Car Accident Severity Study based on Data from Seattle, WA

Applied Data Science Capstone Project, Coursera

Submitted by: Marco Schwab

2020-11-06

CAR ACCIDENT SEVERITY STUDY BASED ON DATA FROM SEATTLE, WA

1. INTRODUCTION

Every day, about 3,700 people die in road traffic worldwide. Road traffic accidents are an increasing problem even in times of modern vehicle technology and well-developed infrastructure. According to the World Health Organization (WHO), 1.35 million people die in traffic accidents every year. According to the Federal Statistical Office, the causes of traffic accidents involving personal injury in Germany are:

- Driver mistakes
- Road conditions, weather influences, obstacles
- Technical defects
- Incorrect pedestrian behavior

This information can probably be transferred to other countries.

If it were possible to predict on the basis of given real time conditions (Driver, Road, Vehicle etc.) how likely it is that an accident involving personal injury will occur, this would be a valuable aid and decision-making tool for the following interest groups:

Stakeholders¶

- Trafic participants
- Responsible cities and road planners
- Car Manufacturer
- Motor vehicle insurer
- Emergency and Health care personal

2. DATA

The dataset used in this study is based on a document called 'Collisions—All Years' from the organisation 'SDOT Traffic Management Division, Traffic Records Group'. It includes all types of collisions counted by the Seatle Police Department and Traffic records in the city of Seatle, WA in the timeframe beginning 2004 until present. The document is weekly updated.

The project purpose is to analyze and predict the severity of an accident based on some particular features as follows:

Table 1: Features used to predict the severity of a potential accident

Feature	Description
INATTENTIONIND	Whether or not collision was due to inattention. (Y/N)
UNDERINFL	Whether or not a driver involved was under the influence of drugs or alcohol.
WEATHER	A description of the weather conditions during the time of the collision.
ROADCOND	The condition of the road during the collision.
LIGHTCOND	The light conditions during the collision.
SPEEDING	Whether or not speeding was a factor in the collision. (Y/N)

The diagram below shows the amount of data given for each feature and for the target value (severity) in the original dataset.

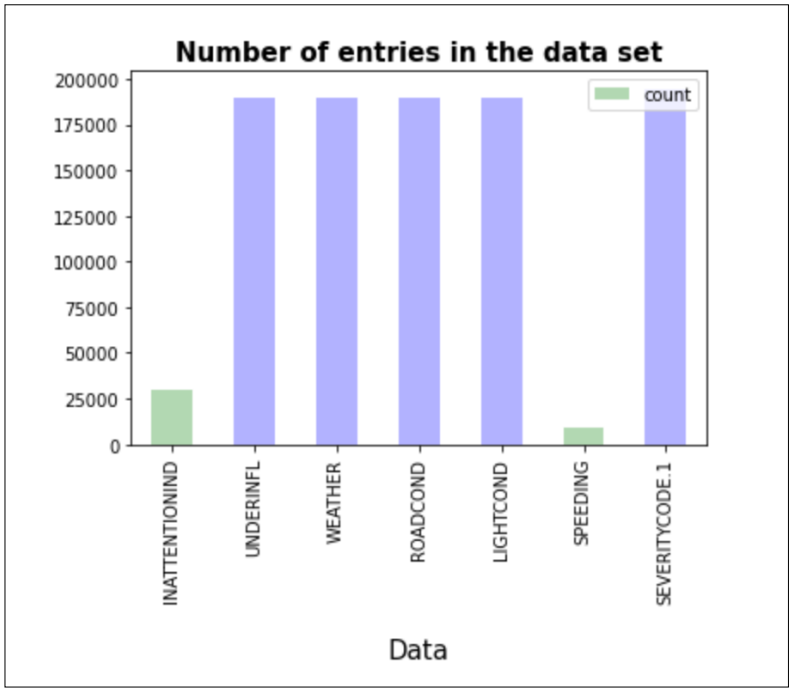

Figure 1: Entries in the given data set

3. DATA CLEANING

At the time, when this study was prepared the data consists of 194673 entries (counts of collisions). Each collision was closer described in categories shown in the 37 columns of the dataset. The number of usable entries in the columns of the datasets varies significantly. For this reason, a data cleaning was firstly done on the set of data. Missing or not usable entries were deleted, others were transformed or recoded. During the data cleaning the following steps were performed:

