Car Accident Severity Analysis: Seattle, Washington

(Applied Data Science Capstone)

The project aims to understand the factors which play a role in the severity of accidents using Machine Learning Models.

SUBMITTED BY : SIDDHARTH M DHELIA

DATE: 10th OCTOBER 2020

GITHUB: https://github.com/siddhelia/Coursera_Capstone

Contents

1. Introduction 2
2. Data 3
3. Methodology 5
4. Analysis ..... ……8
5. Results & Discussion……14
6. Conclusion…….. 16

1.Introduction

Motor vehicle accidents continue to be one of the leading causes of accidental deaths and injuries in the United States. It is estimated more than six million car accidents occur each year in the U.S according to the NHTSA and about 6% of all motor vehicle accidents in the United States result in at least one death.

Roughly 27% of all vehicle accidents result in nonfatal injuries. However, some of these injuries can cause tremendous pain or lead to permanent disabilities.

In this project we will leverage the accident data of "Seattle city" to predict the different accidents' severity.

Our project can be a valuable asset to Governments, states, provinces and municipalities which could use our model to not only prevent road accidents,but also to identify key factors that can lead to a road accident,and consequently, help elaborate new policies.

Goverment agencies can also use this data to warn you, given the weather and the road conditions about the possibility of you getting into a car accident and severity of it.This can enable the driver to drive more carefully or even change his travel plans.

2.Data

Based on definition of our problem, factors that will influence our decission are:

* Number of people and number of vehicles involved in the accident.
* Location,whether,road and light conditions are also infudencial factors.
* Speed of car and junctions are another factors we would consider.

The Following data source will be needed to extract/generate the required information:

https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv

The dataset contains details of car accidents which have taken place within the city of Seattle,Washington in the past fifteen years(2004-2020).The dataset is highly extensive and contains all details of above mentioned factors.

Data Cleaning

There are a lot of problems with the data set keeping in mind that this is a machine learning project which uses classification to predict a categorical variable. The dataset has total observations of 194673 with variation in number of observations for every feature. First of all, the total dataset was high variation in the lengths of almost every column of the dataset. The dataset had a lot of empty columns which could have been beneficial had the data been present there. But since we have have large enough data base,missing entries where removed .Also columns which had no impact in severity of the accidents where omitted from the report and only select few which had most impact on severity of accidents were considered in the study. In order to deal with the issue of columns having a variation in frequency, arrays were made for each column which were encoded according to the original column and had equal proportion of elements as the original column. Then the arrays were imposed on the original columns in the positions which had ‘Other’ and ‘Unknown’ in them. This entire process of cleaning data led to a loss of almost 5000 rows which had redundant data, whereas other rows with unknown values were filled earlier.

Feature Selection

A total of 8 features were selected for this project along with the target variable being Severity Code.

FEATURE VARIABLES	DESCRIPTION
UNDERINFL	Whether or not the driver was under influence. (Y/N)
WEATHER	Weather Condition During Collison. Clear/Rain/Snow
ROADCOND	Road Condition During Collison. Dry/Wet/Snow
LIGHTCOND	Light Condition During Collison. Day/Lights-off/Lights-On
SPEEDING	Whether driver was above speed limit causing accident (Y/N)
JUNCTIONTYPE	Whether Accident occurred at Intersection .(Y/N)
NO. OF PERSONS INVOLVED	Number of people involved in the accident.(1,2,3)
NO. OF VEHICLES INVOLVED	Number of Vechicles involved in the accident.(1,2,3)

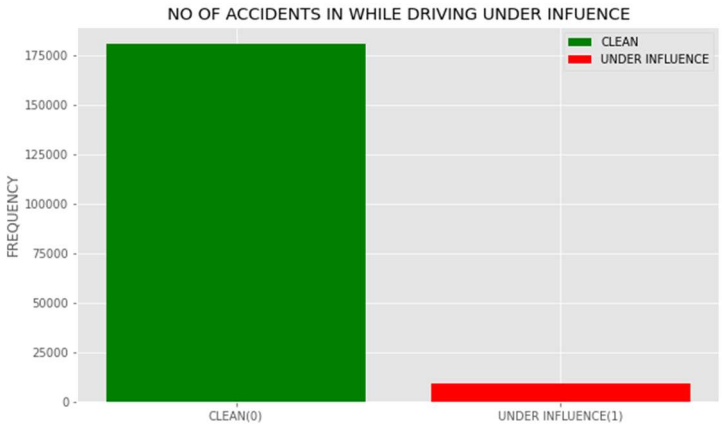
The models aim was to predict the severity of an accident, considering that, the variable of Severity Code was in the form of 1 (Property Damage Only) and 2 (Injury Collision) which were encoded to the form of 0 (Property Damage Only) and 1 (Injury Collision). Furthermore, the Y was given value of 1 whereas N and no value was given 0 for the variables, “Speeding” and “Under the influence”. For lighting condition,Day Light was given 0 ,with Street Light ON as 1 and Street light off as 2. For “Road Condition”, Dry was assigned 0, Wet was assigned 1 and Snow/Ice was given 2. As for “Weather Condition”, 0 is Clear, rain/overcast is 1, Snow/hail is 2. If the accident was caused at intersection the value given was 0 and if not than 1 in “Junction Type”.Also no of persons and vehicles involved where also considered.A visual representation of each feature Variable –“value count” is given in the following page.

