Universiteit Utrecht

MASTER'S THESIS
ICA-5534941

Machine Learning for Classifying Certificate of Competency Applications

Author:
ing. G. de Vreugd

Internal Supervisors:
dr. A.J. Feelders
prof. dr. A.P.J.M. Siebes

External Supervisors:
Nicole Holla, MSc
Boaz Pat-El, MSc

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

in

Computing Science
Department of Information and Computing Sciences

December 14, 2016

"To find signals in data, we must learn to reduce the noise - not just the noise that resides in the data, but also the noise that resides in us. It is nearly impossible for noisy minds to perceive anything but noise in data."

Stephen Few

UTRECHT UNIVERSITY

Abstract

Faculty of Science
Department of Information and Computing Sciences

Master of Science

Machine Learning for Classifying Certificate of Competency Applications

by ing. G. de Vreugd

In this thesis we research the possibility of classifying Certificates of Competency. An application for such a certificate consists of selecting which competencies you want, providing various seagoing claims and providing various diplomas. We try to create a machine learning model that, based on all these knowledge, is able to correctly classify whether the application should be accepted or denied. Our main conclusion is that it is possible, as long as the labels are updated. There are multiple binary classifiers that can be used but only the boosted decision tree worked consistently on each problem we tested.

Acknowledgements

I would like to express my special thanks of gratitude to my supervisors dr. A.J. Feelders, and Boaz Pat-El for helping me throughout the project, and for providing useful feedback whenever I needed it. I would also like to thank Nicole Holla for the interesting internship meetings, and presentations. Lastly, I would like to thank Avanade at whom I was allowed to perform my graduation project.

Contents

List of Figures

List of Tables

List of Abbreviations

IT	**I**nformation **T**echnology
ML	**M**achine **L**earning
CoC	**C**ertificate **o**f **C**ompetency
AUC	**A**rea **U**nder the **C**urve
TP	**T**rue **P**ositive
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
TN	**T**rue **N**egative

Chapter 1

Introduction

Data are becoming the new raw material of business

Craig Mundie (Microsoft)

Crafting this raw material (quote above) into something useful is, however, still something that companies struggle with. Doing so successfully could reduce costs and improve the value of customer relationships (Thearling, 2016). The research proposed will aid yet another company in reaching this goal.

1.1 Background

Avanade is a global IT service provider, which introduces itself as “the leading provider of innovative digital services, business solutions and design-led experiences, delivered through the power of people and the Microsoft ecosystem” (Avanade, 2016). One of their Dutch clients is a company called Kiwa, which is "an independent highly qualified organization having certification as its core activity" (Kiwa, 2016).

One of Kiwa’s activities is providing Certificates of Competency (CoC). In order to approve an application, a Kiwa employee has to verify if the applicant has provided enough evidence in the form of experience and/or education. This process is done manually, because the number of different experience types and educations is too large to program in the system itself. The actual system, from applying to validating an application, is being developed by Avanade.

Every application that is approved or denied is stored, therefore Kiwa wonders if it is possible to predict the outcome of an application based on the history of handled applications, using machine learning. This, in essence, is the goal of this research project.

1.2 Application process

When applicants apply for a CoC, they have to choose between the following categories:

- Merchant navy / Sailing - Officer
- Merchant navy / Sailing - Mate
- Fishing

Secondly the applicants have to choose the nature of the request:

- First request
- Renewal, in case it is (about to be) expired
- Increment, in case they want more competencies
- Duplicate, in case the CoC was lost

After this, the applicants have to choose the competencies that they wish to have added on their CoC (see Appendix A). There are at most thirty-six different competencies to choose from in a single category and an applicant can select a combination of them. It is possible to select unusual combinations, for example selecting both *Master - No limitations* and *Master - All ships less than 3000 GT*. This, however, will not occur often and is easily corrected by the person that handles the request. It is also possible to select illegal combinations, for example choosing a master competency as a first request. If an applicant chooses an illegal combination, the application is denied and they will have to apply again. The costs will not be reimbursed.

The fourth step is to provide a medical claim. This can be either with or without vision and hearing. It depends on the competencies chosen, which of the medical claims should be provided. This is not checked automatically and providing the wrong claim will result in a request for more information.

Now, the applicant has to provide their sailing time. They have to provide the period the sailing time occurred in including duration in days. They also have to provide the position they had and the type of vessel (see Appendix B). With this sailing time they prove that they have the required experience for the requested CoC. Here they can also provide illegal combinations, for example by entering a duration longer than the selected period or by entering positions on the wrong type of vessel. However, the applicant has to provide proof for the experience as well and if that does not match the entered sailing time, the application will be denied or it will result in a request for more information.

Lastly, they have to provide their education. Here they have to select the institution and the education (see Appendix C), they also have to upload a certificate that proves it.

1.3 Handling process

Once an application is fully finished and paid for, it will be ready to be handled. The person that handles the application will first verify the requested competences and whether the nature of the request is correct. After this they will verify the medical claim and check if this claim is sufficient for the requested competences. Following up they will verify the sailing time and also check if the provided sailing time is sufficient. Lastly they will do the same for the education claims. When certain documents are unreadable, or if those documents do not meet the requirements, they will ask for more information. Once everything is correct, they will approve the application.

1.4 Scientific and social relevance

Automating parts of this process could potentially help Kiwa to speed up the handling times. This is beneficial to both Kiwa and the customers. For Kiwa it can reduce costs as it would require less man-hours to process the

applications while the customers receive their CoC faster. The model explained above can also be applied to other fields Kiwa is active in, for example to flight licenses. Therefor a positive result can help Kiwa and their customers in a much broader sense than CoCs alone.

For Avanade it is relevant, as when this research is successful it can be used as a showcase for other clients. This helps Avanade in their goal to develop more machine learning related projects for their clients.

From a scientific perspective it is relevant because this is yet another case that could show how computers could help us in our daily lives. More specifically this research could provide yet another way where machine learning can be effective. Also this research gives insight how the different machine learning techniques compare to each other for this specific problem.

1.5 Methodology

A much used methodology for data science is CRISP-DM , which was introduced in the late nineties, however, their official website is no longer online (Piatetsky, 2014). IBM introduced a follow-up for this methodology, called Foundational Methodology for Data Science (IBM Analytics, 2015).

We will only explain the IBM methodology; for more information about CRISP-DM see Wirth and Hipp, 2000.

1.5.1 IBM Foundational Data Science Methodology

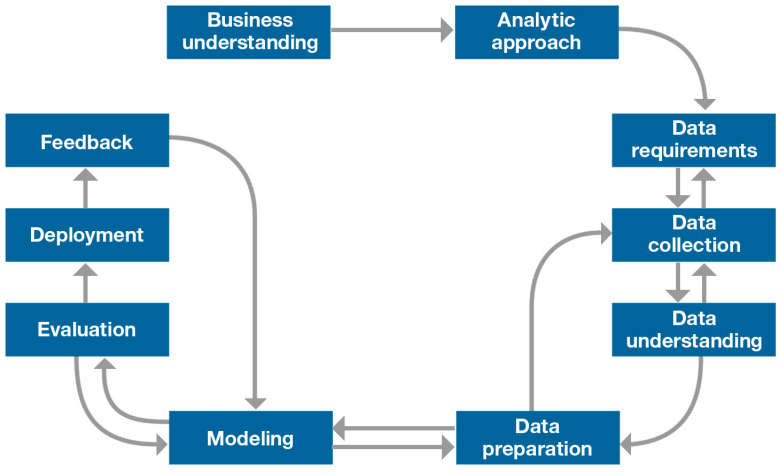


FIGURE 1.1: Process diagram showing the relationship between the different phases (Rollins, 2015).

This methodology is now taught by the Big Data University[1]. An overview of the phases is shown in Figure 1.1. The different phases are (IBM Analytics, 2015):

[1] http://bigdatauniversity.com/courses/data-science-methodology/

Business understanding

The initial phase is business understanding. This phase is all about understanding the actual problem. The problem must be defined, project objectives and solution requirements have to be stated.

Analytics Approach

After the problem is understood, it has to be converted into a data mining problem by selecting the proper (ML) techniques.

Data requirements

Before the actual data collection can start, its requirements have to be stated. These depend on the domain knowledge and the chosen techniques.

Data collection

Now we can start collecting all the data that is relevant to the problem, if we encounter gaps we might need to update the requirements and collect more data.

Data understanding

Phase five is about understanding the data. If we have gaps in understanding the data, we either need to get more domain knowledge or it might be necessary to collect more data (for example, to get more data points so it fits a normal distribution).

Data preparation

The data preparation phase is all about transforming the data into what we are going to feed the model. Here we do data cleaning and feature engineering.

Modeling

Here we select and apply ML techniques to the dataset we prepared so far.

Evaluation

Here the model is evaluated and checked against the business problem. If the result is insufficient we have to go back to the modeling phase.

Deployment

After a satisfactory model has been created, it has to be deployed.

Feedback

The last phase is the feedback phase. In this phase the deployed model is monitored and feedback about its performance and accuracy is provided to the data scientist. This feedback can be used to improve the model and to deploy a new version of it.

The methodology of IBM looks a lot like CRISP-DM, however it has a few important improvements. First of all, the business understanding and data related phases have been extended which makes it more clear. Secondly, the deployment phase is no longer an endpoint but it is part of the iterative cycle. This is important, as a data mining process is an ongoing task and after deployment it should not be ignored. Lastly, the new methodology makes it more clear that without understanding the business, there is no point in continuing and creating a model anyway. You have to know exactly what the problem is, instead of learning what the problem is, in an iterative way.

We will conduct this research according to the IBM methodology, with the exception of the Deployment and Feedback phases. This research will only focus on developing the model itself. Even though we will not structure this thesis explicitly to the phases of this methodology, you should be able to discover the different phases along the way.

1.6 Research questions

In this thesis we address the following research questions:
How could machine learning be used to successfully classify certificate of competence applications?

(a) Which ML techniques can be used to classify those applications?

(b) Which features should be engineered from the data?

(c) How do the different ML techniques compare to each other, in terms of classification performance?

The remainder of this thesis is structured as follows. Chapter 2 discusses the data. The applicable machine learning techniques are stated in Chapter 3, followed by the evaluation in Chapter 4. Lastly, Chapter 5 presents the conclusion of the research and also gives a brief discussion.

