Bachelor of Science in Computer Science
May 2018

Applied machine learning in the logistics sector

A comparative analysis of supervised learning algorithms

Petrus Allberg

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Bachelor of Science in Computer Science. The thesis is equivalent to 10 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author:
Petrus Allberg
E-mail: petrusallberg@hotmail.se, peal15@student.bth.se

University advisor:
Universitetslektor Hans Tap
Department of Creative Technologies

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Background. Machine learning is an area that is being explored with great haste these days, which inspired this study to investigate how seven different supervised learning algorithms perform compared to each other. These algorithms were used to perform classification tasks on logistics consignments, the classification is binary and a consignment can either be classified as missed or not.
Objectives. The goal was to find which of these algorithms perform well when used for this classification task and to see how the results varied with different sized datasets. Importance of the features which were included in the datasets has been analyzed with the intention of finding if there is any connection between human errors and these missed consignments.
Methods. The process from raw data to a predicted classification has many steps including data gathering, data preparation, feature investigation and more. Through cross-validation, the algorithms were all trained and tested upon the same datasets and then evaluated based on the metrics recall and accuracy.
Results. The scores on both metrics increase with the size of the datasets, and when comparing the seven algorithms, two does not perform equally compared to the other five, which all perform moderately the same.
Conclusions. Any of the five algorithms mentioned prior can be chosen for this type of classification, or to further study based on other measurements, and there is an indication that human errors could play a part on whether a consignment gets classified as missed or not.

Keywords: Machine learning, supervised learning, logistics, transport.

Acknowledgments

I want to thank the company WIP and especially the CTO, Mats Karlsson for letting me use their data for my study. I also want to thank the students Alexander Ljung, Emil Hörnlund, Magnus Greiff and Marcus Ljungström who were involved in my project at WIP, and they helped me with input during my work. Lastly, I'd like to thank my supervisor at BTH, Hans Tap who guided me along the way of writing my thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

My thesis is inspired by the group project me and my fellow students from BTH did for WIP during spring 2018. However, this comparative analysis is a side project where I dug deeper into one of the problems that we analyzed during the project. This was to use machine learning techniques to classify transport bookings as *missed* or *non-missed*. *Missed* is one status type out of a set of status types in the WIP database, and this status is acquired when a transport consignment did not reach its intended destination. The *non-missed* consignments can be any of the other status types that are not *missed*. During this analysis, I have studied the results of seven different supervised learning algorithms based on two different metrics. These results were then compared to each other with the goal to see if there was any difference between the algorithms. I have also studied how the results differ when exposing these algorithms to different sized datasets. Lastly, I investigated if these results are connected to human errors.

I used the machine learning tool scikit-learn throughout the research together with many more tools and Python libraries introduced in chapter 4, method. The reason why I chose to use scikit-learn in this study is that I have experience with the tool because of the group project mentioned above, and also because I am familiar with the Python programming language. The tool also has a vast community and great documentation which is excellent when exploring and solving problems that occur throughout the project.

Motivation

The group project mentioned above was done for a company in Karlskrona called Wireless Independent Provider (WIP). WIP is a company that delivers many services such as app-development, security solutions and more. They also provide a logistics management system for a large number of haulage contractors foremost in Sweden but also in other parts of Europe. The project we did was for the Research and Development part of WIP, called WIP Labs. The goal of the group project was to provide a proof of concept about how WIP can integrate machine learning to benefit from the vast amount of data they have saved over a period of years from the logistics management system.

The reason behind this group project is that WIP wanted to get into the machine learning topic and at the same time investigate how their logistics management system could be more efficient for the operators that use this system on a daily basis. A massive loss in the transport and logistics sector is due to human errors. For ex-

ample, a transport consignment gets double-booked meaning that one of the drivers going to pick up this consignment will miss it because the first driver has already taken care of it, this was the kind of errors we wanted to investigate. By building predictive models based on supervised classification algorithms we wanted to see if machine learning could help to mitigate these errors.

According to a report done by Svenskt Näringsliv, there's often a link between the economy of a nation and its transport and logistics infrastructure. They claim that the cost of the Swedish logistics industry is nearly 18% of Swedens GDP and that the average cost for all European countries is around 12% of their GDP[1]. This claim means that there's an economic motivation to improve the logistics sector, and there's an environmental motivation as well because more efficient transports could lower the environmental impact.

Aims and objectives

As previously motivated there is much room for improvement in the logistics sector that can be done to create more efficiency in the daily workflow for these traffic operators. The goal of this thesis is to find an algorithm that can be used as a classifier in this specific problem. Furthermore, WIP is interested in the human factor behind these faulty consignments, and one objective is to find connections between human-related factors and the consignments that are missed.

Research questions

This research revolves around trying to answer the following three questions:

1. *Is there any difference between different machine learning algorithms when used to classify transport-bookings?*
2. *Does this difference change with the dataset size?*
3. *Is there any signs that the transport-bookings classified as missed could be based on human errors?*

Chapter 2

Background

Machine learning is the process of letting a computer learn by analyzing data and train it to predict an outcome, classify a class or automate a task, based on that data. T.M. Mitchell provides the following definition "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."[2].

There are three main techniques used in machine learning called supervised learning, unsupervised learning, and reinforcement learning. Each learning method is chosen based on the data available and the goal you want to achieve. I will briefly describe each technique below.

Supervised learning

Supervised learning is applicable when the data you want to train on is labeled. The goal is to predict the value of an outcome based on input measures[3] and the value you want to predict is known, for example, if a person has diabetes or not. I have used supervised learning throughout my research because it fits my problem very well, the classes to identify are already known (*missed* and *non-missed*), and I have access to a large quantity of data.

Unsupervised learning

In unsupervised learning, the goal is to find associations and patterns amongst the data that's analyzed. A result of this is that there is no precise outcome measurement when using unsupervised learning as in contrast to supervised learning[3]. This technique is used to find anomalies or clusters in data, and a common field is outlier detection.

Reinforcement learning

Reinforcement learning is learning based on how humans and other intellectual animals learn by trial-and-error. The machine is reinforced by evaluating the sum of a stochastic immediate reward and a change of state based on the actions it takes in a particular environment. The value of this procedure then gets evaluated by a form of bootstrapping since the value of each state is depending on the value of reachable states from it[4].

Reinforcement learning is often used in game theory and was used in the famous experiment with AlphaGo by Google Deepmind[5] to master the boardgame Go.

Data preparation

The most important part of machine learning is to structure your data correctly or else the results you want to get from a machine learning algorithm will not make any sense. There are several techniques used to prepare the data before you can feed it into the algorithm of choice. First off you will need to select the data that is going to be analyzed, then this data must be preprocessed, and later you can transform it to fit your needs. Sometimes you will get the best result by using all the data available, and sometimes you can get a good result by only using parts of the available data. The data preprocessing is something that you need to do before the training sessions can start.

To select data can be challenging but there are things you can do to help you choose the correct features. You can talk to persons that are involved in the type of problem you are analyzing, in my case that is the logistics and transport sector. There are also feature selection and feature evaluation algorithms that will evaluate the influence of each feature based on the outcome.

2.1 Algorithms

During my analysis, I have investigated seven different machine learning algorithms, all which I have used for supervised classification. I have also used a feature selection algorithm to analyze the importance of each feature in my dataset. The algorithms Logistic Regression 2.1.1 and Random Forest 2.1.6 was studied in the works by Lind Nilsson and Andersson [6], [7]. Moreover, Bakhtyar and Henesey[8] have used versions of the following algorithms, K-Nearest Neighbors 2.1.2, Support Vector Machine 2.1.5, AdaBoost 2.1.7 and Decision Tree 2.1.3 but in another framework than scikit-learn, called WEKA. Lastly, Gaussian Naïve Bayes 2.1.4 has been used by Lind Nilsson and Andersson in scikit-learn, and also by Bakhtyar and Henesey, though in another version. The fact that these algorithms have previously been used for similar problems is the reason why I have applied them to my problem. One thing to notice is that I used the default constructing values for all the algorithms, except enabling multi-threading for the algorithms that supported it, with the intention to speed up the analysis process.

2.1.1 Logistic Regression(LR)

Logistic Regression, also known as logit regression, MaxEnt classifier or log-linear classifier, is an algorithm used for classification even though its name includes *regression*. It is a purely statistical approach of making predictions and can be used for both binary classification and multiclass classification problems. The algorithm estimates the probability of an outcome based on independent variables that are input to the algorithm. In scikit-learn, the Logistic Regression algorithm is implemented

with regularization which can be adjusted by passing additional arguments to the algorithm's constructor[9].

