Car Accident Severity Analysis Capstone Project

1. Introduction - Business Understanding

1.1 Business problem –

Today world has changed and has higher usage of automobiles for moving from point to point. The movement of vehicles/ automobiles is influenced by many factors like local traffic, weather, inclement conditions, locality, terrain, roads etc. During movement of vehicles different accidents happen which can be avoided if certain precautions are taken by the person who is driving the automobile.

Through this project we would help the agency and the community to avoid car accidents by understanding the different causes for car accidents and provide solutions to overcome accidents and in turn reduce casualties and expenses incurred on people and vehicles. This will result in agencies taking necessary measures, early warnings, road signs, check posts, deployment of emergency vehicles, road barriers etc. to avoid car accidents.

We will be using machine learning algorithms of data science to identify the parameters which cause accidents and also provide solutions on the measurements to be taken to avoid these accidents

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.

The Seattle government is going to prevent avoidable car accidents by employing methods that alert drivers, health system, and police to remind them to be more careful in critical situations.
In most cases, not paying enough attention during driving, abusing drugs and alcohol or driving at very high speed are the main causes of occurring accidents that can be prevented by enacting harsher regulations. Besides the aforementioned reasons, 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 last but not least, car insurance institutes. The model and its results are going to provide some advice for the target audience to make insightful decisions for reducing the number of accidents and injuries for the city.

2. Data -

2.1 Data description –

The data is obtained from the Seattle department of transportation (SDOT) which identifies different parameters when there have been accidents. This can be easily obtained from the SDOT website in CSV files. The data presented here is from February 2006 till May

2020. The data represents various parameters like the street, date, severity, severity, place of accident and several others which are cause of car accidents

2.2 Source of the data – Seattle Department of Transport database

2.3 Data interpretation –

The current data consists of **38 columns** and **194673 rows** of data. The different columns which have been captured are provided below along with data types and descriptions. Descriptions are provided in the attachment – Metadata.pdf

SEVERITYCODE
X
Y
OBJECTID
INCKEY
COLDETKEY
REPORTNO
STATUS
ADDRTYPE
INTKEY
LOCATION
EXCEPTRSNCODE
EXCEPTRSNDESC
SEVERITYCODE
SEVERITYDESC
COLLISIONTYPE
PERSONCOUNT
PEDCOUNT
PEDCYLCOUNT
VEHCOUNT
INCDATE
INCDTTM
JUNCTIONTYPE
SDOT_COLCODE
SDOT_COLDESC

INATTENTIONIND
UNDERINFL
WEATHER
ROADCOND
LIGHTCOND
PEDROWNOTGRNT
SDOTCOLNUM
SPEEDING
ST_COLCODE
ST_COLDESC
SEGLANEKEY
CROSSWALKKEY
HITPARKEDCAR

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

5 rows × 38 columns

The dependent variable for our study is **“SEVERITYCODE”** and has measurements for severity of an accident on a **scale of 0 to 3**. It contains several numbers which are as follows –

Severity Codes are as follows –

- **0**: unknown
- **1**: Property Damage
- **2**: Injury
- **2b**: serious Injury
- **3**: fatality

```
SEVERITYCODE          int64
X                   float64
Y                   float64
OBJECTID              int64
INCKEY                int64
COLDETKEY             int64
REPORTNO             object
STATUS               object
ADDRTYPE             object
INTKEY              float64
LOCATION             object
EXCEPTRSNCODE        object
EXCEPTRSNDESC        object
SEVERITYCODE.1        int64
SEVERITYDESC         object
COLLISIONTYPE        object
PERSONCOUNT           int64
PEDCOUNT              int64
PEDCYLCOUNT           int64
VEHCOUNT              int64
INCDATE              object
INCDTTM              object
JUNCTIONTYPE         object
SDOT_COLCODE          int64
SDOT_COLDESC         object
INATTENTIONIND       object
UNDERINFL            object
WEATHER              object
ROADCOND             object
LIGHTCOND            object
PEDROWNOTGRNT        object
SDOTCOLNUM          float64
SPEEDING             object
ST_COLCODE           object
ST_COLDESC           object
SEGLANEKEY            int64
CROSSWALKKEY          int64
HITPARKEDCAR         object
dtype: object
```