- Data were imported: **df = pd.read_csv('Data-Collisions.csv', index_col=0)**
- Questions marks were replaced by NaN: **df.replace("?", np.nan, inplace = True)**
- Changing severity code from 1=Property Damage Only and 2=Physical Injury to 0=Property Damage Only and 1=Physical Injury

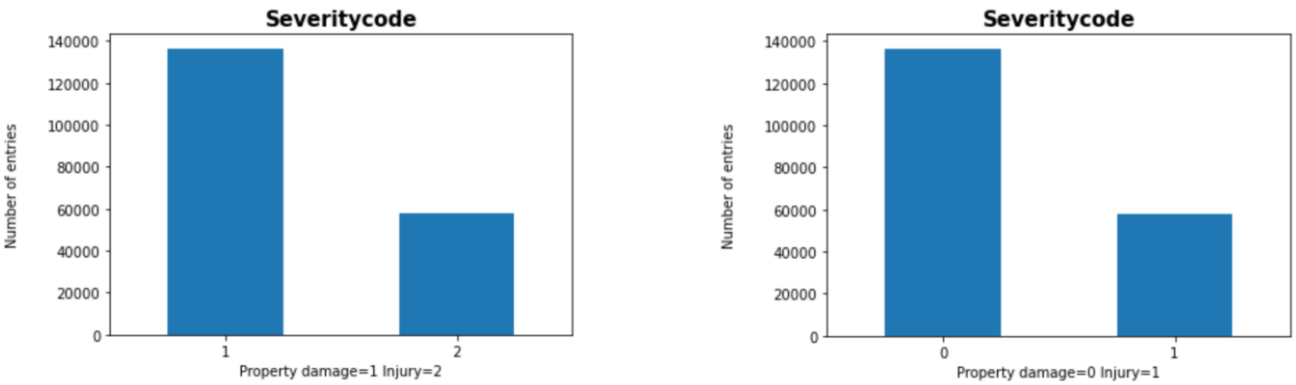

figure 2: Severity Code changed to 0=Property Damage Only and 1=Physical Injury

4. DATA PREPARATION

The data preparation section consists of:

- Feature Selection
- Recoding to numeric values
- Replace Unknown or NaN to statistical representative values
- Convert data types to integer

Feature Selection

The target variable for this analysis was set as severity code, which should be predicted by the following variables:

- **INATTENTIONIND** Whether or not collision was due to inattention. (Y/N)
- **UNDERINFL** Whether or not a driver involved was under the influence of drugs or alcohol.
- **WEATHER** A description of the weather conditions during the time of the collision.
- **ROADCOND** The condition of the road during the collision.
- **LIGHTCOND** The light conditions during the collision.
- **SPEEDING** Whether or not speeding was a factor in the collision. (Y/N)

Recoding to numeric values¶

Most of the variables were described as object in string format. To apply Machine Learning algorithm these variables were coded as numeric values in the following way:

- Coding key for INATTENTIONIND (0 = No, 1 = Yes)
- Coding key for UNDERINFL under influence (0 = No, 1 = Yes)
- Coding key for SPEEDING (0 = No, 1 = Yes)
- Coding key for LIGHTCOND Light Conditions(0 = Light, 1 = Medium, 2 = Dark)
- Coding key for WEATHER(0 = Clear, 1 = Overcast and Cloudy, 2 = Windy, 3 = Rain and Snow

- Coding key for ROADCOND Road Conditions(0 = Dry, 1 = Muddy, 2 = Wet)

Rows without values were deleted.