2.1.2 K-Nearest Neighbors(KNN)

K-Nearest Neighbors is a neighbor based algorithm that bases its prediction on a majority voting process where the K nearest neighboring samples to the input sample is analyzed. The number K is user supplied and the higher this number is, the more noise will be in the predictions, for binary classification the number K is recommended to be an odd number to avoid tied votes. The scikit-learn implementation of this algorithm is weighting each neighboring point uniformly and the default number K=5[9].

The following figure 1 shows how the algorithm makes a classification, the sample that is unknown is the blue test sample, and by looking at the K=5 nearest neighbors, the algorithm decides if the new sample is more likely to belong to *missed* or *non-missed*. In this case, out of the five nearest neighbors, four of them belong to the class *missed*, and the test sample is classified as such because of the majority voting.

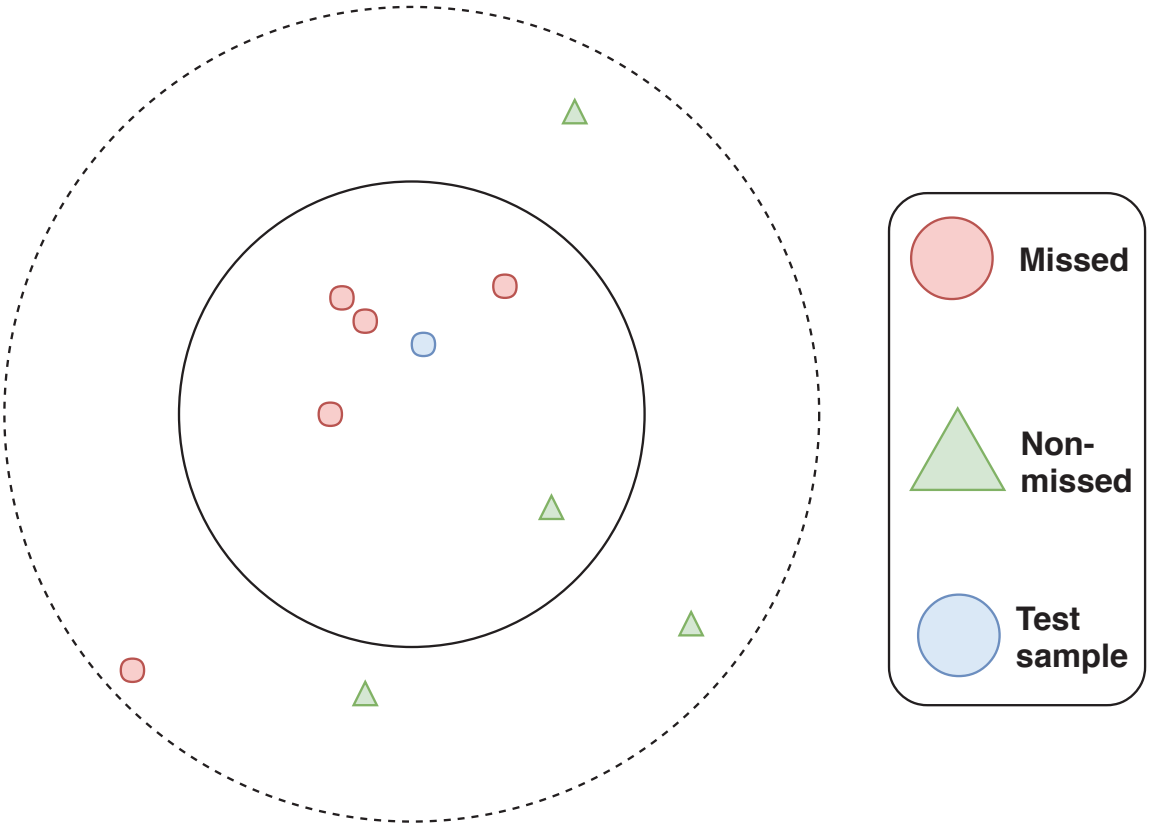

Figure 1: Nearest Neighbor.

2.1.3 Decision Tree(DCC)

Decision trees are non-parametric supervised learning. The goal is to create a predictive model of a target variable based on learning decision rules extracted from the

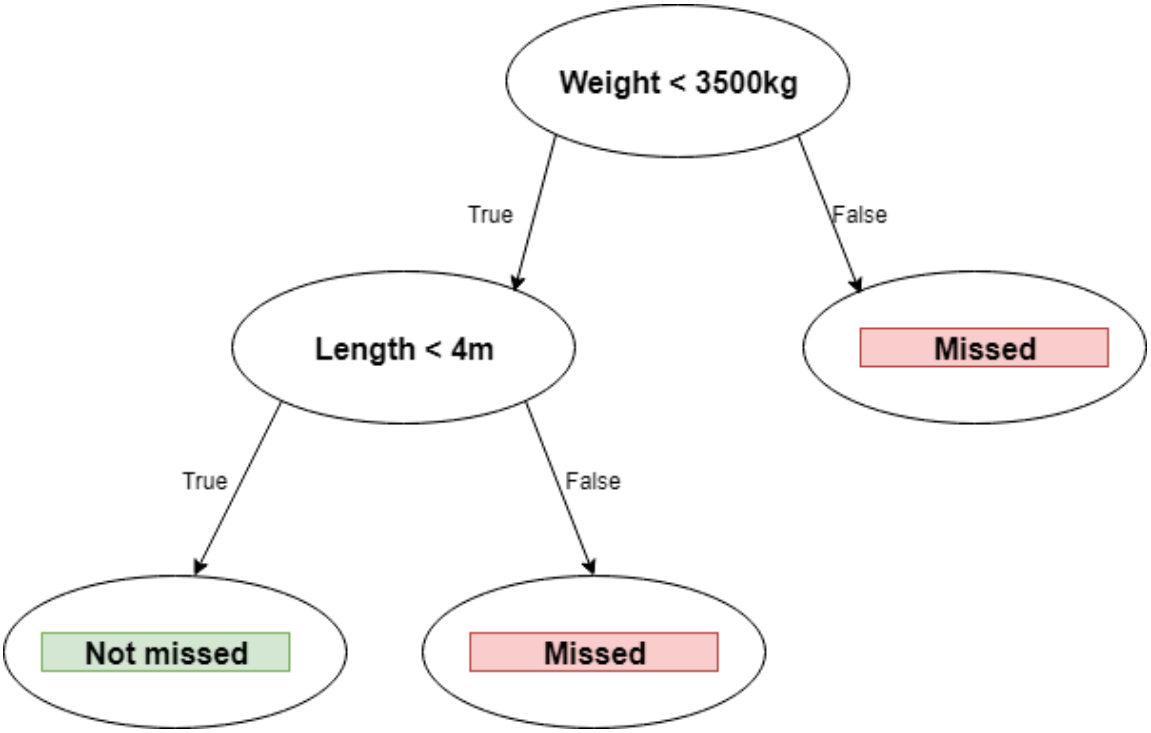

Figure 2: Decision tree classification (dummy data).

features of a dataset[10],[9]. This decision process can be thought of as a series of if-then-else statements that are often used in programming in general. An advantage with Decision Trees is that they are easily visualized as a tree model, see figure 2. Decision trees can lead to overfitting on large datasets with many features, which is one of the disadvantages of this algorithm. Overfitting can be prevented with dimensionality reduction, e.g., Feature Selection[9].

2.1.4 Gaussian Naïve Bayes(GNB)

Naïve Bayes (Gaussian)[11],[9] applies Bayes' theorem on the dataset to specify which class a sample of the dataset belongs to. The algorithm is called naïve because the features are assumed to be independent of each other. The algorithm also assumes that the data is normally distributed hence the name Gaussian Naïve Bayes (Gaussian distribution = Normal distribution).
Bayes' theorem[12] is as follows:

$$P(c \mid d) = \frac{P(d \mid c)\, P(c)}{P(d)}$$

Where P(c) is the probability of the classification being true, also called prior probability. P(d) is the probability of the data regardless of the hypothesis. P(d|c) is the probability of the data given that the classification was true and P(c|d) is the probability of the classification being true given the data and is called posterior probability.

The different probabilities above are calculated with the probability density func-

tion for the Gaussian distribution which looks like this:

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

After the posterior probability has been calculated the algorithm makes a classification decision by comparing the computed probabilities of the classes, and the class with the highest probability is suggested as the predicted class.

2.1.5 C-Support Vector Machine(SVC)

Support Vector Machines are very popular machine learning algorithms used for both regression and classification. The algorithm creates a hyperplane or a set of hyperplanes depending on if its used for binary classification or classifying multiple classes[13],[9]. When used for binary classification, the training samples from the dataset are represented as points in space that are separated by a line. The samples on one side of that line belong to class A, and the samples on the opposite side belong to class B, see figure 3. By analyzing on which side of the line new samples appear the sample is predicted to belong to that class represented on that side. It is effective in datasets with many features but may be sensitive to overfitting if the number of features is more than the number of samples.