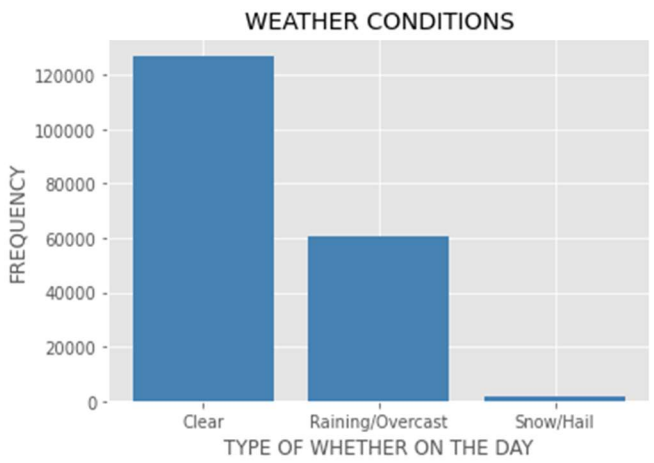
3.Methodology

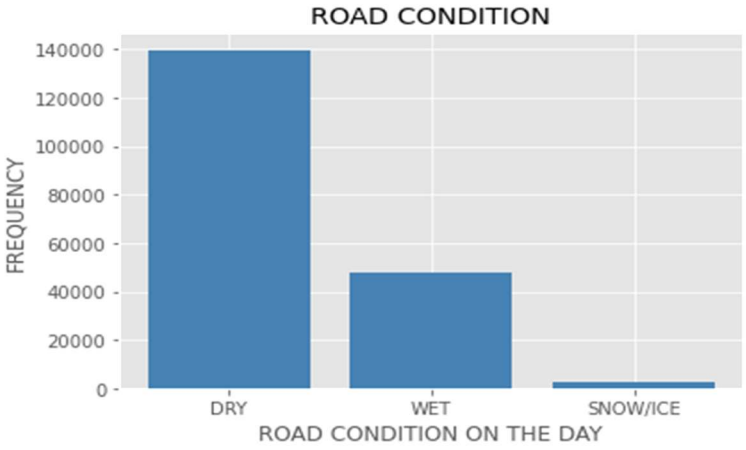
After selecting the most important features to predict severity of accidents in Seattle.Jupyter Notebook was used to preprocess data and build Machine Learning models.Python was used for coding along with its popular packages such as Pandas, Numpy and Matloblip as it gives plethora of data visualization options.

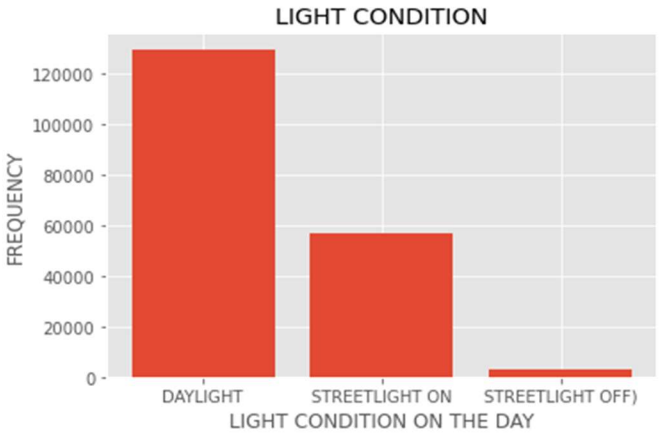
After loading the data into Pandas datafram from the above mentioned link,dtypes command was used to check feature names and their data types.After which value count of each variable was calculated and data chart was plotted for better visual understanding.

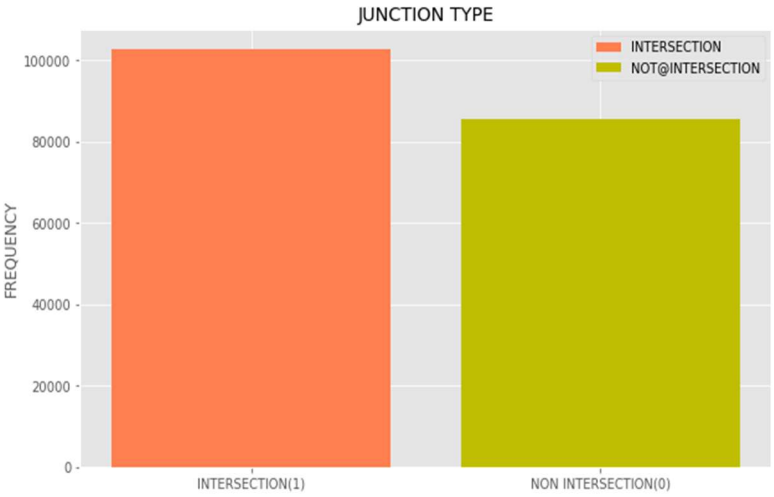
Data Charts For Each Feature Variable is as follows.

