Classification based on smartphone sensor data

Mikko Seppi

School of Electrical Engineering

Bachelor's thesis
Espoo 05.05.2021

Supervisor

Dr. Samuli Aalto

Advisor

Prof. Esa Ollila

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the bachelor's thesis

Author Mikko Seppi

Title Classification based on smartphone sensor data

Degree programme Bachelor's Programme in Electrical Engineering

Major Information Technology **Code of major** ELEC3015

Teacher in charge Dr. Samuli Aalto

Advisor Prof. Esa Ollila

Date 05.05.2021 **Number of pages** 31 **Language** English

Abstract

Today's smartphones include many kinds of sensors that produce large amounts of data. This data can be used in different machine learning applications. One of these applications is human activity recognition (HAR), where sensor data for example can be used for transportation mode detection. In previous studies, different smartphone sensors have been used for transportation mode detection. One sensor, which has proven to be effective for this purpose is accelerometer, because it is energy efficient and has obtained good accuracies in this task.

This thesis investigates how well two state-of-the-art classification methods perform in transportation mode detection based on smartphone accelerometer data. Data used in this thesis contains pre-processed magnitude of the accelerometer values from seven transportation modes. These modes are driving a car, taking a bus, taking a subway, taking a train, bicycling, walking and being stationary.

The classification methods which were used for transportation mode detection were random forest and boosting. Random forest is an ensemble classification method that uses many decision trees with randomized number of features and randomized cases (observations) in the training set to perform the classification task. Boosting methods are ensemble classifiers that take the majority vote of many weak learners in their classification rule. Boosting method that is used in this thesis is called as adaptive boosting or AdaBoost.

The AdaBoost classifier performed slightly better in transportation mode detection than the random forest classifier with the accuracy of 92.357 %. The random forest classifier produced classification with the accuracy of 91.600 %. However, the performance differences were rather marginal. Both models classified the non-motorized transportation modes better than the motorized modes and the most misclassification happened between the motorized transportation modes. This why the point of development was thought to be the discovery of new features that would improve the classification accuracy of the motorized transportation modes.

Keywords machine learning, classification, human activity recognition, transportation mode detection, random forest, AdaBoost, accelerometer

Aalto University
School of Electrical Engineering

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Tekniikan kandidaatintyön tiivistelmä

Tekijä Mikko Seppi

Työn nimi Luokittelu älypuhelimen sensoridatan perusteella

Koulutusohjelma Sähkötekniikan kandidaattiohjelma

Pääaine Informaatioteknologia **Pääaineen koodi** ELEC3015

Vastuuopettaja TkT. Samuli Aalto

Työn ohjaaja Prof. Esa Ollila

Päivämäärä 05.05.2021 **Sivumäärä** 31 **Kieli** Englanti

Tiivistelmä

Älypuhelimissa on nykyään monia sensoreita, kuten GPS, gyroskooppi ja kiihtyvyysanturi, jotka tuottavat suuria määriä dataa. Tätä dataa voidaan käyttää esimerkiksi erilaisiin koneoppimistehtäviin, kuten käyttäjän toiminnan tunnistamiseen, jossa sensoridatan perusteella voidaan määrittää esimerkiksi käyttäjän kulkumuoto. Kulkumuodon tunnistamisesta on paljon aikaisempaa tutkimusta, joissa on käytetty eri älypuhelimien sensoreita kulkumuodon tunnistamiseen. Yksi hyväksi havaittu sensori on kiihtyvyysanturi. Sen avulla on saavutettu hyviä tarkkuuksia, ja se on energiatehokkaampi sensori kuin esimerkiksi GPS, jota on myös käytetty kulkumuodon tunnistamiseen useissa tutkimuksissa.

Tässä kandidaatintyössä esitellään kaksi luokittelumetodia, sekä testataan, kuinka hyvin nämä metodit pystyvät tunnistamaan käyttäjän kulkumuodon älypuhelimen kiihtyvyysanturin datan perusteella. Työssä käytetty data sisältää esikäsiteltyä älypuhelimen kiihtyvyysanturin dataa mitattuna seitsemästä eri kulkumuodosta. Nämä kulkumuodot ovat auto, linja-auto, pyörä, kävely, metro, juna ja paikallaan oleminen. Datan esikäsittelyssä mittausdatasta oli poistettu painovoiman vaikutus sekä osat, jolloin kiihtyvyydessä on tapahtunut nopeita muutoksia. Nopeita muutoksia kiihtyvyydessä tapahtuu esimerkiksi silloin, kun käyttäjä ottaa älypuhelimen taskusta käteensä. Lopulliset kiihtyvyys anturin arvot, joita käytettiin kulkumuodon tunnistamiseen, olivat kiihtyvyyden suuruuksia eri ajan hetkillä. Näistä arvoista muodostettiin 10,24 sekunnin ikkunoita 1,28 sekunnin liu'ulla, jotta kulkumuodon luokittelu saataisiin tehtyä reaaliaikaiseksi. Näistä ikkunoista lopulta muodostettiin piirteet luokittelua varten.

Luokittelumetodit, joilla kulkumuodon tunnistusta kokeiltiin, olivat satunnaismetsä (engl. Random forest) ja tehostaminen (engl. Boosting). Satunnaismetsä on luokittelualgoritmi, joka käyttää useaa satunnaistettua päätöspuuta luokittelun tekemiseksi. Luokittelu tehdään enemmistöäänestyksellä: jokainen päätöspuu valitsee mieleisensä luokan ja eniten ääniä (valintoja) saanut luokka on lopullisen luokittelun tulos.

Tehostus luokittelumetodit ovat luokittelijoita, jotka käyttävät useaa heikkoa oppijaa (esim. päätöspuu), luokittelun tekemiseen. Boostaus-algoritmi, jota tässä työssä käytettiin, oli mukautuvan tehostamisen menetelmä (engl. adaptive boosting (AdaBoost)). AdaBoost-algoritmissa opetetaan peräkkäin useaa heikkoa oppijaa, siten että näytteet, jotka on luokiteltu heikosti, saavat aina isomman painoarvon seuraavan heikon oppijan opetukseen. Näin algoritmi pystyy keskittymään paremmin näytteisiin, jotka se luokittelee heikosti. Lopullinen luokittelu tehdään enemmistöäänestyksellä samoin kuin satunnaismetsässä, mutta heikot oppijat, jotka tuottivat opetuksessa pienemmän virheen, saavat isomman painoarvon äänestyksessä.

Kulkumuodon tunnistuksessa AdaBoost-algoritmi onnistui hieman paremmin kuin satunnaismetsäalgoritmi. Adaboost-algoritmi luokitteli kuljetusmuodot 92,357 %:n tarkkuudella, kun taas satunnaismetsä onnistui tässä 91,600 %:n tarkkuudella. Tulos oli kuitenkin hyvin samankaltainen molemmilla luokittelumetodeilla. Molemmat luokittelivat moottorittomat kulkumuodot paremmin kuin moottorilliset. Eniten vääriä luokitteluja tapahtui moottorillisten kulkumuotojen kesken. Tämän vuoksi kehityskohdaksi katsottiin sellaisten uusien piirteiden löytäminen, jotka erottaisivat paremmin moottoriset kulkumuodot.

Avainsanat koneoppiminen, luokittelu, kulkumuodon tunnistus, satunnaismetsä, AdaBoost, kiihtyvyysanturi

Contents

1 Introduction

Modern smartphones include many kinds of built-in sensors such as accelerometer, proximity sensors, gyroscope and many more. The sensors provide a large amount of data that can be used for various machine learning tasks which makes possible many useful applications. One of these applications is human activity recognition (HAR) where sensor data is used for tasks such as detection if the person is sitting, running, walking or standing.

