CAPSTONE PROJECT

CAR ACCIDENT SEVERITY

SUBMITTED BY

VIVEK KUMAR SONI

TABLE OF CONTENT

CHAPTER: 1
INTRODUCTION

Background:

Every year the lives of approximately 1.35 million people are cut short as a result of a road traffic crash. Between 20 and 50 million more people suffer non-fatal injuries, with many incurring a disability as a result of their injury.

Problem:

Road traffic injuries cause considerable economic losses to individuals, their families, and to nations as a whole. These losses arise from the cost of treatment as well as lost productivity for those killed or disabled by their injuries, and for family members who need to take time off work or school to care for the injured. Road traffic crashes cost most countries 3% of their gross domestic product. If somehow we can predict road accident on basis of different condition then we can reduce risk of accident.

Solution:

In this project i am trying to make machine learning based model which can classify severity of accident I term of injury and property damage on basis of some condition such as road condition, traffic condition, light condition etc.

CHAPTER: 2

DATA

Data which I have used in this project I have collected it from example dataset which can be downloaded from here. This dataset is in comma separated file format (csv). It has 38 columns and 194673 entries. Many of attributes are null values or not defined which must be removed or replaced with values like mean, standard deviation or median according to suitability of data cleaning. Among 38 columns all columns cannot be used for modelling. We need to exploratory analysis and some correlation analysis to select features. I have done some analysis which are shown in form of figure in next I will do data cleaning which will include null value handling, feature selection on basis of different analysis.

CHAPTER: 3

METHODOLOGY

Data understanding:

Data understanding is the knowledge that you have about the data, the needs that the data will satisfy, its content and location. To be clear, it is much more than current location and a definition of what a data element means in situ within an application or data base. Here we can understand about our dataset by applying some method such as describe(), info() etc. results are shown below.

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADCOND	LIGHTCOND	PEI
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

Figure 1 First top rows of data

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADCOND	LIGHTCOND
194668	2	-122.290826	47.565408	219543	309534	310814	E871089	Matched	Block	NaN	...	Dry	Daylight
194669	1	-122.344526	47.690924	219544	309085	310365	E876731	Matched	Block	NaN	...	Wet	Daylight
194670	2	-122.306689	47.683047	219545	311280	312640	3809984	Matched	Intersection	24760.0	...	Dry	Daylight
194671	2	-122.355317	47.678734	219546	309514	310794	3810083	Matched	Intersection	24349.0	...	Dry	Dusk
194672	1	-122.289360	47.611017	219547	308220	309500	E868008	Matched	Block	NaN	...	Wet	Daylight

5 rows × 38 columns

Figure 2 Last rows of dataset

```
x1.dtypes
```

```
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
```

Figure 3 Different datatypes

```
x1.describe()
```

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

Figure 4 Description of dataset

```
x1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 194673 entries, 0 to 194672
Data columns (total 38 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   SEVERITYCODE    194673 non-null  int64
 1   X               189339 non-null  float64
 2   Y               189339 non-null  float64
 3   OBJECTID        194673 non-null  int64
 4   INCKEY          194673 non-null  int64
 5   COLDETKEY       194673 non-null  int64
 6   REPORTNO        194673 non-null  object
 7   STATUS          194673 non-null  object
 8   ADDRTYPE        192747 non-null  object
 9   INTKEY          65070 non-null   float64
 10  LOCATION        191996 non-null  object
 11  EXCEPTRSNCODE   84811 non-null   object
 12  EXCEPTRSNDESC   5638 non-null    object
 13  SEVERITYCODE.1  194673 non-null  int64
 14  SEVERITYDESC    194673 non-null  object
 15  COLLISIONTYPE   189769 non-null  object
 16  PERSONCOUNT     194673 non-null  int64
 17  PEDCOUNT        194673 non-null  int64
 18  PEDCYLCOUNT     194673 non-null  int64
```

Figure 5 Impotent info of dataset

```
x1.columns
```

```
Index(['SEVERITYCODE', 'X', 'Y', 'OBJECTID', 'INCKEY', 'COLDETKEY', 'REPORTNO',
       'STATUS', 'ADDRTYPE', 'INTKEY', 'LOCATION', 'EXCEPTRSNCODE',
       'EXCEPTRSNDESC', 'SEVERITYCODE.1', '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'],
      dtype='object')
```

Figure 6 All columns of dataset

Data preprocessing:

Data preprocessing is an important step of machine learning. Raw data contains different types of noise in terms of missing value, wrong data type, mismatch of data etc. So before going to further step first we should filter our data. In this project I have first removed different categorical dataset for which numerical values are already available in other column. Then I have removed some other columns such as date, id, location etc. which are not relevant in modeling. Then I have converted categorical dataset into numerical values and to overcome 'Nan' value problem I have removed those rows from dataset and formed a new dataset named 'data'. Some basic insight are shown in form of fig. below.

