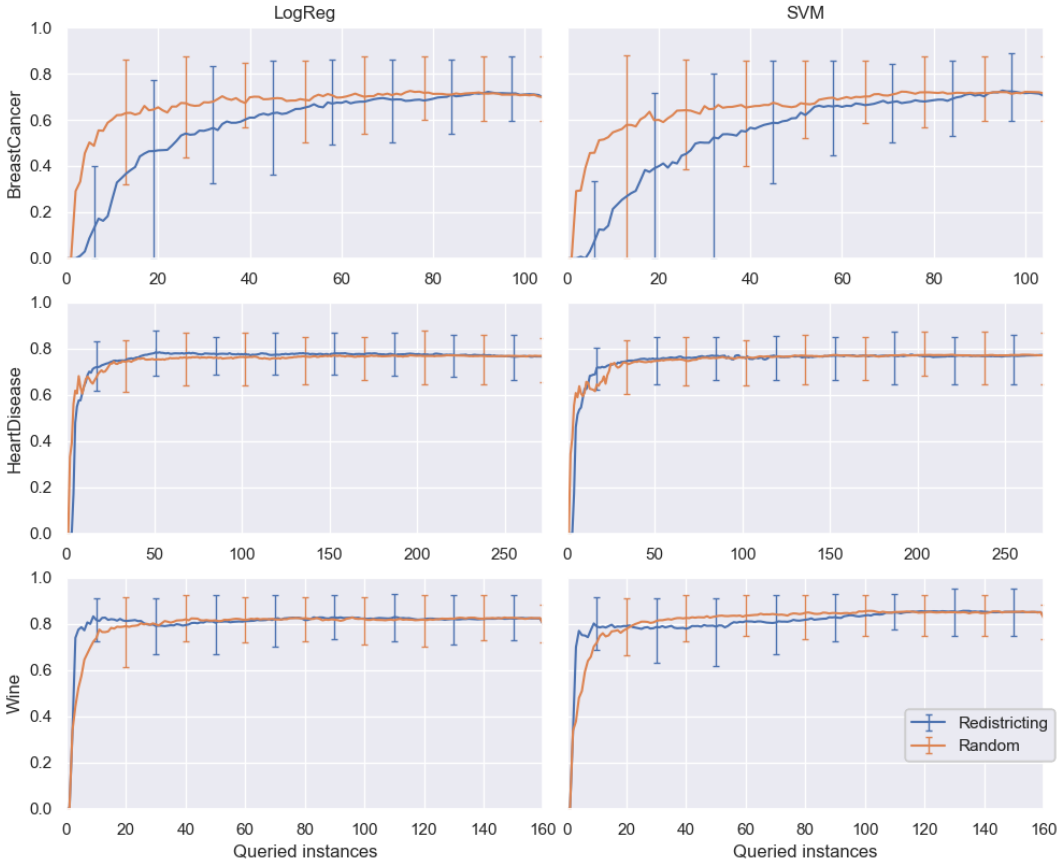



FIGURE 9.5: Comparison of the F1-score of the Redistricting method with increased number of partitions and the random baseline, on the UCI datasets.

As can be seen in figure 9.5, the results are unfortunately disappointing. For the latter two datasets, the results are similar to the random baseline—no significant difference in performance—whereas the performance with the Breast Cancer dataset is significantly better when using the Random baseline method.

As such, we do not recommend further usage of the Active Pseudo Class Selection method for the Active Selection of Classification Features.

# Chapter 10

# Expectation Maximization

As was shown in chapter 8, the most effective utility methods for our problem are AVID and Probability-based. As such, we will use these as our utility method for this approach as well. This approach can possibly give us a more representative indication of the true utility of an instance, removing the variance from using a single instance.

Figures 10.1 and 10.2 indicate that the relevance of the number of clusters differs by dataset: for both methods the results are similar. The Breast Cancer implies that the number of clusters benefits from a higher value; while for the Wine dataset a similar factor is implied albeit less significant. For the Heart Disease, there does not seem to be much correlation and the results is near-identical for each value.

TABLE 10.1: Data Utilization Rates for the EM-based approach, with the AVID utility method.

| | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|
| $n$ | BreastCancer | HeartDisease | Wine | BreastCancer | HeartDisease | Wine |
| 3 | 2.96 | 0.53 | 0.54 | 2.40 | **0.59** | 2.32 |
| 4 | 3.04 | 0.61 | 1.17 | 2.33 | 0.61 | 1.44 |
| 5 | 2.68 | 0.58 | 1.1 | 2.13 | 1.0 | 1.4 |
| 6 | 2.20 | 0.68 | 1.21 | 1.93 | 0.80 | 1.36 |
| 7 | **1.00** | 1.00 | **0.38** | **1.17** | 1.64 | 3.24 |
| 8 | 1.12 | **0.34** | 0.92 | 1.57 | **0.59** | **1.28** |

TABLE 10.2: Data Utilization Rates for the EM-based approach, with the Probability-based utility method.

| | Logistic Regression | | |
|---|---|---|---|
| $n$ | BreastCancer | HeartDisease | Wine |
| 3 | 3.52 | 0.37 | 2.63 |
| 4 | 3.48 | 0.34 | 1.83 |
| 5 | 3.48 | 0.34 | 1.71 |
| 6 | 0.72 | **0.26** | 1.83 |
| 7 | **0.60** | 0.42 | **0.33** |
| 8 | 0.76 | 0.42 | 0.42 |

This is also reflected in the data utilization rates in tables 10.1 and 10.2. The probability-based method provides consistent sub-1 results for $n = 7$ and $n = 8$, which indicates that this method works well with this approach, with those parameters. The AVID method generally indicates the same thing, although it should be noted that the method performs generally worse with the Support Vector Machine classifier which is notable.

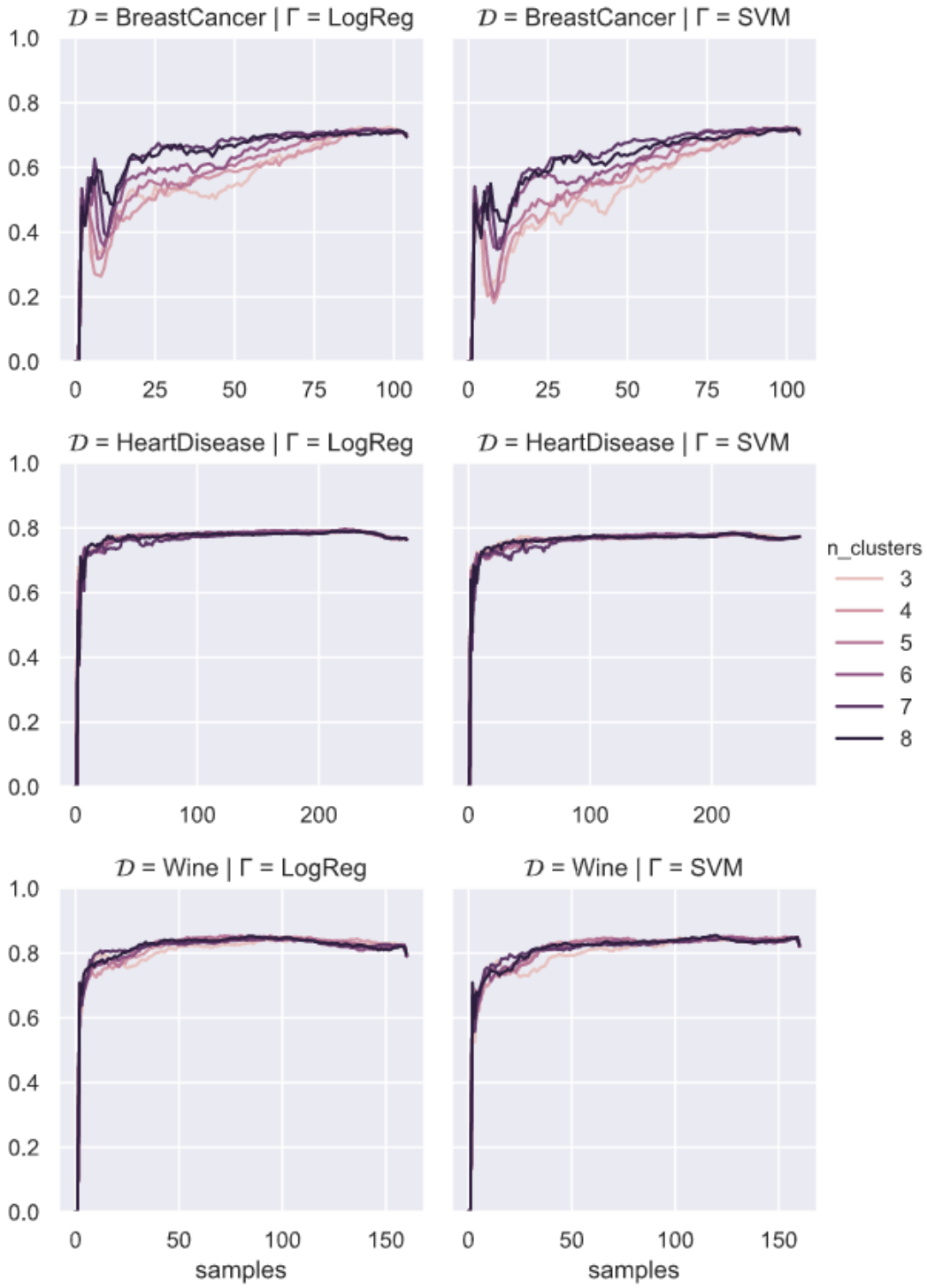



FIGURE 10.1: Comparison of the F1-score of the AVID utility method with varying number of clusters, on the UCI datasets.

## 10.1 Automatic tuning of number of clusters

Looking at tables 10.1 and 10.2, we can generally conclude that a value of $n = 7$ should provide reliable results.

However, the optimal number of clusters might depend on the dataset and even the current selection of instances of this dataset. Tuning this hyperparameter for

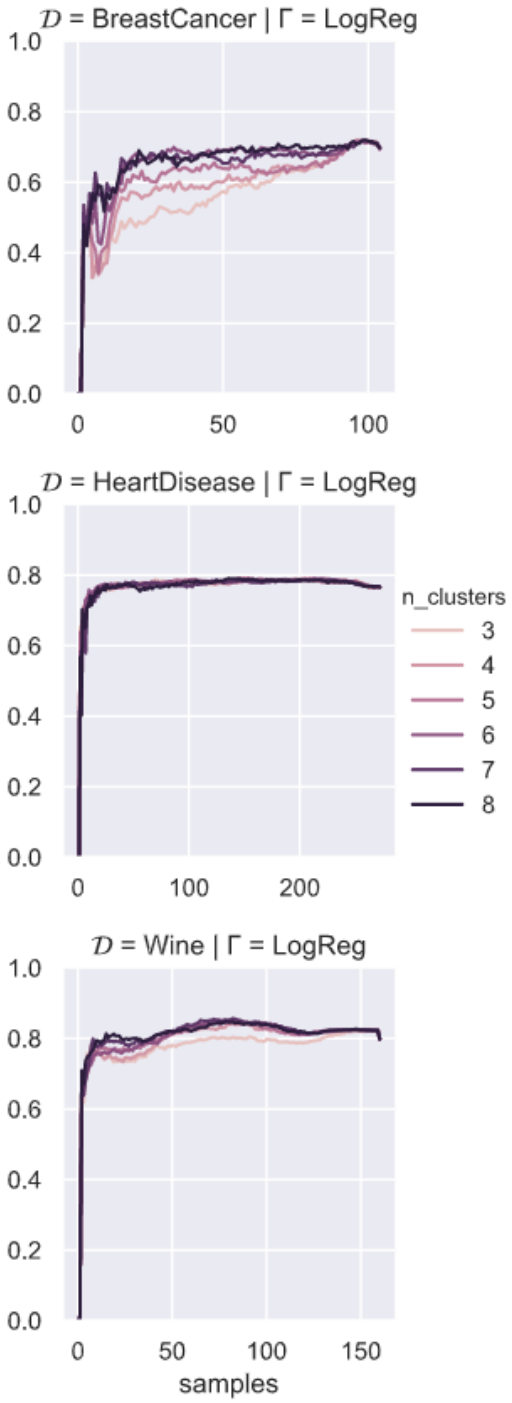



FIGURE 10.2: Comparison of the F1-score of the Probability-based utility method with varying number of clusters, on the UCI datasets.

the dataset is not easily applicable for real-world instances though, as the dataset is unknown at the start of the experiment.

