PREDICTING CAR ACCIDENTS

Introduction

- Today, there are more vehicles running on the road.
- And because of this, there is also an increase in accident
- In this study, we will looking at car accident data to check which factors would cause accidents
- Might benefit government agencies and insurance companies to determine what can they do to avoid accidents

Data - Data description

- We will be using car accident information from the Seattle area from information gathered by the Seattle Department of Transportation (SDOT)
- The dependent variable that will be used in this study is the SEVERITYCODE which measures the degree of the accident.

Out[2]:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADCOND	LIGHTCOND	PEDROWNOTGRNT	SDOTCOLNUM	SPEEDING
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matched	Intersection	37475.0	...	Wet	Daylight	NaN	NaN	NaN
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matched	Block	NaN	...	Wet	Dark - Street Lights On	NaN	6354039.0	NaN
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matched	Block	NaN	...	Dry	Daylight	NaN	4323031.0	NaN
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matched	Block	NaN	...	Dry	Daylight	NaN	NaN	NaN
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matched	Intersection	34387.0	...	Wet	Daylight	NaN	4028032.0	NaN

5 rows × 38 columns

Data - Data preprocessing

- Dropped irrelevant variables

```
In [6]: #Drop unwanted variables

In [7]: df2 = df.drop(columns = ['OBJECTID', 'SEVERITYCODE.1', 'REPORTNO', 'INCKEY', 'COLDETKEY', 'X', 'Y',
                                 'STATUS', 'ADDRTYPE', 'INTKEY','LOCATION','EXCEPTRSNCODE','EXCEPTRSNDESC','SEVERITYDESC',
                                 'INCDATE','INCDTTM','JUNCTIONTYPE','SDOT_COLCODE','SDOT_COLDESC','PEDROWNOTGRNT', 'SDOTCOLNUM',
                                 'ST_COLCODE','ST_COLDESC','SEGLANEKEY','CROSSWALKKEY', 'HITPARKEDCAR', 'PEDCOUNT', 'PEDCYLCOUNT',
                         'PERSONCOUNT', 'VEHCOUNT', 'COLLISIONTYPE', 'SPEEDING', 'UNDERINFL', 'INATTENTIONIND'])
```

Methodology

- Import Python Libraries

```
In [1]: import pandas as pd
        import numpy as np
        import pandas as pd
        import itertools
        import matplotlib.pyplot as plt
        from matplotlib.ticker import NullFormatter
        import pandas as pd
        import numpy as np
        import matplotlib.ticker as ticker
        from sklearn import preprocessing
        %matplotlib inline
        import matplotlib.pyplot as plt
        import matplotlib.image as mpimg

        from sklearn import preprocessing, svm, metrics, ensemble, tree
        from sklearn.preprocessing import OneHotEncoder, RobustScaler
        from sklearn.compose import make_column_transformer
        from sklearn.pipeline import Pipeline
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score, classification_report
        from sklearn.linear_model import LogisticRegression
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.naive_bayes import GaussianNB
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import f1_score
        from sklearn.metrics import jaccard_similarity_score
```

Methodology

- Training and Testing data - Data will be split into 70% training and 30% testing

```
In [29]: X_train, X_test, Y_train, Y_test  = train_test_split(X,Y, test_size=0.3, random_state=42)
```

Modelling - K-Nearest Neighbors(KNN)

```
In [36]: #K-Nearest Neighbors

In [37]: from sklearn.neighbors import KNeighborsClassifier
         k = 25

In [38]: neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train, Y_train)
         neigh

         KNNyhat = neigh.predict(X_test)
         KNNyhat[0:5]

Out[38]: array([1, 1, 1, 1, 1])
```

```
In [58]: #KNN Evaluation

In [59]: # Jaccard Similarity Score
         jaccard_similarity_score(Y_test, KNNyhat)

Out[59]: 0.5237017729785467

In [60]: # F1-Score
         f1_score(Y_test, KNNyhat, average='macro')

Out[60]: 0.5196155093297656
```

