IBM DATA SCIENCE PROFESSIONAL CERTIFICATE

APPLIED DATA SCIENCE CAPSTONE PROJECT

CAR ACCIDENT SEVERITY

Yazan Hassan

September 26, 2020

INTRODUCTION

The global epidemic of road crash fatalities and disabilities is gradually being recognized as a major public health concern. The first step to being informed about global road safety and to developing effective road safety interventions is to have access to facts. Approximately 1.35 million people die in road crashes each year, with the daily global death toll being about 3,700 people. An additional 20-50 million suffer non-fatal injuries every year, often resulting in long-term disabilities (https://www.asirt.org/safe-travel/road-safety-facts/).

Accidents that occur can be slight, fatal and serious. As such, a good mitigation tactic is to be able to reduce the risk of severity before accidents occur. In an effort to reduce the frequency of car collisions in a community, an algorithm can be developed to predict the severity of an accident given the weather, road and visibility conditions. When conditions are unfavourable, the model will alert drivers to remind them to be more careful or to take a safer route. The aim of this project is to build a model that can predict the severity of accidents in junctions around different types of address block and collision types using attributes such as weather and light conditions. For this model, Seattle will be used as a test case.

Target Audience and Stakeholders

The target audiences of the project are local Seattle government, police, rescue groups, and car insurance institutes. The model and its results can provide some advice for the target audiences to make insightful decisions for reducing the number of accidents and injuries in the city. For instance, this project can help local authorities discover the address type where severity of a road accident is serious due to lack of road lighting, and subsequently apply new road safety measures. The Seattle government can prevent avoidable car accidents by employing methods that alert drivers, health system, and police to remind them to be more careful in critical situations.

DATA

Data Cleaning

The dataset has information gathered on the road traffic accidents of Seattle City. 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 0 to 4. The dataset can be found here: https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv.

There are numerous problems with the initial dataset. Firstly, there are many variations in the available data points for each entry. The dataset had many empty columns that resulted in not being able to use those rows as is.

Secondly, there were many categorical features that needed to be encoded before being used in a machine learning algorithm. The model aims 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, *Yes* was given value of 1 whereas *No* and no value was given 0 for the variables *Inattention*, *Speeding* and *Under the Influence*. For lighting condition, no 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.

Thirdly, there were numerous entries for every variable which were either 'Other' or 'Unknown.' Deleting all the rows with the aforementioned entries entirely would have led to a large loss of data which is not preferred. As such, in order to deal with the issue of columns having a variation in frequency, arrays were made for each column which were encoded according to the original column and had equal proportion of elements as the original column. Then, the arrays were imposed on the original columns in the positions which had 'Other' and 'Unknown' in them. The data cleaning process led to a loss of approximately 5000 rows, which is a sufficiently small number given the initial 194,673 rows.