As opposed to the APCS approaches (as defined in chapter 5) though, known methods exist with which we can try to automatically determine the number of clusters or components of the Gaussian mixture model. We will use these methods to see if they can help improve our performance.

We will use a varational Bayesian estimation of a Gaussian Mixture using a

Dirichlet process [8, 5, 9] as implemented in `scikit-learn` [50], as well as a method that fits on a range of possible values and selects the highest scoring value—either with the BIC score [56] or the AIC score [2].

The results are shown in figure 10.3 and table 10.3. An initial look at the Data Utilization Rates show an indication that automatically determining the number of clusters with this straightforward manner does not actually increase performance. Whereas using a set number such as simply using 3 clusters allows sub-1 rate for three of the dataset/classifier combinations, this is only true for one or two combinations for all automatic methods.

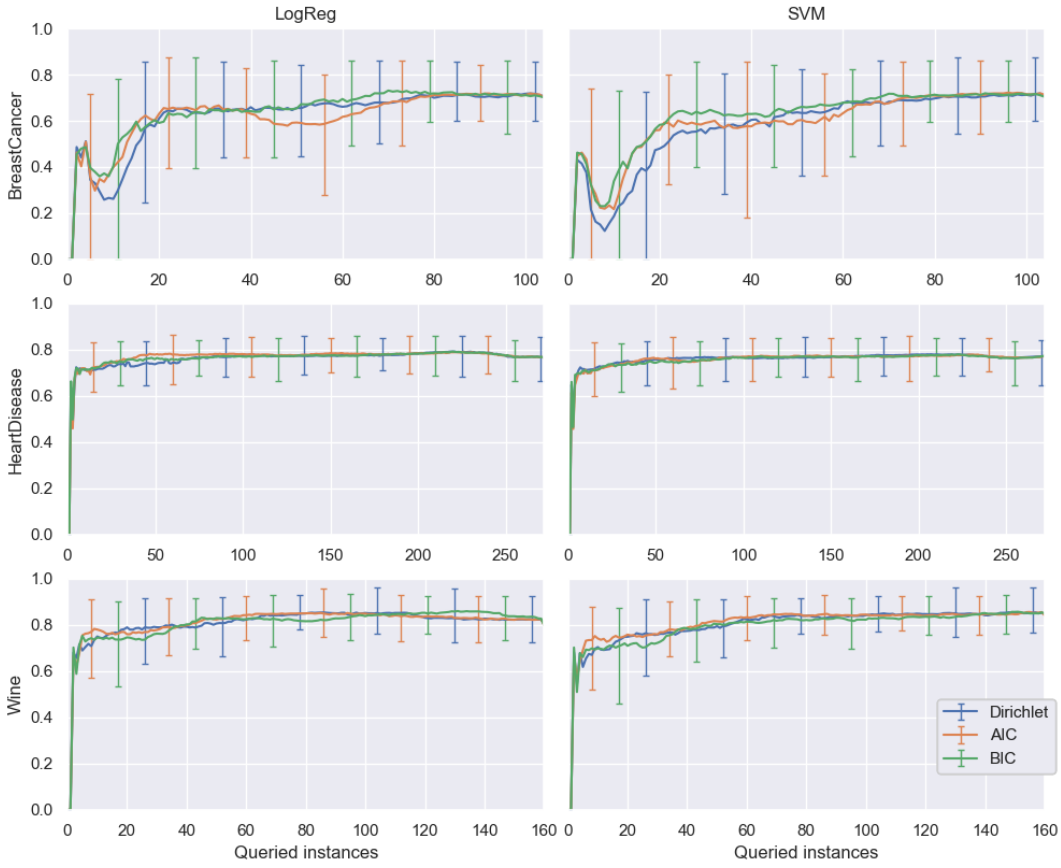



FIGURE 10.3: Comparison of the F1-score of the different tuning methods of number of clusters with the AVID utility method, on the UCI datasets.

TABLE 10.3: Data Utilization Rates for the EM-based approach and the AVID utility method, with estimation of number of components.

| | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|
| method | BreastCancer | HeartDisease | Wine | BreastCancer | HeartDisease | Wine |
| Dirichlet | 2.08 | 1.47 | 1.38 | 1.90 | 0.78 | 2.28 |
| BIC | 1.76 | 0.66 | 1.50 | 1.03 | 1.12 | 2.24 |
| AIC | 1.16 | 0.74 | 1.58 | 2.03 | 0.83 | 1.88 |

Perhaps this is due to adding a layer of complexity in an already complex problem. We are fitting the Gaussian mixture model on a subset of the complete dataset, and as such we might say that the automatic methods are more prone to overfitting.

This is because they have the possibility to increase the number of components quite easily if it allows a better description of this subset.

However, it should be noted that comparing active learning methods is not as simple as this one metric, and looking at the plots does show that these methods perform significantly better at the Breast Cancer dataset than using a relatively low number of components. Still, there is a decrease in performance when compared to random sampling. As such, we can conclude that this increase in performance is more due to less bias in its samples, but not in better selection. The complexity of the Breast Cancer dataset punishes methods with high bias, but does not reward interesting selection methods.

## 10.2 Comparison to baseline

To evaluate the effectiveness of the Expectation Maximization-based approach, we compare it to the Random baseline: we should expect it to get some reasonable improvements in at least some of the dataset/classifier combinations. We will compare $n = 7$ and $BIC$ to the Random baseline: previous sections have shown that these are the most consistently stable results.

Figures 10.4 and 10.5 show the learning curves compared to the learning curve of the random baseline. The results show that automatically tuning the number of clusters can cause a significant dip in the curve; while it can indicate an increase in the latter part of the curve. Using a set value of 7 for the number of clusters gives more consistent results and can cause improvements upon the Random baseline—although it can also cause some dips when compared to the baseline.

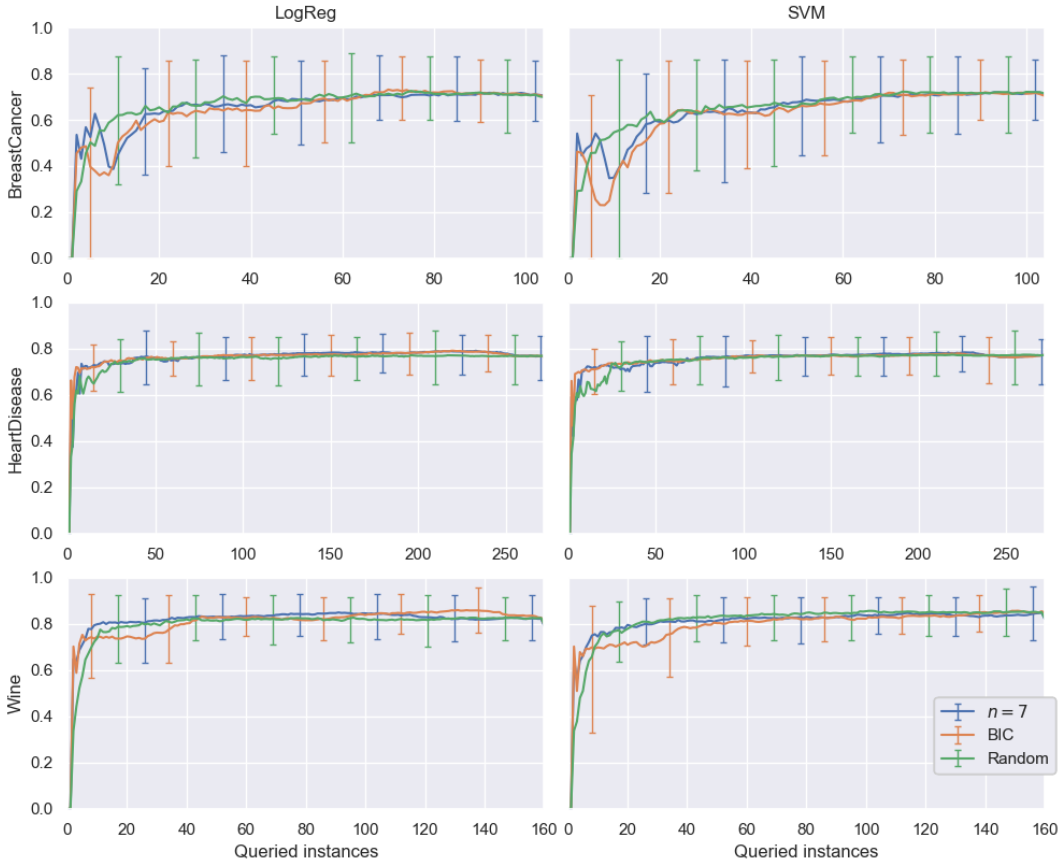



FIGURE 10.4: Comparison of the F1 score of the EM-based approach and the AVID utility method and the random baseline, on the UCI datasets.

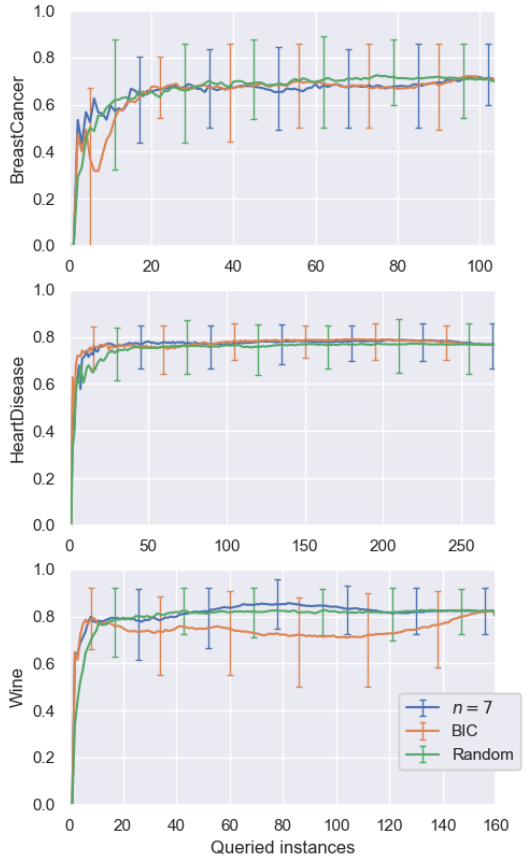



FIGURE 10.5: Comparison of the F1 score of the EM-based approach and the Probability-based utility method and the random baseline, on the UCI datasets.

Tables 10.4 and 10.5 show the relevant statistics for these three methods. What these figures illustrate are where each method performs best, as one statistic cannot summarize an entire learning curve properly, but can indicate strengths and weakness of a learning curve.

Looking at the statistical figures for the Probability-based method, we can see that the Data Utilization Rates are excellent: both parameter settings result in a sub-1 rate for each dataset. However, the F1@80% shows only improvement upon random selection for one dataset. This indicates that using this approach with the Probability-based method results in improvement early on in the process, while conforming to random later on.

For the BIC parameter with the AVID utility method, the opposite seems to be true. The Data Utilization Rates are not as interesting, but for most dataset and classifier settings (and for all datasets with Logistic Regression), the F1@80% score shows improvement upon random selection. So, one would expect a slow start but ultimately an improvement upon random selection with a significant enough sample set size.

Both these suspicions can be confirmed in the learning curves 10.5 and 10.4. This indicates that the correct approach to take for a problem can also depend on the wanted effect: are we only able to query a minimal amount of instances, or do we want to shave off the last few unnecessary ones and gain an improvement with a larger set? For the former goal, we would prefer the Probability-based method, for the latter we would prefer the AVID utility method with parameter tuning.