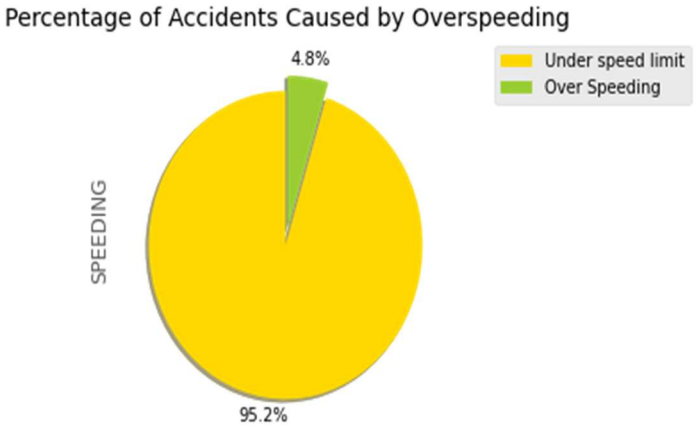


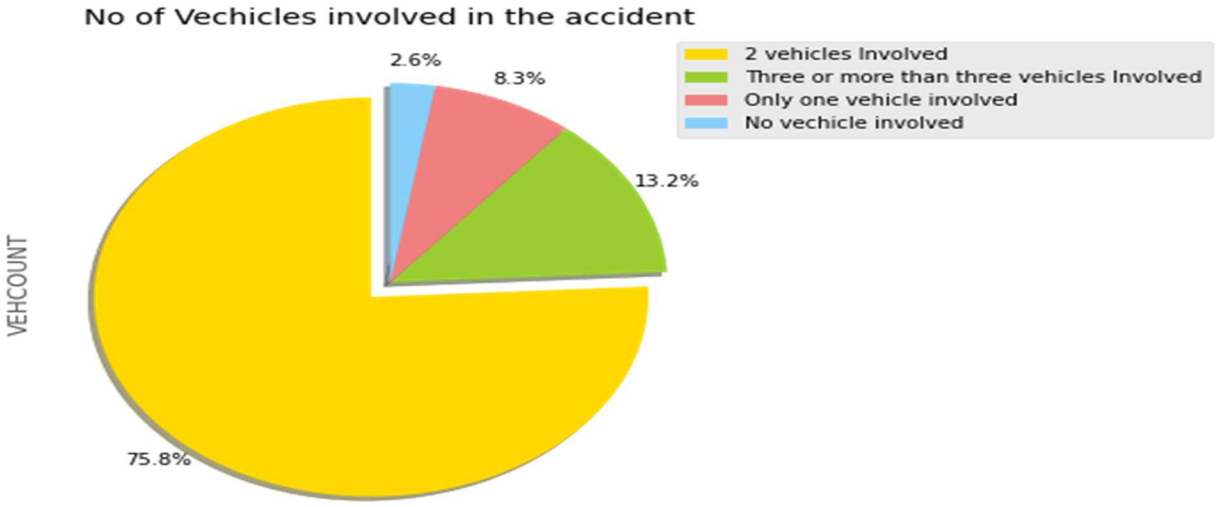


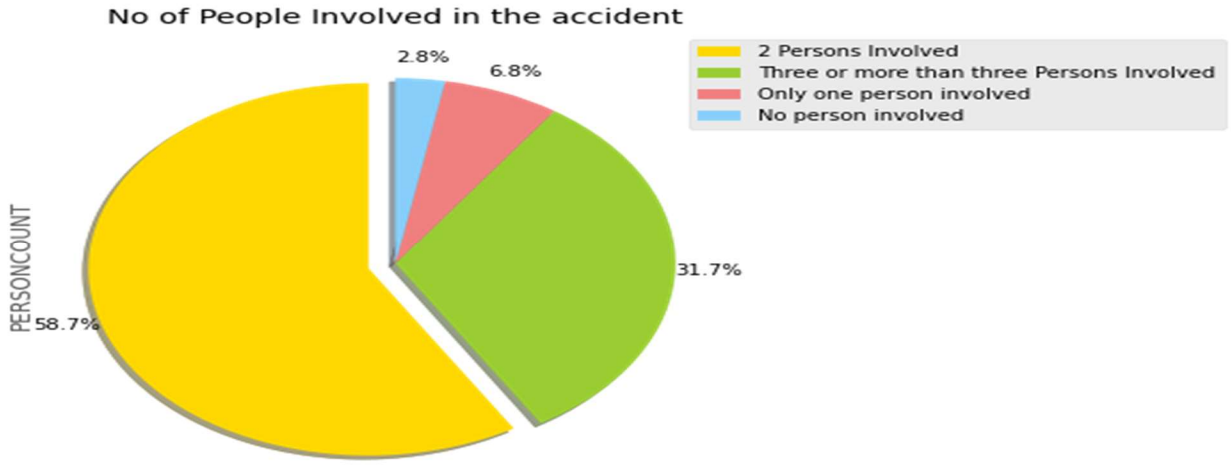




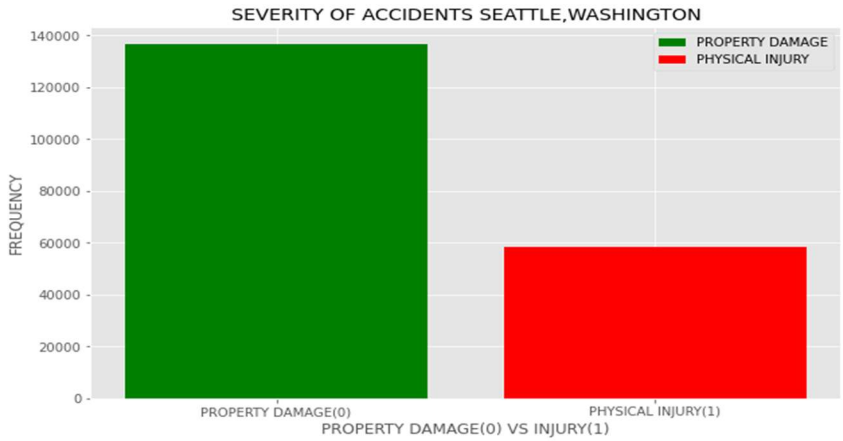

JUNCTION TYPE
INTERSECTION
NOT@INTERSECTION
FREQUENCY
100000
80000
60000
40000
20000
0
INTERSECTION(1)
NON INTERSECTION(0)


Percentage of Accidents Caused by Overspeeding
Under speed limit
Over Speeding
SPEEDING
4.8%
95.2%

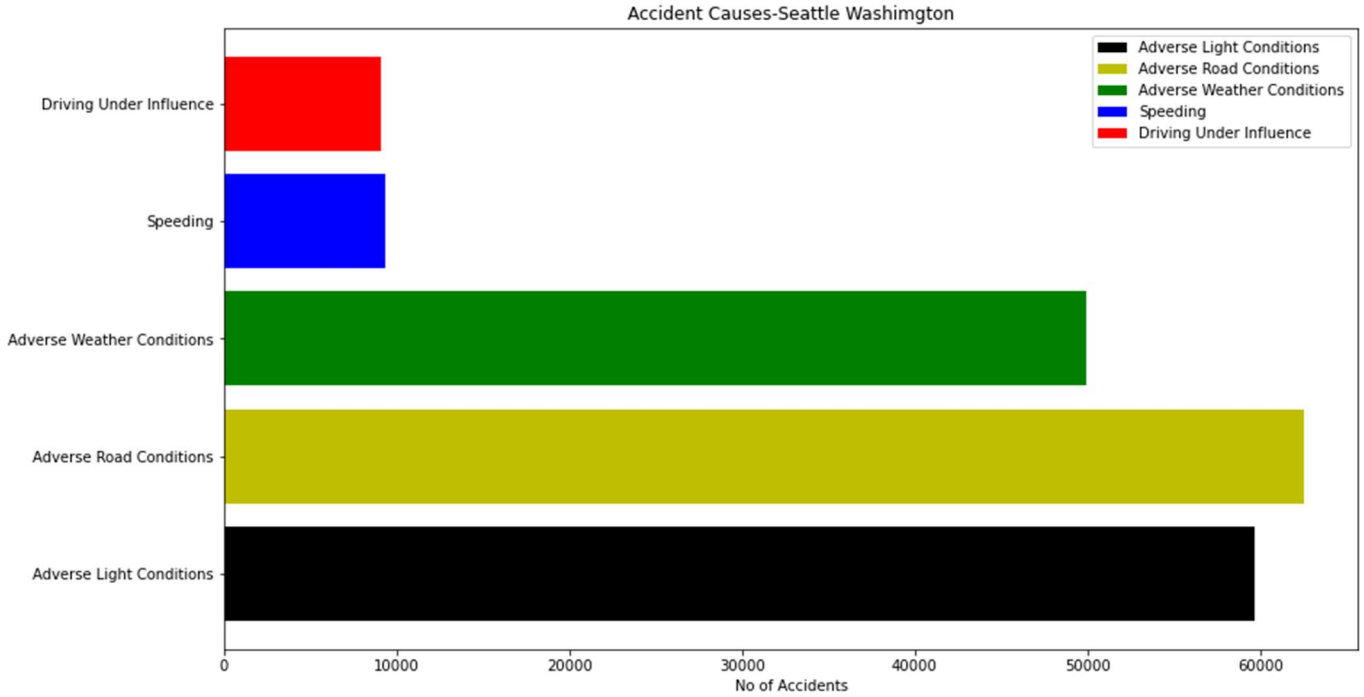

No of Vechicles involved in the accident
2 vehicles Involved
Three or more than three vehicles Involved
Only one vehicle involved
No vechicle involved
VEHCOUNT
2.6%
8.3%
13.2%
75.8%


No of People Involved in the accident
2 Persons Involved
Three or more than three Persons Involved
Only one person involved
No person involved
PERSONCOUNT
2.8%
6.8%
31.7%
58.7%

Also as mentioned "SEVERITYCODE" is the target variable.



A Cummulative Bar Chart for All feature Variables.



After balancing SEVERITYCODE feature, and standardizing the input feature, the data was ready for building machine learning models.Three machine learning were employed:

Decision Tree ,Logistic Regression and KNN model. After importing necessary packages and splitting preprocessed data into test and train sets, for each machine learning model.Models were evaluated and results where compared to find the optimum solution.

4.Analysis

Decision Tree Analysis

Decision Tree Classifier from the scikit-learn library was used to run the Decision Tree Classification model on the Car Accident Severity data. The criterion chosen for the classifier was ‘entropy’ and the max depth was ‘4’.

