Jupyter_Notebook2

September 23, 2020

1 Car Accident Severity Analysis using Machine Learning Algorithms

1.0.1 Importing the libraries

[4]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

[5]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier

# Import DecisionTreeClassifier from sklearn.tree
from sklearn.tree import DecisionTreeClassifier

# Import RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

# Import LogisticRegression
from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import accuracy_score

print("Hello Capstone Project Course!")
```

```
Hello Capstone Project Course!
```

1.0.2 Introduction & Business Understanding

Road accidents are one of the major causes of death and disability all over the world. Reason for road accidents can be environmental conditions such as weather, traffic on road, type of road, speed and light conditions. This paper addresses the in-depth analysis that identifies as the contributory factors behind the road accidents and the quantification of the factors that affect the frequency and severity of accidents based on the crash data available.The severity of each accident can be predicted quite accurately with various classification machine learning algorithms. This can ultimately help government, traffic police, medical institutions, individual drivers and the insurance companies by getting useful insights of the accident severity regarding the causes and consequences of the accidents.The Machine Learning 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. The model will predict the accident severity with various supervised machine learning algorithms i.e. * Algorithm A. Logistic regression * Algorithm B. The K-Nearest Neighbors (KNN) algorithm * Algorithm C. Decision Tree * Algorithm D. Random Forest And finally, the accuracy score versus algorithm will be plotted to check which algorithm performs better.

1.0.3 Data Understanding

The data used for this project was collected by the SDOT traffic management Division and Seattle Traffic Records Group from 2004 to present. It was downloaded from the link shared in the IBM Applied Data Science Capstone course. The data consists of 38 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 to 2. Severity codes are as follows: 1: Property Damage Only Collision 2: Injury Collision Furthermore, as there are null values in some records, the data needs to be pre-processed before any further processing.

Reading the CSV Data

[6]:
```
# Reading the CSV file "Data-Collisions"

df = pd.read_csv (r"C:\Users\salma\Desktop\Data-Collisions.csv")
df.info()
pd.options.display.max_columns=200
df.head()
```

```
c:\users\salma\desktop\projects\venv\new\new\lib\site-
packages\IPython\core\interactiveshell.py:3145: DtypeWarning: Columns (32) have
mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 194673 entries, 0 to 194672
Data columns (total 37 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   SEVERITYCODE    194673 non-null  int64
 1   longitude       189339 non-null  float64
 2   latitude        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  SEVERITYDESC    194673 non-null  object
 14  COLLISIONTYPE   189769 non-null  object
 15  PERSONCOUNT     194673 non-null  int64
 16  PEDCOUNT        194673 non-null  int64
 17  PEDCYLCOUNT     194673 non-null  int64
 18  VEHCOUNT        194673 non-null  int64
 19  INCDATE         194673 non-null  object
 20  INCDTTM         194673 non-null  object
 21  JUNCTIONTYPE    188344 non-null  object
 22  SDOT_COLCODE    194673 non-null  int64
 23  SDOT_COLDESC    194673 non-null  object
 24  INATTENTIONIND  29805 non-null   object
 25  UNDERINFL       189789 non-null  object
 26  WEATHER         189592 non-null  object
 27  ROADCOND        189661 non-null  object
 28  LIGHTCOND       189503 non-null  object
 29  PEDROWNOTGRNT   4667 non-null    object
 30  SDOTCOLNUM      114936 non-null  float64
 31  SPEEDING        9333 non-null    object
 32  ST_COLCODE      194655 non-null  object
 33  ST_COLDESC      189769 non-null  object
 34  SEGLANEKEY      194673 non-null  int64
 35  CROSSWALKKEY    194673 non-null  int64
 36  HITPARKEDCAR    194673 non-null  object
dtypes: float64(4), int64(11), object(22)
memory usage: 55.0+ MB
```

[6]:
```
   SEVERITYCODE   longitude   latitude  OBJECTID  INCKEY  COLDETKEY REPORTNO  \
0             2 -122.323148  47.703140         1    1307       1307  3502005
1             1 -122.347294  47.647172         2   52200      52200  2607959
2             1 -122.334540  47.607871         3   26700      26700  1482393
3             1 -122.334803  47.604803         4    1144       1144  3503937
4             2 -122.306426  47.545739         5   17700      17700  1807429

    STATUS      ADDRTYPE   INTKEY  \
0  Matched  Intersection  37475.0
1  Matched         Block      NaN
2  Matched         Block      NaN
3  Matched         Block      NaN
4  Matched  Intersection  34387.0

                                         LOCATION EXCEPTRSNCODE EXCEPTRSNDESC  \
0                    5TH AVE NE AND NE 103RD ST                          NaN
```

```
1   AURORA BR BETWEEN RAYE ST AND BRIDGE WAY N          NaN           NaN
2  4TH AVE BETWEEN SENECA ST AND UNIVERSITY ST          NaN           NaN
3     2ND AVE BETWEEN MARION ST AND MADISON ST                        NaN
4              SWIFT AVE S AND SWIFT AV OFF RP          NaN           NaN

                  SEVERITYDESC COLLISIONTYPE  PERSONCOUNT  PEDCOUNT  \
0              Injury Collision        Angles            2         0
1  Property Damage Only Collision     Sideswipe            2         0
2  Property Damage Only Collision    Parked Car            4         0
3  Property Damage Only Collision         Other            3         0
4              Injury Collision        Angles            2         0

   PEDCYLCOUNT  VEHCOUNT                 INCDATE          INCDTTM  \
0            0         2  2013/03/27 00:00:00+00   3/27/2013 14:54
1            0         2  2006/12/20 00:00:00+00  12/20/2006 18:55
2            0         3  2004/11/18 00:00:00+00  11/18/2004 10:20
3            0         3  2013/03/29 00:00:00+00    3/29/2013 9:26
4            0         2  2004/01/28 00:00:00+00    1/28/2004 8:04

                               JUNCTIONTYPE  SDOT_COLCODE  \
0  At Intersection_related to intersection            11
1  Mid-Block (not related to intersection)            16
2  Mid-Block (not related to intersection)            14
3  Mid-Block (not related to intersection)            11
4  At Intersection_related to intersection            11

                                      SDOT_COLDESC INATTENTIONIND UNDERINFL  \
0  MOTOR VEHICLE STRUCK MOTOR VEHICLE, FRONT END …            NaN         N
1  MOTOR VEHICLE STRUCK MOTOR VEHICLE, LEFT SIDE …            NaN         N
2       MOTOR VEHICLE STRUCK MOTOR VEHICLE, REAR END            NaN         N
3  MOTOR VEHICLE STRUCK MOTOR VEHICLE, FRONT END …            NaN         N
4  MOTOR VEHICLE STRUCK MOTOR VEHICLE, FRONT END …            NaN         N

    WEATHER ROADCOND                LIGHTCOND PEDROWNOTGRNT  SDOTCOLNUM  \
0  Overcast      Wet                 Daylight           NaN         NaN
1   Raining      Wet  Dark - Street Lights On           NaN   6354039.0
2  Overcast      Dry                 Daylight           NaN   4323031.0
3     Clear      Dry                 Daylight           NaN         NaN
4   Raining      Wet                 Daylight           NaN   4028032.0

  SPEEDING ST_COLCODE                                         ST_COLDESC  \
0      NaN         10                                  Entering at angle
1      NaN         11  From same direction - both going straight - bo…
2      NaN         32                             One parked--one moving
3      NaN         23                   From same direction - all others
4      NaN         10                                  Entering at angle
```