Chapter 2

Data

2.1 Data requirements

In order to properly classify the cases, we need to get the data of all previously handled applications. This data should include all the requested competencies, all the claims that were provided and some basic information like the outcome of the case. Kiwa disallowed the data to contain any sensitive information, including, but not limited to, names, addresses, and emails.

2.2 Data collection

These cases are currently stored in a [redacted] database. Figure 2.1 shows a simplified ER diagram containing the tables and columns involved.

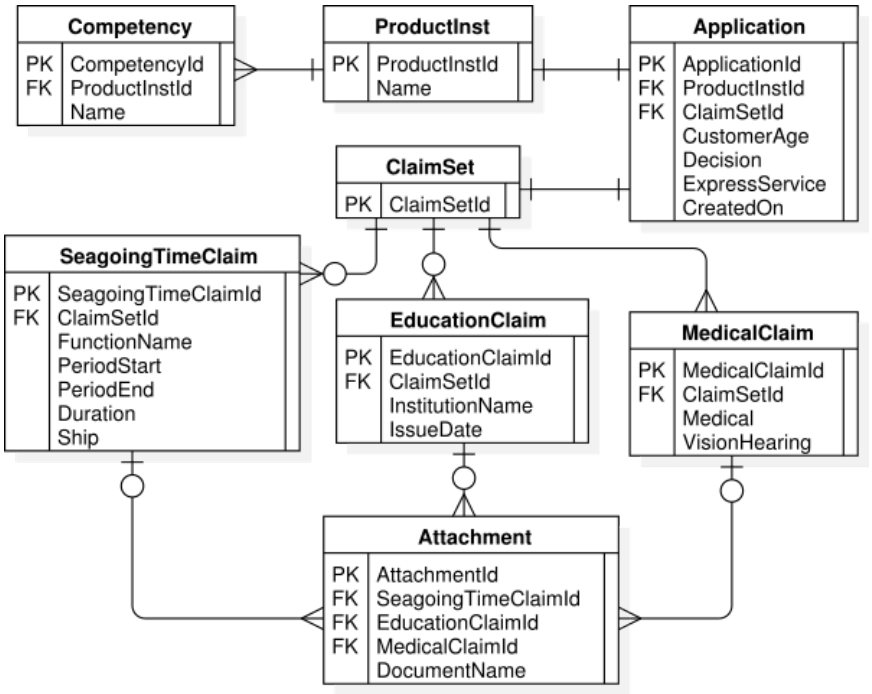


FIGURE 2.1: A simplified diagram that shows the different tables as represented inside the database.

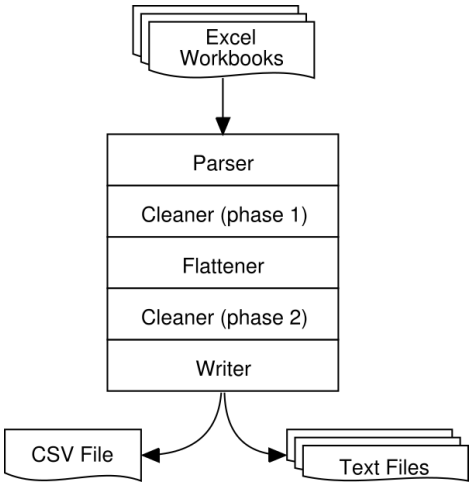


FIGURE 2.2: A simplified flow diagram that shows the structure of the console application.

To retrieve this information we have written seven queries (see Appendix D). Those queries were executed on a recent backup of the production database and exported to Excel Workbook (.xlsx) files, after which they were handed to us for future analysis.

2.3 Data preparation

The next step is to prepare this data for usage inside Azure Machine Learning, which means we have to flatten the relational data and we have to make sure to correct any issues in the data. To aid us in this process we have created a console application using C#. We will explain the basic elements of this program (as shown in Figure 2.2) one by one.

2.3.1 Parser

The first part of our program is a parser that parses the provided Excel Workbooks into an object oriented structure that closely represents the tables in the database. Most of this is pretty straightforward, except for two things:

Multiple Workbooks

We were provided with two Excel Workbooks because we did not get all the data needed the first time we requested it. This was our fault, since we forgot to include some queries. Therefore we had to support parsing multiple files and we had to make it configurable to select which sheet contains which data. The upside to this is, that if in the future an updated version of the data is given, it can be imported easily by simply updating this configuration.

TABLE 2.1: Data issue for cells containing multi-line strings.

ID	Accepted	Reason	Year
1	TRUE	This is fine	2010
2	TRUE	Some nice colors:	
a.	Blue		
b.	Red	2011	
3	FALSE	Fine again	2012

Multi-line strings

The largest issue we had, was the fact that multi-line strings were not stored inside a single cell but actually span multiple rows. This also caused the next columns of that particular row to be shifted. To complicate this issue even more: strings that contained lists had their bullets converted into single cells as well (see Table 2.1). To solve this issue, we detect whether the next cell is a null or not, since multi-line strings do not occur in the last column we know a line break occurred and we continue parsing on the next line. We also detect whether the cell is one of the known bulletins, if so, we move to the next cell.

2.3.2 Cleaner

The next part is where the cleaning happens. This actually happens in two phases: phase one is right after parsing and phase two is after flattening the data.

Phase one

The first phase is responsible for very basic cleaning that removes or fixes any data that could cause the flattening to fail. The operations that it performs are as follows:

- Remove all applications that do not point to a valid product.
- Remove all applications with seagoing claims that do not have a valid period start or period end.
- Remove all claims that do not have an attachment.
- Remove all claims that are not referenced by any of the applications.
- Fix all seagoing claims which have a reversed period start and end.

Phase two

The second phase is responsible for more advanced cleaning that removes or fixes any flattened data that cannot be used for classification as is. The operations that it performs are:

- Remove all applications that request a duplicate CoC.
  Duplicates cannot be classified properly as they are granted if you already own the same one, therefore it does not depend on provided claims.
- Calculate an estimated duration for seagoing claims that contain a duration of zero or one day(s).
  There are many people that do not bother filling in the correct duration when the time served spans a long period.

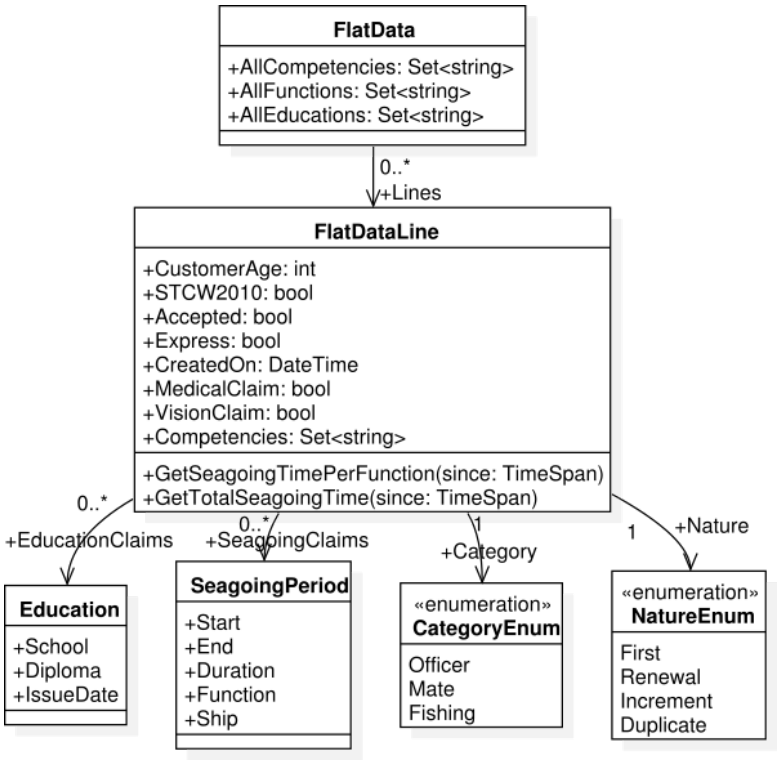


FIGURE 2.3: A class diagram that shows the flattened data structure.

2.3.3 Flattener

The flattener converts the relational data into a new data structure. As you can see in Figure 2.3; the data is still not flat. However, the current structure gives the writer everything it needs to print flat files. This flattening process is structured as follows:

Lookup tables

For each application, the flattener has to retrieve data from various tables. Since looping all of them to find the correct entries is a tedious process, we convert them to lookup tables first. The lookup tables are simple hash maps which allow for constant lookup times.

TABLE 2.2: Lookup tables constructed during flattening.

Table	Key	Value
`MedicalClaims`	`ClaimSetId`	`MedicalClaim[]`
`SeagoingClaims`	`ClaimSetId`	`SeagoingTimeClaim[]`
`EducationClaims`	`ClaimSetId`	`EducationClaim[]`
`EducationDocs`	`EducationClaimId`	`Attachment[]`
`Competencies`	`ProductId`	`Competency[]`

General information

While going through the applications, we can set most of the general information right away as this information is already present in the application table. The fields `STCW2010`[1], `Category`, and `Nature` are parsed from the product name.

Competencies

The selected competencies are retrieved using their lookup table, after which all competency names are added to the `FlatDataLine`. For future reference all competency names are also stored inside `AllCompetencies`.

Medical claims

The fields `MedicalClaim` and `VisionClaim` are set by retrieving the medical claims and checking each to see if either field is set. Multiple claims are possible if one claim contains the general medical information and the second one contains the vision/hearing information.

Education claims

The provided education claims are slightly harder to retrieve. First we retrieve the `EducationClaim` rows using the lookup table. This gives us the information for `School` and `IssueDate` directly. In order to retrieve the `Diploma` field, we lookup the attachments and retrieve the document name from the first one. For future reference the document names are also stored inside `AllEducations`.

Seagoing time claims

The seagoing time claims are also retrieved using the lookup table. For future reference all function names are stored inside `AllFunctions`.

2.3.4 Writer

The final component of our program is the writer. This component is able to generate the flattened data in two separate ways.