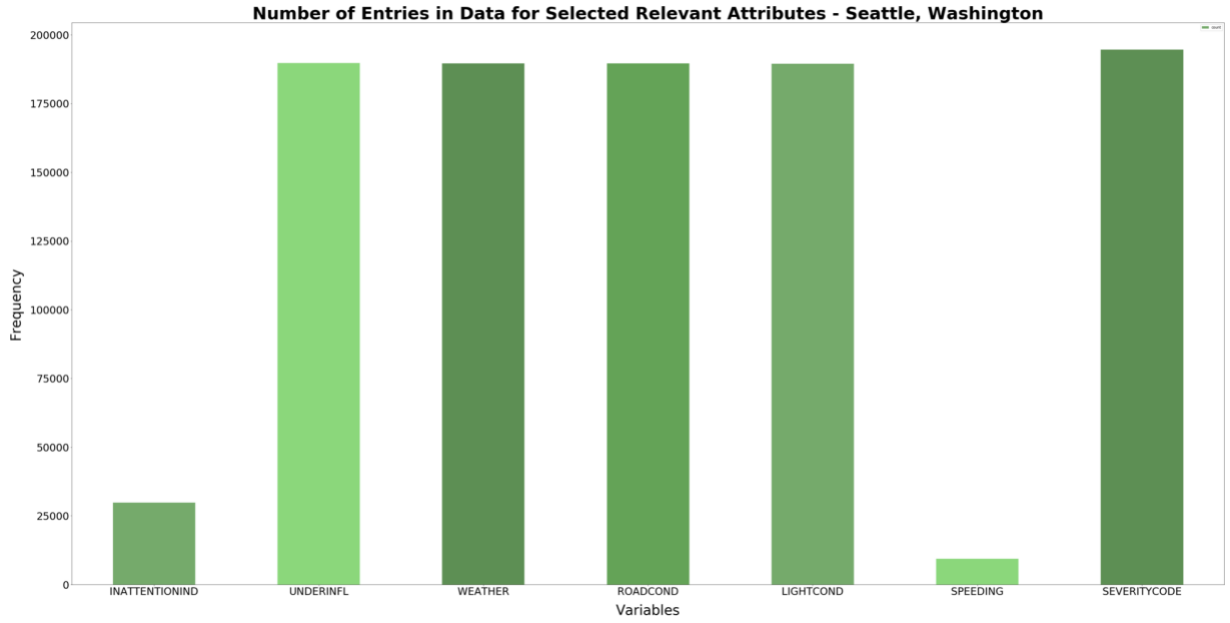
Feature Selection

The following features were selected for this machine learning project:

1. WEATHER - Weather condition during time of collision (Overcast/Rain/Clear)
2. ROADCOND - Road condition during the collision (Wet/Dry)
3. LIGHTCOND - Light conditions during the collision (Lights On/Dark with light on)
4. INATTENTIONIND - Whether or not the driver was inattentive (Y/N)
5. UNDERINFL - Whether or not the driver was under the influence (Y/N)
6. SPEEDING - Whether the car was above the speed limit at the time of collision (Y/N)

METHODOLOGY

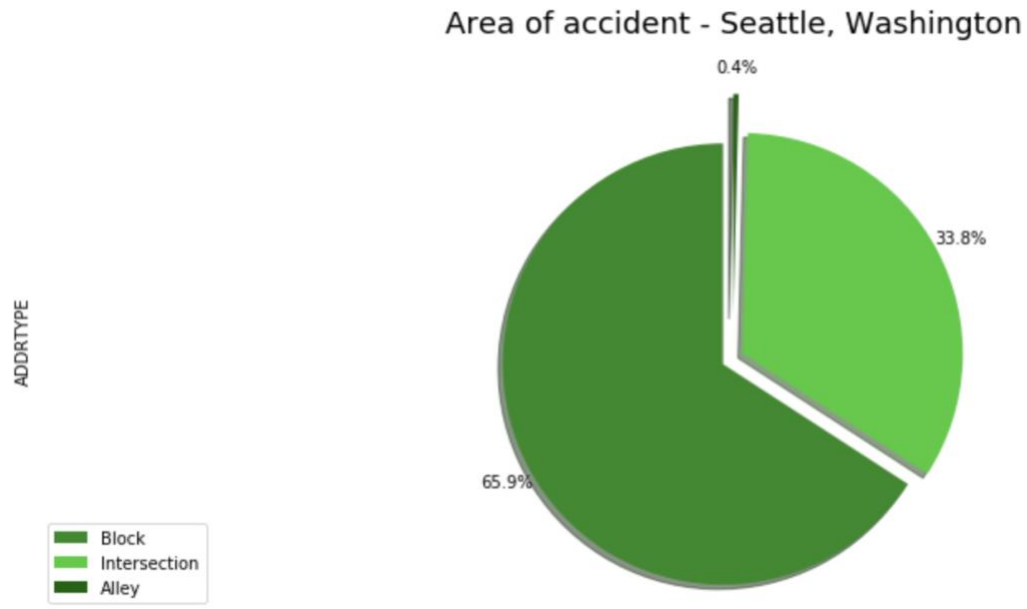
To begin, the frequencies of the different feature variables were explored. In order to deal with the issue of columns having a variation in frequency, arrays were made for each column which were encoded according to the original column and had equal proportion of elements as the original column.