```
   SEGLANEKEY  CROSSWALKKEY HITPARKEDCAR
0           0             0            N
1           0             0            N
2           0             0            N
3           0             0            N
4           0             0            N
```

Checking the percentage (%) of missing values in the columns

[7]:
```
df.isna().mean().round(4) * 100
```

[7]:
```
SEVERITYCODE       0.00
longitude          2.74
latitude           2.74
OBJECTID           0.00
INCKEY             0.00
COLDETKEY          0.00
REPORTNO           0.00
STATUS             0.00
ADDRTYPE           0.99
INTKEY            66.57
LOCATION           1.38
EXCEPTRSNCODE     56.43
EXCEPTRSNDESC     97.10
SEVERITYDESC       0.00
COLLISIONTYPE      2.52
PERSONCOUNT        0.00
PEDCOUNT           0.00
PEDCYLCOUNT        0.00
VEHCOUNT           0.00
INCDATE            0.00
INCDTTM            0.00
JUNCTIONTYPE       3.25
SDOT_COLCODE       0.00
SDOT_COLDESC       0.00
INATTENTIONIND    84.69
UNDERINFL          2.51
WEATHER            2.61
ROADCOND           2.57
LIGHTCOND          2.66
PEDROWNOTGRNT     97.60
SDOTCOLNUM        40.96
SPEEDING          95.21
ST_COLCODE         0.01
ST_COLDESC         2.52
SEGLANEKEY         0.00
CROSSWALKKEY       0.00
```

```
HITPARKEDCAR       0.00
dtype: float64
```

[8]:
```
df.shape
```

[8]:
```
(194673, 37)
```

Checking the list of features to include

[9]:
```
numeric_features = df[["PERSONCOUNT","PEDCOUNT","PEDCYLCOUNT", "VEHCOUNT",␣
 ↪"SEVERITYCODE"]]

categorical_features=df[["ADDRTYPE","LOCATION","COLLISIONTYPE",␣
 ↪"INCDATE","INCDTTM","JUNCTIONTYPE",
                         "SDOT_COLDESC", "UNDERINFL", "WEATHER", "ROADCOND",␣
 ↪"LIGHTCOND", "ST_COLDESC", "HITPARKEDCAR"]]
```

Checking the Target Variable

[10]:
```
df["SEVERITYCODE"].value_counts()
```

[10]:
```
1    136485
2     58188
Name: SEVERITYCODE, dtype: int64
```

Description of the Numeric Features

[11]:
```
numeric_features.describe()
```

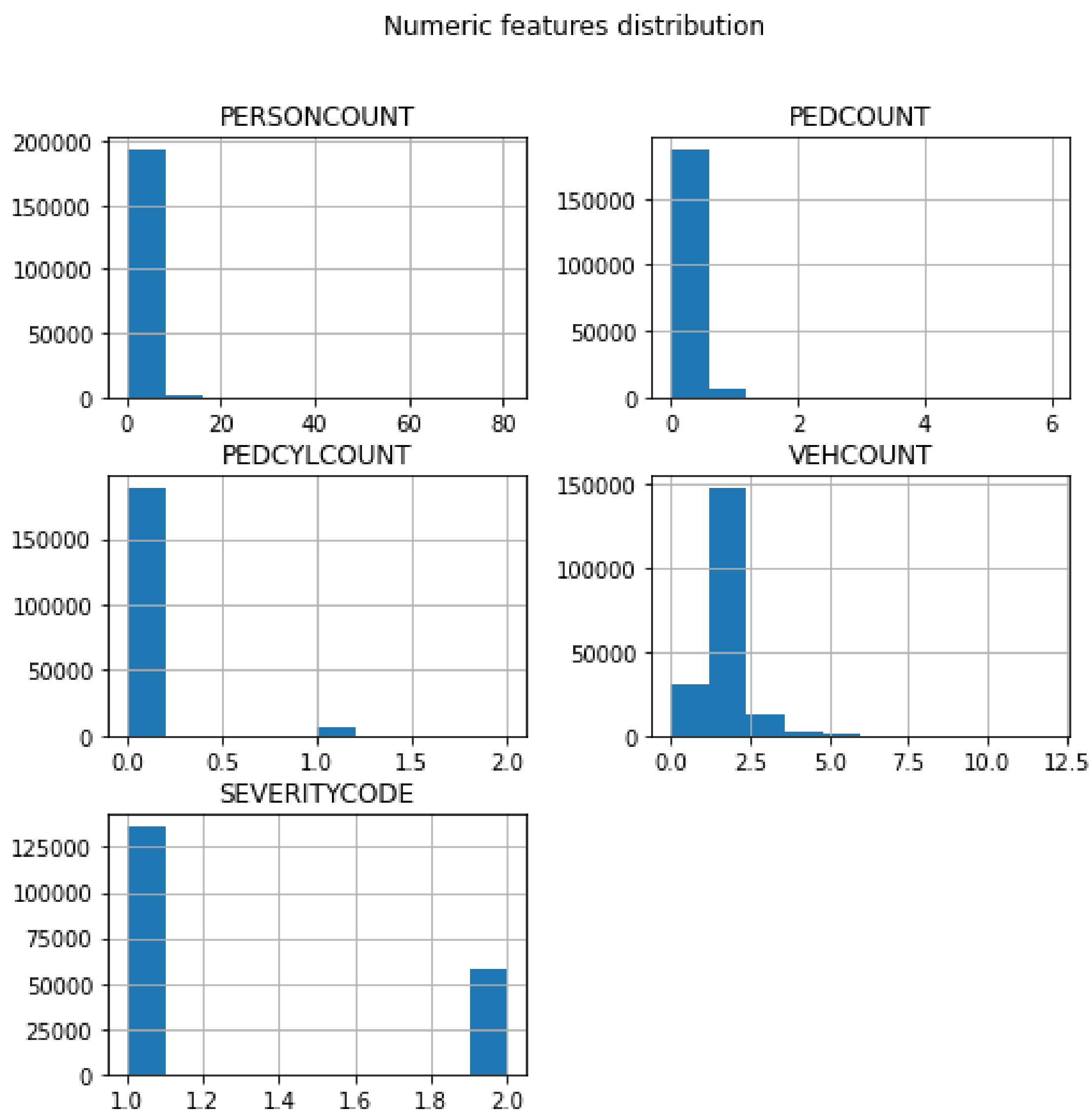
[11]:
```
         PERSONCOUNT       PEDCOUNT    PEDCYLCOUNT       VEHCOUNT  \
count  194673.000000  194673.000000  194673.000000  194673.000000
mean        2.444427       0.037139       0.028391       1.920780
std         1.345929       0.198150       0.167413       0.631047
min         0.000000       0.000000       0.000000       0.000000
25%         2.000000       0.000000       0.000000       2.000000
50%         2.000000       0.000000       0.000000       2.000000
75%         3.000000       0.000000       0.000000       2.000000
max        81.000000       6.000000       2.000000      12.000000

        SEVERITYCODE
count  194673.000000
mean        1.298901
std         0.457778
min         1.000000
25%         1.000000
50%         1.000000
75%         2.000000
max         2.000000
```

Numeric Features Distribution

[12]:
```
numeric_features.hist(figsize=[8,8])
plt.suptitle("Numeric features distribution")
plt.show()
```

[12]:
```
array([[<AxesSubplot:title={'center':'PERSONCOUNT'}>,
        <AxesSubplot:title={'center':'PEDCOUNT'}>],
       [<AxesSubplot:title={'center':'PEDCYLCOUNT'}>,
        <AxesSubplot:title={'center':'VEHCOUNT'}>],
       [<AxesSubplot:title={'center':'SEVERITYCODE'}>, <AxesSubplot:>]],
      dtype=object)
```

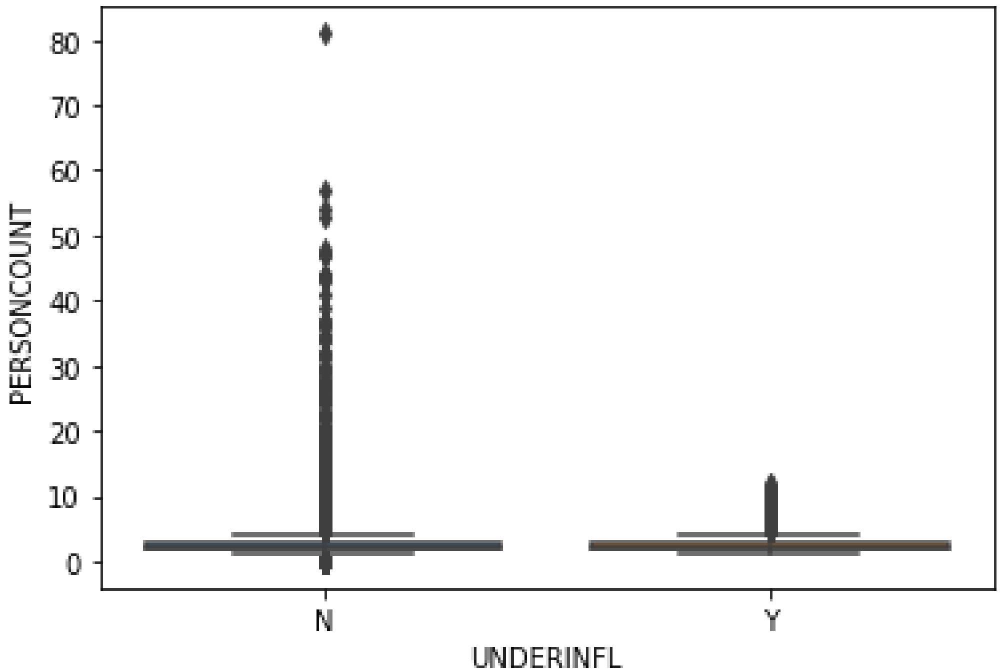
[12]:
```
Text(0.5, 0.98, 'Numeric features distribution')
```



Plotting the under Influence of alcohol along with the person count

[13]:
```
sns.boxplot(x='UNDERINFL',y='PERSONCOUNT',data=df)
plt.show()
```

[13]: <AxesSubplot:xlabel='UNDERINFL', ylabel='PERSONCOUNT'>



1.0.4 Data Preparation

Categorical Features percentage (%) of samples from the selected Data

"COLLISIONTYPE"

[29]:
```
counts = categorical_features["COLLISIONTYPE"].value_counts()
percent100 = categorical_features["COLLISIONTYPE"].value_counts(normalize=True).
↪mul(100).round(1).astype(str) + '%'
collision=pd.DataFrame({'counts': counts, 'Percent': percent100})
print(collision)
```

