



PREDICTING ACCIDENT SEVERITY

- Seattle Car Accident Data Predictive Analysis

Sourabh Dixit

IBM DATA SCIENCE CAPSTONE – October 2020

Predicting Car Accident Severity in Seattle

INTRODUCTION / BUSINESS PROBLEM

Road traffic injuries are currently estimated to be the eighth leading cause of death across all age groups globally, and are predicted to become the seventh leading cause of death by 2030.

Analysing a significant range of factors, including weather conditions, special events, roadworks, traffic jams among others, an accurate prediction of the severity of the accidents can be performed.

These insights, could allow law enforcement bodies to allocate their resources more effectively in advance of potential accidents, preventing when and where a severe accidents can occur as well as saving both, time and money. In addition, this knowledge of a severe accident situation can be warned to drivers so that they would drive more carefully or even change their route if it is possible or to hospital which could have set everything ready for a severe intervention in advance.

Governments should be highly interested in accurate predictions of the severity of an accident, in order to reduce the time of arrival and thus save a significant amount of people each year. Others interested could be private companies investing in technologies aiming to improve road safeness.

The Seattle government is going to prevent avoidable car accidents by employing methods that alert drivers through alert display system and police to be more careful in critical situations. In most cases, not paying enough attention during driving, drugs and alcohol or driving at very high speed for fun/competition are the main causes of occurring accidents that can be prevented by deploying harsher regulations. Besides the aforementioned causes, weather, visibility, or road conditions are the major uncontrollable factors that can be prevented by revealing hidden patterns in the data and announcing warning to the local government, police and drivers on the targeted roads.

The target audience of the project is local Seattle government, police, rescue groups and insurance organisation as well. This model results will provide them to make insightful decisions for reducing the number of accidents for the city.

DATASET

The data was collected by the Seattle Police Department and Accident Traffic Records Department from 2004 to present. The data consists of 37 independent variables and 194,673 rows. The dependent variable, “SEVERITYCODE”, contains numbers that correspond to different levels of severity caused by an accident from 1 and 2.

Furthermore, because of the existence of null values in some records, the data needs to be preprocessed before any further processing. Among all the features, I believe the following features have the most influence in the accuracy of the predictions:

“WEATHER”, “ROADCOND”, “LIGHTCOND” The target variable is “SEVERITYCODE”.

The models aim was to predict the severity of an accident, considering that, the variable of Severity Code was in the form of 1 (Property Damage Only) and 2 (Injury Collision) which were encoded to the form of 0 (Property Damage Only) and 1 (Injury Collision). Furthermore, the Y was given value of 1 whereas N and no value was given 0 for the variables Inattention, Speeding and Under the influence. For lighting condition, Light was given 0 along with Medium as 1 and Dark as 2. For Road Condition, Dry was assigned 0, Mushy was assigned 1 and Wet was given 2. As for Weather Condition, 0 is Clear, Overcast is 1, Windy is 2 and Rain and Snow was given 3. 0 was assigned to the element of each variable which can be the least probable cause of severe accident whereas a high number represented adverse condition which can lead to a higher accident severity.

The Target Variable – ‘SEVERITYCODE’

Find the value distribution below from the notebook:

```
In [7]: df['SEVERITYCODE'].value_counts()
Out[7]: 1    136485
        2     58188
        Name: SEVERITYCODE, dtype: int64
```

Weather vs Severity code value counts :

```
In [8]: df.groupby(["WEATHER"])["SEVERITYCODE"].value_counts(normalize=True)
```

