Predicting car accident severity

Introduction

Car accidents are always a big problem in any major city. A car accident can result in:

- Money spent in government property being destroyed;
- People being injured;
- Money spent on hospital bills;
- Lost of human lives.

Introduction

Being able to predict severity of an accident based on some informations about it, before reaching the crash site, can improve how hospitals and police departments react to the situation.

- Hospital can be better prepared to receive the injured people;
- Police and traffic departments can use the same data to reinforce caution in traffic.

Objective

This project's objective is to be able to predict the severity of the accidents based on information that can be obtained before any police officer or ambulance reach the crash site. By doing this, the people responsible in dealing with the situation can be better prepared to do so.

Data acquisition and cleaning

The dataset used for this project is the dataset "Collisions—All Years", provided by the Seattle Department of Transportation - Traffic Management Division. This dataset contains records of all types of collisions that happened since 2004 in the city of Seattle.

This dataset contains 37 attributes, with 194673 rows, and further information about it can be found at:

https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Metadata.pdf

Data acquisition and cleaning

The dataset contains a lot of attributes that aggregate no real value to the classification. Id's and other attributes used only to identificate the accident can be discarded in the analysis.

The attributes that will be used in the analysis are: COLLISIONTYPE (Type of the collision), PERSONCOUNT (The total number of people involved in the collision), VEHCOUNT (The total number of vehicles involved in the collision), JUNCTIONTYPE (Category of junction at which collision took place), 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) and SPEEDING (Whether or not speeding was a factor in the collision).

Data acquisition and cleaning

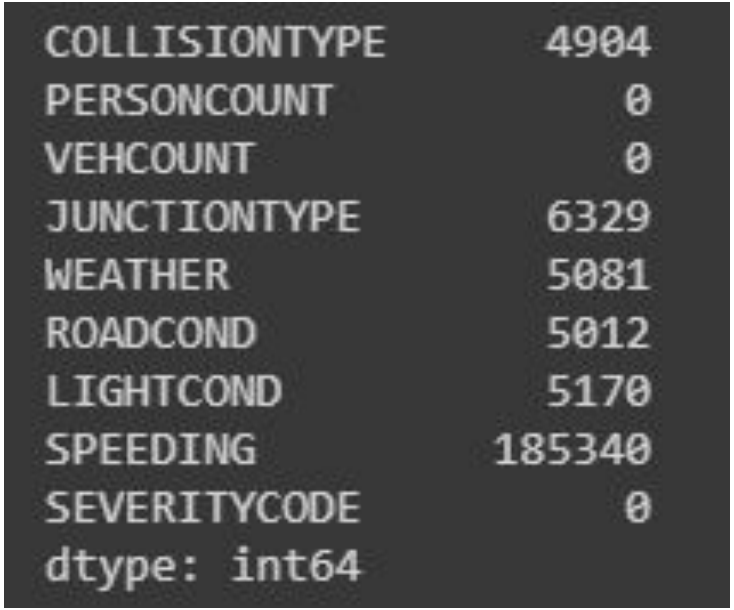
This figure is a subset of the dataset used

	COLLISIONTYPE	PERSONCOUNT	VEHCOUNT	JUNCTIONTYPE	WEATHER	ROADCOND	LIGHTCOND	SPEEDING	SEVERITYCODE
0	Angles	2	2	At Intersection (intersection related)	Overcast	Wet	Daylight	NaN	2
1	Sideswipe	2	2	Mid-Block (not related to intersection)	Raining	Wet	Dark - Street Lights On	NaN	1
2	Parked Car	4	3	Mid-Block (not related to intersection)	Overcast	Dry	Daylight	NaN	1
3	Other	3	3	Mid-Block (not related to intersection)	Clear	Dry	Daylight	NaN	1
4	Angles	2	2	At Intersection (intersection related)	Raining	Wet	Daylight	NaN	2

Data acquisition and cleaning

The next step was finding the amount of null values in each attribute, the figure shows that amount. In the SPEEDING column, all the null values were transformed to “N”, meaning that the driver wasn’t speeding.

The remaining null registers were dropped, because they were irrelevant compared to the total of rows (194673).