Basic insight of data for machine learning

```
data.head()
```

	ADDRTYPE	COLLISIONTYPE	PERSONCOUNT	VEHCOUNT	SDOT_COLCODE	WEATHER	ROADCOND	LIGHTCOND	SEVERITYCODE
0	2.0	0.0	2	2	11	4.0	8.0	5.0	2
1	1.0	9.0	2	2	16	6.0	8.0	2.0	1
2	1.0	5.0	4	3	14	4.0	0.0	5.0	1
3	1.0	4.0	3	3	11	1.0	0.0	5.0	1
4	2.0	0.0	2	2	11	6.0	8.0	5.0	2

```
data.tail()
```

	ADDRTYPE	COLLISIONTYPE	PERSONCOUNT	VEHCOUNT	SDOT_COLCODE	WEATHER	ROADCOND	LIGHTCOND	SEVERITYCODE
194668	1.0	2.0	3	2	11	1.0	0.0	5.0	2
194669	1.0	7.0	2	2	14	6.0	8.0	5.0	1
194670	2.0	3.0	3	2	11	1.0	0.0	5.0	2
194671	2.0	1.0	2	1	51	1.0	0.0	6.0	2
194672	1.0	7.0	2	2	14	1.0	8.0	5.0	1

Figure 7 Top and bottom rows of real dataset

```
data.describe()
```

	ADDRTYPE	COLLISIONTYPE	PERSONCOUNT	VEHCOUNT	SDOT_COLCODE	WEATHER	ROADCOND	LIGHTCOND	SEVERITYCODE
count	192747.000000	189769.000000	194673.000000	194673.000000	194673.000000	189592.000000	189661.000000	189503.000000	194673.000000
mean	1.333697	4.504034	2.444427	1.920780	13.867768	3.083843	2.599802	4.399825	1.298901
std	0.479726	2.784029	1.345929	0.631047	6.868755	2.855272	3.651150	1.713750	0.457778
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	1.000000	3.000000	2.000000	2.000000	11.000000	1.000000	0.000000	2.000000	1.000000
50%	1.000000	5.000000	2.000000	2.000000	13.000000	1.000000	0.000000	5.000000	1.000000
75%	2.000000	7.000000	3.000000	2.000000	14.000000	6.000000	8.000000	5.000000	2.000000
max	2.000000	9.000000	81.000000	12.000000	69.000000	10.000000	8.000000	8.000000	2.000000

Figure 8 Basic description of real dataset

```
data.dtypes
```

```
ADDRTYPE         float64
COLLISIONTYPE    float64
PERSONCOUNT        int64
VEHCOUNT           int64
SDOT_COLCODE       int64
WEATHER          float64
ROADCOND         float64
LIGHTCOND        float64
SEVERITYCODE       int64
dtype: object
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 194673 entries, 0 to 194672
Data columns (total 9 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   ADDRTYPE       192747 non-null  float64
 1   COLLISIONTYPE  189769 non-null  float64
 2   PERSONCOUNT    194673 non-null  int64
 3   VEHCOUNT       194673 non-null  int64
 4   SDOT_COLCODE   194673 non-null  int64
 5   WEATHER        189592 non-null  float64
 6   ROADCOND       189661 non-null  float64
 7   LIGHTCOND      189503 non-null  float64
 8   SEVERITYCODE   194673 non-null  int64
dtypes: float64(5), int64(4)
memory usage: 13.4 MB
```

Figure 9 Datatype and info of real dataset

Exploratory analysis:

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

Descriptive analysis:

Descriptive statistics are used to describe the basic features of the data in a study. They provide simple summaries about the sample and the measures. Together with simple graphics analysis, they form the basis of virtually every quantitative analysis of data.

1. Descriptive analysis

```
d1=data['ADDRTYPE'].value_counts()
d1
```

```
1.0     123315
2.0      63447
0.0        742
Name: ADDRTYPE, dtype: int64
```

```
d2=data['COLLISIONTYPE'].value_counts()
d2
```

```
5.0     46679
0.0     34555
7.0     33794
4.0     23440
9.0     18442
3.0     13659
6.0      6589
1.0      5399
```

Figure 10 Descriptive analysis result of two feature

Boxplot Analysis:

A box plot (also known as box and whisker plot) is a type of chart often used in explanatory data analysis to visually show the distribution of numerical data and skewness through displaying the data quartiles (or percentiles) and averages.