TABLE 10.4: Numerical results for the EM-based approaches, with the AVID utility method. Improvements upon random selections are bolded.

| | | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|---|
| | | Breast Cancer | Heart Disease | Wine | Breast Cancer | Heart Disease | Wine |
| $n = 7$ | DUR | 1.00 | 1.00 | **0.38** | 1.17 | 1.63 | 1.24 |
| | AUC | 0.65 | **0.76** | **0.81** | 0.63 | 0.75 | 0.81 |
| | F1@80% | 0.71 | **0.79** | 0.82 | **0.72** | **0.78** | 0.84 |
| | F1@90% | 0.71 | **0.79** | 0.82 | 0.72 | 0.77 | 0.83 |
| BIC | DUR | 1.76 | **0.66** | 1.50 | 1.03 | 1.12 | 2.24 |
| | AUC | 0.64 | **0.76** | **0.80** | 0.61 | 0.75 | 0.79 |
| | F1@80% | **0.72** | **0.78** | **0.86** | 0.71 | **0.78** | 0.83 |
| | F1@90% | 0.71 | **0.78** | **0.85** | 0.71 | 0.77 | 0.85 |
| Random | DUR | 1 | 1 | 1 | 1 | 1 | 1 |
| | AUC | 0.66 | 0.75 | 0.79 | 0.64 | 0.75 | 0.81 |
| | F1@80% | 0.71 | 0.77 | 0.82 | 0.71 | 0.77 | 0.85 |
| | F1@90% | 0.71 | 0.77 | 0.82 | 0.72 | 0.77 | 0.85 |

TABLE 10.5: Numerical results for the EM-based approaches, with the Probability-based utility method. Improvements upon random selections are bolded.

| | | Logistic Regression | | |
|---|---|---|---|---|
| | | Breast Cancer | Heart Disease | Wine |
| $n = 7$ | DUR | **0.60** | **0.42** | **0.33** |
| | AUC | 0.65 | **0.77** | **0.81** |
| | F1@80% | 0.69 | **0.79** | 0.82 |
| | F1@90% | 0.70 | 0.78 | 0.82 |
| BIC | DUR | **0.80** | **0.34** | **0.38** |
| | AUC | 0.64 | **0.77** | 0.74 |
| | F1@80% | 0.67 | 0.79 | 0.74 |
| | F1@90% | 0.71 | **0.78** | 0.79 |
| Random | DUR | 1 | 1 | 1 |
| | AUC | 0.66 | 0.75 | 0.79 |
| | F1@80% | 0.71 | 0.77 | 0.82 |
| | F1@90% | 0.71 | 0.77 | 0.82 |

# Chapter 11

# Comparing the Approaches

In the previous chapters (8, 9, 10) we considered and evaluated each approach and examined their results and potential parameter configurations. We concluded what methods and parameter settings worked well, and what methods and parameter settings had subpar results. In this chapter, we compare the three approach to see what method generally provides the best results.

We will compare the methods as described in table 11.1, as these showed the most consistent or best results in the previous chapters. Although the APCS approach showed lackluster results, we will include one configuration for referential purposes. The approaches with the Probability-based utility method (suffixed with —Prob) will only be compared using the Logistic Regression classifier.

| Approach | Parameters |
|---|---|
| Active Classification Feature Acquisition—AVID | $B = 10, \lambda = 1$ |
| Active Classification Feature Acquisition—Prob | $b =$ auto |
| Active Pseudo-Class Selection—Redistricting | $n = 5, k = 10$ |
| Expectation Maximization—AVID | n_clusters $= 7$, $B = 10$ |
| Expectation Maximization—Prob | n_clusters $= 7$, $b =$ auto |
| Random | - |

TABLE 11.1: The approaches and configurations which we will compare to each other.

The learning curves are as shown in figure 11.1, using the F1-score. For additional insight, the learning curves using the ROC AUC score are shown in figure 11.2.

The differences for the approaches are, as in all previous experiments, dependent on the dataset (and the classifier). The Active Pseudo-Class Selection performs better than expected on the Heart Disease and the Wine datasets, but most significantly performs very poorly on the Breast Cancer dataset.

In general, it is hard to draw any conclusions from the Heart Disease dataset: all approaches perform similarly; the only that differs from the rest is the ACFA-Prob approach, which does so in a negative way. The Wine dataset shows that the ACFA-AVID approach has a bit more disappointing results compared to the other methods, while the Expectation Maximization-based approach seem to perform slightly better thajn the other ones.

The Breast Cancer dataset displays the most significant differences in performance when looking at the learning curves. The Active Pseudo-Class Selection method has, as mentioned, very poor performance with this dataset. The approaches using the AVID utility method, seem to have inconsistent performance here: with a mediocre start in the process but improving later on. The Probability-based methods

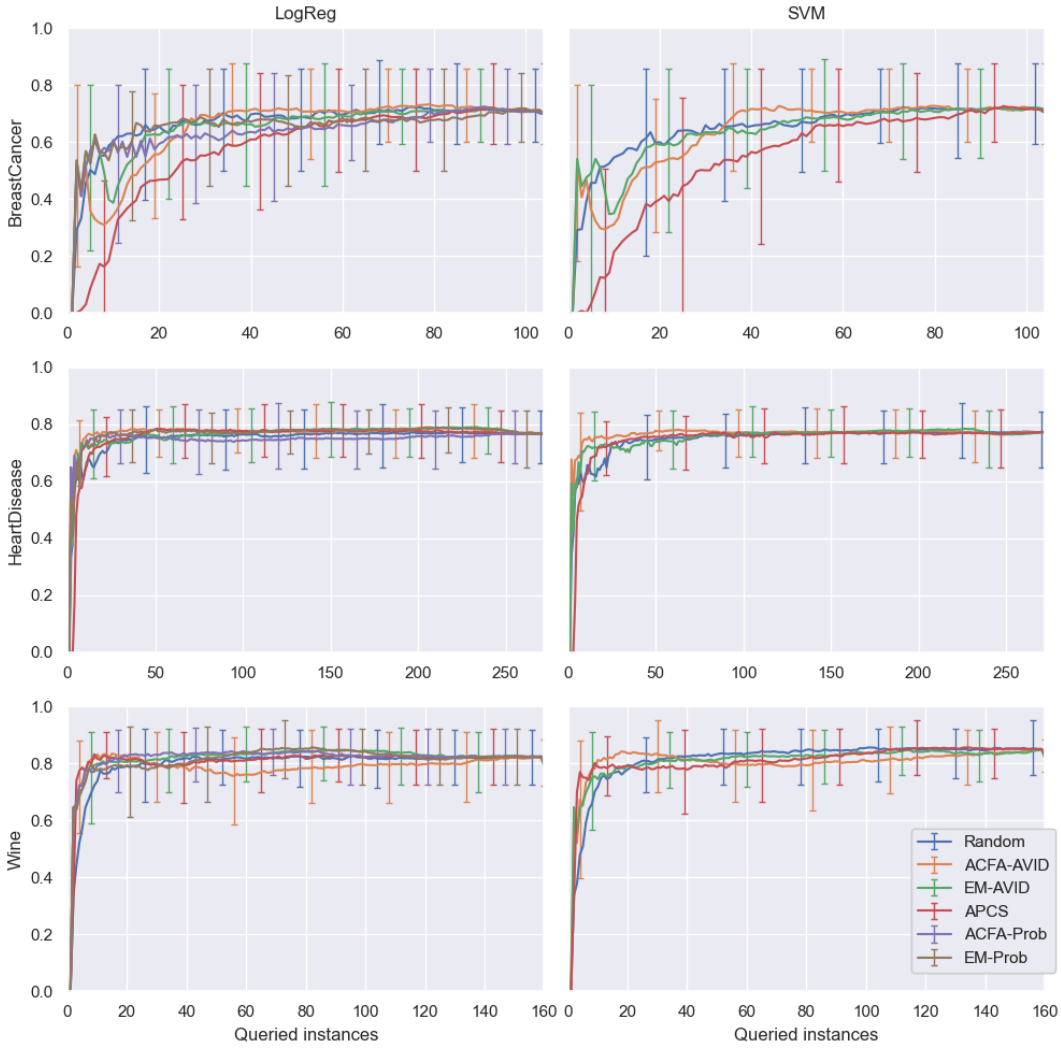



FIGURE 11.1: Comparison of the F1-score of the selected approaches and configurations, on the UCI datasets.

provide the most consistent results; with a decent start but being overtaken by the AVID utility methods when enough instances are sampled.

The resulting metrics as defined in chapter 7 are shown in table 11.2.

The combination of all these metrics allows us to consider multiple aspects simultaneously: the approaches, the classifiers, the datasets, and specific aspects of the learning curves.

Considering the datasets first, these metrics show that all approaches are not able to consistenly improve upon the baseline for the Breast Cancer dataset, and this dataset thus creates a more complicated problem. It should be noted that although this dataset is hard for all approaches, the results do differe significantly as this is likely where unwanted bias can be introduced for the approaches.

The ACFA-AVID approach has a slightly (negligible) above 1 Data Utilization Rate, but is able to improve upon the F1 score when only 80% of the instances are able to sampled and is thus appropriate for scenarios where a similar amount of instances can be queried. The EM-Prob approach is able to obtain a lower Data Utilization Rate, at the cost of a lower F1 score at 80% and 90% of all instances; and could thus be appropriate for small query sets.

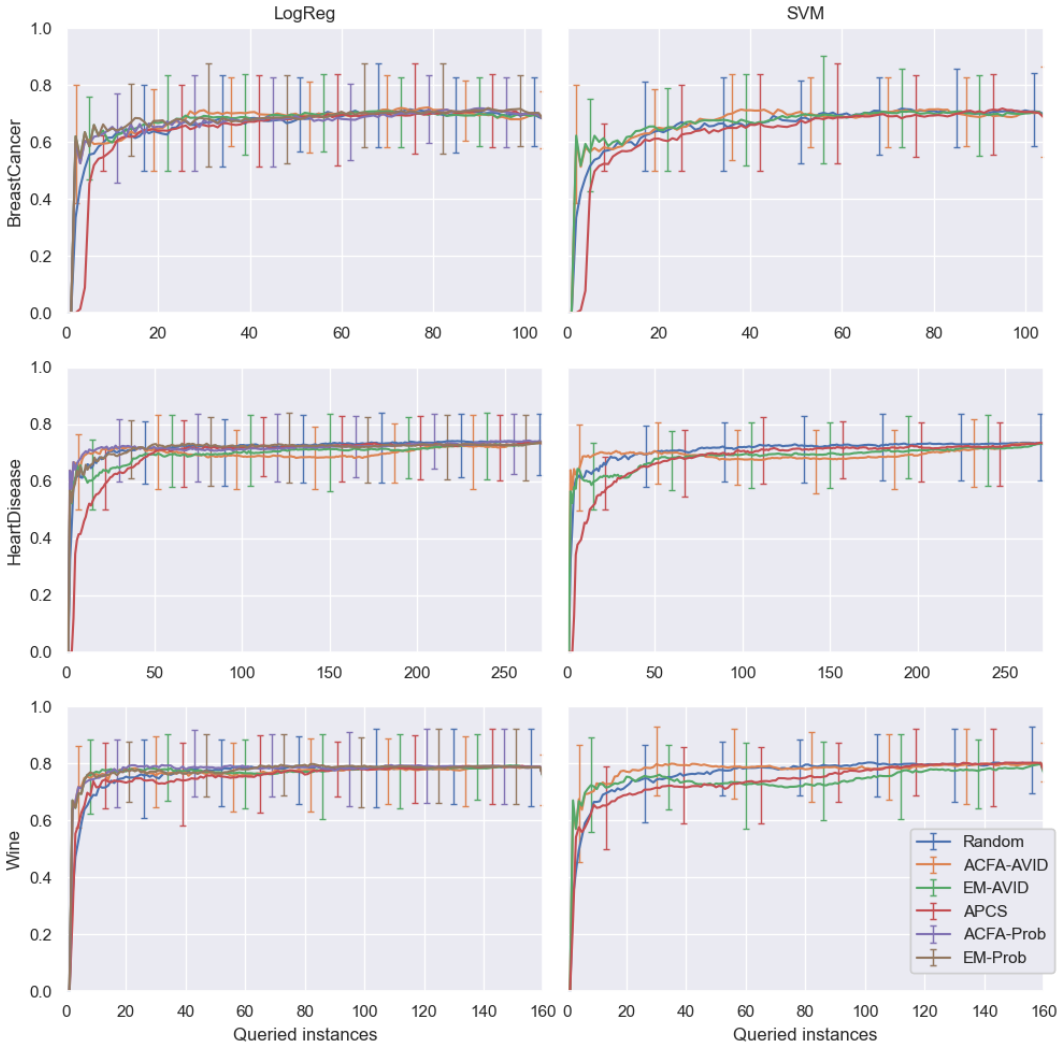



FIGURE 11.2: Comparison of the ROC AUC-score of the selected approaches and configurations, on the UCI datasets.

The Active Pseudo Class Selection is an outlier here when concerning Area Under the Curve. Although this metric simplifies the entire learning curve into only one number, it can still be a good general indication of the entire curve which the other metrics cannot. It is only 0.57 and 0.54 which is almost 0.1 lower than all the other approaches: this is a representation of the huge dip in the curve as seen in the figures 11.1.