The Following Lines of Code Where Used

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
X_traindt, X_testdt, y_traindt, y_testdt = train_test_split(X, y, test_size=0.2, random_state=4)
print('Trainset: ',X_traindt.shape,y_traindt.shape)
print('Testset: ',X_testdt.shape,y_testdt.shape)
```

```
Trainset:  (146556, 8) (146556,)
Testset:  (36640, 8) (36640,)
```

```
Tree = DecisionTreeClassifier(criterion = "entropy", max_depth=4).fit(X_traindt,y_traindt)
Tree
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=4,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

```
predTree=Tree.predict(X_testdt)
print("Decision Tree's Accuracy : ",metrics.accuracy_score(y_testdt,predTree))
```

```
Decision Tree's Accuracy :  0.7372816593886463
```

```
from sklearn.metrics import jaccard_similarity_score
jaccard_similarity_score(y_test,predTree)
```

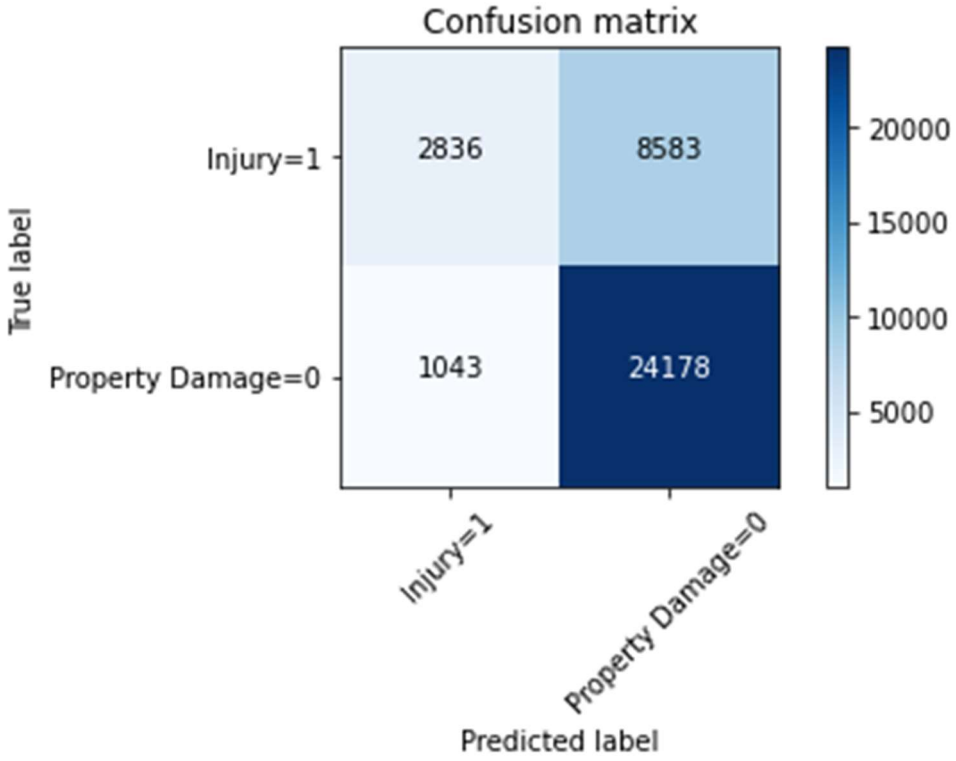
```
0.7372816593886463
```

```
print (classification_report(y_test, predTree))
```

Decision Tree Analysis

The Following Classification Report and Confusion Matrix was obtained from decision tree analysis.

CLASSFICATION REPORT -DECISION TREE				
	PRECISION	**RECALL**	**F-1**	**SUPPORT**
0	0.74	0.96	0.83	25221
1	0.73	0.25	0.37	11419
micro avg	0.74	0.74	0.74	36640
macro avg	0.73	0.6	0.6	36640
weighted avg	0.74	0.74	0.69	36640

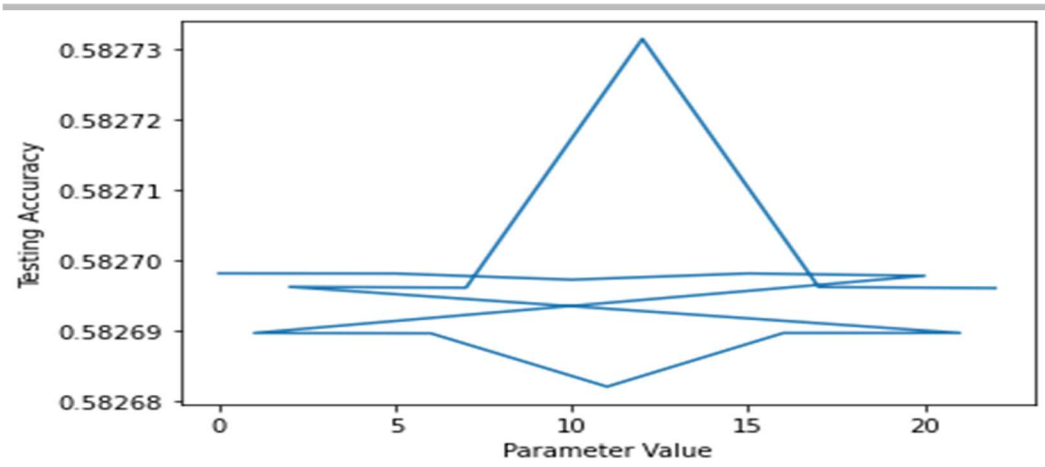


Logistic Regression

Logistic Regression from the scikit-learn library was used to run the Logistic Regression Classification model on the Car Accident Severity data. The C used for regularization strength was '0.01' whereas the solver used was 'liblinear'.

