Coursera_Capstone : Car Accident Severity(Seattle)

Hossain Mohammad Amran

Intro

Road accidents are an unstoppable problem in our life and societies. Death on the road is a big puzzle in the current world.

- ➔ According to the World Health Organization (WHO), an estimated 1.35 million deaths worldwide were related to road traffic injuries in 2016 [1]
- ➔ Every 25 seconds, one person died in road accidents.
- ➔ In 2016, 34,436 motor vehicle crashes were recorded in the USA and 37,461 people died with an average of 102 per day [2]
- ➔ Develop a machine learning model to predict the severity of accidents

The local government want to implement some method to alert the car user, police , traffic system and health system about critical situations to reduce the death and injuries on the road.

we want to analyze the accident “severity” in terms of human fatality, traffic delay, property damage, or any other type of accident bad impact.

- The data was collected by Seattle SPOT Traffic Management Division. This dataset is updated weekly. It contains information such as severity code, address type, location, collision type, weather, road condition, speeding, among others.

- The target audience is local Seattle government, police, rescue groups, and last but not least, car insurance institutes.

The model going to provide some advice for the target audience to make insightful decisions for reducing the number of accidents and injuries for the city.

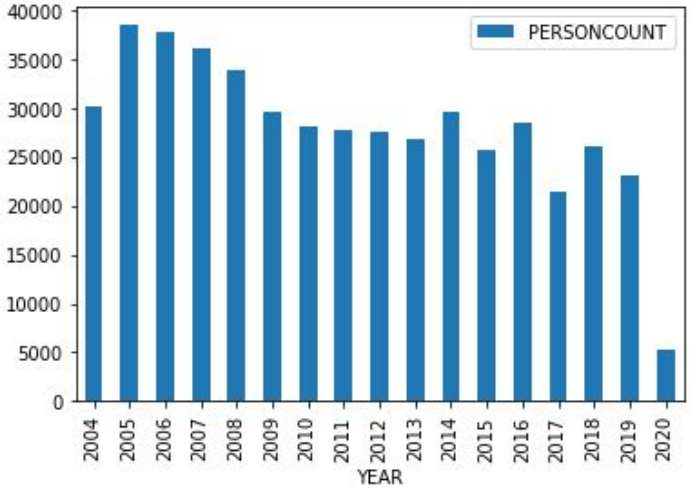
Data Understanding

The dataset has 194,673 rows and 37 different independent variables.

➔ **SEVERITY CODE as dependent variable Y**

➔ **independent variables X to identify the cause of road accidents and level of severity**

Data Preparation.



- We have to prepare the data set to build a perfect model. We have to drop some non-relevant variable and drop some null values as missing data, which will not be useful to model.

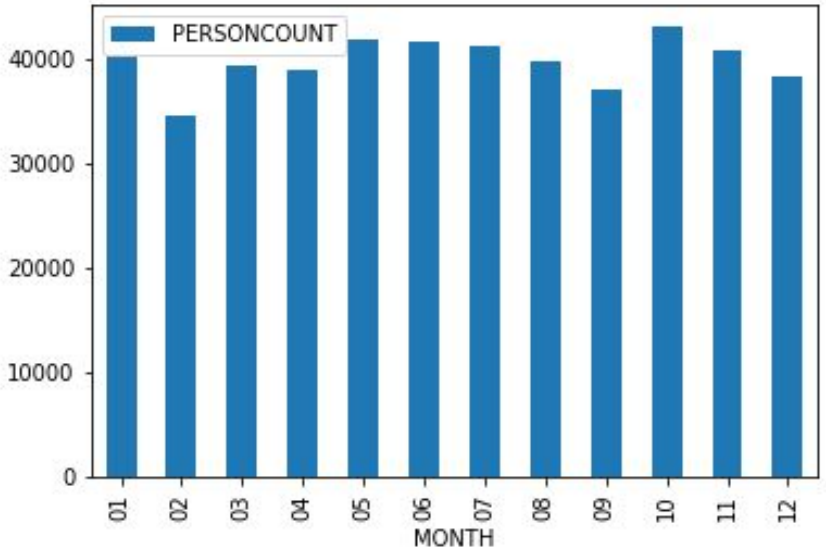
- Here we check the number of people affected by car accidents per year. We can told number of incidents are decreasing but we can't conclude it.