Other descriptive exploratory analysis was performed to investigate the data on hand. This included descriptive statistics on the dataset:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...
count	194673.000000	189339.000000	189339.000000	194673.000000	194673.000000	194673.000000	194673	194673	192747	65070.000000	...
unique	NaN	NaN	NaN	NaN	NaN	NaN	194670	2	3	NaN	...
top	NaN	NaN	NaN	NaN	NaN	NaN	1780512	Matched	Block	NaN	...
freq	NaN	NaN	NaN	NaN	NaN	NaN	2	189786	126926	NaN	...
mean	1.298901	-122.330518	47.619543	108479.364930	141091.456350	141298.811381	NaN	NaN	NaN	37558.450576	...
std	0.457778	0.029976	0.056157	62649.722558	86634.402737	86986.542110	NaN	NaN	NaN	51745.990273	...
min	1.000000	-122.419091	47.495573	1.000000	1001.000000	1001.000000	NaN	NaN	NaN	23807.000000	...
25%	1.000000	-122.348673	47.575956	54267.000000	70383.000000	70383.000000	NaN	NaN	NaN	28667.000000	...
50%	1.000000	-122.330224	47.615369	106912.000000	123363.000000	123363.000000	NaN	NaN	NaN	29973.000000	...
75%	2.000000	-122.311937	47.663664	162272.000000	203319.000000	203459.000000	NaN	NaN	NaN	33973.000000	...
max	2.000000	-122.238949	47.734142	219547.000000	331454.000000	332954.000000	NaN	NaN	NaN	757580.000000	...

Additionally, other visual exploratory analysis was performed to understand the relationship of various variables to the independent variable. For example, the area of accidents was explored to see what address type was more likely to result in an accident in Seattle.



After performing some data cleaning, the data was analyzed to see the distribution of the feature variables. Some of the distribution of the various features can be found below:

```
feature_df['LIGHTCOND'].value_counts()
```

```
0    125113
1     61212
2      3012
Name: LIGHTCOND, dtype: int64
```

```
feature_df['ROADCOND'].value_counts()
```

```
0    134254
2     53924
1      1159
Name: ROADCOND, dtype: int64
```

```
feature_df['WEATHER'].value_counts()
```

```
0    121159
3     37252
1     30218
2       708
Name: WEATHER, dtype: int64
```

Ensuring that the target variable is balanced is also vital to ensure that the model doesn't form a bias for any of the outcomes. When the target variable was explored, it showed that there was almost a 1:2 ratio in favour of property damage when compared to physical injury. As such, we needed to balance the dataset to reduce inaccuracies in our machine learning algorithms. To do this, SMOTE was utilised from the *imblearn* library to balance the target variable in equal proportions. This removes biases from classification models as the model is trained on equal number of instances of both elements.

The data was also prepared for model evaluation. Model evaluation tells us how our model performs in the real world. In-sample evaluation tells how well the model fits the data already given to train it. It does not give an estimate of how well the trained model

can predict new data. The solution is to this is to split the data up and use the in-sample data or training data to train the model. The rest of the data, called test data, is used as out-of-sample data. This data is then used to approximate how the model performs in the real world. Separating data into training and testing sets is an important part of model evaluation. The data was split into 75% training data and 25% testing data.

MACHINE LEARNING MODELS

The machine learning algorithm tested are: Decision Tree algorithm, Logistic Regression, and k-Nearest Neighbour (kNN). Classification algorithms were chosen as they attempt to learn the relationship between a set of feature variables and a target variable of interest, which is precisely what is needed for this project. The target attribute in classification is a categorical variable with discrete values, which in this model is SEVERITYCODE. Given a set of training data points along with the target labels, classification determines the class label for an unlabeled test case.

