# Road safety with Android Auto and Machine Learning

Anton Tobias Jensen ANTJ@ITU.DK

Advisors: Fabricio Narcizo and Jørgen Staunstrup
Submitted: May 2020 Course: BIBAPRO1PE

IT University
of Copenhagen

## Abstract

This thesis aims to research the question of how to predict road safety and how a driver can safely receive relevant information on road safety during a drive.

This has become a relevant field of research, with sophisticated computing hardware available as a feature in cars. Additionally, operation areas and computation capability of mobile devices are expanding.

The results of the experiment in this thesis has been an Android application which implements Machine Learning Models and Statistical Models to predict accidents, based on the current situation of the user. The Machine Learning Models do not provide valid scientific evidence for the predictions to be correct, due to the supervised historical traffic data, used to train the Machine Learning models, having inconsistent patterns of how accidents happen. The Machine Learning models are activated by Statistical Models using historical traffic data. The models are only compatible to some extent. This is limited by a historical weather data set, which only enables the model to predict accidents within a range incorrect with a level of abstraction.

Thus the Statistical Models and the Machine Learning Models are implemented in the application using the Android System compatible with the Android Auto subsystem. Android Auto enables a safe communication channel with the drive.

The application is distributable to Android Users and compatible with 60.3 % of all android devices. In the future the models predictions might be invalid, as the behaviour of a car might change. Although the experiment does not provide any sophisticated pipeline for extending the models with new data.

# Contents

Chapter 1

# Introduction

This is a bachelor thesis about experimenting with Machine Learning and mobile application to prevent traffic accidents.

Machine Learning can detect what kind of accident can be present and applied with statistical models using historical data of accidents, and the world models can detect when a situation is differentiating from other situations. Divide the models of these experiments into three parts, where one part will be using supervised data, another part will be using statistical data, and a third-part will use the results of the other models to evaluate risks.

This experiment utilizes data from the real world into the models, as it will enable the models to simulate the behavior in the real world.

The collected data is computed to simulate the real world with various methods in order to collect useful information.

The experiment tries to find patterns in the data where the Android Operating System distributes notifications to a user and gathers information about a car driving.

The Android Operating System contains an Android Auto subsystem, which can not enable features of the car, but enables a communication channel to the driver behind the wheel of the car. This thesis will analyzes how this compiles with safely delivering messages to the driver.

Furthermore, the analysis of this bachelor thesis will evaluate how the models perform and how the product distributes in the Android System.

Chapter 2

# Methodology

The study of this thesis has been to make a system that can predict road safety and study how a driver safely can receive relevant information about the road safety during a drive, using machine learning and historic traffic data.

## 2.1 Collecting data

Before doing Machine Learning and statistical models, it is fundamentally required to have some data to train and test the model or create models with.

The data used to create the various models were collected from various sources. This is due to complicated restrictions obtaining data on individual road accidents, since this data is considered to be a personal sensitive information.

Statistics Denmark is a danish authority who provide StatBank as a statistical service, which contains subjects of living conditions and traffic accidents [1]

The StatBank service does not fulfill the requirements of the data needed to train the machine learning model this project targets. To obtain the correct format of the data it is required to process the statistics into data. This was done through a small Java Console Application, made as a part of this thesis, which converts statistical sets into data sets.

The statistics used from StatBank has been:

[1]StatBank, Statistics denmark, `https://www.statbank.dk/statbank5a/default.asp?w=1280(Visited10/04/20)`

- UHELDK1: Injured and killed in road traffic accidents by region, casualty, motor vehicles involved, age and sex
- UEHLD3. Road traffic accidents by type of accident, accident situation, urban area and speed limit
- UHELD4: Road traffic accidents by type of accident, type of transport unit involved, hour, day of the week and month
- UHELDK7: Road traffic accidents by type of accident, municipality, urban area and accident situation
- BY2: Population 1. January by municipality, size of the city, age and sex

The UHELD 4 contains statistics on time and month of accidents. This means the statistics is to map some specific time frames in a month by the individual years. To extend the values of the data set, the accident types are mapped with historical weather data.

The historical weather data is retrieved from DMI, which is a Danish authority responsible for measurements of meteorological data in Denmark. As of March 2020 DMI does not offer any data as a service, although their website offers single sets of historical data, divided by date.

As a part of this bachelor thesis a small website is made[15], to load the data from DMI. The website makes a series of API calls to the DMI API/server, approximately 3000 calls, and returns the data as one series of JSON objects. This data set is then structured inside a CSV file with the data set containing the following information:

- Average temperature in Denmark within a given hour span of a day, from 2012-2020
- Average sunlight minutes in Denmark within a given hour span of a day, from 2012-2020

In the process of merging the data from the weather data set to the accidents, the small Java console application[16] enables the data to be processed as entities of hour spans on a given weekday separated by month and year.

This is due to the UHELD 4 data set only containing the hour, day of week, month and year as time/date reference. The mapping of the data is done by calculating the value of the average temperature for all the individual weekdays of a specific month and year. Fx. if the UHELD 4 data set contains an accident on a Wednesday between 2-3 am in November 2016, the data will be mapped to the average temperature between 2-3 am of all Wednesdays occurring in November 2016.

Regarding the sunlight, this data is stored within the temperature entity described above. The sunlight value is stored as a Boolean value where it declares an hour to be night if there is less than 5 minutes of sunlight given the average hour.

In the data set the value of the sunlight per hour has the following pattern: hours registered with 5 minutes or less minutes of sunlight is always followed by an hour with a sunlight per hour value close to zero. This can detect the hour for the sunset. And vice versa for the sunrise.

This data should also depend on the geographical location of the accident, but since the UHELD 4 data set does not contain the geographical information, it is not possible to determine this. Only the UHELD 1 and UHELD 7 contains geographical information at the form of a municipality.

## 2.2 Data sets

After processing the collected data, the data for the ML model consists of four sets.

A small data analysis describes that the data has ten different types of accidents and data on the single accident in regards to the:

- Speed Limits and urban/non-urban environment
- Municipality and urban/non-urban environment

Whereas the two other data sets, do not refer to a specific type of accident but instead accumulate to a value of how represented an accident is.

- Municipality of the accident, age and gender
- Approximate temperature and daylight

### 2.2.1 Speed and urban

The data set from UHELD3 consists of records from 2001-2018 and has approximately 120.000 rows, with all accidents involving the authorities recorded, independent from casualties or injuries. The urban column consists of a Boolean and the speed column is classified into nine different speed limits, from less than 20 km/h, 20-50 km/h and afterward incriminating by 10 km/h up til 130 km/h, except for 120 km/h since there is no representation of 120 km/h in the data set. [17]

### 2.2.2 Municipality and urban

The data set from UHELD 7 consists of records from 1998-2018 and has approximately 186.000 rows, with all accidents involving the authorities are recorded, independent from casualties or injuries. The urban value consists of a Boolean and the municipality is categorized into 99 different municipalities. Municipalities that have been restructured over the years from 1998 until 20018 have in the data set been merged, such that they fit the representations of danish municipalities as of 2020. At a level of abstraction, Denmark is divided by 98 different municipalities and one area without a municipality (Christianso on Bornholm).[18]

### 2.2.3 Municipality, age and gender

The data set from UHELD1 consists of records from 2001-2018 but has only approximately 46.000 records. This is since the data set only consists of accidents involving injuries. The Municipality column consists of 99 different municipalities, describing where the accident happened [22]. The age column describes the age of the injured driver, classified into five different columns of 0-17,18-24, 25-44, 44-64 and 65 or higher. Lastly, the gender column is a Boolean value.[19]

### 2.2.4 Approximate temperature and daylight

The data set from UHELD4 consists of records from 2001-2018 and approximately 46.000 records, whereas the data set from DMI consists of records from 2012-2020 and consists of approximately 70.200 rows. The mapped data sets consist of 20.400 records. The columns of this data set are temperature classified into numbers rounded to closest integers. The daylight column is classified as if the sun is up or not.[20]

## 2.3 Machine learning

One fundamental part of this experiment is to predict accidents based on the data sets, using the Python TensorFlow framework consisting of various machine learning models and tools. For this study, the models and tools used have come from the Keras Open Source project. The machine learning models are both supervised since the supervised data is available through the types of accidents the model can predict.