The following lines of code where used .

```
Capstone1-checkpoint.ipyn
Code  git  Submit Notebook ...  Python

]: from sklearn.model_selection import train_test_split
   X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
   print ('Train set:', X_train.shape,  y_train.shape)
   print ('Test set:', X_test.shape,  y_test.shape)

   Train set: (146556, 8) (146556,)
   Test set: (36640, 8) (36640,)

]: from sklearn.linear_model import LogisticRegression
   from sklearn.metrics import log_loss

   import matplotlib.pyplot as plt
   %matplotlib inline

   ...
   Test 1: Accuracy at C = 0.1 when Solver = lbfgs is : 0.5826980984571006
   Test 2: Accuracy at C = 0.1 when Solver = saga is : 0.5826980772678851
   Test 3: Accuracy at C = 0.1 when Solver = liblinear is : 0.5826972002741603
   Test 4: Accuracy at C = 0.1 when Solver = newton-cg is : 0.5826980872867948
   Test 5: Accuracy at C = 0.1 when Solver = sag is : 0.5826977486074866


   Test 6: Accuracy at C = 0.01 when Solver = lbfgs is : 0.5826896350312154
   Test 7: Accuracy at C = 0.01 when Solver = saga is : 0.5826895920823781
   Test 8: Accuracy at C = 0.01 when Solver = liblinear is : 0.5826820201827667
   Test 9: Accuracy at C = 0.01 when Solver = newton-cg is : 0.5826896359009541
   Test 10: Accuracy at C = 0.01 when Solver = sag is : 0.5826896463253558


   Test 11: Accuracy at C = 0.001 when Solver = lbfgs is : 0.5826961850196422
   Test 12: Accuracy at C = 0.001 when Solver = saga is : 0.5826960596682123
   Test 13: Accuracy at C = 0.001 when Solver = liblinear is : 0.582731376284626
   Test 14: Accuracy at C = 0.001 when Solver = newton-cg is : 0.5826961599936722
   Test 15: Accuracy at C = 0.001 when Solver = sag is : 0.5826960063748953
```

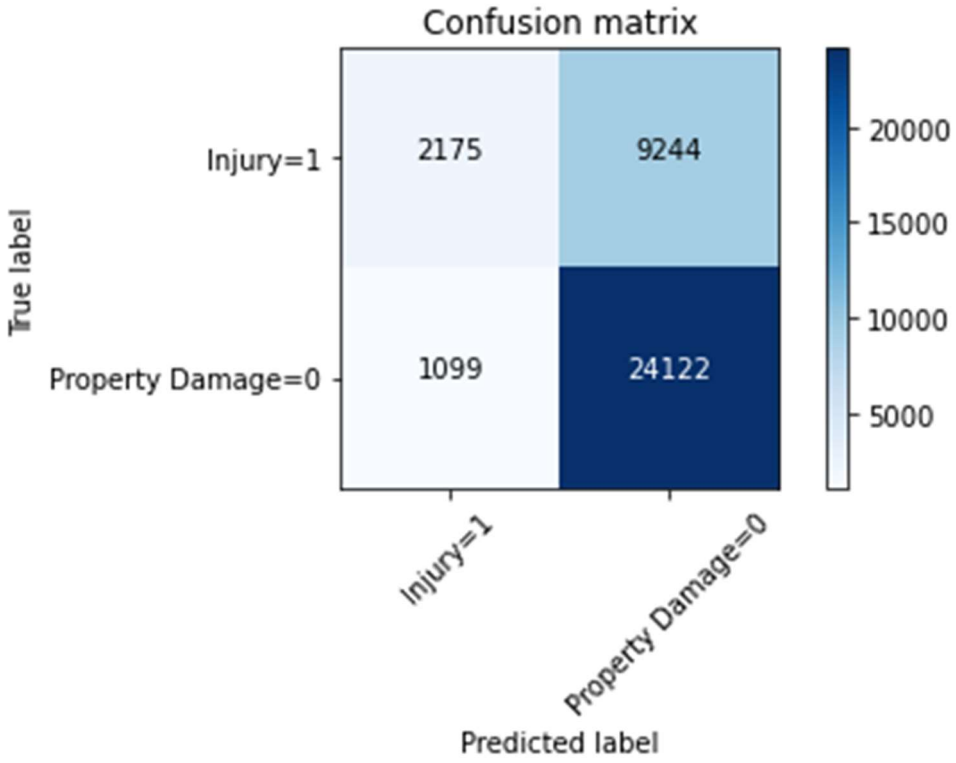



```
[75]: lr_model = LogisticRegression(C = 0.001, solver = 'liblinear')
      lr_model.fit(X_train, y_train)
      lr_model

[75]: LogisticRegression(C=0.001, class_weight=None, dual=False, fit_intercept=True,
                intercept_scaling=1, max_iter=100, multi_class='warn',
                n_jobs=None, penalty='l2', random_state=None, solver='liblinear',
                tol=0.0001, verbose=0, warm_start=False)

[93]: from sklearn.metrics import jaccard_similarity_score

      jaccard_similarity_score(y_test, lr_yhat)

[93]: 0.7177128820960699

[94]: print (classification_report(y_test, lr_yhat))
```

Logistic Regression Analysis

The Following Classification Report and Confusion Matrix was obtained from logistic regression analysis.