The Heart Disease datasets provides a much more approachable problem, and has better results for most classifiers (considering for now the Logistic Regression model). Especially the AVID utility method-based approaches (ACFA and EM) have consistently better (or at least as good) results in every metric. This implies that we can expect the entire learning curve to consistently improve upon the random baseline. The same is true for the EM-Prob approach. The ACFA-Prob and APCS approaches provide an improved Data Utilization Rate, but not much more than that.

The Wine dataset (considering for now the Logistic Regression model) allows for a consistent improvement upon the Data Utilization Rate for all approaches, as well as the Area Under the Curve. However, with the exception of the ACFA-Prob

TABLE 11.2: All numerical results for comparing the several approaches. Improvements upon random selections are bolded.

| | | Logistic Regression | | | Support Vector Machine | | |
|---|---|---|---|---|---|---|---|
| | | Breast Cancer | Heart Disease | Wine | Breast Cancer | Heart Disease | Wine |
| ACFA-AVID | DUR | 1.08 | **0.26** | **0.29** | 1.13 | **0.24** | **0.40** |
| | AUC | 0.64 | **0.77** | 0.78 | 0.63 | **0.76** | 0.80 |
| | F1@80% | **0.73** | **0.79** | 0.80 | **0.72** | 0.77 | 0.83 |
| | F1@90% | 0.71 | 0.77 | 0.81 | 0.72 | 0.77 | 0.84 |
| ACFA-Prob | DUR | 2.36 | **0.34** | **0.46** | | | |
| | AUC | 0.64 | 0.74 | **0.81** | | | |
| | F1@80% | 0.71 | 0.76 | 0.82 | | | |
| | F1@90% | **0.72** | 0.77 | **0.83** | | | |
| APCS | DUR | 2.16 | **0.71** | **0.29** | 1.80 | **0.83** | 2.40 |
| | AUC | 0.57 | 0.75 | **0.80** | 0.54 | 0.74 | 0.81 |
| | F1@80% | 0.71 | 0.77 | 0.82 | 0.69 | 0.77 | 0.85 |
| | F1@90% | 0.71 | 0.77 | 0.82 | 0.72 | 0.77 | 0.85 |
| EM-AVID | DUR | 1.00 | 1.00 | **0.38** | 1.17 | 1.63 | 1.24 |
| | AUC | 0.65 | **0.76** | **0.81** | 0.63 | 0.75 | 0.81 |
| | F1@80% | 0.71 | **0.79** | 0.82 | **0.72** | **0.78** | 0.84 |
| | F1@90% | 0.71 | **0.79** | 0.82 | 0.72 | 0.77 | 0.83 |
| EM-Prob | DUR | **0.60** | **0.42** | **0.33** | | | |
| | AUC | 0.65 | **0.77** | **0.81** | | | |
| | F1@80% | 0.69 | **0.79** | 0.82 | | | |
| | F1@90% | 0.70 | 0.78 | 0.82 | | | |
| Random | DUR | 1 | 1 | 1 | 1 | 1 | 1 |
| | AUC | 0.66 | 0.75 | 0.79 | 0.64 | 0.75 | 0.81 |
| | F1@80% | 0.71 | 0.77 | 0.82 | 0.71 | 0.77 | 0.85 |
| | F1@90% | 0.71 | 0.77 | 0.82 | 0.72 | 0.77 | 0.85 |

they cannot make improvements upon the query set of 80% or 90%. This implies that perhasp there is not much to improve in the latter stages of the process, but all approaches consistently find improvement in the earlier stages where this is still possible.

Finally, it should be noted that the results for the Support Vector Machine classifier model are generally worse than the results for the Logistic Regression model (although not for every configuration).

Unfortunately, it is hard to give a clear-cut conclusion about the 'best' approach. There is no approach that provides consistently better-than-random results for the entire learning curve, for all classifiers and datasets. On the other hand, there is no possibility for trial-and-error decisions of this kind, as the experiments are being performed as they are evaluated. This is discussed more extensively in part IV.

# Chapter 12

# Case Study: Schizophrenia Prediction

In this chapter, we consider the real-world problem of the prediction of schizophrenia diagnosis based on MRI scans. This dataset originally is from [46], and has been reused for our purpose. In the previous four chapters—8, 9, 10, 11, we compared the approaches and deduced the most effective approaches based on benchmark and synthetic datasets. We will now use those methods with the setting of schizophrenia prediction, hoping to reduce the number of MRI scans needed for a similar performance.

## 12.1 Evaluation

Whereas we used just the F1-score in the previous chapters (as defined in chapter refchap:methodology) to evaluate the approaches on the benchmark and synthetic datasets, we will use both the F1-score and the well-known and straightforward accuracy score for this case study. This performance measure is regularly used in similar research, as well as in the previous study [46]. We will still use the F1-score, as some of the data is not balanced regarding the binary labels and as such the F1-score is able to illustrate more properly the performance compared to random guessing the majority class. To illustrate the imbalance between the binary labels, table 12.1 shows the number of labels for both configurations of the dataset.

TABLE 12.1: The binary label imbalance in both configurations of the dataset.

| | 0 | 1 | Total |
|---|---|---|---|
| Instances with IQ feature | 400 | 233 | 633 |
| All instances | 489 | 325 | 814 |

As our goal for this case study is to reduce the number of instances needed with similar performance, we will evaluate primarily using the learning curve with a focus on the mid-to-latter part of the curve.

Although additionally we would be interested in if the performance would be increased with the same number of samples, we can not evaluate this as we can only sample from the existing real-world instances. The last instances are in a way *forced* as well, as only the remaining instances can be selected. Thus, the goal of reducing the instances needed can represent this goal as well: only with a smaller training set.

Perhaps the model could also improve on the final model, by being able to filter out instances that are detrimental. This does not seem likely though, as the instances that are detrimental are generally outliers and thus hard to predict or estimate.

## 12.2 Exploring the dataset

The dataset consists of a set of selection features—age, sex and IQ—and a much larger set of classification features, as well as a binary label. These classification features consist of: the rotation of the left and right hemisphere, the volume of multiple subcortical areas, and for 68 cortical areas of interest is defined the cortical thickness, the volume and the area.

To get some ideas about the dataset, we have done some preprocessing and fitted several classification algorithms on the entire dataset (via passive learning). This allows us to get an idea of the dataset and what is reasonable regarding expected performance, as well as the appropriate models.

### 12.2.1 Preprocessing

The dataset needs some basic preprocessing. Two rows are faulty (5 and 481), and are dropped. The IQ value is missing for a significant amount of instances, as such we will run the experiment both without the IQ score (and thus a larger dataset size) and with the IQ score (but more instances with missing values filtered out).

318 Rows contain missing values, such as the diagnosis of the patient or the IQ score, which are all dropped as well leaving 633 rows to be included. For the instantiation of the experiment without IQ scores, 814 rows are included.

All MRI features are standardized, such that the mean $\mu$ equals 0, and the standard deviation $\sigma$ equals 1.

A subset of the features should also be removed: all features concerning the putamen, caudate, and pallidum values. These form the stratium, which is known to be influenced by antipsychotic medication. These features can thus be used to find patients using medication, instead of patients diagnosed.

This leaves a dataset consisting of 633 or 814 instances, all with two or three selection features—age, sex and IQ, 264 classification features deduced from the MRI scans, and a binary label.

### 12.2.2 Classification algorithms

Using the dataset, we have fitted and evaluated several classification algorithms for the purposes of considering the most appropriate classifier models for this task. For explainability reasons, *random forest* and *neural network* methods are excluded. Other than those two, we used standard methods. The results are shown in table 12.3.

The classifiers are evaluated via 5-fold cross-validation. The dataset is split up into five folds, with each fold being used as the test set once; the other four folds are then the training set. The average accuracy of those five runs is then its final metric. Note that for this evaluation, we included all instances without an IQ score but otherwise complete. These results show that SVM-RBF and Logistic Regression are the most well-fitted models for this problem, and will be used for this experiment.

It should be noted that these results are meant to illustrate the most fit classifier models for our problem and thus tuned the hyperparameters on the full dataset, but for our experiments we will use default parameters for reasons explained in chapter 7. As mentioned, this is expected to slightly decrease the performance during the

experiments, but it allows us to more fairly evaluate and compare the several approaches. The values for these parameters are as mentioned in chapter 7, and seen in table 12.2. In the real-world scenario, one would tune the hyperparameters to increase performance, but in this scenario we aim to construct a setting as similar as possible.

TABLE 12.2: The hyperparameters used for both classifier models.

| Classifier | Parameters |
|---|---|
| Logistic Regression | $C = 1.0, L2$ |
| Support Vector Machine (RBF) | $C = 1.0, \gamma = \frac{1}{\texttt{n\_features} \cdot \texttt{var(X)}}$ |

TABLE 12.3: The results for each classification algorithm on the dataset.

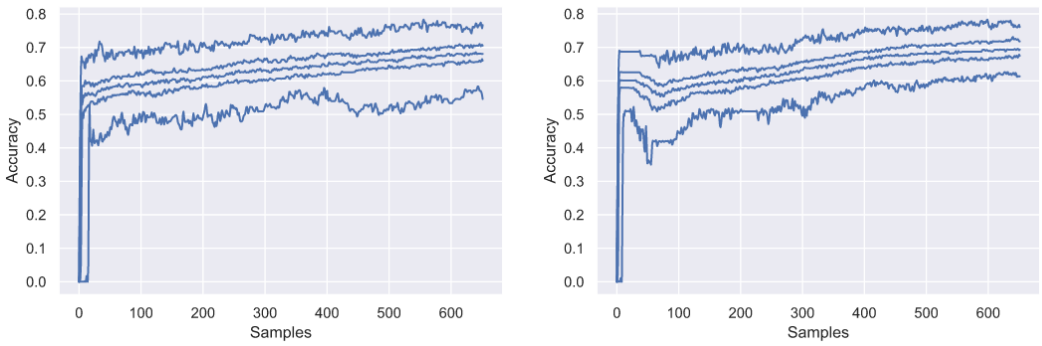
| Classifier | Accuracy |
|---|---|
| k-Nearest Neighbors | 0.642 |
| Support Vector Machine (linear) | 0.671 |
| Support Vector Machine (RBF) | 0.688 |
| Support Vector Machine (polynomial) | 0.644 |
| Logistic Regression | 0.700 |

### 12.2.3 Random sampling

To get an idea of the difference in information between instances, we can consider the random sampling method. This is simple to implement—and does not use the selection features—and can be tested with the methodology described in Section 7.1.

Although it must be taken into account that because this sampling is random and unorchestrated, with the high number of samples the 'good' runs will be subdued. Still, if there is a significant variance within the best and worst runs, it shows that there is the possibility of selecting more favorably or less favorably.

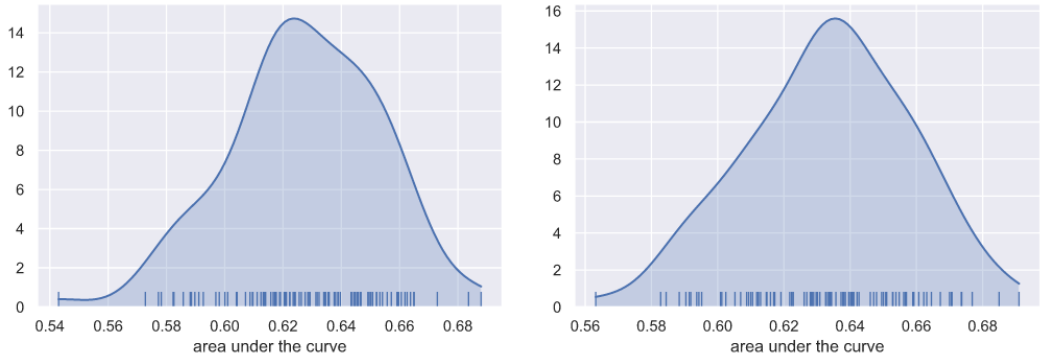
Figure 12.1 shows the quartiles of 100 runs, with both support vector machines and logistic regression.




(A) Random sampling with the RBF kernel. (B) Random sampling with Logistic Regression.

FIGURE 12.1: Performance of the random sampling baseline, quartiles of 100 runs.

Figure 12.2 shows the distribution of the area under the learning curve for each run. Although area under the curve is not the most effective measure, ones such as Data Utilization Rate (7.1) are not as easy applicable for random sampling, and the area under the curve still allows us good insight into the difference inbetween runs.




