Applied Data Science Capstone Project

On

Car Accident Severity

Submitted By

Abhishek Vasant Girkar

Under Guidance

Instructor: Alex Aklson

➢ Introduction

Average number of car accidents in the U.S. every year is 6 million. 3 million people in the U.S. are injured every year in car accidents. Analysing the conditions that contribute to these accidents would lead to the prevention of significant loss of life and financial resources.

The project is aimed at predicting the severity of a car accident given the location, weather, and road and visibility conditions in order to reduce the frequency of car collusions in a community based on dataset provided by Seattle PD. Consequently, this analysis would aid drivers to exercise more caution while driving or even choose an alternative route or time for their travel if possible. It could also potentially help the local government, the police department and car insurance providers to gain deeper insight into road accidents

➢ Data:

The data used for this study is given for Applied Data Science Capstone available on coursera.org via the following link
https://s3.us.cloud-object-storage.appdomain.cloud/cfcoursesdata/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv

The dataset has information gathered on the road traffic accidents of Seattle City. Python packages will be used to conduct this study. The dataset will be cleaned according to the requirements of this project. We chose the unbalanced dataset provided by the Seattle Department of Transportation Traffic Management Division with 194673 rows (accidents) and 37 columns (features) where each accident is given a severity code. It covers accidents from January 2004 to May 2020. Some of the features in this dataset include and are not limited to Severity code, Location/Address of accident, Weather condition at the incident site, Driver state (whether under influence or not), collision type. Hence we think it’s a good generalized dataset which will help us in creating an accurate predictive model. The unbalance with respect to the severity code in the dataset is as follows. The initial dataset consists of 38 columns (features/attributes) and 194673 rows. Using pandas drop function we drop all columns expect the desired ones. Columns with the same information but in either categorical or numerical is dropped and only one is chosen.

Detailed dataset of all road collisions (since 2004 to present) can be found here. This data was provided by the Seattle Police Department and recorded by Traffic Records Department. The dataset consists of 37 independent fields and 194673 records, which includes both numerical and categorical data. The dependent field or label for the data set is SEVERITYCODE, which describes the fatality of an accident. The values under this label are categorised into fatality (3), serious injury (2b), injury (2), prop damage (1) and unknown (0).

➢ <u>**Methodology**</u>

Data cleaning is defined as removal of noisy and irrelevant data from collection

- Cleaning in case of Missing values.
- Cleaning noisy data, where noise is a random or variance error.
- Cleaning with Data discrepancy detection and Data transformation tools

Here We Replace NAN Values using Padas Function for each attributes of data Set.