Another application of HAR using smartphone sensor data is transportation mode detection which aims to recognize what transportation mode a person is using. This information is useful because in urban transportation planning knowledge of people's transportation mode is important. This information can be used to improve the performance of tracking and positioning systems. It can also be used to provide real-time journey planning and customized trip suggestions or to provide targeted and customized advertisements and services.[1]

Transportation mode detection as a subfield of HAR is a widely studied field. Some previous studies have used GPS sensor data for transportation mode detection [2], [3]. However, the problem with GPS sensor is that it consumes a significant amount of phone battery, and GPS signal may not be available everywhere. Other studies have used smartphone accelerometer data [1], [4]. Using accelerometer is energy efficient and it has produced good results for detecting transportation modes.

In this thesis, I investigate two state-of-the-art classification methods, random forest and boosting, and how well they perform in transportation mode detection based on smartphone accelerometer data. Random forest is an ensemble classification method that uses many decision trees with the randomized number of features and randomized cases (observations) in the training set to perform the classification task. Boosting methods are ensemble classifiers that take the majority vote of many weak learners in their classification rule. Boosting method that is used in this thesis is called as adaptive boosting or AdaBoost.

The smartphone sensor data used in transportation mode detection in this thesis is obtained from a study "A Deep Learning Model for Transportation Mode Detection Based on Smartphone Sensing Data" made by Liang et al. [1]. The data contains pre-processed magnitude of the accelerometer values from seven transportation modes. These modes are driving a car, taking a bus, taking a subway, taking a train, bicycling, walking and being stationary.

The aim of this thesis is to investigate how well some state-of-the-art classification methods perform in HAR classification problem using real-world data. The second chapter explains the basics of machine learning, classification and evaluation of the classification algorithm. The third chapter presents two state-of-the-art classification methods: boosting and random forests. The last chapter provides the results of data analysis, where methods presented in the third chapter are used in transportation mode detection based on smartphone sensor data.

2 Basics and concepts

In this chapter, the basics of machine learning, classification, and evaluation of classification algorithms are presented. First is briefly introduced what machine learning and classification are. Then basic concepts cross-validation, overfitting and underfitting, and feature engineering are presented. Lastly is introduced ways to evaluate classification algorithms.

2.1 Machine learning

According to Mohammed et al. [5] "Machine learning is a branch of artificial intelligence that aims at enabling machines to perform their jobs skillfully by using intelligent software". In other words, machine learning studies algorithms that have ability to improve through experience and perform tasks such as prediction and classification. Machine learning is usually divided into three different areas: reinforcement learning, unsupervised learning and supervised learning. In the reinforcement learning, the machine aims to use observations from the environment to take actions that will maximize the reward or minimize the risk. In the unsupervised learning, machine tries to find a hidden pattern from unlabelled data.[5] Aim of the supervised learning is to learn from past information. In supervised learning, the model parameters are estimated using training data. The training data consists of N input-output pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where the inputs $\mathbf{x}_i$ are p-dimensional vectors consisting of measurements on p feature variables and the outputs y_i are the associated outcomes or responses of the model. The machine builds a predictive model based on training data and uses this model to predict the associated output.[6]

2.2 Classification

One category of supervised learning is classification. A supervised learning problem is a classification task when the purpose is to classify input data into the certain specified number of classes. In other words when we are predicting qualitative outputs.[7] Good example of a classification task is spam detection of emails, where the task is to classify if emails are spam or no spam. In this example feature vector $\mathbf{x} = (x_1, \ldots, x_p)^\top$ is formed from emails, spam and no spam are labels where emails are classified based on the features of each email.[8]

The classification process usually contains two phases: training phase and predicting phase. In the training phase, a set of features are generated from the data to be input vector $\mathbf{x}$. Features are attributes of a data set that are used to represent the data. Features may be categorical, ordinal, integer-valued or real-valued depending on the algorithm and objective of the classification. Once the feature vectors $\mathbf{x}_i$, $i = 1, \ldots, N$ have been generated from the available observations, they are paired with the corresponding true class label y_i associated with that feature vector. Learning algorithm will use the information of input-output pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, to learn a predictor function f from features to labels using a mapping of the form

$$f(features) \rightarrow labels.$$

In the prediction phase, data is represented in the input vector $\mathbf{x}$ by the same set of features that were generated in the training phase. The classification algorithm will use the map function learned in the training phase to predict the labels y from input vector $\mathbf{x}$.[9]

2.3 Cross-validation

According to Hastie et al. [7] "Probably the simplest and most widely used method for estimating prediction error is cross-validation." One type of cross-validation is K-fold cross-validation. K-fold cross-validation is useful when there is not enough training data for extracting large enough validation data set to estimate the performance of the prediction model. In K-fold cross-validation, part of the data is used to fit the prediction model and another part is used to test the model. Data is divided into K parts of the same size. One part is selected as a validation set and other K - 1 parts are used to fit the model. An example of the fold selection can be seen in figure 1. After fitting, the fitted model is used to predict the validation set and the prediction error of the model is calculated. This is repeated K times so that the validation set is always a different part of the data. Finally, the average is taken from the prediction errors estimated in each iteration to get the final prediction error estimation for the model.[7]

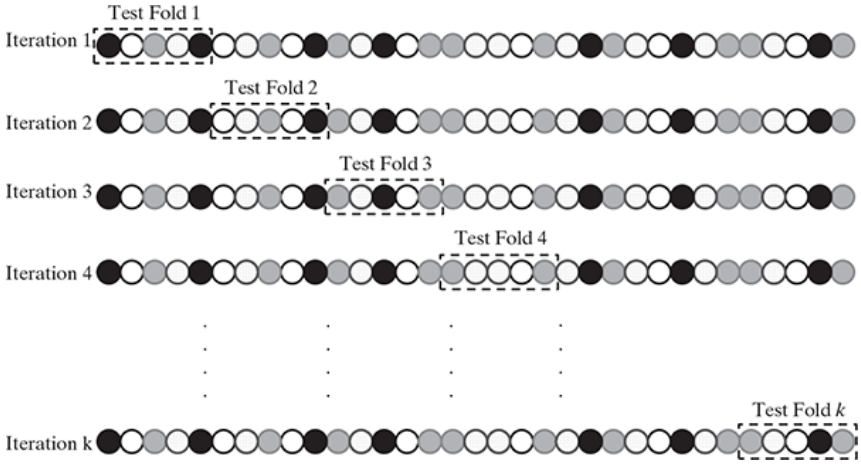

Figure 1: Example of the fold selection in K-fold cross-validation. Samples selected in the test fold are used as a validation set to estimate the prediction error of that iteration. The samples that are not in the test fold are used to fit the model. [6]

2.4 Overfitting and underfitting

A situation where the classification model is designed such that it emulates the training data too closely is called overfitting. Overfitting causes a situation that training data deviations, like noise, end up in the model. This will result in a good performance for training data, but poor performance for the test data and therefore causes poor classification quality. Overfitting is usually a result of trying to fit too

complex or overparametrized model to the training data. The resampling techniques like K-fold cross-validation can be used to avoid overfitting.[6]

Underfitting means a situation when the training of the model is kept too simple and it may not be able to capture key elements of the data and represent it well enough. This could happen for example when trying to represent a nonlinear input-output relationship using a linear model. Underfitting results in poor performance for both the training and the test data. It can be avoided by using more training data and by selecting better features for the data set. There is an example of both overfitting and underfitting is displayed in figure 2. [6]

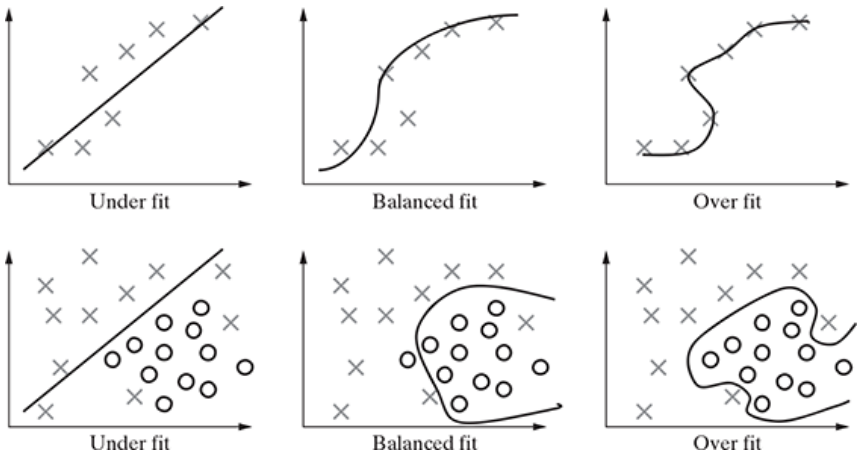

Figure 2: Example of overfitting and underfitting. [6]

