Car Accident Severity

Sweta Swarupa

29/01/22

Contents

1 Introduction

The aim of this project is to predict car accident severity. The data has been taken from the Traffic Records Group, Traffic Management Division, Seattle Department of Transportation. It comprises of all collisions and crashes that have occurred in the state from 2004 to 2019.The data has the key information that include Severity of the accident, Incident Date, number of vehicles and persons involved in accidents, Road type and conditions, Weather and light conditions.

1.1 Importing needed packages

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix
```

1.2 Reading data

```
import pandas as pd
data = pd.read_csv("C:\\Users\\sswarupa\\Documents\\GitHub\\CarAccidentSeverity\\Data-Collisions.csv", low
data.head()
```

```
##    SEVERITYCODE           X          Y  ...  SEGLANEKEY  CROSSWALKKEY  HITPARKEDCAR
## 0             2 -122.323148  47.703140  ...           0             0             N
## 1             1 -122.347294  47.647172  ...           0             0             N
## 2             1 -122.334540  47.607871  ...           0             0             N
## 3             1 -122.334803  47.604803  ...           0             0             N
## 4             2 -122.306426  47.545739  ...           0             0             N
##
## [5 rows x 38 columns]
```

1.3 Checking data type of the variables

```
data.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
## 19 VEHCOUNT 194673 non-null int64
## 20 INCDATE 194673 non-null object
## 21 INCDTTM 194673 non-null object
## 22 JUNCTIONTYPE 188344 non-null object
## 23 SDOT_COLCODE 194673 non-null int64
## 24 SDOT_COLDESC 194673 non-null object
## 25 INATTENTIONIND 29805 non-null object
## 26 UNDERINFL 189789 non-null object
## 27 WEATHER 189592 non-null object
## 28 ROADCOND 189661 non-null object
## 29 LIGHTCOND 189503 non-null object
## 30 PEDROWNOTGRNT 4667 non-null object
## 31 SDOTCOLNUM 114936 non-null float64
## 32 SPEEDING 9333 non-null object
## 33 ST_COLCODE 194655 non-null object
## 34 ST_COLDESC 189769 non-null object
## 35 SEGLANEKEY 194673 non-null int64
## 36 CROSSWALKKEY 194673 non-null int64
## 37 HITPARKEDCAR 194673 non-null object
## dtypes: float64(4), int64(12), object(22)
## memory usage: 56.4+ MB
```

There are 194673 rows and 38 columns. There are several variables in the dataset which looks like are ids and not useful for our model.

1.4 Checking count of unique values of each variables

```
counts = data.nunique()
counts
```

```
## SEVERITYCODE           2
## X                  23563
## Y                  23839
## OBJECTID          194673
## INCKEY            194673
## COLDETKEY         194673
## REPORTNO          194670
## STATUS                 2
## ADDRTYPE               3
## INTKEY              7614
```

```
## LOCATION           24102
## EXCEPTRSNCODE          2
## EXCEPTRSNDESC          1
## SEVERITYCODE.1         2
## SEVERITYDESC           2
## COLLISIONTYPE         10
## PERSONCOUNT           47
## PEDCOUNT               7
## PEDCYLCOUNT            3
## VEHCOUNT              13
## INCDATE             5985
## INCDTTM           162058
## JUNCTIONTYPE           7
## SDOT_COLCODE          39
## SDOT_COLDESC          39
## INATTENTIONIND         1
## UNDERINFL              4
## WEATHER               11
## ROADCOND               9
## LIGHTCOND              9
## PEDROWNOTGRNT          1
## SDOTCOLNUM        114932
## SPEEDING               1
## ST_COLCODE            63
## ST_COLDESC            62
## SEGLANEKEY          1955
## CROSSWALKKEY        2198
## HITPARKEDCAR           2
## dtype: int64
```

1.5 Selecting the variables useful for the model

```
colli = data.iloc[:,[0,8,15,20,22,27,28,29,37]]
colli.head()
```

```
##    SEVERITYCODE      ADDRTYPE  ...                LIGHTCOND HITPARKEDCAR
## 0             2  Intersection  ...                 Daylight            N
## 1             1         Block  ...  Dark - Street Lights On            N
## 2             1         Block  ...                 Daylight            N
## 3             1         Block  ...                 Daylight            N
## 4             2  Intersection  ...                 Daylight            N
##
## [5 rows x 9 columns]
```

I chose the severity of the accidents as the dependent variable. SEVERITYCODE is a categorical variable and follows a code that corresponds to the severity of the collision: 2 (injury) and 1 (property damage). I chose 8 attributes from the 37 available in the dataset: address type, collision type, incident date, junction type, weather, road condition, light condition and hit parked car.

1.6 Checking for any missing values

```
#Checking for the null values
colli.isnull().sum()
```

```
## SEVERITYCODE        0
## ADDRTYPE         1926
## COLLISIONTYPE    4904
## INCDATE             0
## JUNCTIONTYPE     6329
## WEATHER          5081
## ROADCOND         5012
## LIGHTCOND        5170
## HITPARKEDCAR        0
## dtype: int64
```

1.7 Removing rows with any missing values and checking the final dataset

```
# using dropna() function
mod_colli = colli.dropna(axis=0, how='any')
mod_colli.shape
```

```
## (182895, 9)
```

```
mod_colli.isnull().sum()
```

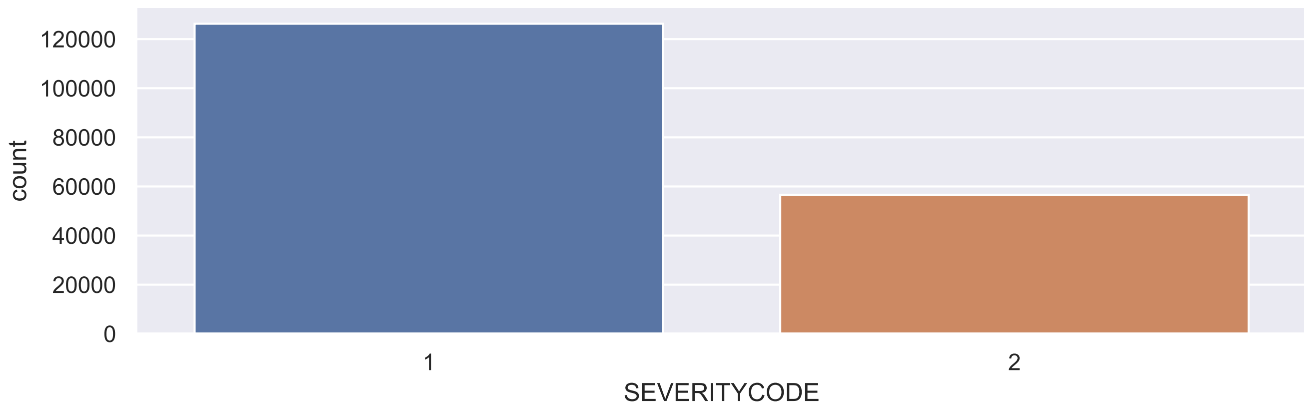
```
## SEVERITYCODE     0
## ADDRTYPE         0
## COLLISIONTYPE    0
## INCDATE          0
## JUNCTIONTYPE     0
## WEATHER          0
## ROADCOND         0
## LIGHTCOND        0
## HITPARKEDCAR     0
## dtype: int64
```

Removed rows with any missing values. 11,778 rows were deleted. The final dataset has 182895 rows, 9 columns and there are no missing values.

2 Data Analysis

2.1 Checking the target variable