Here we show the various statistical parameters for the data

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	INTKEY	SEVERITYCODE.1	PERSONCOUNT	PEDCOUNT
count	194673.000000	189339.000000	189339.000000	194673.000000	194673.000000	194673.000000	65070.000000	194673.000000	194673.000000	194673.000000
mean	1.298901	-122.330518	47.619543	108479.364930	141091.456350	141298.811381	37558.450576	1.298901	2.444427	0.037139
std	0.457778	0.029976	0.056157	62649.722558	86634.402737	86986.542110	51745.990273	0.457778	1.345929	0.198150
min	1.000000	-122.419091	47.495573	1.000000	1001.000000	1001.000000	23807.000000	1.000000	0.000000	0.000000
25%	1.000000	-122.348673	47.575956	54267.000000	70383.000000	70383.000000	28667.000000	1.000000	2.000000	0.000000
50%	1.000000	-122.330224	47.615369	106912.000000	123363.000000	123363.000000	29973.000000	1.000000	2.000000	0.000000
75%	2.000000	-122.311937	47.663664	162272.000000	203319.000000	203459.000000	33973.000000	2.000000	3.000000	0.000000
max	2.000000	-122.238949	47.734142	219547.000000	331454.000000	332954.000000	757580.000000	2.000000	81.000000	6.000000

Further we will have to see if the data consists of any NaN and null values which may not be contributing to the solution analysis. We will have to do some data cleansing. There may be certain records which may not be necessary for our problem which can be filtered so that we can obtain more accurate results but while filtering we should be careful not to lose out any data as studied at early stages of our course. Also, the data records of **194673** have at present hold only values of ‘1’ or ‘2’ under the column **“SEVERITYCODE”.** The data is obtained in **CSV format** and can be easily uploaded through the “read_csv” function of python

2.3 Data Pre-processing –

The dataset in its original form is not completely fit for data analysis. There are many columns which may not be relevant and may need to be dropped or discarded. To prepare the data, first, we need to drop the non-relevant columns. Also we notice that most of the data types are of type object which need to be converted into numerical data types

The following will be columns for our data analysis and Machine Learning models!

SEVERITYCODE X Y OBJECTID INCKEY COLDETKEY
REPORTNO STATUS ADDRTYPE INTKEY LOCATION
EXCEPTRSNCODE EXCEPTRSNDESC SEVERITYCODE SEVERITYDESC
COLLISIONTYPE PERSONCOUNT PEDCOUNT PEDCYLCOUNT
VEHCOUNT INCDATE INCDTTM JUNCTIONTYPE SDOT_COLCODE
SDOT_COLDESC INATTENTIONIND UNDERINFL WEATHER
ROADCOND LIGHTCOND PEDROWNOTGRNT SDOTCOLNUM
SPEEDING ST_COLCODE ST_COLDESC SEGLANEKEY
CROSSWALKKEY HITPARKEDCAR

we will check the data types of these new columns through the Python DataFrame and move ahead with the data analysis further.

```
In [42]: # Variable for dropping of unwanted columns
coldat = 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', 'PEDC
              'PERSONCOUNT', 'VEHCOUNT', 'COLLISIONTYPE', 'SPEEDING', 'UNDERINFL', 'INATTENTIONIND'])

# We will be labelling the different varibales which we will use for analysis
#we will only consider 3 variables for our ease - WEATHER, ROAD, LIGHT

# Convert column to a category variable
# We will assign a new varibale for all our Categorical Attributes for our analysis
coldat["ROADCOND"] = coldat["ROADCOND"].astype('category')
coldat["WEATHER"] = coldat["WEATHER"].astype('category')
coldat["LIGHTCOND"] = coldat["LIGHTCOND"].astype('category')

coldat["ROADCOND_CATG"] = coldat["ROADCOND"].cat.codes
coldat["WEATHER_CATG"] = coldat["WEATHER"].cat.codes
coldat["LIGHTCOND_CATG"] = coldat["LIGHTCOND"].cat.codes

#here we get all variables which we will be considering for our purpose
coldat.dtypes

Out[42]: SEVERITYCODE        int64
         WEATHER          category
         ROADCOND         category
         LIGHTCOND        category
         ROADCOND_CATG        int8
         WEATHER_CATG         int8
         LIGHTCOND_CATG       int8
         dtype: object
```

