ACCIDENT SEVERITY PREDICTION

Capstone Project for the IBM Data Science Professional Certificate

Christian

Table of Contents

Introduction

The scope of this case study will be the prediction of the severity of a potential car accident, depending on various conditions, such as weather, time of the day, time of the year, road conditions, etc.

Such information might be interesting in several different scenarios. Local authorities could change the traffic flow such that the severity of incidents is kept at a minimum, especially during rush-hour or specific weather events (e.g. heavy rain or fog). Such information can also be included in future infrastructural planning. Similarly, navigation systems and especially self-driving cars could include such information in their routing, in order to ensure safety for their passengers. Lastly, first-response, such as fire-fighters or ambulance can approximate the severity of a reported accident, if more detailed information is not available (e.g. autonomous emergency call from car, but no physical person to ask questions on the phone).

About the Data

The data was collected in Seattle City, Washington, US and can be found under the following link:

Dataset: LINK TO DATASET

Description of Dataset: LINK TO DESCRIPTION

The dataset consists of 37 features, with the "SEVERITYCODE" being the label that is to be predicted. It contains the following values:

0	**Unknown**
1	**Property Damage**
2	**Injury**
2b	**Serious Injury**
3	**fatality**

The following table lists all Columns which will be integrated in the feature set for the prediction models. For further analysis and statistics, please refer to chapter "Data Analysis".

COLUMN NAME	POSSIBLE VALUES	COMMENT
X	Longitude e.g. -122.3231484	Location of accident
Y	Latitude e.g. 47.70314032	Location of accident
INCDTTM	3/27/2013 2:54:00 PM	Incident Date-Time
WEATHER	Overcast, Raining, Clear	Weather conditions
ROADCOND	Wet, Dry, Unknown	Road Conditions
LIGHTCOND	Daylight, Dark- Street lights on, Dark, Street lights off, Dusk, Dawn, unknown	Light conditions
ADDRTYPE	Block, Intersection, Alley,	Address type

Methodology

Data Analysis and Wrangling

As a first step, the data as was investigated with respect to their datatypes, and general information.

```
In [10]: df_col.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 194673 entries, 0 to 194672
Data columns (total 38 columns):
SEVERITYCODE       194673 non-null int64
X                  189339 non-null float64
Y                  189339 non-null float64
OBJECTID           194673 non-null int64
INCKEY             194673 non-null int64
COLDETKEY          194673 non-null int64
REPORTNO           194673 non-null object
STATUS             194673 non-null object
ADDRTYPE           192747 non-null object
INTKEY             65070 non-null float64
LOCATION           191996 non-null object
EXCEPTRSNCODE      84811 non-null object
EXCEPTRSNDESC      5638 non-null object
SEVERITYCODE.1     194673 non-null int64
SEVERITYDESC       194673 non-null object
COLLISIONTYPE      189769 non-null object
PERSONCOUNT        194673 non-null int64
PEDCOUNT           194673 non-null int64
PEDCYLCOUNT        194673 non-null int64
VEHCOUNT           194673 non-null int64
INCDATE            194673 non-null object
INCDTTM            194673 non-null object
JUNCTIONTYPE       188344 non-null object
SDOT_COLCODE       194673 non-null int64
SDOT_COLDESC       194673 non-null object
INATTENTIONIND     29805 non-null object
UNDERINFL          189789 non-null object
WEATHER            189592 non-null object
ROADCOND           189661 non-null object
LIGHTCOND          189503 non-null object
PEDROWNOTGRNT      4667 non-null object
SDOTCOLNUM         114936 non-null float64
SPEEDING           9333 non-null object
ST_COLCODE         194655 non-null object
ST_COLDESC         189769 non-null object
SEGLANEKEY         194673 non-null int64
CROSSWALKKEY       194673 non-null int64
HITPARKEDCAR       194673 non-null object
dtypes: float64(4), int64(12), object(22)
memory usage: 56.4+ MB
```

```
In [12]: df_col.describe()
Out[12]:
```