2.5 Feature engineering

Feature engineering plays an important role in solving any machine learning problem. It is an important pre-processing step for machine learning and its purpose is to translate data set into features that are able to represent the data set more effectively and optimize the learning of the machine learning model. Feature engineering has three major components: feature transformation, feature construction and feature selection. In the feature transformation, structured or unstructured data is transformed into a set of features that are essential for the current problem that machine learning is trying to solve.[6]

The objective of the feature construction is to transform a given set of input features into a new set of more powerful features. In other words, in the feature construction, new features that discover missing information about relationships of current features can be added. A good example of feature construction is that there is a real estate data set having details of all apartments sold in a specific region. The data set contains three features: apartment length, apartment breadth, and price of the apartment. The machine learning problem is to predict the price of the apartments which have just come to sale and whose price is not known yet. Instead of using only the length and breadth of the apartment to predict the price, constructing

the area of the apartment from these and adding it to the feature data set would give a much better feature data set for predicting the price.[6]

Unlike in feature transformation and feature construction, feature selection does not create new features, but it is however the most critical phase of the pre-processing activity in the machine learning problem. The aim of the feature selection is to choose a subset of features from the full feature set that are most meaningful and that produce the best result for the current machine learning problem. Objectives of the feature selection are to make the machine learning model faster and more cost-effective, to improve the efficiency of the machine learning model and to give a better understanding of the underlying model that generated the data.[6]

2.6 Evaluation of the classification algorithm

Classification algorithm may yield four different cases, which are illustrated below in the binary classification problem:

- True Positive (TP). Eg., the classification model has predicted that the team wins and the team also wins. So, the prediction is correct.
- False Positive (FP). Eg., the classification model has predicted that the team wins, but the team loses. So, the prediction is incorrect.
- False Negative (FN). E.g., the classification model has predicted that the team looses, but the team wins. So, the prediction is incorrect.
- True Negative (TN). E.g., the classification model has predicted that the team loses, and the team loses. So, the prediction is correct.

Summary of TPs, FPs, FNs and TNs can be shown in a confusion matrix. Two class classification problem generates 2x2 confusion matrix but if classification problem has, for example, three classes, the confusion matrix would be 3x3, etc. An example of a confusion matrix is shown in figure 3.[6]

	ACTUAL WIN	ACTUAL LOSS
Predicted Win	85	4
Predicted Loss	2	9

Figure 3: Example of confusion matrix in two class classification problem. In this case, total count of TPs = 85, FPs = 4, FNs = 2 and FPs = 9. [6]

The counts of TPs, FPs, FNs and TNs for any classification problem can be calculated from the confusion matrix and they are used to evaluate the classification models. Next, I present the most common ways to evaluate the classification models by using values of TPs, FPs, FNs and TNs.[6]

2.6.1 Accuracy

Accuracy is the most basic and simplest way to estimate the performance of the classification algorithm. It is calculated by taking the proportion of the correct predictions to the total number of perditions.[8] Accuracy is measured as

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}. \tag{1}$$

2.6.2 Error rate

Error rate stands for the percentage of misclassifications.[6] Error rate is measured as

$$Errorrate = \frac{FP + FN}{TP + FP + FN + TN} = 1 - Accuracy. \tag{2}$$

2.6.3 Specificity

Specificity of the model is used to indicate the balance of the model. It measures the proportion of negative samples that are correctly classified. It equals the true negative rate (TNR).[6] Specificity is measured as

$$Specificity = \frac{TN}{TN + FP}. \tag{3}$$

2.6.4 Precision

The proportion of positive predictions that are truly positive is called precision. It indicates the reliability of a model in predicting a class of interest. So, a model with higher precision can be considered as a reliable model.[6] It is measured as

$$Precision = \frac{TP}{TP + FP}. \tag{4}$$

2.6.5 Recall

The proportion of correct predictions of positives to the total number of positives is called recall or sensitivity. It equals the true positive rate (TPR).[6] Recall is measured as

$$Recall = \frac{TP}{TP + FN}. \tag{5}$$

2.6.6 F1 score

F1 score combines precision and recall to a single number that gives a better evaluation of the model than precision and recall alone. It is measured as harmonic mean of the precision and recall. F1 score is always between 0 and 1. The higher F1 score of the model means better performance and the lower means worse performance.[8] The formula of the F1 score is

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{6}$$

2.6.7 Receiver Operating Characteristic (ROC) curve and Area under curve (AUC)

Receiver Operating Characteristic (ROC) curve visualizes the performance of the classification model. It plots false positive rate (FPR) in the horizontal axis against the recall (i.e., TPR) in the vertical axis at different classification thresholds.[6] FPR is measured as

$$FPR = \frac{FP}{FP + TN}. \tag{7}$$

The area under curve (AUC) value is the area under the ROC curve where each point on the curve gives a set of false and true positive values at a specific classification threshold. AUC indicates the predictive quality of a model. The value of the AUC is between 0 and 1. If AUC is under 0.5, the classifier has no predictive ability. An example of the ROC curve and AUC is given in Figure 4.[6]

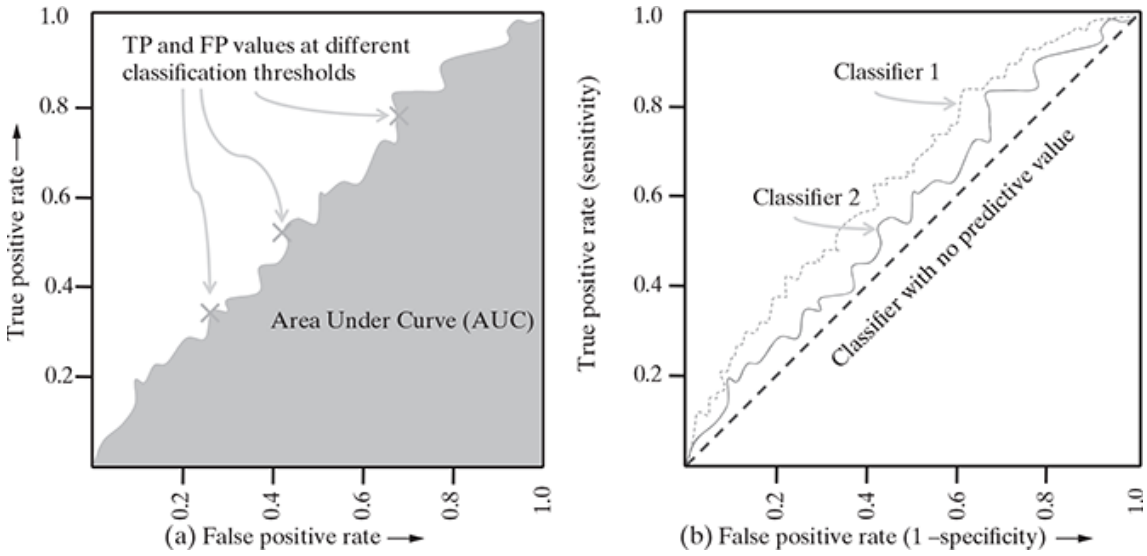

Figure 4: Example of the ROC curve and AUC.[6]

3 Classification methods

This chapter introduces two ensemble classification methods. Ensemble classification methods are methods that use and combine many classifiers to produce the final classifier. Classification methods that will be presented are random forest and boosting.