```
import seaborn as sns
sns.set()
sns.catplot(x='SEVERITYCODE', kind='count', data=mod_colli, height=3, aspect=3)
```



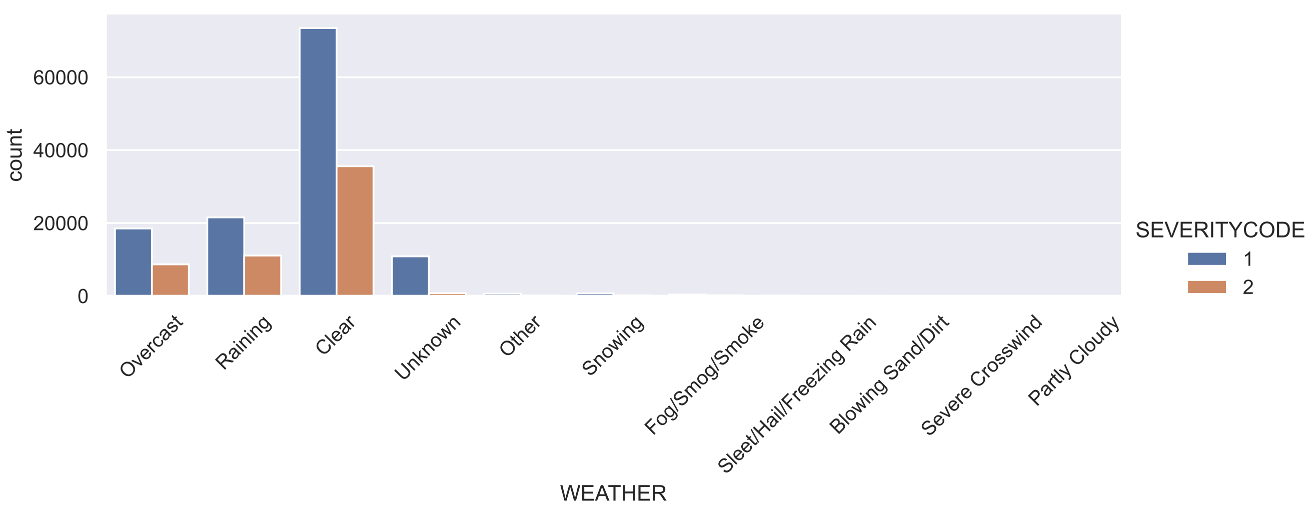

```
mod_colli['SEVERITYCODE'].value_counts()
```

```
## 1    126270
## 2     56625
## Name: SEVERITYCODE, dtype: int64
```

The target data looks imbalanced, however there are sufficient rows with SEVERITYCODE 2 to train the model. There are 12K rows with SEVERITYCODE 1 and only 5K rows with SEVERITYCODE 2. I tried balancing the target by simple random sampling however it did not help in improving the accuracy so excluding that step.

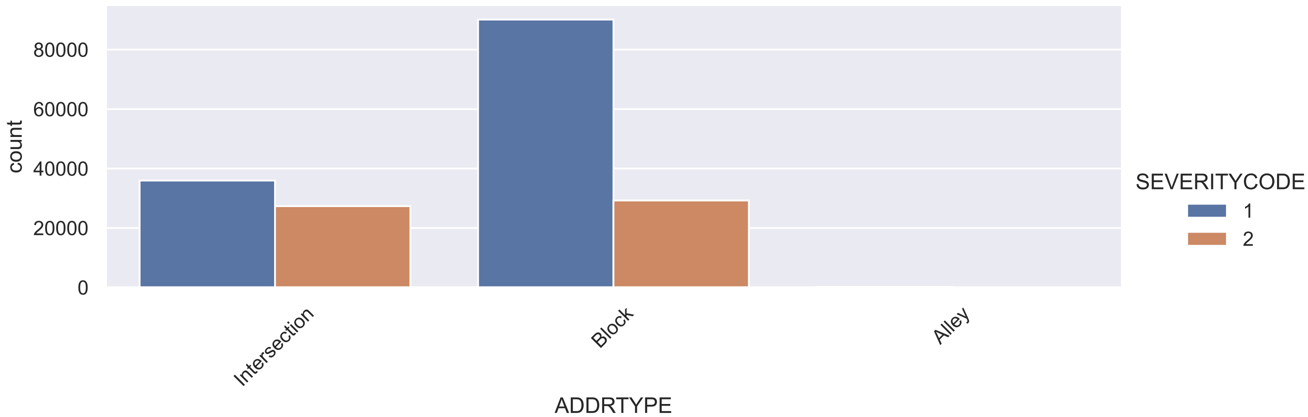
2.2 Plotting all features by severity code