Replace Unknown or NaN to statistical representative values

¶

Unknown or NaN values were handled as follows:

- Missing 'WEATHER' values were replaced by the most frequent which is 0
- Missing 'lightcond' values were replaced by the most frequent which is 0
- Missing 'roadcond' values were replaced by the most frequent which is 0

Finally all datatypes were converted to integer.

5. METHODOLOGY

The Methodology used in this project consists of the following steps:

Data understanding and Data cleaning

In this step the content of the given data was examined, and first considerations were made, how the data analysis could look like. Stakeholders are identified who could benefit from the analysis. Further, the data are cleaned up to make it easier to use Python algorithms later. A selection of the variables necessary for the analysis is made.

Data preparation

In this section the features were defined which are used to predict the severity of a potential collision. Object type features were recoded to numeric values (integer) to apply Machine Learning algorithm on them. Unknown values were either deleted or assessed by representative values. The following features were selected:

- **INATTENTIONIND** Whether or not collision was due to inattention. (Y/N)
- **UNDERINFL** Whether or not a driver involved was under the influence of drugs or alcohol.
- **WEATHER** A description of the weather conditions during the time of the collision.

- **ROADCOND** The condition of the road during the collision.
- **LIGHTCOND** The light conditions during the collision.
- **SPEEDING** Whether or not speeding was a factor in the collision. (Y/N)

They were recoded as follows:

- Coding key for INATTENTIONIND (0 = No, 1 = Yes)
- Coding key for UNDERINFL under influence (0 = No, 1 = Yes)
- Coding key for SPEEDING (0 = No, 1 = Yes)
- Coding key for LIGHTCOND Light Conditions(0 = Light, 1 = Medium, 2 = Dark)
- Coding key for WEATHER(0 = Clear, 1 = Overcast and Cloudy, 2 = Windy, 3 = Rain and Snow
- Coding key for ROADCOND Road Conditions(0 = Dry, 1 = Muddy, 2 = Wet)

Unknown or NaN values were handled as follows:

- Missing 'WEATHER' values were replaced by the most frequent which is 0
- Missing 'lightcond' values were replaced by the most frequent which is 0
- Missing 'roadcond' values were replaced by the most frequent which is 0

Machine Learning Section

The following ML algorithm were applied on the prepared features. The aim of the task was to develop a method which can predict the severity (injury or property damage) of a collision when the data according the feature selection are known.

- Decision Tree
- K nearest Neighbor (KNN)
- Logistic Regression

To achieve this, the data were split into a train and test set using the train_test_split from *sklearn.cross_validation*.

Train_test_split returns 4 different parameters. They are named:

X_trainset, X_testset, y_trainset, y_testset

The train_test_split will need the parameters:

X, y, test_size=0.3, and random_state=3.

The X and y are the arrays required before the split, the test_size represents the ratio of the testing dataset, and the random_state ensures that we obtain the same splits.

The performance and precision of the applied ML functions, are visualized in a confusion matrix. In addition the F1 score were calculated to evaluate the accuracy of the analysis. These scores will help compare the ML algorithms.

6. RESULTS

6.1 Decision Tree Analysis¶

After sufficient training and testsets were defined, the Decision Tree Analysis was performed using the Scikit learn library. First an instance of the Decision Tree Classifier called DT was created. Entropy was taken as criterion and the max depth was 6. Predictions were then made on the testing dataset and stored into a variable called yhatDT.

The performance and precision of the DT functions, were visualized in a confusion matrix. In addition the F1 score were calculated to evaluate the accuracy of the analysis. In addition a classification report were printed.

