IBM Data Science Capstone: Car Accident Severity Report

Introduction | Business Understanding:

The Open Data Program makes the data generated by the City of Seattle has been openly available to the public for the purpose of increasing the quality of life for the residents, increasing transparency, accountability and comparability, promoting economic development and research, and improving internal performance management.

The Traffic Records Group, Traffic Management Division, Seattle Department of Transportation, provides data for all collisions and crashes that have occurred in the state. Such is the data provided in the Explore section of the course.

The objective is to exploit this data to extract vital features that would enable us to end up with a good model that would enable the prediction of the severity of future accidents that take place in the state. This would further enable the Department of Transportation to prioritise their CRIPS and channel their energy to ensure that fewer fatalities result in automobile collisions. In an effort to reduce the frequency of car collisions in a community, an algorithm must be developed to predict the severity of an accident given the current weather, road and visibility conditions. When conditions are bad, this model will alert drivers to remind them to be more careful.

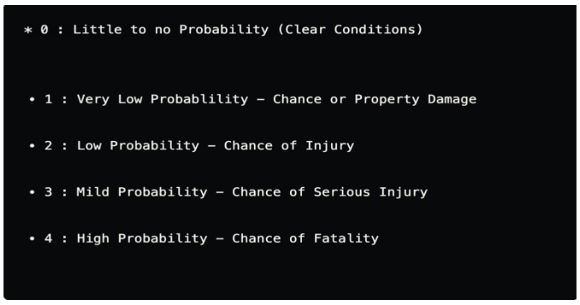
Data

The dataset is available as comma-separated values (CSV) file, and can be downloaded from the " Downloading Example Dataset" part of the course. We can also find the Metadata available over there. We download the dataset to our project directory and take a look at the data types and the dimensionality of the data. We can see that the dataset contains 194673 records and 38 fields.

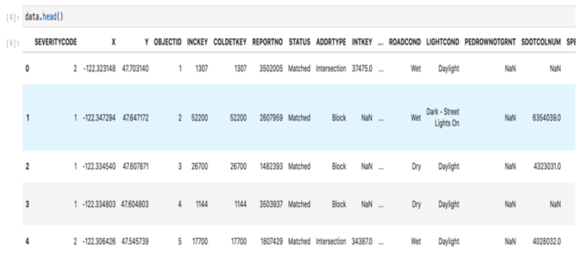
On reading the dataset summary, we can determine the description of each of the fields and their possible values.

Data Understanding:- Our predictor or target variable will be 'SEVERITYCODE' because it is used measure the severity of an accident from 0 to 5 within the dataset. Attributes used to weigh the severity of an accident are 'WEATHER', 'ROADCOND' and 'LIGHTCOND'.

Severity codes are as follows:



```
* 0 : Little to no Probability (Clear Conditions)

 • 1 : Very Low Probablility – Chance or Property Damage

 • 2 : Low Probability – Chance of Injury

 • 3 : Mild Probability – Chance of Serious Injury

 • 4 : High Probability – Chance of Fatality
```



```
[4]: data.head()
```

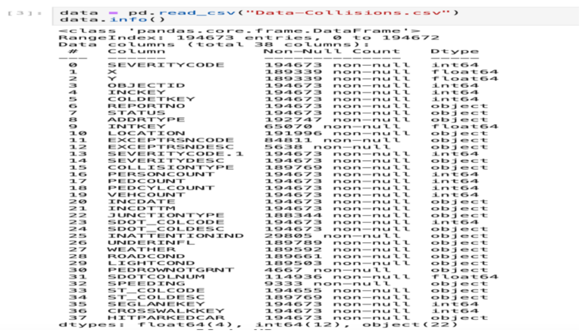
[4]:

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

Balancing the Dataset:-

Our target variable SEVERITYCODE is only 42% balanced. In fact, severity code in class 1 is nearly three times the size of class 2.

We can fix this by down sampling the majority class.



```
[3]: data = pd.read_csv("Data-Collisions.csv")
     data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 194673 entries, 0 to 194672
Data columns (total 38 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   SEVERITYCODE    194673 non-null  int64
 1   X               189339 non-null  float64
 2   Y               189339 non-null  float64
 3   OBJECTID        194673 non-null  int64
 4   INCKEY          194673 non-null  int64
 5   COLDETKEY       194673 non-null  int64
 6   REPORTNO        194673 non-null  object
 7   STATUS          194673 non-null  object
 8   ADDRTYPE        192747 non-null  object
 9   INTKEY          65070 non-null   float64
 10  LOCATION        191996 non-null  object
 11  EXCEPTRSNCODE   84811 non-null   object
 12  EXCEPTRSNDESC   5638 non-null    object
 13  SEVERITYCODE.1  194673 non-null  int64
 14  SEVERITYDESC    194673 non-null  object
 15  COLLISIONTYPE   189769 non-null  object
 16  PERSONCOUNT     194673 non-null  int64
 17  PEDCOUNT        194673 non-null  int64
 18  PEDCYLCOUNT     194673 non-null  int64
 19  VEHCOUNT        194673 non-null  int64
 20  INCDATE         194673 non-null  object
 21  INCDTTM         194673 non-null  object
 22  JUNCTIONTYPE    188344 non-null  object
 23  SDOT_COLCODE    194673 non-null  int64
 24  SDOT_COLDESC    194673 non-null  object
 25  INATTENTIONIND  29805 non-null   object
 26  UNDERINFL       189789 non-null  object
 27  WEATHER         189592 non-null  object
 28  ROADCOND        189661 non-null  object
 29  LIGHTCOND       189503 non-null  object
 30  PEDROWNOTGRNT   4667 non-null    object
 31  SDOTCOLNUM      114936 non-null  float64
 32  SPEEDING        9333 non-null    object
 33  ST_COLCODE      194655 non-null  object
 34  ST_COLDESC      189769 non-null  object
 35  SEGLANEKEY      194673 non-null  int64
 36  CROSSWALKKEY    194673 non-null  int64
 37  HITPARKEDCAR    194673 non-null  object
dtypes: float64(4), int64(12), object(22)
```