CSV File

The most important feature of the writer is the ability to generate a CSV file which can be used with Azure ML (or any other ML suite). In order to print the competencies it will create a feature for each possible competency as found in `AllCompetencies`. For each individual application it will check which competencies are set, and print these as `True` while keeping the remaining on `False`. In Chapter 2.4 we discuss how the seagoing time and education claims are printed.

[1] STCW2010 indicates that the application is subject to the updated legislation.

Simple text files

The second output method creates a single file for each application. To be more specific it creates files with the path:

```
[Competency]/[Accepted]/[ID].txt
```

In this file all the related data is printed vertically, which makes it easy to see all information of the application. This allows us to view all cases for a single competency and accept state in a very quick way. Chapter 2.5 goes deeper into this output method.

2.4 Feature engineering

In order to print the education claims and seagoing periods, we had to think about which information is actually useful from a ML perspective. According to STCW 95, the Dutch seafarers law: Wet Zeevarenden, and Kiwa[2] the following information can be used for determining whether an application is correct or not:

- Having a certain degree.
- How long ago a certain degree has been issued.
- Total seagoing time (for certain functions).
- Total seagoing time (for certain functions) in the last six months.
- Total seagoing time (for certain functions) in the last five years.

Using this information we decided to generate the following features:

Education claims

For each diploma we print the number of days since the issue date. For diplomas the applicant does not possess, we print `int.MaxValue`. The name of the school is discarded as it is irrelevant for the application.

Seagoing time claims

For seagoing claims we print the total seagoing time and also the total seagoing time for each function as found in `AllFunctions`. This information is also printed for the last 200 days and the last 1850 days. The feature names in those two are suffixed with `_200` and `_1850` to distinguish them from the regular ones. These features are used to cover the seagoing time in the last six months and the last five years.

We added a bit more days to these periods because we saw that many accepted applications provided their seagoing time up to a period of one or two weeks before the application. For example, if they create their application on July 24th they would enter a seagoing time of the period January 10th till July 10th. We did not see applications that were denied for this reason.

We also need to support periods that are entered over a period longer than the last 200 / 1850 days. We did so by assuming that the duration entered is distributed evenly over the whole period. For example, if they

[2] https://www.kiwaregister.nl/aanleverinformatie.aspx

enter a duration of 1000 days over the last ten years, we assume that each year contains 100 seagoing time days. The formula we use to calculate the proper duration is:

```
div = wholePeriod.TotalDays / matchingPeriod.TotalDays;
actualDuration = duration / div;
```

It first calculations the fraction of the period we are interested in after which it uses that fraction to calculate the actual duration value.

2.5 Relabeling the data

While classifying the data we noticed that the performance was lower than we expected (see Chapter 4). To see whether the issue was related to the data or to our features, we started to inspect each application for the competency: Rating forming part of a navigational watch (code 301). We used the newly created output method that allows us to view all cases belonging to a single competency. During this inspection we stumbled upon the following issues:

- Many applications were accepted without any seagoing time and education claims, even though such were required.
  We were aware that the data contained test data, which we tried to filter already. When tracing some of these invalidly accepted applications they seem to be created by accounts that did testing before, however, many other applications entered by those accounts were valid. This made it impossible to automatically filter them.
- Some applications were accepted with claims that did not get close to the required information as stated in applicable law.
  Some of those applications had the same issue as the one above, others filled in incorrect data but the uploaded proof did contain proper data, and others provided additional data afterwards.
- Some applications were denied with claims that seems to be perfectly fine.
  Applications can be denied when, for example, the uploaded proof is invalid. This means that the entered data itself could be fine after all.

All those issues made us realize that we cannot trust the current acceptance labels. In order to test this hypothesis we manually checked all applications of this specific competency, and deleted everything that was obviously incorrect. When in doubt, we kept the application. After classifying this competency again we saw that the performance improved. To be completely sure we did the same for four other mate competencies and for one officer competency. All the results are discussed in depth in Chapter 4.

Chapter 3

ML techniques

We will be using the Azure Machine Learning Studio[1] to create the models. Azure ML is the machine learning suite developed by Microsoft that runs in their own cloud. We use this suite because Avanade is Microsoft oriented and they want to use the gained information in future projects.

The problem seems to be a perfect example of binary classification: the application is either approved or denied. According to Rohrer, 2016 this leaves us with nine algorithms of which four of them train linear models (see Table 3.1).

TABLE 3.1: Two-class algorithms supported bij Azure ML.

Algorithm	**When to use**
Averaged perceptron	Fast training, linear model
Bayes point machine	Fast training, linear model
Boosted decision tree	Accuracy, fast training, large memory footprint
Decision forest	Accuracy, fast training
Decision jungle	Accuracy, small memory footprint
Locally deep SVM	>100 features
Logistic regression	Fast training, linear model
Neural network	Accuracy, long training times
SVM	>100 features, linear model

In the coming sections we will explain all these algorithms, and we will point out their up- and downsides. We will not go too deep into the matter but at the end of this chapter people within the IT sector but without prior ML knowledge should be able to understand the principles of these algorithms.

3.1 Decision trees, forests, and jungles

Decision trees are intuitive models that resemble real life thinking closely. For example, we take the model of Figure 3.1. We can easily deduce: `if A <= 5 or (A <= 10 and B >= 6) then Y else X`. This makes these kind of models easy to work with, as you can visualize what it is doing and therefore you are able to spot errors as well.

[1] https://studio.azureml.net

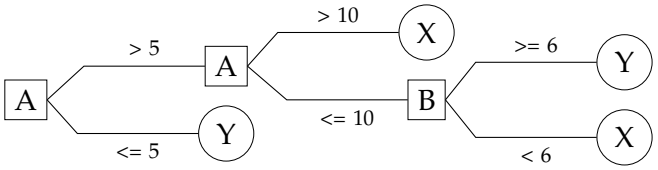


FIGURE 3.1: An example of a decision tree.

3.1.1 Boosted decision tree

The boosted decision tree classifier builds on this principle. Instead of learning one decision tree, it will learn multiple whereby each added tree corrects for the errors in the previous trees (Microsoft, 2016).

The main advantages of this classifier are: good accuracy and the learned trees can be inspected. The main disadvantages are: it is memory-intensive.

3.1.2 Decision forest

The decision forest classifier also learns multiple decision trees. Its predictions are combined into a final prediction (Rokach, 2016). By building such a forest, the mistakes of a single tree are compensated for by the other trees.

The main advantages and disadvantages are the same as boosted decision trees.

3.1.3 Decision jungle

The decision jungle classifier is an extension of decision forests which uses rooted decision directed acyclic graphs as a means to obtain compact and still accurate classifiers (Shotton et al., 2013).

The main advantage over decision forests is the advantage that tree branches are allowed to merge therefore reducing memory usage.

3.2 Neural networks and perceptrons

Neurons are cells that can have multiple inputs and a single output. The inputs fire signals at various rates and the neuron decides, based on these signals, what it should fire on its output. This is a simplified analogy for how our brain works.

3.2.1 Averaged perceptron

According to Clabaugh, Myszewski, and Pang, 2000 perceptrons are inspired by this biological phenomenon. More precisely, a perceptron is a single neuron which will receive the input features, process them and output a single value. The processing part applies a vector of learned weights to the input features, calculates the sum and then compares this to a certain threshold.

The weights have to be learned, this is done by an algorithm that loops through the dataset. The algorithm will test each entry against its current weights, if it classifies it incorrectly it will update the weights and continue.

It will keep doing so, until either the weights reach a fixed point or the maximum iteration count is reached.

The averaged perceptron is an improved version of this algorithm (Daumé III, 2012). This improved version will take the age of weights into account, so that weights that worked well for a long time are not easily changed by new events. For example, if your weights classified a thousand examples well, then the next one it fails, you do not want the weights to be changed so drastically that it will no longer be able to classify the previous examples.

The main advantages of this classifier are: it is very easy to implement, it learns very quickly, and it supports online learning meaning it can keep learning while new samples are added. The main disadvantages are: only capable of learning linearly separable patterns therefore unable to learn complex class boundaries and it prefers the data to be normalized which adds another preprocessing task.

3.2.2 Neural network

Neural networks are an extension of the previously explained principle where, instead of modeling one neuron, it models a network of neurons (Nielsen, 2016). A single neuron in such a network looks like a perceptron as described, however instead of outputting a `1` or `0` it outputs a value between `1` and `0`. Usually this value is calculated by a sigmoid function, although other activation functions can be used as well. This new activation function is chosen to make sure that learning new samples can only adjust the output in a small way, therefore the neurons that depend on this value do not have to adjust their weights drastically either.

The network consists of an input layer, one or more hidden layers, and an output layer. A hidden layer is nothing more than a layer that is not an input or an output layer. There are no limits on the number of hidden layers, although adding more layers will increase the learning time. Networks with many hidden layers are often called deep neural networks and can be effective in image or speech recognition (Krizhevsky, Sutskever, and Hinton, 2012; Hinton et al., 2012).

The main advantages of this classifier are: it can learn complex class boundaries and it usually has a good accuracy. The main disadvantages of this classifier are: it can take a long time to learn, the knowledge learned by the model is hard to understand (but not impossible, e.g. Castro, Mantas, and Benítez, 2002) and it is prone to overfitting (Tu, 1996).

3.3 Support vector machines

3.3.1 SVM (linear kernel)

Support vector machines (SVMs) are classifiers that represent the examples as points in space (Burges, 1998). The classifier will try to separate the instances of either class with a hyperplane (e.g. a line in two-dimensional space). When testing a new sample, it should fall on either side of the hyperplane, which allows the classifier to predict the class it belongs to. In order to minimize error the hyperplane should be in the center of the gap between classes. For example, in Figure 3.2 the lines H_2 and H_3 separate the classes, but only H_3 keeps most margin between the two.

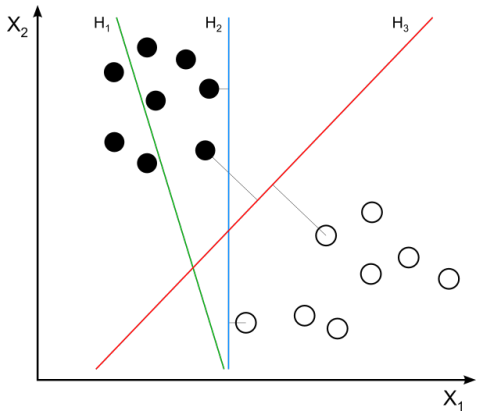


FIGURE 3.2: An example of hyperplanes (Wikipedia, 2016).