### 2.3.1 Speed and Surroundings Characteristic supervised learning model

With a level of abstraction, it is evident that a lot of the accidents in the urban areas will happen in low-speed limit areas, which the data set also states if we look at the statistics of the data, as shown in Figure 2.1.

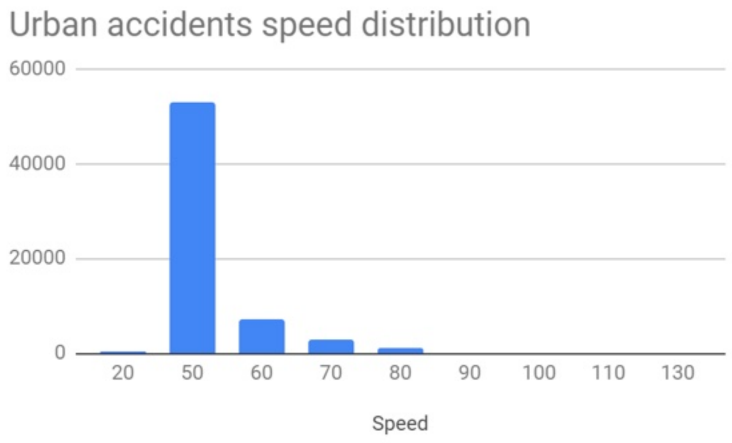


Figure 2.1: Distribution of the speed limits based on number of accidents in urban areas. Data from UHELD3

Whereas with some abstraction of the non-urban data, it will also be evident that most of the accidents will happen within ranges of higher speed limits, as shown on figure 2.2

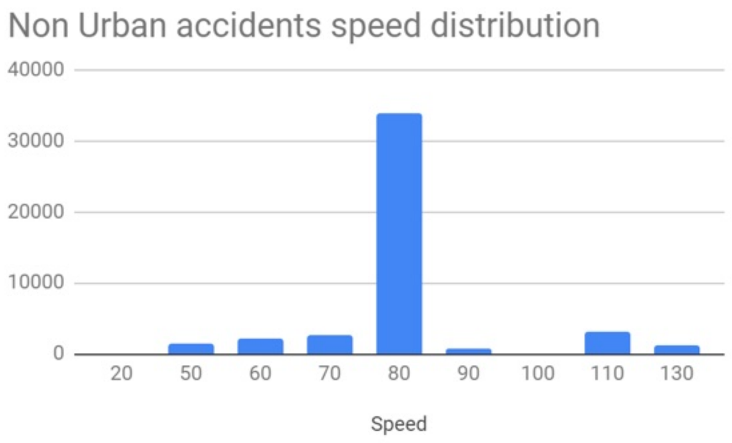


Figure 2.2: Distribution of the speed limits based on number of accidents in non-urban areas. Data from UHELD3

This states an obvious problem with this data set, since it contains how many accidents are registered within a speed limit range but do not contain how often this speed limit occurs compared to how often other speed limits occur. Therefore it is necessary to look at the pattern of how the risk of a specific accident evolves, which means that the Application will implement predictions for the individual accident type, therefore the model needs to output probabilities for all accident types.

This model aimsis to predict the odds all type of accident, using the Keras neural network framework. When building a neural network, a set of layers are needed. For this particular model, there are 180 various combinations of the data since there are nine different types of speed limits, two different types of environments (urban/non-urban) and 10 different types of accidents. Therefore the first layer has 180 nodes, and then these neurons should be narrowed down to 10 different outputs.

So these layers are intended to estimate the relationship between input variables in order to predict an outcome variable. The nodes inside the layers have the functionality to activate. In other words, they can be something in between on and off. The activation of these nodes are for this model defined by an activation function, where a sigmoid function is used to provide a non linear transformation of the data.

$$Sigmoid(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

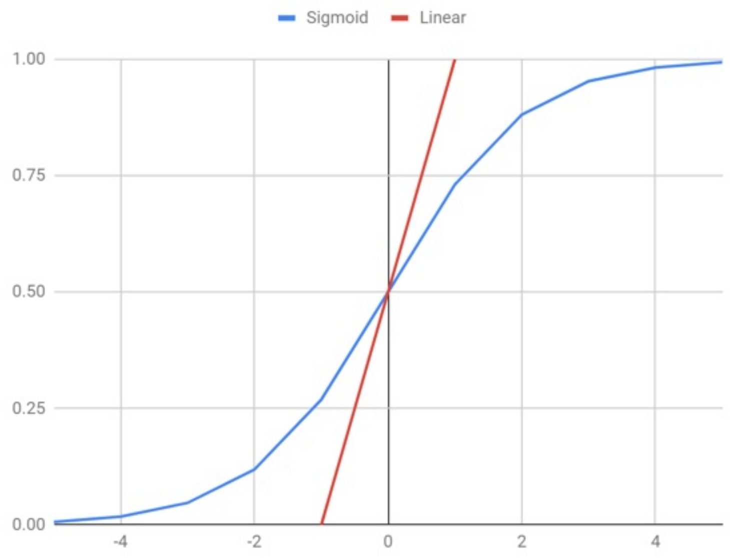


Figure 2.3: Sigmoid function and linear function

This model intends that speed limits do not have relationships to each other since the abstraction of speed limits are supposed to say something about the current road conditions and not something about the actual speed the driver is going. Therefore the speed limits are one hot encoded so that the various speed limits represented in the data set are categorized in binary data.

Since the categories of accidents do not have relationships with each other, it is important to make sure the machine learning model also understands this. Doing this by one hot encoding the accident categories before training the model, adding a categorical cross-entropy loss function, which is a cross-entropy categorized by the binary values representing the various categories of outputs.

To evaluate the model, the data is split such that 10% of the data set is used to test the model and 90% of the data set is used to train the model.

The model is trained with 200 epochs/iterations of the data set, where the accuracy of each iteration has been steadily between 24.2% and 24.6% for each iteration of the training data.[23]

```
Accuracy on training data: 0.2463388442993164
Error on training data: 0.7536611557006836
```

```
Accuracy on test data: 0.25020480155944824
Error on test data: 0.7497951984405518
```

When evaluating this model with the test data, the accuracy increased from 24.63% to 25.0%. This indicates that the model is underfitting.

The accuracy applies a Softmax activation function to the calculate the output, which returns the probability of the individual nodes representing an accident type, and then selecting the accident with the highest probability.

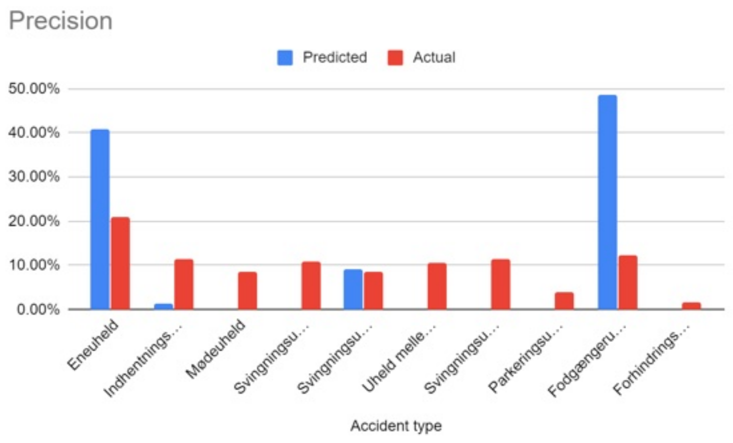


Figure 2.4: Distribution of accident types predictions and actual values for the Speed and Surroundings Characteristic Model

Looking at the precision of the predictions om figure 2.4, the results do not comply well with the distribution of the accident types in the actual test data.

### 2.3.2 Municipality and Surroundings Characteristic supervised learning model

With a level of abstraction, it is evident that the number of accidents in fx Copenhagen will be much higher than the number of accidents in fx Christianso, especially since Christianso is an island of where there are no cars allowed. But this we do not worry about, since this model does not intend to predict how likely any accident is, but how likely one of the ten specific types of accident is.

