# Introduction/Business Problem

The area I’ll investigate is about accidents and traffic flow at the highways. A lot of incidents can happen and impact whether the traffic goes smooth or not. Severe car crashes are one of them.

Based on a set of conditions like weather, light, road conditions and more, we’ll build and train a model to forecast emergency situations. This is to alert drivers for high risk so that they will adjust to drive more carefully or make other choices – like taking the train or staying home.

So, the idea is to predict the likelihood of accidents in regard to the traffic flow. As I’ve chosen to work with the shared dataset, I’ll be analyzing data from the Seattle Department of Transportation. The model should make sense in most other places too, but it’s probably important to be aware of the need for local adaption. What independent variables that should be used in the local model could vary.

We’ll build our model based on Machine Learning to predict the risk for severe accidents. We’ll clean the data and try to make sure we have the best possible set of variables to work with. Along the way we must make tests to chose the best machine learning models to work with, to optimize our results.

Our project should be of great value to both drivers and traffic-related governments and organizations, in perspective of the push for safer streets. This expanded to Seattle's "Vision Zero" in 2015, after Vision Zero has proved successful across Europe - originally implemented in Sweden in the 1990’s. Many cities across the U.S. have signed on to the idea that even just one death caused by these accidents is unacceptable and preventable. This study will hopefully reveal what, if any, measures we can take as individuals and municipalities to make travel in Seattle safer.

# Data

I chose to work in Anaconda and had to read in the shared data file with ‘data = pd.read.csv(‘filename’)’. That gave me some problems as the error “module ‘pandas’ has no attribute ‘read’” was returned. Research pointed at the pandas.py file could make some confusion, but I was not able to fix it.

Workaround was to turn to IBM and their environment, where I could drop the data file. I still encountered some problems with reading in the data, probably due to some displacement of the columns/cells/delimiters when saving the file locally. This time I was able to fix it by adding some code in the read function:

# df = pd.read\_csv(body)

df = pd.read\_csv(body, header=None, sep='\n')

df = df[0].str.split(',', expand=True)

The imported data then had the shape (194674, 4), so I needed to extract each of the 4 columns and then concatenate so that I had all data in one dataframe (194674, 82). Now it’s quite obvious that there are several columns I don’t need to keep working with, but I need a better understanding of what data there is in the respective columns to start cleaning. I’ll investigate them at a high level and make a rough selection:

* Selected: SEVERITYCODE, X (LONGITUDE), Y(LATITUDE), ADDRTYPE, SEVERITYCODE, COLLISIONTYPE, PERSONCOUNT, PEDCOUNT, PEDCYLCOUNT, VEHCOUNT, INCDATE, JUNCTIONTYPE, SDOT\_COLDESC, INATTENTIONIND, UNDERINFL, WEATHER, ROADCOND, LIGHTCOND, SPEEDING, ST\_COLCODE, ST\_COLDESC
* Not selected: OBJECTID, INCKEY, COLDETKEY, REPORTNO, STATUS, INTKEY, LOCATION, EXCEPTRSNCODE, EXCEPTRSNDESC, INCDTTM, SDOT\_COLCODE, PEDROWNOTGRNT, SDOTCOLNUM, SEGLANEKEY, CROSSWALKKEY, SEVERITYDESC, HITPARKEDCAR and all columns in the original column 2, 3 and 4

At this point I’ve two questions up for discussion:

1. I had this conception of working with regression and correlation analysis to identify the set of variables that influenced the probability for severe car accidents most. But I’ve realized that was a sort of misconception. On one hand since all variables needs to be numeric to work with correlation, the data would need to be prepared quiet a lot to be converted to numeric values (that could probably be done with the .get-dummies function -> (later on I realized I had to convert the variables anyway)). But also, on the other hand it seems obvious that there’s the 3 columns with conditions that has to make up the featured data, so the correlation idea is not relevant.
2. When selecting variables to include in the featured data set, I not only chose all the condition variables, but also several variables I had an idea I could use to predict the size or type of accident – like number of cars involved or where in the traffic the accident would be most likely to happen. I’m pretty sure that this would be possible/useful, but I deem it out of scope. Which means I should split my selected data into A) conditions that appears before the accident and B) characteristics of the accident. And then work further with A) only.
   1. SEVERITYCODE, WEATHER, ROADCOND, LIGHTCOND
   2. X (LONGITUDE), Y(LATITUDE), ADDRTYPE, COLLISIONTYPE, PERSONCOUNT, PEDCOUNT, PEDCYLCOUNT, VEHCOUNT, INCDATE, JUNCTIONTYPE, SDOT\_COLDESC, INATTENTIONIND, UNDERINFL, SPEEDING, ST\_COLCODE, ST\_COLDESC