CLASSFICATION REPORT -LOGISTIC REGRESSION				
	PRECISION	**RECALL**	**F-1**	**SUPPORT**
0	0.72	0.96	0.82	25221
1	0.66	0.19	0.3	11419
micro avg	0.72	0.72	0.72	36640
macro avg	0.69	0.57	0.56	36640
weighted avg	0.7	0.72	0.66	36640



K-Nearest Neighbor

K-Nearest Neighbor classifier was used from the scikit-learn library to run the k-Nearest Neighbor machine learning classifier on the Car Accident Severity data. The best K, as shown below, for the model where the highest elbow bend exists is at 8.

Best kNN value

Code git Submit Notebook ...

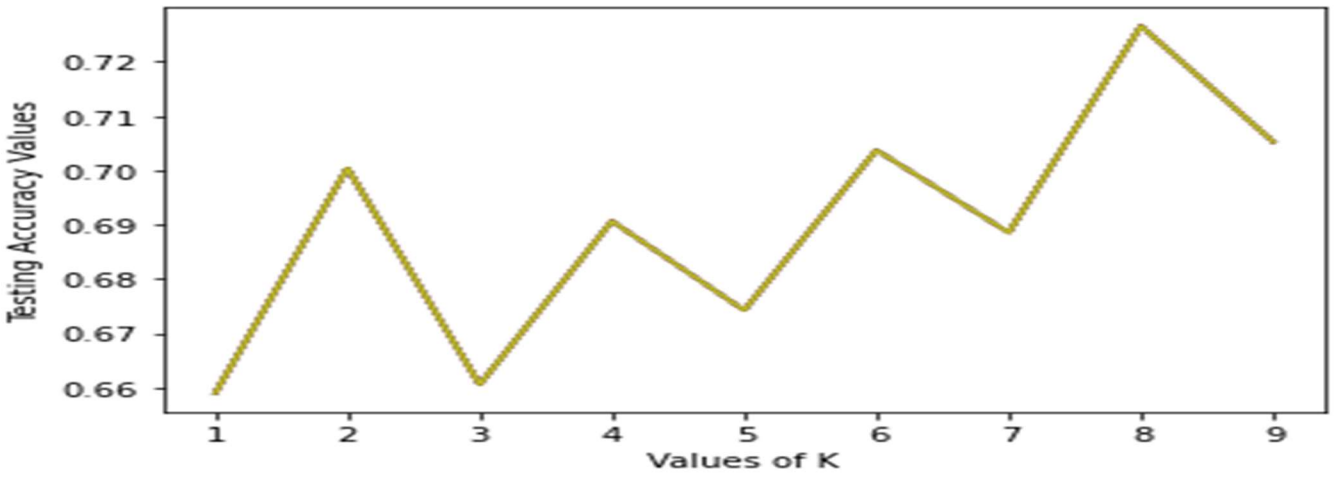
[79]:
```
from sklearn.model_selection import train_test_split
X_trainknn, X_testknn, y_trainknn, y_testknn = train_test_split(X, y, test_size=0.2, random_state=4)
print('Trainset: ',X_trainknn.shape,y_trainknn.shape)
print('Testset: ',X_testknn.shape,y_testknn.shape)
```

```
Trainset:  (146556, 8) (146556,)
Testset:  (36640, 8) (36640,)
```

•••

```
k = 1 has a Score = 0.6589519650655021
k = 2 has a Score = 0.7004093886462882
k = 3 has a Score = 0.6605622270742358
k = 4 has a Score = 0.6906659388646288
k = 5 has a Score = 0.674235807860262
k = 6 has a Score = 0.7037390829694323
k = 7 has a Score = 0.6885371179039301
k = 8 has a Score = 0.7265829694323144
k = 9 has a Score = 0.7052674672489083
```

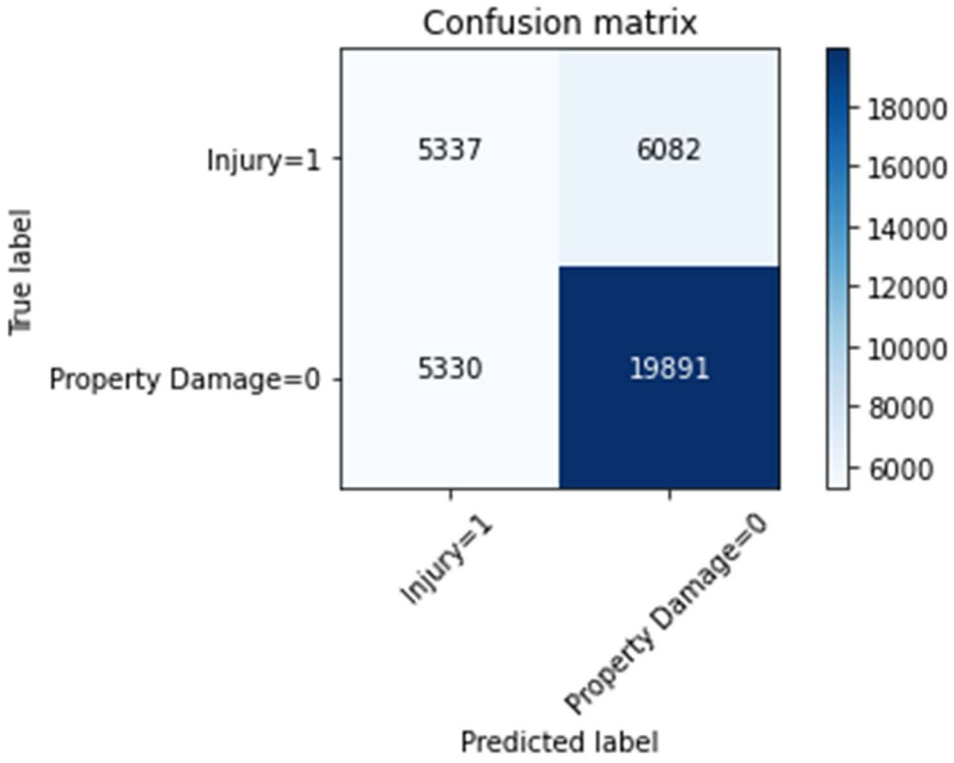
[83]:
```
Text(0, 0.5, 'Testing Accuracy Values')
```