The feature engineering of the Speed and Surroundings Characteristic supervised learning model are very similar to the Municipality and Surroundings Characteristic supervised learning model, since the output variables are equal and the surroundings characteristics are in both sets of data represented. The main difference is the size of the input variables for the Municipality and Surroundings Characteristic model, which is significantly bigger.

This means that the two models can be applied with the same setup.

The municipalities do not need to have relationships with each other, and therefore the municipalities for this model is one hot encoded, so the data set has 98 municipality columns (Without Christians O since it it not represented in the data set of accidents). Equal to the previous model, the accidents have been one hot encoded as well. Therefore the layers for this model can not be equal to the layers of the previous model. Since there are 99 different municipalities, two different surroundings characteristics and ten different accident types, the total combinations of the data is 1980. Again the first layer will have this number of nodes applied with a sigmoid function.

To evaluate this model, the data set is randomly split into a test data set of 10% of the original data set and a training data set of 90% and the output is calculated using Softmax, equivalent to the Speed and Surroundings Characteristics Model.

The model is trained with 200 epochs/iterations of the data set, where the accuracy of each iteration has been steadily between 25.0% and 25.3% for each iteration of the training data. [24]

```
Accuracy on training data: 0.2530960738658905
Error on training data: 0.7469039261341095
Accuracy on test data: 0.24922151863574982
Error on test data: 0.7507784813642502
```

When evaluating this model with the test data, the accuracy decreases from 25.3% to 24.9%. This indicates that the model is overfitting.

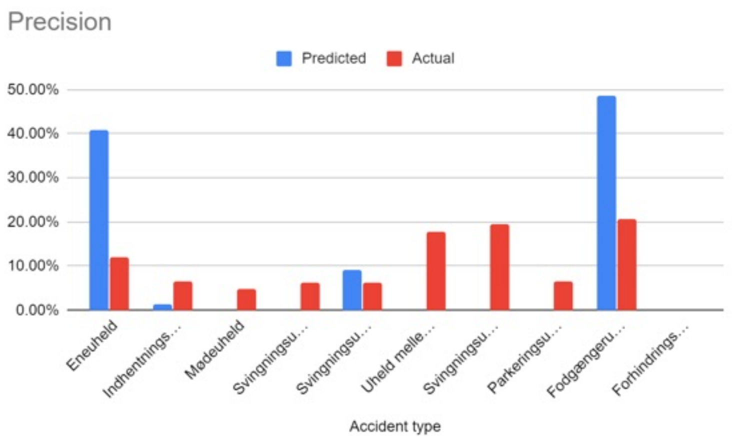


Figure 2.5: Distribution of accident types predictions and actual values for the Municipality and Surroundings Characteristic Model

For this model the pattern of the precision distribution is equivalent, as represented in the Figure 2.5

## 2.4 Statistical models

The output of these models are tables of factors of how likely accidents are to happen based on some parameters. This means storing the models as a tables in the Android application, where the program loads the table and lookup values based on the inputted parameters.

This model does not use the Machine Learning since this model does not need a neural network to do statistical probability calculations.

### 2.4.1 Temperature and Sun setting statistical model

This model is supposed to predict the probability of an accident based on the current outside temperature registered in the area where the user is located. For this model, the data set with accident records mapped to temperatures from the DMI data set.

The temperature data set from DMI is processed such that the format matches the accident records mapped to temperature. Doing this by recording all the records of average temperature within a time span by weekday, month and year, with no regards to the accident probability. This way, it is possible to record the values of how often temperatures were recorded, with one set for when the sun is up and one set for when the sun is down.

With a small data analysis of these records, there are no records of temperatures recorded with rounded values over 25 Celsius or less than -4 Celsius. So the shape for valid temperature inputs must be within -4 to 25 after the values in both data sets have been rounded.

To find the probability for a single temperature, the method for all four data sets will be to apply a simple probability formula to all of the temperature range:

$$P(H) = \frac{NumberOfFavorableOutcomes}{TotalNumberOfPossibleOutcomes}$$

This generates four columns representing the accident temperatures when the sun is up and when the sun is down and the actual temperatures when the sun is up and when the sun is down.

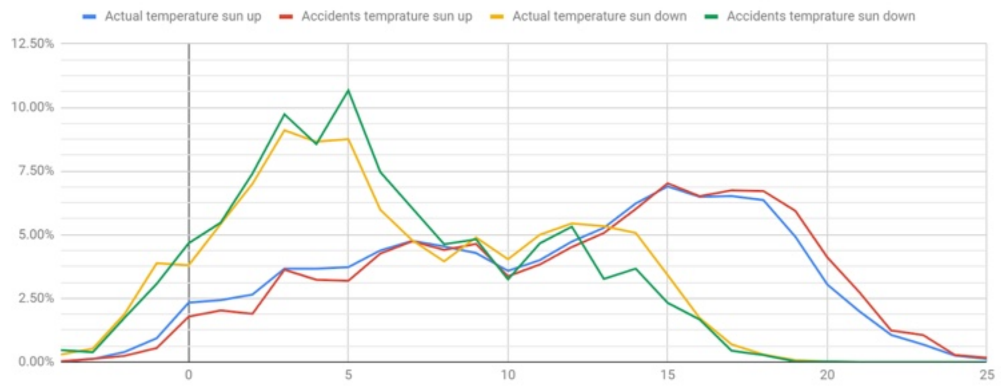


Figure 2.6: Accidents temperatures and actual temperatures categorized by sun setting. Data from DMI and UHELD4

Looking at the graph in Figure 2.6 there is a gap between the actual temperature distribution and the accident temperature distribution for both when the sun is up and when it is down. This gap is calculated by subtracting the accident temperature occurrence from with the actual temperature occurrence.

$$Risk(x) = accidentTemperature(x) - actualTemperature(x)$$

The value of the risk function can be positive and negative, where positive values represent a higher risk than average and negative values represent a lower risk than average. If this function is applied to all values in the temperature range, the following risk factors will be applied to the individual temperature parameters.

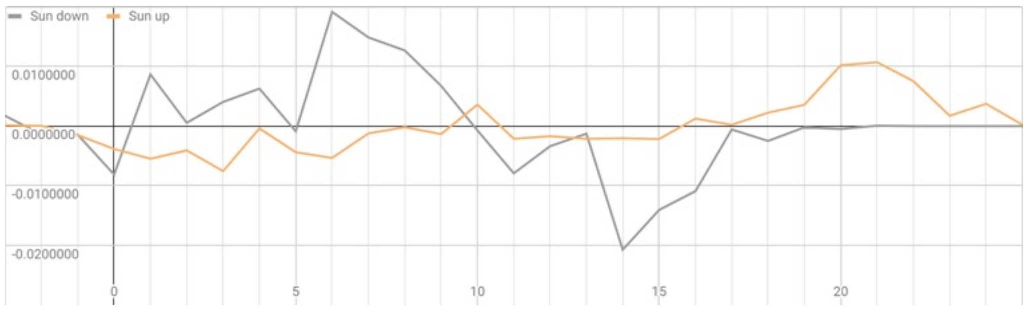


Figure 2.7: Temperatures risk predictions categorized by sun setting. Data from DMI and UHELD4

This means that fx night hours with temperatures of 6 Celsius degrees have a higher risk of an accident than fx night hours with 14 degrees Celsius.

### 2.4.2 Municipality, Age and Gender statistical model

The purpose of this model is to evaluate the probability of an accident based on geographically where the user is located and the users age and gender. This model also uses the data set BY2 which contains the same data but in a more detailed form.

The data set used for this model is the UHELD1, which contains municipality, age and gender. The accidents have 5 categories of age and two genders grouped into 99 different municipalities. The same structure has been built with the data set BY2, such that it is possible to compare the demographics data on accidents with the data on the actual demographics. For this model to make sense all entities representing humans under 18 years of age have been removed from all demographics data sets.

Before it is possible for the model to compare accidents demographics with actual demographics, the data in the accidents demographics and the actual demographics has to be represented as distribution by doing an average calculation of how represented one group is compared to another. By subtracting the distribution of the actual demographics with the accidents demographics, it is possible to get the data represented in the same format as the Temperature and Sun setting model.

This led the risk weights of individual municipalities to be represented as follows for fx Copenhagen.