2.4 Balancing the Dataset –

When we keenly observe we see that the target variable **SEVERITYCODE** only **43% balanced** (out of a total of 194673 datapoints we have code1 = 58188 and code 2 = 136485, hence code1/code2; 58188/136485 42.6%). This can skew our data points and provide wrong results. Hence, we will have to first obtain a balanced data set which can done through simple statistical techniques like downsampling. This will be done for **classs1** and **class2**(please refer Metadata.pdf for classification)

After we have used statistical technique, we have now obtained a balanced dataset. We convert the data into a Categorical Variables as shown below

	SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND	ROADCOND_CATG	WEATHER_CATG	LIGHTCOND_CATG
0	2	Overcast	Wet	Daylight	8	4	5
1	1	Raining	Wet	Dark - Street Lights On	8	6	2
2	1	Overcast	Dry	Daylight	0	4	5
3	1	Clear	Dry	Daylight	0	1	5
4	2	Raining	Wet	Daylight	8	6	5

We will be balancing the data which will be required for analysis

```
#we will consider the count of Severity Code to ensure that we have a balanced data
coldat["SEVERITYCODE"].value_counts()
```

```
44]: 1    136485
     2     58188
     Name: SEVERITYCODE, dtype: int64
```

Data statistics for ROADCOND variable

```
coldat["ROADCOND"].value_counts()
```

```
45]: Dry               124510
     Wet                47474
     Unknown            15078
     Ice                 1209
     Snow/Slush          1004
     Other                132
     Standing Water       115
     Sand/Mud/Dirt         75
     Oil                   64
     Name: ROADCOND, dtype: int64
```

Data statistics for WEATHER variable

```
coldat["WEATHER"].value_counts()
```

```
46]: Clear                       111135
     Raining                      33145
     Overcast                     27714
     Unknown                      15091
     Snowing                        907
     Other                          832
     Fog/Smog/Smoke                 569
     Sleet/Hail/Freezing Rain       113
     Blowing Sand/Dirt               56
     Severe Crosswind                25
     Partly Cloudy                    5
     Name: WEATHER, dtype: int64
```

Data Statistics for LIGHTCOND variable

```
coldat["LIGHTCOND"].value_counts()
```

```
[47]: Daylight                    116137
      Dark - Street Lights On      48507
      Unknown                      13473
      Dusk                          5902
      Dawn                          2502
      Dark - No Street Lights       1537
      Dark - Street Lights Off      1199
      Other                          235
      Dark - Unknown Lighting         11
      Name: LIGHTCOND, dtype: int64
```

Finally balanced the dataset

```
from sklearn.utils import resample
```

```
# We will now balance the dataset to ensure we have equal representation
# Seperate Severity class 1 and class2
coldat_class1 = coldat[coldat.SEVERITYCODE==1]
coldat_class2 = coldat[coldat.SEVERITYCODE==2]

#Resample class1 - property damage
coldat_class1_resampled = resample(coldat_class1,
                                   replace=False,
                                   n_samples=58188,
                                   random_state=123)

# Combine class2 with downsample class1 data
#Now the balanced data is stored in a combined variable coldata_bal
coldat_bal = pd.concat([coldat_class1_resampled, coldat_class2])

# Display new class counts through new balanced variable
coldat_bal.SEVERITYCODE.value_counts()
```

```
[49]: 2    58188
      1    58188
      Name: SEVERITYCODE, dtype: int64
```

Display of final balanced data set

	SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND	ROADCOND_CATG	WEATHER_CATG	LIGHTCOND_CATG
25055	1	Raining	Wet	Dark - Street Lights On	8	6	2
65280	1	Clear	Dry	Daylight	0	1	5
86292	1	Unknown	Unknown	Unknown	7	10	8
155111	1	Clear	Dry	Daylight	0	1	5
64598	1	Clear	Dry	Daylight	0	1	5
119954	1	Clear	Dry	Daylight	0	1	5
64063	1	Clear	Dry	Daylight	0	1	5
105379	1	Clear	Dry	Daylight	0	1	5
181211	1	NaN	NaN	NaN	-1	-1	-1
187708	1	Clear	Dry	Daylight	0	1	5

