COURSERA - CAPSTONE PROJECT

Car Accident Severity

HTAY AUNG

September 27, 2020

Contents

1. Introduction

1.1 Business problem

To prevent car accidents by analyzing the previous cases that alerts drivers, and traffic police to remind them to careful in critical situations.

1.2 Target audience

The target audiences for this project are Seattle government, police, recuse groups and drivers.

2. Data

2.1 Data sources

The data were collected by the Seattle Police department and Accident Traffic Records Department from 2004 to present. The data consists of 37 independent variables and 1 dependent variable.

```
In [9]: print('Number of columns: ', len(df.columns))
            Number of columns:  38
```

There are 194673 records.

```
print('Number of rows: ',len(df))
    Number of rows:  194673
```

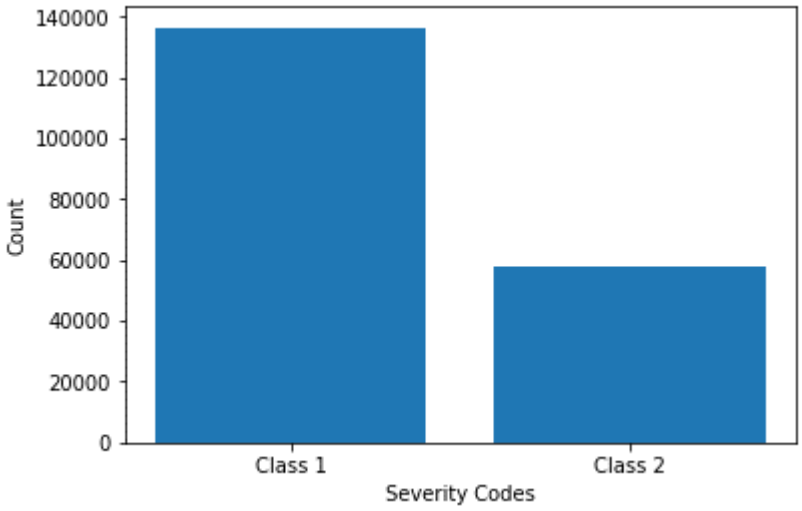
There are 5 SEVERITYCODE that correspond to different levels of severity.

Severity codes are as follows:

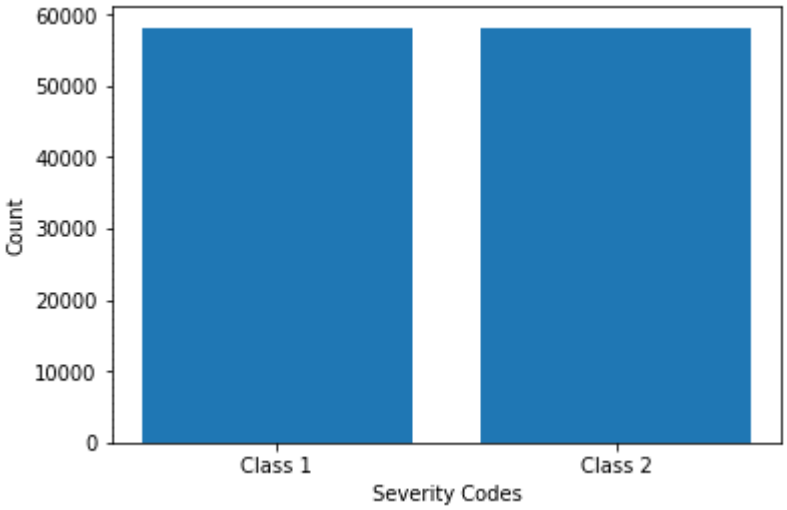
- 3 – fatality
- 2b – serious injury
- 2 – injury
- 1 – prop damage
- 0 – unknown

2.2 Data Processing

The original dataset is not ready for data analysis. Before analyzing data, I have to clean data. Firstly, I have to drop non-relevant columns. And I have decided to use independent variables (WEATHER, ROADCOND, LIGHTCOND). The data from dataset is imbalance, so I used simple technique to balance it.



This is original data



This is balanced data

Now, data is balance. So, we can perform data analysis task.

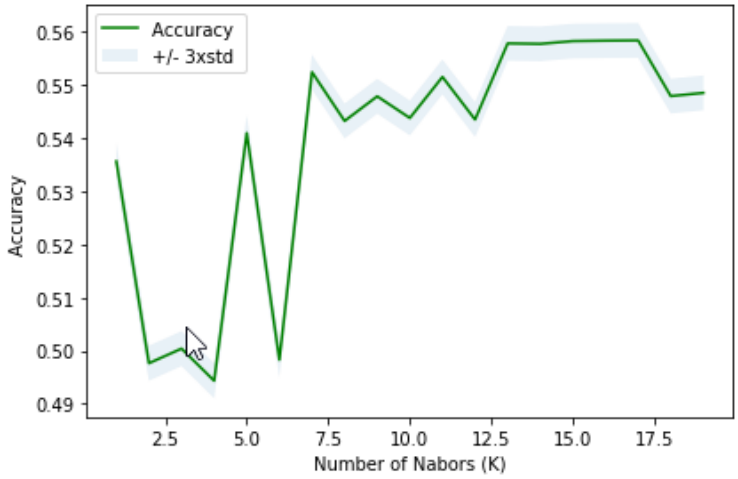
3. Methodology

3.1 Calculation of target variable

I have employed three Machine Learning models:

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

3.1.1 KNN




```
print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

```
The best accuracy was with 0.5584292833820244 with k= 17
```

```
k = 17
#Train Model and Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
# neigh
knn_yhat = neigh.predict(X_test)
print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, knn_yhat))
```

```
Train set Accuracy:  0.5548549946294307
Test set Accuracy:  0.5584292833820244
```

K nearest neighbor - Evaluation

```
jaccard_similarity_score(y_test, knn_yhat)
```

```
3]: 0.5584292833820244
```

```
f1_score(y_test, knn_yhat, average='macro')
```

```
4]: 0.5292817759393895
```

3.1.2 Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
dt.fit(X_train,y_train)
```

```
6]: 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')
```

```
ds_yhat = dt.predict(X_test)
```

```
jaccard_similarity_score(y_test, ds_yhat)
```

```
8]: 0.5629403677607836
```

```
f1_score(y_test, ds_yhat, average='macro')
```

```
9]: 0.48503950717245164
```

```
print("Test set Accuracy: ", metrics.accuracy_score(y_test, ds_yhat))
```

```
Test set Accuracy:  0.5629403677607836
```

3.1.3 Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
```

```
lr_yhat = LR.predict(X_test)
lr_yhat_prob = LR.predict_proba(X_test)
```

```
jaccard_similarity_score(y_test, lr_yhat)
```

```
2]: 0.5537893108781577
```

```
f1_score(y_test, lr_yhat, average='macro')
```

```
3]: 0.5363513619725261
```

```
log_loss(y_test, lr_yhat_prob)
```

```
4]: 0.6766835731232653
```

```
print("Test set Accuracy: ", metrics.accuracy_score(y_test, lr_yhat))
```

```
Test set Accuracy:  0.5537893108781577
```

4. Results

Algorithm	Jaccard	F1-score	LogLoss	Accuracy
KNN	0.56	0.53	NA	0.56
Decision Tree	0.56	0.48	NA	0.56
LogisticRegression	0.55	0.53	0.68	0.55

Based on the above table, Decision Tree is the best model to predict accident severity.

5. Conclusion

Based on the dataset, certain classes are pointing to weather, road and light conditions. We can decide to travel or not if we have these conditions. This concluded my report.

Thanks.