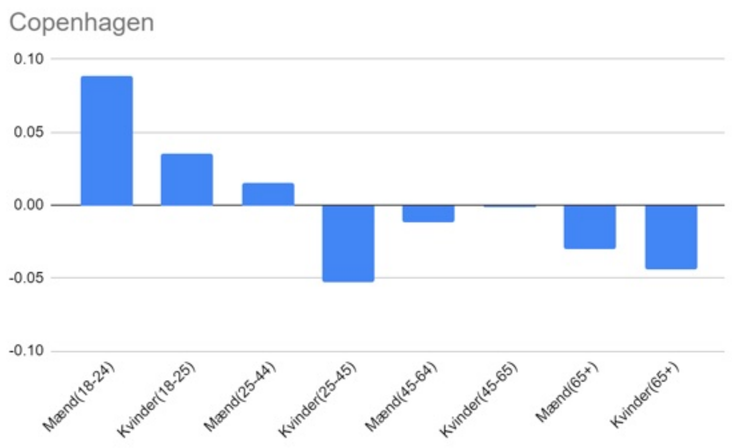


Figure 2.8: Distribution of accident risk the grouped age range and gender in Copenhagen. Data from UHELD1

The negative values represent a lower risk whereas positive values represent a greater risk. With some level of abstraction it is evident that young drivers will have more accidents than older drivers due to inexperience for most drivers young of age.

This is a general trend for most of the municipalities, which the following column charts also represents with an average calculation of each group by all municipalities.

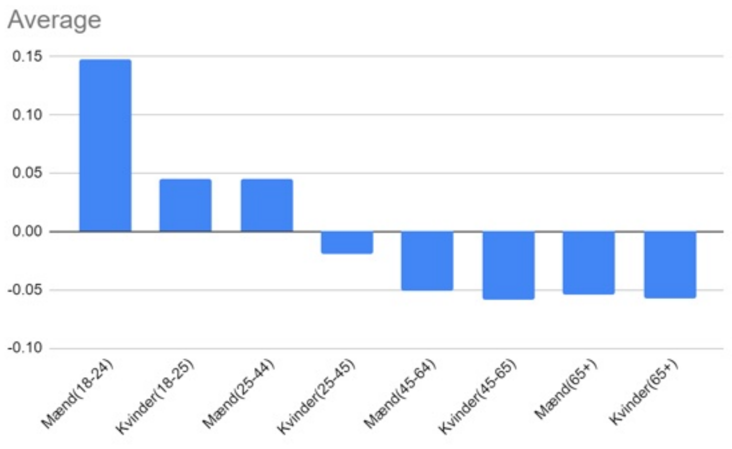


Figure 2.9: Distribution of accident risk the grouped age range and gender for the average municipality in Denmark. Data from UHELD1

This also represents the purpose of this model since the fx woman with an age between 45-64 in Copenhagen has a greater risk of an accident compared to the average risk of an accident for women with an age between 45-64.

Finally the model enables outputs, based on the input parameters of gender, age and municipality, represent how a specific location has a greater or a lower risk of an accident compared to the average.

## 2.5 Android Application

The android application is structured with one part handling the user data and one part handling the model and the predictions used. The android application is written in Kotlin language and using the Android API version 29, configured with a minimum Android API version of 26. Although the Kotlin language has been used, the functionality of the Java API from the Android API is available in the framework of this application stack.

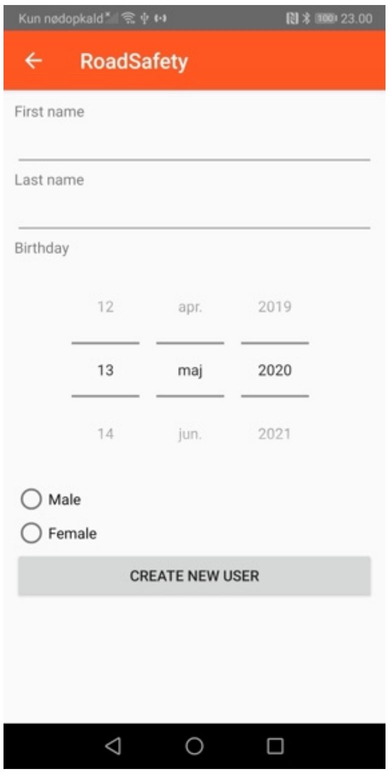


Figure 2.10: Create a user activity from the RoadSafety Project[10]

The Figure 2.10 demonstrates how the application allows a user to register an arbitrary number of users with a name, age and gender. The application is limited to only two genders and only drivers with an age of 18 years or older. These users are stored in a local database, using the Android Room DB, and a selected user has a reference in the shared preferences of the application.

### 2.5.1 Parameter Model

For the model to work it is necessary that the model has all the data parameters applied. This means that a single model will not work without the access to all the models parameters. All the parameters registries in at Data Transfer Object created by parameter model.

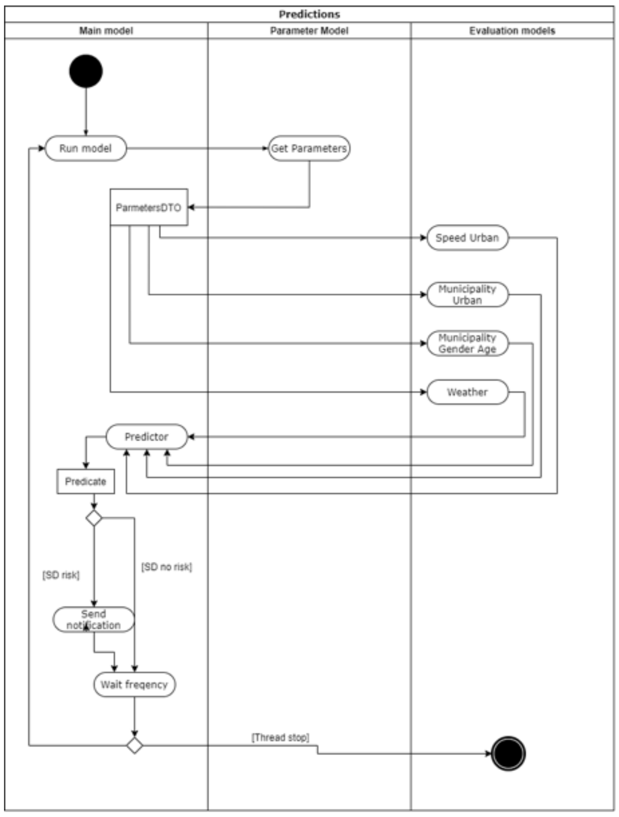


Figure 2.11: Activity UML Diagram of evaluating risks

As Figure 2.12 illustrates the model updates the parameters every time the model evaluates. The figure also illustrates that the model only check if it is suppose to stop after an evaluation, meaning that it can take up to 20 seconds to stop the model, depending on the frequency and when in the frequency the model is stopped. The parameters are in the program stored as Data Transfer Objects for all the models are as follow:

#### User: Name, gender and age

Comes from the User package of the application and are referenced when the model starts.

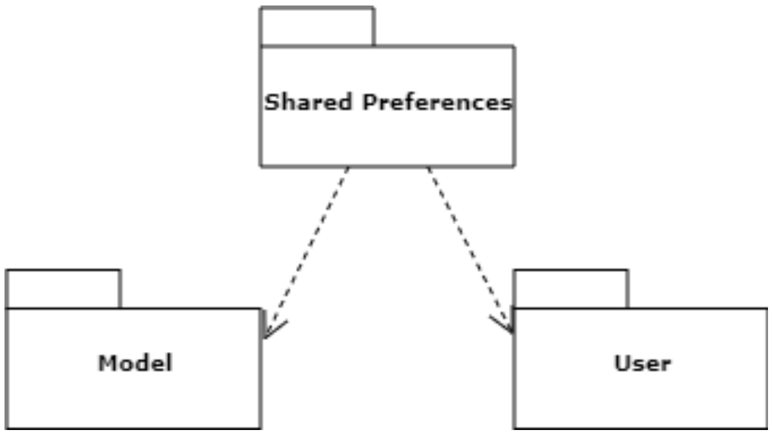


Figure 2.12: Package relationship for User and model package

The communication with the user package is only through the Shared Preferences in order to make the application data persistent, as demonstrated in Figure 2.11.