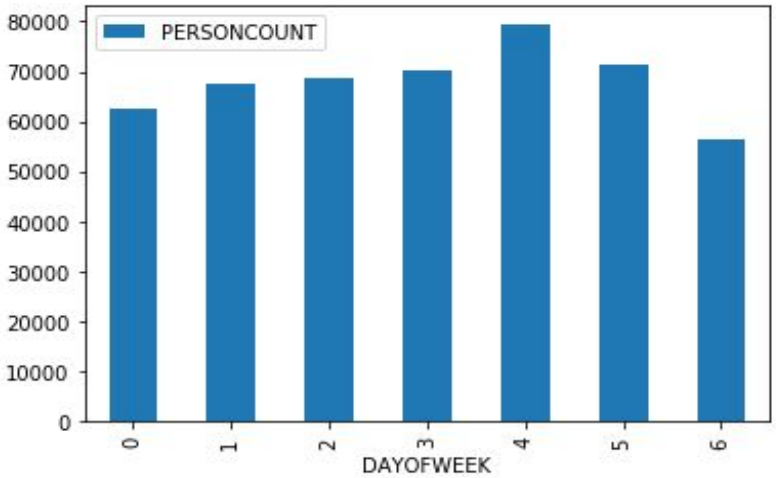
Number of person

January and October have more person incidents compare with others.

Highest number of person are face incidents on Friday and Saturday.

** driving on weekend should be more carefully





K-Nearest Neighbor (KNN)

- KNN will help us predict the severity code of an outcome by finding the most similar to the data point within k distance.

In [34]:
```
from sklearn.neighbors import KNeighborsClassifier
k = 4
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
yhat = neigh.predict(X_test)
yhat[0:5]
```

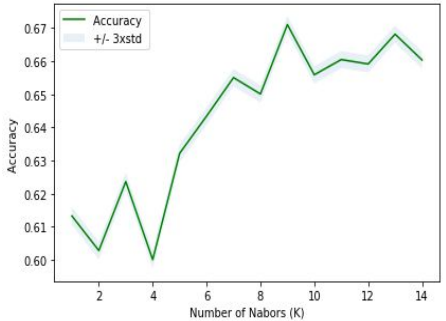
Out[34]: array([1, 2, 1, 2, 2])

In [35]:
```
from sklearn import metrics
Ks = 15
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
ConfustionMx = [];
for n in range(1,Ks):

    neigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat=neigh.predict(X_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)


    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])

mean_acc
```

Out[35]:
```
array([0.61328841, 0.60285512, 0.62366259, 0.6001064 , 0.63217474,
       0.64340604, 0.65502158, 0.65008571, 0.67101141, 0.6558787 ,
       0.66045989, 0.65912987, 0.66811491, 0.66034167])
```

The Best Accuracy with k=9

In [36]:
```
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.10)
plt.legend(('Accuracy ', '+/- 3xstd'))
plt.ylabel('Accuracy ')
plt.xlabel('Number of Nabors (K)')
plt.tight_layout()
plt.show()

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




```
The best accuracy was with 0.6710114086421942 with k= 9
```

model with K = 9

In [37]:
```
k = 9
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
yhat = neigh.predict(X_test)
yhat[0:5]
```

Out[37]: array([1, 2, 2, 2, 2])

KNN Model Evaluation

Knn Jaccard Score: 0.6710

KNN F1 score: 0.67073

In [34]:
```
from sklearn.metrics import jaccard_similarity_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
from sklearn.metrics import accuracy_score
```

In [35]:
```
#Predict y
knn_yhat = neigh.predict(X_test)

# jaccard
knn_jaccard = jaccard_similarity_score(y_test, knn_yhat)
print("KNN Jaccard index: ", knn_jaccard)


# f1_score
knn_f1_score = f1_score(y_test, knn_yhat, average='weighted')
print("KNN F1-score: ", knn_f1_score)
```

```
KNN Jaccard index:  0.6710114086421942
KNN F1-score:  0.670738826660399
```

In [36]:
```
# import library DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier

# create DecisionTreeClassifier as named d_tree
d_tree = DecisionTreeClassifier(criterion = 'entropy', max_depth = 7)
#train
d_tree.fit(X_train,y_train)
```

Out[36]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=7,
                       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')
```

In [37]:
```
dt_yhat = d_tree.predict(X_test)
dt_yhat
```

Out[37]:
```
array([1, 2, 2, ..., 1, 2, 1])
```

Decision Tree (DT)

A decision tree model gives us a layout of all possible outcomes so we can fully analyze the consequences of a decision. It context, the decision tree observes all possible outcomes of different weather conditions.