3.1 Random forest

3.1.1 Decision tree

Random forest is an ensemble classifying method that uses and combines many decision tree classifiers. In other words, decision trees form the base learner of the random forest algorithm. Decision tree learning is a widely used algorithm for classification. It can be used for multi-class classification tasks and its goal is to create a model based on the past data that can predict the value of the output using the input variables in the feature vector. A decision tree builds a model that is in the form of a tree structure. The last nodes of the tree are called leaf nodes. Each leaf node of the decision tree represents a possible output of the classification. An example of the decision tree is shown in figure 5. [6]

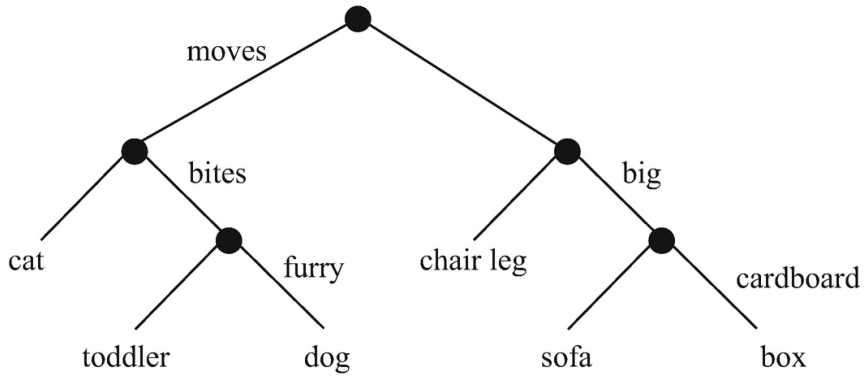

Figure 5: An example of the decision tree. In this example, the decision criteria is labelled to the other outgoing branches and the other one means negation. For example, if the features would be it moves, bites and it is not furry, the result of the classification would be a toddler.[10]

Decision trees are built with an approach called recursive partitioning from the training data. In this approach, the data is divided into multiple subsets based on the feature values. Dividing is started from the root node, which is the first node of the tree and represents the entire data set. Then feature that predicts the target class in the strongest way is selected. The data set is then divided into multiple partitions so that each partition has a distinct value of the feature that was selected. These are the first branches of the decision tree. The algorithm continues building the tree from each branch that was created by dividing the data set based on the feature that predicts the target class strongest way, as was done in the first iteration. This

will continue as long as the stopping criterion has been reached. Stopping criteria can be for example: most of the examples at a particular node have the same class, every feature has been used in the partitioning or the tree has grown so big that pre-defined threshold limit has been reached.[6]

The choice of which feature should be chosen to split the data in each node affects the most for the outcome of the decision tree. One way to do this decision is to choose a feature randomly and test if it is good enough to do the splitting. Testing can be done by counting an information gain of the split.[9] Information gain for the selected feature can be calculated by the difference between the entropy before the split and the entropy after the split. $Entropy(S)$ of the decision tree is defined as

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i, \tag{8}$$

where S is the sample set of training examples, c is the number of different class labels and p_i is the proportion of values falling into the i-th class label [6].

3.1.2 Random forest

Classification with a single decision tree does not always give the best possible outcome for the classification. One reason for this is that the decision tree can give the best possible result on the training data but still can perform poorly on the test data. In a random forest, instead of using only one decision tree, multiple decision trees are made, and results of the decision trees are then merged to get an accurate classification model.[10] Randomization and averaging in random forest reduce variance and hence improves accuracy compared to the single decision tree.

One way of building and evaluating random forest is called bagging. In bagging, training data is randomly selected from the original set of labelled data for each tree. The training set will contain duplicates and it is called a bag. There will be a set of labelled data that do not appear in the bag for each tree and it is called "out of bag" data. These data sets can be used to evaluate random forest algorithm by evaluating each tree with own "out of bag" data set and then taking the average of each evaluation.[10]

The random forest algorithm works as follows[6]:

1. Training data for the tree is selected using for example bagging. After the data set of N features for the tree has selected, m features of the tree are selected randomly from the data set such that $m < N$.

2. Selected splitting function is used to split m features to make new nodes.

3. Splitting nodes continues as long as end criteria has reached.

4. New data set will be selected to train new decision tree and steps 1-3 will be repeated. This is repeated as many times as is the desired number of decision trees in the random forest.

5. Finally classification with random forest can be done. Result of the classification will be determined such that result of each decision tree is vote and class with most votes is a result of the classification.

3.2 Boosting

Boosting methods are ensemble classifiers that use many weak learners to obtain good performance. Classifiers are called weak learners when learning algorithm can produce a classifier that performs relatively poorly with a probability of an error being strictly less than in random guessing. Classifiers that have arbitrarily small error probability are called strong classifiers.[11]

3.2.1 AdaBoost

The adaptive boosting (AdaBoost) is an effective and well-known algorithm to build ensembles of classifiers. The algorithm uses a set of weak learners to produce the final classifier. The idea of the algorithm is to train weak classifiers sequentially with the same set of training data but in each iteration modify individually weight of each sample such that those samples that were not correctly classified have larger weight in the next round. This will help weak classifiers to focus on samples that were not well classified by the weak classifier in the previous round. Training of a new weak classifiers and determination of the sample weights will be made as many times as is the defined number of weak learners in the AdaBoost algorithm. After the classifier is trained, the final classification is made by weighted majority vote where classifiers with smaller error have more weight.[11]

The AdaBoost algorithm works as follows for binary classification problem[11]:

1. Input dataset $Z = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i$ is a feature vector, $y_i \in \{-1, 1\}$ and N is the size of the input data set. M is the number of classifiers.

2. Weights of data set are initialized to be $w_i = 1/N$, where w_i is the weight of i:th data point.

3. Weak learner is trained with data set Z, and with using weights w_i to produce classifier $H : H(x_i) \rightarrow \{-1, 1\}$.

4. Weighted error of the mth classifier of the M classifiers is computed with $err_m = \sum_{i=1}^{N} w_i h(-y_i H(x_i))$, and it is used to compute the weight of the mth weak classifer $\alpha_m = \frac{1}{2}\log(\frac{1-err_m}{err_m})$. The function $h : \mathbb{R} \rightarrow \{0, 1\}$ used in formula of weighted error is the Heaviside function, that is defined as $h(x) = 1$, if $x >= 0$, and $h(x) = 0$, if $x < 0$. This means that $h(-y_i H(x_i))$ gives 0 if prediction is correct and 1 if prediction is incorrent.

5. Weight of each sample is updated. Updated weigth for each sample is $v_i = w_i \exp(-\alpha y_i H(x_i))$. After this weights are renormalized with computing $S = \sum_{j=1}^{N} v_j$ and then computing normalized weigths $w_i = \frac{v_i}{S}$ for the next round.

6. Steps 3-5 are repeated M times.

7. Final classifier $H(x) = sign(\sum_{m=1}^{M} \alpha_m H_m(x))$ is ready. Final classification in AdaBoost is made by weighted majority vote.

3.2.2 Multi-class classification with AdaBoost

The AdaBoost algorithm was originally designed for two-class classification problems and it has been proven to be very effective for producing accurate classifiers for binary classification problems. However, it can be also used for multi-class classification, but it does not produce very effective classifiers in these cases. This is because AdaBoost requires that the error of the weak classifiers err_m is less than $\frac{1}{2}$, so that the weight of the weak classifier α_m is positive. This is easily satisfied in two-class classification cases, because the error rate of random guessing is $\frac{1}{2}$, but it is much harder to achieve in the multi-class cases. This is the reason why AdaBoost can easily fail in multi-class classification cases.[12]

To use AdaBoost effectively for multi-class classification, several different modifications have been proposed. One of these is called AdaBoost-SAMME. It works otherwise the same way as AdaBoost, but the difference is in the step where the weight of weak classifier α is computed. It is computed with

$$\alpha = \frac{1}{2}\log(\frac{1-err}{err}) + \log(K-1), \tag{9}$$

where err is the weighted error of the classifier and K is the number of the classes in classification. The difference to the AdaBoost algorithm is the extra term $\log(K-1)$ which is critical for multi-class cases. With adding this term now α to be positive it is only needed to $(1-err) > 1/k$, or that the accuracy of each classifier is better than random guessing rather than $\frac{1}{2}$.[12]

4 Data analysis

In this chapter, two state-of-the-art classification methods are used to perform transportation mode detection based on smartphone sensor data. First I present the used tools and data sets. Then I explain how the hyperparameters are selected for the classifiers. Finally, the results are presented.