Applied Machine Learning Algorithms on data Set

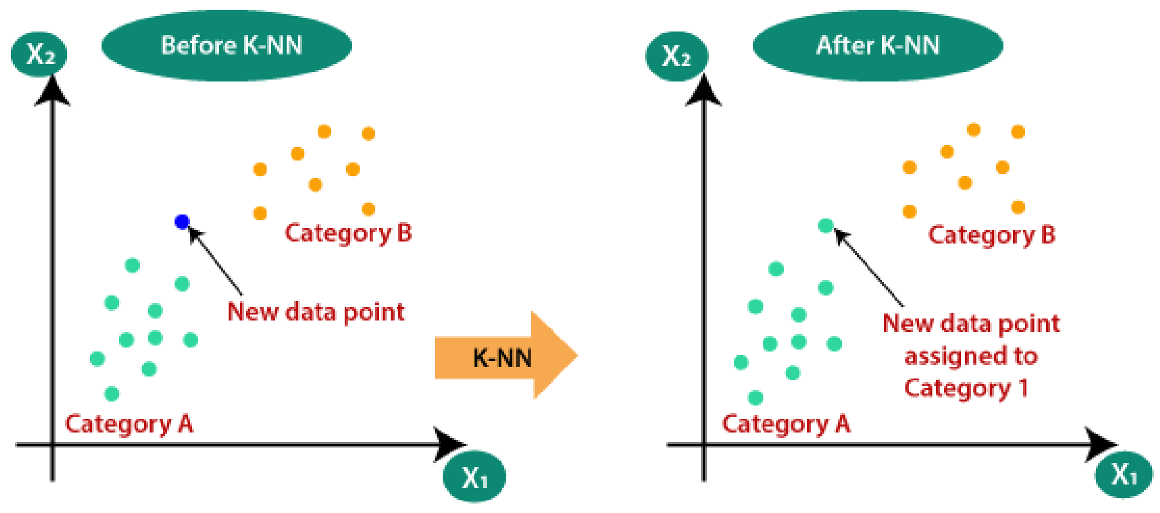
1. <u>KNN Algorithm</u>

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. **K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.** K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

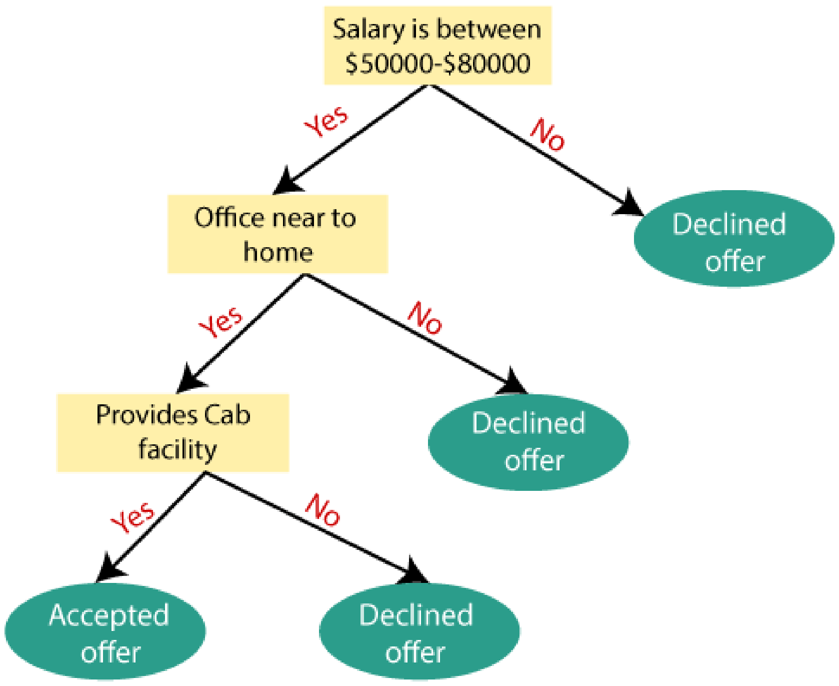


2. Decision Tree

Decision Tree is a supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

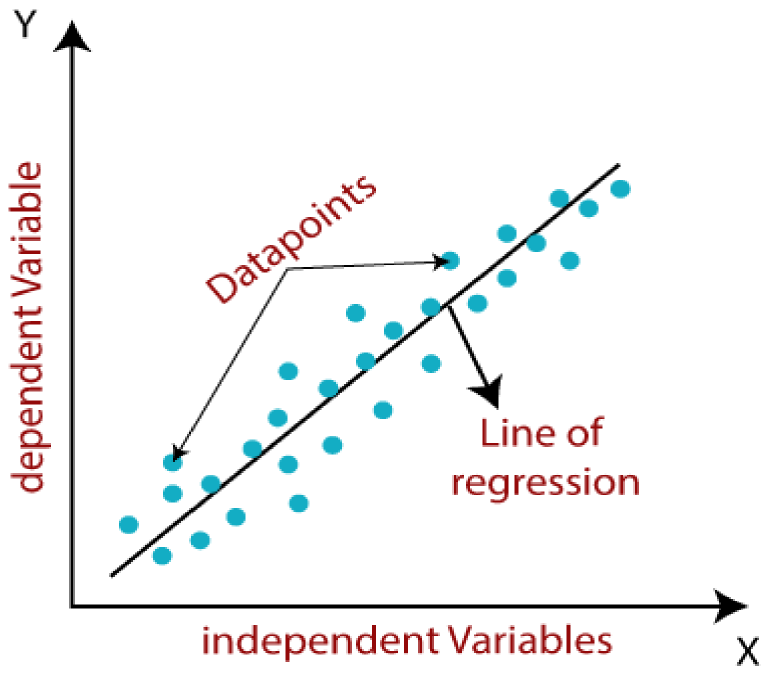
Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