(A) Random sampling with the RBF kernel. (B) Random sampling with Logistic Regression.

FIGURE 12.2: Distribution of the random sampling performance, area under the curve.

As can be seen from both of these figures, even within the random sampling approach there can be seen a notable difference in performance between runs. Although most runs fall within a smaller range, some runs are noticably better than the others. If it is possible to consistently sample runs in similar performance to these, that is already an improvement. If improvement can be found on the better random sampling runs, that is even better.

### 12.2.4 Feature selection

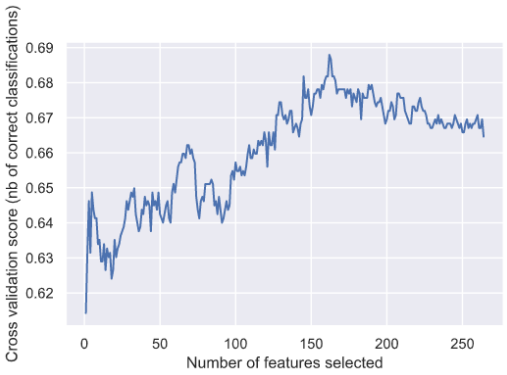
The dataset contains 264 MRI features, which can be used for classification. In reality, not all of these features might prove to be useful for prediction. Some features might actually be detrimental to prediction, especially when a larger number of features are present: this can make it more complicated for the model to converge. Even when there are no detrimental features, we still might consider reducing the feature set. Working with a large feature set incurs extra computational costs, so working with less features can make for a more computationally efficient model.

There are many feature selection methods out there. For our case, we consider those that are focused on numerical inputs and binary categorical outputs. Preferably, we use *filter* methods so we can keep the feature set consistent for all potential classifiers.

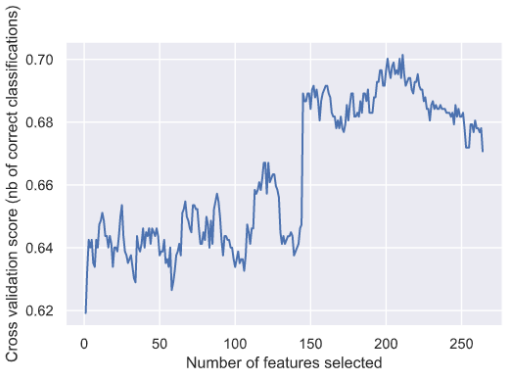
#### ANOVA

For the case of binary classification with numerical inputs, one of the most used filter methods is known as ANOVA (Analysis of Variance) [58]. With ANOVA, we analyse the variance between each feature and the binary output label. The features with the most between-group variance with the output, is most informative.

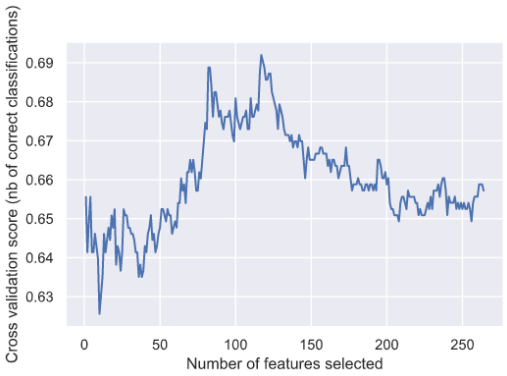
Figure 12.3 shows the performance of the model (with default hyperparameters) based on the number of features, where the $k$ best features are selected using ANOVA.



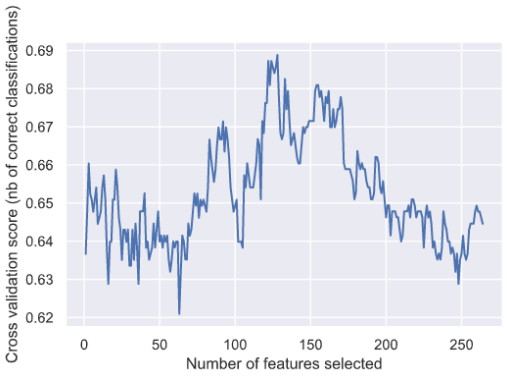

(A) Feature selection results with the RBF kernel, without IQ features (and thus more instances).




(B) Results with Logistic Regression, without IQ features (and thus more instances).




(C) Feature selection results with the RBF kernel, with IQ features.




(D) Results with Logistic Regression, with IQ features.

FIGURE 12.3: Results of 5-fold cross validation with $k$ features selected by ANOVA.

The figures show that the performance can actually be increased by reducing the number of features. If the IQ score is included as a selection feature, around 120 classification features seems to provide the best results—and is significantly more efficient than using all 264 features. When not using the IQ score, the optimal number of features is somewhat higher, around 150 to 200.

Other than the optimal performance, this figure shows that the number of features can be reasonably reduced without a significant drop in performance.

Table 12.4 shows the ten most influential, and ten least influential features as selected by ANOVA.

| Feature | F-Value |
|---|---|
| rh_parsopercularis_CT | 33.4173 |
| rh_lingual_CT | 32.6225 |
| lh_phcg_CT | 31.7673 |
| rh_rostralmiddlefrontal_-CT | 30.2867 |
| rh_lateralorbitofrontal_-CT | 30.1568 |
| lh_temporal_CT | 29.7408 |
| lh_fusiform_CT | 29.5005 |
| rh_frontal_CT | 28.9450 |
| lh_superiorfrontal_CT | 28.8622 |
| rh_insula_CT | 28.8752 |

(A) Top 10 most relevant features without IQ features (and thus more instances).

| Feature | F-Value |
|---|---|
| rh_frontalpole_CT | 0.0000 |
| lh_pericalcarine_VOL | 0.0000 |
| rh_paracentral_area | 0.0000 |
| lh_posteriorcingulate_VOL | 0.0002 |
| lh_accumbens_VOL | 0.0003 |
| lh_supramarginal_area | 0.0003 |
| lh_frontal_pial_area | 0.0012 |
| rh_posteriorcingulate_VOL | 0.0024 |
| lh_caudalanterior- cingu-late_area | 0.0027 |
| lh_cingulate_pial_area | 0.0043 |

(B) Top 10 least relevant features without IQ features (and thus more instances).

| Feature | F-Value |
|---|---|
| lh_parstriangularis_CT | 25.70 |
| rh_lateralorbitofrontal_CT | 24.43 |
| lh_phcg_CT | 23.90 |
| rh_parsopercularis_CT | 23.64 |
| lh_lateralorbitofrontal_CT | 23.59 |
| lh_superiorfrontal_CT | 22.93 |
| rh_rostralmiddlefrontal_CT | 22.92 |
| lh_inferiortemporal_CT | 22.13 |
| lh_middletemporal_CT | 21.64 |
| lh_temporal_CT | 21.53 |

(C) Top 10 most relevant features with IQ features.

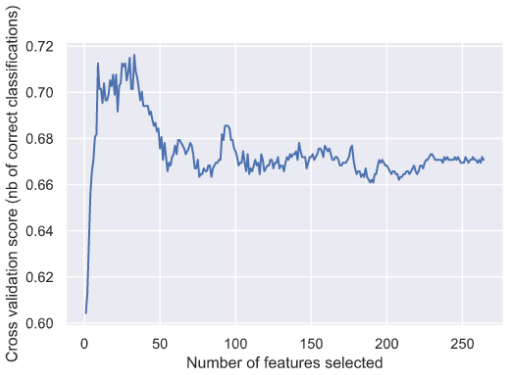
| Feature | F-Value |
|---|---|
| lh_middletemporal_area | 0.0000 |
| rh_temporalpole_VOL | 0.0000 |
| lh_bankssts_VOL | 0.0001 |
| rh_posteriorcingulate_VOL | 0.0005 |
| lh_insula_VOL | 0.0008 |
| cerebralWhiteMatter_VOL | 0.0010 |
| lh_amygdala_VOL | 0.0021 |
| rh_phcg_VOL | 0.0023 |
| lh_fusiform_area | 0.0024 |
| rh_cingulate_pial_area | 0.0029 |

(D) Top 10 least relevant features with IQ features.

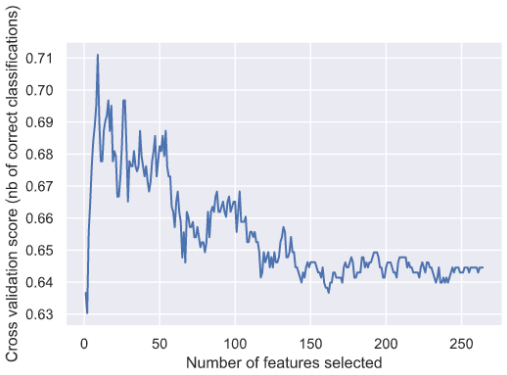
TABLE 12.4: The most and least relevant features, as determined by ANOVA. All features are suffixed by `_freesurfer`.

**Recursive Feature Elimination**

We can use recursive feature elimination to eliminate features [26], using the corresponding classifier—and thus the selection of features is dependent on the classifier. It selects features based on the cross-validated results. This method does not work with RBF kernels and thus we show the results only for the logistic regression model (figure 12.4). To accompany the experiments, default parameters are used.




(A) Recursive feature elimination results with the Logistic Regression model, with the instances taken into account with missing values for IQ.




(B) Recursive feature elimination results with the Logistic Regression model, with the instances removed with missing values for IQ.

FIGURE 12.4: Recursive feature elimination results.

The optimal number of features found is respectively 33 and 9. Table 12.5 shows those features.

**Volume vs Cortical Thickness vs Area**

The dataset contains three kinds of features: volume features, cortical thickness features and area features. To get an idea of which of these feature sets can be the most influential, we trained the classifiers on each of these feature subsets. The results are shown in table 12.6.

| Support Vector Machine (rbf) | |
|---|---|
| Feature set | Accuracy |
| Volume | 0.704 |
| Cortical Thickness | 0.634 |
| Area | 0.580 |
| **Logistic Regression** | |
| Feature set | Accuracy |
| Volume | 0.698 |
| Cortical Thickness | 0.643 |
| Area | 0.576 |

TABLE 12.6: Comparing the subsets of features.

The results show that there is a significant difference between these feature sets, with just Volume features performing around as well as the entire feature set, and Area features performing much worse.

## 12.3 Dimensionality Reduction for the Experiments

For the experiments, we will use the feature selection as just described. We reduce the number of features as mentioned to improve the performance of the classifier, as well as reduce the complexity. We do this with the Recursive Feature Elimination methods used previously for the Logistic Regression classifier, and with ANOVA for

| Feature |
|---|
| lh_thalamus_VOL |
| lh_hippocampus_VOL |
| rh_thalamus_VOL |
| subcortgray_VOL |
| lh_inferiortemporal_CT |
| lh_middletemporal_CT |
| lh_parahippocampal_CT |
| lh_superiortemporal_CT |
| rh_inferiortemporal_CT |
| rh_lateralorbitofrontal_CT |
| rh_lingual_CT |
| rh_parsopercularis_CT |
| lh_inferiortemporal_VOL |
| lh_middletemporal_VOL |
| lh_parahippocampal_VOL |
| lh_superiortemporal_VOL |
| lh_insula_VOL |
| rh_fusiform_VOL |
| rh_inferiortemporal_VOL |
| rh_lateralorbitofrontal_VOL |
| rh_supramarginal_VOL |
| lh_inferiortemporal_area |
| lh_middletemporal_area |
| lh_parahippocampal_area |
| lh_superiortemporal_area |
| lh_insula_area |
| rh_fusiform_area |
| rh_inferiortemporal_area |
| rh_lateralorbitofrontal_area |
| rh_posteriorcingulate_area |
| rh_superiortemporal_area |
| lh_frontal_pial_area |
| rh_cingulate_pial_area |

(A) Feature set found without the IQ selection feature.

| Feature |
|---|
| lh_hippocampus_VOL |
| rh_thalamus_VOL |
| subcortgray_VOL |
| lh_inferiortemporal_CT |
| rh_lateralorbitofrontal_CT |
| lh_inferiortemporal_VOL |
| rh_lateralorbitofrontal_VOL |
| lh_inferiortemporal_area |
| rh_lateralorbitofrontal_area |

(B) Feature set found with the IQ selection feature.