K-Nearest Neighbor

The Following Classification Report and Confusion Matrix was obtained from knn analysis.

CLASSFICATION REPORT -K NEIGHBOURS CLASSIFIER				
	PRECISION	**RECALL**	**F-1**	**SUPPORT**
0	0.77	0.79	0.78	25221
1	0.5	0.47	0.48	11419
micro avg	0.69	0.69	0.69	36640
macro avg	0.63	0.63	0.63	36640
weighted avg	0.68	0.69	0.69	36640



5.Results and Discussion

Algorithm	Average f1-Score	Property Damage (0) vs Injury (1)	Precision	Recall
Decision Tree	0.69	0	0.74	0.96
		1	0.73	0.25
Logistic Regression	0.66	0	0.72	0.96
		1	0.66	0.19
KNN	0.69	0	0.77	0.79
		1	0.5	0.47

Average F1-Score

F1-score is a measure of accuracy of the model, which is the harmonic mean of the model's precision and recall. Perfect precision and recall is shown by the f1-score as 1, which is the highest value for the f1-score, whereas the lowest possible value is 0 which means that either precision or recall is 0. The f1-score shown above is the average of the individual f1-scores of the two elements of the target variable i.e. Property Damage and Injury. When comparing the f1-scores of the three models, we can see that k-Nearest Neighbor and Decision have the same f-1 score at 0.69 with logistic regression just slightly behind at 0.66. However, the average f1-score doesn't depict the true picture of the models accuracy because of the different precision and recall of the model for both the elements of the target variable. Hence, it is biased more towards the precision and recall of Property Damage due to its weightage in the model.

Precision

Precision refers to the percentage of results which are relevant, in simpler terms it can be seen as how many of the selected items from the model are relevant. Mathematically, it is calculated by dividing true positives by true positive and false positive. The highest precision for Property Damage is for KNN , whereas for Injury it is Logistic Regression. The Precision is calculated individually above in order to understand how accurate the model is at predicting Property Damage and Injury individually. For the Decision Tree the precision of 0 is 0.74 and for 1 it is 0.73 which is fairly good. As for the Logistic Regression model, for 0 it is at 0.72 and for 1 it is 0.66. Lastly, for the k-Nearest Neighbor at 0 it is 0.77, ,however for 1 it is only 0.5, which is low. In terms of precision, the best performing model is the decision tree.

Recall

Recall refers to the percentage of total relevant results correctly classified by the algorithm. In simpler terms, it tells how many relevant items were selected. It is calculated by dividing true positives by true positive and false negative. The highest recall for 0 is when using the The Decision tree model at 0.96 as for 1 it is the KNN at 0.47. The recall for both Property Damage and Injury is almost identical for the Decision Tree and Logistic Regression. As for the KNN , the recall for Property Damage is 0.79 and for Injury it is 0.47. The recall of knn is more balanced more in terms of being good for both the outputs of the target variable.

CONCLUSION

When comparing all the models by their f1-scores, Precision and Recall, we can have a clearer picture in terms of the accuracy of the three models individually as a whole and how well they perform for each output of the target variable. When comparing these scores, we can see that the f1-score is similar for all three models,0.69 for both decision tree and Knn with logistic regression just behind at 0.66.

However, later when we compare the precision for each of the model, we can see that the knn-model performs poorly in the precision of 1 at only 0.5,although the precision is highest of 0 at 0.77. While the decision tree gives good precison for both variables 0 and 1 at 0.73 and 0.74 respectively.In case of logistic regression,it is slightly behind decision tree for both variables at 0.72 and 0.66.So in terms of precision we have a clear winner Decision Tree .In terms of recall decision tree and logistic regression both give super high recall for variable 0 at 0.96 ,but perform poorly at recall of variable 1.While KNN gives more balanced recall .

So after comparing all the models in terms of f1-scores,precion and recall .We have a clear winner -Decision Tree as it has the highest F1 score, good precison rate for both variables and excellent recall of variable 0 the only draw back being low recall rate for variable 1.But inspite of the draw **“Decision Tree”** model gives optimum results and should be considered over other two.

THE END OF THE REPORT