3. **<u>Linear Regression</u>**

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables.



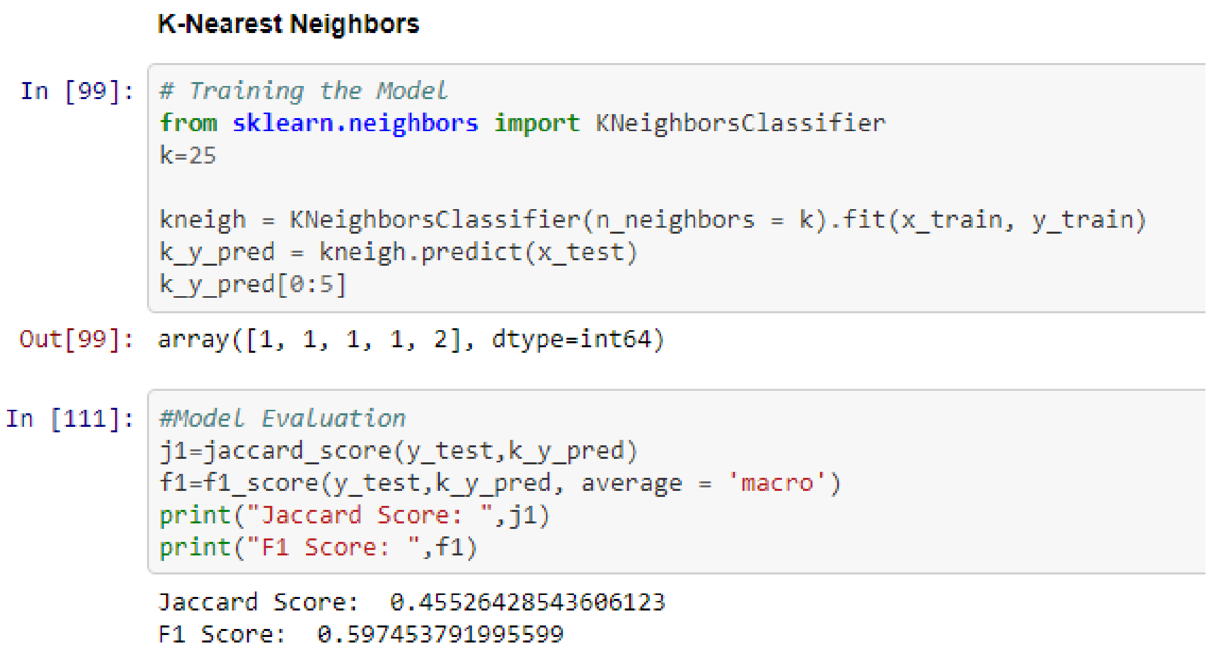
Mathematical Representation of Linear Regression:

$$y = a_0 + a_1 x + \varepsilon$$

Y= Dependent Variable (Target Variable)
X= Independent Variable (predictor Variable)
a0= intercept of the line (Gives an additional degree of freedom)
a1 = Linear regression coefficient (scale factor to each input value).
ε = random error