TABLE 12.5: Optimal feature sets found. All features are suffixed by `_freesurfer`.

the Support Vector Machine classifier (as the Recursive Feature Elimination method is not available with the RBF kernel). For this, we use the top 10% of features as also in [46]. This set of classification features and the known labels are used during the *evaluation step*, where we retrain the classifier model given all sampled instances.

## 12.4 Experimental Results

For this section, we will consider the approaches that we considered in chapter 11 to be most appropriate for this problem. These approaches are listed in table 12.7.

| Approach | Parameters |
|---|---|
| Active Classification Feature Acquisition—AVID | $B = 10, \lambda = 1$ |
| Active Classification Feature Acquisition—Prob | $b =$ auto |
| Expectation Maximization—AVID | n_clusters $= 7$, $B = 10$ |
| Expectation Maximization—Prob | n_clusters $= 7$, $b =$ auto |
| Random | – |

TABLE 12.7: Comparing the subsets of features.

The learning curves are displayed in figures 12.5 (using the accuracy score) and 12.6 (using the F1-score).

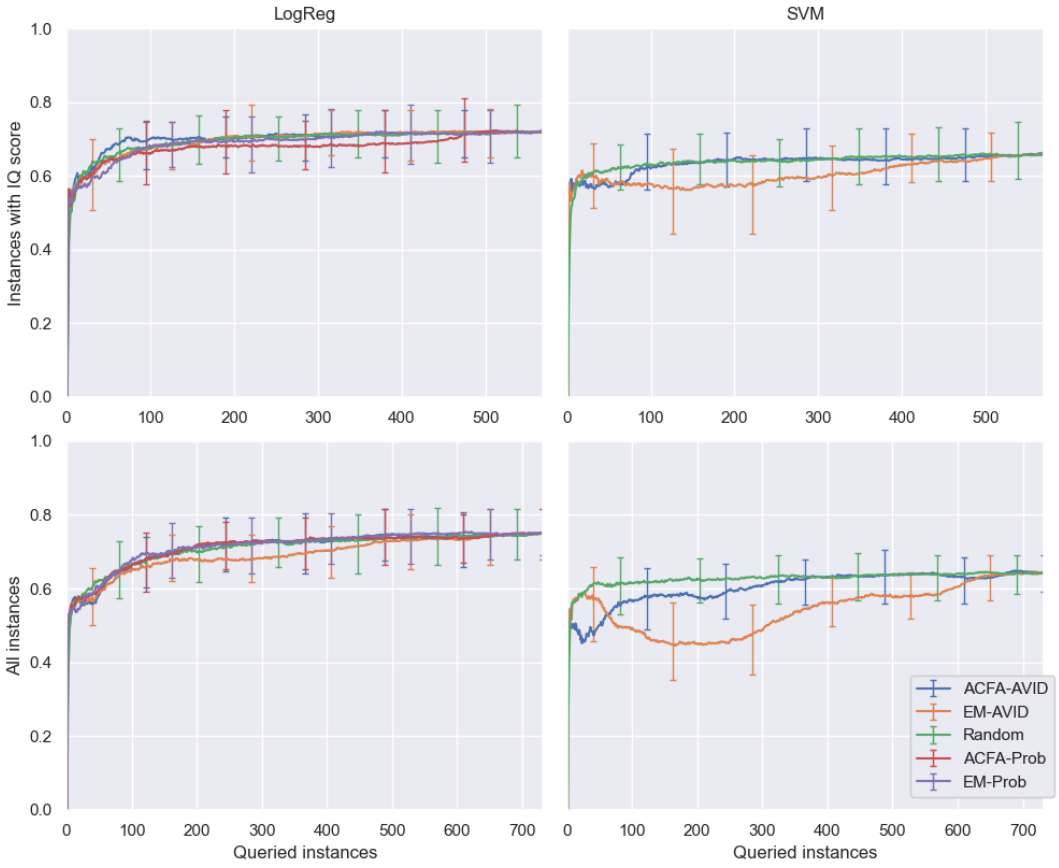



FIGURE 12.5: The learning curves of the experiments, using the accuracy score.

The learning curves displaying the accuracy score by number of sampled instances seems to display no advantage of any of the methods over the random baseline. We can note some things though, when looking at these curves: most notably, the Expectation Maximization-based approach using the AVID utility method seems to generally give subpar results: only in the combination of using the Logistic Regression classifier model on the IQ-based dataset does its learning curve not noticably decrease upon the random curve. This is especially noticable with the Support Vector Machine-based learning curves.

In the one configuration just mentioned where EM-AVID performs up to par, the Acitve Classification Features Acqusition approach using the Probability-based

utility method performs subpar: up until when most of the instances are queried its accuracy score is lower than that with random sampling.

Finally, the Active Classification Feature Acqusition approach using the AVID utility method seems to perform better than the random baseline, only with the Logistic Regression model. It has a higher accuracy score at some points in the learning curve, while never having a lower one.

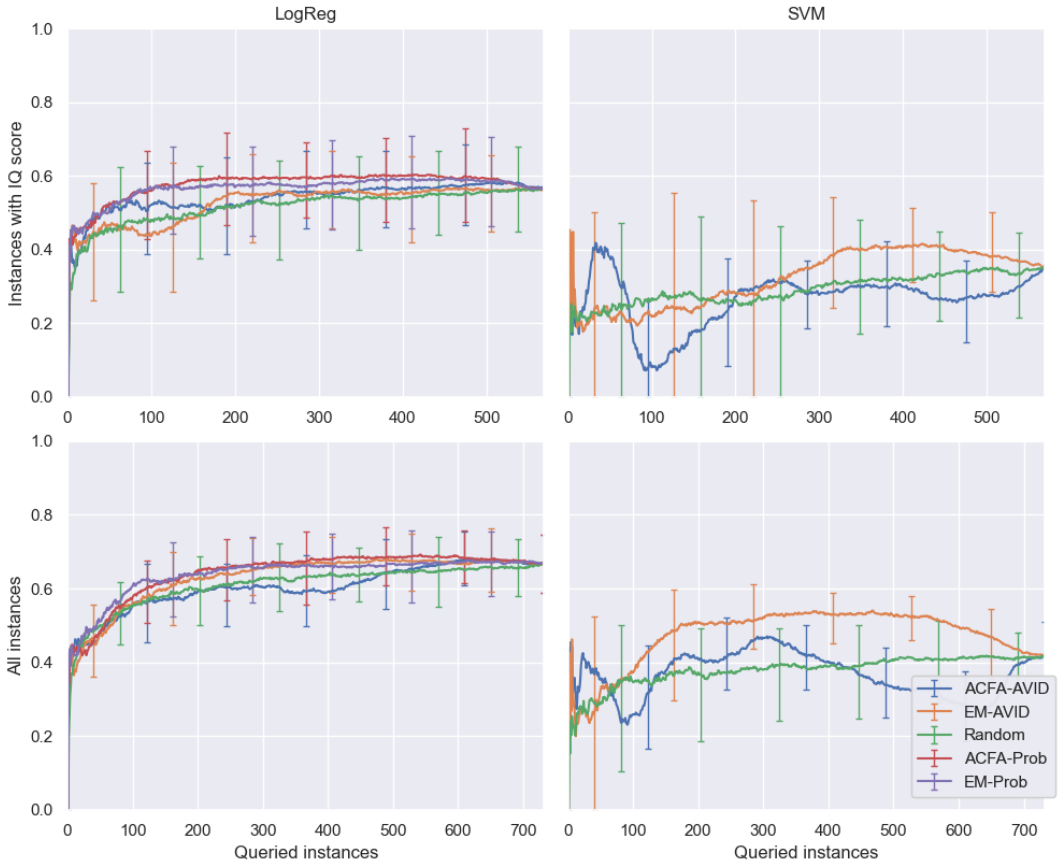



FIGURE 12.6: The learning curves of the experiments, using the F1-score.

When considering the learning curves displaying the F1-score, the results are significantly different and suggest notable advantages of using an informed selection approach instead of the random baseline. This difference is in all likelihood due to the imbalanced label distribution in the dataset, which the F1-score is able to more accurately display (and more similar to what the approaches aim to optimize than the accuracy score).

First, we will consider the differences with the Logistic Regression classifier model (as this supports all approaches, while having less erratic results). In both configurations of the dataset, three out of the four approaches are able to consistently improve over the random baseline for the entire process of the learning curve. The fourth approach in both cases (respectively EM-AVID and ACFA-AVID) improves upon the random baseline given a reasonable number of sampled instances—depending on the expected use case.

In both configurations of the dataset, the two approaches using the Probability-based utility method consistently improve upon the random baseline. This suggests

that if these approaches would have been used in the real-world scenario, the number of needed instances could have been reduced or the performance with the same number of instances could have been improved.

This is especially support by looking at the end of the learning curve. In the case of the learning curve of the random baseline, it consistently increases until it reaches its best performance score at the end—as makes sense, as more information is obtained. However, for these two approaches, the learning curve is able to reach its most optimal performance much earlier, and even decreases near the end. This means that these approaches are able to *filter out the instances with negative usefulness*. These are instances that do not make sense with the rest of the dataset or the model, and thus we can say that these approaches are able to select instances in a useful manner.

As for the results with the Support Vector Machine classifier, these are more erratic—and missing the two Probability-based approaches which perform best. However, we can still note some important information. The ACFA-AVID method provides the most erratic results, and mostly provides a more unreliable learning curve than the random baseline. However, the EM-AVID method provides more consistent and useful results, especially with the complete dataset. As can be seen, it very significantly improves upon the random baseline and displays the same pattern as just mentioned where it drops in the end of the curve, but even more significantly. This again means it is able to select what instances are and are not useful in a good manner.

To aid these learning curves, we can also analyze the statistics as mentioned in chapter 7. Again, these are separated by classifier model, configuration of the dataset and performance metric.

The results considering the accuracy score are shown in table 12.8, and they broadly confirm what we were already able to conclude from the learning curves.

Broadly speaking, when looking at the accuracy score, there is no significant change in performance when considering the mentioned approaches compared to the random baseline. The exception here is the EM-Prob method, which provides consistent results especially with the complete dataset. The ACFA-AVID method also provides solid results, most notably with the IQ-based dataset.

The more significant results can again be seen in the results considering the F1-score, as shown in table 12.9. These statistics mostly confirm and conform to the conclusions we drew earlier from the learning curves. Considering the Logistic Regresion classifier model, all four approaches are able to improve upon every single statistic for both dataset configuration (with two exceptions where it is equal). Here is where we can see the potential of using these approaches.

The Probability-based utility methods seem to be the most consistent, when considering the F1@80% and F1@90% statistics, with consistent improvement. This is not surprising when looking at figure 12.6. Similarly, the EM-AVID method has very significant improvements for the Support Vector Machine classifier model.

All in all, we can consider these approaches to be very useful in this experimentation setting, if we are to consider the F1-score performance measure. This allows for consistent and significant improvement upon the random sampling baseline. The most consistent seem to be the Probability-based utility method-based approaches.

However, when considering the accuracy score only, there is no significant improvement. We thus have to consider our evaluation of our model, as well as the practical implications of both the improved F1-scoring as well as the implementation costs.