What about missing data? There is no missing data in the severity code column, but in the 3 others it’s a lot. I think I should remove rows with missing values. Looking further into this I get that not only NaN occurs but also categories as ‘unknown’, ‘other’ and ‘ ‘ (empty), which is not useful as a condition, so I probably remove rows with these values as well. Yes, doing so gives me a dataframe with the shape of (23985, 4) which will the basis for further work.

# Methodology

## Balancing the data

Data imbalance reflects an unequal distribution of classes within a dataset, so that one class outweight the other. We have a binary outcome of the severity class we are to train; severity code 1 or 2. And we know the dataset is imbalanced with 131.362 code 1 (property damage only) and 57.673 code 2 (people injured). So, we must fix this problem to avoid training the classification model with poor input. Balancing can be done by either undersample, which is downsizing the abundant class, or oversample, which is to generate synthetic data in the rare class to grow it bigger. As the number of observations in our rare class is so high (57.673) it seems obvious to go for undersampling.

During the process of cleaning the data, the total number of rows with valid data is heavy reduced and the imbalance is reduced consequently – down to 12.897 vs 11.088 units. That does not affect the choice of balancing via undersampling. Balancing then gives me a dataframe with the shape of 22.176 (2 x 11.088, 4) to continue with.

# balancing the severitycode column

df15 = df14.loc[df14['SEVERITYCODE'] == '2']

df16 = df14.loc[df14['SEVERITYCODE'] == '1'].sample(n=len(df15))

## Playing (to) smart

Ok, then I had this idea evolving, to both visualize where the accidents happened by mapping with ‘add.child’.

And include the date of each accident, to convert to weekdays to work with the days of the week in bins of weekend, Monday, midweek, Friday to look for dependencies based on weekdays.

The idea of including the weekdays got lost on me not being able to convert 'incdate' to a datetime format - on the first try ..

Regarding mapping I came somewhat further, setting up a dataframe with severity codes together with latitude and longitude coordinates. With Folium I created this Seattle map. But when setting up the add\_child code I experienced some syntax error messages. After having given it a shot solving it, I prioritized to leave it as it’s out of scope.

## Datatypes

By now all data in our 4 columns are of the class ‘object’. One of the major problems with machine learning is that a lot of algorithms cannot work directly with categorical (non-numeric) data. Categorical data are variables that can take on one of a limited number of possible values – in our case such as the ‘Weather’ column can be; clear, raining, overcast, snowing etc. In the step of data processing in machine learning, we can perform a One-Hot encoding to prepare our data to be numeric. One Hot Encoding is a process that is applied to categorical data, to convert it into a binary vector representation. With One-Hot Encoding, the binary vector arrays representation allows a machine learning algorithm to leverage the information contained in a category value.

Our target value, the severity code is already 1 or 2, but to avoid any potential trouble with the severity code being of type object I changed that column to be integer – which is done by -> df17["SEVERITYCODE"] = df17["SEVERITYCODE"].astype("int")

Next step is to use one hot encoding technique to convert the three remaining columns from object to binary variables. To convert the 3 other columns (weather, light- and road condition) this is the code used ->

df18 = pd.concat([df17,pd.get\_dummies(df17['WEATHER'])], axis=1)

## Machine learning algorithms for classification

Now closing in on working with different machine learning algorithms to see with what probability we can say something about the risk for severe car accidents. In other words, the use of a classification model. Other types of machine learning models could help us solve recommender issues, estimating a continuous value, uncover clusters in the structure of data and more. But here we need to predict either severity code 1 or 2 based on the set with conditions.