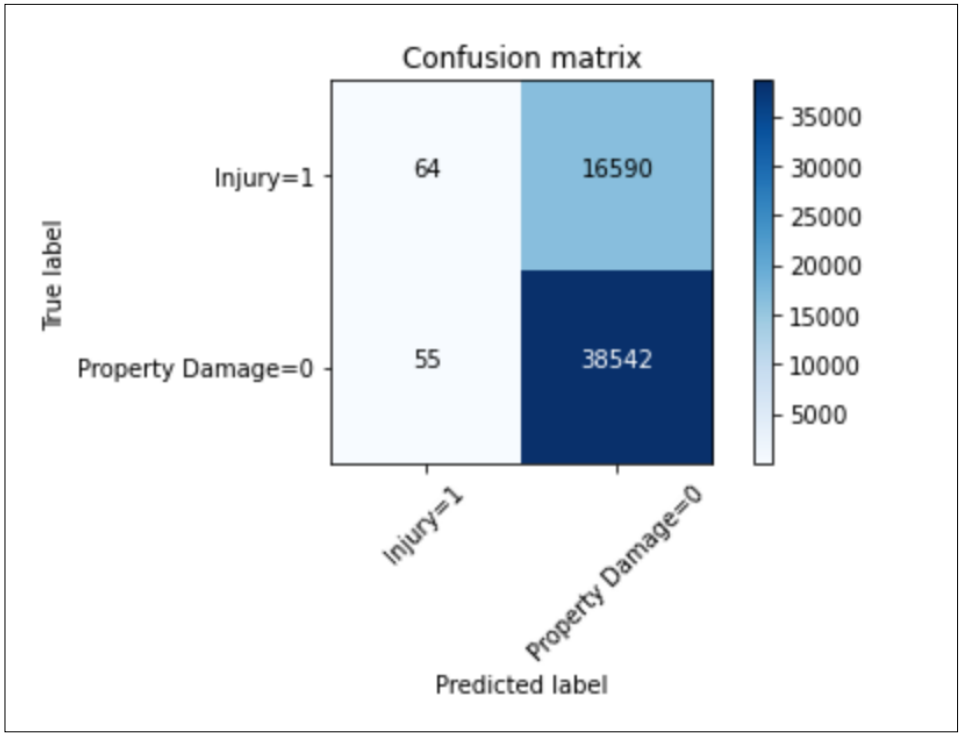

Figure 3: Confusion Matrix Decision Tree

Confusion Matrix Decision Tree

According the confusion matrix the DT algorithm detects 64 injury cases which are in real also injury cases. In addition 38542 cases were detected as Property Damage which are also Property Damage cases in reality. 16590 Injury cases were wrongly detected as Property Damage cases and 55 Property Damage cases were detected wrongly as Injury cases.

Based on the count of each section, the precision and recall of each label can be calculated:

- Precision is a measure of the accuracy provided that a class label has been predicted. It is defined by: precision = TP / (TP + FP)
- Recall is true positive rate. It is defined as: Recall = TP / (TP + FN)

Precision and recall of each class can be calculated.

F1 score: Now the F1 scores for each label based on the precision and recall of that label can be calculated.

The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. It is a good way to show that a classifer has a good value for both recall and precision.

And finally, the average accuracy for this classifier is the average of the F1-score for both labels.

Table 2: Classification Report Decision Tree

	Precision	Recall	f1-score
0	0.70	1.00	0.82
1	0.54	0.00	0.01
accuracy			0.70
macro avg	0.62	0.50	0.42
weighted avg	0.65	0.70	0.58

The following calculation shows how the values in the classification report were calculated:

Precision (0)=TP/(TP+FP)=38542/(38542+16590)=**0.699**

Recall (0)=TP/(TP+FN)=38542/(38542+55)=**0.999**

Precision (1)=TP/(TP+FP)=64/(64+55)=**0.538**

Recall (1)=TP/(TP+FN)=64/(64+16590)=**0.004**

6.2 K Nearest Neighbor (KNN) Analysis

The second ML approach applied on the training and test datatset was the K Nearest Neighbor (KNN) algorithm. The functions were also taken from the Scikit Learn library.