```
             counts Percent
Parked Car    47987   25.3%
Angles        34674   18.3%
Rear Ended    34090   18.0%
Other         23703   12.5%
Sideswipe     18609    9.8%
```

```
Left Turn     13703    7.2%
Pedestrian     6608    3.5%
Cycles         5415    2.9%
Right Turn     2956    1.6%
Head On        2024    1.1%
```

“LIGHTCOND”

[14]:
```
counts = categorical_features["LIGHTCOND"].value_counts()
percent100 = categorical_features["LIGHTCOND"].value_counts(normalize=True).
 ↪mul(100).round(1).astype(str) + '%'
light_conditions=pd.DataFrame({'counts': counts, 'Percent': percent100})
print(light_conditions)
```

```
                          counts Percent
Daylight                  116137   61.3%
Dark - Street Lights On    48507   25.6%
Unknown                    13473    7.1%
Dusk                        5902    3.1%
Dawn                        2502    1.3%
Dark - No Street Lights     1537    0.8%
Dark - Street Lights Off    1199    0.6%
Other                        235    0.1%
Dark - Unknown Lighting       11    0.0%
```

“ROADCOND”

[15]:
```
counts = categorical_features["ROADCOND"].value_counts()
percent100 = categorical_features["ROADCOND"].value_counts(normalize=True).
 ↪mul(100).round(1).astype(str) + '%'
road_condition=pd.DataFrame({'counts': counts, 'Percent': percent100})
print(road_condition)
```

```
                counts Percent
Dry             124510   65.6%
Wet              47474   25.0%
Unknown          15078    7.9%
Ice               1209    0.6%
Snow/Slush        1004    0.5%
Other              132    0.1%
Standing Water     115    0.1%
Sand/Mud/Dirt       75    0.0%
Oil                 64    0.0%
```

“SDOT_COLDESC”

[16]:
```
counts = categorical_features["SDOT_COLDESC"].value_counts()
percent100 = categorical_features["SDOT_COLDESC"].value_counts(normalize=True).
 ↪mul(100).round(1).astype(str) + '%'
collision_desc=pd.DataFrame({'counts': counts, 'Percent': percent100})
```

```python
print(collision_desc)
```

```
                                                    counts Percent
MOTOR VEHICLE STRUCK MOTOR VEHICLE, FRONT END A…   85209   43.8%
MOTOR VEHICLE STRUCK MOTOR VEHICLE, REAR END        54299   27.9%
MOTOR VEHICLE STRUCK MOTOR VEHICLE, LEFT SIDE S…    9928    5.1%
NOT ENOUGH INFORMATION / NOT APPLICABLE              9787    5.0%
MOTOR VEHICLE RAN OFF ROAD - HIT FIXED OBJECT        8856    4.5%
MOTOR VEHCILE STRUCK PEDESTRIAN                      6518    3.3%
MOTOR VEHICLE STRUCK MOTOR VEHICLE, LEFT SIDE A…    5852    3.0%
MOTOR VEHICLE STRUCK OBJECT IN ROAD                  4741    2.4%
MOTOR VEHICLE STRUCK PEDALCYCLIST, FRONT END AT…    3104    1.6%
MOTOR VEHICLE STRUCK MOTOR VEHICLE, RIGHT SIDE …    1604    0.8%
MOTOR VEHICLE STRUCK MOTOR VEHICLE, RIGHT SIDE …    1440    0.7%
PEDALCYCLIST STRUCK MOTOR VEHICLE FRONT END AT …    1312    0.7%
MOTOR VEHICLE OVERTURNED IN ROAD                      479    0.2%
MOTOR VEHICLE STRUCK PEDALCYCLIST, REAR END           181    0.1%
PEDALCYCLIST STRUCK MOTOR VEHICLE LEFT SIDE SID…     180    0.1%
MOTOR VEHICLE RAN OFF ROAD - NO COLLISION             166    0.1%
PEDALCYCLIST STRUCK MOTOR VEHICLE REAR END            139    0.1%
MOTOR VEHICLE STRUCK PEDALCYCLIST, LEFT SIDE SI…     124    0.1%
DRIVERLESS VEHICLE RAN OFF ROAD - HIT FIXED OBJECT    107    0.1%
DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE FRONT E…     104    0.1%
MOTOR VEHICLE STRUCK TRAIN                            102    0.1%
DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE REAR END       93    0.0%
PEDALCYCLIST STRUCK PEDESTRIAN                         75    0.0%
PEDALCYCLIST OVERTURNED IN ROAD                        69    0.0%
DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE LEFT SI…      53    0.0%
PEDALCYCLIST STRUCK MOTOR VEHICLE RIGHT SIDE SI…      50    0.0%
PEDALCYCLIST STRUCK OBJECT IN ROAD                     23    0.0%
MOTOR VEHICLE STRUCK PEDALCYCLIST, RIGHT SIDE S…      17    0.0%
DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE RIGHT S…      12    0.0%
PEDALCYCLIST STRUCK MOTOR VEHICLE LEFT SIDE AT …       9    0.0%
DRIVERLESS VEHICLE STRUCK PEDESTRIAN                    8    0.0%
PEDALCYCLIST STRUCK PEDALCYCLIST REAR END               7    0.0%
DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE RIGHT S…       6    0.0%
PEDALCYCLIST STRUCK PEDALCYCLIST FRONT END AT A…       5    0.0%
PEDALCYCLIST RAN OFF ROAD - HIT FIXED OBJECT            4    0.0%
DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE LEFT SI…       4    0.0%
DRIVERLESS VEHICLE STRUCK OBJECT IN ROADWAY             3    0.0%
PEDALCYCLIST STRUCK MOTOR VEHICLE RIGHT SIDE AT…       2    0.0%
DRIVERLESS VEHICLE RAN OFF ROAD - NO COLLISION          1    0.0%
```

“WEATHER”

[17]:
```python
counts = categorical_features["WEATHER"].value_counts()
percent100 = categorical_features["WEATHER"].value_counts(normalize=True).
 ↪mul(100).round(1).astype(str) + '%'
```

```
weather=pd.DataFrame({'counts': counts, 'Percent': percent100})
print(weather)
```