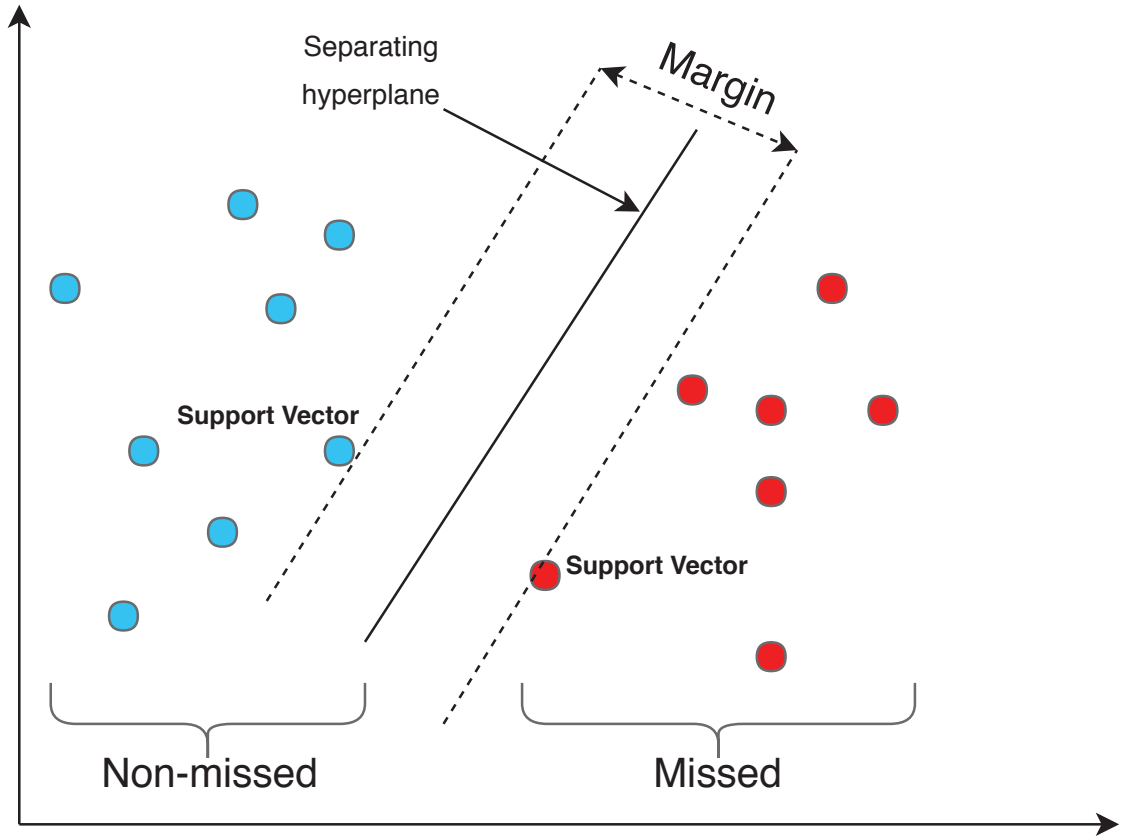

Figure 3: Support Vector Machine.

2.1.6 Random Forest(RFC)

Random Forest is an ensemble algorithm that utilizes perturb-and-combine techniques to create a strong classifier from a set of weak classifiers. The trees in the ensemble of classification trees are created from a sample drawn with replacement from the part of the dataset used for training. The scikit-learn implementation[9] of this algorithm uses the average probabilistic predictions of the classifiers to combine them into one, unlike the original publication[14] which lets each tree vote for the most popular class. The trees in the forest are decision trees, see 2.1.3 for a recap and figure 2 for a visual representation of the trees. The number of trees(estimators) can be changed to any number you would like, the default value in scikit-learn is n_estimators=10.

2.1.7 AdaBoost(ADA)

AdaBoost is a meta-estimator classifier that trains a classifier on the dataset and then makes several copies of the classifier and trains on the same dataset. For each copy, the incorrectly classified samples are weight adjusted to handle more difficult cases. This weight adjustment procedure is a technique referred to as boosting, hence the name *AdaBoost* (Adaptive Boosting). The implementation of AdaBoost in scikit-learn is the variant of this algorithm known as AdaBoost-SAMME[9].

2.1.8 ExtraTreesClassifier

Extremely **Ra**ndomized **Trees** is yet another meta-estimator similar to Random Forest. This algorithm has a second randomized element, in the way how the splitting of a node in the tree is computed. The thresholds are randomly drawn for each feature in the training set and out of these randomly drawn thresholds, the best one is picked as the rule for splitting.

This algorithm can also be used for feature evaluation, and that is the way I have used it. The relative importance of a feature can be interpreted as the depth of a node in the tree. Features that are deeper in the tree is used for making the classification fewer times than features closer to the top.

Chapter 3

Related Work

Bakhtyar and Henesey[8] have done previous studies of machine learning applications in transport and logistics. They investigate how different machine learning algorithms perform compared to each other when predicting the freight type of a transport depending on the origin, destination and weight features. Bakhtyar and Henesey also discuss how to combat missing data by using different techniques such as data imputation.

Metzger et al. discuss the potential efficiency gains in the transport sector by using predictive monitoring. "Predictive monitoring means that critical events, potential deviations, and unplanned situations can be anticipated and proactively managed and mitigated along the execution of business processes" [15]. They do not apply machine learning techniques for their research. However, they do mention that machine learning applications could be leveraged in the transport and logistics area. To do predictive monitoring with favorable accuracy, you need to have a large historical dataset to train machine learning algorithms on. Thanks to WIP I have access to this large quantity of data that Metzger et al. mentions.

Andersson[7] and Lind Nilsson[6] are students from the Luleå University of Technology that wrote their bachelors theses on machine learning in logistics. The project was similar to what I have done, and they were classifying letters and packages to simulate if a letter would be sent to the right address or if it would be incorrect. They investigate the use of machine learning algorithms in the logistics sector and how the performance can be increased with algorithm optimization.

Chapter 4

Method

To make the best use of machine learning algorithms the data that is fed to them must be analyzed and refined. I have used techniques such as data collection, preprocessing of data and computational experiments in this project to retrieve the results. All the following steps were done on my laptop computer with the following system specifications.

1. Intel Pentium Quad-Core processor.
2. 8GB of RAM.
3. Xubuntu 16.04.4 LTS (Xenial Xerxus) 64-bit. - Open source operating system based on Ubuntu[16].
4. DataGrip V.2017.3. - Database management system[17].
5. Python V.3.5.2. - Open source programming language[18].
6. Pandas V.0.22.0. - Python Data Analysis Library[19].
7. NumPy V.1.14.0. - Scientific programming package for Python[20].
8. Matplotlib V.2.1.2. - 2D Plotting library for Python[21].
9. scikit-learn V.0.19.1. - Machine learning in Python[22].

4.1 Manipulation of data

The data was first gathered from a database containing over 30 million rows of transport records. These rows were a dump of the transport database provided by WIP in February 2018. To gather the data I used a powerful database management system called Datagrip[17]. When extracting data from the database, specific rules were set up for what the data should look like, for example, if the status of the transport is *missed* this is represented as the number 1 in the field *new value*, all other values are represented as the number 0 in the field *new value*. This process is done to binarize the data to fit the binary classification problem(true or false).

Next, the data was cleansed of all null-values and incorrect timestamps by defining the SQL-query not to include these values. Null values can occur in the database because of different reasons, e.g., somebody forgot to enter a value into a field, and the default value is null. The timestamps I extract from the database is the difference

in time between pickup and delivery. Incorrect timestamps can occur if somebody had entered a delivery timestamp that would occur before the transport was picked up resulting in a negative time-difference.

All the rows corresponding to the ruleset were saved to a file on my computer as a comma-separated value file(CSV-file). I used the mentioned extracting techniques to create three different datasets. Let's call them small(S), medium(M) and large(L), with 2000, 4000 and 8000 samples respectively where 50% of the samples in the datasets are consignments resulting as *missed* and the other half are *non-missed*.

After the gathering stage, I used Pandas[23] to load the data from the CSV-files into a data frame. The data frame was then split into two, one containing all the features to be evaluated and one containing all the target labels. Next, I used the *model_selection* module from scikit-learn to split the data into training and testing(validation) data. I used the function train_test_split to split the dataset into two parts, 70% of the whole dataset to be used for training and the remaining 30% for testing, all the samples are randomly drawn and shuffled. This function splits the data into a subset used for training the machine learning algorithms and one subset used for testing the model upon for it to be exposed to entirely new data it did not include in the training.