The main advantages of this classifier are: it is fast, and works well on linear separable problems. The main disadvantages of this classifier are: this variant is a linear classifier therefore unable to learn complex class boundaries, also the knowledge learned is, just like neural networks, very hard to understand (but not impossible, e.g. Fung, Sandilya, and Rao, 2005).

3.3.2 Locally deep SVM

Using a so called kernel trick, SVMs are able to do nonlinear classification. The kernel trick transforms the input space, therefore a hyperplane learned in the transformed space may be nonlinear in the original input space (see Figure 3.3). The main problem is that nonlinear SVMs are computationally expensive, especially when the training sample size is large (Ma and Guo, 2014). To solve this issue, Microsoft Research developed a method to reduce

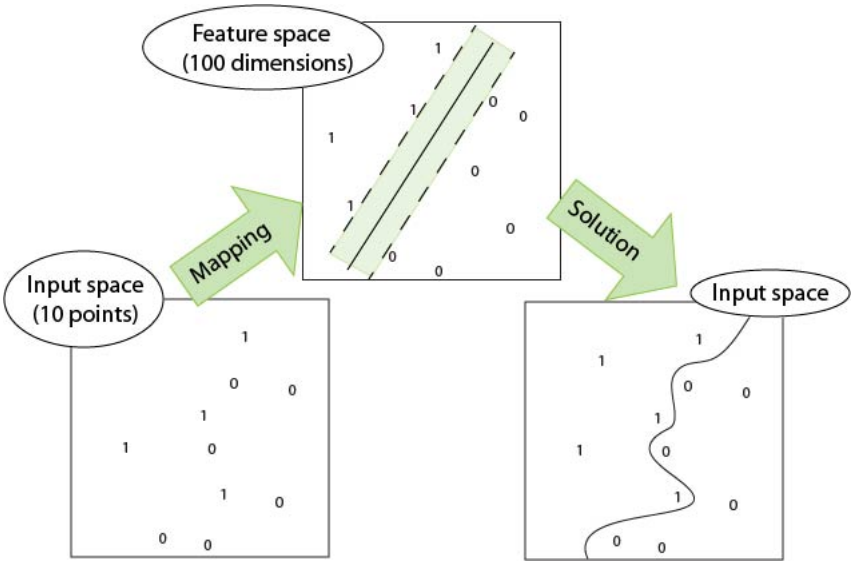


FIGURE 3.3: An example of the kernel trick (Thornton, 2016).

computation time with a moderate sacrifice in accuracy (Jose et al., 2013). We will not explain the details of this method, however, it is important to remember that in the results SVM will be the default linear variant while locally deep SVM is the nonlinear one.

The main advantages of this classifier are: it can learn complex class boundaries, and it works much faster than traditional nonlinear SVM models. The main disadvantage of this classifier is that it loses some accuracy compared to the traditional models.

3.4 Logistic regression

Logistic regression is a classifier that predicts the probability that a certain event occurs (Simonof, 2016). It tries to do so, by fitting the data to a logistic function. The paper by Andrew and Gao, 2007 explains the implementation of this algorithm as used in Azure.

The main advantages of this classifier are: it is very fast, and it works well if the optimal decision boundary is (approximately) linear. The main disadvantage are: it cannot learn complex class boundaries as it is a linear classifier, and it assumes a logistic distribution of the data.

3.5 Bayes point machine

The last classifier we will discuss is the Bayes point machine. The Bayes point machine is based on the concept of Bayes' theorem, while the implementation is actually very close to support vector machines (Herbrich, Graepel, and Campbell, 2001).

The main advantages are: it does not require parameter sweeping to find the best parameters, and it is not prone to overfitting. The main disadvantages are the same as other linear classifiers.

Chapter 4

Evaluation

4.1 Experiment setup

We are using Azure ML to classify the data. For each classifier we initially use the default settings (Appendix E.1) and we always keep the threshold at 0.5. The data is split into two groups: the training sample contains 75%, and the remaining 25% is used for scoring. An overview of this setup is shown on Appendix E.3. Subsequently, we will also try to find the best settings for the classifiers and provide the scores for them. This allows us to compare the classifiers to each other properly.

4.2 Results

We will first show the results of the initial classification that contains the original acceptance labels. After that we will show the results of classifying two individual competencies, followed by the results of classifying multiple mate competencies at once. Lastly we show the results of classifying all the relabeled competencies. The datasets are rather imbalanced, therefore we select the best model(s) based on total errors (accuracy), AUC and the ability to classify most samples from the negative set. Models which classify all samples as positive are ignored. The detailed results can be found in Appendix F.

4.2.1 Initial learning

The dataset used for scoring contains 3369 positive and 269 negative samples. As shown in Table 4.1 the algorithm Boosted Decision Tree was able to correctly classify the most negative samples, but still only 23% of the total negative set. Also, it classified the positive class worse than the other algorithms.

TABLE 4.1: Classification results for initial learning.

Algorithm	Total Errors	True Negative	AUC
Averaged Perceptron	263 (7%)	47 (17%)	0.759
Boosted Decision Tree	290 (8%)	62 (23%)	0.776
Locally-Deep SVM	248 (7%)	50 (19%)	0.729

4.2.2 Rating forming part of a navigational watch (code 301)

This is one of the mate competencies which we tried to relabel. We chose this competency because the majority of the applicants for this competency did not apply for an other competency at the same time. After relabeling we scored the dataset in two different ways. The first way is to remove everything we deemed invalid, the second way is to fix the label instead. The latter is to verify that de relabeled applications can be classified as well.

TABLE 4.2: Classification results for competency: 301.

Mode	Algorithm	Total Errors	True Negative	AUC
Original	Boosted Decision Tree	35 (8%)	5 (16%)	0.616
Original	Decision Forest	30 (7%)	1 (3%)	0.750
Remove	Boosted Decision Tree	14 (4%)	13 (62%)	0.945
Remove	Decision Forest	11 (3%)	10 (48%)	0.941
Fix	Boosted Decision Tree	18 (4%)	81 (88%)	0.972
Fix	Decision Forest	35 (8%)	69 (75%)	0.960

As shown in Table 4.2, the results improved significantly for both methods. It improved not only when compared to the results of classifying 301 with the original data, but also compared to classifying the whole dataset.

4.2.3 Officer in charge of a navigational watch (code 171)

The results of relabeling a mate competency were promising, so we decided to do the same for an officer competency. Officer competencies are much more complex as officers usually apply for many competencies and there are a lot of seagoing time / education combinations that can be used to apply for them. We chose this competency as it is an entry level officer competency for which we could still judge its validity (to a certain degree) and it also has a good number of applicants that only applied for this competency.

TABLE 4.3: Classification results for competency: 171.

Mode	Algorithm	Total Errors	True Negative	AUC
Original	Boosted Decision Tree	24 (9%)	9 (31%)	0.796
Original	Locally-Deep SVM	30 (11%)	8 (28%)	0.771
Original	Neural Network	34 (12%)	14 (48%)	0.713
Remove	Boosted Decision Tree	9 (4%)	10 (56%)	0.908
Remove	Locally-Deep SVM	11 (4%)	8 (44%)	0.903
Remove	Neural Network	16 (6%)	11 (61%)	0.865
Fix	Boosted Decision Tree	21 (7%)	27 (63%)	0.930
Fix	Locally-Deep SVM	33 (12%)	28 (65%)	0.871
Fix	Neural Network	27 (10%)	26 (60%)	0.847

Table 4.3 shows that the results improve here as well, although slightly less

than on competency 301. However, this competency is much harder to classify on its own as a large group of its members applied for multiple competencies. Also it was harder for us to determine the correct label, increasing the chance of errors.

4.2.4 Multiple mate competencies

After proving that classifying individual competencies improved after relabeling, we wanted to do the same while classifying multiple mate competencies at once. This should also give insight whether it is possible to create a classifier for combined competencies. To be exact: we ran the classifier on five competencies (code 301 to 305).

TABLE 4.4: Classification results for multiple mate competencies.

Mode	Algorithm	Total Errors	True Negative	AUC
Remove	Boosted Decision Tree	19 (4%)	13 (50%)	0.832
Remove	Decision Forest	18 (4%)	10 (38%)	0.824
Remove	Decision Jungle	19 (4%)	7 (27%)	0.866
Fix	Boosted Decision Tree	26 (5%)	98 (84%)	0.977
Fix	Decision Forest	51 (9%)	70 (60%)	0.951
Fix	Decision Jungle	65 (12%)	56 (48%)	0.952

The results (see Table 4.4) are rather interesting. For the case where we removed data, the classifiers are struggling. However, for the case where we fixed the data the Boosted Decision Tree performs well. We believe the issue is caused by the small number of negative samples on the additional competencies.

4.2.5 All relabeled competencies

We also wanted to see whether classifying all competencies at once is feasible. Especially since the mate and officer competencies are separate choices, therefore there are no applications that have both mate and officer competences.

TABLE 4.5: Classification results for all relabeled competencies.

Mode	Algorithm	Total Errors	True Negative	AUC
Remove	Boosted Decision Tree	28 (4%)	25 (57%)	0.930
Remove	Locally-Deep SVM	41 (6%)	20 (45%)	0.858
Fix	Boosted Decision Tree	57 (7%)	126 (79%)	0.930
Fix	Locally-Deep SVM	103 (13%)	101 (64%)	0.878

According to Table 4.5 the results for Boosted Decision Tree are fairly close to what they were when classified separately, although slightly worse on the total errors for the fix mode.

4.2.6 All relabeled competencies with optimized classifiers

Lastly, we wanted to see if we could improve the previously found results by optimizing the classifier parameters. In order to find the best settings we did a grid search over a combination of parameters (see Appendix E.2). For each combination we perform 10-fold cross validation with a random split (on the training data) after which the best model is selected and scored on the validation dataset.

TABLE 4.6: Optimal classification results for all relabeled competencies.