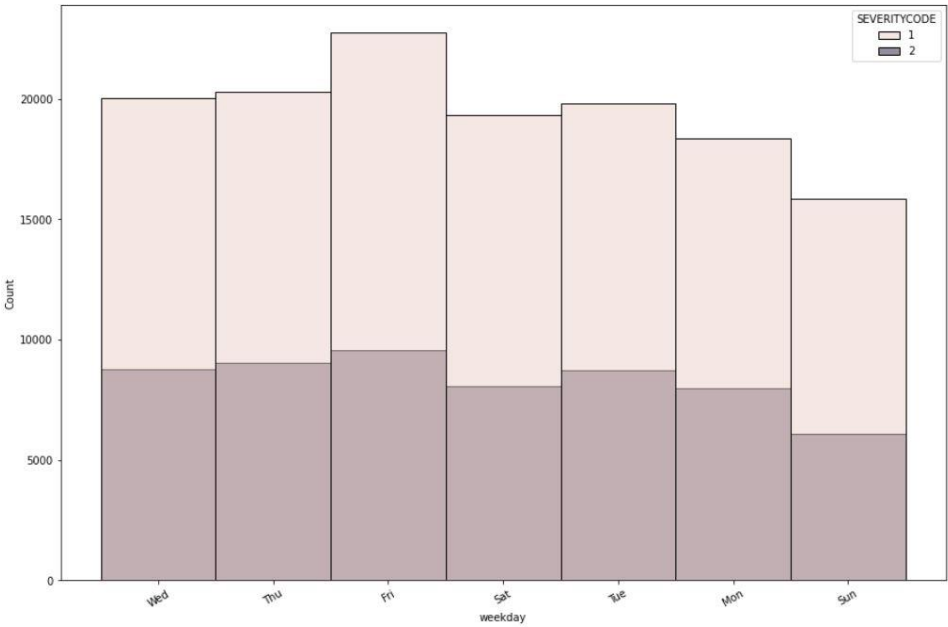
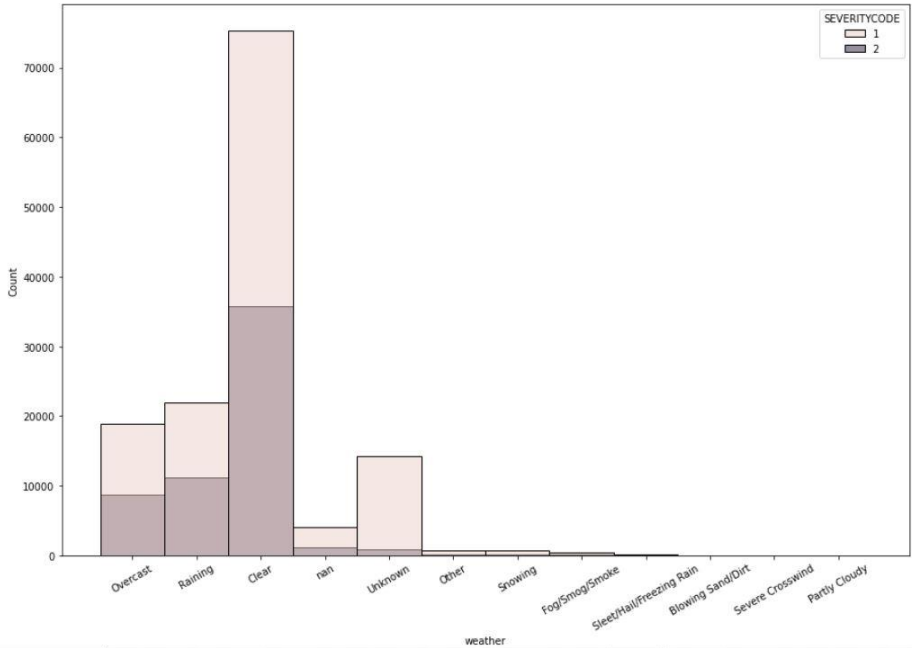
```
Out[8]: WEATHER      SEVERITYCODE
Blowing Sand/Dirt    1      0.732143
                   2      0.267857
Clear                1      0.677509
                   2      0.322491
Fog/Smog/Smoke       1      0.671353
                   2      0.328647
Other                1      0.860577
                   2      0.139423
Overcast             1      0.684456
                   2      0.315544
Partly Cloudy        2      0.600000
                   1      0.400000
Raining              1      0.662815
                   2      0.337185
Severe Crosswind     1      0.720000
                   2      0.280000
Sleet/Hail/Freezing Rain 1      0.752212
                   2      0.247788
Snowing              1      0.811466
                   2      0.188534
Unknown              1      0.945928
                   2      0.054072
Name: SEVERITYCODE, dtype: float64
```