```
#Plotting Weather by Severity
import seaborn as sns
import matplotlib.pyplot as plt
chart = sns.catplot(x='WEATHER', kind='count', hue='SEVERITYCODE', data=mod_colli, height=3, aspect=3)
chart.set_xticklabels(rotation=45)
```



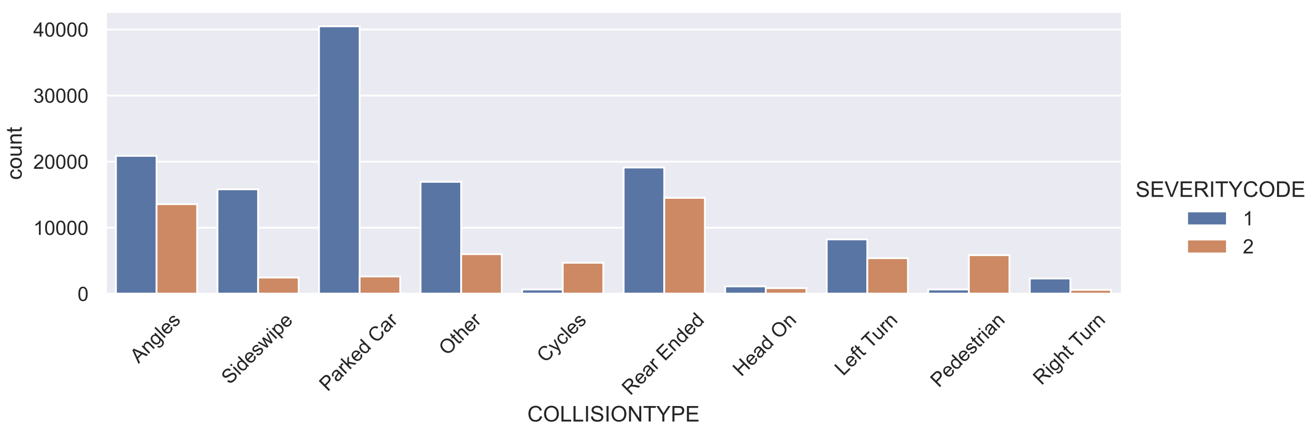
Most of the accidents happened with weather type clear, raining and overcast.

```
#Plotting Address type by Severity
import seaborn as sns
chart = sns.catplot(x='ADDRTYPE', kind='count', hue='SEVERITYCODE', data=mod_colli, height=3, aspect=3)
chart.set_xticklabels(rotation=45)
```



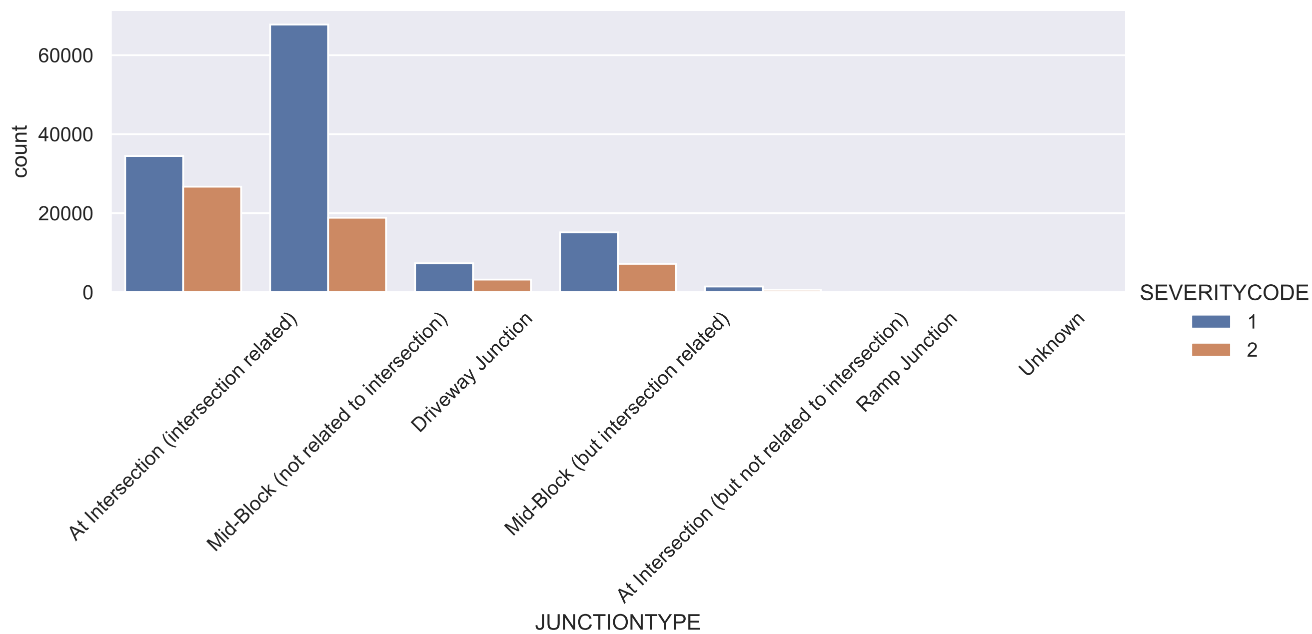
Most of the accidents happened on the block however the proportion of severity code 2 is significantly high on intersection.

```
#Plotting Collision type by Severity
import seaborn as sns
chart = sns.catplot(x='COLLISIONTYPE', kind='count', hue='SEVERITYCODE', data=mod_colli, height=3, aspect=
chart.set_xticklabels(rotation=45)
```

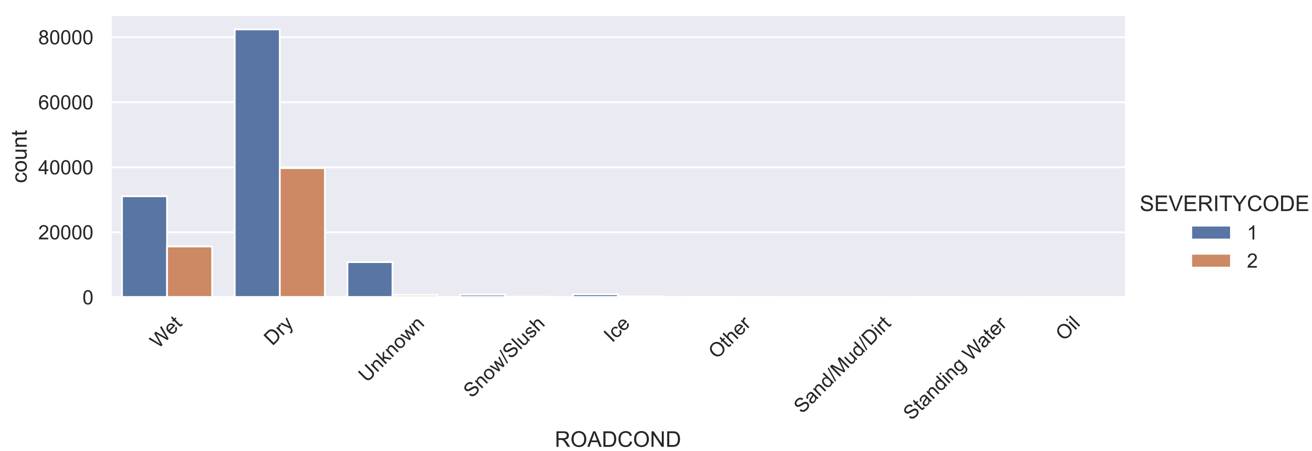


Most of the accidents happened with collision type parked car however they are mostly severity code 1. Most of the severity code 2 accidents happened with collision type rear ended and Angles.