Mode	Algorithm	Total Errors	True Negative	AUC
Remove	Boosted Decision Tree	21 (3%)	26 (59%)	0.919
Remove	Locally-Deep SVM	35 (5%)	20 (45%)	0.858
Fix	Boosted Decision Tree	53 (6%)	123 (77%)	0.941
Fix	Locally-Deep SVM	84 (10%)	104 (65%)	0.873

As you can see in Table 4.6 the accuracy did slightly increase, however it did not change the best performing classifier. The classifier that benefited the most is the Decision Forest (Table 4.7) with: 32 trees, depth of 64, and 1024 random splits per node. Other notable improvements were: Decision Jungle on the fix mode, and SVM on the remove mode. Most classifiers had various settings that performed well on either the fix mode or the remove mode.

TABLE 4.7: Decision Forest optimization results for all relabeled competencies.

Mode	Optimized	Total Errors	True Negative	AUC
Remove	No	43 (6%)	1 (2%)	0.869
Remove	Yes	29 (4%)	17 (38%)	0.897
Fix	No	101 (12%)	67 (42%)	0.904
Fix	Yes	58 (7%)	118 (74%)	0.945

Chapter 5

Conclusion

Machine learning can be used to successfully classify certificate of competence applications. It will, however, take time, and money to relabel the data. We saw that the classification performance improved drastically when we corrected the labels, not only when we removed the applications with incorrect labels but also when we fixed those labels.

To answer our other research questions: in Chapter 2.4 we saw that the engineered features are easily extracted from the available data, therefore it should not take much effort to generate them for future applications. Classification results show that those features provide enough information to give a solid prediction. There are multiple binary classifiers that can be used (Chapter 3), although Chapter 4 showed that only the Boosted Decision Tree performed consistently on all the problems we tested. Optimization results show that most classifiers benefit from finding the best parameter values, however, only the Decision Forest showed a major improvement on both the remove mode and the fix mode. The Decision Jungle only improved on the fix mode, while SVM (linear kernel) improved most on the remove mode. The Boosted Decision Tree remains the best classifier after optimization.

5.1 Discussion and future work

The classification only works properly when we fix the labels. Although it was easy to spot the obvious errors, more complicated applications were much harder to judge. If we did not know for sure, we kept the current label. It is likely that a more experienced Kiwa employee could correct more issues, resulting in even better performance.

Manually fixing all applications will take so much time that the question arises whether the benefits outweigh the costs. Every application will still need manual checking to verify that the uploaded documents match the entered data, therefore the process cannot be fully automated. This makes it unlikely that the time it takes to correct all applications will be compensated by the improved handling process. The classification results could also be shown during the application process, to the applicant. This could reduce the number of invalid applications which makes it more likely that the investment can pay itself back in the long run.

An alternative to fixing existing data, is to adjust the handling process to add a step that indicates whether the application is correct or not based on the entered data. This data can be used to create a new classifier to classify future applications. The upside to this alternative is that it should be cheap to implement, the downside is that your current application history

is useless and that it adds another step (although small) to the handling process.

Appendix A

Overview of Competencies

A.1 Merchant navy / Sailing - Officer

- Master
  - No limitation
  - All ships less than 3000 GT[1]
- Master near coastal voyages (ships less than 500 GT)
  - Dutch territorial waters and the adjacent zone of the Kingdom[2]
  - Dutch territorial waters and Dutch exclusive economical zone
  - International coast
- Chief mate
  - No limitations
  - All ships less than 3000 GT[1]
- Chief mate near coastal voyages (ships less than 500 GT)[2]
  - Dutch territorial waters and the adjacent zone of the Kingdom
  - Dutch territorial waters and Dutch exclusive economical zone
  - International coast
- First maritime officer
  - No limitations
  - All ships less than 3000 GT and a propulsion power less than 3000 kW[1]
- Maritime officer
- Officer in charge of a navigational watch (all ships)
- Chief engineer
  - No limitations
  - All ships with a propulsion power less than 3000 kW[1]
- Chief engineer near coastal voyages (ships less than 3000 kW)
  - Dutch territorial waters and the adjacent zone of the Kingdom
  - Dutch territorial waters and Dutch exclusive economical zone
  - International coast
- Second engineer
  - No limitations
  - All ships with a propulsion power less than 3000 kW[1]
- Second engineer near coastal voyages (ships less than 3000 kW)

[1] Has the option to include all contractor material, supply vessels and tugs.
[2] Has the option to be exempted from the Advanced Fire Fighting training.

  - Dutch territorial waters and the adjacent zone of the Kingdom
  - Dutch territorial waters and Dutch exclusive economical zone
  - International coast
- Officer in charge of an engineering watch (all ships)
- Electro-technical officer (all ships)
- Master sailing ships
  - No limitations
  - Less than 500 GT in the trading area I and II
  - Less than 500 GT in the trading area I, II and IIIA
  - Less than 500 GT in the trading area unlimited
- Chief mate sailing ships
- Officer in charge of a navigational watch sailing ships
- Officer in charge of a navigational watch sailing ships less then 500 GT
  - Less than 500 GT in the trading area I, II and IIIA
  - Less than 500 GT in the trading area unlimited
- GMDSS
  - General Radio Operator
  - Restricted Radio Operator

A.2 Merchant navy / Sailing - Mate

- Rating forming part of a navigational watch
- Rating forming part of a navigational watch Sailing Ships less than 500 GT
- Rating forming part of an engineering watch
- Rating forming part of a navigational and engineering watch
- Able seafarer deck
- Able seafarer engine
- Able seafarer deck and engine
- Electro-technical rating

A.3 Fishing

- Skipper Fishing Vessel
  - No limitations
  - Less than 60 meter and a propulsion power less than 3000 kW
  - Less than 45 meter and a propulsion power less than 1125 kW in the trading area I
  - Less than 24 meter and a propulsion power less than 750 kW
  - Less than 24 meter and a propulsion power less than 750 kW in the trading area I
  - Less than 45 meter in the trading area II
  - Less than 24 meter in the trading area II
- Skipper Mussel Vessels
  - Within the trading area 1a between the West Frisian Islands (Wadden) and the Eastern Schelde
- Substitute Skipper Fishing Vessel

  - No limitations
  - Less than 60 meter and a propulsion power less than 3000 kW
  - Less than 45 meter and a propulsion power less than 1125 kW in the trading area I
  - Less than 45 meter and a propulsion power less than 1500 kW
- Mate Fishing Vessel
  - Less than 45 meters
- Mate / Engineer Fishing Vessel
  - No limitations
  - With a propulsion power less than 3000 kW
- Engineer Fishing Vessels
  - No limitations
  - Less than 45 meter and a propulsion power in the trading area II
  - Less than 24 meter and a propulsion power less than 750 kW in the trading area II
- GMDSS
  - General Radio Operator
  - Restricted Radio Operator

Appendix B

Overview of Sailing Time

There are a total of thirty-four positions on nine vessels.

B.1 Positions

- Able seafarer deck
- Able seafarer engine
- Chief Engineer (no limitations)
- Chief Engineer < 3000 kW
- Chief Mate (no limitations)
- Chief Mate < 3000 GT
- Chief Mate near coastal voyage
- Chief Mate sailing ships
- Cook
- Electro-technical officer
- Electro-technical rating
- Engineer fishing vessel
- First Maritime Officer (no limitations)
- First Maritime Officer < 3000 GT & < 3000 kW
- Maritime Officer (no limitation)
- Maritime Officer < 3000 GT & < 3000 kW
- Master (no limitations)
- Master < 3000 GT
- Master near coastal voyage
- Master sailing ships
- Mate fishing vessel
- Mate/engineer fishing vessel
- Officer in charge of a navigational watch
- Officer in charge of a engineering watch
- Rating apprentice
- Rating deck
- Rating engine room
- Rating engineer
- Rating sailing ships
- Second Engineer (no limitations)
- Second Engineer < 3000 kW
- Skipper fishing vessel
- Skipper mussel vessels
- Substitute Skipper fishing vessel

B.2 Type of vessels

- Chemical tanker
- Fishing vessel
- High-speed craft
- Liquefied gas tanker
- Merchant ship
- Oil tanker
- Other
- Passenger ship
- Sailing vessel

Appendix C

Overview of Education

There are a total of thirty-six institutions and 113 educations.

C.1 Institutions

- ABB Marine Academy
- Alphatron Marine B.V.
- Arbode Maritiem
- Berechja College
- Cooks Education
- CSMART
- De Ruyter Training & Consultancy
- Deltion College
- DHTC
- Emergency Control Maritime Training BV
- Enkhuizer Zeevaartschool
- Falck Nutec b.v.
- FMTC
- Foreign trainer
- ForestWave Navigation B.V.
- G4S Training & Safety Solutions
- Hogeschool van Amsterdam
- Hogeschool Zeeland
- Holland America Line N.V.
- Kon.Marine - Defensie Vaarschool
- Maritiem Instituut Willem Barentsz
- Maritieme Academie
- Nova College
- Nova Contract Maritieme Academie
- NTTA
- Quercus Technical Services B.V.
- ROC Frisian Port, Location Urk
- ROC Kop van N-Holland
- ROC Kop van Noord Holland
- ROC Zeeland
- RoodBovenGroen
- SAIO On-& Offshore Safety Training BV
- STC B.V.
- STC-KNRM Offshore Safety
- Telecom Agency
- TvK Instructie BV

- Zeevaartschool Abel Tasman

C.2 Educations

- Advanced Fire Fighting Certificate
- Basic Safety Training Certificate
- Certificate Able seafarer deck
- Certificate Able seafarer deck and engine
- Certificate Able seafarer engine
- Certificate Advanced training for oil tanker
- Certificate Advanced training for chemical tanker
- Certificate Advanced training for gas tanker
- Certificate Basic training for gas tanker
- Certificate Basic training for oil and chemical tanker
- Certificate captain limited area unlimited propulsion
- Certificate Cook
- Certificate Electro-technical officer
- Certificate Electro-technical rating
- Certificate Gasturbine propulsion
- Certificate medical care
- Certificate medical first aid
- Certificate of Radar Detection
- Certificate of Radar Navigation
- Certificate Rating in charge of a navigational and engineering watch all ships
- Certificate Rating in charge of a navigational watch all ships
- Certificate Rating in charge of a engineering watch all ships
- Certificate Ship Management Engineer Near-coastal voyages
- Certificate Ship Management Nautical Near-coastal voyages
- Certificate Steam propulsion
- Certificate Type Rating HSC
- Chemicals Tanker Certificate
- Deck Off. < 3000
- Deck Officer 200 Mile
- Deck Officer All Fishing
- Deck Officer all vessels
- Deck Officer/Engineer Small Fishing Vessels Certificate of Competency
- Diploma Engineer small ships
- Diploma Engineer all ships
- Diploma Engineer all ships
- Diploma Koopvaardij officier: Stuurman Werktuigkundige Kleine Schepen
- Diploma Large Sail
- Diploma Maritime Waterbuilder
- Diploma Mate all ships
- Diploma Mate small ships
- Diploma S4
- Diploma Small Sail
- Diploma W4
- Dredger Deck Officer