```
                          counts Percent
Clear                     111135   58.6%
Raining                    33145   17.5%
Overcast                   27714   14.6%
Unknown                    15091    8.0%
Snowing                      907    0.5%
Other                        832    0.4%
Fog/Smog/Smoke               569    0.3%
Sleet/Hail/Freezing Rain     113    0.1%
Blowing Sand/Dirt             56    0.0%
Severe Crosswind              25    0.0%
Partly Cloudy                  5    0.0%
```

"JUNCTIONTYPE"

[18]:
```
counts = categorical_features["JUNCTIONTYPE"].value_counts()
percent100 = categorical_features["JUNCTIONTYPE"].value_counts(normalize=True).
↪mul(100).round(1).astype(str) + '%'
junction=pd.DataFrame({'counts': counts, 'Percent': percent100})
print(junction)
```

```
                                                  counts Percent
Mid-Block (not related to intersection)            89800   47.7%
At Intersection_related to intersection            62810   33.3%
Mid-Block (but intersection related)               22790   12.1%
Driveway Junction                                  10671    5.7%
At Intersection_not related to intersection         2098    1.1%
Ramp Junction                                        166    0.1%
Unknown                                                9    0.0%
```

"UNDERINFL"

[19]:
```
counts = categorical_features["UNDERINFL"].value_counts()
percent100 = categorical_features["UNDERINFL"].value_counts(normalize=True).
↪mul(100).round(1).astype(str) + '%'
under_influence=pd.DataFrame({'counts': counts, 'Percent': percent100})
print(under_influence)
```

```
   counts Percent
N  180668   95.2%
Y    9121    4.8%
```

Formatting the Date & time for the analysis

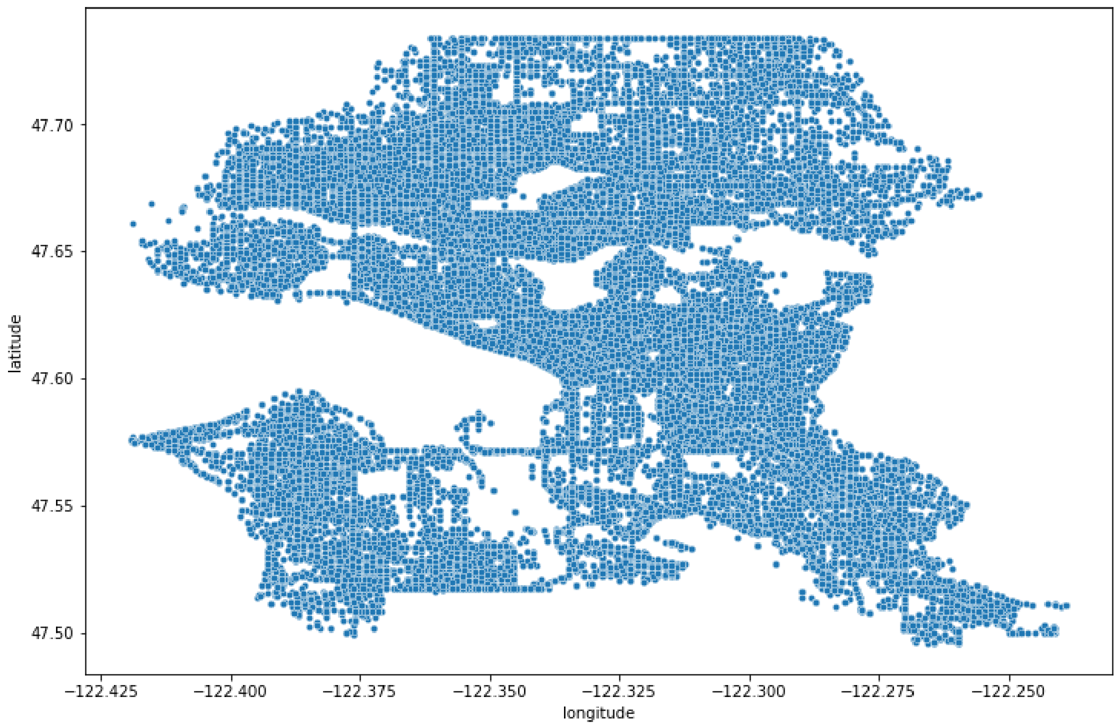
[20]:
```
df['INCDTTM'] = pd.to_datetime(df['INCDTTM'], errors='coerce')
df['Year']=df['INCDTTM'].dt.year
```

```
df['Month']=df['INCDTTM'].dt.strftime('%b')
df['Day']=df['INCDTTM'].dt.day
df['Hour']=df['INCDTTM'].dt.hour
df['Weekday']=df['INCDTTM'].dt.strftime('%a')
```

Scatter plot of the accident coordinates

[36]:
```
fig = plt.gcf()
fig.set_size_inches(12, 8)
sns.scatterplot(x='longitude', y='latitude', data=df, legend=False, s=20)
plt.show()
```

[36]: <AxesSubplot:xlabel='longitude', ylabel='latitude'>



Checking the Null values in the Dataframe

[38]:
```
df.isnull()
```

[38]:

	SEVERITYCODE	longitude	latitude	OBJECTID	INCKEY	COLDETKEY	\
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	

```
…              …         …        …         …       …        …
194668         False      False     False     False   False      False
194669         False      False     False     False   False      False
194670         False      False     False     False   False      False
194671         False      False     False     False   False      False
194672         False      False     False     False   False      False

        REPORTNO  STATUS  ADDRTYPE  INTKEY  LOCATION  EXCEPTRSNCODE  \
0          False   False     False   False     False          False
1          False   False     False    True     False           True
2          False   False     False    True     False           True
3          False   False     False    True     False          False
4          False   False     False   False     False           True
…          …      …       …      …       …            …
194668     False   False     False    True     False          False
194669     False   False     False    True     False          False
194670     False   False     False   False     False          False
194671     False   False     False   False     False          False
194672     False   False     False    True     False          False

        EXCEPTRSNDESC  SEVERITYDESC  COLLISIONTYPE  PERSONCOUNT  PEDCOUNT  \
0                True         False          False        False     False
1                True         False          False        False     False
2                True         False          False        False     False
3                True         False          False        False     False
4                True         False          False        False     False
…               …           …             …           …       …
194668           True         False          False        False     False
194669           True         False          False        False     False
194670           True         False          False        False     False
194671           True         False          False        False     False
194672           True         False          False        False     False

        PEDCYLCOUNT  VEHCOUNT  INCDATE  INCDTTM  JUNCTIONTYPE  SDOT_COLCODE  \
0             False     False    False    False         False         False
1             False     False    False    False         False         False
2             False     False    False    False         False         False
3             False     False    False    False         False         False
4             False     False    False    False         False         False
…             …      …      …      …            …            …
194668        False     False    False    False         False         False
194669        False     False    False    False         False         False
194670        False     False    False    False         False         False
194671        False     False    False    False         False         False
194672        False     False    False    False         False         False

        SDOT_COLDESC  INATTENTIONIND  UNDERINFL  WEATHER  ROADCOND  LIGHTCOND  \
```

```
0               False             True      False    False      False       False
1               False             True      False    False      False       False
2               False             True      False    False      False       False
3               False             True      False    False      False       False
4               False             True      False    False      False       False
…               …              …        …      …       …        …
194668          False             True      False    False      False       False
194669          False            False      False    False      False       False
194670          False             True      False    False      False       False
194671          False             True      False    False      False       False
194672          False             True      False    False      False       False

        PEDROWNOTGRNT  SDOTCOLNUM  SPEEDING  ST_COLCODE  ST_COLDESC  \
0                True        True      True       False       False
1                True       False      True       False       False
2                True       False      True       False       False
3                True        True      True       False       False
4                True       False      True       False       False
…              …          …       …         …         …
194668           True        True      True       False       False
194669           True        True      True       False       False
194670           True        True      True       False       False
194671           True        True      True       False       False
194672           True        True      True       False       False

        SEGLANEKEY  CROSSWALKKEY  HITPARKEDCAR   Year  Month    Day   Hour  \
0            False         False         False  False  False  False  False
1            False         False         False  False  False  False  False
2            False         False         False  False  False  False  False
3            False         False         False  False  False  False  False
4            False         False         False  False  False  False  False
…            …            …            …     …    …    …    …
194668       False         False         False  False  False  False  False
194669       False         False         False  False  False  False  False
194670       False         False         False  False  False  False  False
194671       False         False         False  False  False  False  False
194672       False         False         False  False  False  False  False

