1
```
%tensorflow_version 2.x
import tensorflow as tf
print("Tensorflow version " + tf.__version__)

try:
  tpu = tf.distribute.cluster_resolver.TPUClusterResolver()  # TPU detection
  print('Running on TPU ', tpu.cluster_spec().as_dict()['worker'])
except ValueError:
  raise BaseException('ERROR: Not connected to a TPU runtime; please see the previous cell in this no

tf.config.experimental_connect_to_cluster(tpu)
tf.tpu.experimental.initialize_tpu_system(tpu)
tpu_strategy = tf.distribute.experimental.TPUStrategy(tpu)
```

```
Tensorflow version 2.3.0
Running on TPU  ['10.94.13.50:8470']
INFO:tensorflow:Initializing the TPU system: grpc://10.94.13.50:8470
INFO:tensorflow:Initializing the TPU system: grpc://10.94.13.50:8470
INFO:tensorflow:Clearing out eager caches
INFO:tensorflow:Clearing out eager caches
INFO:tensorflow:Finished initializing TPU system.
INFO:tensorflow:Finished initializing TPU system.
WARNING:absl:`tf.distribute.experimental.TPUStrategy` is deprecated, please use  the non experimental

INFO:tensorflow:Found TPU system:
INFO:tensorflow:Found TPU system:
INFO:tensorflow:*** Num TPU Cores: 8
INFO:tensorflow:*** Num TPU Cores: 8
INFO:tensorflow:*** Num TPU Workers: 1
INFO:tensorflow:*** Num TPU Workers: 1
INFO:tensorflow:*** Num TPU Cores Per Worker: 8
INFO:tensorflow:*** Num TPU Cores Per Worker: 8
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:localhost/replica:0/task:0/device:CPU:0,

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:localhost/replica:0/task:0/device:CPU:0,

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:localhost/replica:0/task:0/device:XLA_CPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:localhost/replica:0/task:0/device:XLA_CPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:CPU:0, CPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:CPU:0, CPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:0, TPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:0, TPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:1, TPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:1, TPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:2, TPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:2, TPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:3, TPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:3, TPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:4, TPU

INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:4, TPU
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:5, TPU
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:5, TPU
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:6, TPU
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:6, TPU
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:7, TPU
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU:7, TPU
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU_SYSTEM
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:TPU_SYSTEM
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:XLA_CPU:0,
INFO:tensorflow:*** Available Device: _DeviceAttributes(/job:worker/replica:0/task:0/device:XLA_CPU:0,
```

Data Mining Project : Analysis Of Road Accidents In US

Description

This is a countrywide car accident dataset, which covers 49 states of the USA. The accident data are collected from February 2016 to June 2020, using two APIs that provide streaming traffic incident (or event) data. These APIs broadcast traffic data captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. Currently, there are about 3.5 million accident records in this dataset. Check here to learn more about this dataset.

Content

This dataset has been collected in real-time, using multiple Traffic APIs. Currently, it contains accident data that are collected from February 2016 to June 2020 for the Contiguous United States.

Inspiration

US-Accidents can be used for numerous applications such as real-time car accident prediction, studying car accidents hotspot locations, casualty analysis and extracting cause and effect rules to predict car accidents, and studying the impact of precipitation or other environmental stimuli on accident occurrence. The most recent release of the dataset can also be useful to study the impact of COVID-19 on traffic behavior and accidents.

Applications of Dataset

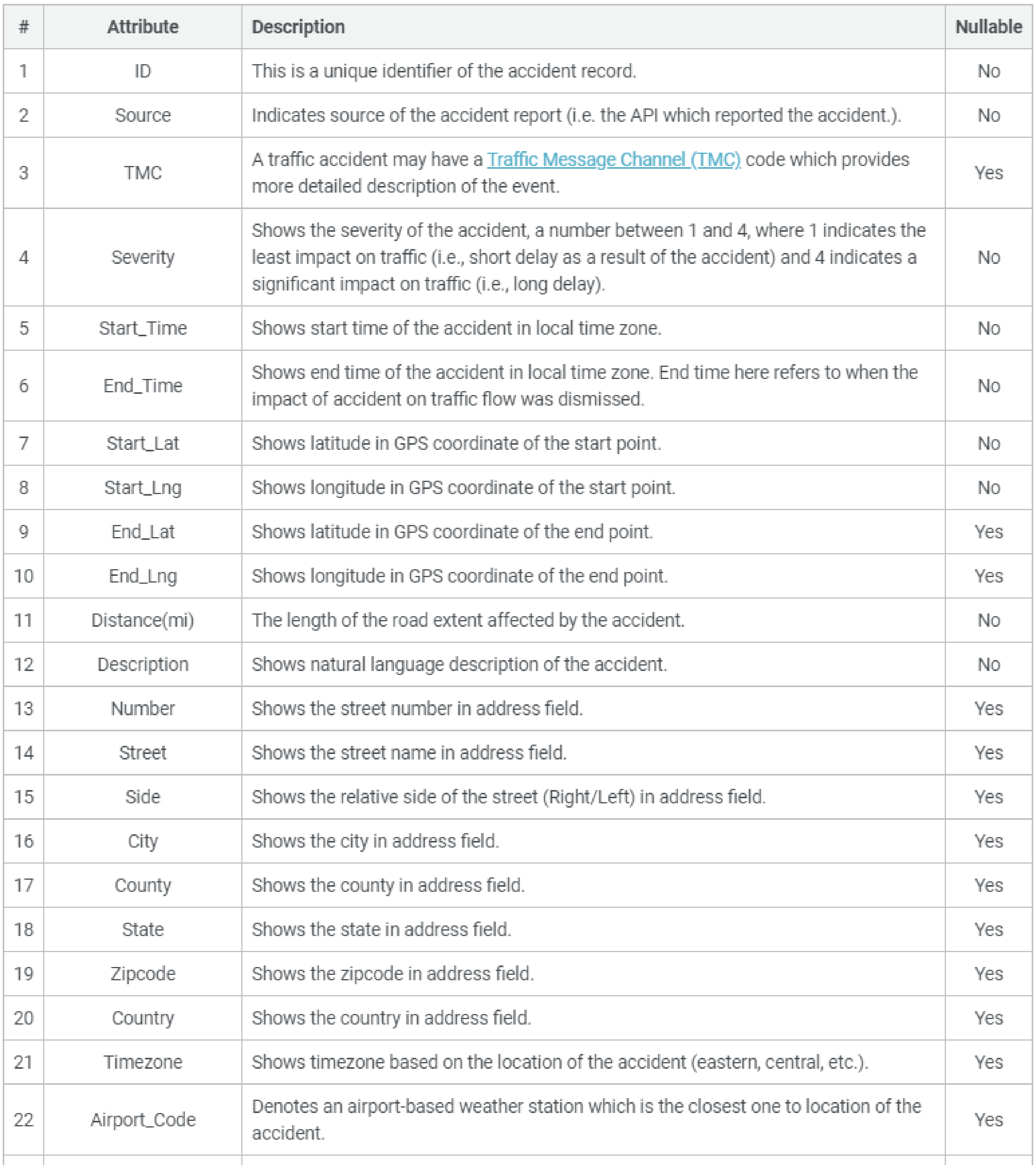
US-Accidents can be used for numerous applications such as real-time accident prediction, studying accident hotspot locations, casualty analysis and extracting cause and effect rules to predict accidents, or studying the impact of precipitation or other environmental stimuli on accident occurrence.

Setting Up Google Drive

2
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

Attributes Information



#	Attribute	Description	Nullable
1	ID	This is a unique identifier of the accident record.	No
2	Source	Indicates source of the accident report (i.e. the API which reported the accident.).	No
3	TMC	A traffic accident may have a Traffic Message Channel (TMC) code which provides more detailed description of the event.	Yes
4	Severity	Shows the severity of the accident, a number between 1 and 4, where 1 indicates the least impact on traffic (i.e., short delay as a result of the accident) and 4 indicates a significant impact on traffic (i.e., long delay).	No
5	Start_Time	Shows start time of the accident in local time zone.	No
6	End_Time	Shows end time of the accident in local time zone. End time here refers to when the impact of accident on traffic flow was dismissed.	No
7	Start_Lat	Shows latitude in GPS coordinate of the start point.	No
8	Start_Lng	Shows longitude in GPS coordinate of the start point.	No
9	End_Lat	Shows latitude in GPS coordinate of the end point.	Yes
10	End_Lng	Shows longitude in GPS coordinate of the end point.	Yes
11	Distance(mi)	The length of the road extent affected by the accident.	No
12	Description	Shows natural language description of the accident.	No
13	Number	Shows the street number in address field.	Yes
14	Street	Shows the street name in address field.	Yes
15	Side	Shows the relative side of the street (Right/Left) in address field.	Yes
16	City	Shows the city in address field.	Yes
17	County	Shows the county in address field.	Yes
18	State	Shows the state in address field.	Yes
19	Zipcode	Shows the zipcode in address field.	Yes
20	Country	Shows the country in address field.	Yes
21	Timezone	Shows timezone based on the location of the accident (eastern, central, etc.).	Yes
22	Airport_Code	Denotes an airport-based weather station which is the closest one to location of the accident.	Yes

23	Weather_Timestamp	Shows the time-stamp of weather observation record (in local time).	Yes
24	Temperature(F)	Shows the temperature (in Fahrenheit).	Yes
25	Wind_Chill(F)	Shows the wind chill (in Fahrenheit).	Yes
26	Humidity(%)	Shows the humidity (in percentage).	Yes
27	Pressure(in)	Shows the air pressure (in inches).	Yes
28	Visibility(mi)	Shows visibility (in miles).	Yes
29	Wind_Direction	Shows wind direction.	Yes
30	Wind_Speed(mph)	Shows wind speed (in miles per hour).	Yes
31	Precipitation(in)	Shows precipitation amount in inches, if there is any.	Yes
32	Weather_Condition	Shows the weather condition (rain, snow, thunderstorm, fog, etc.)	Yes
33	Amenity	A POI annotation which indicates presence of amenity in a nearby location.	No
34	Bump	A POI annotation which indicates presence of speed bump or hump in a nearby location.	No
35	Crossing	A POI annotation which indicates presence of crossing in a nearby location.	No
36	Give_Way	A POI annotation which indicates presence of give_way in a nearby location.	No
37	Junction	A POI annotation which indicates presence of junction in a nearby location.	No
38	No_Exit	A POI annotation which indicates presence of no_exit in a nearby location.	No
39	Railway	A POI annotation which indicates presence of railway in a nearby location.	No
40	Roundabout	A POI annotation which indicates presence of roundabout in a nearby location.	No
41	Station	A POI annotation which indicates presence of station in a nearby location.	No
42	Stop	A POI annotation which indicates presence of stop in a nearby location.	No
43	Traffic_Calming	A POI annotation which indicates presence of traffic_calming in a nearby location.	No
44	Traffic_Signal	A POI annotation which indicates presence of traffic_signal in a nearby location.	No
45	Turning_Loop	A POI annotation which indicates presence of turning_loop in a nearby location.	No
46	Sunrise_Sunset	Shows the period of day (i.e. day or night) based on sunrise/sunset.	Yes
47	Civil_Twilight	Shows the period of day (i.e. day or night) based on civil twilight.	Yes
48	Nautical_Twilight	Shows the period of day (i.e. day or night) based on nautical twilight.	Yes
49	Astronomical_Twilight	Shows the period of day (i.e. day or night) based on astronomical twilight.	Yes

Importing required packages

3

```
# Miscellaneous
import inspect
from sklearn.preprocessing import MinMaxScaler
import os

# To handle and analyze data
import pandas as pd

# To perform numeriacal operations
import numpy as np
```

```
# For missing values
import missingno as msno
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

# For outlier analysis
from sklearn.covariance import EllipticEnvelope

# For encoding
from sklearn.preprocessing import LabelEncoder

# For co-relation
!pip install mlens
from mlens.visualization import corrmat

# For visualization
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go

import graphviz
from sklearn import tree

from sklearn.metrics import plot_confusion_matrix

# For splitting data
from sklearn.model_selection import train_test_split

# For metrics
from sklearn import preprocessing
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import confusion_matrix, roc_auc_score

!pip install pyclustertend
from pyclustertend import hopkins
!pip install kneed
from kneed import KneeLocator
from sklearn import metrics

# For classification
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
import xgboost as xgb

from sklearn.linear_model import LogisticRegression
from mlxtend.classifier import StackingClassifier

# For Clustering
from sklearn.cluster import KMeans
```

```
Collecting mlens
[?25l  Downloading https://files.pythonhosted.org/packages/0b/f7/c04bda423ac93ddb54bc4c3a21c79c9a24b
[K     |████████████████████████████████| 235kB 7.0MB/s
[?25hRequirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.6/dist-packages (from mlen
Requirement already satisfied: scipy>=0.17 in /usr/local/lib/python3.6/dist-packages (from mlens) (1.
Installing collected packages: mlens
Successfully installed mlens-0.2.3

[MLENS] backend: threading
Collecting pyclustertend
  Downloading https://files.pythonhosted.org/packages/a3/67/5dd390479122860d3f0ea947e45561c6d4469edf9
```

```
Installing collected packages: pyclustertend
Successfully installed pyclustertend-1.4.9
Collecting kneed
  Downloading https://files.pythonhosted.org/packages/c3/6b/e130913aaaad1373060e259ab222ca2330672db696
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from kneed) (1.4.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from kneed) (3.2
Requirement already satisfied: numpy>=1.14.2 in /usr/local/lib/python3.6/dist-packages (from kneed) (1
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matpl
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from ma
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/di
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateuti
Installing collected packages: kneed
Successfully installed kneed-0.7.0

/usr/local/lib/python3.6/dist-packages/sklearn/externals/six.py:31: FutureWarning:

The module is deprecated in version 0.21 and will be removed in version 0.23 since we've dropped suppo
```

4
```
!unrar x "/content/gdrive/MyDrive/Data Mining Project Files/US_Accidents_June20.rar" "/content"

data = pd.read_csv("/content/US_Accidents_June20.csv")

data.head()
```

```
UNRAR 5.50 freeware      Copyright (c) 1993-2017 Alexander Roshal


Extracting from /content/gdrive/MyDrive/Data Mining Project Files/US_Accidents_June20.rar

Extracting  /content/US_Accidents_June20.csv                              □□□□  0%□□□□  1%□□□□  2%
All OK
```

4

	ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_
0	A-1	MapQuest	201.0	3	2016-02-08 05:46:00	2016-02-08 11:00:00	39.865147	-84.058723	NaN
1	A-2	MapQuest	201.0	2	2016-02-08 06:07:59	2016-02-08 06:37:59	39.928059	-82.831184	NaN
2	A-3	MapQuest	201.0	2	2016-02-08 06:49:27	2016-02-08 07:19:27	39.063148	-84.032608	NaN
3	A-4	MapQuest	201.0	3	2016-02-08 07:23:34	2016-02-08 07:53:34	39.747753	-84.205582	NaN

	ID	Source	TMC	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_
4	A-5	MapQuest	201.0	2	2016-02-08 07:39:07	2016-02-08 08:09:07	39.627781	-84.188354	NaN

5
```
size = os.path.getsize('/content/US_Accidents_June20.csv')
print('File size : {0:.2f} GB'.format(size/1e+9))
```

```
File size : 1.33 GB
```

6
```
# Removing unwanted columns
data.drop(['ID', 'Source', 'TMC', 'Airport_Code', 'Description', 'Country', 'Timezone', 'Zipcode', 'T
```

Getting Data Information

7
```
data.shape
```

7
```
(3513617, 40)
```

8
```
data.describe()
```

8

	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(m
count	3.513617e+06	3.513617e+06	3.513617e+06	1.034799e+06	1.034799e+06	3.513617e+(
mean	2.339929e+00	3.654195e+01	-9.579151e+01	3.755758e+01	-1.004560e+02	2.816167e-0
std	5.521935e-01	4.883520e+00	1.736877e+01	4.861215e+00	1.852879e+01	1.550134e+(
min	1.000000e+00	2.455527e+01	-1.246238e+02	2.457011e+01	-1.244978e+02	0.000000e+(
25%	2.000000e+00	3.363784e+01	-1.174418e+02	3.399477e+01	-1.183440e+02	0.000000e+(
50%	2.000000e+00	3.591687e+01	-9.102601e+01	3.779736e+01	-9.703438e+01	0.000000e+(
75%	3.000000e+00	4.032217e+01	-8.093299e+01	4.105139e+01	-8.210168e+01	1.000000e-0
max	4.000000e+00	4.900220e+01	-6.711317e+01	4.907500e+01	-6.710924e+01	3.336300e+(