4.2 Feature Selection

During this part of the method, you need to choose which features to use as input to the machine learning algorithm. The database I gathered the data from had over 60 different tables, and it was hard to determine which features might have a significant impact on whether a consignment will be classified as missed or not. I talked to the transport database administrator at WIP and also the people from my time in the group project, and we discussed which features might be useful to extract. One could argue that attributes such as the weight, volume, length and height of a transport vehicle, the package itself or both could have a significant impact on whether a consignment will reach the expected destination or not. Those attributes were included in the dataset, as well as information about the consignments departure and destination cities and more. I chose to exclude things that I did not think had any meaning in determining the state of a consignment, e.g., the driver's phone number and home address.

After I had picked out a number of features that I thought were interesting to include in the classification, I evaluated the weight of each feature with a Feature Selection algorithm called ExtraTreesClassifier 2.1.8. I imported it from scikit-learns module *sklearn.ensemble* and trained the algorithm on the dataset. Lastly, I extracted the features importances from the trained model. I could then analyze these importances with the goal of finding which of the features could be excluded and which were beneficial to keep.

See figure 4, on the Y-axis, is the number and name of the feature, and the X-axis represents the features importances with the most important feature on the bottom. The feature names are the exact names from the database, and most of them should be self-explanatory except those with *flak* and *pall* in the name, *flak* means truck bed and *pall* means pallet. I chose to exclude the 9 features in the top of figure 4, since

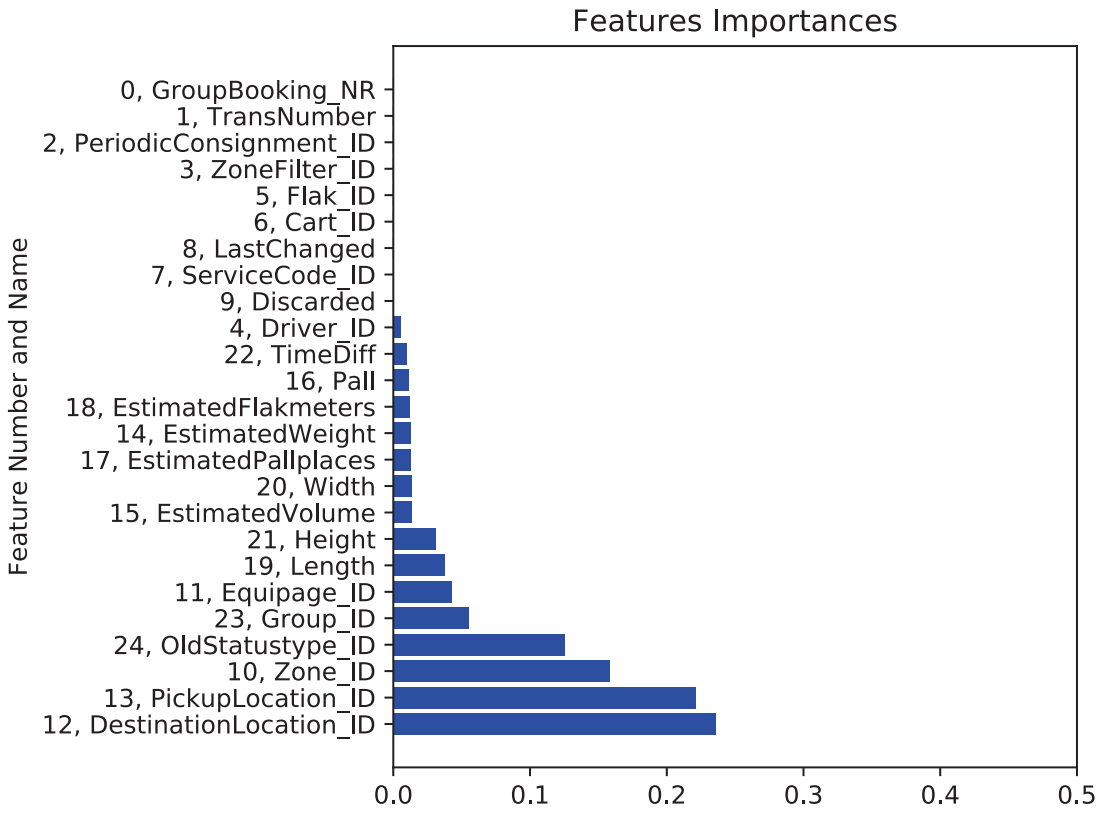

Figure 4: Feature weights.

they didn't have a large impact on the result. Features F_{12}, F_{13}, F_{10}, F_{24}, F_{23}, F_{11}, F_{19}, F_{21}, F_{15}, F_{20}, F_{17}, F_{14}, F_{18}, F_{16}, F_{22} and F_4 were then used as input variables during the training and testing stage.

4.3 Model training and testing

As in the previous stage I used scikit-learn's model_selection module again, to arrange all the models with K-Fold cross-validation(CV) set to 10-folds, see figure 5 for a visualization of this technique.

This is done to split the data even further into subsets for training and testing, and it creates subsets called folds according to the parameter K, in this case, 10. So the training subset is split into K=10 folds where each fold is used once for validation(testing) and the K-1 folds that remains are used for training. When each fold has been used for validation, the result of the cross-validation is the average of the values computed for each iteration. To summarize, the CV procedure uses the training subsets to create a classifier and the testing subset which the classifier has not seen before, to validate the performance of the classifier. The reason for using CV is to prevent overfitting and to create a more generalized model.

The models were then trained and tested on the folds produced in the cross-validation, and the results were stored in Python lists that were used to plot the results.

The saved results from the cross-validation procedure were the mean value of the following metrics: recall(RC) and accuracy(ACC). These metrics was calculated with scikit-learn's *cross_val_score* function from the *model_selection* module, where the metric you want to extract is sent as an argument to the function e.g. scoring='accuracy'.

I also collected confusion matrices for all models. From the confusion matrices, I was able to extract true positives(TP), true negatives(TN), false positives(FP) and false negatives(FN). The datasets S, M, L were trained and tested individually for the sake of investigating how the results varied with larger or smaller datasets.

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN}$$

4.4 Constructing results

When the results had been generated from the calculations during the previous phase, I visualized the results that were stored in the previously mentioned Python lists. I used the pyplot module from matplotlib[21] to plot the results in the form of boxplots into graphs and to create the barplot representing the features importances. These graphs were then saved to my computer. With the use of a bash script, I redirected the output of my program to a text file, so-called piping. At last, these graphs and text files were used during the analysis and presentation of the results.