        Weekday
0         False
1         False
2         False
3         False
4         False
…         …
194668    False
194669    False
```

```
194670    False
194671    False
194672    False

[194673 rows x 42 columns]
```

[39]:
```
df.isnull().sum()
```

[39]:
```
SEVERITYCODE           0
longitude           5334
latitude            5334
OBJECTID               0
INCKEY                 0
COLDETKEY              0
REPORTNO               0
STATUS                 0
ADDRTYPE            1926
INTKEY            129603
LOCATION            2677
EXCEPTRSNCODE     109862
EXCEPTRSNDESC     189035
SEVERITYDESC           0
COLLISIONTYPE       4904
PERSONCOUNT            0
PEDCOUNT               0
PEDCYLCOUNT            0
VEHCOUNT               0
INCDATE                0
INCDTTM                0
JUNCTIONTYPE        6329
SDOT_COLCODE           0
SDOT_COLDESC           0
INATTENTIONIND    164868
UNDERINFL           4884
WEATHER             5081
ROADCOND            5012
LIGHTCOND           5170
PEDROWNOTGRNT     190006
SDOTCOLNUM         79737
SPEEDING          185340
ST_COLCODE            18
ST_COLDESC          4904
SEGLANEKEY             0
CROSSWALKKEY           0
HITPARKEDCAR           0
Year                   0
Month                  0
```

```
Day                    0
Hour                   0
Weekday                0
dtype: int64
```

Selecting and finalizing the features for Machine Learning Model

[40]:
```
selected_features = ["SEVERITYCODE", "longitude", "latitude", "PERSONCOUNT",
                     "PEDCOUNT", "PEDCYLCOUNT", "VEHCOUNT", "ADDRTYPE",␣
 ↪"COLLISIONTYPE", "WEATHER", "ROADCOND",
                     "LIGHTCOND", "SDOT_COLDESC", "HITPARKEDCAR", "Hour"]
df_sel=df[selected_features].copy()
df_sel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 194673 entries, 0 to 194672
Data columns (total 15 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   SEVERITYCODE    194673 non-null  int64
 1   longitude       189339 non-null  float64
 2   latitude        189339 non-null  float64
 3   PERSONCOUNT     194673 non-null  int64
 4   PEDCOUNT        194673 non-null  int64
 5   PEDCYLCOUNT     194673 non-null  int64
 6   VEHCOUNT        194673 non-null  int64
 7   ADDRTYPE        192747 non-null  object
 8   COLLISIONTYPE   189769 non-null  object
 9   WEATHER         189592 non-null  object
 10  ROADCOND        189661 non-null  object
 11  LIGHTCOND       189503 non-null  object
 12  SDOT_COLDESC    194673 non-null  object
 13  HITPARKEDCAR    194673 non-null  object
 14  Hour            194673 non-null  int64
dtypes: float64(2), int64(6), object(7)
memory usage: 22.3+ MB
```

Checking the Null values in the selected dataframe and dropping the rows with the null values

[41]:
```
df_sel.isnull().mean()
```

[41]:
```
SEVERITYCODE     0.000000
longitude        0.027400
latitude         0.027400
PERSONCOUNT      0.000000
PEDCOUNT         0.000000
PEDCYLCOUNT      0.000000
```

```
VEHCOUNT          0.000000
ADDRTYPE          0.009894
COLLISIONTYPE     0.025191
WEATHER           0.026100
ROADCOND          0.025746
LIGHTCOND         0.026557
SDOT_COLDESC      0.000000
HITPARKEDCAR      0.000000
Hour              0.000000
dtype: float64
```

[42]:
```python
df_sel.shape
```

[42]:
```
(194673, 15)
```

[43]:
```python
df_sel.dropna(subset=df_sel.columns[df_sel.isnull().mean()!=0], how='any',␣
 ↪axis=0, inplace=True)
df_sel.shape
```

[43]:
```
(184146, 15)
```

[44]:
```python
# Export the data with selected features

df_sel.to_csv('./Data-Collisions_clean_sel.csv',index=False)
```

Generating the dummies for the Categorical Data

[45]:
```python
# Generate dummies for categorical data
df_dummy = pd.get_dummies(df_sel, drop_first=True)
# Export data
df_dummy.to_csv("./Data-Collisions_{}_dummy.csv", index=False)
df_dummy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 184146 entries, 0 to 194672
Data columns (total 83 columns):
 #   Column
Non-Null Count   Dtype
---  ------
--------------   -----
 0   SEVERITYCODE
184146 non-null  int64
 1   longitude
184146 non-null  float64
 2   latitude
184146 non-null  float64
 3   PERSONCOUNT
184146 non-null  int64
```

```
 4   PEDCOUNT
184146 non-null  int64
 5   PEDCYLCOUNT
184146 non-null  int64
 6   VEHCOUNT
184146 non-null  int64
 7   Hour
184146 non-null  int64
 8   ADDRTYPE_Intersection
184146 non-null  uint8
 9   COLLISIONTYPE_Cycles
184146 non-null  uint8
 10  COLLISIONTYPE_Head On
184146 non-null  uint8
 11  COLLISIONTYPE_Left Turn
184146 non-null  uint8
 12  COLLISIONTYPE_Other
184146 non-null  uint8
 13  COLLISIONTYPE_Parked Car
184146 non-null  uint8
 14  COLLISIONTYPE_Pedestrian
184146 non-null  uint8
 15  COLLISIONTYPE_Rear Ended
184146 non-null  uint8
 16  COLLISIONTYPE_Right Turn
184146 non-null  uint8
 17  COLLISIONTYPE_Sideswipe
184146 non-null  uint8
 18  WEATHER_Clear
184146 non-null  uint8
 19  WEATHER_Fog/Smog/Smoke
184146 non-null  uint8
 20  WEATHER_Other
184146 non-null  uint8
 21  WEATHER_Overcast
184146 non-null  uint8
 22  WEATHER_Partly Cloudy
184146 non-null  uint8
 23  WEATHER_Raining
184146 non-null  uint8
 24  WEATHER_Severe Crosswind
184146 non-null  uint8
 25  WEATHER_Sleet/Hail/Freezing Rain
184146 non-null  uint8
 26  WEATHER_Snowing
184146 non-null  uint8
 27  WEATHER_Unknown
184146 non-null  uint8
```