First the best k was evaluated by calculating the accuracy of k between 1 and 10.

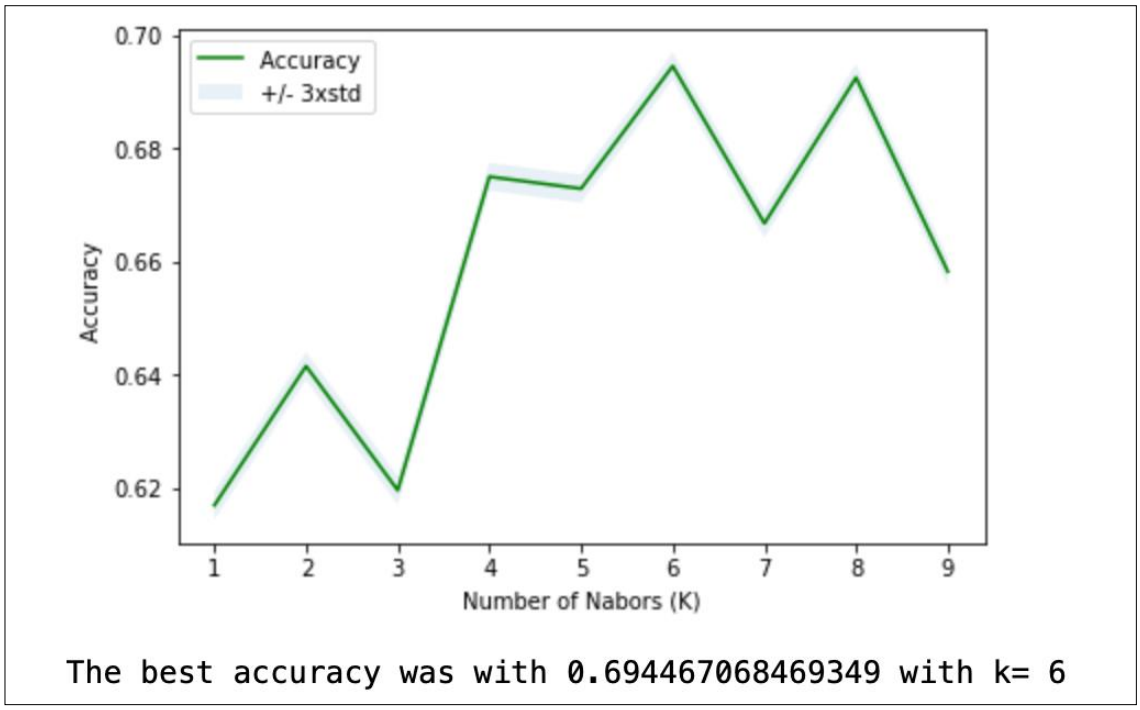

Figure 4: Best k calculated by the accuracy

As for the DT study, the performance and precision of the KNN function, was visualized in a confusion matrix. In addition the F1 score were calculated to evaluate the accuracy of the analysis. In addition a classification report were printed.