```
#Plotting Junction type by Severity
import seaborn as sns
chart = sns.catplot(x='JUNCTIONTYPE', kind='count', hue='SEVERITYCODE', data=mod_colli, height=3, aspect=3
chart.set_xticklabels(rotation=45)
```

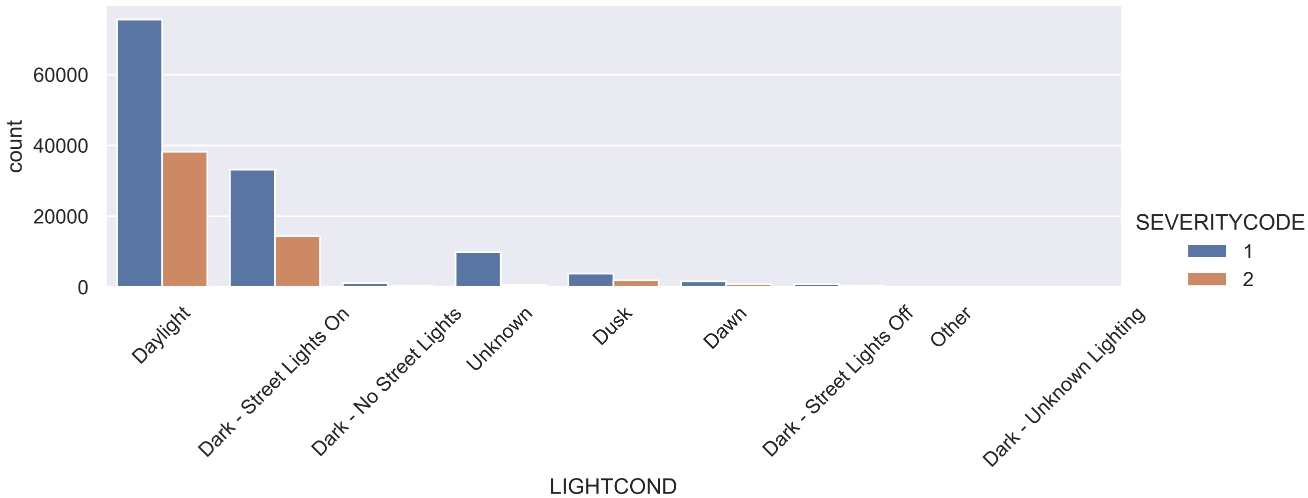



```
#Plotting Road Condition by Severity
import seaborn as sns
chart = sns.catplot(x='ROADCOND', kind='count', hue='SEVERITYCODE', data=mod_colli, height=3, aspect=3)
chart.set_xticklabels(rotation=45)
```



Most of the accidents happened on wet and dry road conditions suggesting that traffic must be low on the riskier conditions.

```
#Plotting Light Condition by Severity
import seaborn as sns
chart = sns.catplot(x='LIGHTCOND', kind='count', hue='SEVERITYCODE', data=mod_colli, height=3, aspect=3)
chart.set_xticklabels(rotation=45)
```




```
#Plotting Hit parked car by Severity
import seaborn as sns
chart = sns.catplot(x='HITPARKEDCAR', kind='count', hue='SEVERITYCODE', data=mod_colli, height=3, aspect=3
```

Most of the accidents happened when the HITPARKEDCAR type 'N'.

```
#Exacting Month from Incident Date and replacing incident date with incident month
mod_colli['INCMONTH'] = pd.DatetimeIndex(mod_colli['INCDATE']).month
```

```
## <string>:1: SettingWithCopyWarning:
## A value is trying to be set on a copy of a slice from a DataFrame.
## Try using .loc[row_indexer,col_indexer] = value instead
##
## See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
```

```
del mod_colli['INCDATE']
mod_colli.head()
```

```
##    SEVERITYCODE      ADDRTYPE  ... HITPARKEDCAR INCMONTH
## 0             2  Intersection  ...            N        3
## 1             1         Block  ...            N       12
## 2             1         Block  ...            N       11
## 3             1         Block  ...            N        3
## 4             2  Intersection  ...            N        1
##
## [5 rows x 9 columns]
```

```
#Plotting Incident Month by Severity
import seaborn as sns
chart = sns.catplot(x='INCMONTH', kind='count', hue='SEVERITYCODE', data=mod_colli, height=3, aspect=3)
```

Plotted month of the incident to see if there is any seasonal pattern however doesn't look like there is any seasonal pattern.

3 Data Transformation

3.1 Encoding the features into numerical values

```
from sklearn import preprocessing

severity = preprocessing.LabelEncoder()
severity.fit(mod_colli['SEVERITYCODE'])
```

```
## LabelEncoder()
```

```
mod_colli['SEVERITYCODE'] = severity.transform(mod_colli['SEVERITYCODE'])
```

```
## <string>:1: SettingWithCopyWarning:
## A value is trying to be set on a copy of a slice from a DataFrame.
## Try using .loc[row_indexer,col_indexer] = value instead
##
## See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
```

```
addrtype = preprocessing.LabelEncoder()
severity.fit(mod_colli['ADDRTYPE'])
```

```
## LabelEncoder()
```

```
mod_colli['ADDRTYPE'] = severity.transform(mod_colli['ADDRTYPE'])

collisiontype = preprocessing.LabelEncoder()
severity.fit(mod_colli['COLLISIONTYPE'])
```

```
## LabelEncoder()
```

```
mod_colli['COLLISIONTYPE'] = severity.transform(mod_colli['COLLISIONTYPE'])

junctiontype = preprocessing.LabelEncoder()
severity.fit(mod_colli['JUNCTIONTYPE'])
```

```
## LabelEncoder()
```

```
mod_colli['JUNCTIONTYPE'] = severity.transform(mod_colli['JUNCTIONTYPE'])

weather = preprocessing.LabelEncoder()
severity.fit(mod_colli['WEATHER'])
```

```
## LabelEncoder()
```

```
mod_colli['WEATHER'] = severity.transform(mod_colli['WEATHER'])

roadcond = preprocessing.LabelEncoder()
severity.fit(mod_colli['ROADCOND'])
```

```
## LabelEncoder()
```

```
mod_colli['ROADCOND'] = severity.transform(mod_colli['ROADCOND'])

lightcond = preprocessing.LabelEncoder()
severity.fit(mod_colli['LIGHTCOND'])
```