```
 28  ROADCOND_Ice
184146 non-null  uint8
 29  ROADCOND_Oil
184146 non-null  uint8
 30  ROADCOND_Other
184146 non-null  uint8
 31  ROADCOND_Sand/Mud/Dirt
184146 non-null  uint8
 32  ROADCOND_Snow/Slush
184146 non-null  uint8
 33  ROADCOND_Standing Water
184146 non-null  uint8
 34  ROADCOND_Unknown
184146 non-null  uint8
 35  ROADCOND_Wet
184146 non-null  uint8
 36  LIGHTCOND_Dark - Street Lights Off
184146 non-null  uint8
 37  LIGHTCOND_Dark - Street Lights On
184146 non-null  uint8
 38  LIGHTCOND_Dark - Unknown Lighting
184146 non-null  uint8
 39  LIGHTCOND_Dawn
184146 non-null  uint8
 40  LIGHTCOND_Daylight
184146 non-null  uint8
 41  LIGHTCOND_Dusk
184146 non-null  uint8
 42  LIGHTCOND_Other
184146 non-null  uint8
 43  LIGHTCOND_Unknown
184146 non-null  uint8
 44  SDOT_COLDESC_DRIVERLESS VEHICLE RAN OFF ROAD - NO COLLISION
184146 non-null  uint8
 45  SDOT_COLDESC_DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE FRONT END AT ANGLE
184146 non-null  uint8
 46  SDOT_COLDESC_DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE LEFT SIDE AT ANGLE
184146 non-null  uint8
 47  SDOT_COLDESC_DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE LEFT SIDE SIDESWIPE
184146 non-null  uint8
 48  SDOT_COLDESC_DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE REAR END
184146 non-null  uint8
 49  SDOT_COLDESC_DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE RIGHT SIDE AT ANGLE
184146 non-null  uint8
 50  SDOT_COLDESC_DRIVERLESS VEHICLE STRUCK MOTOR VEHICLE RIGHT SIDE SIDESWIPE
184146 non-null  uint8
 51  SDOT_COLDESC_DRIVERLESS VEHICLE STRUCK OBJECT IN ROADWAY
184146 non-null  uint8
```

52 SDOT_COLDESC_DRIVERLESS VEHICLE STRUCK PEDESTRIAN
184146 non-null uint8
53 SDOT_COLDESC_MOTOR VEHCILE STRUCK PEDESTRIAN
184146 non-null uint8
54 SDOT_COLDESC_MOTOR VEHICLE OVERTURNED IN ROAD
184146 non-null uint8
55 SDOT_COLDESC_MOTOR VEHICLE RAN OFF ROAD - HIT FIXED OBJECT
184146 non-null uint8
56 SDOT_COLDESC_MOTOR VEHICLE RAN OFF ROAD - NO COLLISION
184146 non-null uint8
57 SDOT_COLDESC_MOTOR VEHICLE STRUCK MOTOR VEHICLE, FRONT END AT ANGLE
184146 non-null uint8
58 SDOT_COLDESC_MOTOR VEHICLE STRUCK MOTOR VEHICLE, LEFT SIDE AT ANGLE
184146 non-null uint8
59 SDOT_COLDESC_MOTOR VEHICLE STRUCK MOTOR VEHICLE, LEFT SIDE SIDESWIPE
184146 non-null uint8
60 SDOT_COLDESC_MOTOR VEHICLE STRUCK MOTOR VEHICLE, REAR END
184146 non-null uint8
61 SDOT_COLDESC_MOTOR VEHICLE STRUCK MOTOR VEHICLE, RIGHT SIDE AT ANGLE
184146 non-null uint8
62 SDOT_COLDESC_MOTOR VEHICLE STRUCK MOTOR VEHICLE, RIGHT SIDE SIDESWIPE
184146 non-null uint8
63 SDOT_COLDESC_MOTOR VEHICLE STRUCK OBJECT IN ROAD
184146 non-null uint8
64 SDOT_COLDESC_MOTOR VEHICLE STRUCK PEDALCYCLIST, FRONT END AT ANGLE
184146 non-null uint8
65 SDOT_COLDESC_MOTOR VEHICLE STRUCK PEDALCYCLIST, LEFT SIDE SIDESWIPE
184146 non-null uint8
66 SDOT_COLDESC_MOTOR VEHICLE STRUCK PEDALCYCLIST, REAR END
184146 non-null uint8
67 SDOT_COLDESC_MOTOR VEHICLE STRUCK PEDALCYCLIST, RIGHT SIDE SIDESWIPE
184146 non-null uint8
68 SDOT_COLDESC_MOTOR VEHICLE STRUCK TRAIN
184146 non-null uint8
69 SDOT_COLDESC_NOT ENOUGH INFORMATION / NOT APPLICABLE
184146 non-null uint8
70 SDOT_COLDESC_PEDALCYCLIST OVERTURNED IN ROAD
184146 non-null uint8
71 SDOT_COLDESC_PEDALCYCLIST RAN OFF ROAD - HIT FIXED OBJECT
184146 non-null uint8
72 SDOT_COLDESC_PEDALCYCLIST STRUCK MOTOR VEHICLE FRONT END AT ANGLE
184146 non-null uint8
73 SDOT_COLDESC_PEDALCYCLIST STRUCK MOTOR VEHICLE LEFT SIDE AT ANGLE
184146 non-null uint8
74 SDOT_COLDESC_PEDALCYCLIST STRUCK MOTOR VEHICLE LEFT SIDE SIDESWIPE
184146 non-null uint8
75 SDOT_COLDESC_PEDALCYCLIST STRUCK MOTOR VEHICLE REAR END
184146 non-null uint8

```
 76  SDOT_COLDESC_PEDALCYCLIST STRUCK MOTOR VEHICLE RIGHT SIDE AT ANGLE
184146 non-null  uint8
 77  SDOT_COLDESC_PEDALCYCLIST STRUCK MOTOR VEHICLE RIGHT SIDE SIDESWIPE
184146 non-null  uint8
 78  SDOT_COLDESC_PEDALCYCLIST STRUCK OBJECT IN ROAD
184146 non-null  uint8
 79  SDOT_COLDESC_PEDALCYCLIST STRUCK PEDALCYCLIST FRONT END AT ANGLE
184146 non-null  uint8
 80  SDOT_COLDESC_PEDALCYCLIST STRUCK PEDALCYCLIST REAR END
184146 non-null  uint8
 81  SDOT_COLDESC_PEDALCYCLIST STRUCK PEDESTRIAN
184146 non-null  uint8
 82  HITPARKEDCAR_Y
184146 non-null  uint8
dtypes: float64(2), int64(6), uint8(75)
memory usage: 25.8 MB
```

1.0.5 Modelling

[46]:
```
# Assign the data
df=df_dummy
# Set the target for the prediction
target="SEVERITYCODE"
# Create arrays for the features and the response variable
# set X and y
y = df[target]
X = df.drop(target, axis=1)

# Split the data set into training and testing data sets
# Split the data set into training and testing data sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0,␣
↪test_size=0.33, stratify=y)
```

Selecting the different Algorithms

[47]:
```
algo_lst=['Logistic Regression',' K-Nearest Neighbors','Decision Trees','Random␣
↪Forest']
```

[48]:
```
# Initialize an empty list for the accuracy for each algorithm
accuracy_lst=[]
```

1.0.6 Evalution

Logistic Regression

[49]:
```
# Logistic regression
lr = LogisticRegression(solver='lbfgs', max_iter=400, dual=False).fit(X_test,␣
↪y_test)
```

```
y_pred=lr.predict(X_test)

# Get the accuracy score
acc=accuracy_score(y_test, y_pred)

# Append to the accuracy list
accuracy_lst.append(acc)

print("[Logistic regression algorithm] accuracy_score: {:.3f}.".format(acc))
```

```
[Logistic regression algorithm] accuracy_score: 0.756.
```

```
c:\users\salma\desktop\projects\venv\new\new\lib\site-
packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

K-NN Neighbors

[50]:
```
# Create a k-NN classifier with 6 neighbors
knn = KNeighborsClassifier(n_neighbors=6)

# Fit the classifier to the data
knn.fit(X_train,y_train)

# Predict the labels for the training data X
y_pred = knn.predict(X_test)

# Get the accuracy score
acc=accuracy_score(y_test, y_pred)

# Append to the accuracy list
accuracy_lst.append(acc)
print('[K-Nearest Neighbors (KNN)] knn.score: {:.3f}.'.format(knn.score(X_test,
 ↪y_test)))
print('[K-Nearest Neighbors (KNN)] accuracy_score: {:.3f}.'.format(acc))
```

[50]:
```
KNeighborsClassifier(n_neighbors=6)
```

```
[K-Nearest Neighbors (KNN)] knn.score: 0.739.
[K-Nearest Neighbors (KNN)] accuracy_score: 0.739.
```

Setting arrays for storing the train and test data accuracies

[51]:
```
# Setup arrays to store train and test accuracies
neighbors = np.arange(1, 9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop over different values of k
for i, n_neighbor in enumerate(neighbors):
    # Setup a k-NN Classifier with n_neighbor
    knn = KNeighborsClassifier(n_neighbors=n_neighbor)

    # Fit the classifier to the training data
    knn.fit(X_train,y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train)
    #Compute accuracy on the testing set
    test_accuracy[i] = knn.score(X_test, y_test)
```

[51]:
```
KNeighborsClassifier(n_neighbors=1)
```

[51]:
```
KNeighborsClassifier(n_neighbors=2)
```

[51]:
```
KNeighborsClassifier(n_neighbors=3)
```

[51]:
```
KNeighborsClassifier(n_neighbors=4)
```

[51]:
```
KNeighborsClassifier()
```

[51]:
```
KNeighborsClassifier(n_neighbors=6)
```

[51]:
```
KNeighborsClassifier(n_neighbors=7)
```

[51]:
```
KNeighborsClassifier(n_neighbors=8)
```

Generating a plot for K-NN with varying number of Neighbors

[ ]:
```
# Generate plot

plt.title('k-NN: Varying Number of Neighbors')
plt.plot(neighbors, test_accuracy, label = 'Testing Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training Accuracy')
plt.legend()
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.show()
```