4.1 Tools

Data analysis in this thesis has been done with Python programming language [13]. Python was chosen because it is a popular programming language for machine learning applications and there are many useful libraries for this data analysis. Python libraries used in this data analysis are NumPy, Pandas, SciPy, Matplotlib and scikit-learn [14], [15], [16], [17], [18]. NumPy Python library is used for processing the data and for numeric calculations. Pandas-library is used for importing the data. Scipy-library is used for numeric calculations. Matplotlib is used for visualization tasks.

Scikit-learn library is used for different machine learning tasks. Scikit-learn library contains implementations of several machine learning algorithms including implementations of AdaBoost and random forest used in this data analysis. Scikit-learn library also contains tools for module validation which are also used in my analysis.

4.2 Explanation of the data set and pre-processing

The used data is obtained from a study "A Deep Learning Model for Transportation Mode Detection Based on Smartphone Sensing Data" made by Liang et al. [1]. The data set contains pre-processed smartphone accelerometer values measured in different transportation modes. Data has been collected with an android application, which has been installed on Google Nexus 5X and Google Nexus 6 smartphones. Application recorded the accelerometer data with a 50 Hz sampling frequency. Four different users carried smartphones in different transportation modes and labelled manually in which transportation mode accelerometer data belongs to. The phone was allowed to be held as desired by the user when collecting the data. The transportation modes in the study were driving a car, taking a bus, taking a subway, taking a train, bicycling, walking and being stationary.[1]

The smartphone's accelerometer data recorded by the application before pre-processing contains a three-dimensional vector where each vector has values in the unit of $\frac{m}{s^2}$ from one axis of the smartphone's coordinate system. In the smartphone's coordinate system, the x axis is from left to right, y axis is from down to up and z axis is from back to front looking from the front of the phone.[1]

The first pre-processing step for the accelerometer data was removing gravity. Removing gravity from data, leaves only characteristics of the different transportation modes to the data. Gravity was removed with a low-pass filter by the next equation,

$$G_k = \alpha G_{k-1} + (1 - \alpha) A_k, \tag{10}$$

$$L_k = A_k - G_k, \tag{11}$$

where G_k is the kth gravity vector, A_k is the kth accelerometer data vector, α is the exponential decay weight between the estimations of new gravity and the old one. L_k is the accelerometer data after gravity is removed.[1]

After removing gravity, data was smoothed. Smoothing the data is important because it will remove parts from the data where the smartphone's movement has not been consistent. For example, the user may have suddenly picked up the smartphone and because of this, the acceleration has changed suddenly. To remove these parts from the data, a central moving average was used. Data smoothing was performed as follows,

$$\hat{L_k} = \begin{cases} \frac{\sum_{i=1}^{2k-1} L_i}{2k-1} & k \in (1, \lfloor \frac{m}{2} \rfloor) \\ \frac{\sum_{i=k-\lfloor \frac{m}{2} \rfloor}^{k+\lfloor \frac{m}{2} \rfloor} L_i}{m} & k \in (\lfloor \frac{m}{2} \rfloor, K - \lfloor \frac{m}{2} \rfloor) \\ \frac{\sum_{i=2k-K}^{K} L_i}{2(K-k)+1} & k \in (K - \lfloor \frac{m}{2} \rfloor, K), \end{cases} \tag{12}$$

where L_k is the kth original acceleration value, $\hat{L_k}$ is the kth acceleration value after smoothing, K is the number of vectors, and m is the predefined constant, which was set to 5.[1]

After removing gravity and smoothing the data, the final acceleration values which are used in the classification were formed by taking the magnitude of the acceleration values. Magnitude of the kth smoothed acceleration value $\hat{L_k} = (\hat{L_{k_x}}, \hat{L_{k_y}}, \hat{L_{k_z}})$ was calculated by,

$$|\hat{L_k}| = \sqrt{\sum_{i=x,y,z} \hat{L_{k_i}}^2}. \tag{13}$$

The magnitude of the acceleration values are used because the phone's position and orientation can vary between the users.[1]

The data set used in this thesis contains the magnitude of the accelerometer values after pre-processing steps presented above. The data of each transportation mode are in their own csv file. They are imported with pandas library method `pandas.read.csv`. After data sets are imported, the data set of each transportation mode is sliced to windows of 10.24 s (512 samples) with a slide size of 1.28 s (64 samples), so that the data set reflects to real time data as in the study [1]. The slide size means that there are 64 new samples in each window, so that the transportation mode detection happens every 1.28 seconds, with 64 new samples and 448 old samples. Sliced windows for each transportation mode are taken 2000 for this data analysis, and features are converted by these windows of the magnitude of the accelerometer values.

4.3 Features

Features are converted from 10.28 second windows of accelerometer magnitude values. Same features as in [1] are used in this data analysis. Those features are selected

because they produced good accuracies in [1] with the same data set and windows size. These features are listed in table 1.

Table 1: Table of features used in this data analysis.

Features	
Time domain	mean standard devitation median root mean square min max range kurtosis skewness
Frequency domain	power spectral centroid log-scale power spectral density at 1-10Hz

Features can be divided into time domain features and frequency domain features. Features that are converted from the time domain are intended to include characteristics of acceleration behaviour of each transportation mode which can be seen from figure 6, which displays plots of 512 point windows of the acceleration magnitudes of each transportation mode. As can be noted the transportation modes have different patterns. Walk and bicycle have biggest values and seem to have some periodicity. Being stationary mode has the lowest acceleration magnitude values which was expected. Acceleration magnitude values of car, bus, subway and train are closer to each other, than walk, bicycle and stationary, but still, show some differences in their acceleration behaviour. Selected time domain features listed in table 1 should catch differences between the acceleration behaviours of the transportation modes.

Frequency domain features are intended to include characteristics of the repetitious nature of the accelerometer magnitudes. Log-scale power spectral density at 1-10Hz are selected for features because in previous work [1], it was argued that the most power is in frequencies less than 10Hz. Also, the power spectral centroid is selected to the feature set. Before data windows are transformed to the frequency domain, Hamming window function is used to scale the data window to reduce the power leakage effect as in the study [1].

After the features are converted from the 512 sample windows, label vector y is created to store the information whether the feature vector $\mathbf{x}$ belongs to class walk, car, train, bicycle, subway, bus or stationary. Then scikit-learn `train_test_split` method is used to divide the data set randomly into training set and test set. The test set contains 30% and the training set 70% of the data. The training set is used to train the models and in selecting the optimal hyperparameters with 10-fold cross-validation for the models. The test set is used to evaluate how the model with optimal parameters performs with unseen data.

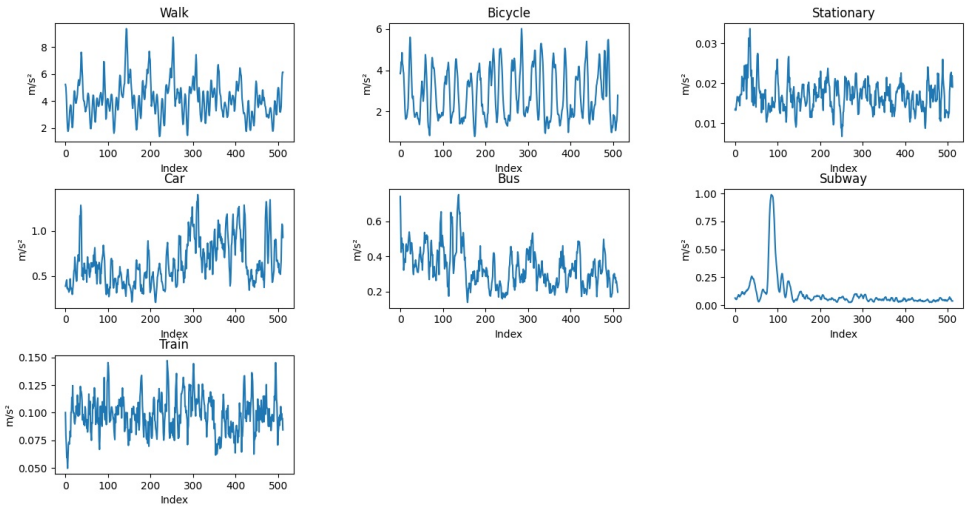

Figure 6: An example of 512 point window of acceleration magnitudes of different transportation modes.