```
## LabelEncoder()
```

```
mod_colli['LIGHTCOND'] = severity.transform(mod_colli['LIGHTCOND'])

hitparkedcar = preprocessing.LabelEncoder()
severity.fit(mod_colli['HITPARKEDCAR'])
```

```
## LabelEncoder()
```

```
mod_colli['HITPARKEDCAR'] = severity.transform(mod_colli['HITPARKEDCAR'])

incmonth = preprocessing.LabelEncoder()
severity.fit(mod_colli['INCMONTH'])
```

```
## LabelEncoder()
```

```
mod_colli['INCMONTH'] = severity.transform(mod_colli['INCMONTH'])

mod_colli.head()
```

```
##    SEVERITYCODE  ADDRTYPE  COLLISIONTYPE  ...  LIGHTCOND  HITPARKEDCAR  INCMONTH
## 0             1         2              0  ...          5             0         2
## 1             0         1              9  ...          2             0        11
## 2             0         1              5  ...          5             0        10
## 3             0         1              4  ...          5             0         2
## 4             1         2              0  ...          5             0         0
##
## [5 rows x 9 columns]
```

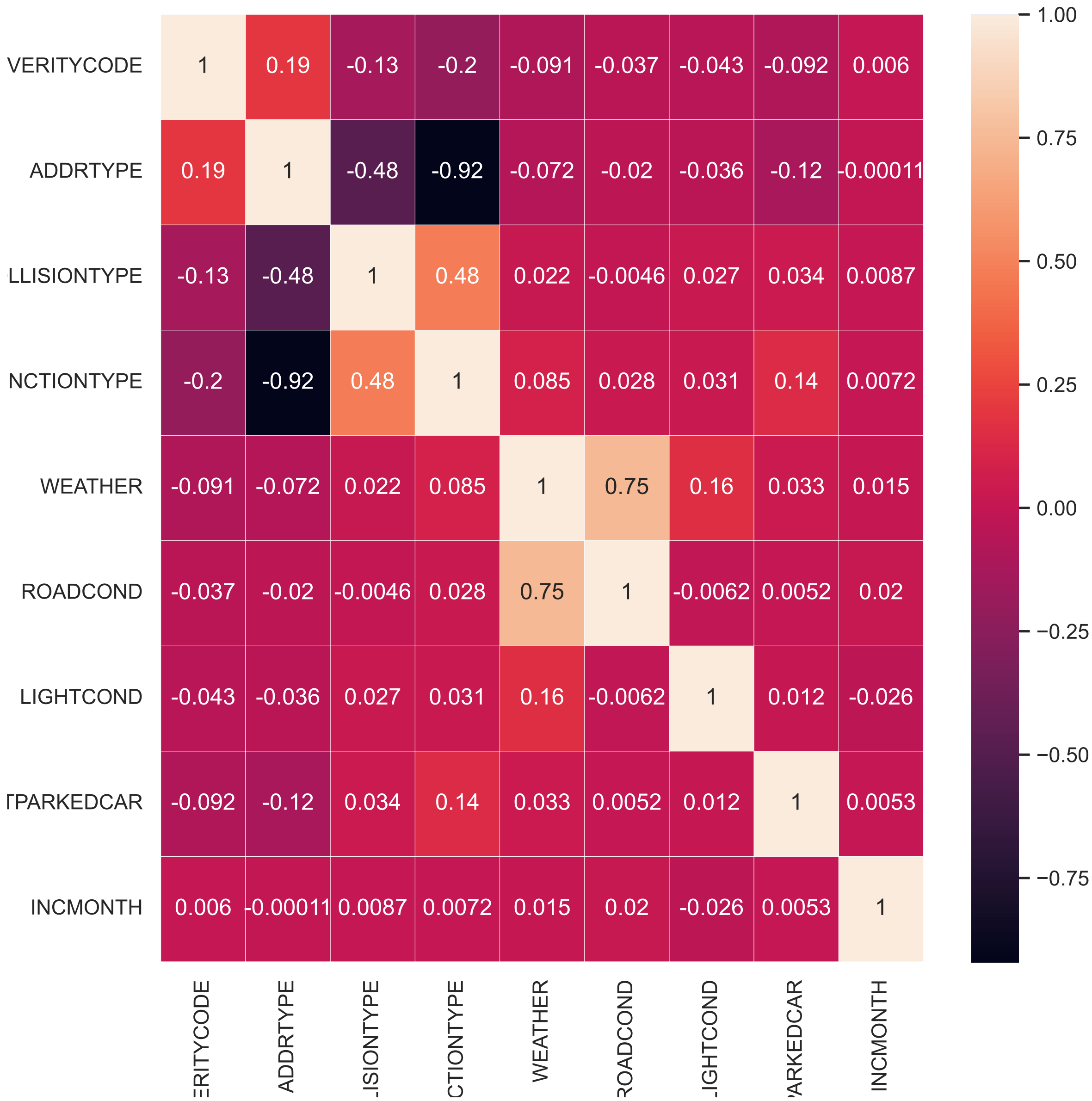
As we can see above that all categorical variables are now converted into numerical values.

4 Understanding the correlation between variables

```
mod_colli.corr()
```

```
##                SEVERITYCODE  ADDRTYPE  ...  HITPARKEDCAR  INCMONTH
## SEVERITYCODE       1.000000  0.192744  ...     -0.092478  0.005963
## ADDRTYPE           0.192744  1.000000  ...     -0.118982 -0.000107
## COLLISIONTYPE     -0.127117 -0.479831  ...      0.033586  0.008729
## JUNCTIONTYPE      -0.200887 -0.919606  ...      0.143587  0.007152
## WEATHER           -0.090768 -0.071629  ...      0.033383  0.015249
## ROADCOND          -0.037238 -0.019876  ...      0.005173  0.019777
## LIGHTCOND         -0.042661 -0.035628  ...      0.012456 -0.026348
## HITPARKEDCAR      -0.092478 -0.118982  ...      1.000000  0.005310
## INCMONTH           0.005963 -0.000107  ...      0.005310  1.000000
##
## [9 rows x 9 columns]
```

```
import seaborn as sns
plt.figure(figsize=(9,9))
sns.heatmap(mod_colli.corr(), annot = True, linewidth=.2, cbar_kws={"shrink":1})
plt.show()
```



5 Splitting data into training and test dataset

```
from sklearn.model_selection import train_test_split
X = mod_colli.iloc[:,[1,2,3,4,5,6,7,8]]
y = mod_colli.iloc[:,0]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=40)
print('Train set:', X_train.shape, y_train.shape)
```