In machine learning classification attempts to learn the relationship between a set of feature variables and a target variable of interest – respectively the conditions and the severity code. Essentially, many problems can be expressed as associations between feature and target variables.

Given a set of training data with the target labels, classification determines the class label for an unlabeled test case. We have our dataset with predefined labels available, which is also why we can call this supervised learning – having the labels we can train the model we need to build a model to be used to predict new cases.

The types of classification algorithms and machine learning includes Decision Trees,

Naive Bayes, Linear Discriminant Analysis, K-Nearest Neighbor, Logistic Regression, Neural Networks, and Support Vector Machines.

## Libraries

To work with so heavy math and stats as is required solving prediction tasks with machine learning, we are in need of the algorithmic library Scikit-learn, which contains tools for statistical modelling.

## K-Nearest Neighbor

How can we find the class of a new case? Can we find the closest case and assign the same class label to our new case? Can we say that the label of a new case is most probably label X, because its nearest neighbor is also of class X? Yes, we can. In fact, it is the first nearest neighbor and what this model is about.

But can we trust the theory behind this model – what if the nearest neighbor is some sort of an outlier? Rather than just choose the first nearest neighbor we might choose the z closest neighbors and do a majority vote among them. Working with the model you can do tests and find the optimum number of neighbors to choose.

We split our X (feature set) and Y (target) data in to respectively a train and test set, to train the model with the train data and thereafter test it based on the test data.

During my work process I got K1 (the closest neighbor only) as optimum in my first try, which I delved into. I discovered that it was not good as it’s a sign of overfitting and to heavy complexity. I reduced the number of variables in each condition category by eliminating those with a very low occurrence – like less than 20 incidents. I then got e.g. K8 but it did not change very much on the accuracy score. It remained around 0,5 ..

Et bilde som inneholder tekst

Automatisk generert beskrivelse

Running the code several times while I’m working with building the model, and hereby randomly splitting my data into train and test data, I experienced that the optimum number of neighbors changed quiet a lot – I think between 3 and 8. But it was not really reflected in the accuracy score which remained around 0,5 with very little variation.

Challenging the number of neighbors, I did a test with the 1.000 nearest neighbors and found that after approx. K15 (peaking about 100) the accuracy score raised from 0,5 to 0,57.

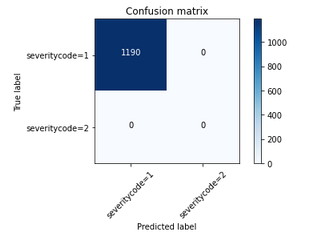
Et bilde som inneholder tekst

Automatisk generert beskrivelse

## Confusion Matrix

A way of looking at the accuracy of a classifier is to look at Confusion Matrix, which gives us a matrix with the numbers of true/false positive and negatives.

But here I’m running in to some sort of problem! And I’m not able to see through what’s happening. Point is I get a meaningful number of true positives, but I get 0 false positive, true negative and false negative – which does not make sense!



It’s said to be a typical problem of imbalanced data, but as my data is balanced that should not be the case. Other keywords I came across delving with the issue was among others over-/undersampling and outlier detection, but it all summed up to the conclusion there is no basic solution for this kind of problem.

I accept I have this problem and deem it out of scope to investigate it further as of now – also because the next set of measurement we will look at, seems not to be affected by this result, even it’s based on the outcome of the confusion matrix.

## Classification Report

Producing a classification report gives us precision, recall and f1-score.

**Precision** is a measure of the accuracy provided that a class label has been predicted. It is defined by TP / (TP + FP)

**Recall** is true positive rate. It is defined as TP / (TP + FN)

F1 score it’s the harmonic average of the Precision and Recall. The F1 score reaches its best value at 1 and worst at 0.

And as this report gives us meaningful scores, I think the problems with the Confusion Matrix is not a showstopper.

Et bilde som inneholder bord

Automatisk generert beskrivelse

## Decision Trees

Decision trees are built using recursive partitioning to classify the data, to determine which attribute is the best or more predictive to split data based on the feature.