4.4 Hyperparameter optimization

4.4.1 Random forest

Class RandomForestClassifier from scikit-learn library is used to perform random forest classification in this data analysis [19]. The classifier can be modified with different hyperparameters such as the number of trees or the maximum depth of the tree, that are given before training the model. To get the model that performs best for this machine learning problem it is important to find optimal hyperparameters for the model.

Grid search was used to find the best hyperparameters for the random forest classifier. In grid search, every combination of parameters is tested and the hyperparameter combination that gives the best average accuracy after 10-fold cross-validation is selected for the final model. Grid search was performed to find best parameters for *max_depth*, *criterion* and *n_estimators*. *Max_depth* defines the maximum depth of the decision trees. Values that were tested as *max_depth* were 5, 50 and 100. *Criterion* defines the function that is used to measure the split quality. Criteria that were tested were entropy and gini impurity. *N_estimators* defines the number of trees in a random forest. The number of estimators tested were 64, 128, 256 and 512.

In study [20], it is showed with 29 data sets that using more than 128 trees in the random forest does not give significantly better performance but increases computational costs. This why they suggest that number of trees should be between 64-128, to get a good balance between performance, memory usage and processing time. In this thesis, random forest with 256 and 512 trees are tested to if they give significantly better performance or should the number of trees be as suggested in the study [20].

Random forest classifiers that produced the best result were using entropy as a

measure of the split quality and 50 as a maximum depth of the decision trees. Results with these parameters with the different numbers of trees can be seen in table 2. As can be noted the accuracy does not significantly increase when using more than 128 trees as mentioned in study [20]. This why it is chosen as the number of estimators to the final model. The random forest classifier chosen for the final evaluation has hyperparameters 50 as a maximum depth of the decision trees, entropy as a measure of the split quality and 128 as a number of the decision trees.

Table 2: Scores of the random forest classifier with using hyperparameters $max\_depth = 50$ and $criterion = entropy$.

Number of trees	64	128	256	512
Score (10-CV)	90.684%	90.980%	90.990%	91.020%

4.4.2 AdaBoost

Class AdaBoostClassifier from scikit-learn library is used to perform AdaBoost classification in this thesis [21]. In AdaBoostClassifier, one can define if it will use AdaBoost.SAMME or AdaBoost.SAMME.R algorithm for classification. AdaBoost.SAMME is used in this thesis. The weak estimator used in AdaBoost is a decision tree.

A grid search was performed for the AdaBoost classifier to find the optimal hyperparameters for the maximum depth of the decision trees and for the learning rate, which defines the contribution of the weak classifiers. Values tested as the maximum depth of the decision trees were 1, 2, 5, 10 and 15. Values tested as learning rate were 0.01, 0.1 and 1. In grid search, the number of estimators of the AdaBoost classifier is defined to be 50. Hyperparameter combination that produced the best average accuracy after 10-fold cross-validation was using 15 as a maximum depth of the decision trees and 0.1 as a learning rate with having average accuracy of 90.480%.

After the hyperparameters for the learning rate and for the maximum depth of the decision trees are defined, the different number of estimators are tested to find the best one for this model. The average accuracies after 10-fold cross-validation with using the different numbers of estimators are reported in table 3.

Table 3: Scores of the AdaBoost classifier with using hyperparameters $learning\_rate = 0.1$ and $max\_depth = 15$.

Number of estimators	25	50	100	150	200	300
Score (10-CV)	89.418%	90.480%	90.755%	91.367%	91.449%	91.643%

As can be noted, using more than 150 estimators do not significantly improve performance. So, to get the model with a good balance between performance and

processing time 150 is selected as the number of estimators. The AdaBoost model selected for the final estimation has hyperparameters 0.1 as the learning rate, 15 as the depth of the decision trees and 150 as the number of estimators.

4.5 Results

Final estimation is made for random forest classifier and AdaBoost classifier with optimal hyperparameters. In the final estimation, models are trained with training data that contains 1400 windows of each transportation mode and evaluated with unseen test data that contains 600 windows of each transportation mode. Models are evaluated with methods presented in chapter 2.

With test set, random forest classifier produces classification with accuracy of 91.600%. Confusion matrix of the classification is shown in figure 7, ROC curve and AUC value of each class can be seen in figure 8, and precision, recall and F1-score of each class are reported in table 4.

Table 4: Precision, recall and F1-score of each class using random forest classification with test set.

Class	Precision	Recall	F1-score
bicycle	0.964	0.995	0.979
bus	0.862	0.773	0.815
car	0.833	0.873	0.853
stationary	0.990	0.997	0.998
subway	0.872	0.907	0.889
train	0.888	0.870	0.879
walk	1.00	0.997	0.998

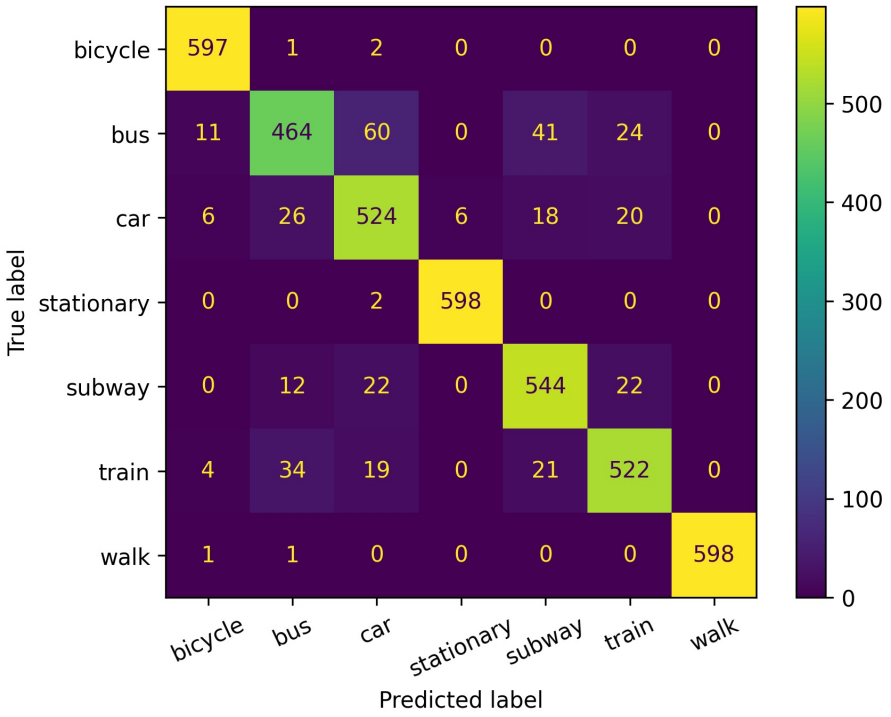

Figure 7: Confusion matrix of the random forest classification with test set.