```
## Train set: (128026, 8) (128026,)
```

```
print('Test set:', X_test.shape, y_test.shape)
```

```
## Test set: (54869, 8) (54869,)
```

6 KNN Model

6.1 Building Model and Predicting on the test dataset

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
import numpy as np

#Training
k = 4
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh

# Predicting
```

```
## KNeighborsClassifier(n_neighbors=4)
```

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

```
## array([0, 1, 0, 0, 0], dtype=int64)
```

6.2 Accuracy Evaluation of the Model

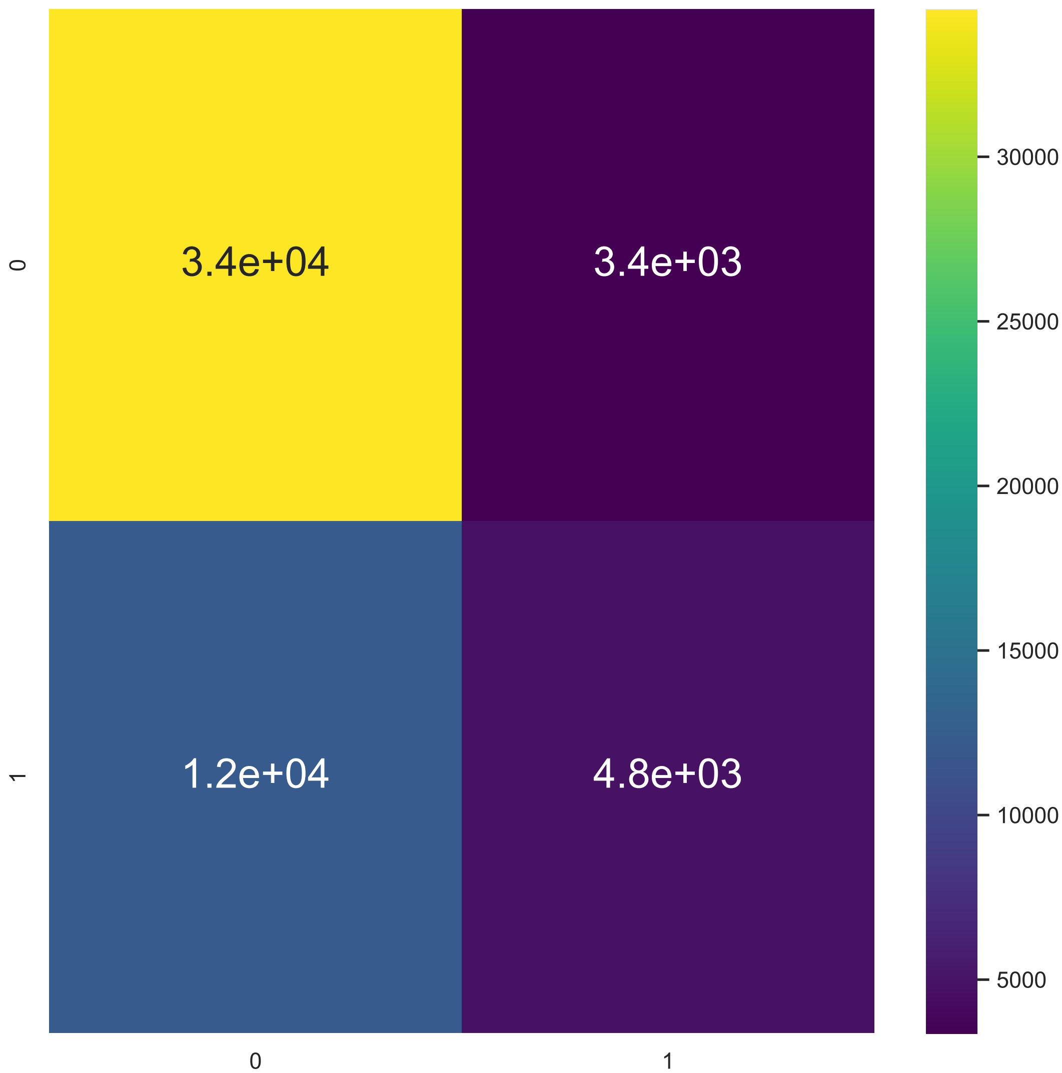
```
from sklearn import metrics
print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(X_train)))
```

```
## Train set Accuracy:  0.7317576117351163
```

```
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))
```

```
## Test set Accuracy:  0.7161056334177769
```

```
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
con_mat=confusion_matrix(y_test,yhat)
sns.heatmap(con_mat,annot=True,annot_kws=
                           {"size":20},cmap="viridis")
plt.show()
```




```
print(classification_report(y_test, yhat))
```

```
##               precision    recall  f1-score   support
##
##            0       0.74      0.91      0.82     37859
##            1       0.59      0.28      0.38     17010
##
##     accuracy                           0.72     54869
##    macro avg       0.66      0.60      0.60     54869
## weighted avg       0.69      0.72      0.68     54869
```

6.3 Lets try with K=6

```
k2 = 6
n2 = KNeighborsClassifier(n_neighbors=k2).fit(X_train,y_train)
n2
```

```
## KNeighborsClassifier(n_neighbors=6)
```

```
yhat2 = n2.predict(X_test)
yhat2[0:5]
```

```
## array([0, 1, 0, 0, 0], dtype=int64)
```