3. Methodology section -

3.1 Main Components –

The above problem, Car accident severity, will be solved using Python Language and the various libraries available within its framework. We will invoke appropriate libraries/ packages like Pandas, NumPy, SciKitlearn.

```
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
```

We will be using Jupyter Notebook for writing all out code and also will be used for documentation purposes where needed for a clear understanding of the data analysis code. We will use github repository for version control and publishing purposes

We will be using several inbuilt functions available within the packages of Python and use them to obtain several outputs for interpretation and analysis.

3.2 Exploratory Data Analysis –

We will be using a CSV file which is uploaded to the local folder. This file will be read through inbuilt functions and the attributes of the file will be obtained. The attributes will be checked for their data type correctness, blank values, non-relevant data. We will convert data types as per our requirement for analysis. We don't have GeoJSON file for obtaining the Seattle map for further analysis. We are using Data-Collisions.csv file which has been obtained from the course.

```
df = pd.read_csv("https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv")
df.head()
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/IPython/core/interactiveshell.py:3020: DtypeWarning: Columns (33) have mixed type
s. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

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

5 rows × 38 columns

3.3 Inferences –

We will be splitting the data into training and testing in the ratio of **30:70** and accordingly apply various statistical tests.

```
import numpy as np
X = np.asarray(coldat_bal[['ROADCOND_CATG', 'WEATHER_CATG', 'LIGHTCOND_CATG']])
X[0:5]
```

```
23]:  array([[ 8,  6,  2],
             [ 0,  1,  5],
             [ 7, 10,  8],
             [ 0,  1,  5],
             [ 0,  1,  5]], dtype=int8)
```

```
len(X)
```

```
24]:  116376
```

```
import numpy as np
y = np.asarray(coldat_bal['SEVERITYCODE'])
y[0:5]
```

```
25]:  array([1, 1, 1, 1, 1])
```

```
len(y)
```

```
26]:  116376
```

```
from sklearn.preprocessing import StandardScaler
X = preprocessing.StandardScaler().fit(X).transform(X)
X[0:5]
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/utils/validatior
was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/utils/validatior
was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

```
27]:  array([[ 1.52797946,  1.15236718, -1.21648407],
             [-0.67084969, -0.67488   ,  0.42978835],
             [ 1.25312582,  2.61416492,  2.07606076],
             [-0.67084969, -0.67488   ,  0.42978835],
             [-0.67084969, -0.67488   ,  0.42978835]])
```

```
X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size=0.3, random_state=42)
```

Result is as follows –

```
print ('Training set:', X_train.shape,  y_train.shape)
print ('Testing set:', X_test.shape,  y_test.shape)

    Training set: (81463, 3) (81463,)
    Testing set: (34913, 3) (34913,)
```

So we will be left with 81463 training set and 34913 test set

3.4 Data Techniques -

We will use the following Machine Learning models:

K-Nearest Neighbor (KNN)

KNN will help us predict the severity code of an outcome by finding the most similar to data point within k distance.

K-Nearest Neighbors (KNN)

```
from sklearn.neighbors import KNeighborsClassifier
k = 25
```

```
#Train Model & Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh

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

```
7]:  array([1, 1, 1, 1, 1])
```

Decision Tree

A decision tree model gives us a layout of all possible outcomes so we can fully analyze the consequences of a decision. It context, the decision tree observes all possible outcomes of different weather conditions.

Decision Tree

```
# Building the Decision Tree
from sklearn.tree import DecisionTreeClassifier
colDataTree = DecisionTreeClassifier(criterion="entropy", max_depth = 7)
colDataTree
colDataTree.fit(X_train,y_train)
```

```
48]:  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')
```

```
# Train Model & Predict
dtyhat = colDataTree.predict(X_test)
print (dtyhat [0:5])
print (y_test [0:5])
```

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

Logistic Regression

Because our dataset only provides us with two severity code outcomes, our model will only predict one of those two classes. This makes our data binary, which is perfect to use with logistic regression.

Logistic Regression