Decision trees are built by splitting the training set into distinct nodes, where one node contains all or most of one category of the data. Decision trees are about testing an attribute and branching the cases based on the result of the test. Each internal node corresponds to a test, and each branch corresponds to a result of the test, and each leaf node assigns a variable to a class. Decision trees are built using recursive partitioning to classify the data.

Logistic regression is a statistical and machine learning technique for classifying records of a dataset based on the values of the input fields. Logistic regression is analogous to linear regression but tries to predict a categorical or discrete target field instead of a numeric one. In linear regression, we predict a continuous value of variables such as the price of a house, blood pressure of a patient, or fuel consumption of a car. But in logistic regression, we predict a variable which is binary such as yes/no, true/false, successful or not successful, and so on, all of which can be coded as zero or one.

The **K-Nearest Neighbours** algorithm is a classification algorithm that takes labeled points and uses them to learn how to label other points. This algorithm classifies variables based on their similarity to other cases. In K-Nearest Neighbours, data points that are near each other are said to be neighbors. K-Nearest Neighbours is based on the paradigm that similar cases with the same class labels are near each other. Thus, the distance between two cases is a measure of their dissimilarity.

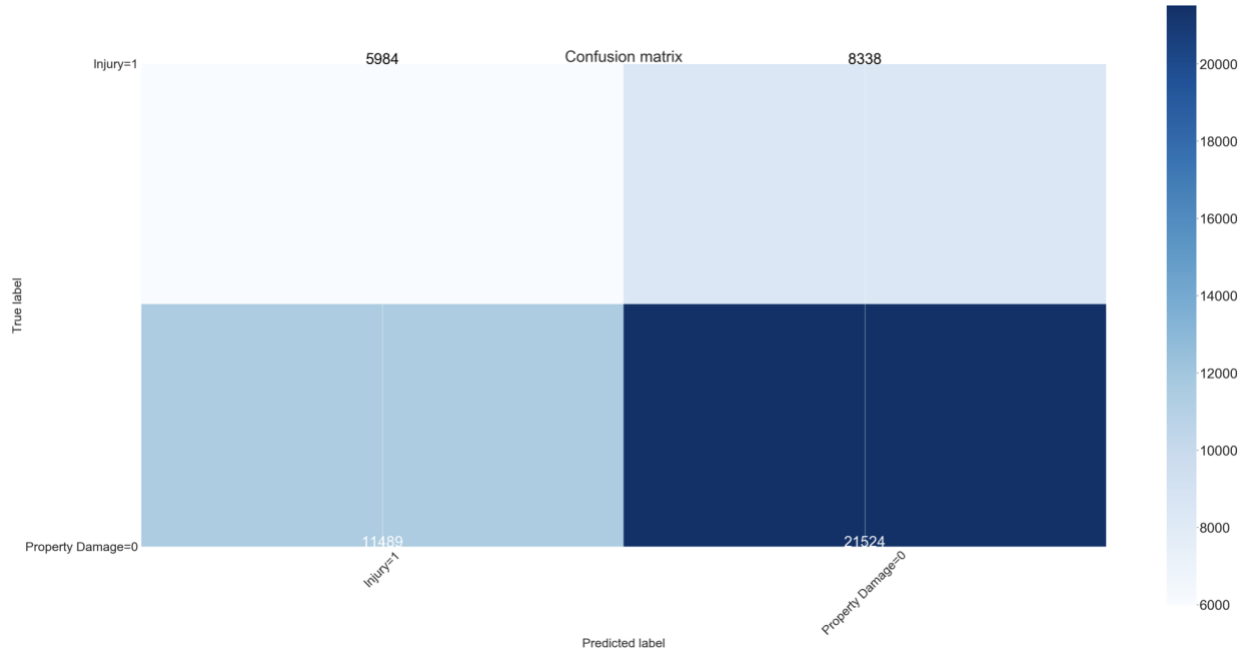
As a note, a Support Vector Machine (SVM) algorithm was not tested as it is not appropriate in this case. SVMs are inaccurate for large datasets and works best when the dataset is filled with text and images. Considering the nature and magnitude of the dataset used, an SVM analysis was disregarded.

