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

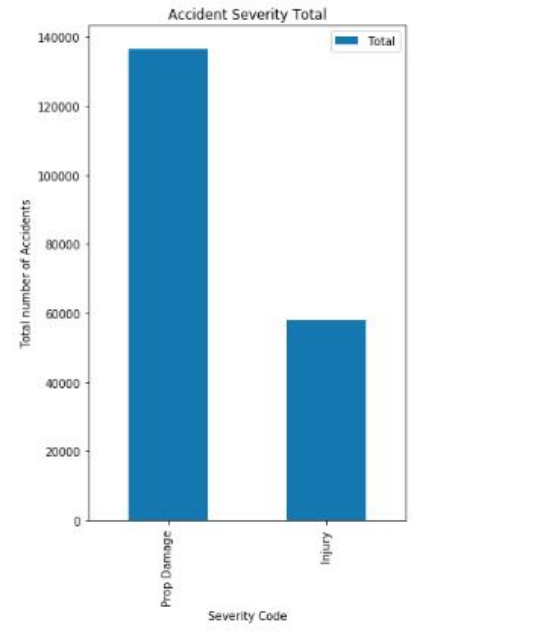
Data

- We will be using car accident information from the Seattle area from information gathered by the Seattle Department of Transportation (SDOT)
- We will be using the following variables: SEVERITYCODE, COLLISIONTYPE, UNDERINFL, INATTENTIONIND, WEATHER, ROADCON, LIGHTCOND, SPEEDING
- However, because of a lot of missing values, we had to remove INATTENTIONIND and SPEEDING as well.

```
In [10]: inatt = df['INATTENTIONIND'].isna().sum() / df.shape[0] * 100
         inatt = round(inatt, 2)
         print ('In the INATTENTIONIND attribute,',inatt,'% of the values are unknown.')

         In the INATTENTIONIND attribute, 84.69 % of the values are unknown.

In [11]: speeding = df['SPEEDING'].isna().sum() / df.shape[0] * 100
         speeding = round(speeding, 2)
         print ('In the SPEEDING attribute,',speeding,'% of the values are unknown.')

         In the SPEEDING attribute, 95.21 % of the values are unknown.
```

Data

- The dependent variable that will be used in this study is the SEVERITYCODE which measures the degree of the accident.



Methodology

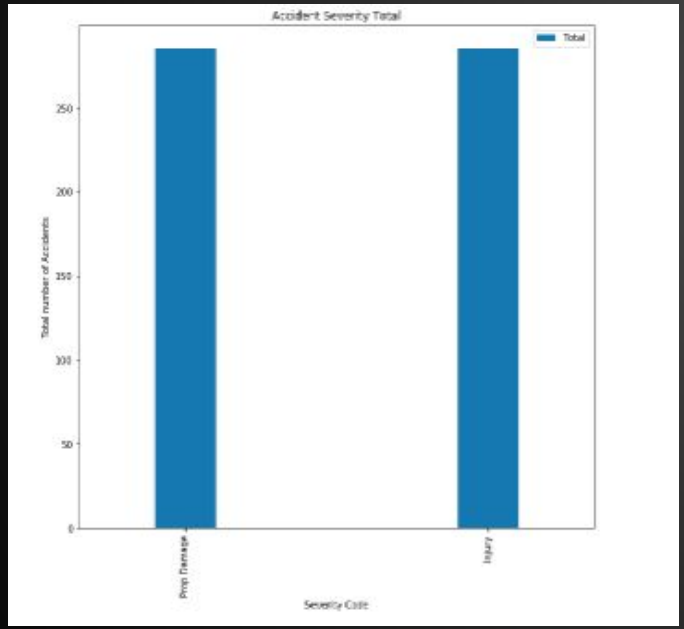
- Import Python Libraries

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
```

Data

- Because of the large difference in each type of SEVERITYCODE, we balanced out the variable



Data

- Applied “One Hot Encoding” to turn categorical variables into numerical variables

```
In [30]: df_test = df_tr[['SEVERITYCODE','COLLISIONTYPE','UNDERINFL','WEATHER','ROADCOND','LIGHTCOND']]
         df_test = pd.concat([df_test,pd.get_dummies(df_tr['COLLISIONTYPE'])], axis=1)
         df_test.drop(['COLLISIONTYPE'], axis = 1,inplace=True)
         df_test = pd.concat([df_test,pd.get_dummies(df_tr['WEATHER'])], axis=1)
         df_test.drop(['WEATHER'], axis = 1,inplace=True)
         df_test = pd.concat([df_test,pd.get_dummies(df_tr['ROADCOND'])], axis=1)
         df_test.drop(['ROADCOND'], axis = 1,inplace=True)
         df_test = pd.concat([df_test,pd.get_dummies(df_tr['LIGHTCOND'])], axis=1)
         df_test.drop(['LIGHTCOND'], axis = 1,inplace=True)