TABLE 12.8: All numerical results for the considered approaches, using the accuracy score Improvements upon random selections are bolded.

| | | Logistic Regression | | Support Vector Machine | |
|---|---|---|---|---|---|
| | | Instances with IQ score | All instances | Instances with IQ score | All instances |
| ACFA-AVID | DUR | **0.46** | **0.85** | 1.10 | 1.88 |
| | AUC | **0.70** | 0.71 | 0.63 | 0.60 |
| | Acc@80% | **0.72** | 0.74 | 0.65 | 0.63 |
| | Acc@90% | 0.72 | 0.74 | 0.65 | 0.63 |
| ACFA-Prob | DUR | 2.99 | **0.81** | | |
| | AUC | 0.68 | 0.71 | | |
| | Acc@80% | 0.70 | 0.74 | | |
| | Acc@90% | 0.72 | **0.74** | | |
| EM-AVID | DUR | 1.13 | 1.89 | 3.20 | 3.46 |
| | AUC | 0.69 | 0.69 | 0.60 | 0.54 |
| | Acc@80% | **0.72** | 0.74 | 0.64 | 0.60 |
| | Acc@90% | 0.72 | 0.74 | 0.65 | 0.64 |
| EM-Prob | DUR | 1.03 | **0.78** | | |
| | AUC | 0.69 | 0.71 | | |
| | Acc@80% | **0.72** | **0.75** | | |
| | Acc@90% | 0.72 | **0.75** | | |
| Random | DUR | 1.00 | 1.00 | 1.00 | 1.00 |
| | AUC | 0.69 | 0.71 | 0.64 | 0.63 |
| | Acc@80% | 0.71 | 0.74 | 0.66 | 0.64 |
| | Acc@90% | 0.72 | 0.74 | 0.66 | 0.64 |

TABLE 12.9: All numerical results for the considered approaches, using the F1-score Improvements upon random selections are bolded.

| | | Logistic Regression | | Support Vector Machine | |
|---|---|---|---|---|---|
| | | Instances with IQ score | All instances | Instances with IQ score | All instances |
| ACFA-AVID | DUR | **0.26** | 1.00 | **0.01** | **0.01** |
| | AUC | **0.54** | **0.69** | 0.26 | 0.37 |
| | F1@80% | **0.57** | **0.67** | 0.26 | 0.29 |
| | F1@90% | **0.58** | **0.67** | 0.27 | 0.30 |
| ACFA-Prob | DUR | **0.24** | **0.57** | | |
| | AUC | **0.57** | **0.64** | | |
| | F1@80% | **0.59** | **0.68** | | |
| | F1@90% | **0.59** | **0.68** | | |
| EM-AVID | DUR | **0.91** | **0.69** | **0.01** | **0.02** |
| | AUC | **0.52** | **0.63** | **0.32** | **0.47** |
| | F1@80% | **0.57** | **0.67** | **0.40** | **0.51** |
| | F1@90% | 0.56 | **0.68** | **0.38** | **0.46** |
| EM-Prob | DUR | **0.30** | **0.44** | | |
| | AUC | **0.56** | **0.64** | | |
| | F1@80% | **0.59** | **0.67** | | |
| | F1@90% | **0.59** | **0.67** | | |
| Random | DUR | 1.00 | 1.00 | 1.00 | 1.00 |
| | AUC | 0.51 | 0.61 | 0.29 | 0.38 |
| | F1@80% | 0.55 | 0.65 | 0.34 | 0.41 |
| | F1@90% | 0.56 | 0.66 | 0.35 | 0.41 |

# Part IV

# Conclusion

# Chapter 13

# Discussion

As seen in the results in part III, there is a notable difference between results. In this chapter, we will briefly discuss the implications of these results and any notable correlations that imply relevant information.

First, we can say that the results are dependent on the dataset. When looking at the results and comparing the results on the three UCI benchmark datasets, it can be concluded that there is no 'best' approach—as one would expect: *there ain't no such thing as free lunch* [68]. However, for the first dataset (the Breast Cancer dataset [49]), if we base our conclusions on the Data Utilization Rate metric, the most efficient approach seems to be the random selection baseline. This seems to imply that for this dataset, the complexities of the features and the model are more complicated, or missing, and thus not possible to improve upon when compared to random selection. However, two things should be noted here:

1. This problem adds another layer to the no free lunch theorem, by definition of being an active learning problem. As mentioned previously, the Data Utilization Rate values for our approaches are worse when compared to random selection for the Breast Cancer dataset. However, this Data Utilization Rate is just one method of converting the learning curve into a single definable value. In truth, the learning curve is too complex to compress into a single quantifiable value, and when looking at the learning curve it becomes clear that some of our approaches actually improve upon the random selection baseline after a disappointing startup phase. Whereas the no free lunch theorem mentions that there is no optimization model that is better for each and every instantiation of the problem (or dataset), for our problem we can even say that there is no optimization model that is better for the same problem! Combining these two factors leads us to a difficult comparison between methods, unless the learning curves clearly improve upon the other at every stage of the curve.

2. There is actually one great advantage of the random sampling approach. If it is properly implemented, the random selection is *unbiased*. This should result in any subset at any size is averaged over multiple runs —or expected to be— a representative sample of the entire dataset. Whereas any approach that is more deterministic, but does not capture the complexity of the problem or is overengineered, introduces bias. This bias *can* help, if this is a bias towards more useful samples. However, if the approach is not able to do as such, a bias is introduced that only removes the representativeness of any sample. We can then simply expect a method that performs worse than random sampling.

Fortunately, although it is hard to conclude which approaches generally perform the best due to these two complexities (the difference in performance by dataset as mentioned in the no free lunch theorem and the lossy compression of the learning

curves to a single value), it is a lot more straightforward to say which approaches perform worse than the others. The Active Pseudo-Class Selection approach in general, and any approach using the Goal-Oriented Data Acquisition utility method, perform poorly in most settings

When comparing these approaches to the ones that perform in general better—the approaches using the Acquisition based on Variance of Imputed Data or the Probability-based utility method—we can see an additional layer of complexity. This implies perhaps that this problem is difficult to find much information for, and/or that these approaches might actually be overly complex for the problem. Possibly the correlations are generally not strong to consider an approach with multiple steps.

Consider the approaches with good results: the AVID and probability-based utility methods. The first only estimates the classification features, and there the exact value *does not even matter*. It focuses on the variance between repeated estimations, which is a nice representation of how much we know or can expect to know about the missing information—and as such how useful it would be to query. The probability-based utility method estimates the classification features directly and uses the estimated feature values, but does not add a second step in the approach or retrain the model. Instead, it quantifies the uncertainty about the classification of these estimated features, as such similarly to the AVID utility method we are focusing on uncertainty of information.

Comparing this to the approaches with lesser results—Active Pseudo-Class Selection and the GODA utility method—we can see an additional layer of complexity in a second step in the approach.

For the GODA utility method, we again estimate the classification features, with a second step in the approach of retraining the classification model (for each seperate instance and its estimated classification features). We then evaluate the retrained model using log or hinge loss. This has an obvious increased level of complexity, and introduces some assumptions that impose larger restrictions than just sampling uncertainty. If the estimation of the classification features is subpar, the estimation of the utility is by definition subpar as well. The method requires on small margins of increase in performance as well, considering only a single sample added to the training set.

For the Active Pseudo-Class Selection approach, we use a notably different method to the others, but it is clear that this introduces an additional layer of complexity as well. We add a first step of constructing pseudo-classes from the existing data, after which we use existing Active Class Selection approaches. By definition, there is an added additional layer as it is this construction of pseudo-classes. The results showed that regardless of the Active Class Selection method used, the performance of this approach was subpar. The problem thus most likely lies within the pseudo-class construction, which does not provide a proper mapping from our problem to the the Active Class Selection problem.

Although we can conclude that these two approaches introduce too much unnecessary complexity to provide solid results, they do not perform just worse than the other approaches but generally even worse than random selection. This is most likely due to the introduction of a *sampling bias*. If we can say that the instances sampled are not due to proper guided selection that improves performance, we are thus querying instances based on a biased selection mechanism. This bias means we are not constructing a less representative training set and thus reduce performance. In this case, random selection performs better as it introduces no *new* biases: only the ones that already exist in the pool.

Hyperparameters should also be noted: several of these approaches contain parameters that are to be set to a specific value for the approach to function. In conventional algorithms and machine learning settings, these hyperparameters can be tuned: by using the known data and retraining the model with different hyperparameters we can find the optimal setting. However, this is not possible for any approach mentioned in this thesis, nor is it a trivial problem to introduce hyperparameter tuning for any of these approaches or this problem setting in general. Because the hyperparameter tuning influences the selection of new instances and the entire experiment process, we simply can not tune them in a real-world scenario. If any configuration of settings is to be properly evaluated, the experiment in itself must be performed.

This reduces us to simply configuring values for these hyperparameters, and hoping they are appropriate. We can of course make educated guesses, based on previous iterations on different datasets and different problem settings. If perhaps, $n = 5$ generally works well, and $n = 12$ generally performs subpar, we can make some conclusions. This is not the case though as far as we have seen currently: each dataset performed optimally with different hyperparameter settings.

This problem causes any approach that requires hyperparameters to be suboptimal for our problem setting. This is especially relevant for the Active Pseudo-Class Selection approach, but also in lesser relevance for the Expectation Maximization-based approach (where we are able to tune the hyperparameters, but which was empirically shown to give worse results than manual setting). Any future approaches would thus also be wise to either not include any hyperparameters, or ones that are able to be properly manually selected.

# Chapter 14

# Conclusion

In this chapter, we will aim to concisely summarize the contents of the thesis: the problem, the approaches taken and the results achieved as well as their implications.

The problem is defined in chapter 1, and related to similar problems discussed in chapter 3. Although these problems are similar, they are in the end different problems. We used these similar but different problems for inspiration in designing several approaches aiming to obtain good results on this problem, as outlined in part II.

The following approaches were formulated:

- *Active Classification Feature Acquisition*: based on the Active Feature Acquistion problem. We use a regression or other imputation model to estimate the missing classification features, and define a utility function to quantize the expected value of the estimated features. The instances with the highest utility is then sampled. For more information on Active Feature Acquisition, refer to: [4, 19, 43, 44, 53, 54, 60, 71, 72].

- *Active Pseudo-Class Selection*: based on the Active Class Selection problem. We use a k-means clustering process, followed by a decision tree construction to create pseudo-classes. This simulates a new Active Class Selection problem, which we are then able to approach with any existing method for the Active Class Selection problem. For more information on Active Class Selection, refer to: [33, 40, 69].

- *Expectation Maximization*: based on the Expectation Maximization method for constructing Gaussian Mixture Models. Several soft clusters are used to model the distribution of the known data, which are then used to estimate the likelihood of each unknown instance belonging to what cluster. This is combined with the same utility functions as used in the Active Classification Feature Acquisition approach, which are determined for each soft cluster. The instance with the highest expected utility is then sampled. For more information on Expectation Maximization and Gaussian Mixture Models, refer to: [42].

These approaches were then evaluated by repeated experiments on several benchmark and synthetic datasets. These results showed differing results. It should be noted that these results were dataset-dependent, as well as approach-dependent.

Several approaches showed promising results: the Active Classification Feature Acquistion approach generally worked well with the utility methods AVID (which favors a high variance of the estimated features) and Probability-based (which favors a probability score dependent on the scoring funtion), as well as the Expectation Maximization approach with the same utility methods.

The Active Pseudo-Class Selection approach showed poor results, regardless of the Active Class Selection approach used. The Active Classification Feature Acquisition and Expectation Maximization approach also performed poorly with the GODA utility method (which retrains the classification model with the estimated features).

The Case Study, where we evaluated the most promising approaches on the dataset for schizophrenia classification, provided interesting results. When using the F1-score as with the evaluation of all previous experiments, the results were significantly improved—both when compared to the previous experiments as well as upon the random baseline. All four approaches evaluated provided solid results and improvements upon random sampling, with especially the two approaches using the Probability-based utility method providing consistently good results.

However, it should be noted that when using the accuracy score (as is often done in the domain of schizophrenia classification), these improvements were not really noticable or significant: most approaches performed similarly to the random sampling baseline. Considering the class imbalance and the F1-score being more appropriate for such suggests that it is more appropriate though.

# Chapter 15

# Future Work

This thesis is simply an introduction of the Active Selection of Classification Features problem with one application, and the consideration of several potential approaches for this problem. There is much potential future work left regarding this problem. The following list summarizes some of this potential future work, but is not meant to be an extensive list of all possible improvements.

- Improvement upon the existing approaches, or new approaches can be considered for this problem. The noted approaches in this thesis are ones that came to mind in an initial study, but potentially other and better approaches exist.
- All current approaches are defined for use with sequential querying, with the results of the previous query being known before selecting the next instance. This might not be a given truth in each setting, and a batch size or step size $\beta$ could be introduced.
- The current approaches are defined given the setting of pool-based sampling. Other potential active learning settings that could be implemented are data stream sampling, and query synthesis.
- Automated hyperparameter selection, or removal of hyperparameters from the approaches. As mentioned in 13 and seen in parts II and III, the selection of hyperparameters is a problem for problems such as this one due to the impossibility of straightforward hyperparameter tuning. Potentially this selection of hyperparameter for some of the approaches can be done automatically, or removed entirely. It should be noted that this is most relevant for the Active Pseudo-Class Selection problem which does not produce great results regardless of the hyperparameter instantiation, and thus is not a priority as of right now.
- Potential improvement on the results could be found by combining approaches in a similar manner to the Dual objective utility method, or similarly to ensemble learning such as XGBoost [15].

# Bibliography


[1] A Acock. "Working with missing data". In: *Family Science Review* 10.1 (1997), pp. 76–102.

[2] H. Akaike. "A new look at the statistical model identification". In: *IEEE Transactions on Automatic Control* 19.6 (1974), pp. 716–723.