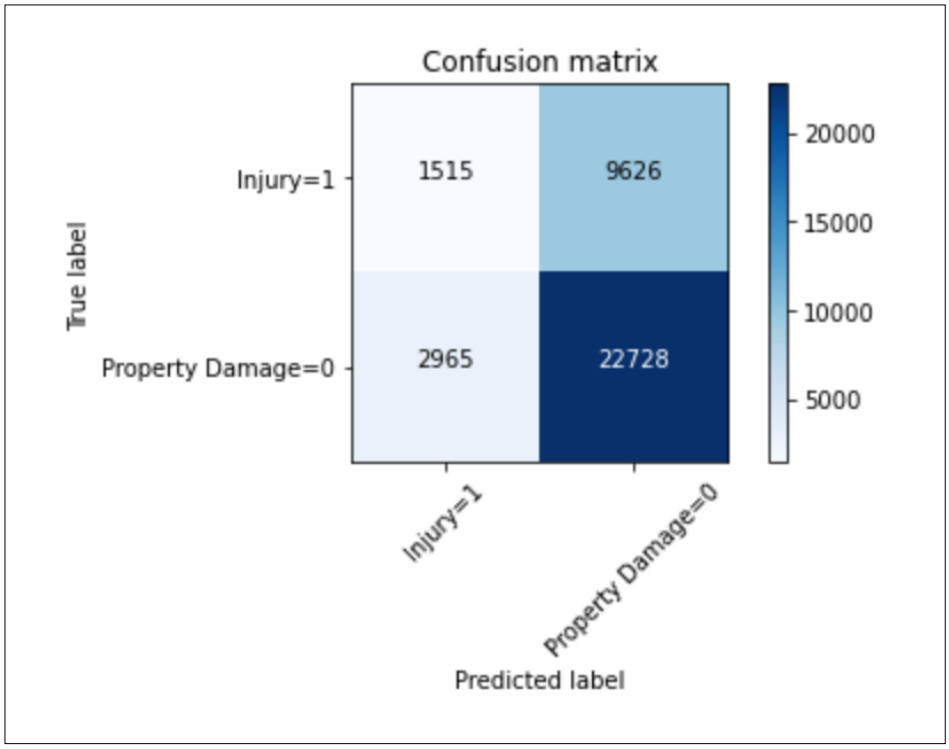

Figure 5: Confusion Matrix KNN

According the confusion matrix the KNN algorithm detects 1515 injury cases which are in real also injury cases. In addition 22728 cases were detected as Property Damage which are also Property Damage cases in reality. 9626 Injury cases were wrongly detected as Property Damage cases and 2965 Property Damage cases were detected wrongly as Injury cases.

Table 3: Classification Report KNN

	Precision	Recall	f1-score
0	0.70	0.88	0.78
1	0.34	0.14	0.19
accuracy			0.66
macro avg	0.52	0.51	0.49
weighted avg	0.59	0.66	0.60

6.3 Logistic Regression Analysis

Finally the Logistic Regression from Scikit-learn package was applied on the datasets. This function implements logistic regression and can use different numerical optimizers to find parameters, including 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga 'solvers. Here the liblinear solver was used and the C value was chosen as 0.01.

As in the studies before, the performance and precision of the Logistic Regression function, was visualized in a confusion matrix and using the F1 score. In addition a classification report were printed.