- Dredging Engineer
- ECDIS Certificate
- Engineer < 3000 KW
- Engineer all vessels
- Engineer Service Certificate
- Gas Tanker Certificate
- GMDSS General
- GMDSS Restricted Operator Certificate
- High-voltage Certificate
- K200 Certificate of Competency
- Maritime Officer
- Maritime Officer Certificate of Competency (senior secondary vocational education)
- Maritime Officer Certificate of Competency (university of applied sciences)
- Master Near Coastal Voyages Certificate of Competency
- Mate Waterbuilder
- Motorman (MM) Certificate (merchant navy certificate)
- Motorman (MM) Certificate of Competency
- Officer . S IV-v Certificate
- Officer. S IV-v Certificate
- Officer/Engineer Sea Fish
- Oil Tanker Certificate
- Proficiency Survival Craft
- Refresh course watch deck/engineer
- Refresher training advanced chemical tanker
- Refresher training Advanced Fire Fighting
- Refresher training advanced gas tanker
- Refresher training advanced oil tanker
- Refresher training basic gas tanker
- Refresher training basic oil and chemical tanker
- Refresher training Basic Safety
- Refresher training Proficiency in survival craft
- Replenishment Diploma Large Sail
- S VII Declaration
- S200 Certificate of Competency
- Sea Fishing Motorman (MvM) Certificate
- Sea Fishing Motorman (MVM) Certificate of Competency
- Ship Management
- Ship Management Engineer
- Ship Management Nautical
- Ship Management Nautical/Engineer
- Ship Management < 3000 GT
- Skipper/Engineer Restricted Working Area Certificate of Competency
- State degree A
- State degree B
- State degree C
- State Degree S1
- State Degree S2
- State Degree S3
- State degree SK

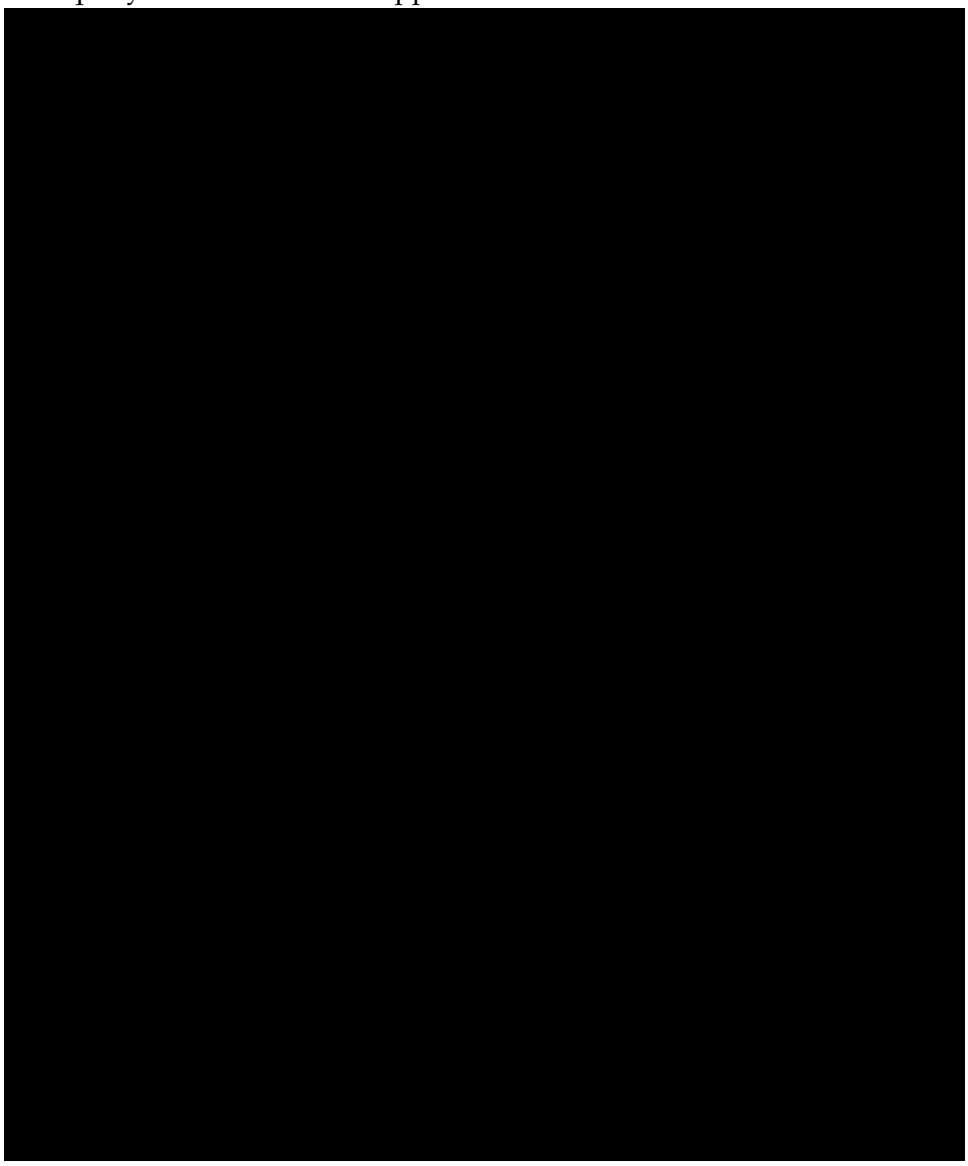
- State degree SKA
- State Degree SW5
- State Degree SWK
- Statement recordbook rating deck
- Statement recordbook rating engine
- Statement recordbook upcoming electro-technical rating
- Statement upcoming rating deck
- Statement upcoming rating engine
- SVB Declaration
- SW 5
- SW V < 24m
- SW V < 45m
- SW V Certificate
- SW VI Certificate
- SW4
- SW5
- SW6
- W IV-v
- W IV-v Certificate
- Written Declaration Master

Appendix D

Database Queries

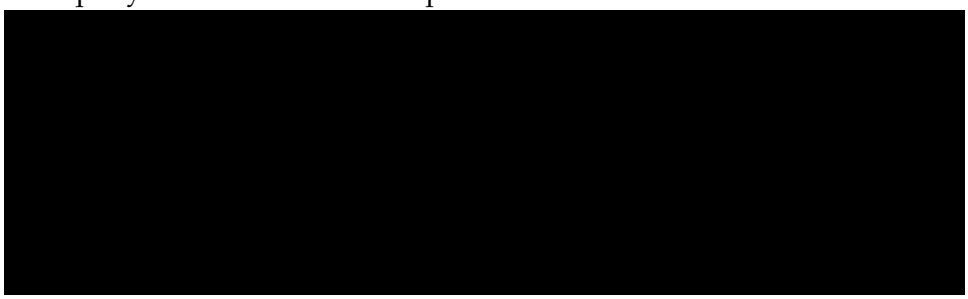
D.1 Applications

This query retrieves the CoC applications:



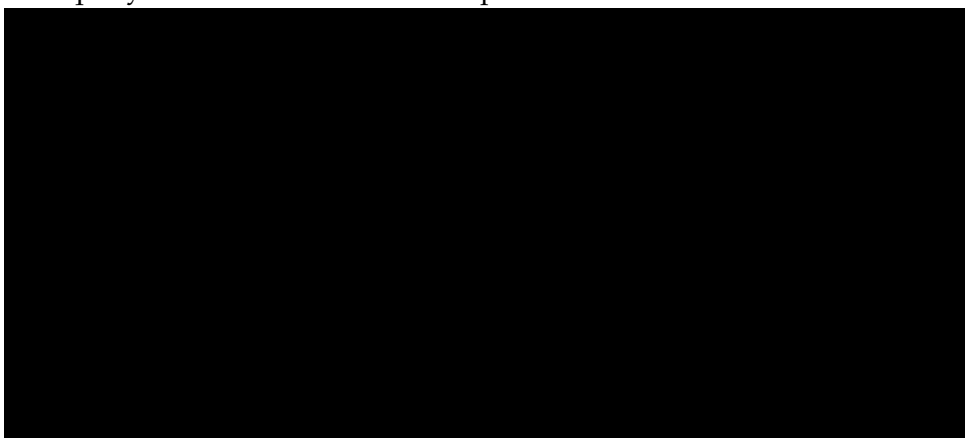
D.2 Product

This query retrieves the selected product:



D.3 Competencies

This query retrieves the selected competencies:



D.4 Medical claims

This query retrieves the medical claims:

D.5 Seagoing time claims

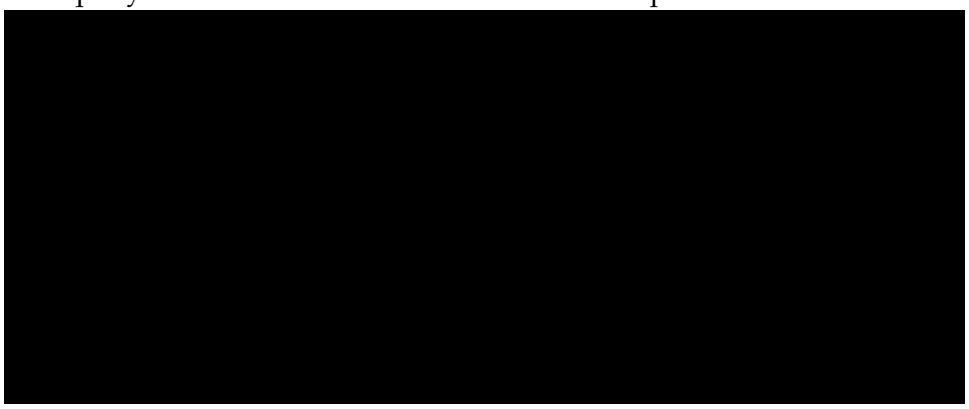
This query retrieves the seagoing time claims.