➢ Code Snippet

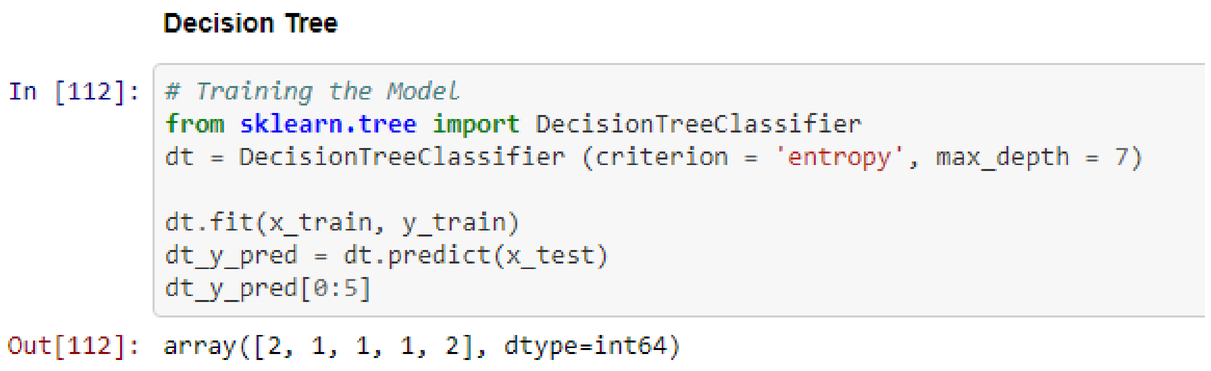
1. KNN Algorithms



K-Nearest Neighbors

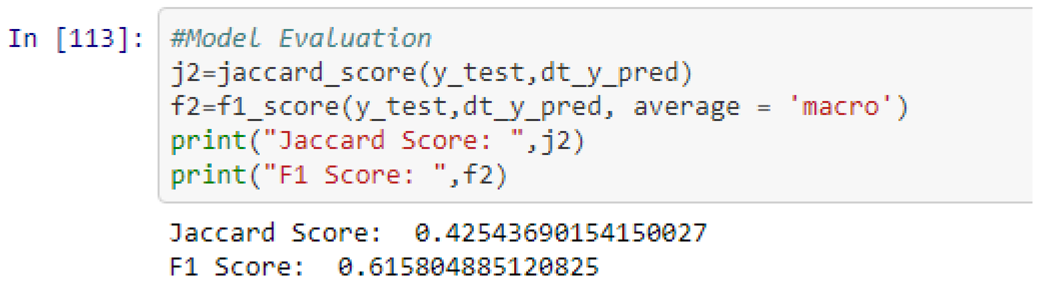
```
In [99]:  # Training the Model
          from sklearn.neighbors import KNeighborsClassifier
          k=25

          kneigh = KNeighborsClassifier(n_neighbors = k).fit(x_train, y_train)
          k_y_pred = kneigh.predict(x_test)
          k_y_pred[0:5]

Out[99]:  array([1, 1, 1, 1, 2], dtype=int64)

In [111]: #Model Evaluation
          j1=jaccard_score(y_test,k_y_pred)
          f1=f1_score(y_test,k_y_pred, average = 'macro')
          print("Jaccard Score: ",j1)
          print("F1 Score: ",f1)

          Jaccard Score:  0.45526428543606123
          F1 Score:  0.597453791995599
```