```
# Building the Linear regression Model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
lr = LogisticRegression(C=0.03, solver='liblinear').fit(X_train,y_train)
lr
```

```
50]:  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)
```

```
# Train Model & Predictor
lryhat = lr.predict(X_test)
print(lryhat)

yhat_prob = lr.predict_proba(X_test)
print(yhat_prob)
```

```
[2 1 2 ... 2 2 1]
[[0.40364293 0.59635707]
 [0.53529771 0.46470229]
 [0.46743605 0.53256395]
 ...
 [0.46293233 0.53706767]
 [0.46743605 0.53256395]
 [0.67878612 0.32121388]]
```

Random Forest Classification

```
Classification of different models Random Forest Classification
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', 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, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

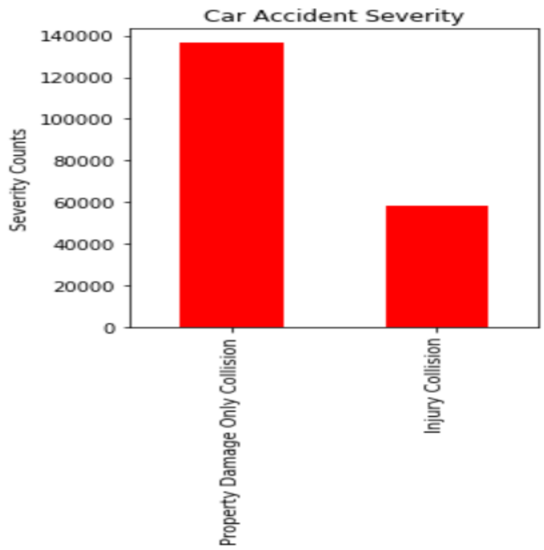
4. Results section

4.1 Analysis results –

Now after performing all our analysis using various data types, functions and packages we will discuss the results obtained from different models to check for the accuracy of these models.

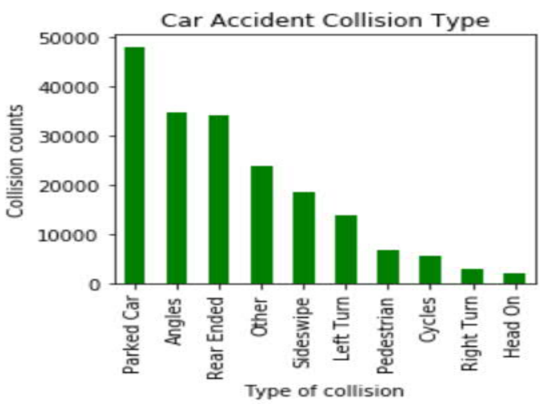
Severity Counts-

	Severity Counts	Severity Percentage
Property Damage Only Collision	136485	70.109877
Injury Collision	58188	29.890123