[ ]:
```
Text(0.5, 1.0, 'k-NN: Varying Number of Neighbors')
```

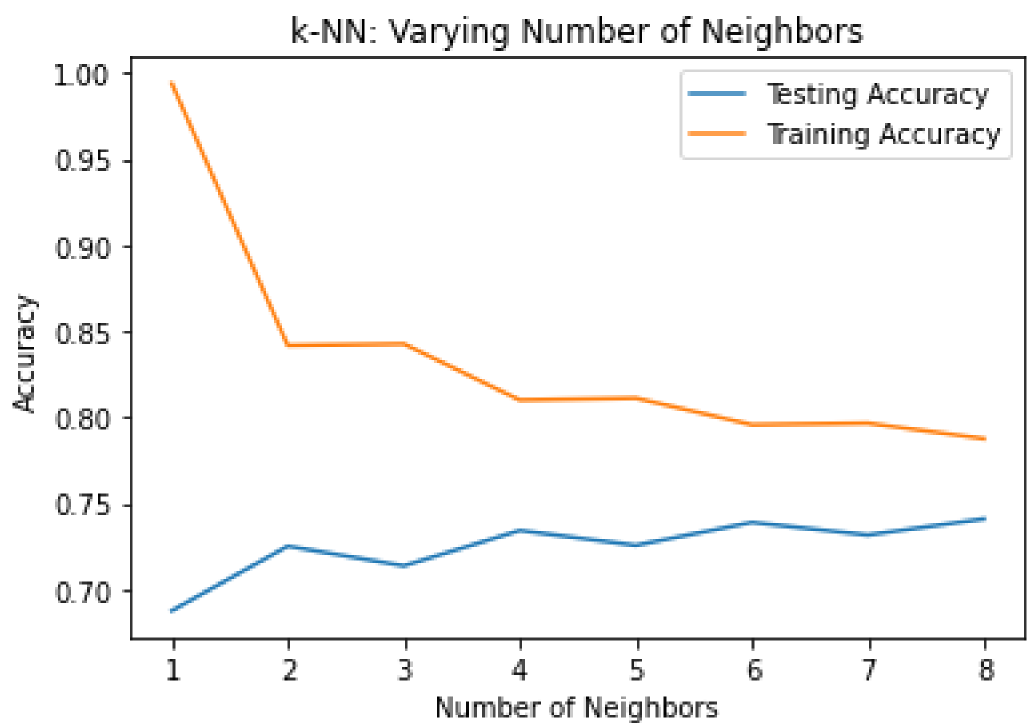
```
[ ]: [<matplotlib.lines.Line2D at 0x1c7dc7bb5b0>]
```

```
[ ]: [<matplotlib.lines.Line2D at 0x1c7dc7bb8e0>]
```

```
[ ]: <matplotlib.legend.Legend at 0x1c7db823bb0>
```

```
[ ]: Text(0.5, 0, 'Number of Neighbors')
```

```
[ ]: Text(0, 0.5, 'Accuracy')
```



Decision Tree Algorithm

Instantiate dt_entropy, set ‘entropy’ as the information criterion

```
[ ]: dt_entropy = DecisionTreeClassifier(max_depth=8, criterion='entropy',␣
     ↪random_state=1)

     # Fit dt_entropy to the training set

     dt_entropy.fit(X_train, y_train)

     # Use dt_entropy to predict test set labels
     y_pred= dt_entropy.predict(X_test)
```

```
# Evaluate accuracy_entropy
accuracy_entropy = accuracy_score(y_test, y_pred)

# Print accuracy_entropy
print('[Decision Tree -- entropy] accuracy_score: {:.3f}.'.
 ↪format(accuracy_entropy))

# Instantiate dt_gini, set 'gini' as the information criterion
dt_gini = DecisionTreeClassifier(max_depth=8, criterion='gini', random_state=1)

# Fit dt_entropy to the training set
dt_gini.fit(X_train, y_train)

# Use dt_entropy to predict test set labels
y_pred= dt_gini.predict(X_test)

# Evaluate accuracy_entropy
accuracy_gini = accuracy_score(y_test, y_pred)

# Append to the accuracy list
acc=accuracy_gini
accuracy_lst.append(acc)

# Print accuracy_gini
print('[Decision Tree -- gini] accuracy_score: {:.3f}.'.format(accuracy_gini))
```

[ ]: DecisionTreeClassifier(criterion='entropy', max_depth=8, random_state=1)

```
[Decision Tree -- entropy] accuracy_score: 0.754.
```

[ ]: DecisionTreeClassifier(max_depth=8, random_state=1)

```
[Decision Tree -- gini] accuracy_score: 0.754.
```

Random Forest Algorithm

[ ]:
```
# Random Forest algorithm

# Create a Gaussian Classifier

clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)

clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
# Get the accuracy score
acc=accuracy_score(y_test, y_pred)
```

```
# Append to the accuracy list
accuracy_lst.append(acc)

# Model Accuracy, how often is the classifier correct?

print("[Random forest algorithm] accuracy_score: {:.3f}.".format(acc))
```

[ ]:
```
RandomForestClassifier()
```

```
[Random forest algorithm] accuracy_score: 0.735.
```

Random Forest Classifier

[ ]:
```
# Create a selector object that will use the random forest classifier to␣
 ↪identify

# features that have an importance of more than 0.03
sfm = SelectFromModel(clf, threshold=0.03)

# Train the selector
sfm.fit(X_train, y_train)

feat_labels=X.columns

# Print the names of the most important features
for feature_list_index in sfm.get_support(indices=True):
    print(feat_labels[feature_list_index])
```

[ ]:
```
SelectFromModel(estimator=RandomForestClassifier(), threshold=0.03)
```

```
longitude
latitude
PERSONCOUNT
Hour
COLLISIONTYPE_Parked Car
```

Visualizing the important features

[ ]:
```
feature_imp = pd.Series(clf.feature_importances_,index=X.columns).
 ↪sort_values(ascending=False)

# Creating a bar plot, displaying only the top k features
k=10
sns.barplot(x=feature_imp[:10], y=feature_imp.index[:k])
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
```

```
plt.legend()
plt.show()
```

[ ]: <AxesSubplot:>

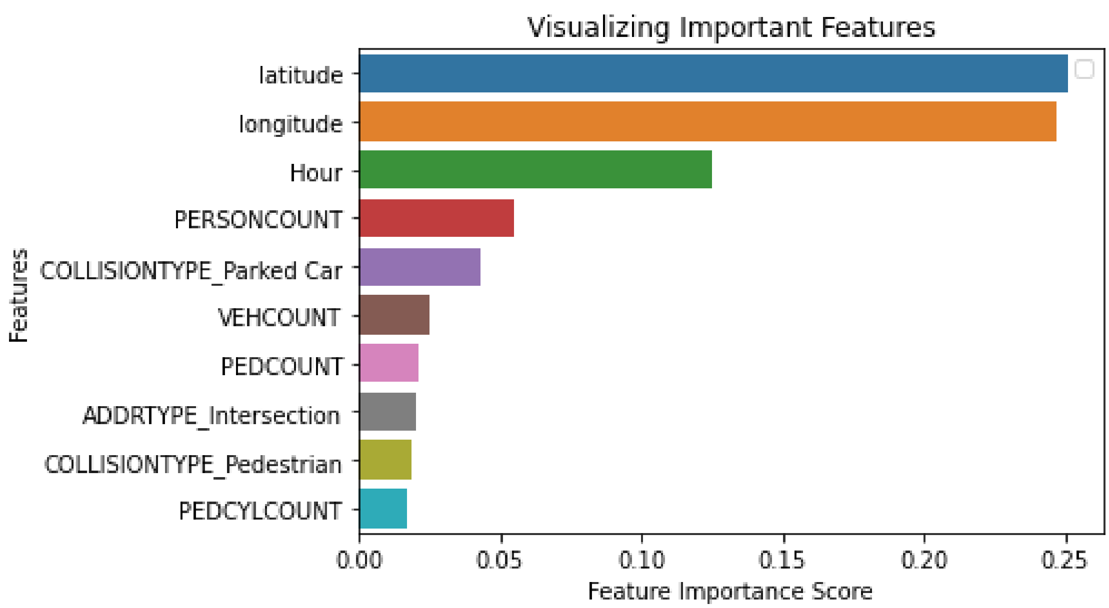
[ ]: Text(0.5, 0, 'Feature Importance Score')

[ ]: Text(0, 0.5, 'Features')

[ ]: Text(0.5, 1.0, 'Visualizing Important Features')

No handles with labels found to put in legend.