2. Decision Tree



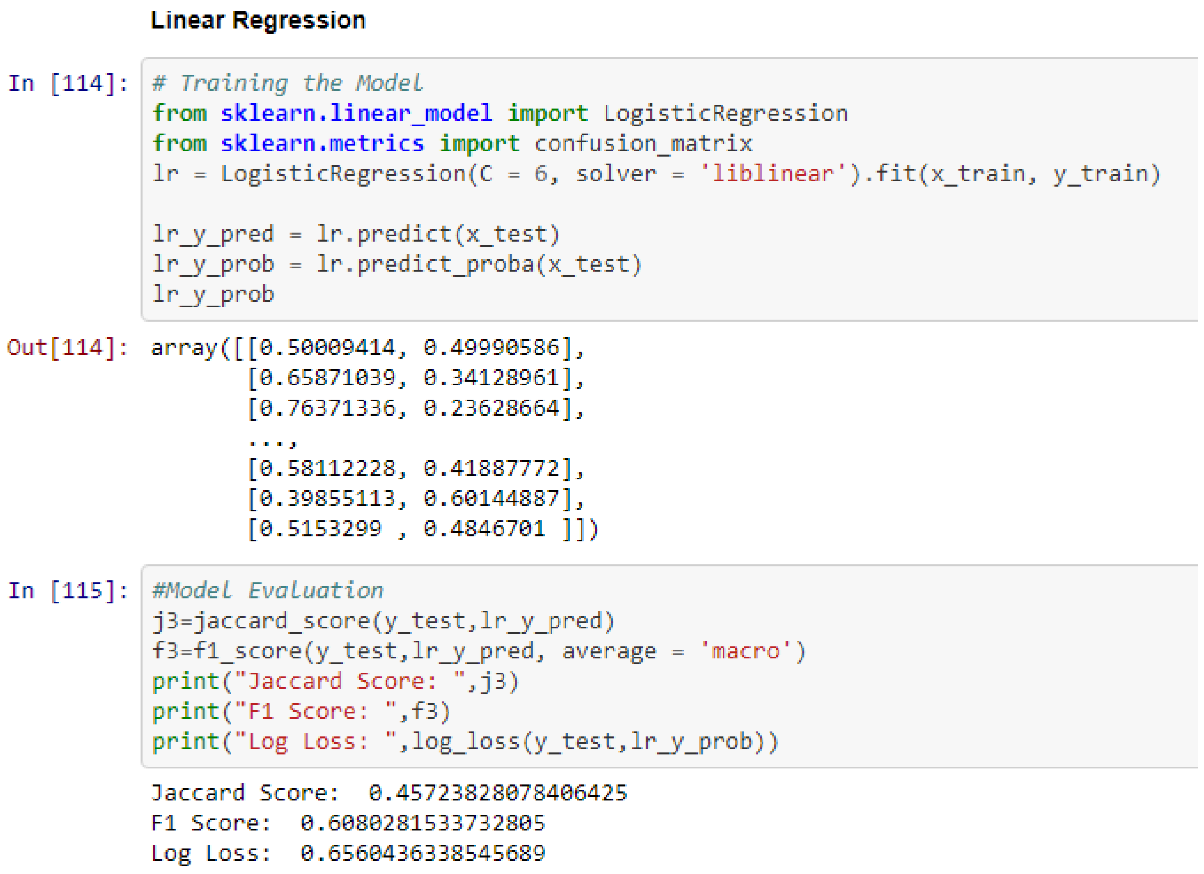
Decision Tree

```
In [112]: # Training the Model
          from sklearn.tree import DecisionTreeClassifier
          dt = DecisionTreeClassifier (criterion = 'entropy', max_depth = 7)

          dt.fit(x_train, y_train)
          dt_y_pred = dt.predict(x_test)
          dt_y_pred[0:5]

Out[112]: array([2, 1, 1, 1, 2], dtype=int64)
```



```
In [113]: #Model Evaluation
          j2=jaccard_score(y_test,dt_y_pred)
          f2=f1_score(y_test,dt_y_pred, average = 'macro')
          print("Jaccard Score: ",j2)
          print("F1 Score: ",f2)

          Jaccard Score:  0.42543690154150027
          F1 Score:  0.615804885120825
```