RESULTS

Decision Tree Algorithm

The scikit-learn library was used to construct the decision tree classification algorithm. The criterion chosen for the classifier is “entropy” with a maximum depth of 6. The results of the algorithm are shown below.

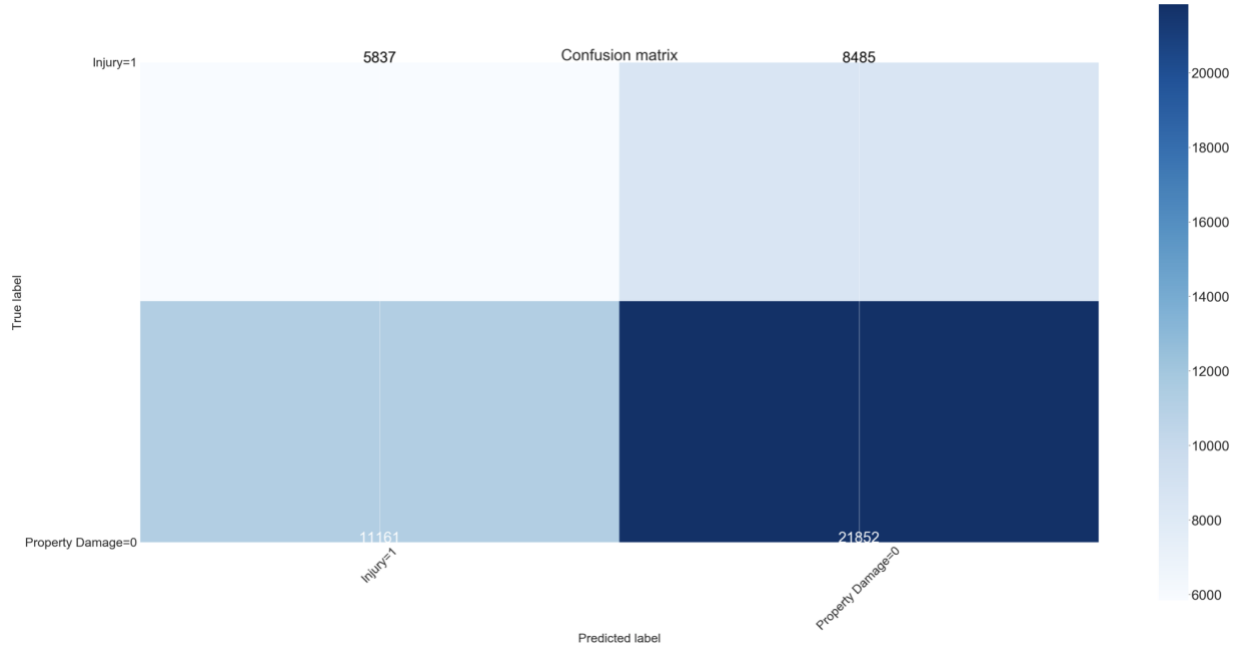
	precision	recall	f1-score	support
0	0.65	0.72	0.68	29862
1	0.42	0.34	0.38	17473
accuracy			0.58	47335
macro avg	0.53	0.53	0.53	47335
weighted avg	0.57	0.58	0.57	47335