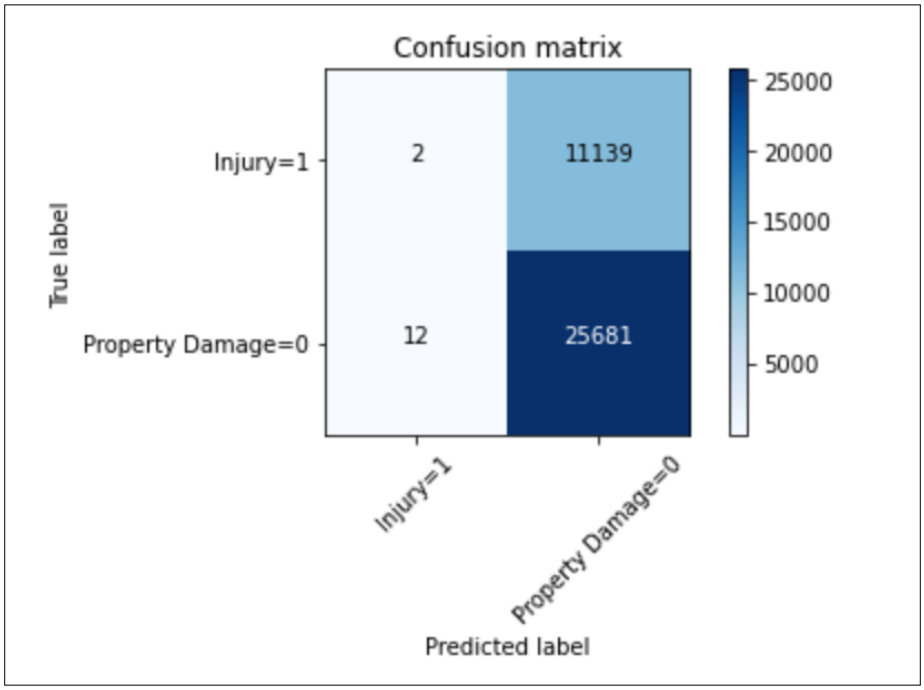

Figure 6: Confusion Matrix Logistic Regression

According the confusion matrix the LR algorithm detects 2 injury cases which are in real also injury cases. In addition 25681 cases were detected as Property Damage which are also Property Damage cases in reality. 11139 Injury cases were wrongly detected as Property Damage cases and 12 Property Damage cases were detected wrongly as Injury cases.

Table 4: Classification Report LR

	Precision	Recall	f1-score
0	0.70	1.00	0.82
1	0.14	0.00	0.00
accuracy			0.70
macro avg	0.42	0.50	0.41
weighted avg	0.53	0.70	0.57

7. DISCUSSION

Table 5: Comparison of different Algorithm

Algorithm	f1-score weighted avg.	Property Damage (0) vs Injury (1)	Precision	Recall
Decision Tree	0.58	1	0.54	0.00
		0	0.70	1.00
KNN	0.60	1	0.34	0.14
		0	0.70	0.88
Logistic Regression	0.57	1	0.14	0.00
		0	0.7	1.00

- Precision is a measure of the accuracy provided that a class label has been predicted. It is defined by: precision = TP / (TP + FP)
- Recall is true positive rate. It is defined as: Recall = TP / (TP + FN)

Precision and recall of each class can be calculated.

The F1 scores for each label based on the precision and recall of that label can be calculated.

The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. It is a good way to show that a classifer has a good value for both recall and precision.

And finally, the average accuracy for this classifier is the average of the F1-score for both labels.

In the Decision Tree as well as in the Logistic Regression algorithm the Recall value for 1 (Injury) is calculated as zero or a very small value. That means that the injury cases could not very good predicted out of the real injury cases. Even for the KNN the Recall for 1 is with 0.14 very low. In contrast, the Recall value of 0 (Property damage) shows very good values. For the Decision Tree the Precision values are very good balanced between 0 and 1. Much better than the Precision value for Logistic Regression and also better than for the KNN.

With a weighted f1 score of 0.6, a good balanced Precision and the best Recall values for 1 (Injury) the KNN seems to be the best alternative ML algorithm for the analyzed data in the prepared condition.

8. CONCLUSION AND RECOMMENDATIONS

With the comparison made in table 5 an assessment on the chosen ML algorithm can be made. According to this comparison, the KNN seems to be the best alternative to use to predict car accidents on given conditions. With a closer look on the confusion matrix of the KNN, it can also be seen, that the predicted values are better balanced than for the other algorithm. According to this approximately 1500 injury cases can be predicted out the given 11100 injury cases and 22700 property damage cases out of 25600 property damage cases can be predicted.

Nevertheless there is potential for improvement. One improvement can be that the entry data should be better balanced. So, there are less than half the amount of injury cases included than cases with property damage. In addition the amount of entries for SPEEDING and INATTENTIONIND are much lower than the other features. The reason for this could be, that there are gaps in the data or they were deleted during the cleaning process. Artificial balancing processes can help here to get better data to feed the ML tool. In addition more features can be chosen to train the algorithm. There might be features which at first sight do not seem to have any influence, but in interaction with the others features can improve the output significantly.

Also, the given data could be analyzed regarding the locations where the accident happens. The amount of accidents took place on special areas in Seatle could be plotted on the map of the city. This can help the city planners to identify dangerous areas and optimize the traffic condition to avoid accidents.