In [31]: df_test.head()
```

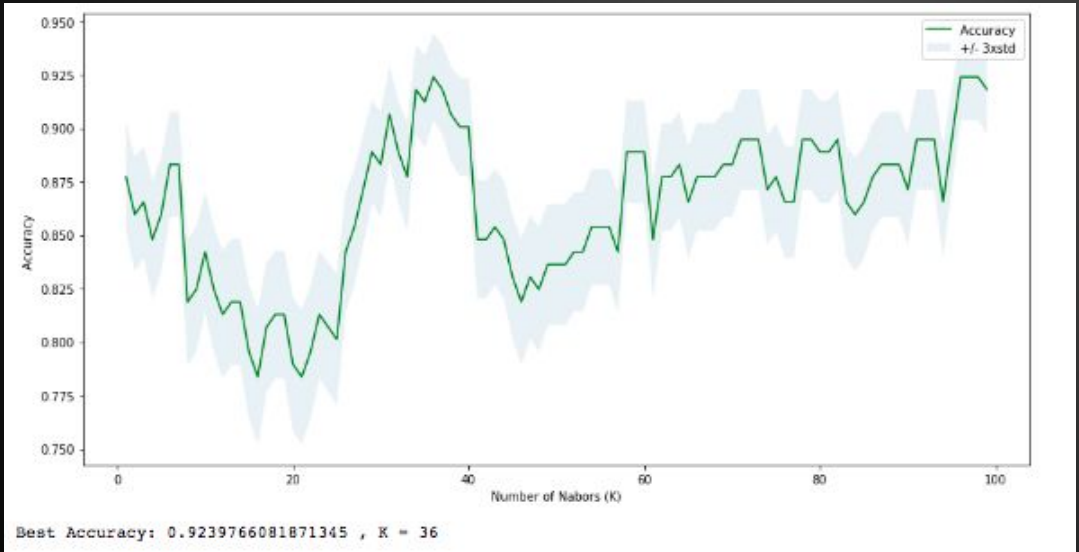
Out[31]:

	SEVERITYCODE	UNDERINFL	Angles	CT_Other	Cycles	Head On	Left Turn	Parked Car	Pedestrian	Rear Ended	...	Standing Water	Wet	Dark - No Street Lights	Dark - Street Lights Off	Dark - Street Lights On	Dawn	Daylight	Dusk	LC_Other	LC_Unknown
11657	1	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	1	0	0	0
20012	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
66649	1	0	0	1	0	0	0	0	0	0	...	0	1	0	0	0	0	1	0	0	0
120972	1	0	0	0	0	1	0	0	0	0	...	0	1	0	0	0	0	1	0	0	0
143764	1	0	0	1	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0

5 rows × 34 columns

Modelling - K-Nearest Neighbors (KNN)

- KNN will be used to categorize the severity of an outcome based on other outcomes with the nearest data points at k distance. For this study, the KNN is most accurate when k = 36.



Modelling - Decision Tree

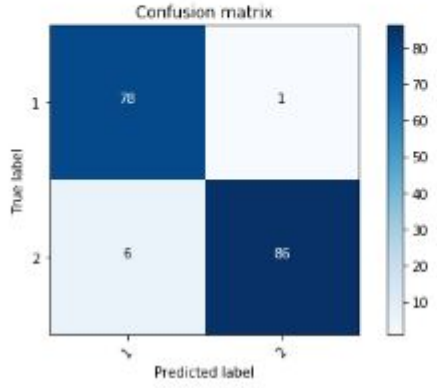
- A Decision Tree builds a classification model in the shape of a tree. It classifies the data into smaller subsets or decision nodes or leaf nodes. A decision node normally has two branches while a leaf node may be a decision node or classification. Unlike some models, decision trees can handle both categorical and numerical data.

```
In [75]: yhatDEC = Tree.predict(X)
         DTJaccard = jaccard_similarity_score(y, yhatDEC)
         DTF1 = f1_score(y, yhatDEC, average='weighted')
         print("Avg F1-Score: %.2f" % DTF1 )
         print("Decision Tree Jaccard Score: %.2f" % DTJaccard)

         Avg F1-Score: 1.00
         Decision Tree Jaccard Score: 1.00
```

Modelling - Support Vector Machines (SVM)

- We will also be constructing a model using the SVM method to categorize the possible severity code of an outcome into two-group classifications.

	precision	recall	f1-score	support
1	0.93	0.99	0.96	79
2	0.99	0.93	0.96	92
micro avg	0.96	0.96	0.96	171
macro avg	0.96	0.96	0.96	171
weighted avg	0.96	0.96	0.96	171



Modelling - Logistic Regression

- Logistic Regression is used to predict a binary outcome which can have only two results. Since we only have two possible outcomes in our Severity Code variable, we could expect that Logistic Regression would be a good modelling method given our data.

```
In [77]: yhatLOG = LogR.predict(X)
         yhatLOGproba = LogR.predict_proba(X)
         LogRJaccard = jaccard_similarity_score(y, yhatLOG)
         LogRF1 = f1_score(y, yhatLOG, average='weighted')
         Logloss = log_loss(y, yhatLOGproba)
         print("Log Loss: : %.2f" % Logloss)
         print("Avg F1-Score: %.4f" % LogRF1)
         print("LOG Jaccard Score: %.4f" % LogRJaccard)

         Log Loss: : 0.30
         Avg F1-Score: 1.0000
         LOG Jaccard Score: 1.0000
```

Results

- Decision Tree has the highest Jaccard and F-1 score

Model	Jaccard Score	F-1	Logloss
KNN	0.94	0.94	
Decision Tree	1	1	
SVM	0.98	0.98	
Logistic Regression	1	1	0.3

Future directions

- Additional data to be used
- Consider also other variables that may affect car accidents
- Consider checking how findings affect existing road policies