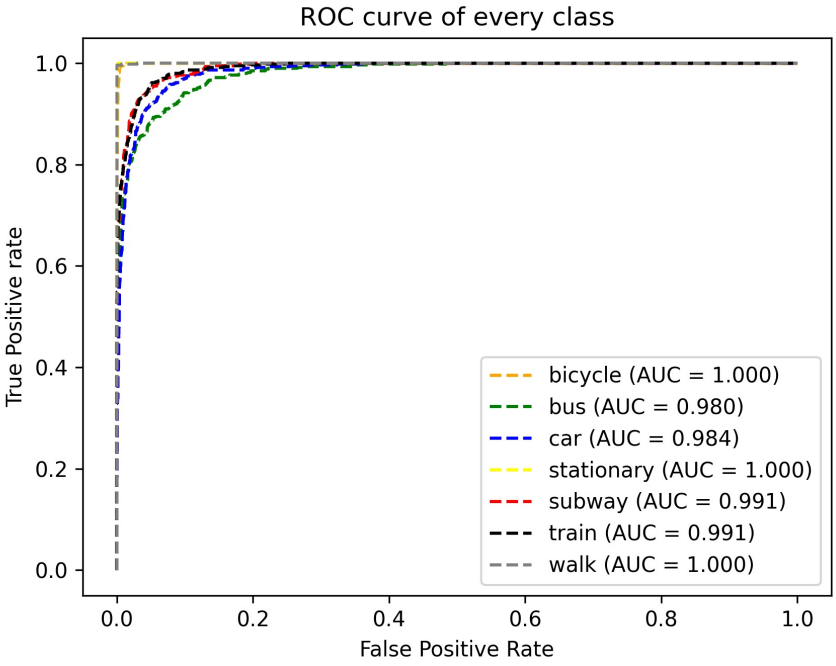

Figure 8: ROC curves and AUC values of each class using the random forest classification with test set.

From the results, we can note that random forest classifiers can be used to classify transportation modes based on smartphone accelerometer data with good accuracy. Classes for which the least misclassification results occurred were a bicycle, stationary and walk with having F1-scores of 0.979, 0.998 and 0.998. All of those three classes have also an AUC value of 1.000. Most misclassifications happened when classifying motorized transportation modes. The class with the most misclassifications was a bus with having F1-score of 0.815. Mostly it was misclassified with other motorized transportation modes and among those most often to the car. From the confusion matrix, one can note that most misclassifications are between motorized transportation modes.

With test set, AdaBoost classifier produces classification with accuracy of 92.357%. Confusion matrix of the classification is shown in figure 9, ROC curve and AUC value of each class can be seen in figure 10, and precision, recall and F1-score of each class are reported in table 5.

Table 5: Precision, recall and F1-score of each class using AdaBoost classification with test set.

Class	Precision	Recall	F1-score
bicycle	0.963	0.988	0.975
bus	0.896	0.807	0.849
car	0.848	0.902	0.874
stationary	0.993	0.992	0.992
subway	0.874	0.900	0.892
train	0.900	0.883	0.892
walk	0.993	0.993	0.993

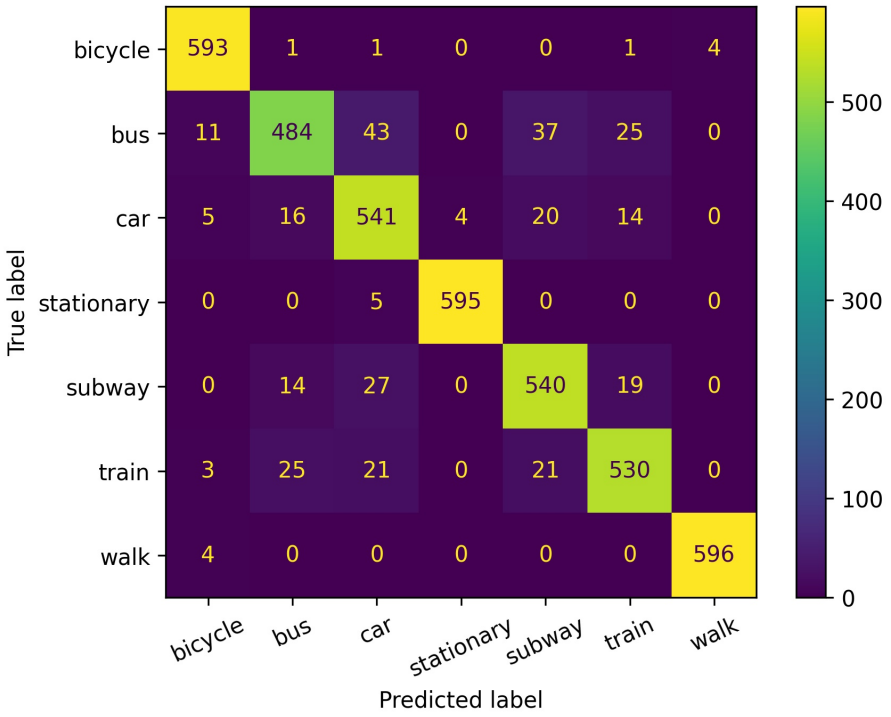

Figure 9: Confusion matrix of the AdaBoost classification with test set.

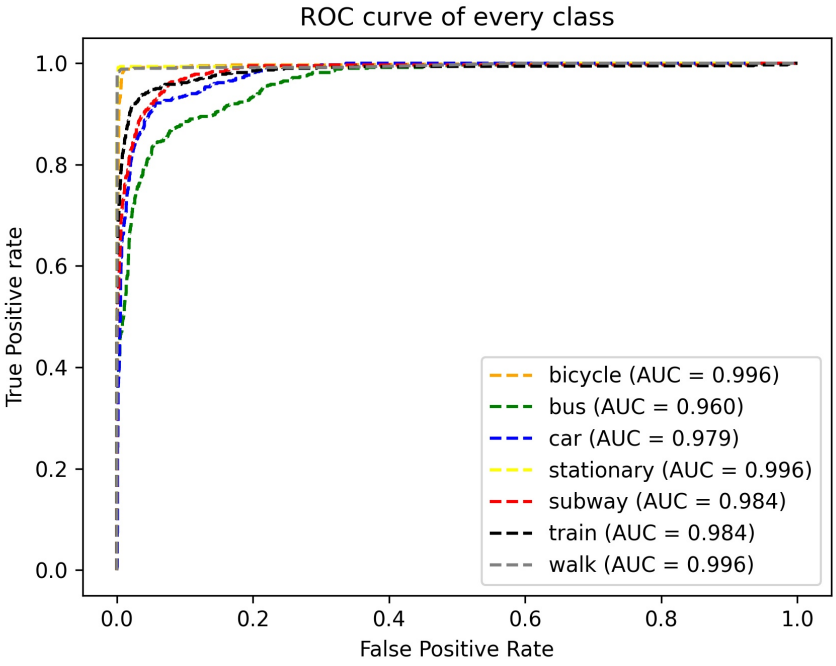

Figure 10: ROC curves and AUC values of each class using the AdaBoost classification with test set.

Based on the results one can state that the AdaBoost classifier can be used to classify transportation modes based on smartphone accelerometer data. AdaBoost produces slightly better accuracy than random forest, but the results are quite similar as can be seen from the confusion matrix. The classes that were classified with the best accuracy are bicycle, stationary and walk, with each having AUC value 0.996 and F1-scores of 0.975, 0.992 and 0.993. The class that had the most misclassifications was bus with an F1-score of 0.849. From the confusion matrix, one can see that it was mostly misclassified with other motorized transportation modes. As a whole, the most misclassification happens between motorized transportation modes.

To conclude the results, illustrate that the random forest and the AdaBoost classifier are both suitable for classifying transportation modes with smartphone sensor data. AdaBoost performs with slightly better accuracy (92.357%) than the random forest classifier (91.600%), but the difference is not very significant. With both classifiers, non-motorized transportation modes were classified with better accuracy than motorized ones. The misclassifications of motorized transportation modes mostly happen between other motorized transportation modes. This was quite expected outcome because motorized transportation modes have more similarities in their acceleration behavior than biking, walking or being stationary. However, most of the samples of each transportation were correctly classified with both classifiers. To achieve better accuracy, the point of development is to find more features that internalize better the differences between motorized transportation modes.

Comparing the results with the results of previous work [1], one can see that they achieved better accuracy in transportation mode detection. They achieved an accuracy of 94.48% by using Convolutional Neural Network (CNN) with the same data set and window size. However, we achieved to get better accuracy with random forest and AdaBoost than they achieved when they compared CNN with traditional machine learning methods. My random forest model outperformed theirs by 91.60% to 90.11% and my AdaBoost model outperforms theirs by 92.36% to 80.73%.