DT Model Evaluation

DT Jaccard Score: 0.698084

DT F1 score: 0.697032

```
In [38]: # jaccard
         dt_jaccard = jaccard_similarity_score(y_test, dt_yhat)
         print("DT Jaccard index: ", dt_jaccard)

         # f1_score
         dt_f1_score = f1_score(y_test, dt_yhat, average='weighted')
         print("DT F1-score: ", dt_f1_score)
```

```
DT Jaccard index:  0.6980847668026245
DT F1-score:  0.6970328787447123
```

Support Vector Machine (SVM)

SVM will provide with the Hyperline of dataset.

In [39]:
```
# import SVM library
from sklearn import svm

clf = svm.SVC()
clf.fit(X_train, y_train)
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/svm/bas
caled features. Set gamma explicitly to 'auto' or 'scale' to avoid t
  "avoid this warning.", FutureWarning)
```

Out[39]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

In [40]:
```
svm_yhat = clf.predict(X_test)
svm_yhat
```

Out[40]:
```
array([1, 2, 2, ..., 1, 2, 1])
```

SVM Model Evaluation

SVM Jaccard Score: 0.69566

SVM F1 score: 0.693033

```
In [41]: # jaccard
         svm_jaccard = jaccard_similarity_score(y_test, svm_yhat)
         print("SVM Jaccard index: ", svm_jaccard)

         # f1_score
         svm_f1_score = f1_score(y_test, svm_yhat, average='weighted')
         print("SVM F1-score: ", svm_f1_score)

         SVM Jaccard index:  0.6956611692380446
         SVM F1-score:  0.6930330013606999
```

Logistic Regression

Dataset only provides two severity code outcomes, model will only predict one of those two classes. This makes our data binary, which is perfect to use with logistic regression.

In [42]:
```
#import Library LogisticRegression
from sklearn.linear_model import LogisticRegression

#Train
lr = LogisticRegression(C=6, solver='liblinear').fit(X_train,y_train)
lr
```

Out[42]:
```
LogisticRegression(C=6, 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)
```

In [43]:
```
lr_yhat = lr.predict(X_test)
lr_yhat
```

Out[43]:
```
array([2, 1, 2, ..., 1, 1, 1])
```

LR Model Evaluation

LR Jaccard Score: 0.630401

LR F1 score: 0.628595

LR logloss: 0.6142731

In [44]:
```
lr_yhat_prob = lr.predict_proba(X_test)

# jaccard
lr_jaccard = jaccard_similarity_score(y_test, lr_yhat)
print("LR Jaccard: ", lr_jaccard)

# f1_score
lr_f1_score = f1_score(y_test, lr_yhat, average='weighted')
print("LR F1-score: ", lr_f1_score)

# logloss
lr_logloss = log_loss(y_test, lr_yhat_prob)
print("LR log loss: ", lr_logloss)
```

```
LR Jaccard:  0.6304013714015487
LR F1-score:  0.6285955849476118
LR log loss:  0.6142731759680288
```

Results

The accuracy of the built model using different evaluation metrics:

In [45]:
```
# create a jaccard list
jaccard_score = [knn_jaccard,dt_jaccard,svm_jaccard,lr_jaccard]

# create a f1-score list
f1_score = [knn_f1_score,dt_f1_score,svm_f1_score, lr_f1_score]

# create a Log_Loss list
log_los = ['NA','NA','NA',lr_logloss]
# fomulate the report format
df_report = pd.DataFrame(jaccard_score, index=['KNN','Decision Tree','SVM','Logistic Regression'])
df_report.columns = ['Jaccard']
df_report.insert(loc=1, column='F1-score', value=f1_score)
df_report.insert(loc=2, column='LogLoss', value=log_los)
df_report.columns.name = 'Algorithm'
df_report
```

Out[45]:

Algorithm	Jaccard	F1-score	LogLoss
KNN	0.671011	0.670739	NA
Decision Tree	0.698085	0.697033	NA
SVM	0.695661	0.693033	NA
Logistic Regression	0.630401	0.628596	0.614273

Closing

The algorithms implemented above did not give an accuracy score equal to or greater than 0.7, they all ranged from 0.6 to 0.7. Meaning, these models can predict the severity code of an accident with an accuracy equalling 60–70%.

- ➔ **The highest accuracy of the classifiers is 69%**
- ➔ **It needs to be trained on more data**
- ➔ **Better effort to collect data to reduce the number of missing values**

Thank You