9
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3513617 entries, 0 to 3513616
Data columns (total 40 columns):
 #   Column                 Dtype
---  ------                 -----
 0   Severity               int64
 1   Start_Time             object
 2   End_Time               object
 3   Start_Lat              float64
 4   Start_Lng              float64
 5   End_Lat                float64
 6   End_Lng                float64
 7   Distance(mi)           float64
 8   Number                 float64
 9   Street                 object
 10  Side                   object
 11  City                   object
 12  County                 object
 13  State                  object
 14  Weather_Timestamp      object
 15  Temperature(F)         float64
 16  Wind_Chill(F)          float64
 17  Humidity(%)            float64
 18  Pressure(in)           float64
 19  Visibility(mi)         float64
 20  Wind_Direction         object
 21  Wind_Speed(mph)        float64
 22  Precipitation(in)      float64
 23  Weather_Condition      object
 24  Amenity                bool
 25  Bump                   bool
 26  Crossing               bool
 27  Give_Way               bool
 28  Junction               bool
 29  No_Exit                bool
 30  Railway                bool
 31  Roundabout             bool
 32  Station                bool
 33  Stop                   bool
 34  Traffic_Calming        bool
 35  Traffic_Signal         bool
 36  Sunrise_Sunset         object
 37  Civil_Twilight         object
 38  Nautical_Twilight      object
 39  Astronomical_Twilight  object
dtypes: bool(12), float64(13), int64(1), object(14)
memory usage: 790.8+ MB
```

10
```
# Getting column names
data.columns
```

10
```
Index(['Severity', 'Start_Time', 'End_Time', 'Start_Lat', 'Start_Lng',
       'End_Lat', 'End_Lng', 'Distance(mi)', 'Number', 'Street', 'Side',
       'City', 'County', 'State', 'Weather_Timestamp', 'Temperature(F)',
       'Wind_Chill(F)', 'Humidity(%)', 'Pressure(in)', 'Visibility(mi)',
       'Wind_Direction', 'Wind_Speed(mph)', 'Precipitation(in)',
       'Weather_Condition', 'Amenity', 'Bump', 'Crossing', 'Give_Way',
       'Junction', 'No_Exit', 'Railway', 'Roundabout', 'Station', 'Stop',
       'Traffic_Calming', 'Traffic_Signal', 'Sunrise_Sunset', 'Civil_Twilight',
       'Nautical_Twilight', 'Astronomical_Twilight'],
      dtype='object')
```

Visualization

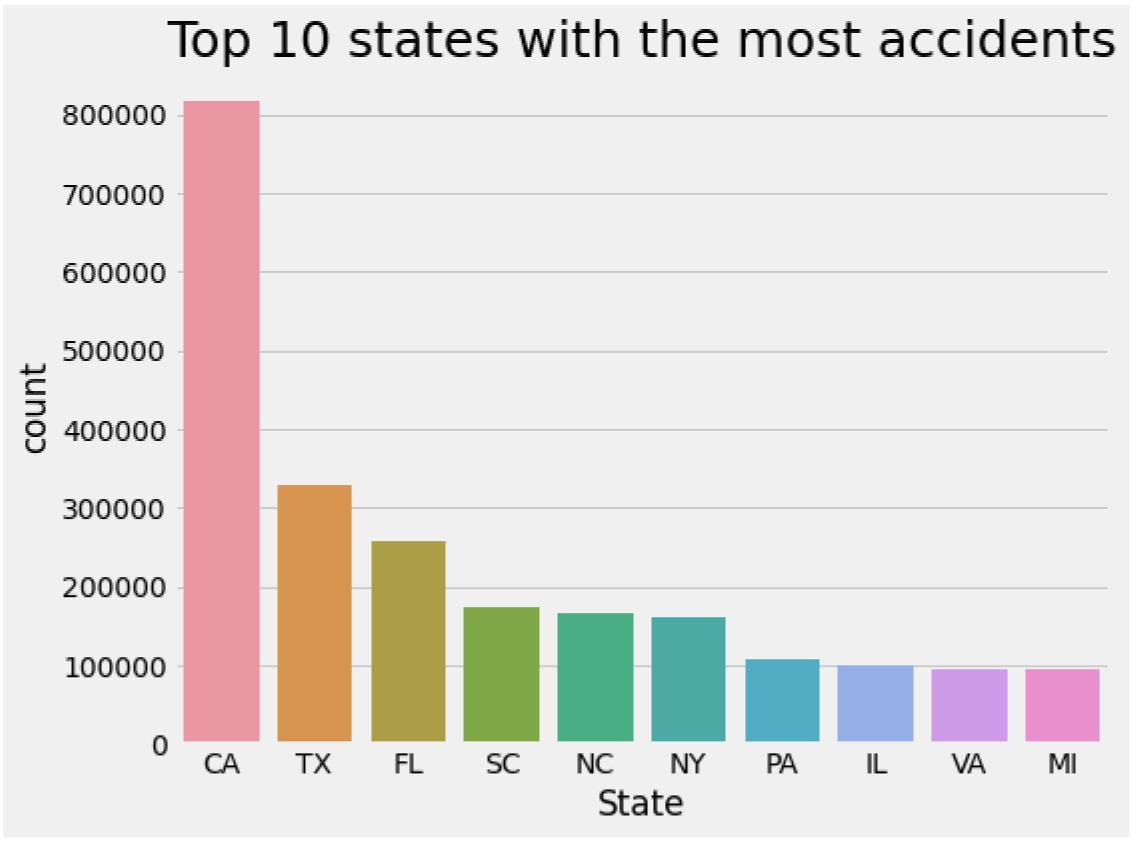
11
```
plt.style.use("fivethirtyeight")
plt.rcParams['figure.figsize'] = (8, 6)
```

Top 10 states with the most accidents

12
```
sns.countplot(data['State'], order=data['State'].value_counts().iloc[:10].index)
plt.xticks(rotation=0)
plt.title("Top 10 states with the most accidents", fontsize=25)
plt.tight_layout()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:
```

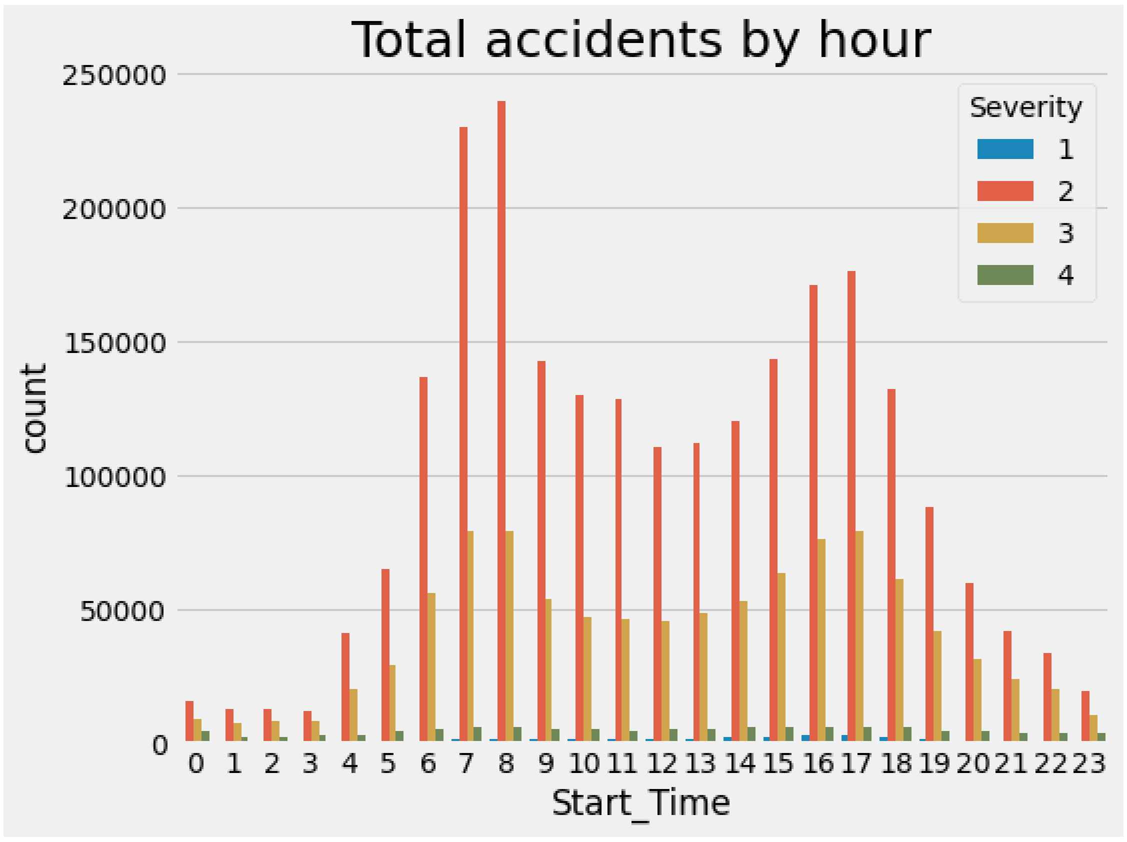
```
Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument
```



Total accidents by hour

13
```
data.Start_Time=pd.to_datetime(data.Start_Time)
data.End_Time=pd.to_datetime(data.End_Time)

sns.countplot(data['Start_Time'].dt.hour, hue=data['Severity'])
plt.xticks(rotation=0)
plt.title("Total accidents by hour", fontsize=25)
plt.tight_layout()
```

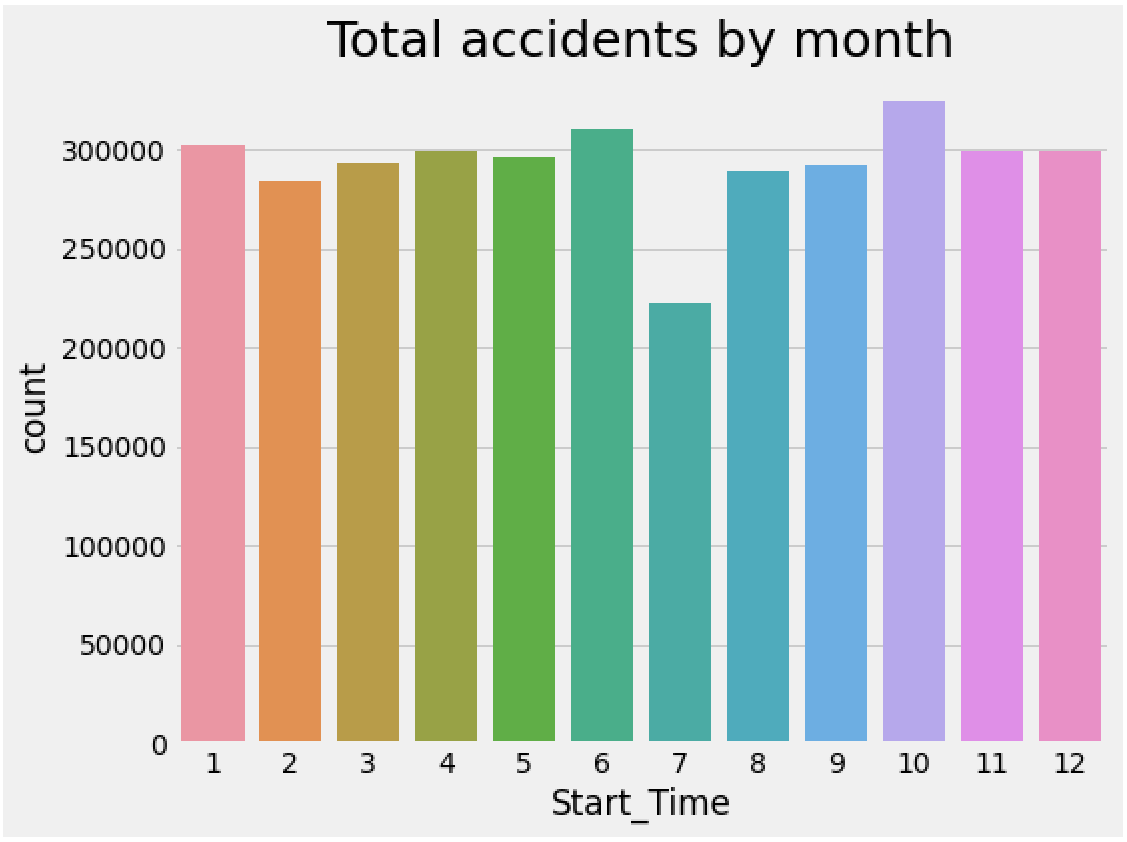
```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument
```



Total accidents by different month

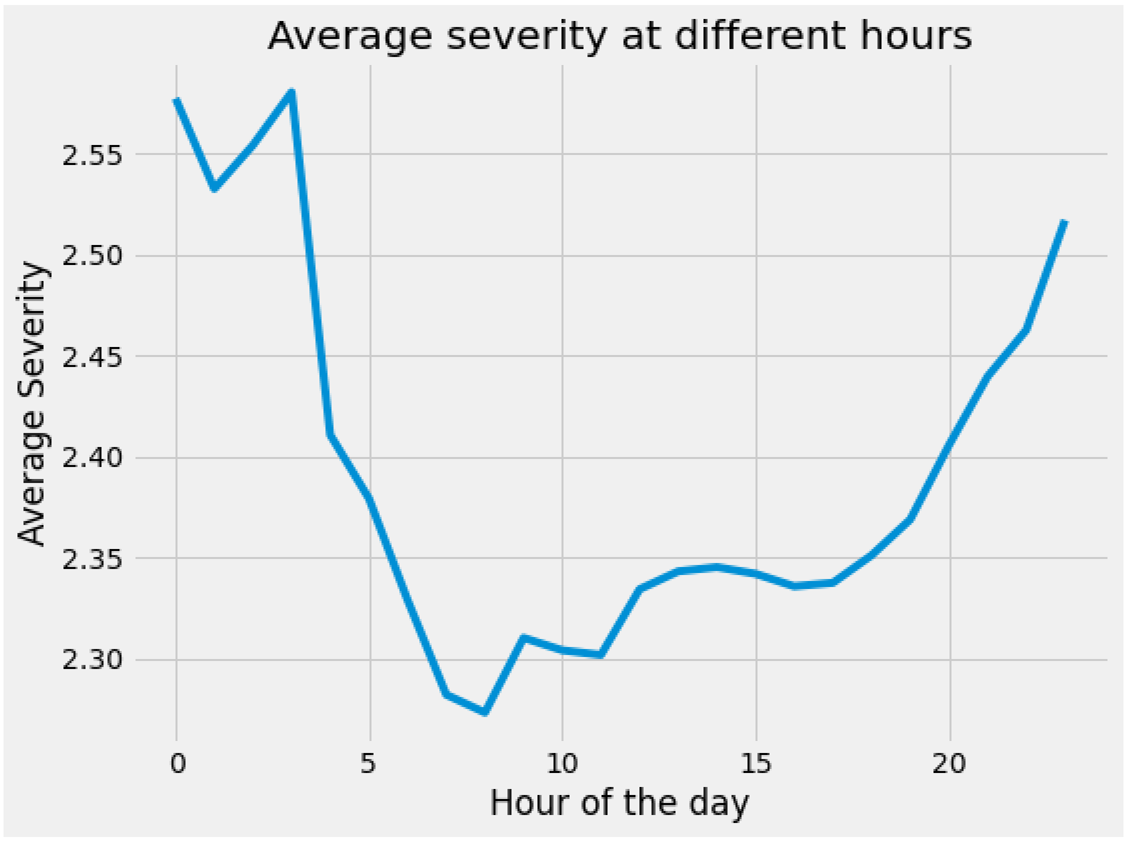
14
```
sns.countplot(data['Start_Time'].dt.month)
plt.xticks(rotation=0)
plt.title("Total accidents by month", fontsize=25)
plt.tight_layout()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argumen
```



Average severity at different hours

15
```
df = data.copy()

df['Accident_hour']=df['Start_Time'].dt.hour

df.groupby('Accident_hour')['Severity'].mean().plot(kind='line')
plt.xlabel('Hour of the day')
plt.ylabel('Average Severity')
plt.title('Average severity at different hours')
plt.tight_layout()
```



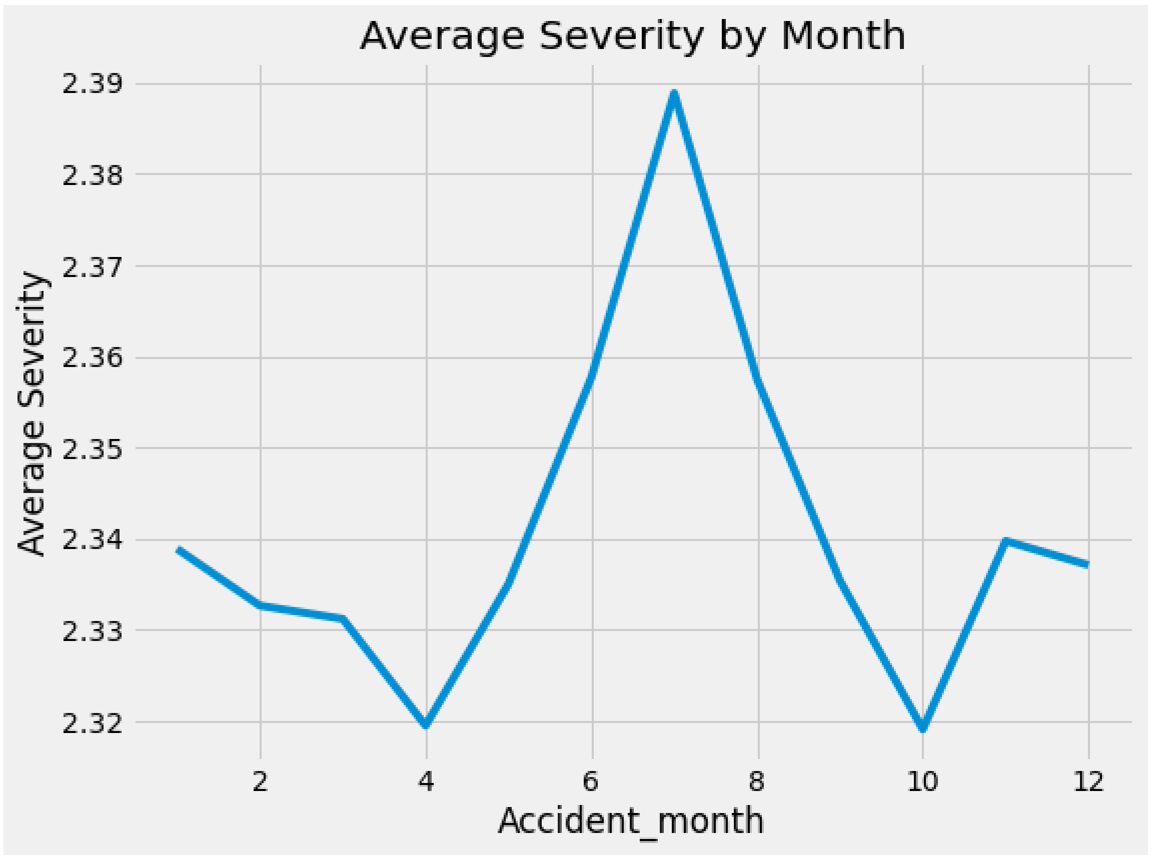
Average Severity by Month

16

```
df['Accident_month']=df['Start_Time'].dt.month
df.groupby('Accident_month')['Severity'].mean().plot(kind='line')
plt.ylabel('Average Severity')
plt.title('Average Severity by Month')
```

16

```
Text(0.5, 1.0, 'Average Severity by Month')
```



Plotting the Maps

Accidents count (group by States)

17

```
state_count_acc = pd.value_counts(data['State'])

fig = go.Figure(data=go.Choropleth(locations=state_count_acc.index, z = state_count_acc.values.astype

fig.update_layout(title_text = '2016 - 2019 US Traffic Accident Dataset by State',geo_scope='usa',)

fig.show()
```

Severity of accidents

18

```
data_sever = data.sample(n=10000)

fig = go.Figure(data=go.Scattergeo(locationmode = 'USA-states',lon = data_sever['Start_Lng'],lat = da
                                   marker = dict(size = 8,opacity = 0.8,reversescale = True,autocolor
            colorbar_title="Severity"
        )))

fig.update_layout(
        title = 'Severity of accidents',
        geo = dict(
            scope='usa',
            projection_type='albers usa',
            showland = True,
            landcolor = "rgb(250, 250, 250)",
            subunitcolor = "rgb(217, 217, 217)",
            countrycolor = "rgb(217, 217, 217)",
            countrywidth = 0.7,
            subunitwidth = 0.7
        ),
    )
fig.show()
```

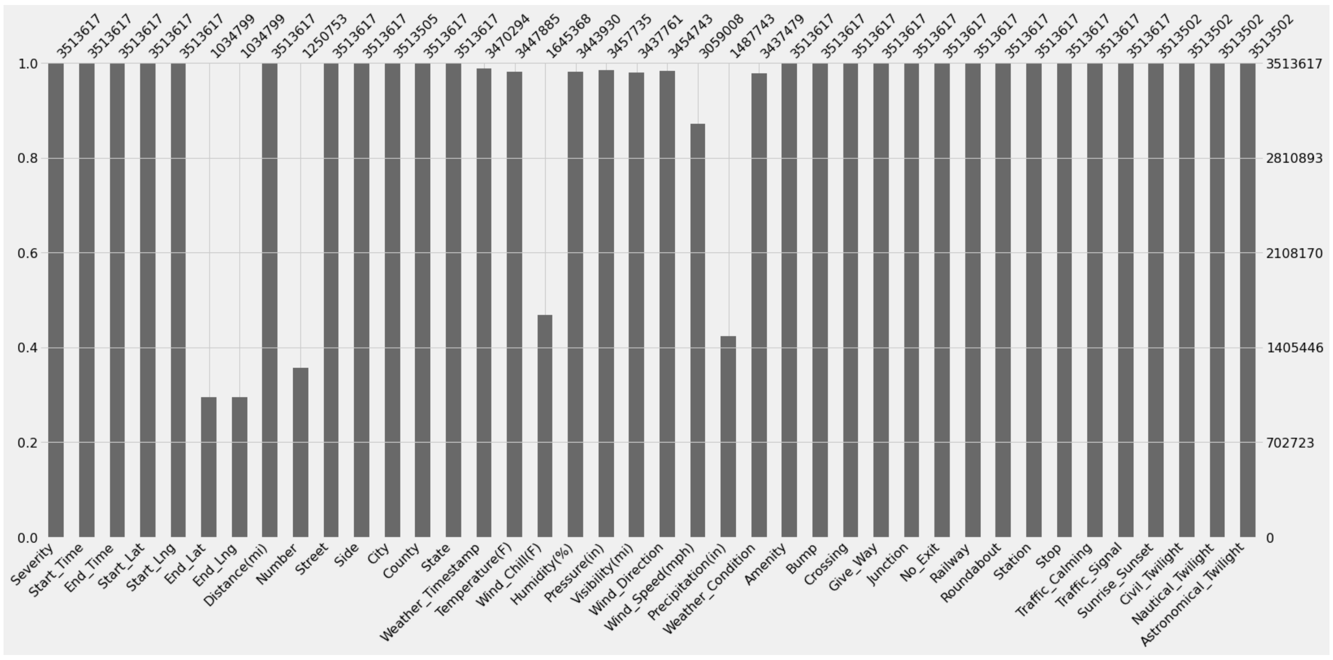
Data Preprocessing

Data Cleaning

Handling Missing Values

19
```
# Visualizing missing values
msno.bar(data)
```

19
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc24e5e9e10>
```



20
```
# Getting number of null values in each column
data.isna().sum()
```

20
```
Severity                    0
Start_Time                  0
End_Time                    0
Start_Lat                   0
Start_Lng                   0
End_Lat               2478818
End_Lng               2478818
Distance(mi)                0
Number                2262864
Street                      0
Side                        0
City                      112
County                      0
State                       0
Weather_Timestamp       43323
Temperature(F)          65732
Wind_Chill(F)         1868249
Humidity(%)             69687
Pressure(in)            55882
Visibility(mi)          75856
Wind_Direction          58874
Wind_Speed(mph)        454609
Precipitation(in)     2025874
Weather_Condition       76138
Amenity                     0
Bump                        0
Crossing                    0
Give_Way                    0
Junction                    0
No_Exit                     0
Railway                     0
Roundabout                  0
Station                     0
Stop                        0
Traffic_Calming             0
Traffic_Signal              0
Sunrise_Sunset            115
Civil_Twilight            115
Nautical_Twilight         115
Astronomical_Twilight     115
dtype: int64
```