DATA VISUALIZATION

Accidents vs Weather visualization

```
In [12]: df['weather'] = df['WEATHER'].astype('str')

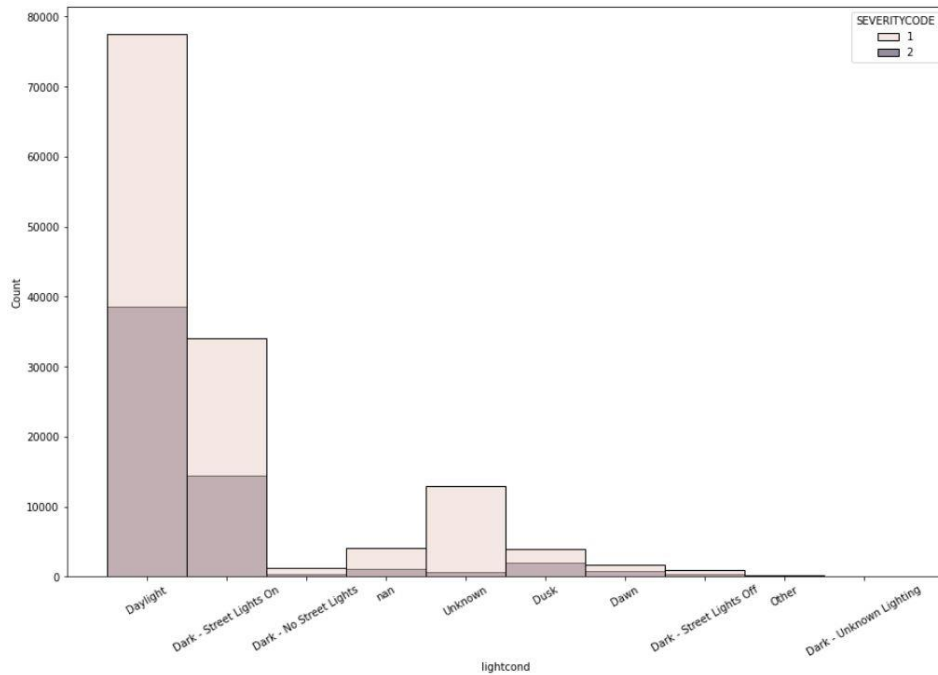
plt.figure(figsize=(15,10))
ax = sns.histplot(x="weather", hue="SEVERITYCODE", data=df)
plt.setp(ax.get_xticklabels(), rotation=30);
```