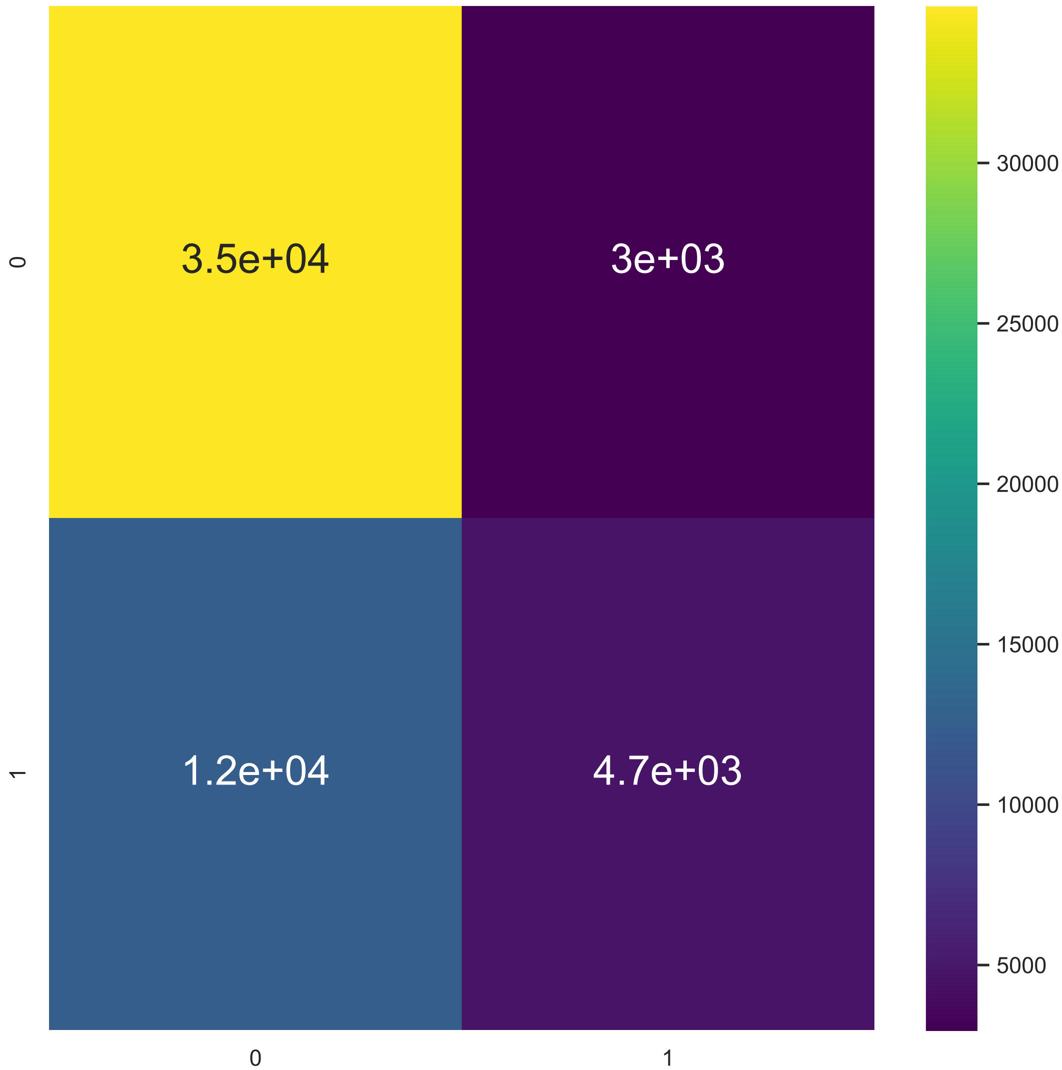
```
print("Train Accuracy:",metrics.accuracy_score(y_train,n2.predict(X_train)))
```

```
## Train Accuracy: 0.7346085951291144
```

```
print("Test Accuracy:",metrics.accuracy_score(y_test,yhat2))
```

```
## Test Accuracy: 0.7207348411671436
```

```
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
con_mat=confusion_matrix(y_test,yhat2)
sns.heatmap(con_mat,annot=True,annot_kws=
                           {"size":20},cmap="viridis")
plt.show()
```




```
print(classification_report(y_test, yhat2))
```

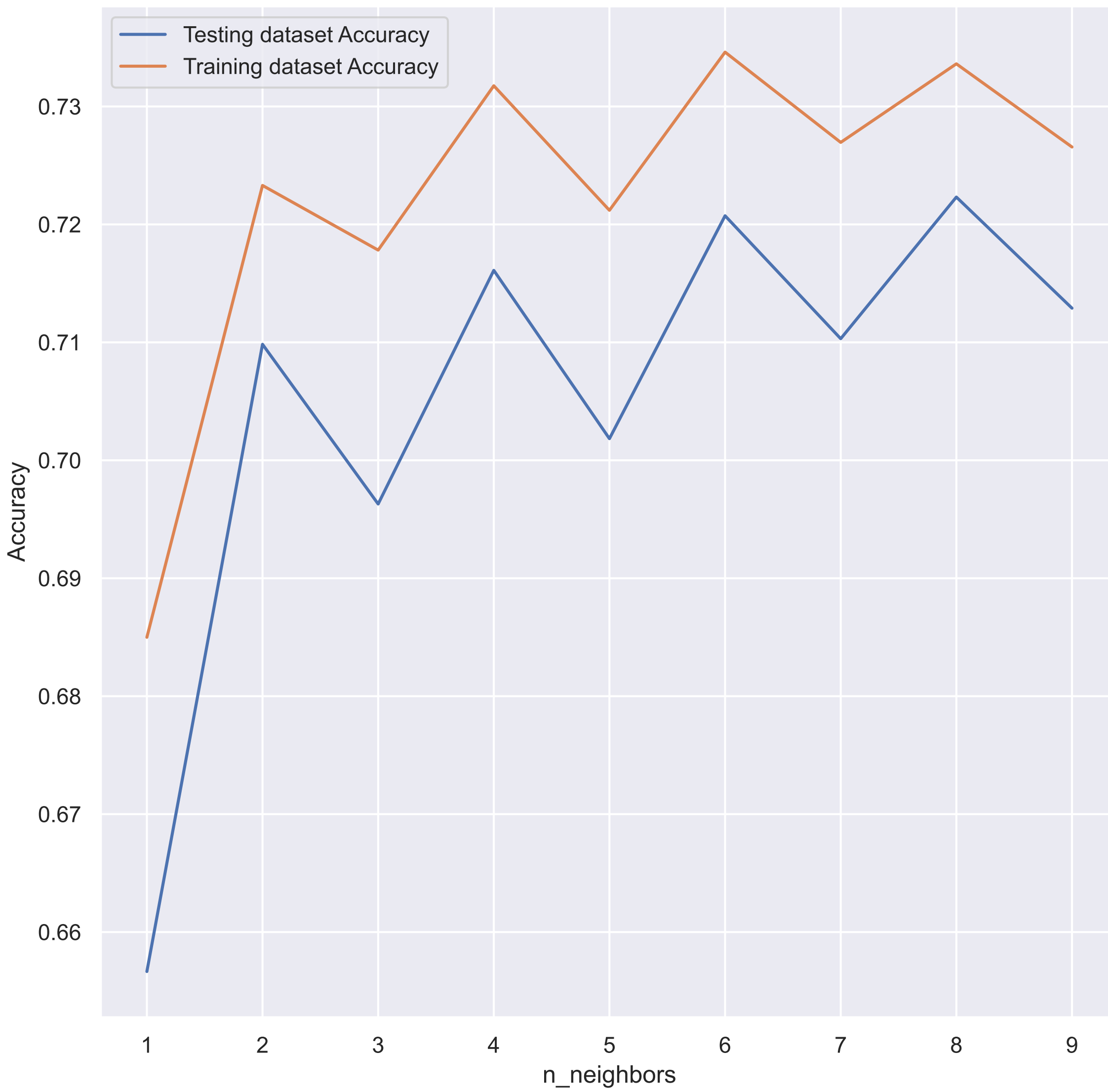
```
##               precision    recall  f1-score   support
##
##            0       0.74      0.92      0.82     37859
##            1       0.61      0.27      0.38     17010
##
##     accuracy                           0.72     54869
##    macro avg       0.67      0.60      0.60     54869
## weighted avg       0.70      0.72      0.68     54869
```