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	INTKEY	SEVERITYCODE.1	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	SI
count	194673.000000	189339.000000	189339.000000	194673.000000	194673.000000	194673.000000	65070.000000	194673.000000	194673.000000	194673.000000	194673.000000	194673.000000	
mean	1.298901	-122.330518	47.619543	108479.364930	141091.456350	141298.811381	37558.450576	1.298901	2.444427	0.037139	0.028391	1.920780	
std	0.457778	0.029976	0.056157	62649.722558	86634.402737	86986.542110	51745.990273	0.457778	1.345929	0.198150	0.167413	0.631047	
min	1.000000	-122.419091	47.495573	1.000000	1001.000000	1001.000000	23807.000000	1.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	-122.348673	47.575956	54267.000000	70383.000000	70383.000000	28667.000000	1.000000	2.000000	0.000000	0.000000	2.000000	
50%	1.000000	-122.330224	47.615369	106912.000000	123363.000000	123363.000000	29973.000000	1.000000	2.000000	0.000000	0.000000	2.000000	
75%	2.000000	-122.311937	47.663664	162272.000000	203319.000000	203459.000000	33973.000000	2.000000	3.000000	0.000000	0.000000	2.000000	
max	2.000000	-122.238949	47.734142	219547.000000	331454.000000	332954.000000	757580.000000	2.000000	81.000000	6.000000	2.000000	12.000000	

The initial Dataset consists of 194673 entries and 38 categories. However, as mentioned earlier, not all those categories are useful features for machine learning. Moving forward, the **Weather conditions** (WEATHER), **Road conditions** (ROADCOND), **Light conditions** (LIGHTCOND) and **type of Address** (ADDRTYPE) will be used as distinguishers. The date and time could in principle be good features as well, however, scikit-learn cannot use them as is.

Datetime could theoretically be either converted to an integer with "distance from first Datetime timestamp" or with one-hot encoding converted into categorical values. Since we have however information about the lightning condition (which implies at least information about the time in the morning, during the day, evening, or night) the DATETIME row was dropped for the purpose of simplicity (one hot encoding would have 12 categories for month, 31 for day of month, x for years, 24 for hour of day, 60 for minute of hour…)

Similarly, the **X, Y** coordinates might contain value information, but cannot be used as is. One way to solve this problem would be grouping them to certain areas, and then assign those areas via one-hot encoding to binary values. Again, for the sake of simplicity, this has been avoided in this work. Nevertheless, some information about the location is given by the address type information.

Thus, all unnecessary **columns where dropped** as well as all **rows that contained missing values** in one of the remaining columns.

Looking closer at the **severity code**, it turns out, that this example data set only contains two values 1 (property damage) and 2 (injury). For the purpose of a Capstone project, that is totally acceptable, for a real-world scenario however, a different dataset would be better since especially the severe accidents are of specific interest. Furthermore, the data is skewed with respect to the SEVERITYCODE, having almost double the amount of '1' entries compared to '2'. In order to avoid a bias of the learning algorithms, only half of the data with '1' entries were used to obtain a balanced dataset.

Consequently, all values of the remaining features were mapped to binary values by one-hot-encoding, leading to a working dataset as shown below with 111217 entries in 26 categories.

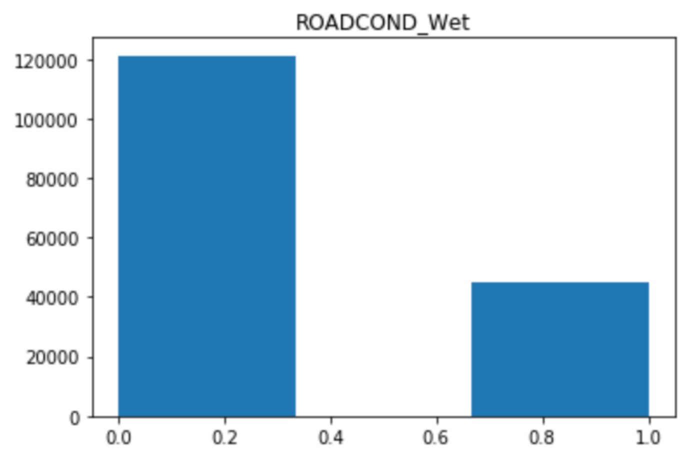
```
In [49]: df_col_bal.shape
Out[49]: (111217, 26)
```

Out[35]:

	SEVERITYCODE	ADDRTYPE_Block	ADDRTYPE_Intersection	WEATHER_Blowing Sand/Dirt	WEATHER_Clear	WEATHER_Fog/Smog/Smoke	WEATHER_Overcast	WI Cl
1	1	1	0	0	0	0	0	0
2	1	1	0	0	0	0	1	0
3	1	1	0	0	1	0	0	0
5	1	0	1	0	1	0	0	0
6	1	0	1	0	0	0	0	0

5 rows × 26 columns

Finally, the distribution of each feature has been observed in the complete data set, as well as separately for the two label values 1 and 2 in order to see any striking correlations. However, none have been found. Below, one histogram of wet road condition in the entire data set is shown exemplary.



The dataset was finally split randomly in train and test data with 70% assigned to the train data set. Note, that normalization is not necessary (and in fact counterproductive), since all values are categorical types.

Machine Learning Algorithms

The following section provides a brief overview over the classification algorithms used in this work.

K-Nearest Neighbour Classifier

The K-nearest neighbour classifier works by comparing a datapoint to its k closest neighbours and classifies it according to the majority class of the neighbours. It's a relatively simple algorithm, but should not be overlooked, since it can often result in sufficiently accurate models at low computing costs. The main hyperparameter that needs to be tuned is the number k of nearest neighbours leading to the highest accuracy of the model on the test set.

Support Vector Machine (SVM)

SVMs are powerful classifiers, albeit typically computationally expensive. They use a mathematical operation, referred to as kernel to map data into a higher dimensional space, such that the data becomes linearly separable. Typical kernels used are the linear kernel, polynomial kernel, as well as the 'RBF' kernel. The model was trained with all four kernels in order to determine the best suited one for the particular problem.

Decision Tree Classifier

A decision tree is often useful, if the decision-making process of the model should be graphically represented in order to analyse it. It's a relatively cheap model computational wise and splits the dataset so that information is gained in every split. This information gain can be measured as 'entropy' or by the 'gini' index, which both have been applied to the dataset.

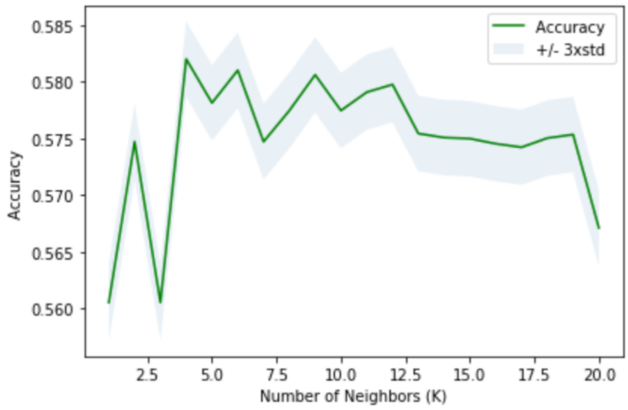
Logistic Regression

Logistic Regression is one of the oldest algorithms for classification, based on the sigmoid or logistic function in order to map the output of the model to a probability of the different labels. It's main hyperparameter to be considered is the strength of regularization, in order to avoid over- and underfitting.

Results

K-Nearest-Neighbour Classifier

The model was trained for k nearest neighbours ranging from 1 to 20, as shown below in the graph, as well as for k=50, k=100 and k=300 in order to assure the best k was found.



Accuracy-score for 50 nearest neighbors is 0.582808847329617

	precision	recall	f1-score	support
1	0.58	0.64	0.61	11230
2	0.59	0.53	0.56	11014
micro avg	0.58	0.58	0.58	22244
macro avg	0.58	0.58	0.58	22244
weighted avg	0.58	0.58	0.58	22244

Accuracy-score for 300 nearest neighbors is 0.5876191332494156

	precision	recall	f1-score	support
1	0.58	0.63	0.61	11230
2	0.59	0.54	0.56	11014
micro avg	0.59	0.59	0.59	22244
macro avg	0.59	0.59	0.59	22244
weighted avg	0.59	0.59	0.59	22244

The best result has been found for k=3 with an accuracy of 0.5876, however, the values do range between 0.56 and 0.58 for all k values and thus are relatively close together. In any case, the K-Nearest neighbour algorithm was not able to make a good prediction of the incident severity.