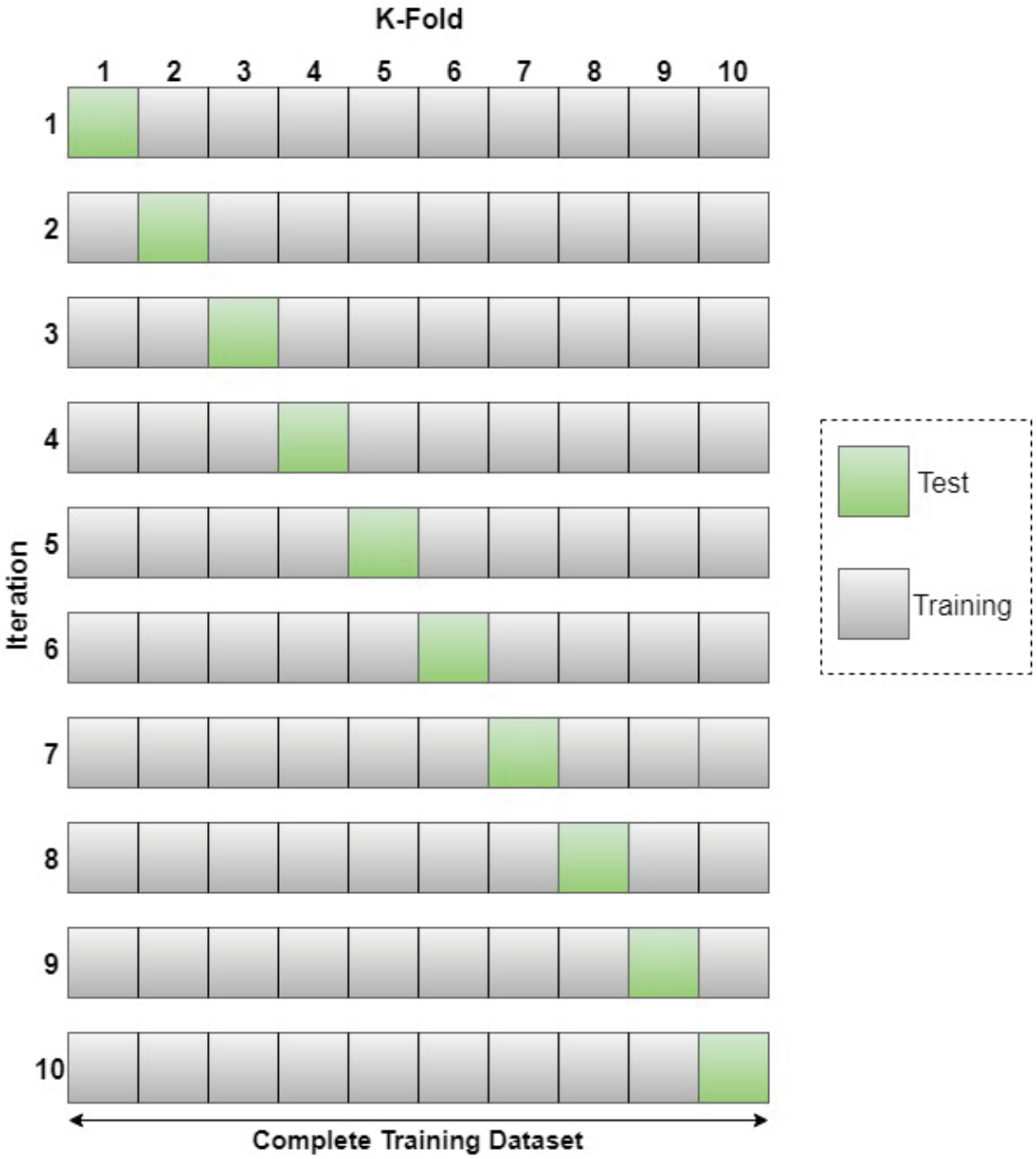

Figure 5: K-Fold Cross Validation.

Chapter 5

Results

In this chapter, I will present the results obtained during my research, and I will discuss the results in the next chapter.

Firstly, as shown above, figure 4 presents the importances of each feature included in the original dataset, then the following tables present the performance of each algorithm on each iteration of the cross-validation procedure. Each dataset has a separate table for recall evaluation and accuracy evaluation. The first column is the iteration number in the cross-validation process, the rest of the columns represent the performance of that specific algorithm on that iteration.

I have chosen to highlight which algorithm performed the best on average based on each dataset, and this is shown in red color in the field mean. Each algorithm also has a bold-styled entry in the table, and that is to highlight on which iteration this algorithm scored highest. You can see the figures below as well that visualizes the tables mean results in boxplots, how to interpret them is displayed in figure 6.

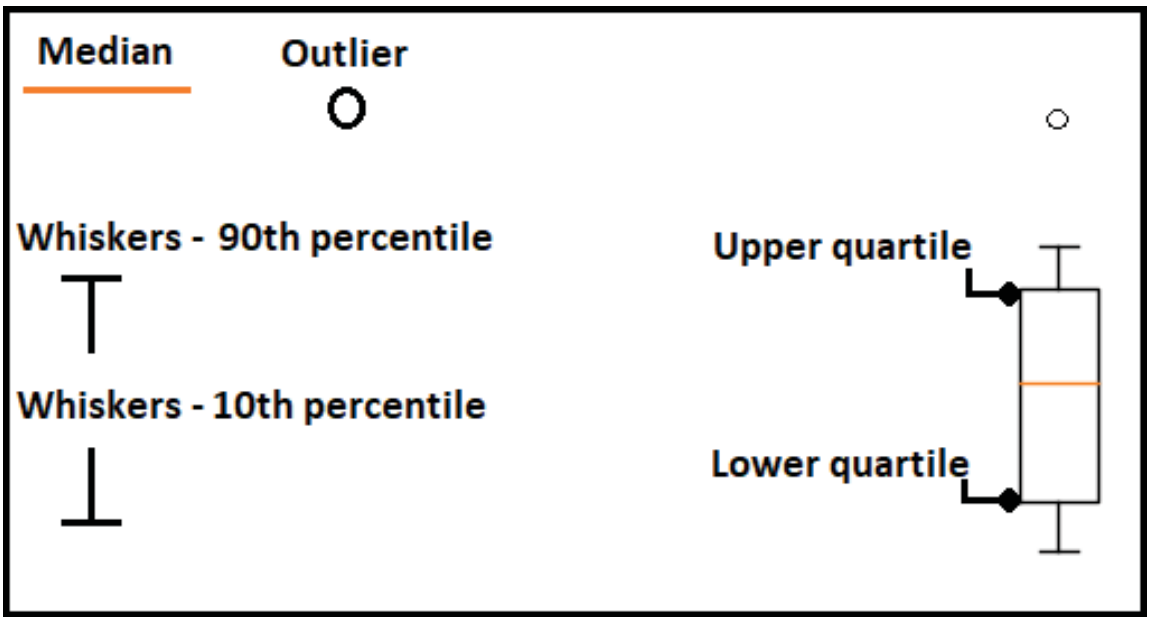

Figure 6: Boxplot legend.

CV Nr	LR	KNN	DCC	GNB	SVC	RFC	ADA
1	0.271429	0.985714	0.985714	0.985714	0.985714	0.985714	0.985714
2	0.516129	0.967742	0.967742	0.983871	**1.000000**	0.967742	0.967742
3	0.405797	0.942029	0.942029	0.942029	0.971014	0.942029	0.942029
4	0.403226	**1.000000**	**1.000000**	**1.000000**	1.000000	**1.000000**	**1.000000**
5	**0.533333**	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
6	0.522388	0.985075	1.000000	0.985075	1.000000	0.985075	1.000000
7	0.378788	0.939394	0.939394	0.939394	0.969697	0.939394	0.939394
8	0.240000	0.986667	1.000000	0.973333	1.000000	0.986667	1.000000
9	0.214286	0.976190	0.988095	0.976190	0.988095	0.976190	0.976190
10	0.263889	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
mean	0.374926	0.978281	0.982297	0.978561	**0.991452**	0.978281	0.981107

Table 1: Recall by each CV iteration and mean recall value of each algorithm for dataset S.

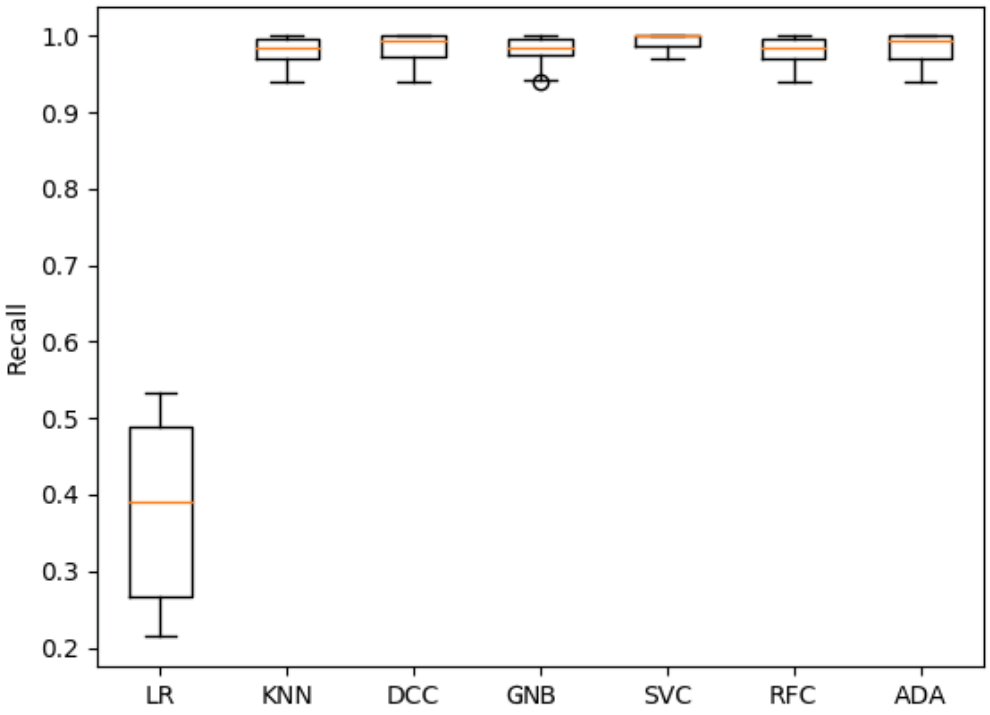

Figure 7: Algorithms recall for small dataset.