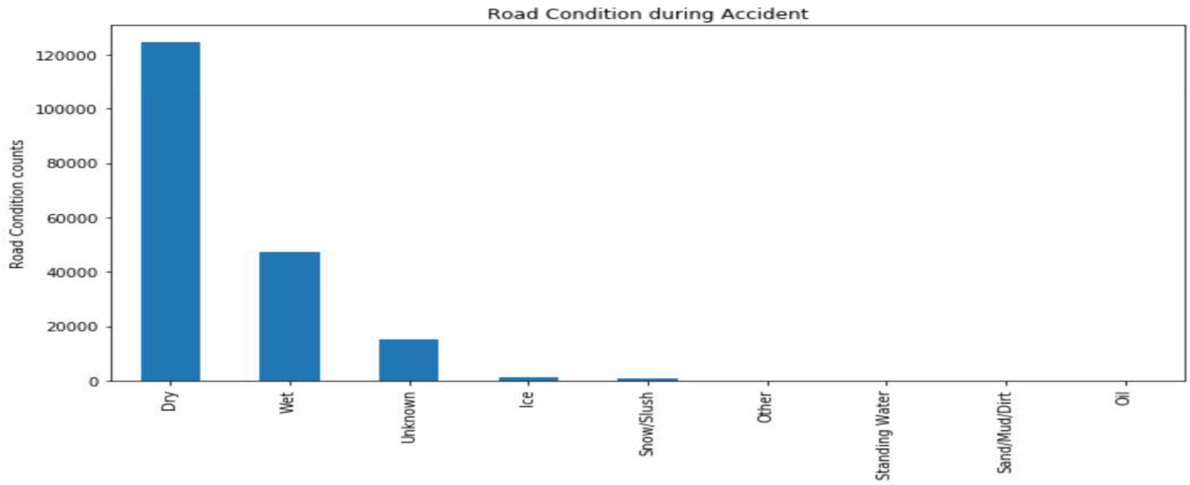
Collision count –

	Collision Counts	Percentage of collisions
Parked Car	47987	25.287060
Angles	34674	18.271688
Rear Ended	34090	17.963946
Other	23703	12.490449
Sideswipe	18609	9.806133
Left Turn	13703	7.220884
Pedestrian	6608	3.482128
Cycles	5415	2.853469
Right Turn	2956	1.557683
Head On	2024	1.066560



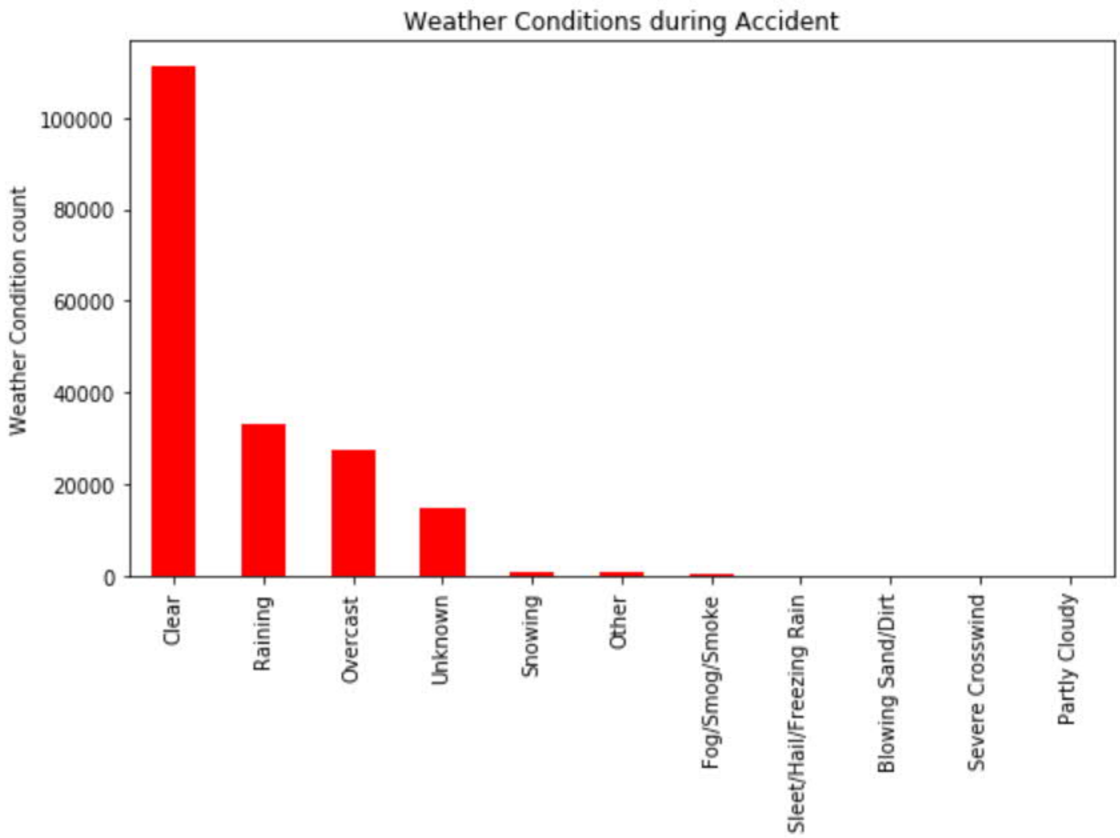
Road Condition Count –

	Road Condition Counts	Road Condition Percentage
Dry	124510	65.648710
Wet	47474	25.030976
Unknown	15078	7.949974
Ice	1209	0.637453
Snow/Slush	1004	0.529366
Other	132	0.069598
Standing Water	115	0.060635
Sand/Mud/Dirt	75	0.039544
Oil	64	0.033744