Support Vector Machine (SVM)

The model was trained with the four kernels 'linear', 'polynomial', and 'rbf' , the results of which are listed below. The accuracy is almost independent of the used kernel, as can be seen below:

Accuracy-score for Kernel "linear" is 0.5934184499190793

	precision	recall	f1-score	support
1	0.58	0.70	0.64	11230
2	0.61	0.48	0.54	11014
micro avg	0.59	0.59	0.59	22244
macro avg	0.60	0.59	0.59	22244
weighted avg	0.60	0.59	0.59	22244

Accuracy-score for Kernel "poly" is 0.5935982736917821

	precision	recall	f1-score	support
1	0.58	0.70	0.64	11230
2	0.61	0.48	0.54	11014
micro avg	0.59	0.59	0.59	22244
macro avg	0.60	0.59	0.59	22244
weighted avg	0.60	0.59	0.59	22244

Accuracy-score for Kernel "rbf" is 0.5934184499190793

	precision	recall	f1-score	support
1	0.58	0.70	0.64	11230
2	0.61	0.48	0.54	11014
micro avg	0.59	0.59	0.59	22244
macro avg	0.60	0.59	0.59	22244
weighted avg	0.60	0.59	0.59	22244

Decision Tree Classifier

The decision tree Classifier was run with 'entropy' and 'gini' as the split criterium. As can be seen below, the 'gini' criterion worked significantly better with an accuracy of 0.9 vs 0.8 for the 'entropy' criterion. Also, as expected, the algorithm was significantly faster processing the data than bot KNN and SVM.

Accuracy-score for criterion "gini" is 0.9

	precision	recall	f1-score	support
1	1.00	0.80	0.89	5
2	0.83	1.00	0.91	5
micro avg	0.90	0.90	0.90	10
macro avg	0.92	0.90	0.90	10
weighted avg	0.92	0.90	0.90	10

Accuracy-score for criterion "entropy" is 0.8

	precision	recall	f1-score	support
1	0.80	0.80	0.80	5
2	0.80	0.80	0.80	5
micro avg	0.80	0.80	0.80	10
macro avg	0.80	0.80	0.80	10
weighted avg	0.80	0.80	0.80	10

Logistic Regression

The logistic regression model was run with 8 different values C for stronger (small C) and less (larger C) regularization. It seems, that with strong regularization the model is underfitting (for C = 0.001, 0.005, 0.01, 0.05, 0.1) and from 0.5 on, it performs better with a accuracy of 0.8. Logistic regression is also one of the computationally less expensive algorithms.

```
Accuracy-score for Regularization Parameter "0.001" is 0.6
              precision    recall  f1-score   support

           1       0.57      0.80      0.67         5
           2       0.67      0.40      0.50         5

   micro avg       0.60      0.60      0.60        10
   macro avg       0.62      0.60      0.58        10
weighted avg       0.62      0.60      0.58        10
```

```
Accuracy-score for Regularization Parameter "0.5" is 0.8
              precision    recall  f1-score   support

           1       0.80      0.80      0.80         5
           2       0.80      0.80      0.80         5

   micro avg       0.80      0.80      0.80        10
   macro avg       0.80      0.80      0.80        10
weighted avg       0.80      0.80      0.80        10
```

Discussion and Conclusion

With the Decision tree classifier with the 'gini' criterion performing best with an accuracy of 0.9, a few things need to be considered.

It is important to try simple, computationally cheap algorithms first, before going to more complex ones, because maybe they provide sufficient accuracy for the given purpose. It should be noted, however, that both SVM and KNN are likely overfitting the training data, thus performing poor on the test data. This could be looked into in greater detail, but for the scope of this work, this should be enough.

Furthermore, in order to improve the performance, the parameters of X, Y and Datetime could be taken into consideration, as already discussed in the Data section. Both might contain important information in order to distinguish between the different severities. Nevertheless, an accuracy of 0.9 is already relatively impressive, given the little effort it takes to do this Capstone Project in comparison with production scale machine learning projects.

Finally, it should be noted, that a larger dataset might also be helpful, as could be specially engineered features. The next step should be to investigate the Decision tree classifier further, using for example 'learning curves' in order to gain more insight in what to spend time on next.