Data acquisition and cleaning

The next step was to map all the text registers to numbers, in order to use the algorithms later on. An example of the mapping that was made is shown here (in the COLLISIONTYPE column).

```
# Replacing:
# Parked Car -> 1
df['COLLISIONTYPE'].replace({'Parked Car': 1}, inplace = True)
# Angles -> 2
df['COLLISIONTYPE'].replace({'Angles': 2}, inplace = True)
# Rear Ended -> 3
df['COLLISIONTYPE'].replace({'Rear Ended': 3}, inplace = True)
# Other -> 4
df['COLLISIONTYPE'].replace({'Other': 4}, inplace = True)
# Sideswipe -> 5
df['COLLISIONTYPE'].replace({'Sideswipe': 5}, inplace = True)
# Left Turn -> 6
df['COLLISIONTYPE'].replace({'Left Turn': 6}, inplace = True)
# Pedestrian -> 7
df['COLLISIONTYPE'].replace({'Pedestrian': 7}, inplace = True)
# Cycles -> 8
df['COLLISIONTYPE'].replace({'Cycles': 8}, inplace = True)
# Right Turn -> 9
df['COLLISIONTYPE'].replace({'Right Turn': 9}, inplace = True)
# Head On -> 10
df['COLLISIONTYPE'].replace({'Head On': 10}, inplace = True)
```

Data acquisition and cleaning

Next, we have an subset of the data after all the transformations were made.

	COLLISIONTYPE	PERSONCOUNT	VEHCOUNT	JUNCTIONTYPE	WEATHER	ROADCOND	LIGHTCOND	SPEEDING	SEVERITYCODE
0	2	2	2	2	3	2	1	1	2
1	5	2	2	1	2	2	2	1	1
2	1	4	3	1	3	1	1	1	1
3	4	3	3	1	1	1	1	1	1
4	2	2	2	2	2	2	1	1	2

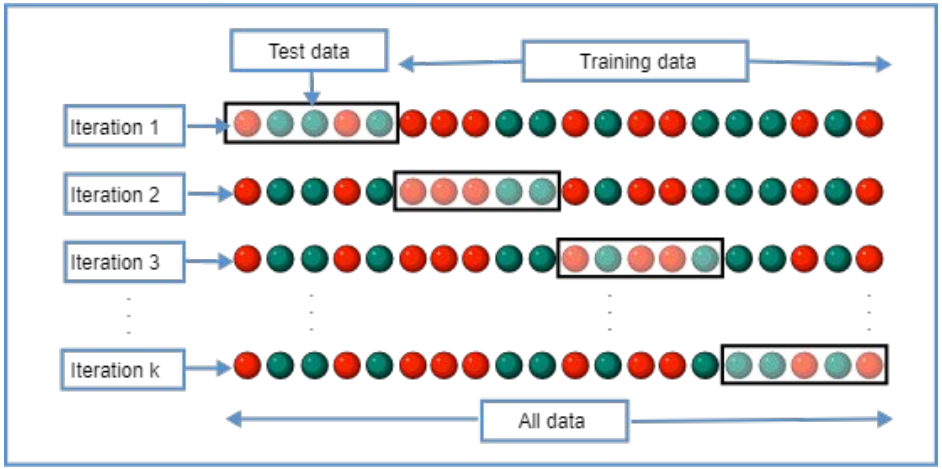
Methodology

The methodology used was to create different models, using different techniques and find the best one amongst them using as parameter the accuracy obtained. The models that were tested were:

- K Nearest Neighbor(KNN)
- Decision Tree Classifier
- Support Vector Machine
- Logistic Regression

Methodology

The dataset was unbalanced (there were more registers belonging to one class). To minimize this impact, the K-Fold Cross Validation technique was used. This technique divides the dataset in train and test sets K times, using a different subset of the data as train and test samples each time.



Modeling and Evaluating

In this section, all the models were created and the training and testing was made using the K-Fold Cross Validation.

The method to calculate the accuracy is:

For each iteration of the K-Fold, the models are trained using the training set given by the K-Fold Cross Validation and are tested using the test set given as well. All the accuracies of each model are put together in a list and after all the iterations, the accuracy of each model is given by the mean of all the accuracies calculated at each iteration.

Results

The results obtained were:

	Accuracy
KNN	0.719686
Decision Tree	0.741387
SVM	0.747703
Logistic Regression	0.717028

Conclusion

In this project, a dataset of accidents in the city of Seattle was used to predict the severity of an accident based on some characteristics of the accident.

- The data was treated to optimize the performance of the models created;
- Four different types of models were tested to see which was better.

Based on the results, its safe to say that the best classifier was the SVM, who achieved the highest accuracy (74,77%).

Next Steps

Further analysis can be made to find out if using the attributes separately, or in different combinations can improve the results obtained.