Logistic Regression

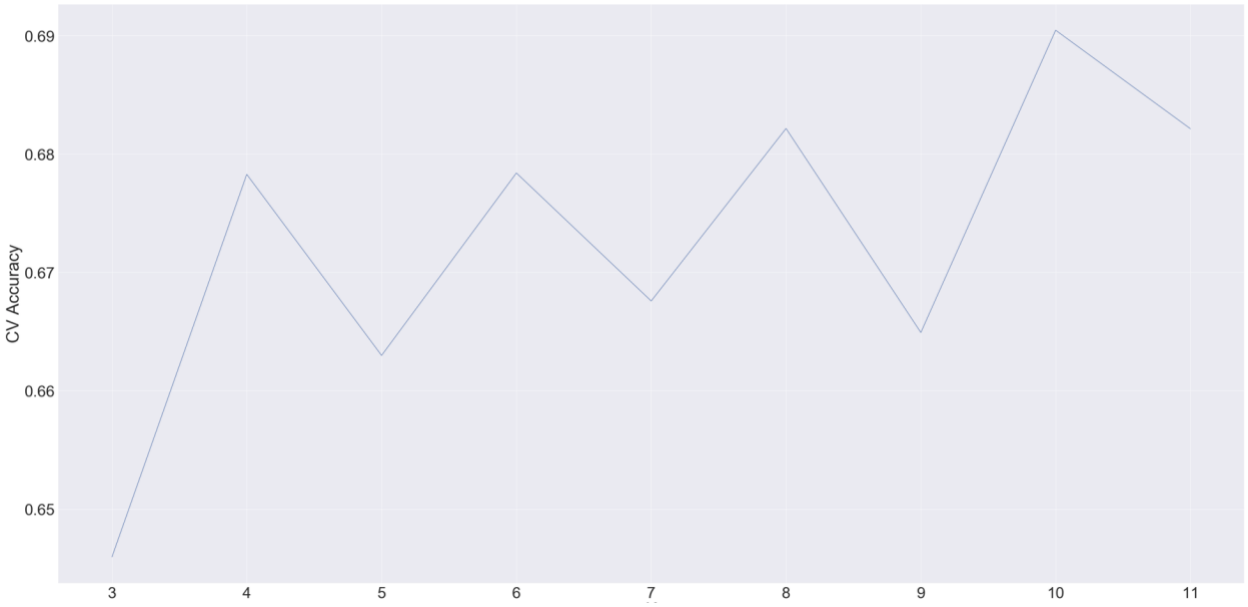
The scikit-learn library was used to run the logistic regression algorithm. The *liblinear* solver was used with a 0.01 C regularization strength.

	precision	recall	f1-score	support
0	0.72	0.66	0.69	33013
1	0.34	0.41	0.37	14322
accuracy			0.58	47335
macro avg	0.53	0.53	0.53	47335
weighted avg	0.61	0.58	0.59	47335



k-Nearest Neighbour (kNN)

The scikit-learn library was used to run the kNN classification algorithm. Through an iteration loop testing the accuracy of the model with various K values, the best K value was determined to be 10.




```
Best K is : 10 | Cross validation Accuracy : 0.690461899717634
```

```
In [71]: KNNf1score =f1_score(y_train, yknn_pred, average='weighted')
         KNNf1score

Out[71]: 0.555808952766952
```

DISCUSSION

	Accuracy	F1 Score
Decision Tree	0.58	0.53 (0: 0.68; 1: 0.38)
Logistic Regression	0.58	0.53 (0: 0.69; 1: 0.37)
kNN	0.69	0.56

Two main metrics were used to evaluate the best performing model: accuracy of the models and the F1 scores. The accuracy metric takes into account training accuracy and out-of-sample accuracy. Training accuracy is the percentage of correct predictions that the model makes when using the test dataset. Out-of-sample accuracy is the percentage of correct predictions that the model makes on data that the model has not been trained on.

Additionally, classification reports were compiled to further assess each model. The classification reports include *precision*, *recall*, and *F1 scores*. Precision is a measure of the accuracy, provided that a class label has been predicted. It is defined by: Precision =

True Positive / (True Positive + False Positive). Recall is the true positive rate. It refers to the percentage of total relevant results correctly classified by the algorithm. The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (which represents perfect precision and recall) and its worst at 0. It is a good way to show that a classifier has a good value for both recall and precision. As such, F1 scores were used to capture both precision and recall.