6.4 Finding the optimal value of K in KNN

```
#Finding Optimal Value of K in KNN
neighbors = np.arange(1, 10)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop over K values
for i, k in enumerate(neighbors):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

    # Compute training and test data accuracy
    train_accuracy[i] = knn.score(X_train, y_train)
    test_accuracy[i] = knn.score(X_test, y_test)


# Generate plot
```

```
## KNeighborsClassifier(n_neighbors=1)
## KNeighborsClassifier(n_neighbors=2)
## KNeighborsClassifier(n_neighbors=3)
## KNeighborsClassifier(n_neighbors=4)
## KNeighborsClassifier()
## KNeighborsClassifier(n_neighbors=6)
## KNeighborsClassifier(n_neighbors=7)
## KNeighborsClassifier(n_neighbors=8)
## KNeighborsClassifier(n_neighbors=9)
```

```
plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')

plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()
```



6.5 Final Model with K=8

```
k4 = 8
n4 = KNeighborsClassifier(n_neighbors=k4).fit(X_train,y_train)
n4
```

```
## KNeighborsClassifier(n_neighbors=8)
```

```
yhat4 = n4.predict(X_test)
yhat4[0:5]
```

```
## array([0, 1, 0, 0, 0], dtype=int64)
```

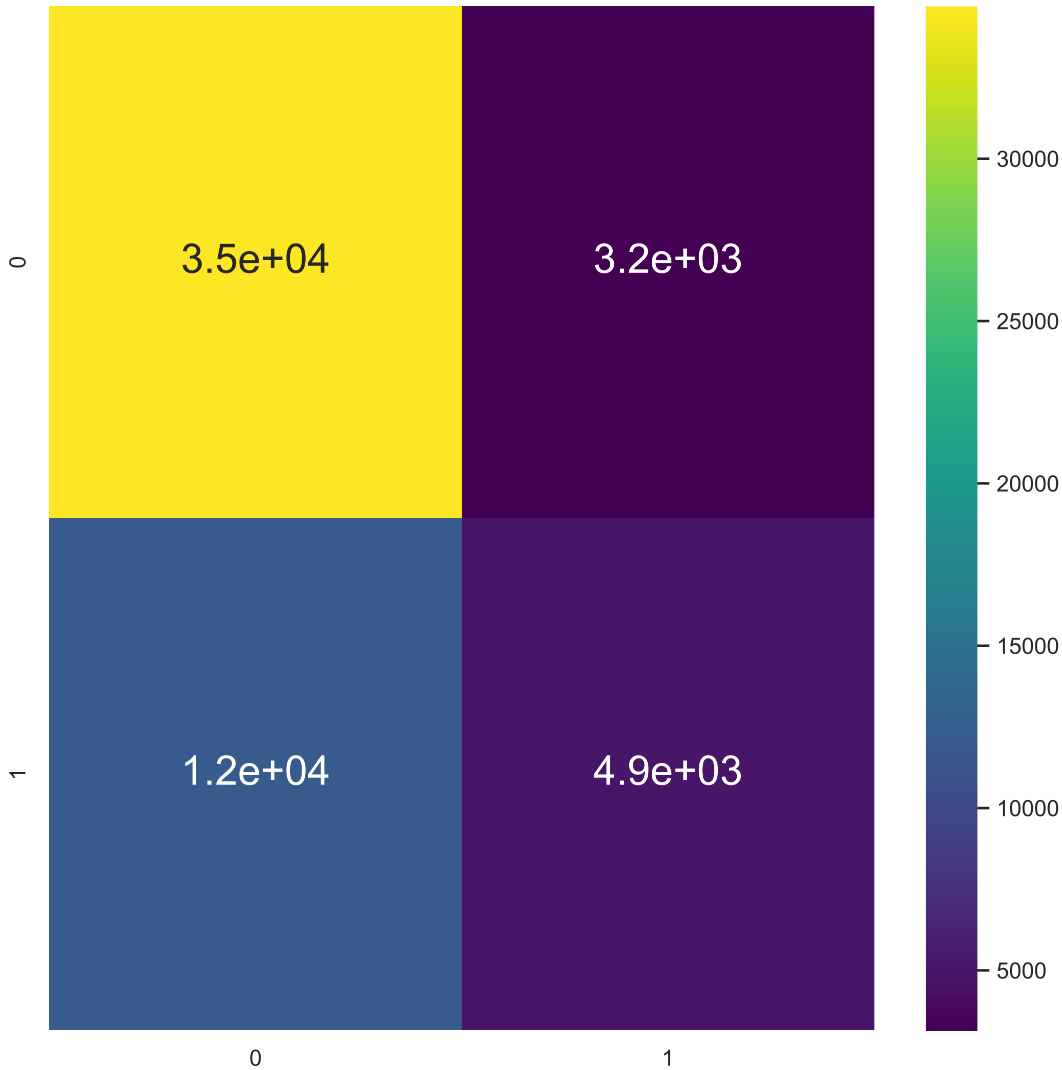
```
print("Train Accuracy:",metrics.accuracy_score(y_train,n4.predict(X_train)))
```

```
## Train Accuracy: 0.7336166091262712
```

```
print("Test Accuracy:",metrics.accuracy_score(y_test,yhat4))
```

```
## Test Accuracy: 0.7223204359474384
```

```
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
con_mat=confusion_matrix(y_test,yhat4)
sns.heatmap(con_mat,annot=True,annot_kws=
                           {"size":20},cmap="viridis")
plt.show()
```




```
print(classification_report(y_test, yhat4))
```

```
##               precision    recall  f1-score   support
##
##            0       0.74      0.92      0.82     37859
##            1       0.61      0.29      0.39     17010
##
##     accuracy                           0.72     54869
##    macro avg       0.68      0.60      0.61     54869
## weighted avg       0.70      0.72      0.69     54869
```

KNN model with K value of 8 is the best KNN model with accuracy of .7223 on the test dataset.