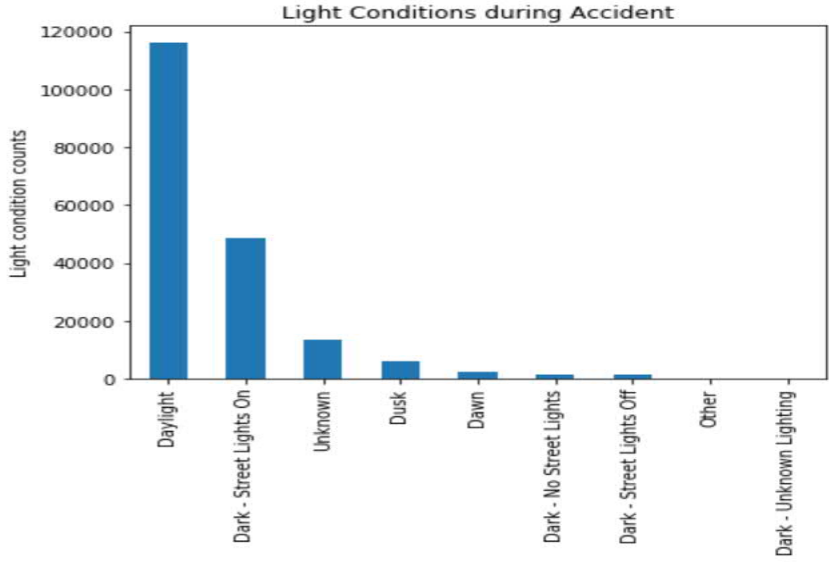
Weather condition count –

	Weather Condition Counts	Weather condition Percentage
Clear	111135	58.617980
Raining	33145	17.482278
Overcast	27714	14.617705
Unknown	15091	7.959724
Snowing	907	0.478396
Other	832	0.438837
Fog/Smog/Smoke	569	0.300118
Sleet/Hail/Freezing Rain	113	0.059602
Blowing Sand/Dirt	56	0.029537
Severe Crosswind	25	0.013186
Partly Cloudy	5	0.002637



Light Condition Count –

	Counts of Light Condition	Light Condition Percentage
Daylight	116137	61.285046
Dark - Street Lights On	48507	25.596956
Unknown	13473	7.109650
Dusk	5902	3.114463
Dawn	2502	1.320296
Dark - No Street Lights	1537	0.811069
Dark - Street Lights Off	1199	0.632708
Other	235	0.124009
Dark - Unknown Lighting	11	0.005805



Here you will see the different parameters for the ML methods used –

```
Classification of different models K-Nearest Neighbors
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=3, p=2,
           weights='uniform')
              precision    recall  f1-score   support

           1       0.63      0.26      0.37     17619
           2       0.53      0.84      0.65     17294

   micro avg       0.55      0.55      0.55     34913
   macro avg       0.58      0.55      0.51     34913
weighted avg       0.58      0.55      0.51     34913
```

```
Classification of different models Linear Regression
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)
              precision    recall  f1-score   support

           1       0.54      0.35      0.43     17619
           2       0.51      0.70      0.59     17294

   micro avg       0.52      0.52      0.52     34913
   macro avg       0.53      0.53      0.51     34913
weighted avg       0.53      0.52      0.51     34913
```

```
Classification of different models Decision Tree
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')
              precision    recall  f1-score   support

           1       0.63      0.33      0.43     17619
           2       0.54      0.80      0.64     17294

   micro avg       0.56      0.56      0.56     34913
   macro avg       0.58      0.56      0.54     34913
weighted avg       0.58      0.56      0.54     34913
```

```
Classification of different models Random Forest Classification
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', 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, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
              precision    recall  f1-score   support

           1       0.63      0.31      0.42     17619
           2       0.54      0.81      0.65     17294

   micro avg       0.56      0.56      0.56     34913
   macro avg       0.58      0.56      0.53     34913
weighted avg       0.58      0.56      0.53     34913
```