3. Linear Regression



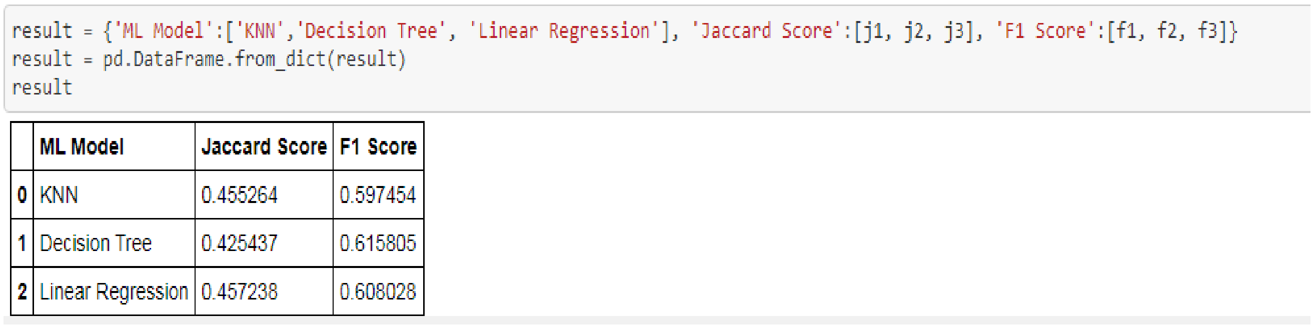
Linear Regression

```
In [114]: # Training the Model
          from sklearn.linear_model import LogisticRegression
          from sklearn.metrics import confusion_matrix
          lr = LogisticRegression(C = 6, solver = 'liblinear').fit(x_train, y_train)

          lr_y_pred = lr.predict(x_test)
          lr_y_prob = lr.predict_proba(x_test)
          lr_y_prob

Out[114]: array([[0.50009414, 0.49990586],
                 [0.65871039, 0.34128961],
                 [0.76371336, 0.23628664],
                 ...,
                 [0.58112228, 0.41887772],
                 [0.39855113, 0.60144887],
                 [0.5153299 , 0.4846701 ]])

In [115]: #Model Evaluation
          j3=jaccard_score(y_test,lr_y_pred)
          f3=f1_score(y_test,lr_y_pred, average = 'macro')
          print("Jaccard Score: ",j3)
          print("F1 Score: ",f3)
          print("Log Loss: ",log_loss(y_test,lr_y_prob))

          Jaccard Score:  0.45723828078406425
          F1 Score:  0.6080281533732805
          Log Loss:  0.6560436338545689
```

➢ **<u>Model Development</u>**

The attributes for the data have categorical variables for their elements and the suitable model type for such data is a classification model. We used KNN, Decision Tree, Linear Regression Algorithms. We have used Jaccard Index and F1-score to evaluate our model. Jaccard Index, it is used to compare set of predicted values to their true values. The higher the score the better the accuracy. From 0 to 1, 1 being the best score. F1-score, it is the weighted average of the precision and recall. The score is between 0 and 1, best score is 1 and worst score at 0.

➢ Result



```
result = {'ML Model':['KNN','Decision Tree', 'Linear Regression'], 'Jaccard Score':[j1, j2, j3], 'F1 Score':[f1, f2, f3]}
result = pd.DataFrame.from_dict(result)
result
```

	ML Model	Jaccard Score	F1 Score
0	KNN	0.455264	0.597454
1	Decision Tree	0.425437	0.615805
2	Linear Regression	0.457238	0.608028

➢ Discussion & Conclusion

This project aimed to study the relationship between severity levels of road traffic collisions in junction types. From our data analysis illustrates that there are different levels of collision count in junction and address types. Such as the difference in severity level in junction types in not so much in all junction expect Mid-block (not related to intersection) junction type. As for address types, at block type we can see the same thing for Property Damage collision only but for Injury collision between Alleys and Block the difference is so much different. To conclude, this project aimed at exploring the data to provide insight in severity levels for road collisions at junctions. **Evaluation metrics used to test the accuracy of our models were jaccard index and f-1 score. Choosing different k, max depth and hyper parameter C values helped to improve the accuracy of our models.** From this Analysis, we notice that we can train our model to determine the severity of an accident to a certain extent using the given dataset.

Based on evaluation of above models, it can be concluded that Linear Regression is the most ideal model for this case. The predictive model would be useful to help local authorities decide on whether to implement new safety measures in certain areas.