[ ]: <matplotlib.legend.Legend at 0x1c78001cac0>



Creating a new Random Forest Classifier for the most important features

```
[ ]: # Transform the data to create a new dataset containing only the most important␣
     ↪features

     # Note: We have to apply the transform to both the training X and test X data.
     X_important_train = sfm.transform(X_train)
     X_important_test = sfm.transform(X_test)

     # Create a new random forest classifier for the most important features
     clf_important = RandomForestClassifier(n_estimators=100, random_state=0,␣
     ↪n_jobs=-1)
```

```
# Train the new classifier on the new dataset containing the most important␣
↪features
clf_important.fit(X_important_train, y_train)
```

[ ]: RandomForestClassifier(n_jobs=-1, random_state=0)

Checking the Accuracy of Random Forest Algorithm with full and the limited features

[ ]:
```
# Apply The Full Featured Classifier To The Test Data
y_pred = clf.predict(X_test)

# View The Accuracy Of Our Full Feature Model
print('[Random forest algorithm -- Full feature] accuracy_score: {:.3f}.'.
↪format(accuracy_score(y_test, y_pred)))

# Apply The Full Featured Classifier To The Test Data
y_important_pred = clf_important.predict(X_important_test)

# View The Accuracy Of Our Limited Feature Model
print('[Random forest algorithm -- Limited feature] accuracy_score: {:.3f}.'.
↪format(accuracy_score(y_test, y_important_pred)))
```

```
[Random forest algorithm -- Full feature] accuracy_score: 0.735.
[Random forest algorithm -- Limited feature] accuracy_score: 0.661.
```

Making a plot of the accuracy scores for different algorithms

[ ]:
```
# Make a plot of the accuracy scores for different algorithms

# Generate a list of ticks for y-axis
y_ticks=np.arange(len(algo_lst))

# Combine the list of algorithms and list of accuracy scores into a dataframe,␣
↪sort the value based on accuracy score
df_acc=pd.DataFrame(list(zip(algo_lst, accuracy_lst)),␣
↪columns=['Algorithm','Accuracy_Score']).
↪sort_values(by=['Accuracy_Score'],ascending = True)

# Export to a file
df_acc.to_csv('./Accuracy_scores_algorithms_{}.csv', index=False)

# Make a plot
ax=df_acc.plot.barh('Algorithm', 'Accuracy_Score',␣
↪align='center',legend=False,color='0.5')

# Add the data label on to the plot
for i in ax.patches:
```

```
    # get_width pulls left or right; get_y pushes up or down
    ax.text(i.get_width()+0.02, i.get_y()+0.2, str(round(i.get_width(),2)),␣
 ↪fontsize=10)

# Set the limit, lables, ticks and title
plt.xlim(0,1.1)
plt.xlabel('Accuracy Score')
plt.yticks(y_ticks, df_acc['Algorithm'], rotation=0)
plt.title('Accuracy Score of each Algorithm')

plt.show()
```

[ ]: Text(0.7546508910793333, -0.04999999999999999, '0.73')

[ ]: Text(0.7589458440981421, 0.95, '0.74')

[ ]: Text(0.7740686863367836, 1.95, '0.75')

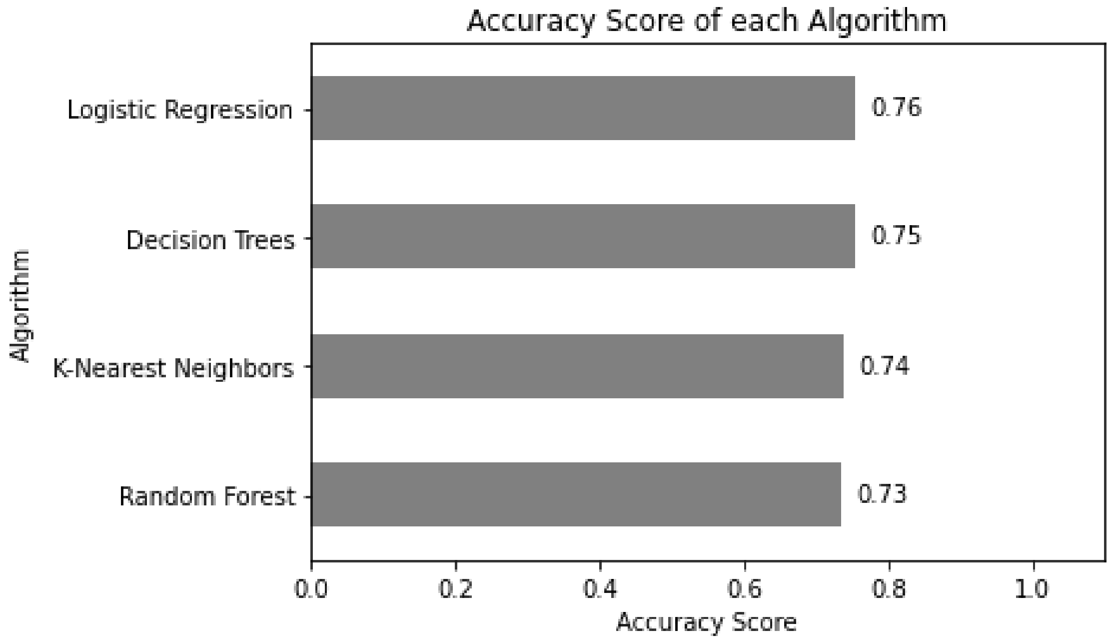
[ ]: Text(0.7757636294821373, 2.95, '0.76')

[ ]: (0.0, 1.1)

[ ]: Text(0.5, 0, 'Accuracy Score')

```
[ ]: ([<matplotlib.axis.YTick at 0x1c7806c6370>,
       <matplotlib.axis.YTick at 0x1c7806c0f70>,
       <matplotlib.axis.YTick at 0x1c7806dbf10>,
       <matplotlib.axis.YTick at 0x1c7806ecf40>],
      [Text(0, 0, 'Random Forest'),
       Text(0, 1, ' K-Nearest Neighbors'),
       Text(0, 2, 'Decision Trees'),
       Text(0, 3, 'Logistic Regression')])
```

[ ]: Text(0.5, 1.0, 'Accuracy Score of each Algorithm')



1.0.7 Deployment

For the deployment phase as it can vary from project to project a simple pdf report has been generated.

1.0.8 Summary

- Seattle road accidents data has been analyzed to get useful insights.
- The data contains multiple attributes e.g. accident severity, collision type, coordinates of the incident, date and time of the incident, weather and road conditions, address types, no of persons injured and property damage and many other attributes.
- There are two accident severity types i.e.
  - Property damage only collision(1)
  - Injury collision(2)
- After the data preparation understanding phase, data preparation phase was carried out by selecting the right features for the machine learning model.
- In the Modeling phase, 4 algorithms were selected where the target class was "accident severity".
- Based on the predictions, "Logistic Regression" relatively performed better among the others with the percentage of 76%.

1.0.9 Conclusion

Based on the selected dataset(features) for this capstone that include mainly, coordinates, hour, person count and the collision type, it can be concluded that these particular classes have a somewhat impact on whether or not travelling along the Seattle roads could result in property damage (class 1) or injury (class 2). In this study, the technique of association rules with a large set of accident data to identify the reasons of road accidents were used.The results show that this model could provide good predictions against traffic accident with 76% correct rate.It should be noted that due to the constraints of data and research condition, there are still some factors, such as engine capacity, traffic flows, gender, age of the driver, attaining the missing data etc. that are not considered in this model and can be taken into account for future study. The results of this study can be used in vehicle safety assistance driving and provide early warnings and proposals for safe driving and hence help in reducing the number of accidents.