5. Discussion section

5.1 Observations –

Initial dataset was very raw in nature where we had to change categorical data of type 'object' into int8 type. Machine Learning algorithms normally accept a data type which is of numerical data type
Later we went onto look at SEVERITYCODE which had uneven data points in the ratio of 43:57 and we had to balance the dataset. we did apply downsampling the majority class (Class 2) SciKitlearn resample tool. We did a downsampling of the minority class to match to the majority class
After the analysis of the data points like SEVERITYCODE, WEATHER, LIGHTCOND, ROADCOND etc. we also cleaned the data and balanced the data to be fed through three ML models; K-Nearest Neighbor, Decision Tree and Logistic Regression, Random Forest Classification. Logiscti regression was the best suitable model for the current data
We used the Evaluation metrics like Jaccard Similarity Score, f-1 score and Logloss for logistic regression used to test the accuracy of our models were . Choosing different k, max depth and hyperparameter values helped to improve our accuracy to be the best possible.

Here in this section we will discuss on the different parameters used for evaluation of the machine learning models which were used for our analysis

We used the following evaluation techniques to measure

1) Jaccard Similarity Score
2) F1 Score
3) Logloss

K-Nearest Neighbor Evaluation

```
# Jaccard Similarity Score
jaccard_similarity_score(y_test, Kyhat)
```

11]: 0.5237017729785467

```
# F1-Score
f1_score(y_test, Kyhat, average='macro')
```

12]: 0.5196155093297656

Decision Tree Evaluation

Decision Tree Evaluation

```
# Jaccard Similarity Score
jaccard_similarity_score(y_test, dtyhat)
```

.3]: 0.5626843869045914

```
# F1-Score
f1_score(y_test, dtyhat, average='macro')
```

.4]: 0.5385207275454998

Logistic Regression Evaluation

Logistic Regression Evaluation

```
# Jaccard Similarity Score
jaccard_similarity_score(y_test, lryhat)
```

5]: 0.523501274596855

```
# F1-Score
f1_score(y_test, lryhat, average='macro')
```

6]: 0.5098573271706865

```
# Logloss
yhat_prob = lr.predict_proba(X_test)
log_loss(y_test, yhat_prob)
```

7]: 0.6855290309651024

Random Forest Classification evaluation

Random Forest Classifier Evaluation

```
# Jaccard Similarity Score
jaccard_similarity_score(y_test, yhat)
```

.8]: 0.5608226162174548

```
# F1-Score
f1_score(y_test, lryhat, average='macro')
```

.9]: 0.5098573271706865

5.2 Recommendations -

Model	Jaccard Score	F1-Score	Logloss
KNN	0.5512563468298398	0.5046608485281039	0
Decision Tree	0.5633381070173155	0.5387154848944434	0
Logistic Regression	0.5237338888165604	0.5097244174539972	0.6855086422628527
Random Forest	0.5617497721650827	0.5097244174539972	0

Based on the different models and their evaluation we have seen that the Jaccard-Similarity score is high for Decision Tree model and if we look at the F1 Score is high for Decision Tree model
Hence, we have an opinion that Decision Tree is the best model to predict car accident severity as per the data provided.

6. Conclusion section

6.1 Report conclusions

Based on dataset we have used for our analysis and after going through several steps of data analysis and fitting them through different machine learning models we can conclude that the weather conditions, road conditions, street light conditions that there are particular weather conditions have impact on whether or travel could result in property damage (class1) or injury (class2). The Seattle government should concentrate on looking at road conditions, street lights conditions and weather predictions with which they can take necessary precautions and measures to ensure that severe car accidents do not occur which will lead to less fatalities and reduce hospitalisation expenses and save many people car damage repairs and unnecessary insurance claims. This will help improvement of infrastructure like road, street maintenance too by other government departments in tackling other civil society needs. These measures in collaboration not only will reduce car accidents severity but also control other factors like waterlogging, road infrastructure, drug menace, crimes, civil harmony. Hence the dataset clearly provides the attributes which need to be concentrated by the civil authorities.

======= > The End < =======