Accidents vs Light Condition Visualization

```
In [14]: df['lightcond'] = df['LIGHTCOND'].astype('str')

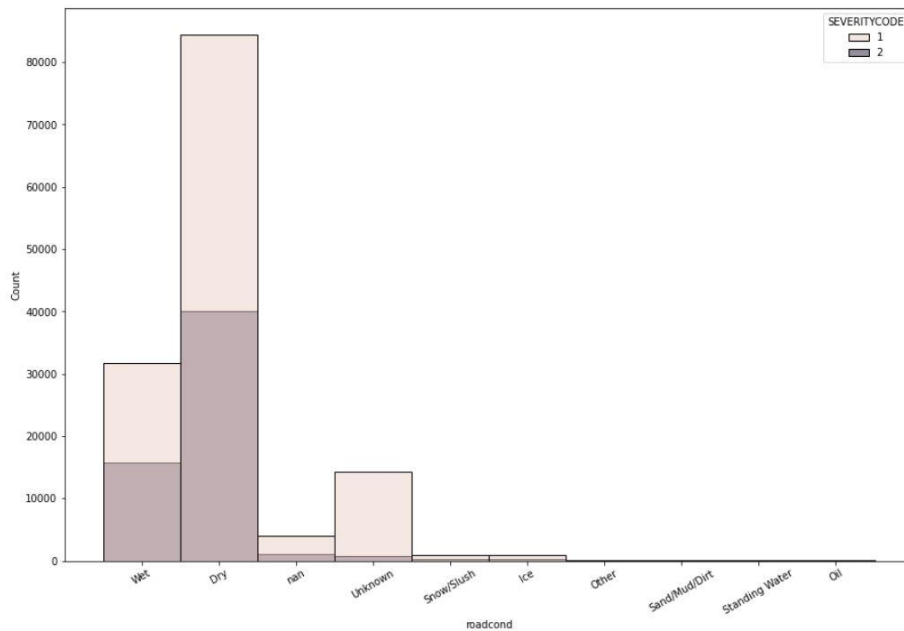
plt.figure(figsize=(15,10))
ax = sns.histplot(x="lightcond", hue="SEVERITYCODE", data=df)
plt.setp(ax.get_xticklabels(), rotation=30);
```

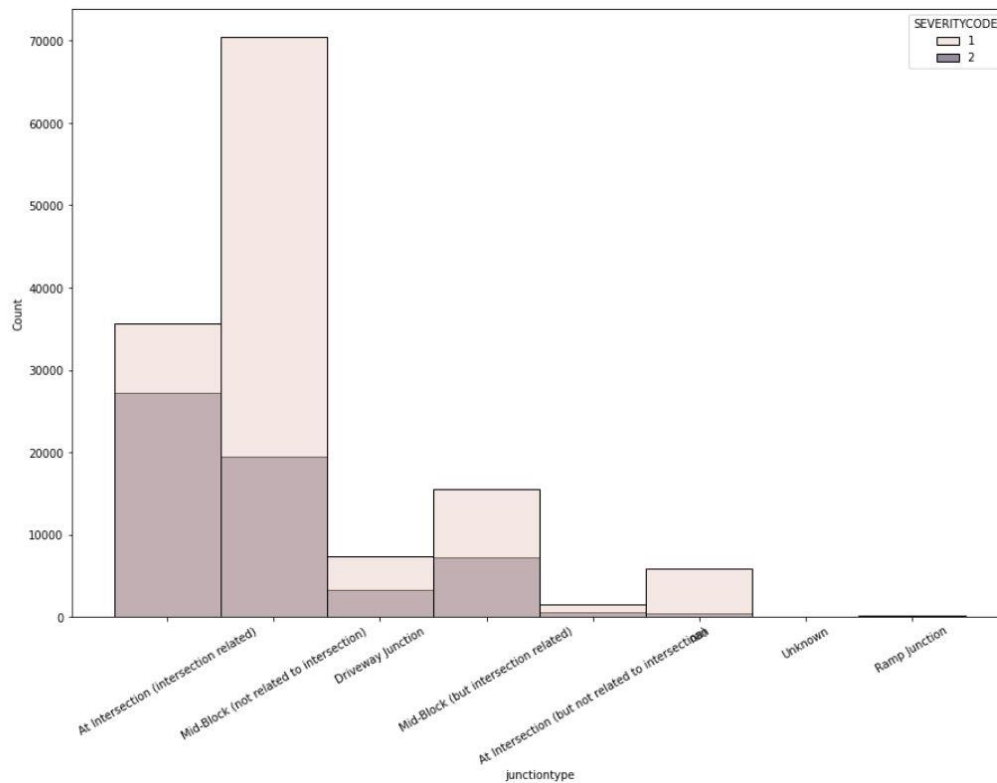


Accidents vs Road Condition Visualization

```
In [15]: df['roadcond'] = df['ROADCOND'].astype('str')

plt.figure(figsize=(15,10))
ax = sns.histplot(x="roadcond", hue="SEVERITYCODE", data=df)
plt.setp(ax.get_xticklabels(), rotation=30);
```





DATA PREPROCESSING

The dataset is analysed and majority of the variables being categorical are converted into Booleans using pandas get_dummies functionality. Once converted, I chose 'Weather', 'Road Conditions' and 'Light Conditions' as the most descriptive independent variables to predict our target variable.