CV Nr	LR	KNN	DCC	GNB	SVC	RFC	ADA
1	0.635714	0.992857	0.992857	0.992857	0.864286	0.992857	0.985714
2	**0.692857**	0.985714	0.985714	0.985714	0.928571	0.985714	0.985714
3	0.578571	0.971429	0.971429	0.964286	0.914286	0.964286	0.964286
4	0.557143	**1.000000**	0.985714	**1.000000**	0.9214285	**1.000000**	**1.000000**
5	0.621429	1.000000	**1.000000**	1.000000	0.9000000	1.000000	0.992857
6	0.650000	0.992857	1.000000	0.992857	0.928571	0.992857	1.000000
7	0.557143	0.971429	0.971429	0.971429	0.914286	0.971429	0.971429
8	0.592857	0.992857	0.992857	0.978571	0.964286	0.992857	0.992857
9	0.528571	0.985714	0.985714	0.985714	0.964286	0.985714	0.985714
10	0.607143	1.000000	0.985714	1.000000	**0.978571**	1.000000	0.992857
mean	0.602143	**0.989286**	0.987143	0.987143	0.927857	0.987143	0.987143

Table 2: Accuracy by each CV iteration and mean accuracy value of each algorithm for dataset S.

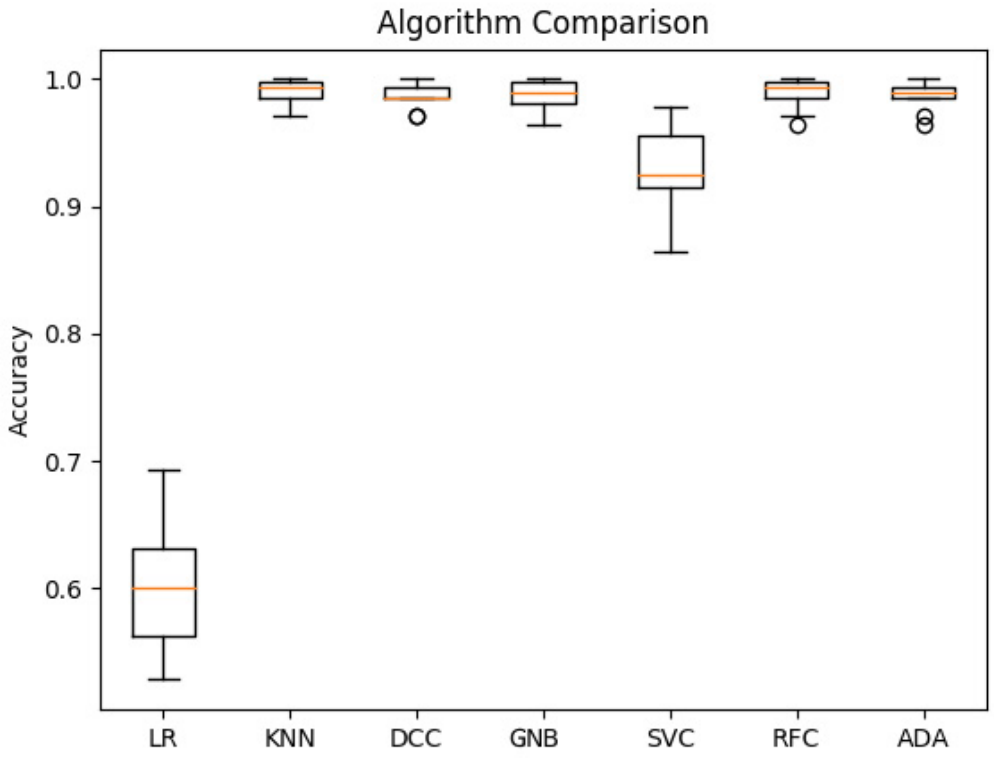

Figure 8: Algorithms accuracy for small dataset.

	LR	KNN	DCC	GNB	SVC	RFC	ADA
1	0.578571	0.992857	0.992857	0.992857	**1.000000**	0.992857	0.992857
2	0.585714	0.992857	0.992857	0.992857	1.000000	0.992857	0.992857
3	0.558621	**1.000000**	**1.000000**	**1.000000**	1.000000	0.993103	**1.000000**
4	0.561644	0.993151	1.000000	1.000000	1.000000	0.986301	0.993151
5	**0.591241**	1.000000	1.000000	1.000000	1.000000	**1.000000**	1.000000
6	0.549296	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
7	0.588652	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
8	0.557143	1.000000	1.000000	1.000000	1.000000	0.992857	1.000000
9	0.547945	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
10	0.565517	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
mean	0.568434	0.997886	0.998571	0.998571	**1.000000**	0.995798	0.997886

Table 3: Recall by each CV iteration and mean recall value of each algorithm for dataset M.

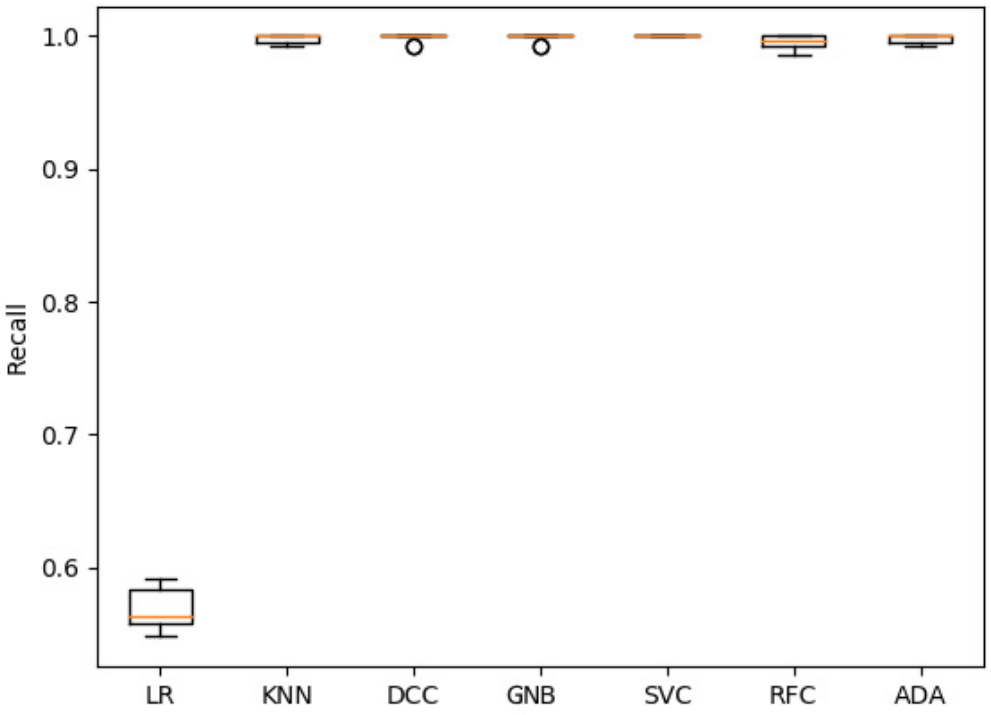

Figure 9: Algorithms recall for medium dataset.

CV Nr	LR	KNN	DCC	GNB	SVC	RFC	ADA
1	0.646429	0.996429	0.996429	0.982143	0.921420	0.996429	0.996429
2	0.689286	0.996429	0.996429	0.989286	0.928571	0.996429	0.996429
3	0.660714	**1.000000**	**1.000000**	0.996429	**0.946429**	0.996429	**1.000000**
4	0.635714	0.996429	0.996429	**1.000000**	0.907143	0.992857	0.996429
5	0.689286	1.000000	1.000000	0.989286	0.910714	**1.000000**	1.000000
6	0.646429	1.000000	1.000000	0.996429	0.910714	1.000000	1.000000
7	**0.707142**	1.000000	1.000000	0.985714	0.935714	1.000000	1.000000
8	0.650000	1.000000	0.996429	0.996429	0.878571	0.996429	1.000000
9	0.678571	1.000000	1.000000	0.996429	0.928571	1.000000	1.000000
10	0.635714	1.000000	1.000000	0.992857	0.910714	1.000000	1.000000
mean	0.663929	**0.998929**	**0.998929**	0.992500	0.917857	0.998571	**0.998929**

Table 4: Accuracy by each CV iteration and mean accuracy value of each algorithm for dataset M.