### Location: speed, longitude and latitude

Are referenced from the Android Fine Location, which uses the Cell Towers, GPS and Wifi to retain this information. The speed calculated by the fine location also categorized the speed limit of which the user is currently driving in. This is done with a margin of 5 km/h, such that a user driving fx 54 km/h will be categorized as a user driving in a 50 km/h zone.

### Municipality and surroundings characteristic

These data parameters are received from Danmarks Adressers Web API (DAWA)[2], which is an Web API that, given a longitude and latitude, can locate a road the user is driving on, by a nearest neighbor search. The road, which the API locates, contains information on the surroundings characteristics categorized by Urban area, Countryside or Cottage area, where the parameters in the android application categorise the Urban area and the Cottage area as Urban area and Countryside as a non-urban area. The road, which the API locates, also contains information about the Municipality where the user is located.

[2]DAWA, Danish Addresses Web API, Reverse Geo Codeing, `https://dawa.aws.dk/dok/dagi#reverse-geokodning(Visited05/05/20)`

#### Weather: Temperature and sun setting

The weather parameters are determined by Open Weather[3] which is an Web API, that given the parameters of the users longitude and latitude will do a nearest neighbor search for the nearest weather station. The data of the nearest weather station contains, what is referred to as, live weather information, which is used to obtain the information of the temperature and the sun setting information. The temperature is registered as Kelvin which the Android Application transforms into Celsius, which is the metric used in the model. The sun setting information is stored as Unix timestamps of when the sun sets and when the sun rises.

### 2.5.2 TensorFlow Models implementation

The two models Speed and Surroundings Characteristic and Municipality and Surroundings Characteristic supervised learning models, are implemented with the Python TensorFlow framework and converted into a TensorFlow Lite model which is a Deep Learning framework optimized for mobile devices [4]. This has been done using the TensorFlow Converter resources provided by the TensorFlow Lite framework [5]. This means that the two models share the format of the non-custom models provided by the Google ML-Kit. As a part of the Google ML-Kit, Google provides free hosting services of the models using the Firebase hosting service. For the purpose of this project the models are not hosted on the Firebase service, but a part of the resource assets of the application.

### 2.5.3 Statistical Models implementation

The models made using calculations on the data set have been exported as Commas Separated Values(CSV) with the files containing only the result of the calculations on the data sets. These files are read by the android application when initializing the data models and afterwards

[3]OpenWeatherMap, By geographic coordinates `https://openweathermap.org/current#geo(Visited05/05/20)`

[4]TensorFlow, Module: tf.lite `https://www.tensorflow.org/api_docs/python/tf/lite` (Visited 05/05/20)

[5]TensorFlow, Module: tf.lite.TFLiteConverter `https://www.tensorflow.org/api_docs/python/tf/lite/TFLiteConverte` (Visited 05/05/20)

stored in the memory of the application. The standard Java BufferRead [6] library is used to read the file.

### 2.5.4 Main Model package

The main model works as an extension to the main activity and runs evaluations on the parameters in a frequency and with a sensitivity based on user inputs and stores the result of all the evaluations in the memory to be evaluated in comparison with other results. As illustrated on Figure 2.12.

In order to decide whether or not a user should receive a notification, every time the main model evaluates new parameters the result of the new parameters is compared with all the previous results. This is done by calculating the standard deviation of all previous results with the following formula represented as the sigma function

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})}$$

Where x_i represents the individual values,  represents the mean and n represents the number of elements in the data set. In order to test if an evaluation represents a risk, the standard deviation is applied with percentiles based on the sensitivity of chosen by the user. Percentiles correspond with areas of the Normal Density Curve which can be applied with standard deviation. The percentiles or the Probit associated with the standard deviation are calculated by computing the inverse of the error function. These values are constants and available through Microsoft Excel or other computation software and for the purpose of this project the following three levels of sensitivity has been used:

- High sensitivity: 90% = 1.645
- Normal sensitivity: 95% = 1.96
- Low sensitivity: 99% = 2.576

In order to test if the inputs are of a high risk, the model applies standard deviation in order to normalize the data with the following function:

[6] Java, BufferedReader Java Platform SDK Documentation, `https://docs.oracle.com/javase/7/docs/api/java/io/BufferedReader.html` (Viseted 14/05/2020)

$$SND(p) = m + p * sd$$

Where p represents the percentiles, m represents the mean and sd represents the standard deviation.

When the test evaluates a risk, the type of risk is evaluated with the same SND function where the accident type most likely to happen is determined by which accident type returns the highest SND.

In order to improve the computation of these functions a simple data class stores the intermediate results possible, including the number of items and the sum of the items, such that the mean can be calculated by a single computation.

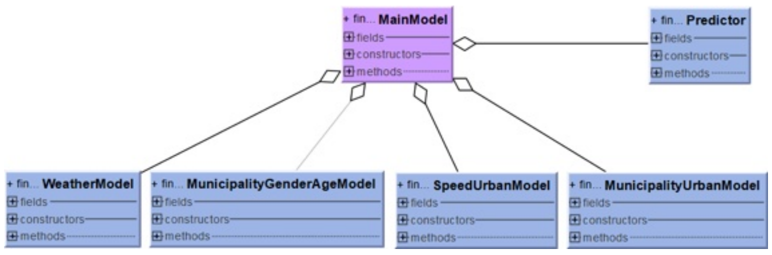


Figure 2.13: Main model implementation with models and predictors

The figure 2.13 describes how the Main Model is associated with all the other models and the predictor for an accident and the predictors for the accident types.

### 2.5.5 Android Auto

Android Auto can be used as an Android Application as well as software installed in the hardware of cars using Android Automotive. The Android Auto Application provides a simulation of the Android Automotive, just running on a phone instead of the hardware of the car. In order to send notifications to Android Auto the regular Android notification library is applied, although with some restrictions in order to secure the safety of the driver. This for instance limits the use of custom views, which means the Android Application sends notifications with a default layout and only a custom icon for the application.

Chapter 3

# Analysis

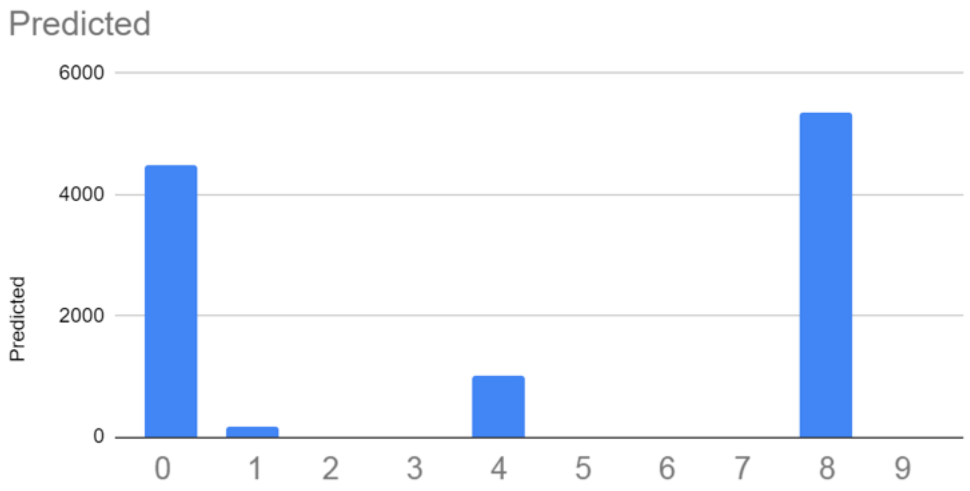
This chapter presents an analyze of the accuracy and precision of the Machine Learning Models and analyse how the Statistical Models are to be correct or incorrect. Furthermore, the implementation and the distribution of the Android application.

## 3.1 Models

This project contains two types of data of supervised and statistical data. Where the supervised data is implemented with the Keras Machine Learning API and the statistical data is implemented with statistical calculations.