To balance the class imbalance, Downsampling to the lower majority class group was applied.

```

In [36]: from sklearn.utils import resample
pre_df_maj = df[df.SEVERITYCODE==1]
pre_df_min = df[df.SEVERITYCODE==2]

pre_df_maj_dsampl = resample(pre_df_maj,
                             replace=False,
                             n_samples=50000,
                             random_state=123)

pre_df_min_sample = resample(pre_df_min, replace = False, n_samples = 50000, random_state=123)
balanced_df = pd.concat([pre_df_maj_dsampl, pre_df_min_sample])

balanced_df.SEVERITYCODE.value_counts()

```

```

Out[36]: 2    50000
         1    50000
         Name: SEVERITYCODE, dtype: int64

```

MACHINE LEARNING MODELS (CLASSIFICATION)

After successfully applying pre-processing techniques, balancing the class imbalance and downsampling, our data is ready to be fed to machine learning algorithms to predict our target class.

Normalizing the data

```
In [55]: X = preprocessing.StandardScaler().fit(X).transform(X)
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3, random_state=42)

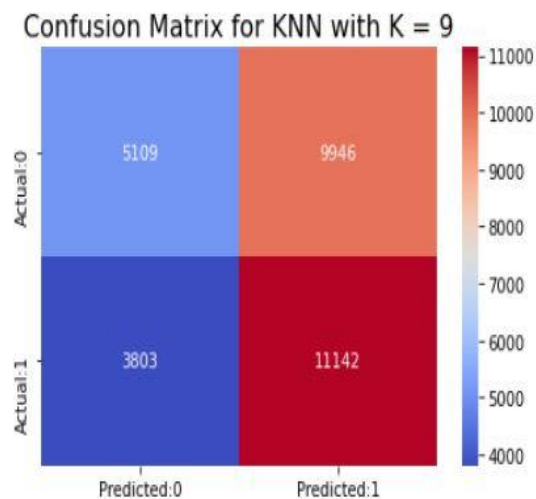
print ('Train set:', X_train.shape, y_train.shape)
print ('Test set:', X_test.shape, y_test.shape)

Train set: (70000, 26) (70000,)
Test set: (30000, 26) (30000,)
```

K Nearest Neighbor

Hyper-parameter tuning was applied to find the best value of K, resulting in best accuracies for k=9. The results were as follows:

```
In [59]: plot_conf_mat(y_test, y_pred_knn, "Confusion Matrix for KNN with K = {}".format(best_k))
```



```
In [72]: from sklearn.metrics import log_loss
f1 = f1_score(y_test, y_pred_knn, average='weighted')
print("KNN F1-score: %.2f" % f1 )
```