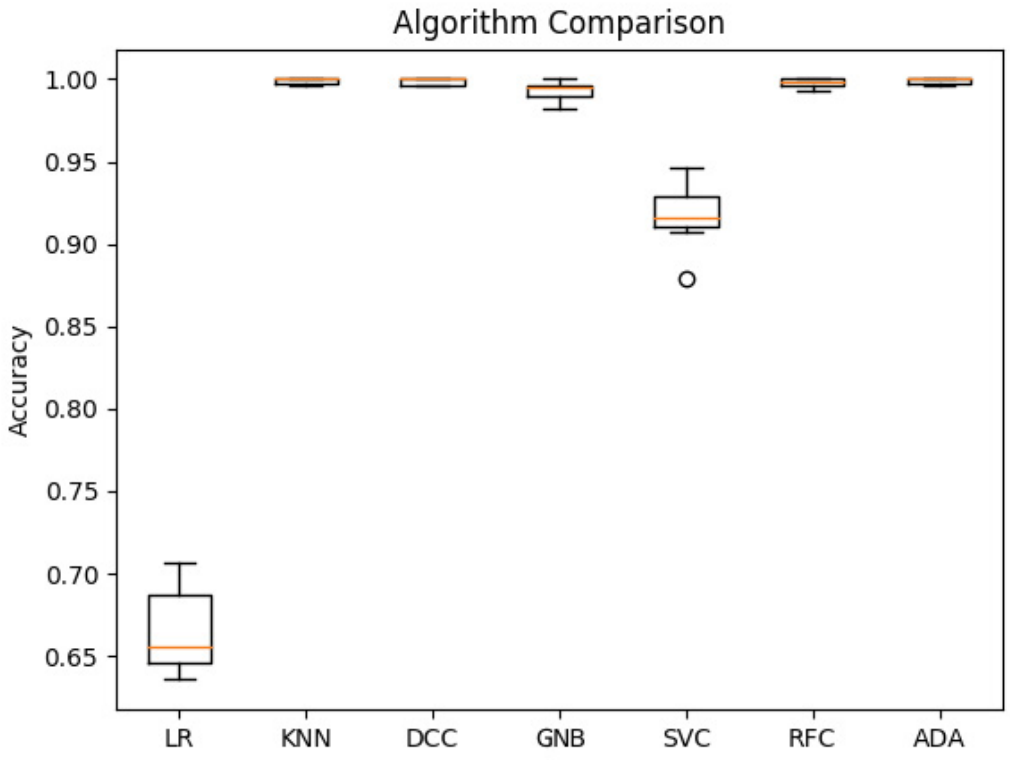

Figure 10: Algorithms accuracy for medium dataset.

CV Nr	LR	KNN	DCC	GNB	SVC	RFC	ADA
1	**0.621908**	0.968198	0.975265	0.971731	0.978799	0.978799	0.975265
2	0.516129	0.992832	**1.000000**	0.989247	0.996416	0.996416	**1.000000**
3	0.491289	0.972125	0.982578	0.979094	0.986063	0.979094	0.979094
4	0.612676	0.985915	0.996479	0.989437	**1.000000**	**0.996479**	0.996479
5	0.524823	0.992908	0.996454	**0.992908**	0.996454	0.992908	0.996454
6	0.565371	0.982332	0.992933	0.982332	0.992933	0.989399	0.982332
7	0.573643	0.984496	0.984496	0.980620	0.988372	0.984496	0.984496
8	0.485714	0.978571	0.978571	0.978571	0.992857	0.978571	0.982143
9	0.540070	**0.993031**	1.000000	0.986063	0.993031	0.993031	0.993031
10	0.526502	0.985866	0.989399	0.982332	0.989399	0.989399	0.992933
mean	0.545813	0.983627	0.989618	0.9832336	**0.991432**	0.987859	0.988223

Table 5: Recall by each CV iteration and mean recall value of each algorithm for dataset L.

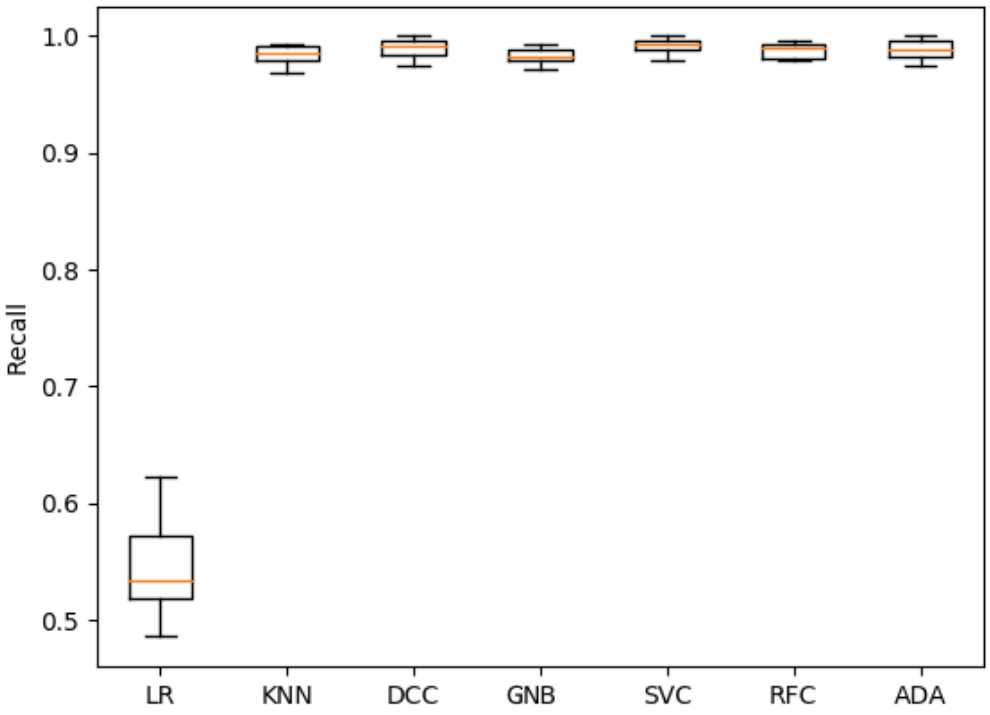

Figure 11: Algorithms recall score for the large dataset.

CV Nr	LR	KNN	DCC	GNB	SVC	RFC	ADA
1	0.678571	0.983929	0.983929	0.985714	0.885714	0.991071	0.985714
2	0.646429	**0.994643**	0.994643	0.994643	0.900000	**0.998214**	**1.000000**
3	0.637500	0.985714	0.989286	0.989286	0.905357	0.987500	0.989286
4	**0.687500**	0.992857	**0.998214**	0.994643	0.919643	0.998214	0.996429
5	0.642857	0.992857	0.992857	**0.996429**	0.907143	0.994643	0.998214
6	0.667857	0.989286	0.985714	0.989286	0.892857	0.985714	0.991071
7	0.660714	0.991071	0.987500	0.989286	0.878571	0.992857	0.992857
8	0.623214	0.987500	0.982143	0.989286	0.919643	0.989286	0.987500
9	0.616071	0.989286	0.994643	0.987500	**0.928571**	0.992857	0.994643
10	0.635714	0.992857	0.987500	0.991071	0.900000	0.991071	0.996429
mean	0.649643	0.989999	0.989643	0.990714	0.903750	0.992500	**0.993214**

Table 6: Accuracy by each CV iteration and mean accuracy value of each algorithm for dataset L.

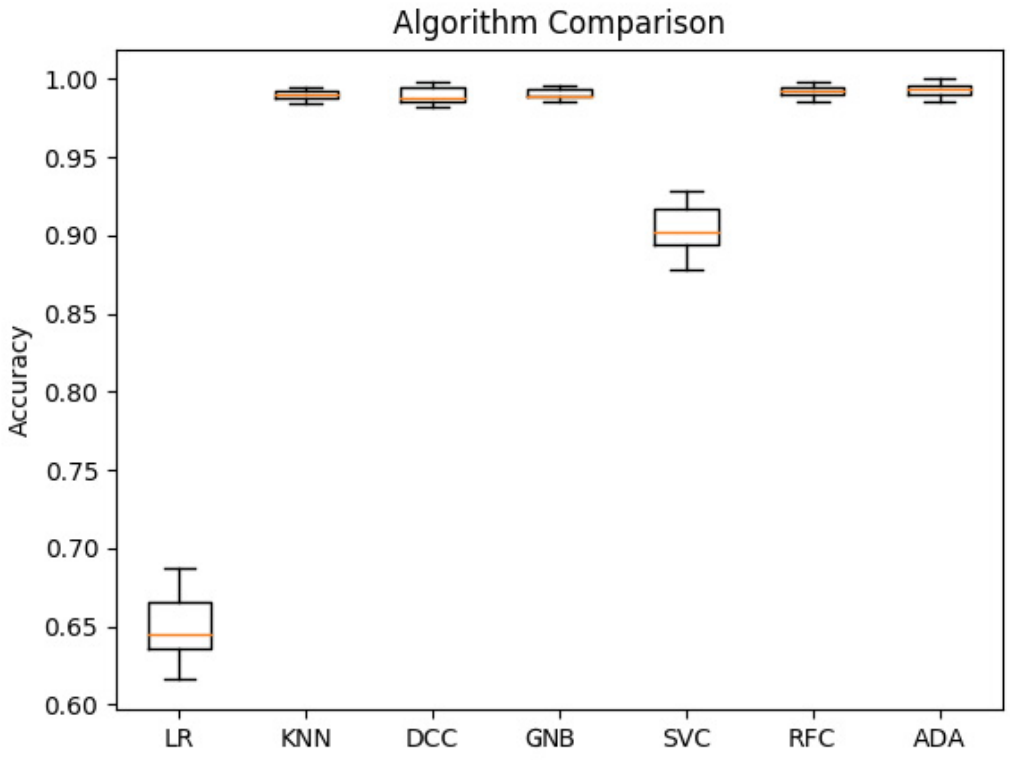

Figure 12: Algorithms accuracy score for the large dataset.