Modelling - Decision Tree

```
In [39]: #Decision Tree

In [40]: colDataTree = DecisionTreeClassifier(criterion="entropy", max_depth = 7)
         colDataTree
         colDataTree.fit(X_train,Y_train)

Out[40]: 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 [41]: DTyhat = colDataTree.predict(X_test)
         print (DTyhat [0:5])
         print (Y_test [0:5])

         [1 2 2 2 2]
         [1 1 2 1 1]
```

```
In [61]: #Decision Tree Evaluation

In [62]: # Jaccard Similarity Score
         jaccard_similarity_score(Y_test, DTyhat)

Out[62]: 0.5626843869045914

In [63]: # F1-Score
         f1_score(Y_test, DTyhat, average='macro')

Out[63]: 0.5385207275454998
```

Modelling - Logistic Regression

```
In [42]: #Logistic Regression

In [43]: lr = LogisticRegression(C=0.03, solver='liblinear').fit(X_train,Y_train)
         lr

Out[43]: LogisticRegression(C=0.03, 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 [44]: LRyhat = lr.predict(X_test)
         LRyhat

         LRyhat_prob = lr.predict_proba(X_test)
         LRyhat_prob

Out[44]: array([[0.40364293, 0.59635707],
               [0.53529771, 0.46470229],
               [0.46743605, 0.53256395],
               ...,
               [0.46293233, 0.53706767],
               [0.46743605, 0.53256395],
               [0.67878612, 0.32121388]])
```

Modelling - Logistic Regression

```
In [64]: #Logistic Regression Evaluation

In [65]: # Jaccard Similarity Score
         jaccard_similarity_score(Y_test, LRyhat)

Out[65]: 0.523501274596855

In [66]: # F1-Score
         f1_score(Y_test, LRyhat, average='macro')

Out[66]: 0.5098573271706865

In [67]: # logloss
         yhat_prob = lr.predict_proba(X_test)
         log_loss(Y_test, yhat_prob)

Out[67]: 0.6855290309651024
```

Modelling - Support Vector Machines (SVM)

```
In [52]: #Use RBF

         clf = svm.SVC(kernel='rbf')
         clf.fit(X_train, Y_train)
         RBYhat = clf.predict(X_test)
         RBYhat

           /opt/conda/envs/Python36/lib/python3.6/site-pa
           2 to account better for unscaled features. Set
             "avoid this warning.", FutureWarning)

Out[52]: array([1, 2, 2, ..., 1, 2, 1])
```

```
In [68]: #SVM Evaluation

In [69]: # Jaccard Similarity Score
         jaccard_similarity_score(Y_test, RBYhat)

Out[69]: 0.5623979606450319

In [70]: # F1-Score
         f1_score(Y_test, RBYhat, average='macro')

Out[70]: 0.5386632235323728
```

Modelling - Random Forest Classifier

```
In [51]: from sklearn.ensemble import RandomForestClassifier

In [52]: clf=RandomForestClassifier(n_estimators=100)

         clf.fit(X_train,Y_train)

         RFY_pred=clf.predict(X_test)


         acc=accuracy_score(Y_test, RFY_pred)



         print("[Randon forest algorithm] accuracy_score: {:.3f}.".format(acc))

         [Randon forest algorithm] accuracy_score: 0.561.
```

```
In [79]: #Random Forest Classifier

In [80]: #Jaccard Similarity Score

In [81]: jaccard_similarity_score(Y_test, RFY_pred)

Out[81]: 0.5610231145991464

In [82]: # F1-Score
         f1_score(Y_test, RFY_pred, average='macro')

Out[82]: 0.5329956691944944
```

Conclusion

- Decision Tree and SVM has the highest Jaccard and F-1 score
- Decision Tree shows that weather, road and light conditions definitely affect the chances of getting into a car accident

Model	Jaccard SCore	F-1	Logloss
KNN	0.5237	0.5196	
Decision Tree	0.5627	0.5385	
Logistic Regression	0.5235	0.5099	0.6855
SVM	0.5624	0.5387	
	0.561	0.533	

Future directions

- Use other statistical models / methods
- Consider also other variables that may affect car accidens
- Consider checking how findings affect existing road policies