### 3.1.1 Evaluation of Machine Learning models

The Municipality and surroundings characteristic model and the speed and surroundings characteristic model goes under the category of supervised machine learning since the model of this project has the individual outcomes for the individual parameters and supervised machine learning applies a mapping function with weights to predict the already known data. The accuracy for both the Municipality- and speed and surroundings characteristic model around 25% percent. Analysing how the predictions distributes shows that the model for some cases never predicts certain outcomes, which tells something about the model is not performing accuracy nor precisely.



0: one-vehicle accidents
1: vehicles on same road going in same direction without turning from road
2: vehicles on same road going in opposite directions without turning from road
3: vehicles on same road going in same direction, turning into T-junction, Y-junction, crossroads
4: vehicles on same road going in opposite direction, turning into T-junction, Y-junction,crossroads
5: vehicles on different roads meeting in crossroads without turning
6: vehicles on different roads meeting in T-junction, Y-junction, crossroads, etc. turning
7: accidents involving parked vehicles
8: accidents involving pedestrians
9: accident involving animals, obstacles, etc., on or above roadway

Figure 3.1: Based on data from evaluating [11] prediction from the urban/speed model

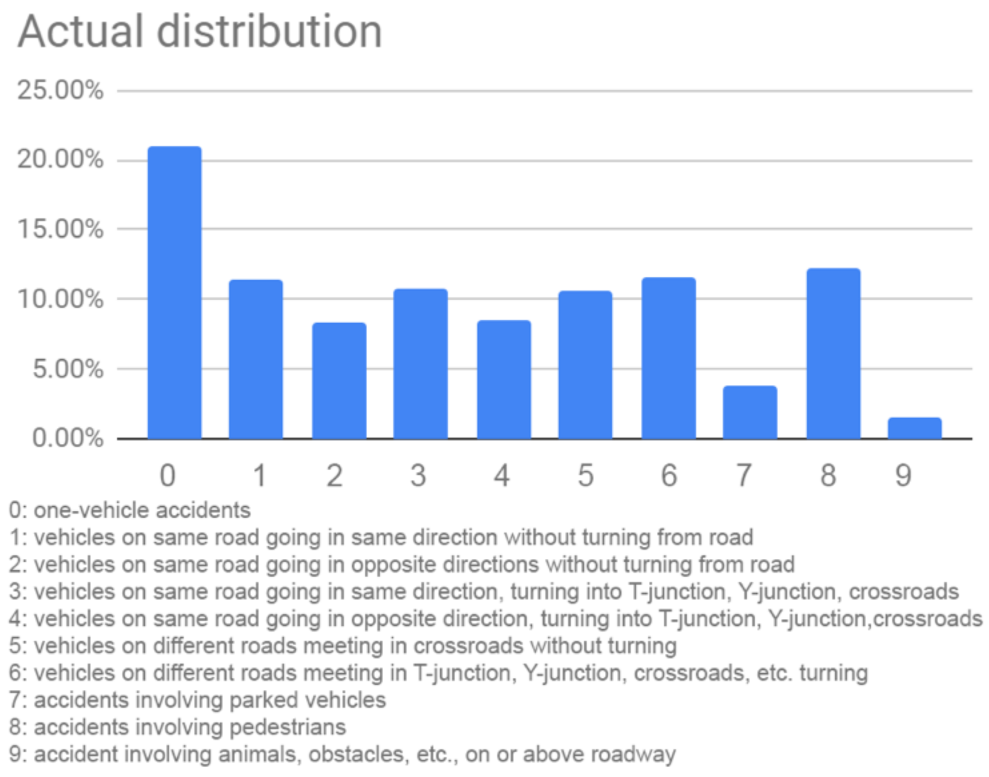


Figure 3.2: Based the distribution of the Speed and urban data set [17]

And looking at how the predictions are distributed and comparing them with the actual distribution, it would almost be possible to get the same accuracy as the model by always predicting the same *one-vehicle accidents* which is represented in more than 20% of the cases.

This raises a question of whether or not the build of the model have the correct mapping function which in the Keras Machine Learning API is equivalent to the layers and activation functions of the model.

The model has been selected based on the literature given from the *Keras documentation*[1] and *Machine Learning Projects for Mobile Applications: Build Android and IOS Applications Using TensorFlow Lite and Core ML* [1]. Also, analysing the model by testing with dummy data [21].

The Sigmoid activation function have the based on the ability to have a nonlinear function behaving like a linear function equivalent to a 2-way Softmax function, whereas a Softmax function applies categorical probability.

[1]Keras, Keras API reference, `https://keras.io/api/(visited11/05/20)`

Choosing the loss function based on the Keras API availability with a library of probabilistic losses and providing a Categorical cross-entropy.

While developing the model, doing small analysis the model tested with dummy data [21] enables a fast-forward testing of the model. This dummy data is structured such that all combinations of inputs maps to individual outputs. In theory, this means that it will be possible to get 100% accuracy.

When training some versions of the model with the dummy data, the accuracy of the model would output accuracy of as low as 60%.

```
Accuracy on training data: 0.6720778942108154
Error on training data: 0.32792210578918457
Accuracy on test data: 0.5974025726318359
Error on test data: 0.40259742736816406
```

Since the data is supposed to enable a 100% accuracy on both the training data and the test data, this indicates that this version of the model has been incorrectly structured. Although this is not the final model.

While developing and analysing the model, in order to test the hypothesis of the model having an incorrect setup, doing another test by training and testing the model with equivalent data. If a model is incorrectly structured, this reveals accuracy to be almost equivalent to the test on correctly split data sets:

```
Accuracy on training data: 0.6720778942108154
Error on training data: 0.32792210578918457
Accuracy on test data: 0.5974025726318359
Error on test data: 0.40259742736816406
```

Therefore, it can be concluded that the model when these tests were done are incorrectly structured. By changing the parameters, loss functions and the number of nodes, the final model is built by iterations of this process with testing the model on dummy data and evaluating the results with theory.

Since the dummy data set is smaller and therefore evaluates faster, the dummy data set enables a test of scaling iterations of the data set. Testing on the final model with the dummy data reveals that the accuracy already scales to accuracy of 100% after three iterations on the data set, and all iterations afterwards reveals the same accuracy.

Another experiment done to select the correct model has been testing and evaluating the data by dividing the test data set and train data set by year instead of randomly selecting the testing data from the data set. Although this has required a lot of trouble since the process of collecting the data from the StatBank and applying the data in the ML model has several subprocedures. This has not shown any improvements of the models test nor training accuracy.

Currently the test data is taken from the training data set using the train_test_split [2] from the SK Learn API, which is a python API providing a library of tools for data analysis in python. This has been since the split of the data set shows continuously equal distribution.

In order to improve this model, there have been experiments with different layer structures such that the model has various options of NN-layers, revealing that only when a layer has any number of nodes less than the shape of the training data, an visible change in the results will appear.

Doing improvements in the evaluation of the model by monitoring the models performance by various metrics, such as using the Categorical Accuracy revealing that the frequency follows the metrics of the accuracy itself, which means that no effect has been shown by expanding the metrics.

Improvements in the model have been by testing various loss functions. Diving deeper into the current loss function used by the model reveals that the Categorical cross-entropy function only applies correctly for hot-one encoded labels[3], which for this particular supervised machine learning model does apply. Also, the non-label data has been one hot encoded.

The Dummy Data experiments revealed that the model during the experiment would output accuracy states noncompliant with the level the theory applies. The theory states that a 100% accuracy would be possible for the dummy data experiment when all the combinations of data mapping individual values. The experiments reveal that the model have to change to one hot encoded labels, which reveals that the following results during the dummy data experiment:

[2] Scikit-learn.org. Sklearn.Model_Selection.Train_Test_Split https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html(Visited11May2020)

[3] Keras, https://keras.io/api/losses/probabilistic_losses/#categorical_crossentropy-function

```
Accuracy on training data: 1.0
Error on training data: 0.0
Accuracy on test data: 1.0
Error on test data: 0.0
```

This concludes that the model can apply the theory and that the model, in theory, is correct. Although, this does not apply for the actual supervised models with the accident data sets since all combinations of data do not map to unique values in the data sets. By this analysis of the model, concludes that when the model gets trained with the supervised training data, it should be able to scale to almost the highest accuracy possible for the model after only a very few iterations. This does not happen for the actual training of the model, as demonstrated in [24, 23].