Chapter 6

Analysis and Discussion

By analyzing the tables and figures presented in the previous chapter, we can see that the scores are very close for most algorithms on both metrics across all three datasets.

Starting with the accuracy tables first, on dataset S, the algorithm KNN has the highest ACC score while tightly followed by RFC, ADA, DCC, GNB which all have the same score, next comes SVC and last LR, see table 2 and figure 8.

When trained on dataset M, once again KNN is in the top of the score together with DCC and ADA which all three have the same score. RFC and GNB both fall, and then there is SVC and LR at the end again, see table 4 and figure 10.

The last table 6 and figure 12 presents that ADA has the highest score now followed by RFC and GNB. KNN has now dropped to the 4th place and next comes DCC, once again SVC and LR perform worst.

Now let's see the recall tables, table 1 presents SVC in the top, followed by DCC then ADA. Next is GNB then KNN and RFC with the same score and last, LR has the lowest score.

The scores for dataset M is SVC with a perfect score on all cross-validation iterations, followed by DCC and GNB with the same score. On place 4 and 5 is KNN and ADA with the same score, placed sixth is RFC and last is LR.

On the large dataset, we can see in table 5 that SVC once again performs best followed by DCC. ADA, RFC, KNN, GNB comes on 3rd, 4th, fifth and sixth place respectively with LR at the end once more.

It seems like the size of the dataset does impact the score of each algorithm. The one algorithm that sticks out most is Logistic Regression, with the lowest score throughout all of the tables. Interesting to see is that all algorithms except SVC overall increases ACC score when trained on larger datasets compared to the dataset called S. Support vector machines are known not to handle large data quantities well, but mostly due to longer training times. When evaluating recall, all of the algorithms increase their score when used on larger datasets and SVC followed by DCC has the highest recall across all datasets. We can see that SVC has a high recall on all datasets, but lower accuracy than most other algorithms, this tells us that this algorithm has many false positives, it does classify most of the correct samples to be *missed* but also classifies samples that are not.

In contrast to Henesey and Bakthyar[8], I have chosen not to use ROC AUC as a metric of evaluation in this study. The reason is that recent papers have discussed

that this metric appears not to be as good of a metric as previous researchers have thought. ROC AUC weighs omission and commission rates the same and also uses separate cost distributions for misclassification on different classifiers[24],[25]. This introduces problems when comparing algorithms to one another, and that is the reason why I chose to use accuracy and recall as evaluation metrics.

The reason why I chose to create datasets of the sizes S=2000, M=4000 and L=8000 is that there existed only 8000 pure samples of *missed* consignments in the database, with pure samples I mean that they fulfilled the criteria I set up when extracting these from the database. When I learned this fact I just split the total amount of *missed* consignments in half and a quarter. Each dataset was as mentioned in chapter 4, split into 50% *missed* and 50% *non-missed* because it would provide balance in the data which is essential to reduce the bias of the classifiers.

When analyzing the features importances we can see in figure 4 that F_{10}, F_{23} and F_{11} has higher importance than F_{15}, F_{20}, F_{17}, F_{14}, F_{18}, F_{16}. The first three, *Zone_ID*, *Group_ID* and *Equipage_ID* are features more related to human factors than latter ones which are more related to the vehicle itself, this indicates that human error can be a part of the problem why a transport is missed.

Chapter 7

Conclusions and Future Work

I think that machine learning can be utilized in a real-world application, for example, the logistics management system from where the data I used in my study came. For example, this can be used to make a faster decision on which vehicle should be used for a specific consignment. Most of the algorithms that I have analyzed perform equally well, the ones that stand out in this experiment is Logistic Regression which performed poorly, and Support Vector Machine that had more false positives than the other well-performing classifiers. For future work, a closer study could be done on only DCC, GNB, KNN, RFC, and ADA. Other effects could also be taken into consideration that I did not include in this analysis, i.e., how time-consuming a particular algorithm is during training and prediction, hyper-parameter-optimization or to include and analyze other features than the ones I included.

My conclusion is that training machine learning algorithms on a larger dataset produce better results overall and that there is an indication that the *missed* consignments are connected to human-errors but to know for sure, you would need to find more evidence other than just looking at which features are more important than others. There is a difference between the algorithms I have studied, and in five out of cases the differences are minimal, in one of the cases the difference is small, and in one case the difference is considerable. In this specific problem, I could choose any of the five algorithms that perform well to be my classifier or to further examine them based on other benchmarks.

References

[1] Svenskt Näringsliv, "Transport och logistik - en spjutspets för konkurrenskraft." https://www.svensktnaringsliv.se/material/rapporter/transport-och-logistik-en-spjutspets-for-konkurrenskraft_534534.html, 2005. [Online; accessed 2018-02-02].

[2] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[3] F. J. Hastie T., Tibshirani R., *The Elements of Statistical Learning*, ch. 14. Springer, New York, NY, 2009.

[4] R. S. Sutton and A. G. Barto, "Reinforcement learning:: An introduction," *Kybernetes*, vol. 27, no. 9, pp. 1093–1096, 1998-12-01.

[5] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, p. 354, 2017-10-18.

[6] R. Lind Nilsson, "Machine learning in logistics : Increasing the performance of machine learning algorithms on two specific logistic problems," 2017.

[7] V. Andersson, "Machine learning in logistics: Machine learning algorithms : Data preprocessing and machine learning algorithms," 2017.

[8] S. Bakhtyar and L. Henesey, "Freight transport prediction using electronic waybills and machine learning," in *Proceedings 2014 International Conference on Informative and Cybernetics for Computational Social Systems (ICCSS)*, pp. 128–133, 2014-10.

[9] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

[10] scikit-learn team, "Decision tree classifier on scikit-learn website." http://scikit-learn.org/stable/modules/tree.html#tree, 2017. [Online; accessed 2018-06-05].

[11] scikit-learn team, "Gaussian naive bayes on scikit-learn website." http://scikit-learn.org/stable/modules/naive_bayes.html#naive-bayes, 2017. [Online; accessed 2018-06-05].

[12] M. Kendall, A. Stuart, J. K. Ord, and A. O'Hagan, "Kendall's advanced theory of statistics, volume 1: Distribution theory," *Arnold, sixth edition edition*, 1994.

[13] scikit-learn team, "Support vector machine on scikit-learn website." http://scikit-learn.org/stable/modules/svm.html#support-vector-machines, 2017. [Online; accessed 2018-06-05].

[14] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001-10.

[15] A. Metzger, R. Franklin, and Y. Engel, "Predictive monitoring of heterogeneous service-oriented business networks: The transport and logistics case," in *2012 Annual SRII Global Conference*, pp. 313–322, July 2012.

[16] The Xubuntu team, "Xubuntu website." https://xubuntu.org/, 2012. [Online; accessed 2018-04-11].

[17] JetBrains s.r.o, "Jetbrains datagrip webpage." https://www.jetbrains.com/datagrip/, 2000. [Online; accessed 2018-04-03].

[18] Python Developers, "Python website." https://www.python.org/, 2001. [Online; accessed 2018-05-12].

[19] Pandas Team, "Pandas website." https://pandas.pydata.org/, 2009. [Online; accessed 2018-05-12].

[20] NumPy developers, "Numpy website." http://www.numpy.org/, 2018. [Online; accessed 2018-05-12].

[21] Matplotlib development team, "Matplotlib website." https://matplotlib.org/, 2002. [Online; accessed 2018-05-12].

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] W. McKinney, "pandas: a foundational python library for data analysis and statistics," 2011.

[24] D. J. Hand, "Measuring classifier performance: a coherent alternative to the area under the ROC curve," *Machine Learning*, vol. 77, no. 1, pp. 103–123, 2009.

[25] J. Lobo, A. Jiménez-Valverde, and R. Real, "AUC: a misleading measure of the predictive distribution models.," *Global Ecology and Biogeography*, vol. 17, 2008.

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden