Accident Severity Prediction to reduce Road Traffic Accidents in Seattle,United States Capstone Project

Kalyan Pathakamuri

Contents

Introduction

- Road traffic accidents has brought sadness and trauma to the loved ones. A study to learn the major/Deciding factors for the cause of traffic accidents is an important area of research. In a effort to decrease the count and frequency of such fatal collisions in the neighbourhood and community, a model/method should be developed to analyze the data by considering the condtions -(Current Weather, road and visibility conditions).With the model, we will be able to alert the user to be more cautious.
- Objective in this project is to develop a supervised prediction model that predicts the severity of an accident given certain circumstances (the current weather, road and visibility conditions).

Data

- The dataset that we will be using is a .csv file named, 'Data-Collisions'. Our target variable is 'SEVERITYCODE' as it helps measure the severity of an accident from 1 to 2. Attributes to weigh the severity of an accident are 'WEATHER', 'ROADCOND' and 'LIGHTCOND'.
- The Original dataset have many unwanted variables and values to make the dataset fit for analysis, Dataset has to be preprocessed and cleansed. Post processing the data we will used to train the machine learning model.

Methodology

- Exploratory Data Analysis The correlation Heat-Map of the dataset was explored. However, it did not provide much of an insight to the problem as our independent variables were shown to be Negatively correlated with the dependent variable. After that, the Pearson Coefficient and p-value were explored, which showed that the Road Condition and Light Condition had a strong relation with the Collision Severity. The initial decision of including the Weather Condition along with the Road and Light Condition was not changed.
- Machine Learning Algorithms & Evaluation
- K-Nearest Neighbor (KNN)* KNN will help us predict the severity code of an outcome by finding the most similar to data point within k distance.

Methodology

- Decision Tree A decision tree model gives us a layout of all possible outcomes so we can fully analyze the concequences of a decision. In this context, the decision tree observes all possible outcomes of different weather conditions.
- Logistic Regression Because our dataset only provides us with two severity code outcomes, our model will only predict one of those two classes. This makes our data binary, which is perfect to use with logistic regression.

KNN

```
Ks=15
mean_acc=np.zeros((Ks-1))
std_acc=np.zeros((Ks-1))
ConfustionMx=[];
for n in range(1,Ks):
 kNN_model= KNN(n_neighbors=n).fit(X_train,y_train)
yhat = kNN_model.predict(X_test)
mean_acc[n-1]=np.mean(yhat==y_test);
std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])
mean_acc
```

```
array([0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.51854989])
```

```
k = 5
kNN_model = KNN(n_neighbors=k).fit(X_train,y_train)
kNN_model
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

```
print("KNN Jaccard index: %.2f" % jaccard_similarity_score(y_test, yhat))
print("KNN F1-score: %.2f" % f1_score(y_test, yhat, average='weighted') )
```

```
KNN Jaccard index: 0.52
KNN F1-score: 0.51
```

DECISION TREE

Decision Tree

```
DT_model = DTC(criterion="entropy", max_depth = 4)
DT_model.fit(X_train,y_train)
DT_model
```

```
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')
```

```
yhat1 = DT_model.predict(X_test)
yhat1
```

```
array([2., 2., 2., ..., 2., 2., 1.])
```

Decision Tree

```
print("DT Jaccard index: %.2f" % jaccard_similarity_score(y_test, yhat1))
print("DT F1-score: %.2f" % f1_score(y_test, yhat1, average='weighted') )
```

```
DT Jaccard index: 0.55
DT F1-score: 0.46
```

Regression

Regression

```
LR_model = LR(C=0.01,solver='liblinear').fit(X_train,y_train)
LR_model
```

```
LogisticRegression(C=0.01, 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)
```

```
yhat2 = LR_model.predict(X_test)
yhat2
```

```
array([2., 2., 2., ..., 2., 2., 1.])
```

```
yhat_prob = LR_model.predict_proba(X_test)
print("LR Jaccard index: %.2f" % jaccard_similarity_score(y_test, yhat2))
print("LR F1-score: %.2f" % f1_score(y_test, yhat2, average='weighted') )
print("LR LogLoss: %.2f" % log_loss(y_test, yhat_prob))
```

```
LR Jaccard index: 0.54
LR F1-score: 0.52
LR LogLoss: 0.68
```

Cumulative Result

Algorithm	Jaccard	F1-score	Logloss
KNN	0.52	0.52	0.94
Decision Tree	0.55	0.46	0.67
Logistic Regression	0.54	0.52	0.68

Conclusion

- With the model build it can predict with 55% accuracy
- To overall conclude we will be able to provide a feasible solution with the real time parameters.