[3] András Antos, Varun Grover, and Csaba Szepesvári. "Active Learning in Multi-armed Bandits". In: *Algorithmic Learning Theory*. Ed. by Yoav Freund et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 287–302. ISBN: 978-3-540-87987-9.

[4] J. Attenberg et al. "Selective data acquisition for machine learning". In: Jan. 2011, pp. 101–155.

[5] Hagai Attias. "A Variational Bayesian Framework for Graphical Models". In: *Adv. Neural Inf. Process. Syst* 12 (Sept. 2000).

[6] Joseph Berkson. "Application of the Logistic Function to Bio-Assay". In: *Journal of the American Statistical Association* 39.227 (1944), pp. 357–365. ISSN: 01621459. URL: http://www.jstor.org/stable/2280041.

[7] Mustafa Bilgic and Lise Getoor. "VOILA: Efficient Feature-value Acquisition for Classification." In: Jan. 2007, pp. 1225–1230.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[9] David Blei and Michael Jordan. "Variational inference for Dirichlet process mixtures". In: *Bayesian Analysis* 1 (Mar. 2006). DOI: 10.1214/06-BA104.

[10] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[11] Ricardo S. Cabral et al. "Matrix Completion for Multi-label Image Classification". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 190–198. URL: http://papers.nips.cc/paper/4419-matrix-completion-for-multi-label-image-classification.pdf.

[12] Lauriane Castin and Benoit Frénay. "clustering with decision trees: divisive and agglomerative approach." In: *ESANN*. 2018.

[13] Gavin Cawley. "Baseline Methods for Active Learning." In: *Journal of Machine Learning Research - Proceedings Track* 16 (Jan. 2011), pp. 47–57.

[14] S. Chakraborty et al. "Active Matrix Completion". In: *2013 IEEE 13th International Conference on Data Mining*. Dec. 2013, pp. 81–90. DOI: 10.1109/ICDM.2013.69.

[15] Tianqi Chen and Carlos Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Aug. 2016). DOI: 10.1145/2939672.2939785. URL: http://dx.doi.org/10.1145/2939672.2939785.

[16] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[17] Kun Deng et al. "New algorithms for budgeted learning". In: *Machine Learning* 90.1 (Jan. 2013), pp. 59–90. ISSN: 1573-0565. DOI: 10.1007/s10994-012-5299-2. URL: https://doi.org/10.1007/s10994-012-5299-2.

[18] Robert Detrano et al. "International application of a new probability algorithm for the diagnosis of coronary artery disease". In: *The American journal of cardiology* 64.5 (1989), pp. 304–310.

[19] Amit Dhurandhar and Karthik Sankaranarayanan. "Improving classification performance through selective instance completion". In: *Machine Learning* 100.2-3 (2015), pp. 425–447.

[20] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[21] Ronald A Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of eugenics* 7.2 (1936), pp. 179–188.

[22] M Forina, S Lanteri, C Armanino, et al. "Parvus-an extendible package for data exploration, classification and correlation, institute of pharmaceutical and food analysis and technologies, via brigata salerno, 16147 genoa, italy (1988)". In: *Av. Loss Av. O set Av. Hit-Rate* (1991).

[23] Milton Friedman. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance". In: *Journal of the American Statistical Association* 32.200 (1937), pp. 675–701. DOI: 10.1080/01621459.1937.10503522.

[24] Andrew Gelman and Jennifer Hill. "Missing-data imputation". In: *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. Cambridge University Press, 2006, pp. 529–544. DOI: 10.1017/CBO9780511790942.031.

[25] Russell Greiner, Adam J. Grove, and Dan Roth. "Learning Cost-sensitive Active Classifiers". In: *Artif. Intell.* 139.2 (Aug. 2002), pp. 137–174. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(02)00209-6. URL: http://dx.doi.org/10.1016/S0004-3702(02)00209-6.

[26] Isabelle Guyon et al. "Gene Selection for Cancer Classification Using Support Vector Machines". In: *Machine Learning* 46 (Jan. 2002), pp. 389–422. DOI: 10.1023/A:1012487302797.

[27] Arthur E Hoerl and Robert W Kennard. "Ridge regression: Biased estimation for nonorthogonal problems". In: *Technometrics* 12.1 (1970), pp. 55–67.

[28] Sheng-Jun Huang et al. "Active Feature Acquisition with Supervised Matrix Completion". In: *CoRR* abs/1802.05380 (2018). arXiv: 1802.05380. URL: http://arxiv.org/abs/1802.05380.

[29] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[30] Pallika Kanani and Prem Melville. "Prediction-time Active Feature-value Acquisition for Cost-Effective Customer Targeting". In: (Jan. 2008).

[31] Daniel Kottke et al. "Challenges of reliable, realistic and comparable active learning evaluation". In: *Proceedings of the Workshop and Tutorial on Interactive Adaptive Learning*. 2017, pp. 2–14.

[32] Daniel Kottke et al. "Multi-class probabilistic active learning". In: *Proceedings of the Twenty-second European Conference on Artificial Intelligence*. IOS Press. 2016, pp. 586–594.

[33] Daniel Kottke et al. "Probabilistic Active Learning for Active Class Selection". In: *Proc. of the NIPS Workshop on the Future of Interactive Learning Machines*. 2016.

[34] Georg Krempl, Daniel Kottke, and Vincent Lemaire. "Optimised probabilistic active learning (OPAL)". In: *Machine Learning* 100.2-3 (2015), pp. 449–476.

[35] Georg Krempl, Daniel Kottke, and Myra Spiliopoulou. "Probabilistic active learning: Towards combining versatility, optimality and efficiency". In: *International Conference on Discovery Science*. Springer. 2014, pp. 168–179.

[36] Charles X. Ling et al. "Decision Trees with Minimal Costs". In: *Proceedings of the Twenty-first International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: ACM, 2004, pp. 69–. ISBN: 1-58113-838-5. DOI: 10.1145/1015330.1015369. URL: http://doi.acm.org/10.1145/1015330.1015369.

[37] Roderick J A Little and Donald B Rubin. *Statistical Analysis with Missing Data*. New York, NY, USA: John Wiley & Sons, Inc., 1986. ISBN: 0-471-80254-9.

[38] B. Liu, Y. Xia, and P.S. Yu. "Clustering Via Decision Tree Construction". In: *Foundations and Advances in Data Mining*. Ed. by Wesley Chu and Tsau Young Lin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 97–124. ISBN: 978-3-540-32393-8. DOI: 10.1007/11362197_5. URL: https://doi.org/10.1007/11362197_5.

[39] Daniel J. Lizotte, Omid Madani, and Russell Greiner. *Budgeted Learning of Naive-Bayes Classifiers*. 2012. arXiv: 1212.2472 [cs.LG].

[40] R. Lomasky et al. "Active Class Selection". In: *Machine Learning: ECML 2007*. Ed. by Joost N. Kok et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 640–647. ISBN: 978-3-540-74958-5.

[41] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[42] G. J. McLachlan and D. Peel. *Finite mixture models*. New York: Wiley Series in Probability and Statistics, 2000.

[43] P. Melville et al. "Active feature-value acquisition for classifier induction". In: *Fourth IEEE International Conference on Data Mining (ICDM'04)*. Nov. 2004, pp. 483–486. DOI: 10.1109/ICDM.2004.10075.

[44] P. Melville et al. "An expected utility approach to active feature-value acquisition". In: Dec. 2005. ISBN: 0-7695-2278-5. DOI: 10.1109/ICDM.2005.23.

[45] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN: 0262018020, 9780262018029.

[46] Mireille Nieuwenhuis et al. "Classification of schizophrenia patients and healthy controls from structural MRI scans in two large independent samples". In: *Neuroimage* 61.3 (2012), pp. 606–612.

[47] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006.

[48] C. Parker. "An Analysis of Performance Measures for Binary Classifiers". In: *2011 IEEE 11th International Conference on Data Mining*. Dec. 2011, pp. 517–526. DOI: 10.1109/ICDM.2011.21.

[49] Miguel Patrício et al. "Using Resistin, glucose, age and BMI to predict the presence of breast cancer". In: *BMC Cancer* 18 (Dec. 2018). DOI: 10.1186/s12885-017-3877-1.

[50] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[51] Airel Pérez-Suárez, José F. Martínez-Trinidad, and Jesús A. Carrasco-Ochoa. "A review of conceptual clustering algorithms". In: *Artificial Intelligence Review* 52.2 (Aug. 2019), pp. 1267–1296. ISSN: 1573-7462. DOI: 10.1007/s10462-018-9627-1. URL: https://doi.org/10.1007/s10462-018-9627-1.

[52] Foster J. Provost, Prem Melville, and Maytal Saar-Tsechansky. "Data acquisition and cost-effective predictive modeling: targeting offers for electronic commerce". In: *ICEC*. 2007.

[53] Maytal Saar-Tsechansky, Prem Melville, and Foster Provost. "Active Feature-Value Acquisition". In: *Management Science* 55.4 (2009), pp. 664–684. DOI: 10.1287/mnsc.1080.0952. URL: https://doi.org/10.1287/mnsc.1080.0952.

[54] Karthik Sankaranarayanan and Amit Dhurandhar. "Intelligently querying incomplete instances for improving classification performance". In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM. 2013, pp. 2169–2178.

[55] Joseph L Schafer and Maren K Olsen. "Multiple imputation for multivariate missing-data problems: A data analyst's perspective". In: *Multivariate behavioral research* 33.4 (1998), pp. 545–571.

[56] Gideon Schwarz. "Estimating the Dimension of a Model". In: *Ann. Statist.* 6.2 (Mar. 1978), pp. 461–464. DOI: 10.1214/aos/1176344136. URL: https://doi.org/10.1214/aos/1176344136.

[57] Burr Settles. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison, 2009.

[58] Lars St, Svante Wold, et al. "Analysis of variance (ANOVA)". In: *Chemometrics and intelligent laboratory systems* 6.4 (1989), pp. 259–272.

[59] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[60] Mohamed Thahir, Tarun Sharma, and Madhavi Ganapathiraju. "An efficient heuristic method for active feature acquisition and its application to protein-protein interaction prediction". In: *BMC proceedings* 6 Suppl 7 (Nov. 2012), S2. DOI: 10.1186/1753-6561-6-S7-S2.

[61] Robert Tibshirani. "Regression Shrinkage and Selection Via the Lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288. DOI: 10.1111/j.2517-6161.1996.tb02080.x. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1996.tb02080.x. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1996.tb02080.x.

[62] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. "The NumPy array: a structure for efficient numerical computation". In: *Computing in Science & Engineering* 13.2 (2011), p. 22.

[63] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[64] S. Veeramachaneni and P. Avesani. "Active sampling for feature selection". In: *Third IEEE International Conference on Data Mining*. Nov. 2003, pp. 665–668. DOI: 10.1109/ICDM.2003.1251003.

[65] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: https://doi.org/10.1038/s41592-019-0686-2.

[66] Michael Waskom et al. *seaborn: v0.5.0 (November 2014)*. Version v0.5.0. Nov. 2014. DOI: 10.5281/zenodo.12710. URL: https://doi.org/10.5281/zenodo.12710.

[67] Frank Wilcoxon. "Individual Comparisons by Ranking Methods". In: *Biometrics Bulletin* 1.6 (1945), pp. 80–83. ISSN: 00994987. URL: http://www.jstor.org/stable/3001968.

[68] D. H. Wolpert and W. G. Macready. "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 67–82.

[69] Dongrui Wu and Thomas Parsons. "Active Class Selection for Arousal Classification". In: *Lecture Notes in Computer Science* 6075 (Oct. 2011), pp. 132–141. DOI: 10.1007/978-3-642-24571-8_14.

[70] Xiaoyong Chai et al. "Test-cost sensitive naive Bayes classification". In: *Fourth IEEE International Conference on Data Mining (ICDM'04)*. Nov. 2004, pp. 51–58. DOI: 10.1109/ICDM.2004.10092.

[71] Zhiqiang Zheng and Balaji Padmanabhan. "Selectively Acquiring Customer Information: A New Data Acquisition Problem and an Active Learning-Based Solution". In: *Management Science* 52 (May 2006), pp. 697–712. DOI: 10.1287/mnsc.1050.0488.

[72] Zhiqiang Zheng and B. Padmanabhan. "On active learning for data acquisition". In: *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* Dec. 2002, pp. 562–569. DOI: 10.1109/ICDM.2002.1184002.