Through the analysis and experimenting with dummy data, the Speed and Surroundings Characteristic and the Municipality and Surroundings Characteristic supervised learning models has the technical details of a Neural Network with two layers of an arbitrary number of nodes, based on the number of combinations in the data set. Applying a Categorical cross-entropy loss function and trained with 200 epochs/iterations.

### 3.1.2 Statistical models

The two models: Temperature and sunsetting model and Gender, age and municipality model, are not categorized as machine models as they do not use any learning algorithms or algorithms in general. These models consist of weights to the individual parameters of which the android application applies in, what is referred to as, the main model. The two models themself [14, 13] have not been applied with any programmatically science but applies Data Analysing Science.

The results of the evaluations are exported through a process, requiring a single procedure. For these models it enables the results to be stored as files which the Android Application can load. From a performance point of view, this is very efficient and enables a small workload for the mobile device. From a Computer Science perspective, this is not extensive.

The result of the calculation is tested with calculations on the accumulation and the intercompany in order to confirm that the weights are organised correctly. This is done by applying a sum function to the weights exported and confirming that the accumulation of weights would be zero, since the weights are not applied with any sort of activation functions before the models get exported.

The Approximate temperature and daylight data set generated explicitly for this model, contains some inaccurate results, since it maps accidents to the average temperature for an hour of the day on a weekday in a month due to the accident data set only containing the hour of the day, the weekday, the month and the year of the accident, and not describing the date of the month. With some level of abstraction, weekdays and weather do not have any meteorological relationship. Although analyzing the size of the data set, it is evident that the average temperature of the month will apply correct, since the number of accidents is higher than the records of days to be registered. The method of mapping to weekdays has been used in order to find more detailed weather data and thereby find more accurate patterns in the data set.

Analysing the performance of the applicated models, it would be tough to improve this. But if the model should be extended with maybe another year of data on the equivalent format of the data already used for the model, it would require a iteration of the whole process of converting the data, compute data, exporting the results and distributing the results to all the users.

### 3.1.3 Main model

The main model aims to manage the various risk factors from the four other models. This enables evaluations as a sequence, where the risk evaluations and the parameters can evaluate, update, and react to risks.

A part of reacting to risk is the evaluation of standard deviation, which serves two purposes: evaluations of any risk and for the evaluation for what type of risk the neural network predicts when there is evidence for a risk. The standard deviation describes a measurement of the variation in the data set. In this application, the standard deviation tests if a value is one of the highest values in the set. The standard deviation can also measure groups of values. If the application for instance were to be extended with a notification whenever there is a low risk of an accident. To apply this, the percentiles can be with the percentile values of low percentages or whatever percentage the model extends.

The Standard Deviation function distributes the data independently from the quantity since standard deviation only expresses how much the values differ from the mean value for the set. For instance, with the sensibility set to medium a user will receive notifications whenever a risk of an accident in a situation is to be evaluated higher than 95% of other evaluations of risks.

The standard deviation will, in theory, always return the same quantity of what is within a range of the percentiles. For instance, if we have 1000 different predictions and calculate the standard deviation with a 95% percentile, 5% of the predictions will always be classified as risks. This would mean that two different drives with an equivalent time of driving would receive the same amount of notifications, even if the risks them self have been very different. Nevertheless, for this implementation of standard deviation and the sequence of how the model runs prediction, this theory will not apply. Since whenever a model predicts an accident and sends a notification to a user, the notification will be stored until a new type of accident is predicted or no is to be predicted.

Therefore the value of evaluation itself does not matter, but how the value compares to previous values determines if a situation is a risk or not. This alerts the user whenever the risk situation of the drive changes and therefore presents a new risk. This enables the user to be alert of how the user should be careful.

In order to make sure correctly enforced standard deviations, there is an initialization of the main model, where the first 20 predictions are not to be evaluated if they are risks or not, but instead only stored in the local memory in order to seed the standard deviation with data before evaluating risks. This will give a user between 3 and 8 minutes, depending on the frequency by the users' input, actually to start driving before the model evaluates the risks. Another critical reason is that the model would evaluate incorrectly distributed data if this were not implemented.

An alternative to this method of implementing the standard deviation function could be to host a server where users post their risks, and these risks are evaluated compared to other drivers risk of an accident.

## 3.2 Android application and Android auto

The android application is currently capable of running on 60,8%[4] of all devices with the setup of minimum Android 26 API. The Android API

[4]Android Studio, Create New Project wizard

26 has extended resources for notification channels. In order to verify that high risks get predicted a small Unit test of the Main model has been made and the results are distributed as following:

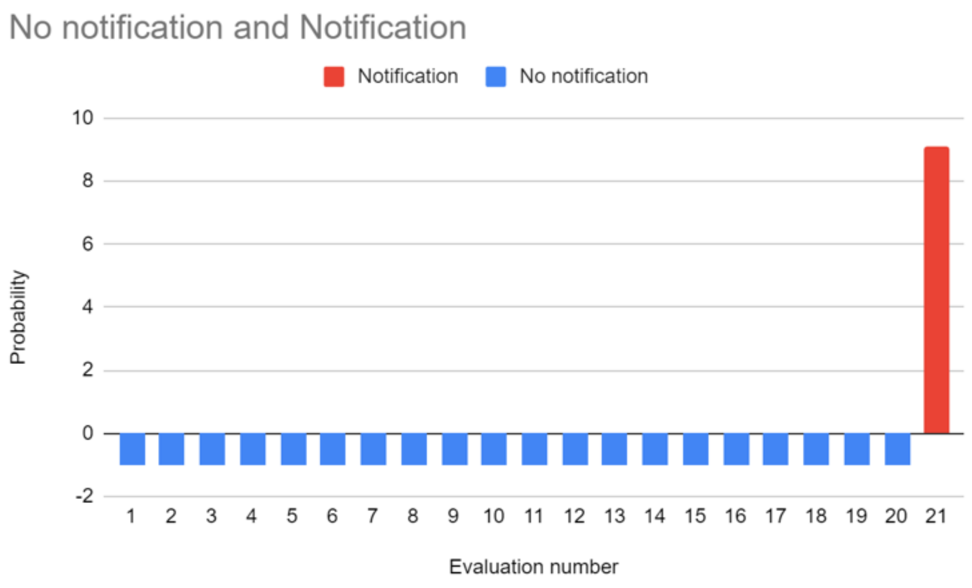


Figure 3.3: Predictions value and when the notifications sent. Based on the Unit Test test_device_receive_notification() from [10]

In order to make sure that users will use the application while driving, the application provides an option where the user can start the model themselves and an option where the model starts whenever the Android Operating System connects to a car or starts the Android Auto simulation. Using the UiModeManager provided by the Android API and a broadcast receiver, which recognizes car mode.[5]

If a user were to stop the notifications, the application itself provides options of whether or not the model should run and whether or not the application should automatically start when entering the car mode.

The user can also turn off the notification in Android settings, and for the users with Android 8.0 or above, the user can define the importance

[5]Android, UiModeManager Android Documentation, `https://developer.android.com/reference/android/app/UiModeManager.html#ACTION_ENTER_CAR_MODE`

of the notification themself. As default the importance is set to high, in order to make sure that noise, vibration and visual intent is available .[6]

The models are available to work within Denmark and only Denmark, since fx municipalities are only recognised by DAWA within Denmark and the Municipality and Surrounds Characteristics model only complies with Danish municipalities.

[6]Android, NotificationManager Android Documentation, https://developer.android.com/reference/android/app/NotificationManager#IMPORTANCE_HIGH

Chapter 4

# Conclusion

As described in the Introduction, this thesis is about experimenting with Machine Learning and mobile application to prevent car accidents, using simulation of the real in the models. The result of these simulations is computed and implemented in a Android mobile application compatible with the Android Auto Subsystem.

This bachelor thesis investigates, the option of merging the two supervised models, since they would be implemented together in the application. Although it is not possible to train the two models as one model with the two different data sets, due to the nature of Neural Networks where weights of one type of input would be equivalent to weights for another type of input. This addresses an underlying problem, where the data sets contain too low coupling in the relationship of the parameters. This expresses how the combinations of the parameters for the Neural Network, are not distributing any patterns in the data set, but are more likely to distribute equivalent to all the accidents, making the model predict unreliable patterns.