D.6 Education claims

This query retrieves the education claims:

D.7 Attachments

This query retrieves some information about the uploaded attachments:



Appendix E

Azure Machine Learning Setup

E.1 Default settings

Averaged perceptron

Setting	Value
Learning rate	1
Maximum number of iterations	10

Bayes point machine

Setting	Value
Number of training iterations	30
Include bias	Yes
Allow unknown categorical levels	Yes

Boosted decision tree

Setting	Value
Maximum number of leaves per tree	20
Minimum number of samples per leaf node	10
Learning rate	0.2
Number of trees constructed	100
Allow unknown categorical levels	Yes

Decision forest

Setting	Value
Resampling method	Bagging
Number of decision trees	8
Maximum depth of the decision trees	32
Number of random splits per node	128
Minimum number of samples per leaf node	1
Allow unknown categorical levels	Yes

Decision jungle

Setting	Value
Resampling method	Bagging
Number of decision DAGs	8
Maximum depth of the decision DAGs	32
Maximum width of the decision DAGs	128
Number of optimization steps per decision DAG layer	2048
Allow unknown categorical levels	Yes

Locally-deep SVM

Setting	Value
Depth of the tree	3
Lambda W	0.1
Lambda Theta	0.01
Lambda Theta Prime	0.01
Sigmoid sharpness	1
Number of iterations	15000
Feature normalizer	Min-Max normalizer
Allow unknown categorical levels	Yes

Logistic regression

Setting	Value
Optimization tolerance	1E-07
L1 regularization weight	1
L2 regularization weight	1
Memory size for L-BFGS	20
Allow unknown categorical levels	Yes

Neural network

Setting	Value
Hidden layer specification	Fully-connected case
Number of hidden nodes	100
Learning rate	0.1
Number of learning iterations	100
The initial learning weights diameter	0.1
The momentum	0
The type of normalizer	Min-max normalizer
Shuffle examples	Yes
Allow unknown categorical levels	Yes

SVM

Setting	Value
Number of iterations	1
Lambda	0.001
Normalize features	Yes
Project to the unit-sphere	No
Allow unknown categorical levels	Yes

E.2 Settings used for grid search

Averaged perceptron

Setting	Value
Learning rate	0.1, 0.5, 1.0
Maximum number of iterations	1, 10, 20

Boosted decision tree

Setting	Value
Maximum number of leaves per tree	2, 8, 32, 128
Minimum number of samples per leaf node	1, 10, 50
Learning rate	0.025, 0.05, 0.1, 0.2, 0.4
Number of trees constructed	20, 100, 500

Decision forest

Setting	Value
Number of decision trees	1, 8, 32
Maximum depth of the decision trees	1, 16, 64
Number of random splits per node	1, 128, 1024
Minimum number of samples per leaf node	1, 4, 16

Decision jungle

Setting	Value
Number of decision DAGs	1, 8, 32
Maximum depth of the decision DAGs	1, 16, 64
Maximum width of the decision DAGs	1, 128, 1024
Number of optimization steps per decision DAG layer	1024, 4096, 16384

Locally-deep SVM

Setting	Value
Depth of the tree	1, 3
Lambda W	0.1, 0.01, 0.001
Lambda Theta	0.1, 0.01, 0.001
Lambda Theta Prime	0.1, 0.01, 0.001
Sigmoid sharpness	1.0, 0.1, 0.01
Number of iterations	10000, 15000, 20000

Logistic regression

Setting	Value
Optimization tolerance	1E-04, 1E-07
L1 regularization weight	0.0, 0.01, 0.1, 1.0
L2 regularization weight	0.01, 0.1, 1.0
Memory size for L-BFGS	5, 20, 50

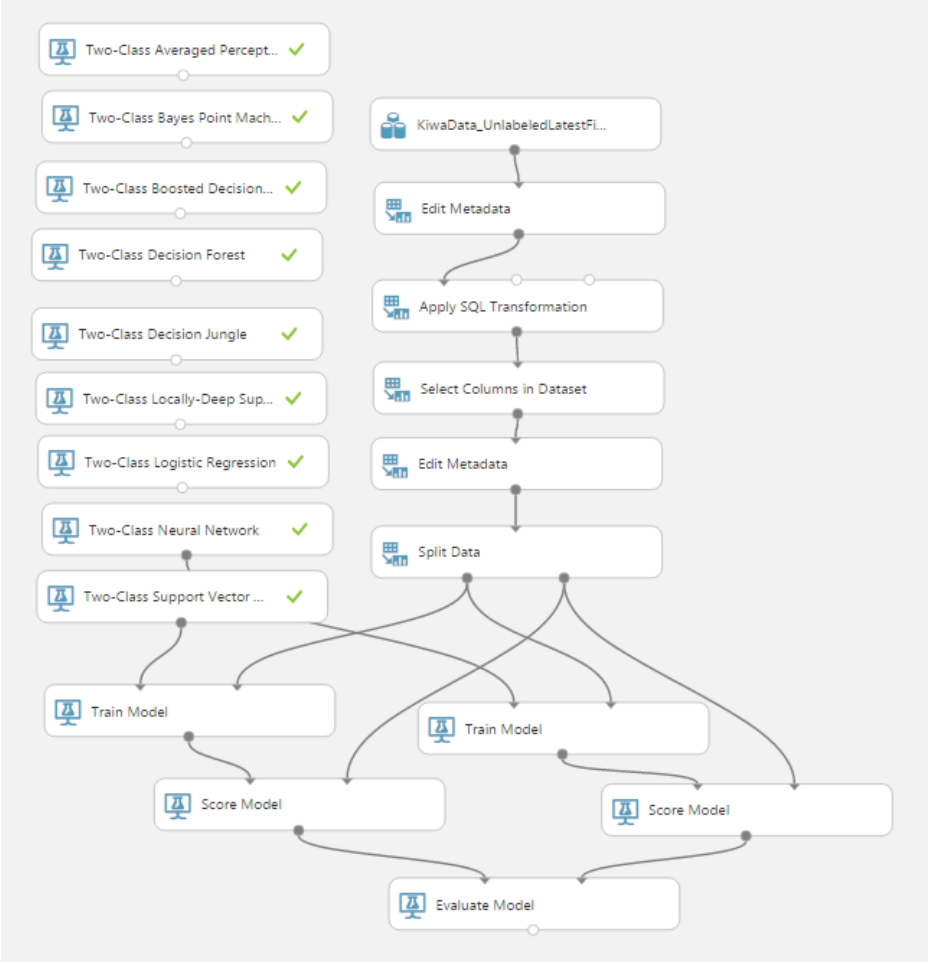
Neural network

Setting	Value
Learning rate	0.01, 0.02, 0.04
Number of learning iterations	20, 40, 80, 160

SVM

Setting	Value
Number of iterations	1, 10, 100
Lambda	1E-05, 1E-04, 0.001, 0.01, 0.1
Normalize features	Yes
Project to the unit-sphere	No
Allow unknown categorical levels	Yes

E.3 Experiment setup



We use two Score Models to quickly run to classifiers in parallel. The SQL transformation is used to select single or multiple competencies.

Appendix F

ML Results

F.1 Complete dataset

Dataset with original entries

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	3328	41	222	47	0.962	0.759
Bayes Point Machine	3369	0	269	0	0.962	0.490
Boosted Decision Tree	3286	83	207	62	0.958	0.776
Decision Forest	3369	0	269	0	0.962	0.653
Decision Jungle	3369	0	269	0	0.962	0.621
Locally-Deep SVM	3340	29	219	50	0.964	0.745
Logistic Regression	3361	8	250	19	0.963	0.754
Neural Network	3366	3	264	5	0.962	0.700
SVM	3361	8	254	15	0.962	0.729

F.2 Competency 301

Dataset with original entries

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	387	3	30	1	0.959	0.650
Bayes Point Machine	390	0	30	1	0.963	0.440
Boosted Decision Tree	381	9	26	5	0.956	0.616
Decision Forest	390	0	30	1	0.963	0.750
Decision Jungle	390	0	30	1	0.963	0.713
Locally-Deep SVM	379	11	29	2	0.950	0.641
Logistic Regression	390	0	30	1	0.963	0.654
Neural Network	386	4	29	2	0.959	0.661
SVM	390	0	31	0	0.962	0.625

Dataset with invalid entries removed

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	314	5	10	11	0.977	0.858
Bayes Point Machine	319	0	17	4	0.974	0.452
Boosted Decision Tree	313	6	8	13	0.978	0.945
Decision Forest	319	0	11	10	0.983	0.941
Decision Jungle	319	0	21	0	0.968	0.927
Locally-Deep SVM	316	3	9	12	0.981	0.818
Logistic Regression	318	1	19	2	0.970	0.902
Neural Network	316	3	9	12	0.981	0.752
SVM	315	4	18	3	0.966	0.822

Dataset with invalid entries fixed

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	313	15	29	63	0.934	0.920
Bayes Point Machine	316	12	39	53	0.925	0.768
Boosted Decision Tree	321	7	11	81	0.973	0.972
Decision Forest	316	12	23	69	0.948	0.960
Decision Jungle	327	1	35	57	0.948	0.952
Locally-Deep SVM	313	15	24	68	0.941	0.902
Logistic Regression	316	12	32	60	0.935	0.909
Neural Network	319	9	28	64	0.945	0.927
SVM	315	13	30	62	0.936	0.893

F.3 Competency 171

Dataset with original entries

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	244	5	21	8	0.949	0.852
Bayes Point Machine	248	1	29	0	0.943	0.433
Boosted Decision Tree	245	4	20	9	0.953	0.796
Decision Forest	248	1	25	4	0.950	0.775
Decision Jungle	249	0	29	0	0.945	0.708
Locally-Deep SVM	240	9	21	8	0.941	0.771
Logistic Regression	248	1	23	6	0.954	0.860
Neural Network	230	19	15	14	0.931	0.713
SVM	246	3	25	4	0.946	0.856