5 Summary

This thesis introduced how two state-of-the-art classification methods perform in classifying smartphone sensor data. First, the basics and concepts of machine learning, classification, and evaluation of classification algorithms were introduced. Then two classification methods random forest and boosting were presented. These two chapters provided the basis for the data analysis that was introduced in the last chapter. The aim of the data analysis was to investigate how well random forest and adaptive boosting (AdaBoost) performs in transportation mode detection based on smartphone accelerometer data.

Machine learning is a study of algorithms that are able to improve through experience. Machine learning can be divided into three different areas: reinforcement learning, unsupervised learning, and supervised learning, of which this thesis focused on supervised learning. In supervised learning, the machine builds a predictive model based on training data, where inputs are paired with desired outputs. This model is then used to predict the outputs for new given inputs.

Classification is a subfield of the supervised learning. A supervised learning problem is a classification task when the model is predicting qualitative outputs. There are different classification algorithms that can be evaluated with using different metrics. The simplest one of these is accuracy, which tells the proportion of the correct predictions of all predictions. Classification methods that were presented in this thesis were random forest and boosting.

In a random forest, multiple different decision trees are used to make the classification. Decision trees are trained with a randomly selected training set and a random number of features. The result of the classification is defined so, that result of each decision tree is vote, and the class with the most votes is the result of the classification.

Boosting methods are ensemble classifiers that use multiple weak learners to make the classification. The boosting method that was introduced in this thesis was AdaBoost. In the AdaBoost algorithm, weak learners are trained sequentially with the same set of training data, with changing the individual weights of samples, so that samples that were incorrectly classified in the current round have a bigger weight in the next round. This will make the weak learners focus more on samples that were classified poorly in previous rounds. The final classification in AdaBoost is made with a weighted majority vote where weak classifiers with smaller error have more weight.

The data analysis investigated how well random forest and AdaBoost perform on classifying transportation modes based on accelerometer data. Features were formed from 10.24 second (512 samples) windows with a slide size of 1.28 seconds (64 samples) of the pre-processed magnitude of the accelerometer values. Pre-processing steps for accelerometer values included gravity removing and data smoothing. The number of used data windows for each transportation mode was 2000 of which 1400 windows were used to train the model and in the selection of the optimal hyperparameters with 10-fold cross-validation and grid search, and 600 windows were used to test how models perform with unseen data.

The hyperparameters selected for the random forest were 50 as a maximum depth of the decision trees, entropy as a measure of the split quality, and 128 as a number of the decision trees, and for the AdaBoost, 0.1 as the learning rate, 15 as a depth of the decision trees and 150 as a number of estimators. Both models performed well in the classification with the test set. The AdaBoost produced slightly better accuracy with an accuracy of 92.357% than the random forest that produced an accuracy of 91.600%. As a whole, the result was quite the same. Both classified non-motorized transportation modes with better accuracy than motorized and therefore the point of development was thought to be the discovery of new features that would improve the classification accuracy of the motorized transportation modes.

References

[1] Liang, X., Zhang, Y., Wang, G. and Xu, S. *"A Deep Learning Model for Transportation Mode Detection Based on Smartphone Sensing Data"*. IEEE Transactions on Intelligent Transportation Systems, 2020. vol. 21. no. 12, pp. 5223-5235. ISSN: 1558-0016.

[2] Stenneth, L., Wolfson, O., Yu, P. and Xu, B. *"Transportation mode detection using mobile phones and GIS information"*. In Argrawal, D. Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '11). Chicago, Illinois, November 1-4, 2011. New York, NY, USA. Association for Computing Machinery. 2011. 54–63. ISBN: 9781450310314

[3] Widhalm, P., Nitsche, P. and Brrödie, N. *"Transport mode detection with realistic Smartphone sensor data"*. Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Tsukuba, Japan, November 11-15, 2012. 2012. pp. 573-576. ISBN: 978-4-9906441-0-9

[4] Hemminki, S., Nurmi, P. and Tarkoma, S. *"Accelerometer-based transportation mode detection on smartphones"*. In Petrioli, C. Cox, L. Whitehouse, K. Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys '13). Roma, Italy, November, 2013. Association for Computing Machinery, New York, NY, USA. 2013. Article 13, 1–14. ISBN: 978-1-4503-2027-6.

[5] Mohammed, M., Khan, M. and Bashier, E. *"Machine Learning: Algorithms and Applications"* 1st edition. CRC Press, 2016. ISBN: 9781315354415

[6] Das, A., Chandramouli, S. and Dutt, S. *"Machine Learning"* Pearson Education India. 2018. ISBN : 93-89588-13-8

[7] Hastie, T., Tibshirani, R. and Friedman, J. *"The Elements of Statistical Learning"* 2nd ed. New York: Springer. 2009. ISBN 978-0-387-84858-7

[8] Bateman, B., Jha, A., Mathur, I. and Johnston, B. *"The supervised learning workshop"* 2nd ed. Birmingham. Packt Publishing. 2020. ISBN: 9781800209046

[9] Aggarwal, C. *"Data Classification"* Chapman and Hall/CRC. 2015. ISBN : 1-4987-6058-9

[10] Forsyth, D. *"Applied machine learning"* Springer, Cham. 2019. ISBN 978-3-030-18113-0

[11] Zhang, C. and Ma, Y. *"Ensemble machine learning: Methods and applications"* Springer, Boston, MA. 2012. ISBN 978-1-4419-9326-7

[12] Hastie, T., Rosset, S. , Zhu, J. and Zou, H. *"Multi-class adaboost"* Statistics and its Interface, 2009, 2.3: 349-360.

[13] *"Python Software Foundation"* Python Language Reference, version 3.6.4. Available at: http://www.python.org

[14] Harris, C.R. , Millman, K.J. , van der Walt, S.J. , Gommers, R. , Virtanen, P. , Cournapeau, D. , Wieser, E. , Taylor, J. , Berg, S. , Smith , N.J, Kern, R. , Picus, M. , Hoyer, S. , van Kerkwijk, M.H. , Brett, M. , Haldane, A. , Fernández del Río, J. , Wiebe, M. , Peterson, P. , Gérard-Marchant, P. , Sheppard, K. , Reddy, T. Weckesser, W. Abbasi, H. Gohlke C. and Oliphant, T.E. *" Array programming with NumPy"* Nature 585, 357–362. 2020. DOI: 0.1038/s41586-020-2649-2.

[15] McKinney, W. *"Data Structures for Statistical Computing in Python"* Proceedings of the 9th Python in Science Conference, 51-56, 2010.

[16] Virtanen, P.,Gommers, R, Oliphant, T.E, Haberland, M, Reddy, T, Cournapeau, D,Burovski, E, Peterson, P.,Weckesser, W Bright, J van der Walt, S.J, Brett, M, Wilson, J K., Millman, J, Mayorov, N, Nelson, A.R.J, Jones, E, Kern, R, Larson, E, Carey, C, Polat, I, Feng, Y, Moore, E.W, VanderPlas, J., Laxalde, D., Perktold, J, Cimrman, R, Henriksen, I, Quintero, E.A, Harris, C.R, Archibald, A.M, Ribeiro, A.H, Pedregosa, F, van Mulbregt, P, and SciPy 1.0 Contributors. *"SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python."* Nature Methods, 17(3), 261-272. 2020.

[17] Hunter, J. D. *"Matplotlib: A 2D Graphics Environment"* Computing in Science & Engineering, 9, 90-95. 2007. DOI:10.1109/MCSE.2007.55

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

[19] *"sklearn.ensemble.randomforestclassifier"* [viewed: 8 Apr 2021] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[20] Oshiro, T.M., Perez, P.S. and Baranauskas, J.A. *"How Many Trees in a Random Forest?"* In: Perner P. (eds) Machine Learning and Data Mining in Pattern Recognition. MLDM 2012. Lecture Notes in Computer Science, vol 7376. Springer, Berlin, Heidelberg. 2012. ISBN: 978-3-642-31537-4

[21] *"sklearn.ensemble.AdaBoostClassifier"* [viewed: 15 Apr 2021] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html