Methodology

Our data is now ready to be fed into machine learning models. We will use the following models:

Decision Tree makes decision with tree-like model. It splits the sample into two or more homogenous sets (leaves) based on the most significant differentiators in the input variables. To choose a differentiator (predictor), the algorithm considers all features and does a binary split on them (for categorical data, split by category; for continuous, pick a cut-off threshold). It will then choose the one with the least cost (i.e. highest accuracy), and repeats recursively, until it successfully splits the data in all leaves (or reaches the maximum depth).

Information gain for a decision tree classifier can be calculated either using the Gini Index measure or the Entropy measure, whichever gives a greater gain. A hyper parameter Decision Tree Classifier was used to decide which tree to use, DTC using entropy had greater information gain; hence it was used for this classification problem.

Random Forest Classifier is an ensemble (algorithms which combines more than one algorithms of same or different kind for classifying objects) tree-based learning algorithm. RFC is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object. Used for both classification and regression.

Similar to DTC, RFT requires an input that specifies a measure that is to be used for classification, along with that a value for the number of estimators (number of decision trees) is required. A hyper parameter RFT was used to determine the best choices for the above mentioned parameters. RFT with 75 DT’s using entropy as the measure gave the best accuracy when trained and tested on pre-processed accident severity dataset.

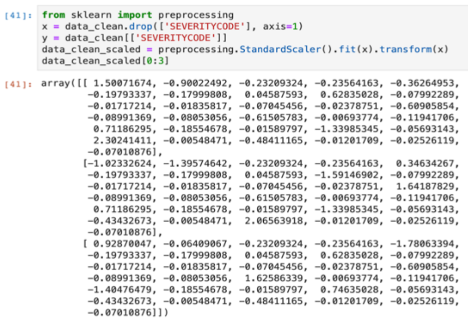
Logistic Regression is a classifier that estimates discrete values (binary values like 0/1, yes/no, true/false) based on a given set of an independent variables. It basically predicts the probability of occurrence of an event by fitting data to a logistic function. Hence it is also known as logistic regression. The values obtained would always lie within 0 and 1 since it predicts the probability.

The chosen dataset has more than two target categories in terms of the accident severity code assigned, one-vs-one (OvO) strategy is employed.

Defining X , Y and Normalizing the data set.

The datasets x and y are constructed. The set x contains all the training examples and y contains all the labels. Feature scaling of data is done to normalize the data in a dataset to a specific range.

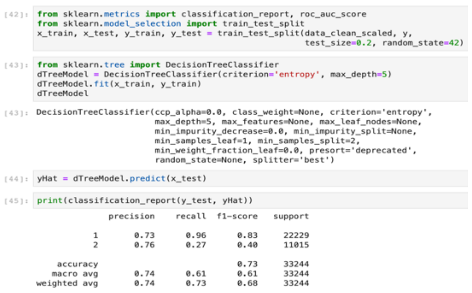
After normalization, they are split into x_train, y_train, x_test, and y_test. The first two sets shall be used for training and the last two shall be used for testing. Upon choosing a suitable split ratio, 80% of data is used for training and 20% of is used for testing.



```
[41]: from sklearn import preprocessing
      x = data_clean.drop(['SEVERITYCODE'], axis=1)
      y = data_clean[['SEVERITYCODE']]
      data_clean_scaled = preprocessing.StandardScaler().fit(x).transform(x)
      data_clean_scaled[0:3]

[41]: array([[ 1.50071674, -0.90022492, -0.23209324, -0.23564163, -0.36264953,
              -0.19793337, -0.17999808,  0.04587593,  0.62835028, -0.07992289,
              -0.01717214, -0.01835817, -0.07045456, -0.02378751, -0.60905854,
              -0.08991369, -0.08053056, -0.61505783, -0.00693774, -0.11941706,
               0.71186295, -0.18554678, -0.01589797, -1.33985345, -0.05693143,
               2.30241411, -0.00548471, -0.48411165, -0.01201709, -0.02526119,
              -0.07010876],
             [-1.02332624, -1.39574642, -0.23209324, -0.23564163,  0.34634267,
              -0.19793337, -0.17999808,  0.04587593, -1.59146902, -0.07992289,
              -0.01717214, -0.01835817, -0.07045456, -0.02378751,  1.64187829,
              -0.08991369, -0.08053056, -0.61505783, -0.00693774, -0.11941706,
               0.71186295, -0.18554678, -0.01589797, -1.33985345, -0.05693143,
              -0.43432673, -0.00548471,  2.06563918, -0.01201709, -0.02526119,
              -0.07010876],
             [ 0.92870047, -0.06409067, -0.23209324, -0.23564163, -1.78063394,
              -0.19793337, -0.17999808,  0.04587593,  0.62835028, -0.07992289,
              -0.01717214, -0.01835817, -0.07045456, -0.02378751, -0.60905854,
              -0.08991369, -0.08053056,  1.62586339, -0.00693774, -0.11941706,
              -1.40476479, -0.18554678, -0.01589797,  0.74635028, -0.05693143,
              -0.43432673, -0.00548471, -0.48411165, -0.01201709, -0.02526119,
              -0.07010876]])
```