KNN F1-score: 0.52

Decision Tree Classifier

Hyper-parameter tuning was applied to find the best value of D (depth), resulting in best accuracies for d=9.

```
In [64]: # HyperParameter tuning

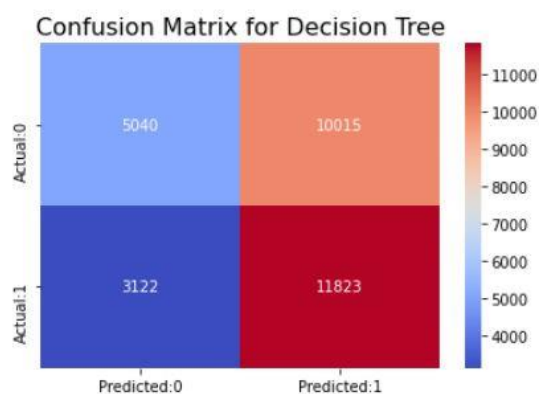
best_d = -1
best_f1 = -1

for d in range(1, 20):
    dt_cl = DecisionTreeClassifier(criterion = 'gini', max_depth = d)
    dt_cl.fit(X_train, y_train)
    y_pred = dt_cl.predict(X_test)
    f1_sc = f1_score(y_test, y_pred, average = 'weighted')
    print("k = {} ; F1 Score = {}".format(d, f1_sc))
    if f1_sc > best_f1:
        best_d = d
        best_f1 = f1_sc
print("Best d =", best_d)

k = 1 ; F1 Score = 0.5366586457160867
k = 2 ; F1 Score = 0.5366586457160867
k = 3 ; F1 Score = 0.5347373595543771
k = 4 ; F1 Score = 0.5329524402912474
k = 5 ; F1 Score = 0.5358445688286566
k = 6 ; F1 Score = 0.5360711636323899
k = 7 ; F1 Score = 0.5363145560310936
k = 8 ; F1 Score = 0.5372532198906355
k = 9 ; F1 Score = 0.5380976117678646
k = 10 ; F1 Score = 0.5372378798657411
k = 11 ; F1 Score = 0.5368783930202639
k = 12 ; F1 Score = 0.5369091285783295
k = 13 ; F1 Score = 0.5368663593048569
k = 14 ; F1 Score = 0.5369217057810111
k = 15 ; F1 Score = 0.5369368032383415
k = 16 ; F1 Score = 0.5368235875524245
k = 17 ; F1 Score = 0.5368386857928988
k = 18 ; F1 Score = 0.5368386857928988
k = 19 ; F1 Score = 0.5368940326341657
Best d = 9
```

Results were as follows:

```
In [66]: plot_conf_mat(y_test, y_pred_dt, "Confusion Matrix for Decision Tree")
```



```
In [73]: f2 = round(f1_score(y_test, y_pred_dt, average = 'weighted'), 2)
print("DT F1-score: %.2f" % f2 )

DT F1-score: 0.54
```

Random Forests

The prediction results were as follows:

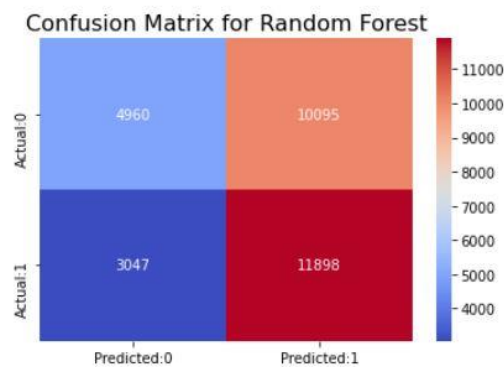
Random Forest

```
In [74]: rdf_cl = RandomForestClassifier(n_estimators=300, random_state=0)
rdf_cl.fit(X_train, y_train)
y_pred_rdf = rdf_cl.predict(X_test)

rdf_cl
```

```
Out[74]: RandomForestClassifier(n_estimators=300, random_state=0)
```

```
In [75]: plot_conf_mat(y_test, y_pred_rdf, "Confusion Matrix for Random Forest")
```