Building a decision tree here, this classifier also gives me an accuracy of 0,57.

## Support Vector Machine

Using the SVM algorithm and evaluating it with the F1 Score gives me 0,56.

Another evaluation model is the Jaccard\_score. The idea behind the Jaccard\_score index is that the higher the similarity of the two groups y\_true and y\_predicted is, the higher the index. The score of this evaluation metric is defined as the size of the intersection divided by the size of the union of two label sets.

So, what we ideally are looking for is a score of 1 – that is if the entire set of predicted labels for a sample strictly matches with the true set of labels. Our SVM model is scored to 0,42.

## Logistic Regression

This is regression used for classification or predicting a binary variable.

In most kind of linear regression, we might try to predict a continuous value of variables such as the price of a car. Logistic regression is a variation of linear regression, useful when the observed dependent variable, y, is categorical. It produces a formula that predicts the probability of the class label as a function of the independent variables. This formula is called the Sigmoid function.

We did build a logistic regression model.

The output of a logistic regression classifier is the probability of a class label, instead of the

label. Logarithmic loss (also known as Log Loss) measures the performance of a classifier where the predicted output is a probability value between 0 and 1.

We can calculate the Log Loss for each row with the Log Loss equation, which measures

how far each prediction is, from the actual label. Then, we calculate the average Log Loss across all rows of the test set. So, the classifier with lower Log Loss has better accuracy. As our Log Loss score turns out to be 0,69, it’s an indication we are not able to predict at a level we want to.

## As a final note I’ll just mention a few things

I got more or less identical scores on the train and test set, I take that as an expression that the models is not affected by overfitting:

*Train set Accuracy: 0.5611612175873731*

*Test set Accuracy: 0.5545536519386834*

I did try to reduce the number of conditions in the feature set. One attempt based on conditions with high volume of incidents and one out from a table with the correlations. But that did not affect the accuracy of the model notably.

And I’ve not normalized the data set as I only work with binary values.

# Results

The accuracy score of the KNN model could be optimized to 0,57. Which seems to close to 50/50 to make up for a useful model.

The different scores from the Classification Report and the Decision Tree showed the same level of accuracy.

The SVM algorithm gave a F1 score of 0,56 and the Jaccard Score 0,42.

The Log Loss score is 0,69, which is not great as with this index we are looking for a score as close to 0 as possible.

All of these results seem to be roughly at the same level. And maintaining that 50/50 feeling, which is the wrong feeling trying to predict a binary outcome ..

# Discussions

It’s been a great challenge to work with this project, a really educational experience.

I think I got good training in using the different machine learning algorithms at the end of the process. Also doing the final wrangling of the data set. It was in some way a greater challenge in the beginning to get the overview over the job and to understand the data. I still feel a bit uncomfortable about the process with finding the best data to work with. It might have been another way of using the data to make a better or more useful prediction?

The result of the Confusion Matrix concerns me as well. Something is wrong in that part. It might not play an important role as the following scores did not seem to be affected.

At the end I don’t think my idea of grouping the different days of the week – which I missed out on – would have made a difference, but I still like the idea.

Visualizing the car accidents grouping them on a map is the one I would have chosen to work further with. That is probably a basic traffic planning tool. Maybe then it would have been possible to dive into geographically high incident groups and support with findings from the conditions in the data set?

Going back to Vision Zero, one recommendation could be to include more factors or do a better and more holistic data collection of car accidents. The human behind the driver, the situation in the car, the car itself and so on – as it’s probably a more complex set of triggers than the more obvious conditions – that plays the critical role.

# Conclusion

We were trying to build a model to predict a higher risk for severe accidents, given a certain set of conditions. For practical use, to warn drivers to take extra care under those circumstances.

Theoretically it was possible with a probability of more than 50%.

But for implementation as a tool for traffic warning is does not seem useful. The uncertainty is too big to be taken as a real-life consideration.

Probably because it’s the driver and some kind of combination among an enormous set of other factors that is what triggers any kind of car accident. And as we all want to avoid taking part in any form of car accident, we do adjust for important but obvious conditions like the weather and light and road conditions already.