2. Box Plot Analysis

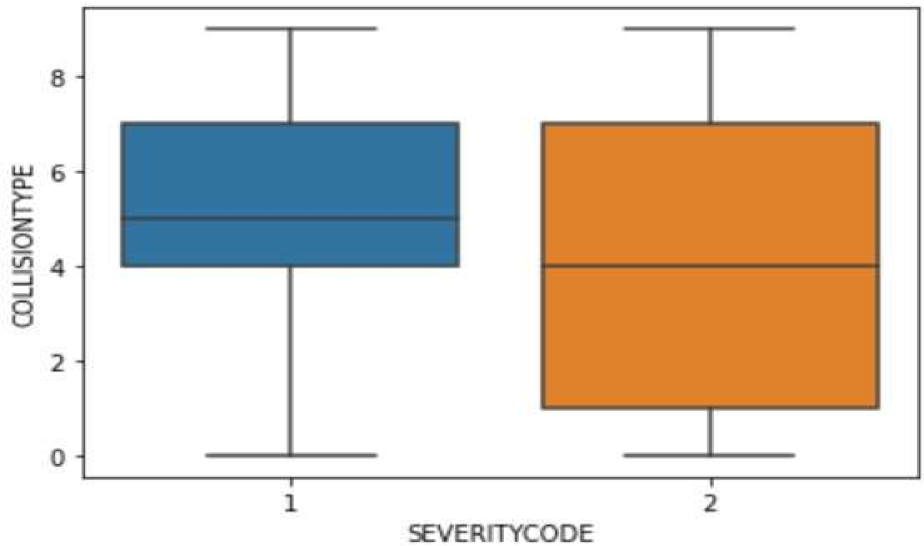
```
import seaborn as sns
```

```
#B1
sns.boxplot(x=data['SEVERITYCODE'],y=data['ADDRTYPE'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x27401e268b0>

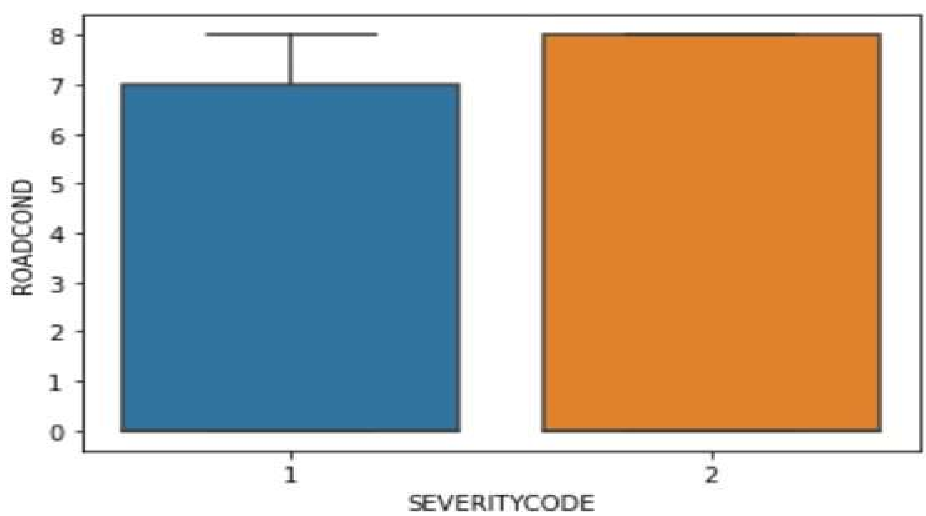
```python
#B2
sns.boxplot(x=data['SEVERITYCODE'],y=data['COLLISIONTYPE'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x27401e26c10>




```python
#B7
sns.boxplot(x=data['SEVERITYCODE'],y=data['ROADCOND'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x274077f5b20>



GroupBy Analysis:

As the name suggests it should group your data into groups. In this case, it will group it into three groups representing different flower species (our target values).

3. GroupBy Analysis

```
g1=data.groupby(['ADDRTYPE'])['SEVERITYCODE'].value_counts(normalize=True)
g1
```

```
ADDRTYPE  SEVERITYCODE
0.0       1               0.892183
          2               0.107817
1.0       1               0.761367
          2               0.238633
2.0       1               0.568727
          2               0.431273
Name: SEVERITYCODE, dtype: float64
```

```
g8=data.groupby(['LIGHTCOND'])['SEVERITYCODE'].value_counts(normalize=True)
g8
```