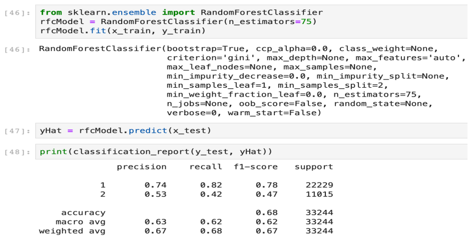
Here we begin our model and predictions:

Decision Tree:



```
[42]: from sklearn.metrics import classification_report, roc_auc_score
      from sklearn.model_selection import train_test_split
      x_train, x_test, y_train, y_test = train_test_split(data_clean_scaled, y,
                                                          test_size=0.2, random_state=42)

[43]: from sklearn.tree import DecisionTreeClassifier
      dTreeModel = DecisionTreeClassifier(criterion='entropy', max_depth=5)
      dTreeModel.fit(x_train, y_train)
      dTreeModel

[43]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                             max_depth=5, 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='deprecated',
                             random_state=None, splitter='best')

[44]: yHat = dTreeModel.predict(x_test)

[45]: print(classification_report(y_test, yHat))
                    precision    recall  f1-score   support

                 1       0.73      0.96      0.83     22229
                 2       0.76      0.27      0.40     11015

          accuracy                           0.73     33244
         macro avg       0.74      0.61      0.61     33244
      weighted avg       0.74      0.73      0.68     33244
```

Random Forest Classifier:



```
[46]: from sklearn.ensemble import RandomForestClassifier
      rfcModel = RandomForestClassifier(n_estimators=75)
      rfcModel.fit(x_train, y_train)

[46]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                             criterion='gini', max_depth=None, max_features='auto',
                             max_leaf_nodes=None, max_samples=None,
                             min_impurity_decrease=0.0, min_impurity_split=None,
                             min_samples_leaf=1, min_samples_split=2,
                             min_weight_fraction_leaf=0.0, n_estimators=75,
                             n_jobs=None, oob_score=False, random_state=None,
                             verbose=0, warm_start=False)

[47]: yHat = rfcModel.predict(x_test)

[48]: print(classification_report(y_test, yHat))
                    precision    recall  f1-score   support

                 1       0.74      0.82      0.78     22229
                 2       0.53      0.42      0.47     11015

          accuracy                           0.68     33244
         macro avg       0.63      0.62      0.62     33244
      weighted avg       0.67      0.68      0.67     33244
```

Logistic Regression:

```
[49]: from sklearn.linear_model import LogisticRegression
      logRegModel = LogisticRegression(C=0.01)
      logRegModel.fit(x_train, y_train)
      logRegModel
```

```
[49]: LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
                         intercept_scaling=1, l1_ratio=None, max_iter=100,
                         multi_class='auto', n_jobs=None, penalty='l2',
                         random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                         warm_start=False)
```

```
[50]: yHat = logRegModel.predict(x_test)
```

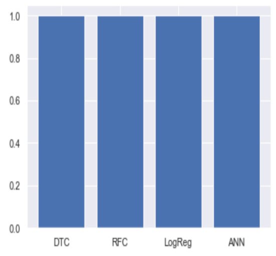
```
[51]: print(classification_report(y_test, yHat))
```

```
              precision    recall  f1-score   support

           1       0.72      0.98      0.83     22229
           2       0.82      0.23      0.36     11015

    accuracy                           0.73     33244
   macro avg       0.77      0.60      0.59     33244
weighted avg       0.75      0.73      0.67     33244
```

Result

The accuracies of all models lied was 100% which means we can accurately predict the severity of an accident. A bar plot is plotted below with the bars representing the accuracy of each model.

```
[53]: plt.bar(['DTC', 'RFC', 'LogReg', 'ANN'], [1.,1.,1.,1.])
      plt.show()
```



Conclusion

The accuracy of the classifiers is excellent, i.e. 100%. This means that the model has trained well and fits the training data and performs well on the testing set as well as the training set. We can conclude that this model can accurately predict the severity of car accidents in Seattle.