In theory models with a similar setup, data-structure and data would perform equivalent. As one of the models is overfitting, the other is underfitting this theory can not apply, due to the models not using the same data. The reason for one model overfitting and one underfitting is related to a differentiation in the data. In which caused the setup to be changed for the individual models since the one model has layers with up to 1900 nodes and one has layers with 180 nodes. However, the theory of each combination of inputs having a single node in relation will apply for both models, leaving the setup to be equivalent. Therefore it concludes that the nonequivalent under- and overfitting is related to data sets, where the Municipality, Urban and Accident Type data set has some relations which are not representative.

Regarding the Speed, Urban and Accident Type data set, the accidents are distributed primarily between the 50 km/h (Figure 2.1) and the 80 km/h (Figure 2.2). The measurement to handle this is that nodes in the Neural Network do not have relationships with each other. Although this limits the models' ability to predict only how likely an individual accident is since it is unknown how the speed limits on roads distributes, since the speed limit is retrieved as a parameter by the speed of the car. In order to use the speed of a driver as a parameter for the evaluation of how likely any accidents were to happen, some data of the actual speed limits distribution in Denmark is required.

For the supervised models it can be concluded the supervised data is too unpredictable to perform with a high level of accuracy. This tells something about the data not being relevant parameters of an accident. Due to the system of which the Authorities use to register accidents, there is no other data available, which conclude that the data recorded by authorities are not relevant in order to perform accurate predictions of accidents.

Implementing a statistical model of how speedlimites distributes and how speed limits of accidents distributes, would also require the application to detect the speed limit of a users current road. This projects Application implements the method of detecting the speed limit based on the speed of the user's. This can represent some incorrect data to the model if the user is speeding since the parameters for the model will not represent the actual speed limit of the road.

With a level of abstracting on the value of a temperature, it should be possible to record a temperature above 25 degrees Celsius. This does not comply with the records used for the Temperature and Surroundings Characteristic model since no records are above 25 degrees Celsius. This indicates that the model, to some extent, does not simulate the real world, and therefore, it can be concluded at some level the precision of the simulation is incompatible with the real world. In order to improve the model, the data could be more precise by evaluating temperatures, for instance, with decimals instead of integers.

Validating the results of statistical data has some limitations, compared to the supervised data, although the calculations and accumulations are valid in the model since it accumulates equivalent to the data sets. Although, the actual validation of the Temperature and Sun

Setting Model and the Gender, Age and Municipality Model happens when they are implemented in the model. As shown in Figure 3.3, the model manages to predict accidents whenever the conditions change to something that is evaluated as dangerous by the models.

Regarding evaluating the model with new data, all the steps in the creation of the model had to be repeated. As the SOLID principle of Software Engineering appeals that a software should be Open for extensions and closed for modifications, which is not correlated with evaluating new data. Therefore a significant improvement would be to implement the functionality of the data creation program as a part of a TensorFlow program and exporting the program as a TensorFlow Lite model equivalent to the format of the supervised models and hosting these models on the Firebase Service. Analyzing this pipeline would enable the Android Application to be automatically updated whenever the Firebase Service gets updated with a new model.

The Android extension of Android Auto/Automotive enables this application to deliver messages to the user by embedding the functionality of the Android Auto, in regards to starting the application when the Auto mode enable from the Android Operating System. This does not explicitly conclude that the Android Auto implementation is safe, but it does conclude that it is the user's responsibility to connect the system.

The measurement to safely deliver messages has to do with only delivering notifications in appropriate order such that a user will not constantly receive messages, but only receive messages in a relevant frequency. However, the application still enables the user to modify the functionality of the application to the user's preferences by changing the model frequency and sensibility. The importance of a message in Android, is defined as an option for the user to change with the android settings subsystem of notifications, which denotes the options for the user of messages will be received. However, this does not conclude that a message are send in a safe situation.

If a message is or is not sent in a safe situation, can not be conclude from this application. While the application might be able to detect if a user is accelerating or braking, by looking at the speed in a frequency, there is no scientific evidence of what is a safe time to send a message, from the experiments of this project. However, the measurement of defining the frequency of how often notifications are sent, set the user interaction to a minimal, which resolves interactions with the hardware while driving.

Overall the model with statistical historical data can, to some extent, predict the safety of a road based on information on the location and the person driving. Furthermore, the safety of a road evaluates by Machine Learning and the supervised historical data of accidents, how likely the specific types of accidents are, by location, and the speed information although the patterns revealed by the Neural Network do not indicate the parameters to correlate with the accident types. The Android application automates the functionality of the app when it has been initialized correctly, with a driver registered as a user.

# Bibliography

## Literature

[1] NG, K., 2018. *MACHINE LEARNING PROJECTS FOR MOBILE APPLICATIONS*. Birmingham: PACKT Publishing Limited.

[2] 2012. *Discrete Mathematics And Its Applications*. New York: McGraw-Hill Education - Europe.

[3] Team, Keras. *"Keras Documentation: Keras API Reference."* Keras.Io, `keras.io/api/`. Accessed 14 May 2020.

[4] *"Documentation | Android Developers."* Android Developers, 2020, `developer.android.com/docs`.

## Products

[5] *RoadSafety Android App*, RoadSafety.apk, Android Application, Anton Tobias Jensen

[6] *Speed and Surroundings Characteristic supervised learning model TensorFlow Lite*, speedUrban.tflite, TensorFlow Lite File, Anton Tobias Jensen

[7] *Municipality and Surroundings Characteristic supervised learning model TensorFlow Lite*, munUrban.tflite, Python File, Anton Tobias Jensen

[8] *Municipality, Age and Gender statistical model Output*, municipalityAgeModel.csv, CSV File, Anton Tobias Jensen

[9] *Temperature and Sun Setting statistical model Output*, weatherModel.csv, CSV File, Anton Tobias Jensen

Projects

[10] *RoadSafety Project*, RoadSafety.zip, Android Studio Project, Anton Tobias Jensen

Alternative download github.com/Statsministeriet/RoadSafety

[11] *Speed and Surroundings Characteristic supervised learning model source*, SpeedUrbanAccidents.py, Python File, Anton Tobias Jensen

[12] *Municipality and Surroundings Characteristic supervised learning model*, MunicipalityUrbanAccidents.py, Python File, Anton Tobias Jensen

[13] *Municipality, Age and Gender statistical model*, Municipality_Age_and_Gender_Statistical_Model.xlsx, Excel File, Anton Tobias Jensen

[14] *Temperature and Sun Setting statistical model*, Temperature_and_Sun_Setting_statistical_model.xlsx, Excel File, Anton Tobias Jensen

[15] *DMI Scrapper Project*, ScraperDMI.zip, Node Project, Anton Tobias Jensen,

Alternative download: github.com/Statsministeriet/ScraperDMI

Alternative demonstration: https://dmi-data.netlify.app/

[16] *Data Generator*, CreateDataProgram.zip, IntelliJ Java Project, Anton Tobias Jensen

Alternative download: github.com/Statsministeriet/CreateDataProgram

Data sets

[17] *Speed and urban data set*, Speed_and_urban.csv, CSV File, Anton Tobias Jensen

[18] *Municipality and urban data set*, Municipality_and_urban.csv, CSV File, Anton Tobias Jensen

[19] *Municipality, age and gender data set*, Municipality_age_and_gender.csv, CSV File, Anton Tobias Jensen

[20] *Approximate temperature and daylight data set*, Approximate_temperature_and_daylight.csv, CSV File, Anton Tobias Jensen

[21] *Dummy data set*, dummy.csv, CSV File, Anton Tobias Jensen

Other

[22] *Mail correspondence with Henning Christiansen from Statistics Denmark, Responsible for the UHELD statistics*, StatBankMail.pdf, PDF File, Anton Tobias Jensen

[23] *Screenshot of the training process of the Speed and Surroundings Characteristic supervised learning model*, SpeedUrbanTraining.PNG, PNG File, Anton Tobias Jensen

[24] *Screenshot of the training process of the Municipality and Surroundings Characteristic supervised learning model*, MunicipalityUrbanTraining.PNG, PNG File, Anton Tobias Jensen