Dataset with invalid entries removed

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	230	1	10	8	0.977	0.943
Bayes Point Machine	230	1	18	0	0.960	0.193
Boosted Decision Tree	230	1	8	10	0.981	0.908
Decision Forest	231	0	13	5	0.973	0.918
Decision Jungle	231	0	18	0	0.963	0.909
Locally-Deep SVM	230	1	10	8	0.977	0.903
Logistic Regression	231	0	17	1	0.965	0.965
Neural Network	222	9	7	11	0.965	0.865
SVM	231	0	18	0	0.963	0.942

Dataset with invalid entries fixed

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	223	15	21	22	0.925	0.854
Bayes Point Machine	238	0	42	1	0.919	0.318
Boosted Decision Tree	233	5	16	27	0.957	0.930
Decision Forest	233	5	21	22	0.947	0.863
Decision Jungle	238	0	43	0	0.917	0.855
Locally-Deep SVM	220	18	15	28	0.930	0.871
Logistic Regression	229	9	31	12	0.920	0.882
Neural Network	228	10	17	26	0.944	0.847
SVM	223	15	32	11	0.905	0.837

F.4 Competency 301, 302, 303, 304, and 305

Dataset with original entries

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	490	11	35	5	0.955	0.661
Bayes Point Machine	500	1	39	1	0.962	0.416
Boosted Decision Tree	480	21	33	7	0.947	0.726
Decision Forest	500	1	39	1	0.962	0.774
Decision Jungle	501	0	40	0	0.962	0.736
Locally-Deep SVM	490	11	31	9	0.959	0.718
Logistic Regression	500	1	39	1	0.962	0.617
Neural Network	499	2	39	1	0.961	0.735
SVM	499	2	40	0	0.960	0.628

Dataset with invalid entries removed

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	398	13	14	12	0.967	0.774
Bayes Point Machine	411	0	24	2	0.972	0.346
Boosted Decision Tree	405	6	13	13	0.977	0.832
Decision Forest	409	2	16	10	0.978	0.824
Decision Jungle	411	0	19	7	0.977	0.866
Locally-Deep SVM	403	8	13	13	0.975	0.771
Logistic Regression	410	1	18	8	0.977	0.766
Neural Network	400	11	12	14	0.972	0.761
SVM	410	1	26	0	0.968	0.738

Dataset with invalid entries fixed

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	416	8	44	73	0.941	0.907
Bayes Point Machine	415	9	51	66	0.933	0.752
Boosted Decision Tree	417	7	19	98	0.970	0.977
Decision Forest	420	4	47	70	0.943	0.951
Decision Jungle	420	4	61	56	0.928	0.952
Locally-Deep SVM	416	8	40	77	0.945	0.906
Logistic Regression	419	5	46	71	0.943	0.908
Neural Network	418	6	44	73	0.944	0.910
SVM	407	17	45	72	0.929	0.883

F.5 Competency 301, 302, 303, 304, 305, and 171

Dataset with original entries

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	737	13	58	11	0.954	0.699
Bayes Point Machine	749	1	69	0	0.955	0.416
Boosted Decision Tree	720	30	46	23	0.950	0.715
Decision Forest	750	0	69	0	0.956	0.688
Decision Jungle	750	0	69	0	0.956	0.646
Locally-Deep SVM	722	28	43	26	0.953	0.734
Logistic Regression	747	3	61	8	0.959	0.743
Neural Network	750	0	60	9	0.962	0.626
SVM	738	12	62	7	0.952	0.701

Dataset with invalid entries removed

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	629	13	28	16	0.968	0.845
Bayes Point Machine	641	1	44	0	0.966	0.297
Boosted Decision Tree	633	9	19	25	0.978	0.930
Decision Forest	642	0	43	1	0.968	0.869
Decision Jungle	642	0	44	0	0.967	0.837
Locally-Deep SVM	625	17	24	20	0.968	0.858
Logistic Regression	638	4	40	4	0.967	0.821
Neural Network	636	6	25	19	0.976	0.691
SVM	640	2	43	1	0.966	0.840

Dataset with invalid entries fixed

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	618	45	67	92	0.917	0.851
Bayes Point Machine	647	16	96	63	0.920	0.623
Boosted Decision Tree	639	24	33	126	0.957	0.930
Decision Forest	654	9	92	67	0.928	0.904
Decision Jungle	663	0	125	34	0.914	0.831
Locally-Deep SVM	618	45	58	101	0.923	0.878
Logistic Regression	635	28	70	89	0.928	0.867
Neural Network	631	32	79	80	0.919	0.862
SVM	598	65	69	90	0.899	0.828

F.5.1 With optimized classifiers

Dataset with invalid entries removed

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	626	16	23	21	0.970	0.853
Boosted Decision Tree	639	3	18	26	0.984	0.919
Decision Forest	640	2	27	17	0.978	0.897
Decision Jungle	641	1	44	0	0.966	0.894
Locally-Deep SVM	631	11	24	20	0.973	0.823
Logistic Regression	634	8	22	22	0.977	0.771
Neural Network	638	4	28	16	0.976	0.693
SVM	621	21	20	24	0.968	0.818

Dataset with invalid entries fixed

Algorithm	TP	FN	FP	TN	F1	AUC
Averaged Perceptron	622	41	67	92	0.920	0.851
Boosted Decision Tree	646	17	36	123	0.961	0.941
Decision Forest	646	17	41	118	0.957	0.945
Decision Jungle	629	34	50	109	0.937	0.918
Locally-Deep SVM	634	29	55	104	0.938	0.873
Logistic Regression	635	28	70	89	0.928	0.867
Neural Network	627	36	52	107	0.934	0.887
SVM	624	39	66	93	0.922	0.865

Bibliography

Andrew, Galen and Jianfeng Gao (2007). "Scalable training of L 1-regularized log-linear models". In: *Proceedings of the 24th international conference on Machine learning*. ACM, pp. 33–40.

Avanade (2016). *Avanade Fast Facts - About Avanade*. URL: http://www.avanade.com/en/about-avanade/about-us/fast-facts (visited on 04/12/2016).

Burges, Christopher JC (1998). "A tutorial on support vector machines for pattern recognition". In: *Data mining and knowledge discovery* 2.2, pp. 121–167.

Castro, Juan L, Carlos J Mantas, and José Manuel Benítez (2002). "Interpretation of artificial neural networks by means of fuzzy rules". In: *IEEE Transactions on Neural Networks* 13.1, pp. 101–116. DOI: 10.1109/72.977279.

Clabaugh, Caroline, Dave Myszewski, and Jimmy Pang (2000). *The artificial neuron*. URL: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Neuron/index.html (visited on 07/14/2016).

Daumé III, Hal (2012). "The Perceptron". In: *A Course in Machine Learning*. Chap. 3, pp. 39–52.

Fung, Glenn, Sathyakama Sandilya, and R Bharat Rao (2005). "Rule extraction from linear support vector machines". In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, pp. 32–40.

Herbrich, Ralf, Thore Graepel, and Colin Campbell (2001). "Bayes point machines". In: *Journal of Machine Learning Research* 1.Aug, pp. 245–279.

Hinton, Geoffrey et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal Processing Magazine* 29.6, pp. 82–97. DOI: 10.1109/MSP.2012.2205597.

IBM Analytics (2015). "Foundational Methodology for Data Science". In: URL: http://public.dhe.ibm.com/common/ssi/ecm/im/en/imw14824usen/IMW14824USEN.PDF.

Jose, Cijo et al. (2013). "Local deep kernel learning for efficient non-linear svm prediction". In: *Proceedings of the 30th international conference on machine learning (ICML-13)*, pp. 486–494.

Kiwa (2016). *About Kiwa*. URL: http://www.1kiwa.com/about_Kiwa/ (visited on 04/12/2016).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Ma, Yunqian and Guodong Guo (2014). *Support vector machines applications*. Springer, p. 166.

Microsoft (2016). *Two-Class Boosted Decision Tree*. URL: https://msdn.microsoft.com/en-us/library/azure/dn906025.aspx (visited on 08/01/2016).

Nielsen, Michael (2016). "Using neural nets to recognize handwritten digits". In: *Neural Networks and Deep Learning*. Chap. 1. URL: http://neuralnetworksanddeeplearning.com/chap1.html (visited on 07/14/2016).

Piatetsky, Gregory (2014). *CRISP-DM, still the top methodology for analytics, data mining, or data science projects*. URL: http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html (visited on 05/24/2016).

Rohrer, Brandon (2016). *Microsoft Azure Machine Learning Algorithm Cheat Sheet*. URL: https://azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-cheat-sheet/ (visited on 04/28/2016).

Rokach, Lior (2016). "Decision forest: Twenty years of research". In: *Information Fusion* 27, pp. 111–125.

Rollins, John (2015). *Why we need a methodology for data science*. URL: http://www.ibmbigdatahub.com/blog/why-we-need-methodology-data-science (visited on 05/24/2016).

Shotton, Jamie et al. (2013). "Decision jungles: Compact and rich models for classification". In: *Advances in Neural Information Processing Systems*, pp. 234–242.

Simonof, Jeffrey S. (2016). *Logistic regression — modeling the probability of success*. URL: http://people.stern.nyu.edu/jsimonof/classes/2301/pdf/logistic.pdf (visited on 10/28/2016).

Thearling, Kurt (2016). "An Introduction to Data Mining: Discovering hidden value in your data warehouse". In: URL: http://www.thearling.com/text/dmwhite/dmwhite.htm (visited on 04/13/2016).

Thornton, Chris (2016). *Machine Learning - Lecture 15 Support Vector Machines*. URL: http://users.sussex.ac.uk/~christ/crs/ml/lec08a.html (visited on 10/28/2016).

Tu, Jack V. (1996). "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes". In: *Journal of Clinical Epidemiology* 49.11, pp. 1225 –1231. ISSN: 0895-4356. DOI: 10.1016/S0895-4356(96)00002-9.

Wikipedia (2016). *Svm seperating hyperplanes*. URL: https://commons.wikimedia.org/wiki/File:Svm_separating_hyperplanes_(SVG).svg (visited on 10/28/2016).

Wirth, Rüdiger and Jochen Hipp (2000). "CRISP-DM: Towards a standard process model for data mining". In: *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, pp. 29–39.