Another way of looking at accuracy of classifiers is to look at confusion matrixes.
We can interpret the confusion matrixes numbers as the count of true positives, false negatives, true negatives, and false positives.

Looking at the accuracy and f1 scores of the three different classification algorithms above, it is evident that the kNN algorithm is the best performing algorithm. The kNN algorithm with a K value of 10 has the best cross validation accuracy, at about 0.69, as well as the highest f1 score at 0.56.

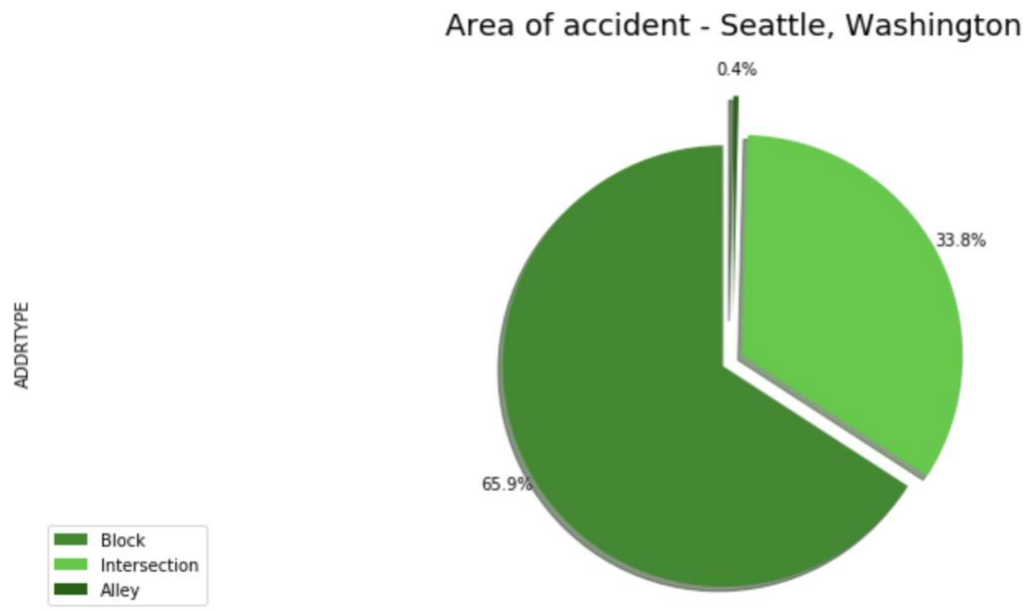
CONCLUSION AND RECOMMENDATIONS

When comparing the F1 scores, accuracy, confusion matrixes, and classification reports of the three different algorithms, we are able to better determine which algorithms have performed better in achieving the goal of the project. Evident by the various scores, it is clear that k-Nearest Neighbour algorithm is the most suitable for this project.

That being said, the metrics do show that it isn't a very high-performing model especially compared to benchmarks within the industry. The model could be improved by some of more favourable conditions, such as:

- Less missing values within the dataset.
- Larger datapoints and records of the accidents in Seattle.
- A more balanced dataset with respect to the target variable.
- More features that could impact car accidents.

Multiple insights can be derived from this project for the stakeholders involved. The Seattle municipal government can assess areas in the city with high accident rates and focus on implementing preventative measures to try to mitigate the accidents. For example, development projects could be launched to target improved light conditions in high-risk areas.



Since almost all of the accidents in Seattle happen in either a block or an intersection, the municipal government can develop initiatives to mitigate unfavourable conditions in those areas. Those can include installing better safety signs to notify drivers of hazards, as well as increase investment in lightening and road conditions in high-risk areas.

Additionally, individual drivers can use this data to be cognizant of areas and conditions that have a higher risk of getting into a car accident. Specific importance can be given to mitigate conditions and areas that cause severe car accidents.