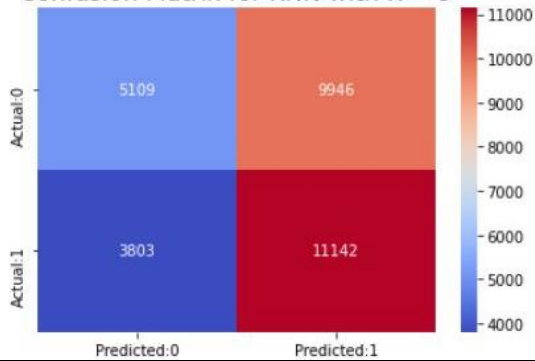
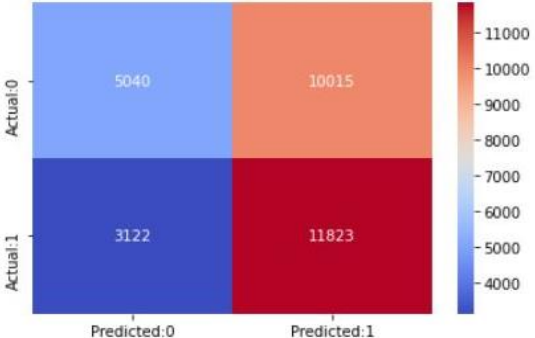
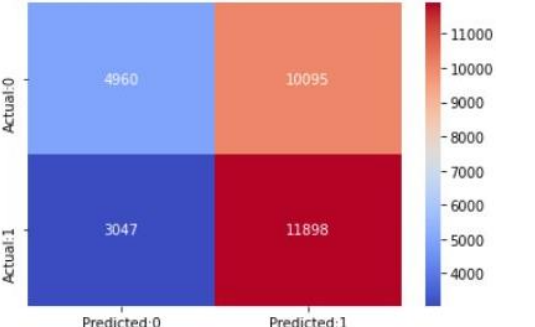


```
In [78]: f3 = round(f1_score(y_test, y_pred_rdf, average = 'weighted'), 2)
print("DT F1-score: %.2f" % f3 )
```

```
DT F1-score: 0.54
```

RESULTS & EVALUATION

Based on the results above, the summary is as follows:

ML Model	F1 Score	Confusion Matrix									
KNN	0.52	<p>Confusion Matrix for KNN with K = 9</p>  <table><thead><tr><th></th><th>Predicted:0</th><th>Predicted:1</th></tr></thead><tbody><tr><th>Actual:0</th><td>5109</td><td>9946</td></tr><tr><th>Actual:1</th><td>3803</td><td>11142</td></tr></tbody></table>		Predicted:0	Predicted:1	Actual:0	5109	9946	Actual:1	3803	11142
	Predicted:0	Predicted:1									
Actual:0	5109	9946									
Actual:1	3803	11142									
Decision Tree	0.54	<p>Confusion Matrix for Decision Tree</p>  <table><thead><tr><th></th><th>Predicted:0</th><th>Predicted:1</th></tr></thead><tbody><tr><th>Actual:0</th><td>5040</td><td>10015</td></tr><tr><th>Actual:1</th><td>3122</td><td>11823</td></tr></tbody></table>		Predicted:0	Predicted:1	Actual:0	5040	10015	Actual:1	3122	11823
	Predicted:0	Predicted:1									
Actual:0	5040	10015									
Actual:1	3122	11823									
Random Forest	0.54	<p>Confusion Matrix for Random Forest</p>  <table><thead><tr><th></th><th>Predicted:0</th><th>Predicted:1</th></tr></thead><tbody><tr><th>Actual:0</th><td>4960</td><td>10095</td></tr><tr><th>Actual:1</th><td>3047</td><td>11898</td></tr></tbody></table>		Predicted:0	Predicted:1	Actual:0	4960	10095	Actual:1	3047	11898
	Predicted:0	Predicted:1									
Actual:0	4960	10095									
Actual:1	3047	11898									

Based on the above summary, Decision Tree and Random Forest classifiers perform the best on the Seattle Car Accident Dataset.

FURTHER DISCUSSION / SCOPE OF IMPROVEMENT

My analysis demonstrates the highest F1-score is achieved using DT and Random Forest Classifiers. Particular attention to be paid towards better feature engineering and data sampling techniques to improve on this base analysis in the future.

CONCLUSION

The achieved accuracies using the ML models can be further improved upon by using better features and will contribute to the future scope of this problem.

- Explore similar algorithms like Weighted XGBoost and Naïve Bayes instead of resampling the dataset.
- Study the features in a bit more detail and derive correlation metrics for each in relation to the target variable.