```
LIGHTCOND  SEVERITYCODE
0.0        1               0.780984
           2               0.219016
1.0        1               0.733953
           2               0.266047
2.0        1               0.701097
           2               0.298903
3.0        1               0.636364
           2               0.363636
4.0        1               0.669478
           2               0.330522
5.0        1               0.667230
           2               0.332770
6.0        1               0.668607
           2               0.331393
7.0        1               0.770925
           2               0.229075
8.0        1               0.953243
           2               0.046757
Name: SEVERITYCODE, dtype: float64
```

Pearson correlation analysis:

Correlation is a technique for investigating the relationship between two quantitative, continuous variables, for example, age and blood pressure. Pearson's correlation coefficient (r) is a measure of the strength of the association between the two variables.

4. Pearson correlation analysis

```
from scipy import stats
```

```
from scipy.stats import pearsonr
```

```
#p1
pearson_coef,p_value=stats.pearsonr(data['ADDRTYPE'],data['SEVERITYCODE'])
pearson_coef,p_value
```

```
(0.19971784115718683, 0.0)
```

```
#p2
pearson_coef,p_value=stats.pearsonr(data['COLLISIONTYPE'],data['SEVERITYCODE'])
pearson_coef,p_value
```

```
(-0.12834127033207823, 0.0)
```

```
#p3
pearson_coef,p_value=stats.pearsonr(data['PERSONCOUNT'],data['SEVERITYCODE'])
pearson_coef,p_value
```

```
(0.12836812235055656, 0.0)
```

Modeling by different machine learning technique:

1. Decision tree

A decision tree is a flowchart-like structure in which each internal node represents a test on a feature (e.g. whether a coin flip comes up heads or tails) , each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2, random_state=4)
```

```
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
DT.fit(x_train,y_train)
yhat = DT.predict(x_test)
```

```
from sklearn import metrics
print("Accuracy: ", metrics.accuracy_score(y_test, yhat))
```

```
Accuracy:  0.7509666408895763
```

2. Logistic regression:

Logistic regression is a classification algorithm, used when the value of the target variable is categorical in nature. Logistic regression is most commonly used when the data in question has binary output, so when it belongs to one class or another, or is either a 0 or 1.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=0.01, solver='liblinear').fit(x_train,y_train)
yhat = LR.predict(x_test)
yhat_prob = LR.predict_proba(x_test)
```

```
C:\Users\soniv\Anaconda3\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

```
from sklearn.metrics import log_loss
log_loss(y_test, yhat_prob)
```

```
0.5579350863787483
```

3. KNN:

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification. KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

In this method first I have defined best 'k' value by calculating accuracy then I have made final model using that 'k' value.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2, random_state=4)
```

```
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
K = 10
mean_acc = np.zeros((K-1))
ConfustionMx = [];
for n in range(1,K):
    model1 = KNeighborsClassifier(n_neighbors = n).fit(x_train,y_train)
    yhat=model1.predict(x_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)
mean_acc
```

```
array([0.70590118, 0.73686035, 0.72704728, 0.74155356, 0.73382043,
       0.74459348, 0.74014026, 0.75099331, 0.7476334 ])
```

CHAPTER: 4

RESULT

After modeling different machine learning technique final step is to test it for new data set I have done it in previous part. After that I have calculated different parameters using confusion matrix for different machine learning algorithm which describes accuracy, precision, f_score, recall etc.

Result of Decision tree:

```
[[24383  1813]
 [ 7526  3779]]
              precision    recall  f1-score   support

           1       0.76      0.93      0.84     26196
           2       0.68      0.33      0.45     11305

    accuracy                           0.75     37501
   macro avg       0.72      0.63      0.64     37501
weighted avg       0.74      0.75      0.72     37501
```

Result of Logistic regression:

```
[[24780  1416]
 [ 8975  2330]]
              precision    recall  f1-score   support

           1       0.73      0.95      0.83     26196
           2       0.62      0.21      0.31     11305

    accuracy                           0.72     37501
   macro avg       0.68      0.58      0.57     37501
weighted avg       0.70      0.72      0.67     37501
```

Result of KNN:

```
[[24416  1780]
 [ 7558  3747]]
              precision    recall  f1-score   support

           1       0.76      0.93      0.84     26196
           2       0.68      0.33      0.45     11305

    accuracy                           0.75     37501
   macro avg       0.72      0.63      0.64     37501
weighted avg       0.74      0.75      0.72     37501
```

CHAPTER: 5

DISCUSSION

In this project I have made model using only few features to reduce complexity but we can also choose some other features and other machine learning algorithm. Future scope for this project can be making of model using ‘Neural network analysis’ because NNA is superior technique so we can go for it.

CHAPTER: 6

CONCLUSION

In this project I have made three model namely Decision tree, Logistic regression and KNN. On basis of different analysis and result I can conclude that KNN is best classifier for this model. This is because KNN gives best result if number of classes are two. But accuracy of Decision tree was also good.