21
```
# Getting value counts

for column in data.columns:
  print(data[column].value_counts())
  print("*" * 40)
```

```
2    2373210
3     998913
4     112320
1      29174
Name: Severity, dtype: int64
****************************************
2017-05-15 09:22:55    74
2018-11-25 01:22:49    53
2019-10-26 08:45:17    49
2018-11-12 00:37:27    40
2018-12-18 07:11:45    37
                       ..
2019-05-20 13:19:16     1
2019-08-08 17:19:04     1
2017-08-22 11:03:41     1
2018-09-26 17:07:20     1
2018-03-19 11:36:23     1
Name: Start_Time, Length: 3200042, dtype: int64
****************************************
2017-05-15 15:22:55    73
2019-10-26 09:14:51    49
2020-02-14 00:00:00    46
2018-11-25 02:51:02    46
2020-02-12 00:00:00    42
                       ..
2016-07-08 11:24:44     1
2018-10-14 11:05:36     1
2019-12-09 11:44:23     1
2018-12-17 12:05:44     1
2019-11-07 18:14:17     1
Name: End_Time, Length: 3246120, dtype: int64
****************************************
37.808498    570
33.941364    566
42.476501    534
33.744976    533
34.858925    494
            ...
39.703094      1
44.294941      1
42.962944      1
41.071260      1
35.139534      1
Name: Start_Lat, Length: 1124695, dtype: int64
****************************************
-122.366852    577
-118.096634    562
-83.111794     534
-84.390343     532
-82.259857     495
              ...
-71.064529       1
-80.841322       1
-77.466064       1
-84.460350       1
-86.214943       1
Name: Start_Lng, Length: 1113407, dtype: int64
****************************************
40.849920    257
40.850020    155
33.876280    150
45.598460    148
41.803290    146
            ...
```

```
30.444399      1
40.738482      1
46.902070      1
40.365760      1
44.363369      1
Name: End_Lat, Length: 375074, dtype: int64
****************************************
-73.944080     262
-80.209950     195
-122.665330    158
-104.987680    155
-122.550430    148
              ...
-94.637950       1
-121.591231      1
-90.154890       1
-77.042650       1
-106.414129      1
Name: End_Lng, Length: 383569, dtype: int64
****************************************
0.000     2457174
0.010      250988
0.010       13359
0.020        5968
0.001        5528
           ...
9.356           1
7.967           1
9.269           1
6.688           1
16.911          1
Name: Distance(mi), Length: 13476, dtype: int64
****************************************
1.0        15347
2.0        15304
101.0      11692
100.0      11461
199.0       3809
           ...
30873.0        1
30878.0        1
30881.0        1
30883.0        1
20286.0        1
Name: Number, Length: 40365, dtype: int64
****************************************
I-5 N                 41264
I-95 N                35593
I-95 S                29517
I-10 E                26107
I-10 W                25045
                        ...
Jacobs Ford Rd            1
W Peebles Rd              1
Old Bradley Rd            1
S9248 State Road 78       1
Settlement Dr             1
Name: Street, Length: 176262, dtype: int64
****************************************
R    2879797
L     633819
           1
Name: Side, dtype: int64
****************************************
Houston        101240
Los Angeles     79169
Charlotte       78952
Dallas          64790
Austin          63889
```

```
                 ...
Green Spring         1
Lisbon Falls         1
Licking              1
Rivesville           1
Houstonia            1
Name: City, Length: 11895, dtype: int64
****************************************
Los Angeles    271627
Harris         107761
Orange          89349
Mecklenburg     84258
Dallas          76668
                ...
Graves              1
Dundy               1
Ness                1
Frontier            1
Sharp               1
Name: County, Length: 1724, dtype: int64
****************************************
CA    816825
TX    329284
FL    258002
SC    173277
NC    165958
NY    160817
PA    106787
IL     99692
VA     96075
MI     95983
GA     93614
OR     90122
MN     81863
AZ     78584
TN     69895
WA     68544
OH     66139
LA     61515
OK     60003
NJ     59059
MD     53593
UT     51685
CO     49731
AL     44625
MA     39044
IN     33746
MO     33643
CT     25901
NE     23970
KY     22553
WI     20120
RI     11753
IA     11475
NV     10643
NH      7984
KS      7939
MS      6585
DE      5739
NM      5523
DC      4820
WV      2381
ME      2243
ID      2044
AR      2012
VT       702
MT       512
WY       508
SD        61
```

```
ND        44
Name: State, dtype: int64
****************************************
2019-09-17 07:53:00    267
2019-09-24 07:53:00    267
2019-08-29 07:53:00    261
2019-10-02 07:53:00    257
2019-12-03 07:53:00    255
                      ...
2017-12-18 19:49:00      1
2018-03-06 10:39:00      1
2020-04-30 14:18:00      1
2016-11-07 08:10:00      1
2016-07-09 03:51:00      1
Name: Weather_Timestamp, Length: 546086, dtype: int64
****************************************
 68.0     77167
 77.0     75531
 59.0     72519
 73.0     67819
 63.0     64722
          ...
-9.8          1
 112.8        1
 167.0        1
 168.8        1
 161.6        1
Name: Temperature(F), Length: 831, dtype: int64
****************************************
 63.0    31512
 73.0    31474
 64.0    30579
 72.0    29726
 70.0    29647
         ...
-24.2        1
-45.3        1
-37.1        1
-43.4        1
-34.5        1
Name: Wind_Chill(F), Length: 974, dtype: int64
****************************************
100.0    139900
93.0     132465
90.0      77591
87.0      77572
89.0      64531
          ...
5.0        2139
4.0        1193
3.0         287
2.0          87
1.0          13
Name: Humidity(%), Length: 100, dtype: int64
****************************************
30.01    71148
29.99    69561
29.96    69547
30.04    68220
29.94    66479
         ...
22.40        1
20.85        1
0.39         1
20.03        1
20.15        1
Name: Pressure(in), Length: 1022, dtype: int64
****************************************
10.0    2736555
```

```
7.0      106068
9.0       91832
8.0       73128
5.0       69251
         ...
43.0          1
46.0          1
54.0          1
58.0          1
16.0          1
Name: Visibility(mi), Length: 85, dtype: int64
****************************************
Calm        368282
CALM        217424
SSW         181645
South       177225
WNW         174115
SW          172252
WSW         165738
NW          164928
West        164624
SSE         163649
North       153252
NNW         147047
SE          132051
NNE         117475
NE          115931
ESE         114855
Variable    113897
ENE         112626
S           103970
East        103462
W            95115
N            70516
VAR          64523
E            60141
Name: Wind_Direction, dtype: int64
****************************************
4.6      217448
0.0      217426
5.8      215965
3.5      203424
6.9      201257
          ...
127.0         1
129.0         1
471.8         1
141.5         1
77.1          1
Name: Wind_Speed(mph), Length: 160, dtype: int64
****************************************
0.00     1238383
0.01       69769
0.02       34993
0.03       23778
0.04       17591
          ...
1.69           1
2.20           1
2.28           1
24.00          1
2.02           1
Name: Precipitation(in), Length: 261, dtype: int64
****************************************
Clear                           808202
Fair                            547721
Mostly Cloudy                   488094
Overcast                        382485
Partly Cloudy                   344815
```

```
                                   ...
Thunder / Wintry Mix / Windy         1
Snow and Thunder                     1
Partial Fog / Windy                  1
Drifting Snow                        1
Blowing Sand                         1
Name: Weather_Condition, Length: 127, dtype: int64
****************************************
False    3471535
True       42082
Name: Amenity, dtype: int64
****************************************
False    3513011
True         606
Name: Bump, dtype: int64
****************************************
False    3239091
True      274526
Name: Crossing, dtype: int64
****************************************
False    3504053
True        9564
Name: Give_Way, dtype: int64
****************************************
False    3229168
True      284449
Name: Junction, dtype: int64
****************************************
False    3509233
True        4384
Name: No_Exit, dtype: int64
****************************************
False    3482442
True       31175
Name: Railway, dtype: int64
****************************************
False    3513433
True         184
Name: Roundabout, dtype: int64
****************************************
False    3443296
True       70321
Name: Station, dtype: int64
****************************************
False    3461641
True       51976
Name: Stop, dtype: int64
****************************************
False    3512216
True        1401
Name: Traffic_Calming, dtype: int64
****************************************
False    2889994
True      623623
Name: Traffic_Signal, dtype: int64
****************************************
Day      2593757
Night     919745
Name: Sunrise_Sunset, dtype: int64
****************************************
Day      2767921
Night     745581
Name: Civil_Twilight, dtype: int64
****************************************
Day      2943398
Night     570104
Name: Nautical_Twilight, dtype: int64
****************************************
Day      3075001
```

```
Night     438501
Name: Astronomical_Twilight, dtype: int64
****************************************
```

22
```
# Getting number of unique values

for column in data.columns:
  print(column, ":", len(data[column].unique()))
  print("Data : ", data[column].unique())
  print("*" * 40)
```

```
Severity : 4
Data :  [3 2 1 4]
****************************************
Start_Time : 3200042
Data :  ['2016-02-08T05:46:00.000000000' '2016-02-08T06:07:59.000000000'
 '2016-02-08T06:49:27.000000000' ... '2019-08-23T19:11:30.000000000'
 '2019-08-23T19:00:21.000000000' '2019-08-23T18:52:06.000000000']
****************************************
End_Time : 3246120
Data :  ['2016-02-08T11:00:00.000000000' '2016-02-08T06:37:59.000000000'
 '2016-02-08T07:19:27.000000000' ... '2019-08-23T19:28:49.000000000'
 '2019-08-23T19:29:42.000000000' '2019-08-23T19:21:31.000000000']
****************************************
Start_Lat : 1124695
Data :  [39.865147 39.928059 39.063148 ... 34.120911 33.943599 34.239104]
****************************************
Start_Lng : 1113407
Data :  [ -84.058723  -82.831184  -84.032608 ... -118.416176 -117.14806
 -117.84779 ]
****************************************
End_Lat : 375075
Data :  [      nan 40.11206  39.86501  ... 34.239104 33.98311  34.13736 ]
****************************************
End_Lng : 383570
Data :  [         nan  -83.03187   -84.04873  ... -118.416176 -118.39565
 -117.23934 ]
****************************************
Distance(mi) : 13476
Data :  [1.0000e-02 0.0000e+00 1.3200e+00 ... 7.4260e+00 3.6350e+01 1.6051e+01]
****************************************
Number : 40366
Data :  [   nan  2584.   376. ... 17742. 68630. 28237.]
****************************************
Street : 176262
Data :  ['I-70 E' 'Brice Rd' 'State Route 32' ... ' SE Dale Ln' ' San Marlo Way'
 '473-401 Cutoff Rd']
****************************************
Side : 3
Data :  ['R' 'L' ' ']
****************************************
City : 11896
Data :  ['Dayton' 'Reynoldsburg' 'Williamsburg' ... 'Paron' 'Clarksdale'
 'American Fork-Pleasant Grove']
****************************************
County : 1724
Data :  ['Montgomery' 'Franklin' 'Clermont' ... 'Mellette' 'Hodgeman' 'Ness']
****************************************
State : 49
Data :  ['OH' 'WV' 'CA' 'FL' 'GA' 'SC' 'NE' 'IA' 'IL' 'MO' 'WI' 'IN' 'MI' 'NJ'
 'NY' 'CT' 'MA' 'RI' 'NH' 'PA' 'KY' 'MD' 'VA' 'DC' 'DE' 'TX' 'WA' 'OR'
 'AL' 'TN' 'NC' 'KS' 'LA' 'OK' 'CO' 'UT' 'AZ' 'MN' 'MS' 'NV' 'ME' 'AR'
 'ID' 'VT' 'NM' 'ND' 'WY' 'SD' 'MT']
****************************************
Weather_Timestamp : 546087
Data :  ['2016-02-08 05:58:00' '2016-02-08 05:51:00' '2016-02-08 06:56:00' ...
 '2019-08-23 12:35:00' '2019-08-23 15:18:00' '2019-08-23 01:20:00']
****************************************
Temperature(F) : 832
```

```
Data :  [ 3.690e+01  3.790e+01  3.600e+01  3.510e+01  3.400e+01  3.330e+01
  3.740e+01  3.560e+01  3.380e+01  3.700e+01  3.990e+01  3.920e+01
  3.420e+01  3.310e+01  2.300e+01  2.280e+01  2.660e+01  2.100e+01
  1.990e+01  2.500e+01  2.610e+01  2.480e+01  2.190e+01  2.120e+01
  2.250e+01  3.200e+01  2.700e+01  1.620e+01  1.580e+01  1.540e+01
  1.400e+01  1.510e+01  1.600e+01  1.760e+01  1.710e+01  7.000e+00
  6.100e+00  5.000e+00  1.200e+01  9.000e+00  7.500e+00  1.800e+01
  1.090e+01  1.360e+01  1.040e+01  1.850e+01  2.160e+01  1.900e+01
  2.550e+01  1.000e+01  8.100e+00  1.290e+01  2.410e+01  3.090e+01
  3.360e+01  3.290e+01  3.220e+01  3.000e+01  3.110e+01  3.240e+01
  3.040e+01  2.840e+01  2.800e+01  3.020e+01  3.070e+01  3.900e+00
  2.590e+01  2.890e+01  3.900e+01  3.540e+01  3.490e+01  4.350e+01
  4.100e+01  4.280e+01  4.300e+01  4.770e+01  4.800e+01  4.960e+01
  5.380e+01  5.700e+01  5.940e+01  4.640e+01  6.510e+01  2.950e+01
  2.790e+01  3.250e+01  4.870e+01  4.690e+01  4.750e+01  3.180e+01
  3.830e+01  4.840e+01  5.400e+01  5.310e+01  5.000e+01  4.460e+01
  4.210e+01  4.060e+01  4.900e+01  4.410e+01  4.600e+01  4.500e+01
  5.200e+01  3.340e+01  3.430e+01  6.310e+01  6.280e+01  5.410e+01
  5.250e+01  4.890e+01  4.660e+01  5.430e+01  5.360e+01  5.590e+01
  4.820e+01  2.980e+01  2.520e+01  3.060e+01  3.160e+01  6.010e+01
  5.790e+01  6.210e+01  5.900e+01  6.240e+01  5.880e+01  5.110e+01
  5.050e+01  5.540e+01  5.520e+01  5.500e+01  5.670e+01  6.120e+01
  6.600e+01  7.020e+01  6.800e+01  7.160e+01  7.110e+01  6.910e+01
  6.690e+01  6.300e+01  6.100e+01  6.400e+01  7.000e+01  7.300e+01
  6.980e+01  5.720e+01  5.920e+01  6.080e+01  5.990e+01  5.850e+01
  5.760e+01  5.860e+01  5.650e+01        nan  4.550e+01  4.590e+01
  5.180e+01  5.740e+01  6.580e+01  6.490e+01  6.780e+01  6.440e+01
  6.040e+01  6.260e+01  3.780e+01  4.680e+01  3.970e+01  5.580e+01
  7.500e+01  8.290e+01  7.590e+01  7.520e+01  8.060e+01  8.600e+01
  7.390e+01  8.200e+01  8.400e+01  8.800e+01  8.240e+01  8.420e+01
  8.780e+01  7.700e+01  8.960e+01  9.000e+01  8.710e+01  9.140e+01
  9.190e+01  9.320e+01  9.390e+01  7.900e+01  9.810e+01  9.680e+01
  9.700e+01  7.880e+01  9.900e+01  9.500e+01  9.860e+01  8.010e+01
  8.100e+01  7.810e+01  7.200e+01  7.340e+01  6.620e+01  8.530e+01
  8.910e+01  9.100e+01  8.490e+01  7.140e+01  9.610e+01  1.004e+02
  9.300e+01  9.660e+01  8.920e+01  1.000e+02  1.040e+02  1.009e+02
  1.022e+02  1.020e+02  1.029e+02  7.360e+01  7.750e+01  6.170e+01
  7.680e+01  5.770e+01  6.820e+01  7.250e+01  8.670e+01  1.036e+02
  1.026e+02  7.740e+01  1.060e+02  8.190e+01  7.930e+01  9.570e+01
  6.030e+01  9.950e+01  8.110e+01  8.870e+01  8.760e+01  8.080e+01
  4.710e+01  4.140e+01  5.610e+01  5.320e+01  3.580e+01  4.370e+01
  3.610e+01  5.040e+01  3.940e+01  4.030e+01  5.470e+01  4.150e+01
  5.810e+01  3.630e+01  5.290e+01  4.190e+01  4.320e+01  7.830e+01
  6.220e+01  6.420e+01  5.090e+01  6.840e+01  6.460e+01  6.670e+01
  6.640e+01  6.870e+01  5.680e+01  5.950e+01  7.430e+01  5.560e+01
  4.440e+01  5.160e+01  4.860e+01  6.150e+01  6.930e+01  6.480e+01
  8.620e+01  5.450e+01  8.350e+01  7.540e+01  4.730e+01  5.490e+01
  6.550e+01  7.790e+01  5.970e+01  6.190e+01  6.890e+01  8.130e+01
  7.630e+01  8.560e+01  8.280e+01  6.530e+01  9.540e+01  8.690e+01
  8.830e+01  7.050e+01  6.960e+01  7.380e+01  6.390e+01  7.480e+01
  6.350e+01  5.830e+01  9.090e+01  6.710e+01  7.230e+01  5.220e+01
  6.130e+01  4.980e+01  8.440e+01  7.950e+01  5.630e+01  8.020e+01
  7.970e+01  7.210e+01  6.330e+01  7.920e+01  7.650e+01  6.850e+01
  8.470e+01  7.990e+01  9.270e+01  7.270e+01  6.940e+01  8.700e+01
  8.370e+01  8.850e+01  9.600e+01  9.400e+01  9.210e+01  9.840e+01
  1.058e+02  8.500e+01  9.250e+01  9.070e+01  9.010e+01  8.300e+01
  7.030e+01  8.510e+01  8.000e+01  7.660e+01  9.340e+01  8.900e+01
  9.770e+01  7.400e+01  7.410e+01  7.800e+01  7.570e+01  9.200e+01
  9.160e+01  9.410e+01  7.470e+01  6.660e+01  6.760e+01  7.600e+01
  8.550e+01  1.090e+02  6.500e+01  9.280e+01  1.063e+02  7.770e+01
  9.430e+01  1.010e+02  1.051e+02  7.450e+01  1.130e+02  9.590e+01
  1.071e+02  8.730e+01  9.550e+01  9.880e+01  7.860e+01  1.006e+02
  9.520e+01  9.460e+01  7.180e+01  1.013e+02  1.024e+02  1.018e+02
  9.730e+01  9.180e+01  8.650e+01  8.380e+01  7.070e+01  8.170e+01
  9.450e+01  6.200e+01  6.000e+01  7.100e+01  9.120e+01  8.260e+01
  7.090e+01  7.290e+01  7.720e+01  8.150e+01  7.560e+01  6.700e+01
  5.800e+01  7.610e+01  9.030e+01  7.320e+01  7.840e+01  1.110e+02
  9.910e+01  9.630e+01  1.008e+02  8.460e+01  1.099e+02  9.640e+01
  1.030e+02  1.050e+02  1.067e+02  1.094e+02  1.112e+02  1.070e+02
```

```
 1.017e+02  1.080e+02  1.085e+02  9.230e+01  9.990e+01  8.820e+01
 9.800e+01  9.930e+01  8.640e+01  8.330e+01  1.031e+02  1.033e+02
 5.600e+01  5.300e+01  5.130e+01  5.340e+01  4.950e+01  3.870e+01
 4.700e+01  4.400e+01  5.100e+01  4.930e+01  5.140e+01  4.780e+01
 4.330e+01  5.270e+01  4.910e+01  4.530e+01  3.520e+01  3.650e+01
 3.720e+01  3.450e+01  4.420e+01  4.240e+01  4.390e+01  4.230e+01
 5.070e+01  5.230e+01  6.370e+01  6.060e+01  4.480e+01  4.620e+01
 4.200e+01  3.960e+01  3.800e+01  3.200e+00  2.970e+01  4.050e+01
 4.000e+01  7.120e+01  6.750e+01  8.980e+01  6.570e+01  6.900e+01
 8.040e+01  8.310e+01  6.730e+01  4.010e+01  5.020e+01  4.510e+01
 4.120e+01  9.790e+01  9.720e+01  1.047e+02  8.220e+01  9.480e+01
 9.050e+01  8.940e+01  8.580e+01  1.076e+02  1.027e+02  1.141e+02
 8.890e+01  9.750e+01  9.820e+01  9.360e+01  4.260e+01  8.740e+01
 9.370e+01  1.101e+02  1.062e+02  1.072e+02  1.096e+02  1.105e+02
 1.103e+02  1.128e+02  1.108e+02  1.042e+02  1.092e+02  3.760e+01
 3.470e+01  3.880e+01  4.170e+01  3.130e+01  2.460e+01  3.670e+01
 2.750e+01  2.820e+01  2.570e+01  2.880e+01  4.570e+01  2.340e+01
 2.530e+01  2.910e+01  1.810e+01  1.830e+01  1.630e+01  1.780e+01
 2.070e+01  2.210e+01  2.440e+01  2.320e+01  1.130e+01  2.370e+01
 2.640e+01  8.600e+00  6.600e+00  1.170e+01  2.350e+01  1.000e+00
 1.740e+01  9.500e+00  1.900e+00  1.150e+01  1.450e+01  1.940e+01
 2.390e+01  2.010e+01  1.720e+01  2.030e+01  1.960e+01  1.400e+00
 1.220e+01  2.080e+01  2.170e+01  1.920e+01  1.310e+01 -6.000e+00
-7.600e+00 -7.100e+00 -4.000e+00  3.000e+00  1.020e+01  1.240e+01
 7.200e+00  9.100e+00  8.800e+00  7.900e+00  9.300e+00  1.180e+01
 1.420e+01  6.800e+00 -2.000e+00  2.710e+01  2.930e+01  2.620e+01
 2.680e+01  2.730e+01  4.080e+01  3.850e+01  1.470e+01  1.890e+01
 1.690e+01  1.380e+01  1.440e+01  1.350e+01  8.200e+00  7.300e+00
 8.400e+00  1.110e+01 -0.000e+00 -2.200e+00  2.300e+00  5.500e+00
 5.900e+00  1.650e+01  3.270e+01  1.270e+01  1.330e+01  1.490e+01
 2.230e+01  1.670e+01  3.150e+01  2.860e+01  3.810e+01 -2.900e+00
-9.000e-01 -6.000e-01  7.000e-01  3.600e+00  2.700e+00  1.200e+00
 4.600e+00  1.560e+01  4.800e+00 -1.300e+01 -5.100e+00 -1.190e+01
-9.000e+00 -7.400e+00 -4.000e-01 -2.600e+00  1.800e+00  4.100e+00
 5.700e+00  1.260e+01  9.700e+00  1.530e+01  2.050e+01  3.700e+00
-2.000e-01  6.400e+00  1.980e+01  1.326e+02  2.430e+01  1.870e+01
 7.700e+00  6.300e+00  9.900e+00  5.400e+00 -2.400e+00  2.260e+01
 1.060e+01  2.770e+01 -8.000e+00 -7.780e+01  5.200e+00  3.400e+00
 2.140e+01  1.080e+01  3.300e+01  1.100e+01  1.500e+01  3.500e+01
 2.600e+01  1.300e+01  1.436e+02  1.364e+02  1.166e+02  1.220e+02
 1.184e+02  1.148e+02  1.328e+02  1.616e+02  1.400e+02  9.970e+01
 2.500e+00  1.238e+02  3.100e+01  2.200e+01  1.150e+02  1.038e+02
 1.119e+02  1.002e+02  1.161e+02  1.069e+02  1.670e+02  1.100e+02
 2.900e+01  2.400e+01  2.000e+01  1.700e+01 -8.900e+01  8.000e+00
-3.000e+00 -5.000e+00  2.000e+00  6.000e+00 -1.800e+01  4.000e+00
-1.000e+00 -1.100e+01 -7.000e+00 -1.900e+01 -1.000e+01 -1.600e+01
-1.500e+01 -2.100e+01 -1.400e+01 -1.200e+01  1.120e+02 -2.200e+01
-3.300e+01 -2.400e+01 -1.410e+01 -9.900e+00 -5.800e+00 -8.000e-01
-1.030e+01 -9.400e+00 -8.100e+00 -1.700e+00 -1.120e+01  9.000e-01
-1.320e+01 -1.620e+01  1.000e-01 -3.500e+00 -4.500e+00  2.800e+00
 4.300e+00 -1.230e+01 -1.660e+01 -2.020e+01 -1.300e+00 -3.100e+00
-1.520e+01  4.500e+00 -4.700e+00 -4.400e+00  5.000e-01 -1.610e+01
-1.700e+01 -5.300e+00 -1.800e+00 -2.000e+01 -2.310e+01 -2.700e+01
-1.890e+01 -1.590e+01 -3.280e+01 -5.600e+00 -2.560e+01 -2.600e+01
-1.840e+01 -1.820e+01 -2.090e+01 -2.380e+01 -1.480e+01 -2.790e+01
-2.510e+01 -2.240e+01 -2.900e+01 -1.500e+00  3.000e-01 -4.900e+00
-7.200e+00 -1.250e+01 -1.160e+01 -8.500e+00 -8.700e+00 -8.900e+00
-7.800e+00 -6.700e+00 -1.010e+01 -1.140e+01 -1.070e+01 -5.400e+00
-4.200e+00 -2.700e+00 -1.280e+01 -1.340e+01 -1.170e+01  2.100e+00
-3.300e+00 -6.500e+00 -6.200e+00 -1.260e+01 -1.080e+01 -1.050e+01
-6.900e+00 -9.600e+00 -1.390e+01 -1.930e+01 -1.530e+01 -2.650e+01
-2.110e+01 -2.490e+01 -2.450e+01 -2.340e+01 -2.130e+01 -2.740e+01
-1.680e+01 -2.420e+01 -2.150e+01 -2.990e+01  1.292e+02  1.011e+02
 1.049e+02  1.015e+02  1.044e+02  1.053e+02  1.054e+02  1.045e+02
 1.035e+02  1.056e+02  1.078e+02  1.117e+02  1.114e+02  1.170e+02
 1.074e+02  1.081e+02  1.139e+02  1.180e+02  1.134e+02  1.116e+02
 1.274e+02  1.087e+02 -6.300e+00 -3.600e+00 -1.100e+00 -8.300e+00
 1.600e+00 -1.440e+01  1.065e+02  1.580e+02  1.098e+02 -2.500e+01
 1.706e+02 -9.800e+00 -2.330e+01 -9.200e+00 -1.350e+01 -1.790e+01
```

```
  1.688e+02 -4.000e+01 -1.210e+01 -3.800e+00]
****************************************
Wind_Chill(F) : 975
Data :  [      nan  3.33e+01  3.10e+01  3.55e+01  3.38e+01  3.07e+01  3.11e+01
  3.21e+01  3.03e+01  2.96e+01  2.86e+01  3.24e+01  3.09e+01  3.44e+01
  2.90e+01  3.29e+01  3.45e+01  3.59e+01  3.16e+01  3.12e+01  2.61e+01
  2.55e+01  1.17e+01  1.15e+01  1.82e+01  1.06e+01  1.24e+01  6.70e+00
  1.55e+01  1.01e+01  1.50e+01  1.61e+01  9.80e+00  8.30e+00  8.90e+00
  1.04e+01  2.25e+01  9.50e+00  1.35e+01  4.70e+00  2.70e+00  1.00e+00
 -1.40e+00  4.20e+00  6.10e+00  1.70e+00  4.40e+00  5.00e+00  3.40e+00
  3.00e+00  8.60e+00  7.30e+00  2.90e+00  5.00e-01  1.20e+00 -5.70e+00
 -3.00e-01 -6.80e+00 -2.10e+00 -1.60e+00 -1.10e+00  1.50e+00 -2.00e-01
  2.00e-01  7.70e+00  2.20e+00  8.70e+00  7.50e+00  9.30e+00  1.30e+01
  1.46e+01  1.14e+01  1.43e+01  1.29e+01  1.11e+01  1.60e+00  3.20e+00
 -4.50e+00 -8.40e+00 -8.30e+00 -1.80e+00 -3.40e+00  4.00e-01  1.39e+01
  1.45e+01  1.36e+01  1.58e+01  1.70e+01  1.79e+01  1.76e+01  1.71e+01
  1.94e+01  2.30e+01  2.64e+01  2.48e+01  2.35e+01  2.77e+01  3.06e+01
  2.80e+01  2.58e+01  2.36e+01  2.22e+01  2.56e+01  2.11e+01  2.42e+01
  2.28e+01  2.87e+01  2.32e+01  2.49e+01  2.54e+01  2.41e+01  2.79e+01
  2.53e+01  2.69e+01  2.12e+01  2.75e+01  2.17e+01  2.08e+01  2.24e+01
  2.65e+01  2.52e+01  2.45e+01  2.34e+01  1.80e+01  2.39e+01  3.22e+01
  2.85e+01  6.90e+00  6.30e+00  1.95e+01  1.25e+01  3.34e+01  3.15e+01
  2.82e+01  2.89e+01  2.93e+01  3.08e+01  2.84e+01  2.71e+01  2.97e+01
  3.71e+01  3.47e+01  3.70e+01  3.92e+01  2.40e+01  2.81e+01  3.01e+01
  2.66e+01  3.43e+01  3.80e+01  3.88e+01  2.19e+01  2.70e+01  2.67e+01
  3.97e+01  3.39e+01  2.73e+01  2.98e+01  3.49e+01  3.48e+01  3.25e+01
  4.50e+01  3.75e+01  3.68e+01  3.31e+01  4.06e+01  3.79e+01  2.59e+01
  3.90e+01  2.02e+01  1.78e+01  2.43e+01  2.16e+01  2.44e+01  1.92e+01
  3.77e+01  2.74e+01  2.47e+01  3.00e+01  3.28e+01  3.20e+01  3.36e+01
  3.72e+01  4.14e+01  4.10e+01  4.26e+01  3.04e+01  1.48e+01  1.93e+01
  2.09e+01  2.00e+01  2.60e+01  2.72e+01  2.83e+01  2.33e+01  3.66e+01
  4.24e+01  6.60e+01  4.07e+01  4.03e+01  3.89e+01  4.08e+01  4.01e+01
  3.95e+01  4.13e+01  3.76e+01  4.11e+01  4.43e+01  4.22e+01  4.31e+01
  4.18e+01  4.39e+01  4.02e+01  3.17e+01  3.84e+01  4.52e+01  4.04e+01
  4.25e+01  3.57e+01  4.19e+01  4.35e+01  4.15e+01  4.48e+01  3.94e+01
  3.50e+01  3.67e+01  3.93e+01  3.78e+01  3.35e+01  3.64e+01  3.58e+01
  2.92e+01  3.51e+01  3.81e+01  1.67e+01  3.99e+01  1.81e+01  4.12e+01
  4.29e+01  3.30e+01  3.86e+01  3.14e+01  2.07e+01  1.98e+01  2.05e+01
  3.42e+01  3.52e+01  3.27e+01  3.91e+01  3.23e+01  3.82e+01  1.91e+01
  3.05e+01  3.98e+01  3.18e+01  2.99e+01  2.88e+01  2.46e+01  3.74e+01
  3.53e+01  3.85e+01  3.69e+01  3.19e+01  3.87e+01  3.62e+01  8.70e+01
  9.60e+01  9.40e+01  8.50e+01  8.30e+01  7.90e+01  8.00e+01  9.00e+01
  8.90e+01  8.20e+01  7.40e+01  7.00e+01  6.30e+01  7.80e+01  8.40e+01
  9.20e+01  7.50e+01  7.30e+01  6.80e+01  7.60e+01  8.10e+01  8.80e+01
  7.70e+01  6.40e+01  6.50e+01  9.70e+01  7.20e+01  1.01e+02  9.10e+01
  6.10e+01  6.20e+01  6.00e+01  7.10e+01  8.60e+01  6.70e+01  5.80e+01
  9.30e+01  9.50e+01  9.90e+01  1.02e+02  1.03e+02  1.05e+02  1.06e+02
  1.07e+02  9.80e+01  1.00e+02  5.20e+01  5.60e+01  5.30e+01  4.80e+01
  3.60e+01  4.40e+01  4.60e+01  4.70e+01  5.70e+01  5.00e+01  5.10e+01
  5.40e+01  5.90e+01  4.30e+01  4.90e+01  5.50e+01  4.16e+01  4.46e+01
  4.00e+01  4.09e+01  3.40e+01  2.57e+01  4.20e+01  1.97e+01  4.27e+01
  3.96e+01  3.63e+01  2.62e+01  2.26e+01  6.90e+01  4.37e+01  4.17e+01
  2.21e+01  4.28e+01  1.09e+02  1.08e+02  1.11e+02  3.54e+01  3.61e+01
  2.04e+01  1.86e+01  1.75e+01  1.73e+01  1.87e+01  3.41e+01  2.50e+01
  2.38e+01  2.63e+01  2.18e+01  2.51e+01  4.21e+01  4.05e+01  1.53e+01
  1.47e+01  5.50e+00  6.00e+00  1.27e+01  1.52e+01  1.68e+01  3.56e+01
  3.73e+01  2.68e+01  2.95e+01  4.34e+01  1.56e+01  1.28e+01  1.57e+01
  1.41e+01  1.44e+01  1.31e+01  1.84e+01  2.06e+01  9.90e+00  6.00e-01
  1.00e+01  5.80e+00  4.10e+00  8.40e+00  5.60e+00  5.70e+00  8.20e+00
  1.42e+01  6.60e+00  1.26e+01  7.80e+00  7.60e+00 -6.50e+00 -3.60e+00
 -7.00e-01  4.50e+00  3.10e+00  1.20e+01  9.70e+00  1.66e+01 -1.30e+00
  1.08e+01  1.23e+01  1.64e+01  1.85e+01 -1.29e+01  2.00e+00 -6.90e+00
  9.10e+00 -7.70e+00 -3.50e+00 -1.28e+01 -1.69e+01 -1.50e+00 -7.00e+00
 -9.00e-01  3.50e+00  4.60e+00  5.20e+00  1.34e+01  6.50e+00 -2.60e+00
  1.40e+01  2.60e+00 -8.00e-01  1.10e+01  7.90e+00 -2.90e+00 -9.90e+00
  4.00e+00  1.80e+00  5.10e+00 -4.30e+00  8.00e+00  2.50e+00 -3.20e+00
  1.02e+01  1.37e+01  1.09e+01  1.74e+01 -5.80e+00 -1.90e+01 -1.17e+01
 -1.92e+01 -1.88e+01 -1.99e+01 -1.78e+01 -9.30e+00 -2.70e+00 -4.10e+00
 -6.60e+00 -1.15e+01 -1.33e+01 -6.10e+00 -2.30e+01 -1.25e+01 -9.80e+00
 5.30e+00 -8.60e+00 -1.96e+01 -5.40e+00 -1.61e+01  5.40e+00  2.03e+01
 4.80e+00  4.32e+01  3.46e+01  3.26e+01  3.32e+01  1.69e+01 -3.80e+00
 1.90e+00  7.20e+00  4.90e+00 -8.80e+00  3.70e+00 -1.00e-01  3.30e+00
-1.10e+01 -3.70e+00  4.30e+00  3.00e-01  1.40e+00 -6.20e+00 -5.10e+00
-1.08e+01 -1.09e+01 -7.90e+00 -8.20e+00 -1.04e+01 -9.50e+00 -2.30e+00
-2.20e+00 -2.40e+00 -4.00e+00 -6.00e-01 -1.42e+01 -5.20e+00 -5.60e+00
 3.60e+00 -1.16e+01  9.40e+00  6.40e+00  1.10e+00  7.40e+00  2.31e+01
 2.27e+01  1.65e+01  1.13e+01  1.30e+00  1.19e+01  1.18e+01  1.83e+01
 1.60e+01  2.40e+00  3.80e+00  1.21e+01  1.49e+01  1.12e+01  9.60e+00
 1.63e+01  2.91e+01  3.02e+01  2.15e+01  4.23e+01  2.94e+01  2.01e+01
 2.14e+01  1.51e+01  1.33e+01  1.38e+01  1.77e+01  1.72e+01  1.62e+01
 1.22e+01  1.89e+01  4.42e+01  2.76e+01  2.37e+01  2.23e+01  2.20e+01
 4.44e+01  4.36e+01  3.65e+01  4.33e+01  7.10e+00  9.00e+00 -4.00e-01
 3.90e+00  8.10e+00  9.20e+00  8.80e+00  6.80e+00  6.20e+00  2.30e+00
 5.90e+00  9.00e-01 -2.00e+00 -4.90e+00 -4.20e+00 -8.00e+00 -1.70e+00
-1.23e+01 -5.90e+00 -1.02e+01 -1.90e+00  7.00e-01  1.00e-01 -7.30e+00
-2.50e+00 -8.90e+00 -3.90e+00 -8.70e+00 -1.62e+01 -1.71e+01 -2.08e+01
-1.66e+01 -1.22e+01 -1.53e+01 -1.58e+01 -1.80e+01 -1.70e+01 -1.41e+01
-1.47e+01 -1.13e+01 -1.48e+01 -1.00e+01 -1.26e+01 -7.80e+00 -9.10e+00
-7.50e+00 -5.30e+00 -4.40e+00 -8.50e+00 -6.00e+00 -8.10e+00 -1.49e+01
-1.50e+01 -2.16e+01 -2.27e+01 -1.76e+01 -2.73e+01 -2.38e+01 -2.15e+01
-2.26e+01 -1.65e+01 -2.80e+00 -2.25e+01 -2.93e+01 -2.05e+01 -1.91e+01
-1.86e+01 -1.40e+01 -2.21e+01 -1.56e+01 -1.20e+01 -1.55e+01 -1.87e+01
-1.11e+01 -1.37e+01 -1.21e+01 -9.70e+00 -7.60e+00 -9.00e+00 -7.10e+00
-3.30e+00 -3.10e+00 -4.70e+00 -3.00e+00  1.07e+01  1.99e+01 -4.80e+00
-9.60e+00 -1.63e+01 -1.27e+01 -1.32e+01 -1.38e+01 -1.18e+01 -5.50e+00
-9.40e+00 -9.20e+00  1.90e+01  3.37e+01  3.83e+01  3.13e+01  2.13e+01
 1.16e+01  1.32e+01  1.96e+01  1.54e+01  0.00e+00 -1.00e+00 -1.07e+01
-7.40e+00 -5.00e+00 -1.12e+01  1.59e+01 -1.39e+01 -1.03e+01 -5.00e-01
 2.10e+00  2.80e+00  2.78e+01 -1.52e+01 -1.94e+01 -1.43e+01  2.10e+01
 8.50e+00  1.05e+01 -1.05e+01  8.00e-01  1.03e+01  2.29e+01  4.41e+01
-7.20e+00  7.00e+00 -1.45e+01  1.88e+01  1.04e+02  1.10e+02 -8.90e+01
-1.60e+01 -2.10e+01 -2.20e+01 -2.00e+01 -2.80e+01 -2.50e+01 -2.90e+01
-1.30e+01 -3.90e+01 -3.20e+01 -2.60e+01 -2.70e+01 -2.40e+01 -3.00e+01
-3.30e+01  1.12e+02  1.13e+02  1.15e+02 -5.40e+01 -4.80e+01 -3.80e+01
-2.99e+01 -2.24e+01 -2.12e+01 -2.31e+01 -1.57e+01 -3.13e+01 -2.75e+01
-2.84e+01 -1.06e+01 -1.31e+01 -2.34e+01 -2.19e+01 -2.04e+01 -2.52e+01
-2.46e+01 -2.41e+01 -2.59e+01 -2.02e+01 -2.39e+01 -1.36e+01 -1.74e+01
-1.89e+01 -1.34e+01 -2.72e+01 -2.43e+01 -2.88e+01 -1.51e+01 -2.44e+01
-1.84e+01 -2.37e+01 -1.83e+01 -1.81e+01 -2.17e+01 -2.29e+01 -2.06e+01
-1.82e+01 -1.75e+01 -2.07e+01 -6.70e+00 -6.30e+00 -1.35e+01 -1.01e+01
-6.40e+00 -2.76e+01 -1.14e+01 -3.57e+01 -3.55e+01 -1.77e+01 -1.73e+01
-1.98e+01 -1.54e+01 -2.33e+01 -3.10e+01 -2.83e+01 -3.09e+01 -3.66e+01
-3.43e+01 -3.01e+01 -3.47e+01 -3.24e+01 -3.27e+01 -2.35e+01 -1.64e+01
-2.63e+01 -1.46e+01 -2.61e+01 -3.99e+01 -2.74e+01 -3.93e+01 -4.02e+01
-1.67e+01 -2.96e+01 -2.14e+01 -1.93e+01 -3.86e+01 -3.68e+01 -3.42e+01
-3.89e+01 -3.75e+01 -3.62e+01 -3.40e+01 -3.17e+01 -2.64e+01 -2.36e+01
-4.57e+01 -2.22e+01 -4.34e+01 -3.14e+01 -2.86e+01 -2.01e+01 -1.68e+01
-3.71e+01 -3.97e+01 -3.98e+01 -4.16e+01 -4.01e+01 -3.61e+01 -3.35e+01
-3.58e+01 -4.25e+01 -3.72e+01 -4.10e+01 -3.91e+01 -4.47e+01 -2.87e+01
-3.23e+01 -3.48e+01 -3.26e+01 -1.24e+01 -3.03e+01 -2.13e+01 -1.59e+01
-2.23e+01 -2.47e+01 -2.09e+01 -3.02e+01 -3.37e+01 -3.06e+01 -3.05e+01
-3.11e+01 -3.21e+01 -3.04e+01 -2.91e+01 -2.95e+01 -2.98e+01 -2.28e+01
-2.68e+01 -3.46e+01 -2.51e+01 -1.79e+01 -1.72e+01 -2.45e+01 -2.32e+01
-2.18e+01 -1.97e+01 -1.85e+01 -2.85e+01 -2.62e+01 -2.71e+01 -3.34e+01
-1.19e+01 -2.57e+01 -2.66e+01 -3.22e+01 -3.16e+01 -3.36e+01 -2.94e+01
-3.33e+01 -3.39e+01 -3.60e+01 -4.06e+01 -4.40e+01 -4.64e+01 -4.54e+01
-4.93e+01 -4.78e+01 -4.73e+01 -5.06e+01 -5.05e+01 -5.13e+01 -5.22e+01
-5.27e+01 -4.99e+01 -2.89e+01 -4.48e+01 -4.35e+01 -4.72e+01 -4.84e+01
-4.66e+01 -4.45e+01 -4.86e+01 -4.70e+01 -4.60e+01 -5.20e+01 -4.75e+01
-4.91e+01 -5.23e+01 -5.01e+01 -4.98e+01 -5.51e+01 -4.92e+01 -2.55e+01
-2.58e+01 -3.12e+01 -2.77e+01 -2.56e+01 -3.56e+01 -3.52e+01 -2.67e+01
-2.78e+01 -3.15e+01 -3.63e+01 -5.12e+01 -3.29e+01 -5.10e+01 -4.96e+01
-5.26e+01 -5.36e+01 -5.35e+01 -4.85e+01 -4.28e+01 -4.22e+01 -3.84e+01
-5.16e+01 -2.97e+01 -2.42e+01 -1.44e+01 -5.17e+01 -5.21e+01 -5.32e+01
-5.09e+01 -5.03e+01 -5.31e+01 -5.14e+01 -5.07e+01 -4.23e+01 -4.13e+01
-3.44e+01 -3.77e+01 -3.64e+01 -5.15e+01 -4.59e+01 -4.89e+01 -4.09e+01
-5.41e+01 -4.55e+01 -4.36e+01 -3.82e+01 -4.68e+01 -3.88e+01 -2.11e+01
-2.48e+01 -2.53e+01 -3.18e+01 -2.49e+01 -2.79e+01 -3.50e+01 -3.79e+01
```

```
 -3.51e+01 -4.08e+01 -3.94e+01 -4.60e+00 -2.92e+01 -3.38e+01 -3.08e+01
 -3.45e+01 -4.15e+01 -3.73e+01 -4.00e+01 -3.70e+01 -4.20e+01 -4.30e+01
 -3.53e+01 -3.81e+01 -3.31e+01 -4.53e+01 -5.04e+01 -3.28e+01 -5.90e+01
 -3.85e+01 -4.32e+01 -4.44e+01 -4.77e+01 -3.59e+01 -4.05e+01 -6.59e+01
 -2.03e+01 -2.69e+01]
****************************************
Humidity(%) : 101
Data :  [ 91. 100.  96.  89.  97.  99.  93.  76.  86.  70.  65.  75.  92.  85.
  88.  84.  90.  81.  82.  73.  77.  79.  78.  56.  74.  87.  59.  83.
  80.  51.  54.  68.  62.  49.  57.  66.  71.  44.  61.  58.  43.  98.
  94.  72.  67.  69.  63.  60.  53.  55.  52.  50.  45.  37.  47.  34.
  46.  64.  35.  32.  40.  41.  42.  39.  48.  31.  95.  nan  38.  30.
  26.  23.  25.  21.  24.  20.  29.  27.  19.  18.  17.  12.  11.  16.
  33.  13.  10.  15.  22.  36.  28.  14.   9.   8.   4.   7.   6.   5.
   3.   2.   1.]
****************************************
Pressure(in) : 1023
Data :  [29.68 29.65 29.67 ... 22.   21.95 21.61]
****************************************
Visibility(mi) : 86
Data :  [1.00e+01 9.00e+00 6.00e+00 7.00e+00 5.00e+00 3.00e+00 2.00e+00 8.00e+00
 2.50e+00 4.00e+00 1.50e+00 8.00e-01 1.80e+00 1.00e+00 1.20e+00 5.00e-01
 2.00e-01 2.80e+00      nan 3.00e+01 2.50e+01 2.00e+01 4.00e+01 8.00e+01
 1.00e-01 1.50e+01 7.00e-01 1.90e-01 4.00e-01 4.20e+00 1.20e-01 1.20e+01
 3.20e+00 5.50e+00 0.00e+00 2.50e-01 2.20e+00 7.50e-01 1.05e+02 6.00e-01
 1.10e+00 3.50e+00 1.30e+01 1.10e+01 1.05e+01 1.11e+02 1.40e+00 1.90e+00
 9.00e-01 6.00e-02 3.50e+01 3.80e-01 7.50e+01 8.80e-01 5.00e+01 1.00e+02
 7.00e+01 6.00e+01 6.30e-01 4.50e+01 3.10e-01 1.90e+01 1.40e+01 7.60e+01
 1.60e+01 1.01e+02 1.60e+00 9.00e+01 2.10e+00 3.70e+00 7.20e+01 5.80e+01
 4.50e+00 6.70e+01 3.40e+01 6.20e+00 1.40e+02 3.60e+01 4.60e+01 5.40e+01
 4.70e+01 2.20e+01 1.10e+02 1.30e+02 6.30e+01 4.30e+01]
****************************************
Wind_Direction : 25
Data :  ['Calm' 'SW' 'SSW' 'WSW' 'WNW' 'NW' 'West' 'NNW' 'NNE' 'South' 'North'
 'Variable' 'SE' 'SSE' 'ESE' 'East' 'NE' 'ENE' 'E' 'W' nan 'S' 'VAR'
 'CALM' 'N']
****************************************
Wind_Speed(mph) : 161
Data :  [  nan   3.5   4.6   1.2   5.8   2.3   6.9   8.1  10.4   9.2  11.5  13.8
  15.   12.7  19.6  21.9  18.4  25.3  16.1  24.2  23.   17.3  27.6  29.9
  20.7  10.   26.5   5.   31.1  33.4  28.8  35.7  42.6  36.8  32.2  40.3
 142.7  73.6  69.   38.    8.    9.    3.   14.    7.    0.   12.    6.
  13.    1.    2.   47.2  17.   16.   21.   28.   20.   26.   18.   22.
  57.5  34.5 241.7  24.  100.1 123.1 822.8  41.4 162.3  66.7  30.   29.
 126.6 127.7  25.   39.1  54.1  97.8  76.  174.9  44.9  31.   37.   32.
  33.   35.   46.   66.   40.   64.   51.   36.  117.   48.   39.   93.
  41.   52.   47.  230.  255.   82.   44.   67.   49.   58.   53.   43.
 161.  116.  113.  127.  157.  175.   49.5  43.7  77.1  51.8 116.2 119.7
 703.1  79.4  61.  254.3 110.5  50.6 124.3 328.  580.  135.8 128.9  48.3
 208.3  62.1 214.  125.4  60.   58.7  81.7 166.9  85.2 471.8 232.  131.
 105.  984.   55.  129.  518.   98.   54.  169.  130.   59.8 114.   99.
 245.1 141.5  45.  142.  110. ]
****************************************
Precipitation(in) : 262
Data :  [2.000e-02 0.000e+00       nan 3.000e-02 1.000e-02 7.000e-02 4.000e-02
 6.000e-02 1.800e-01 5.000e-02 1.600e-01 9.000e-02 1.000e-01 1.100e-01
 2.200e-01 8.000e-02 1.900e-01 1.500e-01 1.200e-01 1.400e-01 2.100e-01
 2.900e-01 1.300e-01 4.100e-01 2.000e-01 4.900e-01 3.100e-01 3.200e-01
 1.700e-01 2.500e-01 2.400e-01 2.300e-01 3.400e-01 4.400e-01 5.100e-01
 3.600e-01 2.700e-01 2.600e-01 5.500e-01 4.300e-01 4.700e-01 3.500e-01
 2.800e-01 4.200e-01 3.000e-01 3.300e-01 3.800e-01 5.600e-01 4.000e-01
 7.000e-01 4.600e-01 3.700e-01 5.700e-01 5.400e-01 6.100e-01 1.310e+00
 7.600e-01 1.680e+00 1.040e+00 1.080e+00 8.100e-01 6.800e-01 6.700e-01
 6.900e-01 4.500e-01 7.100e-01 1.020e+00 5.200e-01 8.600e-01 8.900e-01
 5.000e-01 9.000e-01 1.330e+00 5.900e-01 6.300e-01 7.200e-01 7.700e-01
 1.200e+00 5.800e-01 3.900e-01 7.800e-01 8.300e-01 8.500e-01 5.300e-01
 1.010e+00 4.800e-01 1.270e+00 6.200e-01 6.500e-01 6.400e-01 7.300e-01
 8.700e-01 1.700e+00 8.000e-01 9.100e-01 1.060e+00 9.500e-01 8.200e-01
 8.400e-01 1.720e+00 1.890e+00 1.160e+00 1.530e+00 9.990e+00 1.002e+01
 1.001e+01 9.940e+00 9.920e+00 9.930e+00 1.011e+01 9.850e+00 9.960e+00
 9.980e+00 1.005e+01 1.014e+01 1.004e+01 9.750e+00 9.970e+00 9.790e+00
 1.000e+01 9.900e+00 9.830e+00 1.210e+00 1.010e+01 1.013e+01 9.640e+00
 9.840e+00 9.880e+00 9.950e+00 1.006e+01 6.000e-01 9.300e-01 1.450e+00
 1.030e+00 1.170e+00 9.800e-01 1.340e+00 1.000e+00 1.370e+00 1.130e+00
 2.280e+00 6.600e-01 1.960e+00 2.940e+00 1.320e+00 7.900e-01 9.900e-01
 1.280e+00 2.390e+00 1.110e+00 1.460e+00 2.820e+00 1.390e+00 1.430e+00
 1.150e+00 9.400e-01 1.290e+00 9.910e+00 7.400e-01 2.750e+00 9.200e-01
 1.100e+00 8.800e-01 9.860e+00 1.009e+01 9.330e+00 1.250e+00 1.120e+00
 1.580e+00 1.600e+00 1.400e+00 2.690e+00 1.440e+00 7.500e-01 2.440e+00
 9.650e+00 1.650e+00 1.140e+00 1.500e+00 2.610e+00 1.190e+00 1.220e+00
 1.230e+00 1.350e+00 1.490e+00 9.600e-01 1.050e+00 1.790e+00 1.380e+00
 1.070e+00 1.180e+00 9.700e-01 1.620e+00 1.470e+00 2.070e+00 1.260e+00
 1.090e+00 1.810e+00 2.500e+01 1.750e+00 2.960e+00 1.860e+00 1.760e+00
 1.840e+00 1.670e+00 1.780e+00 3.260e+00 1.410e+00 1.550e+00 1.520e+00
 1.770e+00 1.240e+00 2.020e+00 1.730e+00 1.990e+00 2.300e+00 1.420e+00
 3.350e+00 2.800e+00 2.810e+00 2.530e+00 2.430e+00 1.570e+00 2.010e+00
 1.080e+01 5.060e+00 1.540e+00 1.690e+00 1.300e+00 2.260e+00 1.360e+00
 2.310e+00 9.170e+00 1.016e+01 1.018e+01 1.900e+00 1.820e+00 1.480e+00
 4.090e+00 2.400e+00 2.550e+00 1.590e+00 1.560e+00 1.920e+00 2.090e+00
 1.870e+00 1.710e+00 2.080e+00 1.610e+00 2.620e+00 2.040e+00 2.730e+00
 2.200e+00 2.160e+00 1.970e+00 1.950e+00 1.640e+00 2.400e+01 2.270e+00
 2.100e+00 1.880e+00 1.510e+00]
****************************************
Weather_Condition : 128
Data :  ['Light Rain' 'Overcast' 'Mostly Cloudy' 'Rain' 'Light Snow' 'Haze'
 'Scattered Clouds' 'Partly Cloudy' 'Clear' 'Snow'
 'Light Freezing Drizzle' 'Light Drizzle' 'Fog' 'Shallow Fog' 'Heavy Rain'
 'Light Freezing Rain' 'Cloudy' 'Drizzle' nan 'Light Rain Showers' 'Mist'
 'Smoke' 'Patches of Fog' 'Light Freezing Fog' 'Light Haze'
 'Light Thunderstorms and Rain' 'Thunderstorms and Rain' 'Fair'
 'Volcanic Ash' 'Blowing Sand' 'Blowing Dust / Windy' 'Widespread Dust'
 'Fair / Windy' 'Rain Showers' 'Mostly Cloudy / Windy'
 'Light Rain / Windy' 'Hail' 'Heavy Drizzle' 'Showers in the Vicinity'
 'Thunderstorm' 'Light Rain Shower' 'Light Rain with Thunder'
 'Partly Cloudy / Windy' 'Thunder in the Vicinity' 'T-Storm'
 'Heavy Thunderstorms and Rain' 'Thunder' 'Heavy T-Storm' 'Funnel Cloud'
 'Heavy T-Storm / Windy' 'Blowing Snow' 'Light Thunderstorms and Snow'
 'Heavy Snow' 'Low Drifting Snow' 'Light Ice Pellets' 'Ice Pellets'
 'Squalls' 'N/A Precipitation' 'Cloudy / Windy' 'Light Fog' 'Sand'
 'Snow Grains' 'Snow Showers' 'Heavy Thunderstorms and Snow'
 'Rain / Windy' 'Heavy Rain / Windy' 'Heavy Ice Pellets'
 'Light Snow / Windy' 'Heavy Freezing Rain' 'Small Hail'
 'Heavy Rain Showers' 'T-Storm / Windy' 'Patches of Fog / Windy'
 'Drizzle / Windy' 'Thunder / Windy' 'Wintry Mix' 'Squalls / Windy'
 'Rain Shower' 'Drizzle and Fog' 'Haze / Windy' 'Sand / Dust Whirlwinds'
 'Blowing Dust' 'Fog / Windy' 'Smoke / Windy' 'Wintry Mix / Windy'
 'Snow / Windy' 'Light Rain Shower / Windy' 'Heavy Snow / Windy'
 'Snow and Sleet' 'Light Freezing Rain / Windy' 'Light Drizzle / Windy'
 'Light Snow and Sleet' 'Partial Fog' 'Light Snow Shower'
 'Light Snow and Sleet / Windy' 'Freezing Rain' 'Blowing Snow / Windy'
 'Freezing Drizzle' 'Sleet' 'Light Sleet' 'Rain and Sleet' 'Heavy Sleet'
 'Light Snow Grains' 'Partial Fog / Windy' 'Light Snow with Thunder'
 'Widespread Dust / Windy' 'Sand / Dust Whirlwinds / Windy' 'Tornado'
 'Snow and Thunder' 'Snow and Sleet / Windy' 'Heavy Snow with Thunder'
 'Thunder / Wintry Mix / Windy' 'Light Snow Showers' 'Heavy Blowing Snow'
 'Light Hail' 'Heavy Smoke' 'Heavy Thunderstorms with Small Hail'
 'Light Thunderstorm' 'Heavy Freezing Drizzle' 'Light Blowing Snow'
 'Thunderstorms and Snow' 'Freezing Rain / Windy' 'Dust Whirls'
 'Sand / Dust Whirls Nearby' 'Heavy Rain Shower' 'Thunder and Hail'
 'Drifting Snow' 'Thunder and Hail / Windy']
****************************************
Amenity : 2
Data :  [False  True]
****************************************
Bump : 2
Data :  [False  True]
****************************************
Crossing : 2
```

```
Data :  [False  True]
****************************************
Give_Way : 2
Data :  [False  True]
****************************************
Junction : 2
Data :  [False  True]
****************************************
No_Exit : 2
Data :  [False  True]
****************************************
Railway : 2
Data :  [False  True]
****************************************
Roundabout : 2
Data :  [False  True]
****************************************
Station : 2
Data :  [False  True]
****************************************
Stop : 2
Data :  [False  True]
****************************************
Traffic_Calming : 2
Data :  [False  True]
****************************************
Traffic_Signal : 2
Data :  [False  True]
****************************************
Sunrise_Sunset : 3
Data :  ['Night' 'Day' nan]
****************************************
Civil_Twilight : 3
Data :  ['Night' 'Day' nan]
****************************************
Nautical_Twilight : 3
Data :  ['Night' 'Day' nan]
****************************************
Astronomical_Twilight : 3
Data :  ['Night' 'Day' nan]
****************************************
```

23
```
# Copying the original data
preprocessed_data = data.copy()
```

24
```
cols = ['End_Lat', 'End_Lng', 'Number', 'Temperature(F)', 'Wind_Chill(F)', 'Humidity(%)', 'Pressure(i
imp_mean = IterativeImputer(max_iter=1000)
preprocessed_data[cols] = imp_mean.fit_transform(preprocessed_data[cols])
```

25
```
preprocessed_data.City.fillna(preprocessed_data.City.value_counts().index[0], inplace=True)

preprocessed_data.Side.replace(' ', preprocessed_data.Side.value_counts().index[0], inplace=True)

preprocessed_data.Wind_Direction.fillna(preprocessed_data.Wind_Direction.value_counts().index[0], inp

preprocessed_data.Weather_Condition.fillna(preprocessed_data.Weather_Condition.value_counts().index[0

preprocessed_data.Sunrise_Sunset.fillna(preprocessed_data.Sunrise_Sunset.value_counts().index[0], inp

preprocessed_data.Civil_Twilight.fillna(preprocessed_data.Civil_Twilight.value_counts().index[0], inp

preprocessed_data.Nautical_Twilight.fillna(preprocessed_data.Nautical_Twilight.value_counts().index[0

preprocessed_data.Astronomical_Twilight.fillna(preprocessed_data.Astronomical_Twilight.value_counts()
preprocessed_data.Weather_Timestamp.fillna(method='ffill', inplace=True)
```

26
```
preprocessed_data.isna().sum()
```

26
```
Severity                 0
Start_Time               0
End_Time                 0
Start_Lat                0
Start_Lng                0
End_Lat                  0
End_Lng                  0
Distance(mi)             0
Number                   0
Street                   0
Side                     0
City                     0
County                   0
State                    0
Weather_Timestamp        0
Temperature(F)           0
Wind_Chill(F)            0
Humidity(%)              0
Pressure(in)             0
Visibility(mi)           0
Wind_Direction           0
Wind_Speed(mph)          0
Precipitation(in)        0
Weather_Condition        0
Amenity                  0
Bump                     0
Crossing                 0
Give_Way                 0
Junction                 0
No_Exit                  0
Railway                  0
Roundabout               0
Station                  0
Stop                     0
Traffic_Calming          0
Traffic_Signal           0
Sunrise_Sunset           0
Civil_Twilight           0
Nautical_Twilight        0
Astronomical_Twilight    0
dtype: int64
```

Data Transformation

Label Encoding

27
```
encoder = LabelEncoder()

for col in preprocessed_data.columns:
    preprocessed_data[col] = encoder.fit_transform(preprocessed_data[col])
```

28
```
preprocessed_data.head()
```

28

	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi
0	2	1	22	770170	678272	1961513	1530629	11

	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi
1	1	3	1	776102	713475	1877904	1812466	11
2	1	5	2	707003	678727	1883392	734590	11
3	2	7	3	755433	672642	1977339	1161038	11
4	1	8	4	743260	673752	1890000	430397	11

Handling Outliers

29
```
envelope = EllipticEnvelope()

predicted = envelope.fit_predict(preprocessed_data)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/covariance/_robust_covariance.py:170: RuntimeWarning:

Determinant has increased; this should not happen: log(det) > log(previous_det) (319.581089392016906 

/usr/local/lib/python3.6/dist-packages/sklearn/covariance/_robust_covariance.py:170: RuntimeWarning:

Determinant has increased; this should not happen: log(det) > log(previous_det) (316.192964686106109 
```

30
```
outlier_info = np.unique(predicted, return_counts = True)
print(outlier_info)

print("Number of outliers : {} ; Number of inliers : {}".format(outlier_info[1][0], outlier_info[1][1
```

```
(array([-1,  1]), array([ 351362, 3162255]))
Number of outliers : 351362 ; Number of inliers : 3162255
```

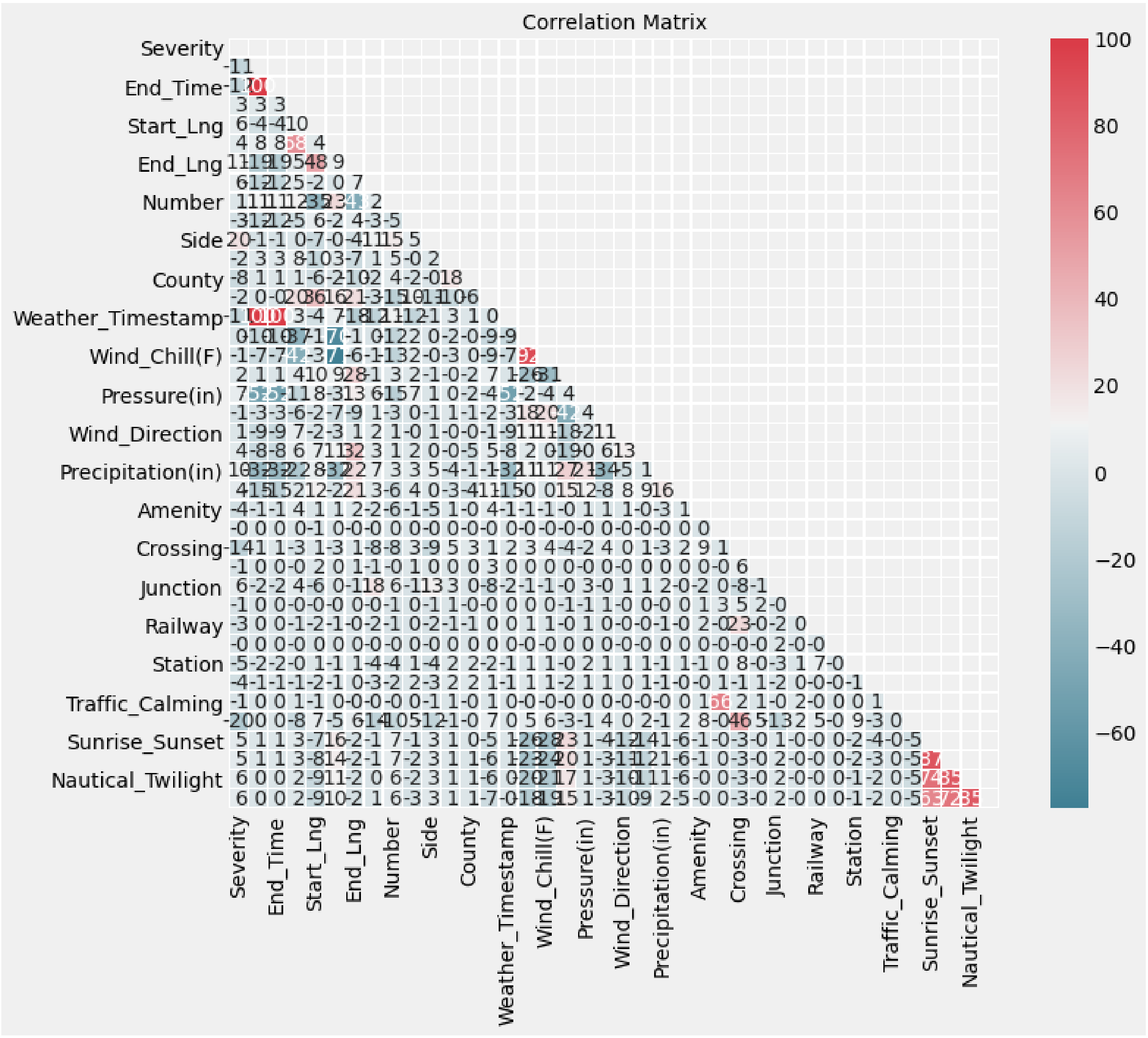
31
```
preprocessed_data = preprocessed_data[predicted == 1]

preprocessed_data.shape
```

31
```
(3162255, 40)
```

Data Integrity

Correlation Analysis

32
```
corrmat(preprocessed_data.corr(), inflate=True)
plt.show()
```



33
```
cor_matrix = preprocessed_data.corr().abs()
cor_matrix
```

33

	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	Enc
Severity	1.000000	0.114718	0.115338	0.034896	0.060168	0.039676	0.10
Start_Time	0.114718	1.000000	0.999934	0.032244	0.041849	0.075835	0.18
End_Time	0.115338	0.999934	1.000000	0.032271	0.042402	0.075520	0.18
Start_Lat	0.034896	0.032244	0.032271	1.000000	0.098380	0.580675	0.04
Start_Lng	0.060168	0.041849	0.042402	0.098380	1.000000	0.035010	0.48
End_Lat	0.039676	0.075835	0.075520	0.580675	0.035010	1.000000	0.09
End_Lng	0.108601	0.185267	0.185676	0.049351	0.480280	0.092127	1.00
Distance(mi)	0.058169	0.123619	0.123953	0.046310	0.021062	0.000999	0.06
Number	0.012641	0.111908	0.112140	0.118542	0.354463	0.231787	0.43
Street	0.031259	0.118213	0.119243	0.046608	0.064709	0.017780	0.03
Side	0.202869	0.013485	0.013236	0.001054	0.069610	0.004082	0.04

	Severity	Start_Time	End_Time	Start_Lat	Start_Lng	End_Lat	Enc
City	0.020971	0.027636	0.027822	0.084004	0.100057	0.033679	0.07
County	0.078456	0.011065	0.011196	0.012854	0.060447	0.024206	0.09
State	0.023754	0.000681	0.000233	0.202015	0.361708	0.164120	0.21
Weather_Timestamp	0.113641	0.999622	0.999449	0.032112	0.039267	0.074581	0.18
Temperature(F)	0.000439	0.095239	0.095200	0.373342	0.005561	0.697849	0.00
Wind_Chill(F)	0.014531	0.074210	0.074143	0.418021	0.029233	0.773967	0.05
Humidity(%)	0.020121	0.008324	0.007858	0.044394	0.100137	0.093594	0.27
Pressure(in)	0.069051	0.521592	0.522496	0.113034	0.075460	0.030619	0.13
Visibility(mi)	0.014649	0.028883	0.028714	0.055251	0.016334	0.073438	0.08
Wind_Direction	0.011697	0.089229	0.088957	0.069026	0.015748	0.029570	0.00
Wind_Speed(mph)	0.040220	0.084122	0.083768	0.059330	0.074222	0.106572	0.32
Precipitation(in)	0.097759	0.316891	0.316925	0.218784	0.077045	0.319916	0.21
Weather_Condition	0.044415	0.153184	0.153750	0.022883	0.121665	0.021744	0.20
Amenity	0.039980	0.007444	0.007458	0.037863	0.007736	0.006810	0.01
Bump	0.004120	0.003266	0.003269	0.002934	0.008693	0.001946	0.00
Crossing	0.143311	0.014446	0.014466	0.031485	0.014354	0.026587	0.01
Give_Way	0.008750	0.000275	0.000254	0.003541	0.016440	0.000440	0.00
Junction	0.062307	0.020752	0.020407	0.037728	0.063680	0.003079	0.01
No_Exit	0.006097	0.001594	0.001569	0.002418	0.001090	0.003583	0.00
Railway	0.025707	0.002161	0.002135	0.011118	0.018177	0.008161	0.00
Roundabout	0.004365	0.000083	0.000068	0.000412	0.001324	0.000456	0.00
Station	0.045275	0.015014	0.015136	0.000397	0.009146	0.010455	0.01
Stop	0.040539	0.007829	0.008118	0.011872	0.021531	0.008387	0.00
Traffic_Calming	0.005717	0.000666	0.000662	0.008944	0.005246	0.003169	0.00
Traffic_Signal	0.201221	0.003977	0.004171	0.083733	0.074773	0.047749	0.05
Sunrise_Sunset	0.047388	0.012666	0.012687	0.032302	0.068243	0.158998	0.01
Civil_Twilight	0.052139	0.009979	0.010069	0.026883	0.080515	0.136857	0.02
Nautical_Twilight	0.056582	0.004750	0.004907	0.018846	0.087879	0.114093	0.02
Astronomical_Twilight	0.057340	0.000171	0.000363	0.016277	0.094758	0.097839	0.02

34

```
upper_tri = cor_matrix.where(np.triu(np.ones(cor_matrix.shape),k=1).astype(np.bool))

to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > 0.95)]
```

```
print();
print("Highly Correlated columns to remove :{}".format(to_drop))
```

```
Highly Correlated columns to remove :['End_Time', 'Weather_Timestamp']
```

35
```
preprocessed_data = preprocessed_data.drop(to_drop, axis=1)
print();
preprocessed_data.shape
```

35
```
(3162255, 38)
```

Data Reduction

Principal Component Analysis (Performance Decreased)

Sampling Without Replacement

36
```
sampled_data = preprocessed_data.sample(100000, random_state = 40)
```

37
```
sampled_data.head()
```

37

	Severity	Start_Time	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance(mi)	N
220379	1	101462	840988	1053967	529212	773684	0	6
1204838	2	2159436	1011935	548066	1778307	591013	0	1
508514	1	662757	162821	368802	322368	646999	0	1
2244684	2	929459	11057	855622	362264	1223693	0	6
248986	1	220040	792127	1007736	1143922	1971170	0	1

Splitting Data Into Training And Testing

38
```
X = sampled_data.drop(['Severity'], axis = 1)
y = sampled_data['Severity']
```

39
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, test_size=0.3, random_sta
```

Classification

Defining Performance Function

```
40  def print_performance (clf, X_train, X_test, y_train, y_test, train=True):  # clf = classifier
        lb = preprocessing.LabelBinarizer()
        lb.fit(y_train)

        if train:
            # Training Performance
            res = clf.predict(X_train)

            print("Train Result:\n")

            print("Accuracy score: {0:.4f}\n"
                  .format(accuracy_score(y_train, res)))

            print("Error rate: {0:.4f}\n"
                  .format(1-accuracy_score(y_train, res)))

            print("recall of the positive class is also known as sensitivity ; recall of the negative cla
            print("Classification Report: \n {}"
                  .format(classification_report(y_train, res)))
            print("")

            cm = confusion_matrix(y_train, res)
            print("Confusion Matrix: \n {}\n".format(cm))

            if type(clf).__name__ is not 'StackingClassifier':
              disp = plot_confusion_matrix(clf, X_train, y_train,cmap=plt.cm.GnBu, values_format = 'd')
              plt.grid(False)
              plt.show()
              print()

            print("ROC AUC: {0:.4f}\n"
                  .format(roc_auc_score(lb.transform(y_train), lb.transform(res))))

            res = cross_val_score(clf, X_train, y_train, cv=10, scoring='accuracy', n_jobs=-1)

            print("Average Accuracy: \t {0:.4f}".format(np.mean(res)))
            print("Accuracy SD: \t\t {0:.4f}".format(np.std(res)))

        else:
            # Testing Performance
            res_test = clf.predict(X_test)

            print("Test Result:\n")

            print("Accuracy score: {0:.4f}\n"
                  .format(accuracy_score(y_test, res_test)))

            print("Error rate: {0:.4f}\n"
                  .format(1-accuracy_score(y_test, res_test)))

            print("recall of the positive class is also known as sensitivity ; recall of the negative cla
            print("Classification Report: \n {}\n"
                  .format(classification_report(y_test, res_test)))

            cm = confusion_matrix(y_test, res_test)
            print("Confusion Matrix: \n {}\n".format(cm))

            if type(clf).__name__ is not 'StackingClassifier':
              disp = plot_confusion_matrix(clf, X_test, y_test,cmap=plt.cm.Oranges, values_format = 'd')
              plt.grid(False)
              plt.show()
              print()

            print("ROC AUC: {0:.4f}\n"
                  .format(roc_auc_score(lb.transform(y_test), lb.transform(res_test))))

            res = cross_val_score(clf, X_test, y_test, cv=10, scoring='accuracy', n_jobs=-1)

            print("Average Accuracy: \t {0:.4f}".format(np.mean(res)))
        print("Accuracy SD: \t\t {0:.4f}".format(np.std(res)))
```

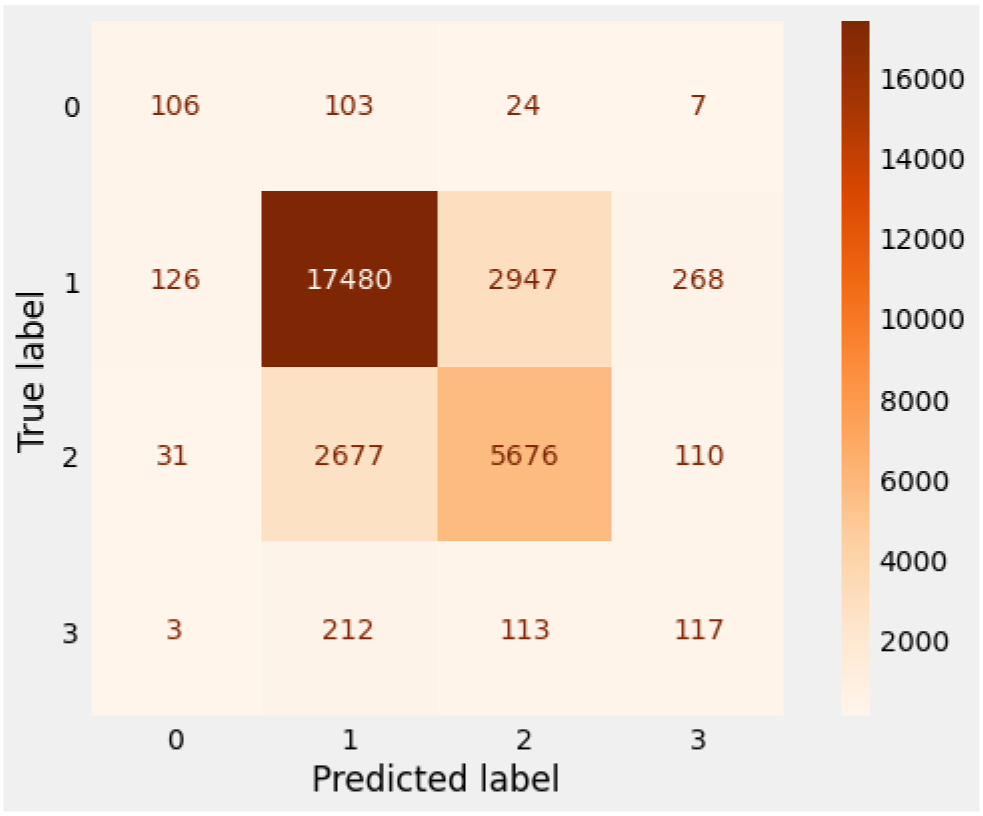
Defining Classification Function

41
```
def classify(clf):

  sig = inspect.signature(clf.__init__)

  try:
    sig.parameters['n_jobs']
    classifier = clf(n_jobs = -1)
  except Exception as e:
    classifier = clf()

  classifier.fit(X_train, y_train)


  print_performance(classifier, X_train, X_test, y_train, y_test, train=False)
  print("=" * 80)
  print_performance(classifier, X_train, X_test, y_train, y_test, train=True)

  return classifier
```

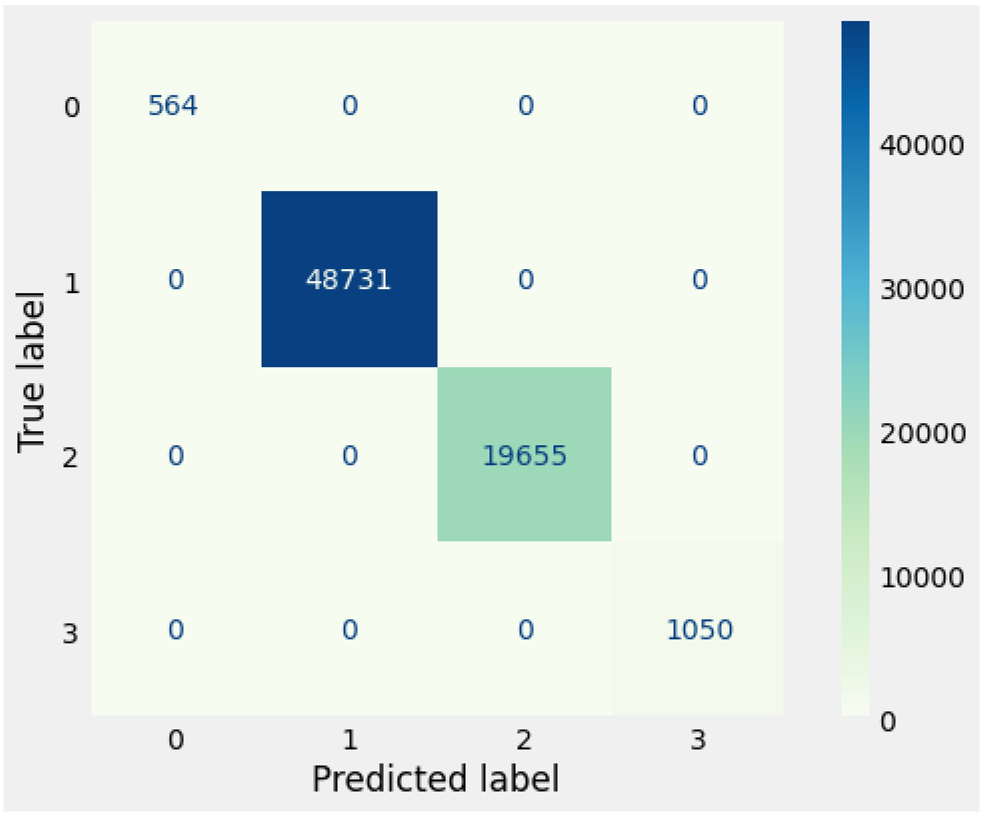
Decision Tree

42
```
desicion_tree_classifier = classify(DecisionTreeClassifier)
```

```
Test Result:

Accuracy score: 0.7793

Error rate: 0.2207

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.40      0.44      0.42       240
           1       0.85      0.84      0.85     20821
           2       0.65      0.67      0.66      8494
           3       0.23      0.26      0.25       445

    accuracy                           0.78     30000
   macro avg       0.53      0.55      0.54     30000
weighted avg       0.78      0.78      0.78     30000


Confusion Matrix:
 [[  106   103    24     7]
 [  126 17480  2947   268]
 [   31  2677  5676   110]
 [    3   212   113   117]]
```



```
ROC AUC: 0.7156

Average Accuracy:        0.7553
Accuracy SD:             0.0057
================================================================================
Train Result:

Accuracy score: 1.0000

Error rate: 0.0000

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       564
           1       1.00      1.00      1.00     48731
           2       1.00      1.00      1.00     19655
           3       1.00      1.00      1.00      1050

    accuracy                           1.00     70000
   macro avg       1.00      1.00      1.00     70000
weighted avg       1.00      1.00      1.00     70000


Confusion Matrix:
 [[  564     0     0     0]
 [    0 48731     0     0]
 [    0     0 19655     0]
 [    0     0     0  1050]]
```

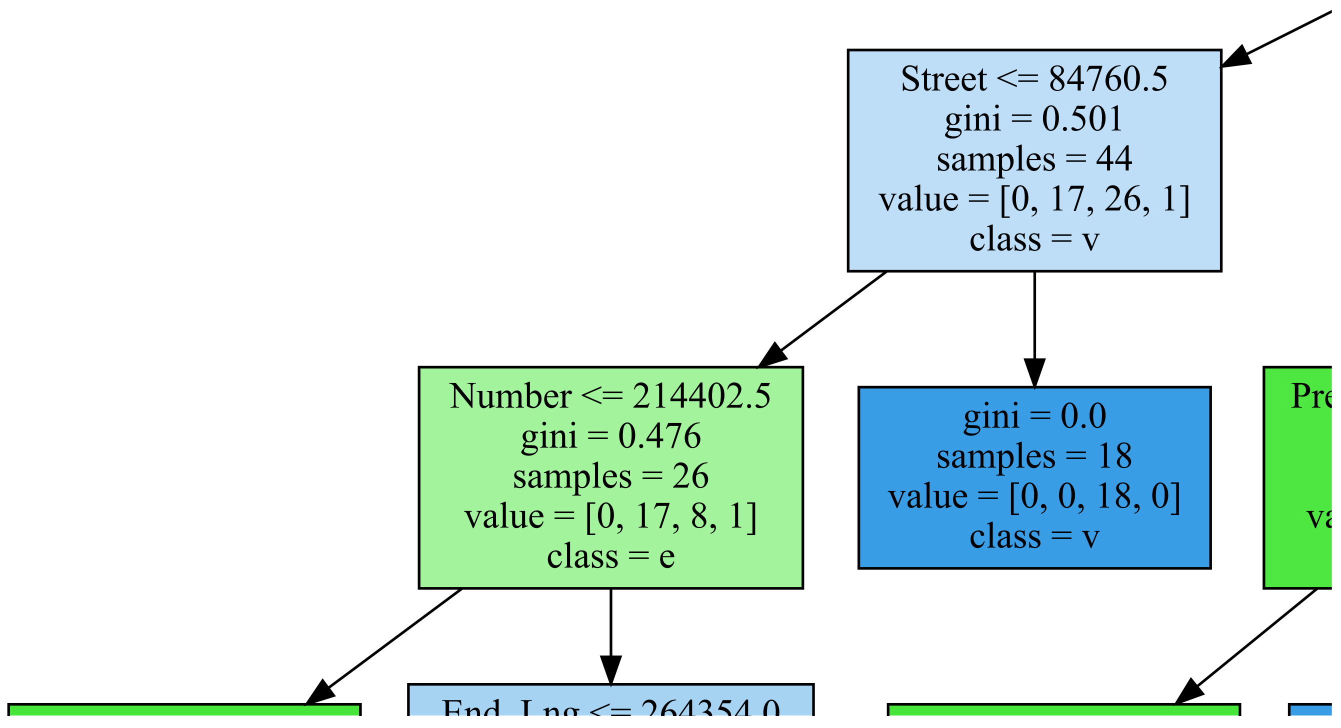


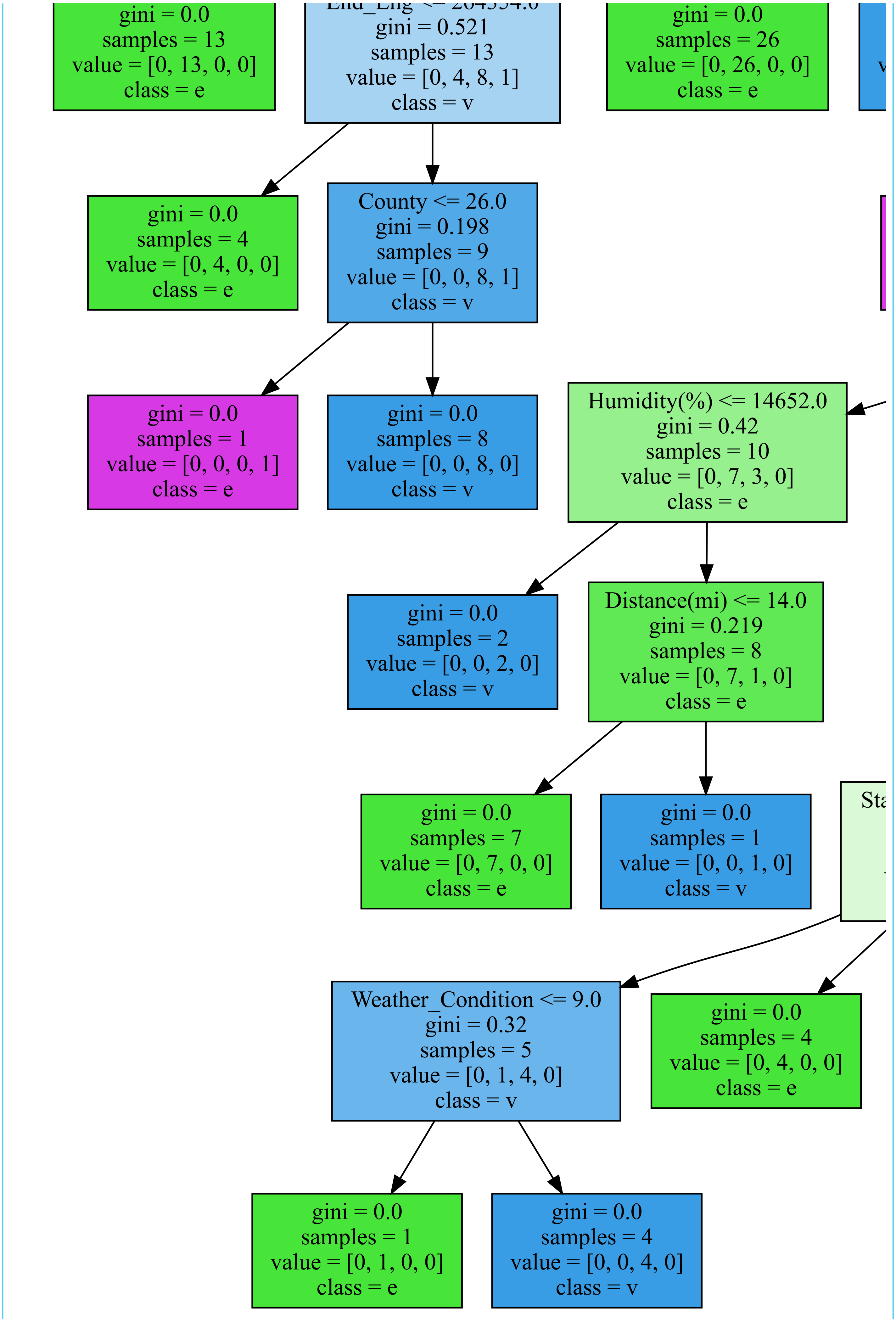
```
ROC AUC: 1.0000

Average Accuracy:        0.7799
Accuracy SD:             0.0031
```

43

```
dot_data = tree.export_graphviz(desicion_tree_classifier, out_file=None,
                                feature_names=X.columns,
                                class_names=y.name,
                                filled=True)

# Draw graph
graph = graphviz.Source(dot_data, format="png")
graph
```

43

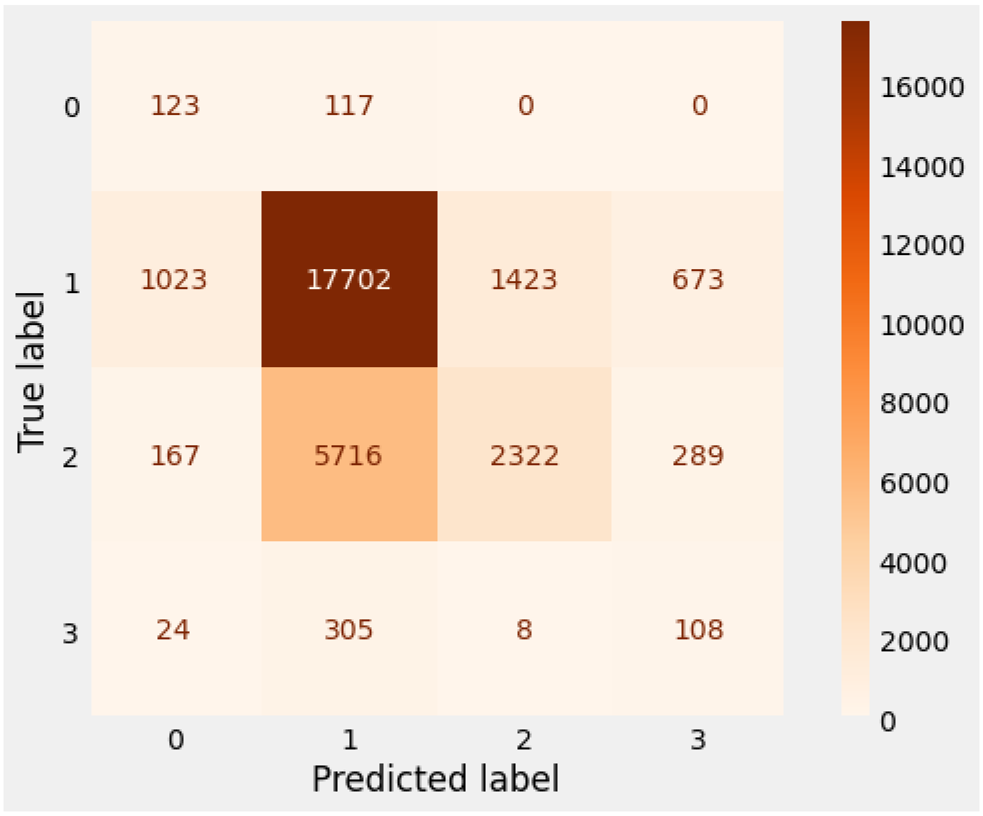

Street <= 84760.5
gini = 0.501
samples = 44
value = [0, 17, 26, 1]
class = v
Number <= 214402.5
gini = 0.476
samples = 26
value = [0, 17, 8, 1]
class = e
gini = 0.0
samples = 18
value = [0, 0, 18, 0]
class = v
End_Lng <= 264354.0


gini = 0.0
samples = 13
value = [0, 13, 0, 0]
class = e
gini = 0.521
samples = 13
value = [0, 4, 8, 1]
class = v
gini = 0.0
samples = 26
value = [0, 26, 0, 0]
class = e
gini = 0.0
samples = 4
value = [0, 4, 0, 0]
class = e
County <= 26.0
gini = 0.198
samples = 9
value = [0, 0, 8, 1]
class = v
gini = 0.0
samples = 1
value = [0, 0, 0, 1]
class = e
gini = 0.0
samples = 8
value = [0, 0, 8, 0]
class = v
Humidity(%) <= 14652.0
gini = 0.42
samples = 10
value = [0, 7, 3, 0]
class = e
gini = 0.0
samples = 2
value = [0, 0, 2, 0]
class = v
Distance(mi) <= 14.0
gini = 0.219
samples = 8
value = [0, 7, 1, 0]
class = e
gini = 0.0
samples = 7
value = [0, 7, 0, 0]
class = e
gini = 0.0
samples = 1
value = [0, 0, 1, 0]
class = v
Weather_Condition <= 9.0
gini = 0.32
samples = 5
value = [0, 1, 4, 0]
class = v
gini = 0.0
samples = 4
value = [0, 4, 0, 0]
class = e
gini = 0.0
samples = 1
value = [0, 1, 0, 0]
class = e
gini = 0.0
samples = 4
value = [0, 0, 4, 0]
class = v

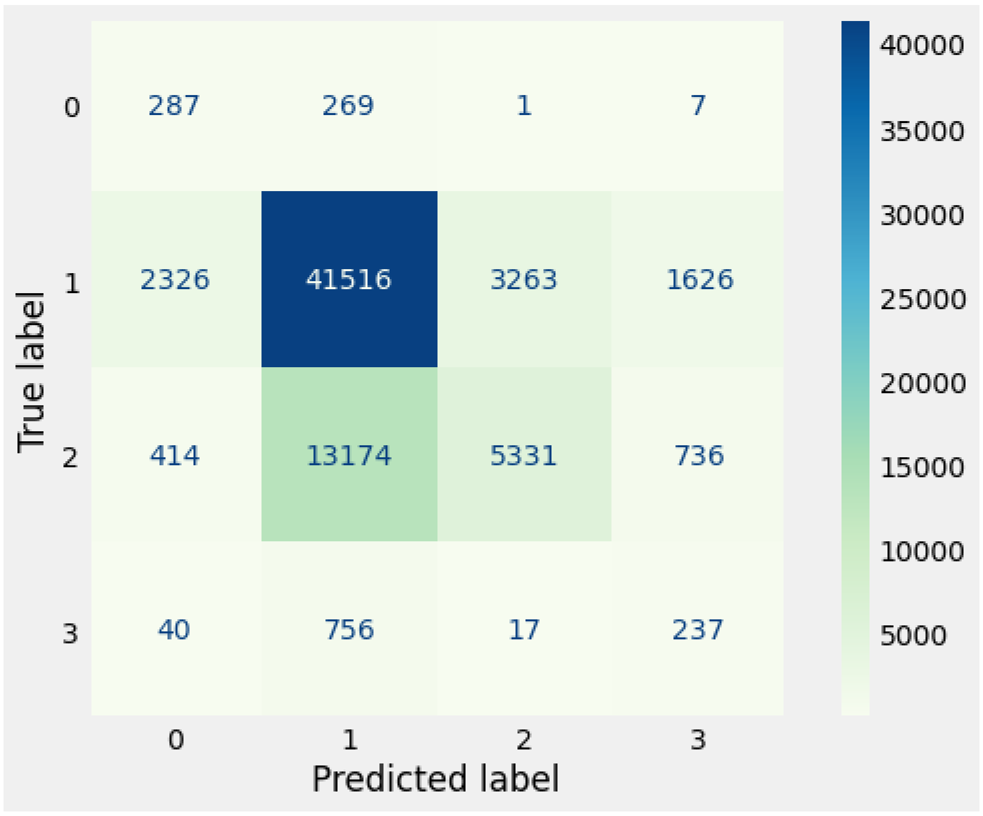
Naive Bayes

44
```
gaussian_naive_bayes_classifier = classify(GaussianNB)
```

```
Test Result:

Accuracy score: 0.6752

Error rate: 0.3248

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.09      0.51      0.16       240
           1       0.74      0.85      0.79     20821
           2       0.62      0.27      0.38      8494
           3       0.10      0.24      0.14       445

    accuracy                           0.68     30000
   macro avg       0.39      0.47      0.37     30000
weighted avg       0.69      0.68      0.66     30000


Confusion Matrix:
 [[  123   117     0     0]
 [ 1023 17702  1423   673]
 [  167  5716  2322   289]
 [   24   305     8   108]]
```



```
ROC AUC: 0.6338

Average Accuracy:        0.6723
Accuracy SD:             0.0072
================================================================================
Train Result:

Accuracy score: 0.6767

Error rate: 0.3233

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.09      0.51      0.16       564
           1       0.75      0.85      0.79     48731
           2       0.62      0.27      0.38     19655
           3       0.09      0.23      0.13      1050

    accuracy                           0.68     70000
   macro avg       0.39      0.46      0.36     70000
weighted avg       0.69      0.68      0.66     70000


Confusion Matrix:
 [[  287   269     1     7]
 [ 2326 41516  3263  1626]
 [  414 13174  5331   736]
 [   40   756    17   237]]
```

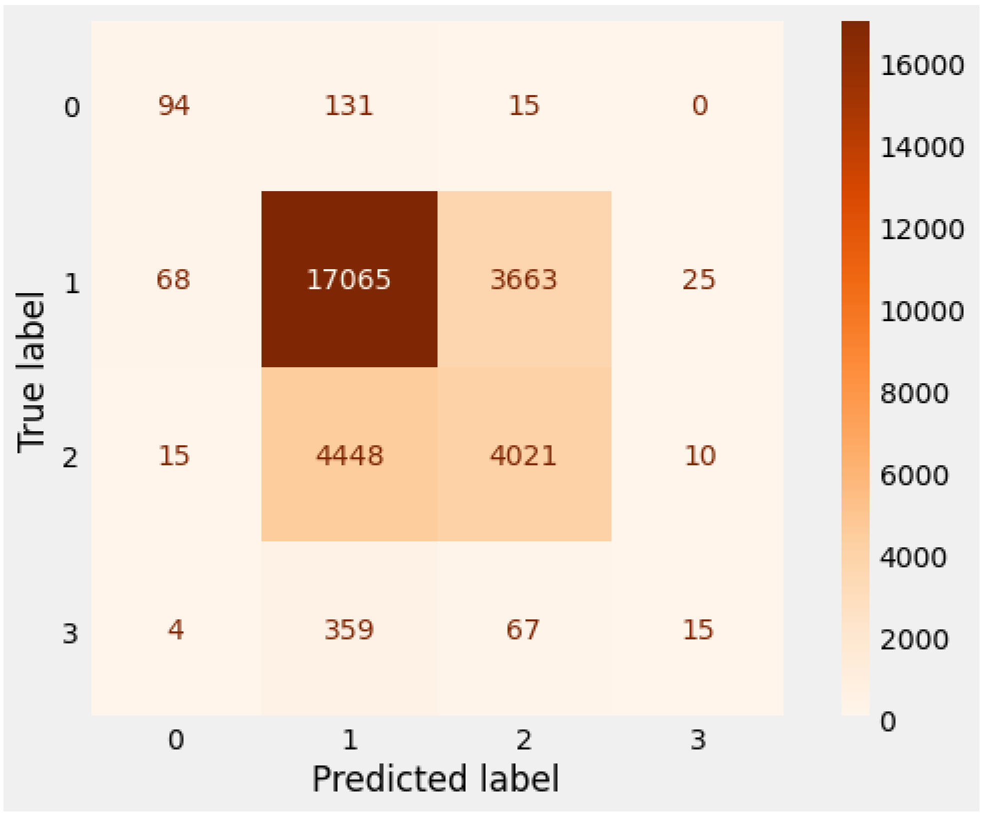


```
ROC AUC: 0.6313

Average Accuracy:        0.6764
Accuracy SD:             0.0049
```

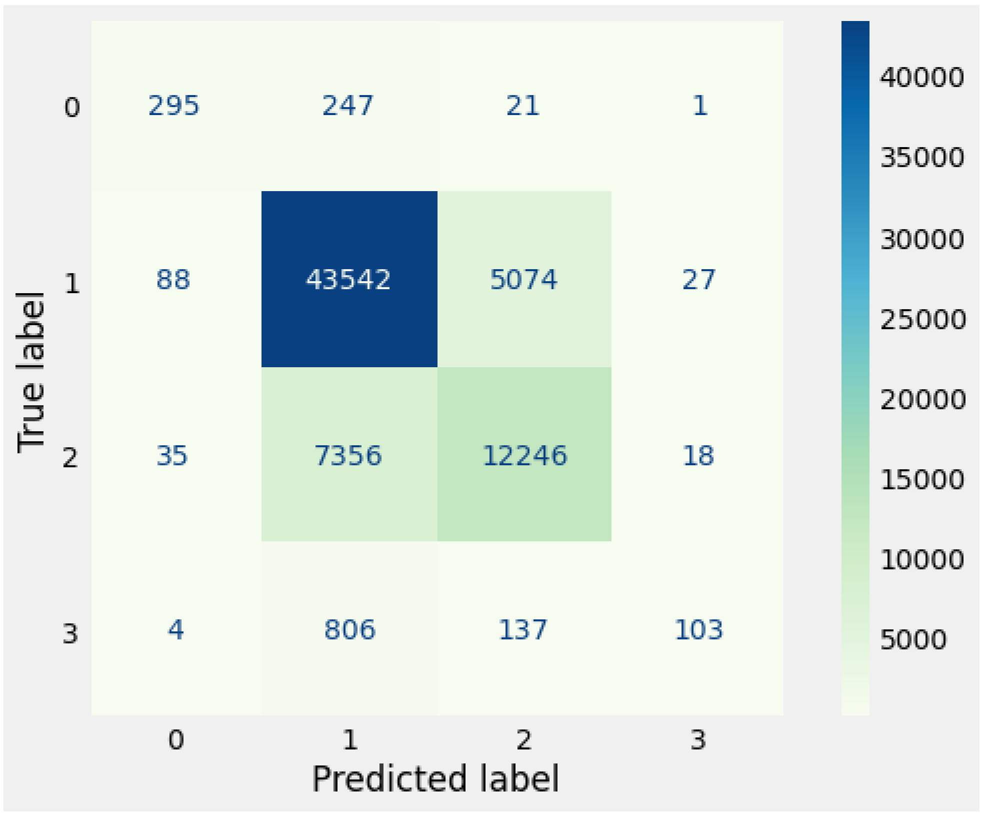
K Nearest Neighbors

45
```
knn_classifier = classify(KNeighborsClassifier)
```

```
Test Result:

Accuracy score: 0.7065

Error rate: 0.2935

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.52      0.39      0.45       240
           1       0.78      0.82      0.80     20821
           2       0.52      0.47      0.49      8494
           3       0.30      0.03      0.06       445

    accuracy                           0.71     30000
   macro avg       0.53      0.43      0.45     30000
weighted avg       0.69      0.71      0.70     30000


Confusion Matrix:
 [[   94   131    15     0]
 [   68 17065  3663    25]
 [   15  4448  4021    10]
 [    4   359    67    15]]
```



```
ROC AUC: 0.6253

Average Accuracy:        0.6993
Accuracy SD:             0.0044
================================================================================
Train Result:

Accuracy score: 0.8027

Error rate: 0.1973

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.70      0.52      0.60       564
           1       0.84      0.89      0.86     48731
           2       0.70      0.62      0.66     19655
           3       0.69      0.10      0.17      1050

    accuracy                           0.80     70000
   macro avg       0.73      0.53      0.57     70000
weighted avg       0.80      0.80      0.79     70000


Confusion Matrix:
 [[  295   247    21     1]
 [   88 43542  5074    27]
 [   35  7356 12246    18]
 [    4   806   137   103]]
```



```
ROC AUC: 0.7045

Average Accuracy:        0.7085
Accuracy SD:             0.0037
```

Classification By Ensemble Methods

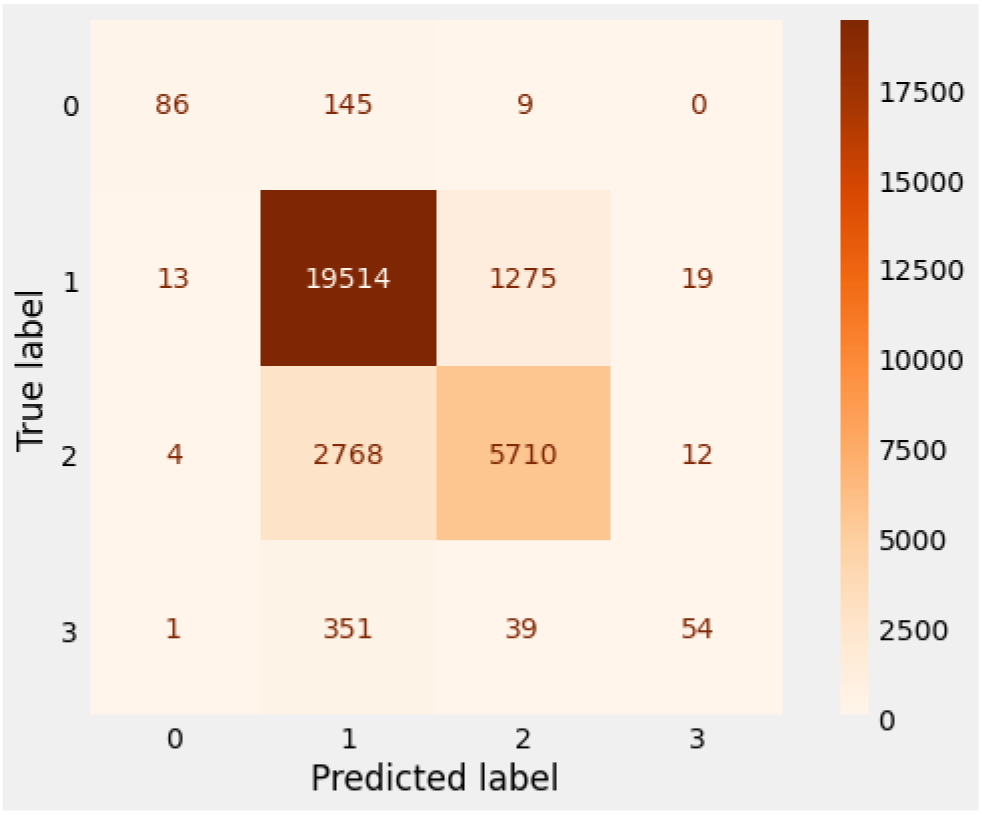
46
```
def ensemble_classify(clf, base = None):

  sig = inspect.signature(clf.__init__)

  try:
    sig.parameters['n_jobs']
    classifier = clf(base_estimator = base, n_jobs = -1)
  except Exception as e:
    classifier = clf(base_estimator = base)

  classifier.fit(X_train, y_train)

  print_performance(classifier, X_train, X_test, y_train, y_test, train=False)
  print("=" * 40)
  print_performance(classifier, X_train, X_test, y_train, y_test, train=True)

  return classifier
```

Random Forest

47
```
forest_classifier = classify(RandomForestClassifier)
```

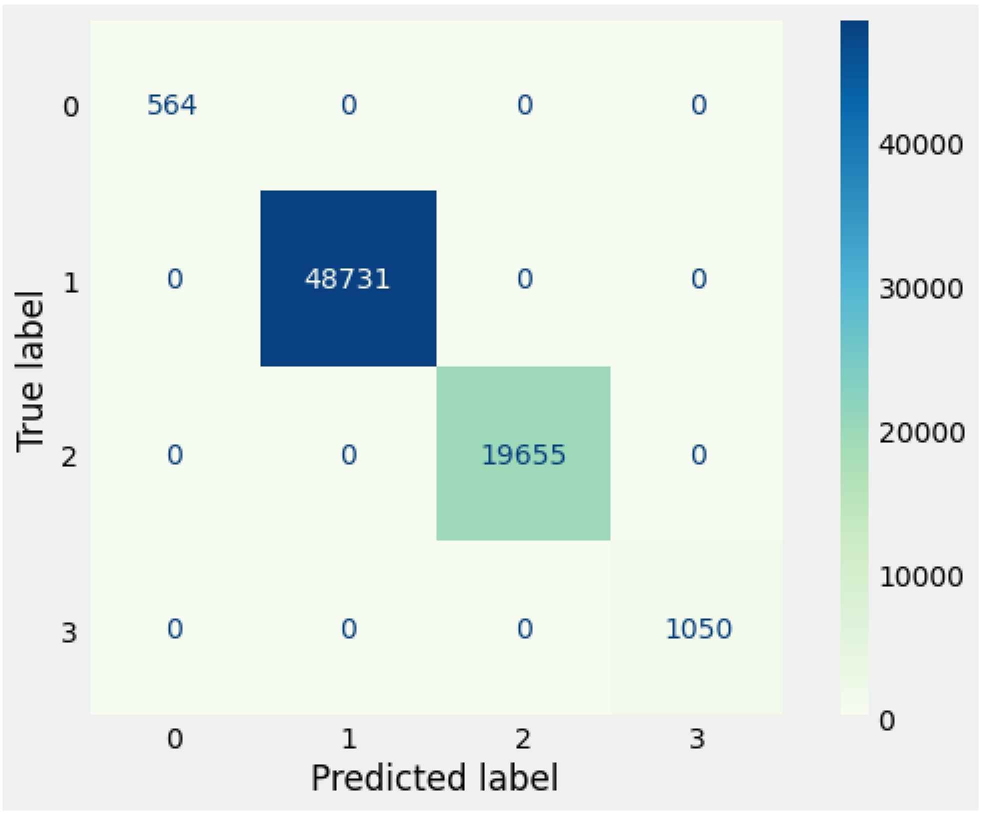
```
Test Result:

Accuracy score: 0.8455

Error rate: 0.1545

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.36      0.50       240
           1       0.86      0.94      0.90     20821
           2       0.81      0.67      0.74      8494
           3       0.64      0.12      0.20       445

    accuracy                           0.85     30000
   macro avg       0.78      0.52      0.58     30000
weighted avg       0.84      0.85      0.84     30000


Confusion Matrix:
 [[   86   145     9     0]
 [   13 19514  1275    19]
 [    4  2768  5710    12]
 [    1   351    39    54]]
```



```
ROC AUC: 0.7088

Average Accuracy:        0.8294
Accuracy SD:             0.0038
================================================================================
Train Result:

Accuracy score: 1.0000

Error rate: 0.0000

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       564
           1       1.00      1.00      1.00     48731
           2       1.00      1.00      1.00     19655
           3       1.00      1.00      1.00      1050

    accuracy                           1.00     70000
```

```
   macro avg       1.00      1.00      1.00     70000
weighted avg       1.00      1.00      1.00     70000


Confusion Matrix:
 [[  564     0     0     0]
 [    0 48731     0     0]
 [    0     0 19655     0]
 [    0     0     0  1050]]
```



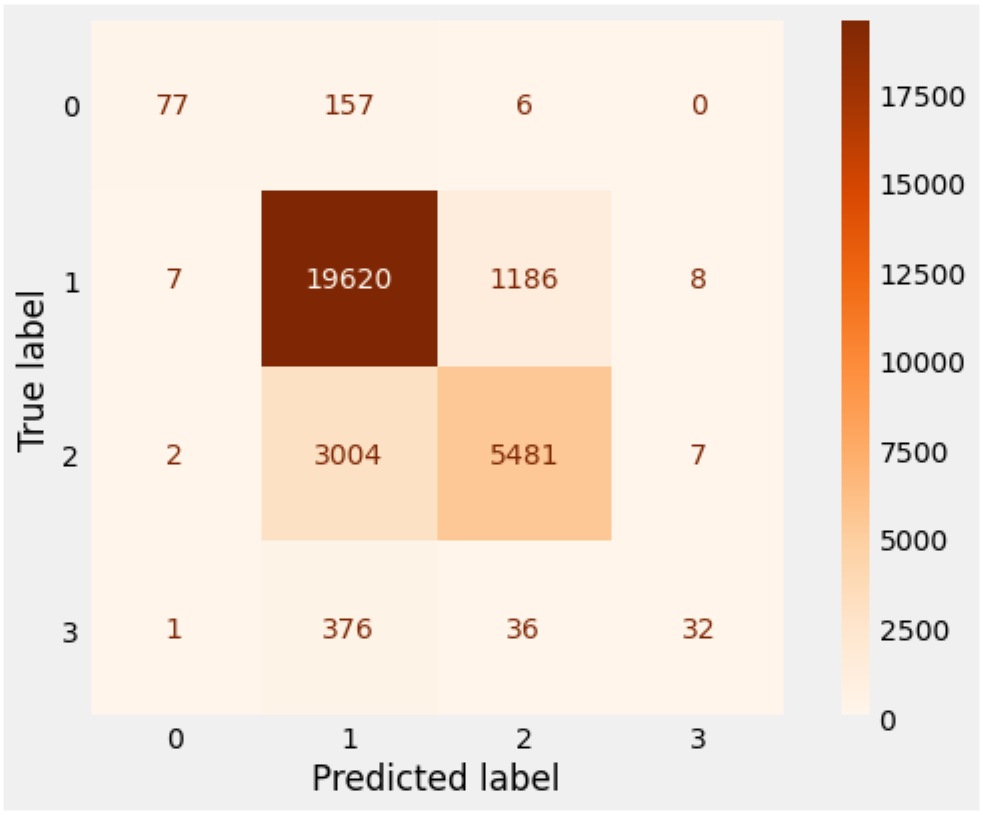
```
ROC AUC: 1.0000

Average Accuracy:        0.8428
Accuracy SD:             0.0021
```

Bagging + Random Forest

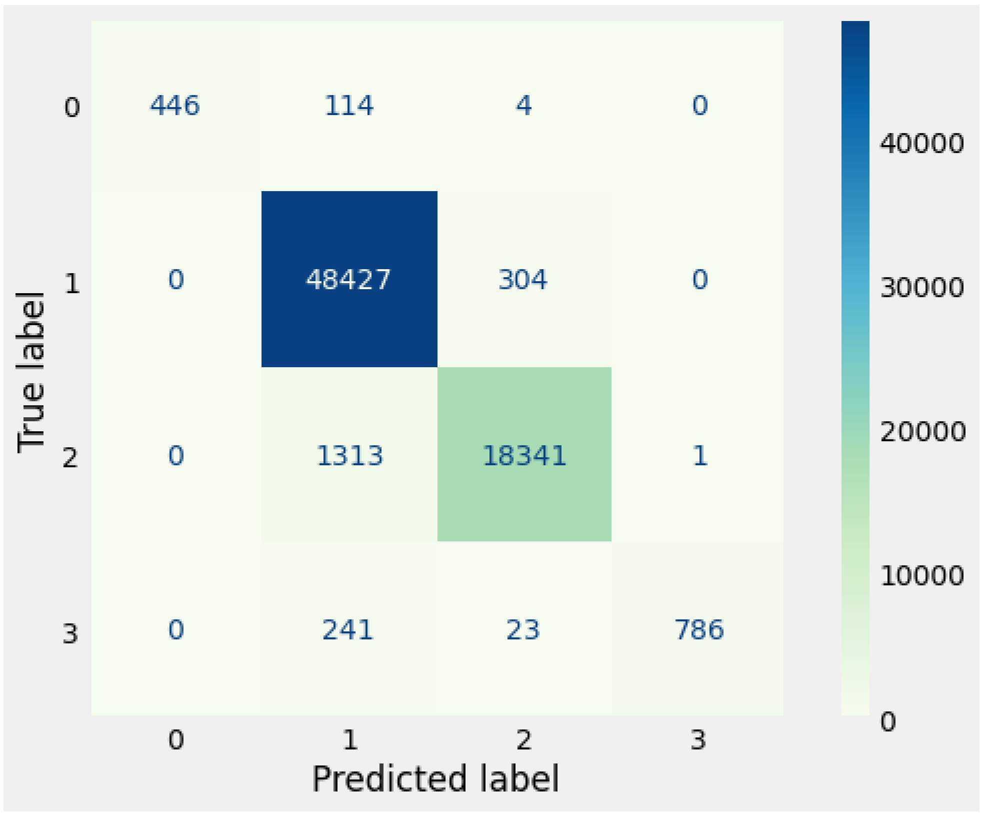
48
```
bag_classifier = ensemble_classify(BaggingClassifier, base = forest_classifier)
```

```
Test Result:

Accuracy score: 0.8403

Error rate: 0.1597

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.32      0.47       240
           1       0.85      0.94      0.89     20821
           2       0.82      0.65      0.72      8494
           3       0.68      0.07      0.13       445

    accuracy                           0.84     30000
   macro avg       0.81      0.50      0.55     30000
weighted avg       0.84      0.84      0.83     30000
```

```
Confusion Matrix:
 [[   77   157     6     0]
 [    7 19620  1186     8]
 [    2  3004  5481     7]
 [    1   376    36    32]]
```



```
ROC AUC: 0.6921

Average Accuracy:        0.8251
Accuracy SD:             0.0045
========================================
Train Result:

Accuracy score: 0.9714

Error rate: 0.0286

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.79      0.88       564
           1       0.97      0.99      0.98     48731
           2       0.98      0.93      0.96     19655
           3       1.00      0.75      0.86      1050

    accuracy                           0.97     70000
   macro avg       0.99      0.87      0.92     70000
weighted avg       0.97      0.97      0.97     70000


Confusion Matrix:
 [[  446   114     4     0]
 [    0 48427   304     0]
 [    0  1313 18341     1]
 [    0   241    23   786]]
```

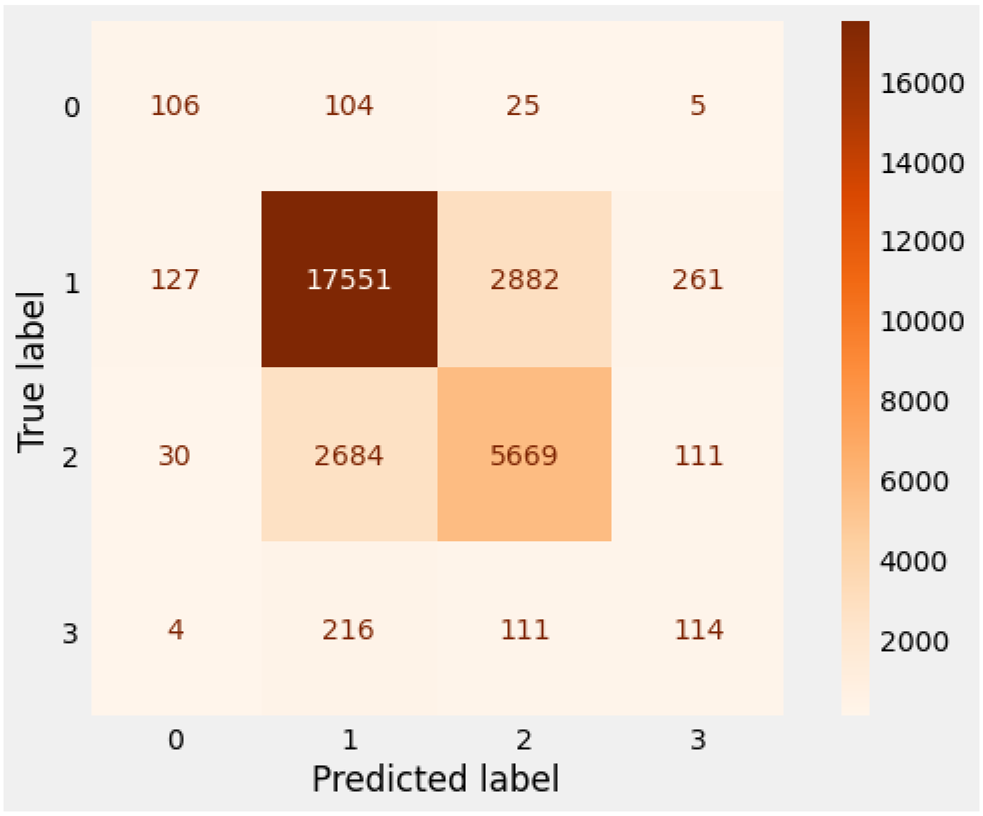


```
ROC AUC: 0.9227

Average Accuracy:        0.8396
Accuracy SD:             0.0022
```

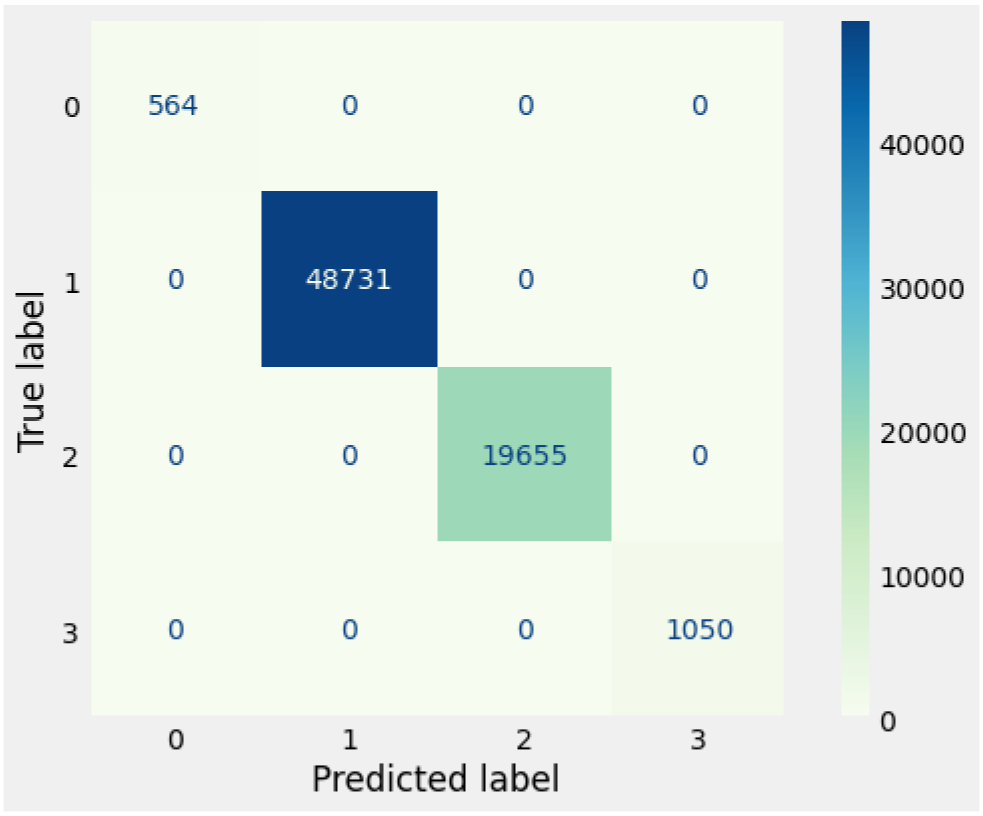
Ada Boost + Decision Tree

49
```
ada_classifier = ensemble_classify(AdaBoostClassifier, base = desicion_tree_classifier)
```

```
Test Result:

Accuracy score: 0.7813

Error rate: 0.2187

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.40      0.44      0.42       240
           1       0.85      0.84      0.85     20821
           2       0.65      0.67      0.66      8494
           3       0.23      0.26      0.24       445

    accuracy                           0.78     30000
   macro avg       0.53      0.55      0.54     30000
weighted avg       0.78      0.78      0.78     30000


Confusion Matrix:
 [[  106   104    25     5]
 [  127 17551  2882   261]
 [   30  2684  5669   111]
 [    4   216   111   114]]
```



```
ROC AUC: 0.7153

Average Accuracy:        0.7563
Accuracy SD:             0.0054
========================================
Train Result:

Accuracy score: 1.0000

Error rate: 0.0000

recall of the positive class is also known as sensitivity ; recall of the negative class is specificit
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       564
           1       1.00      1.00      1.00     48731
           2       1.00      1.00      1.00     19655
           3       1.00      1.00      1.00      1050

    accuracy                           1.00     70000
   macro avg       1.00      1.00      1.00     70000
weighted avg       1.00      1.00      1.00     70000


Confusion Matrix:
 [[  564     0     0     0]
 [    0 48731     0     0]
 [    0     0 19655     0]
 [    0     0     0  1050]]
```

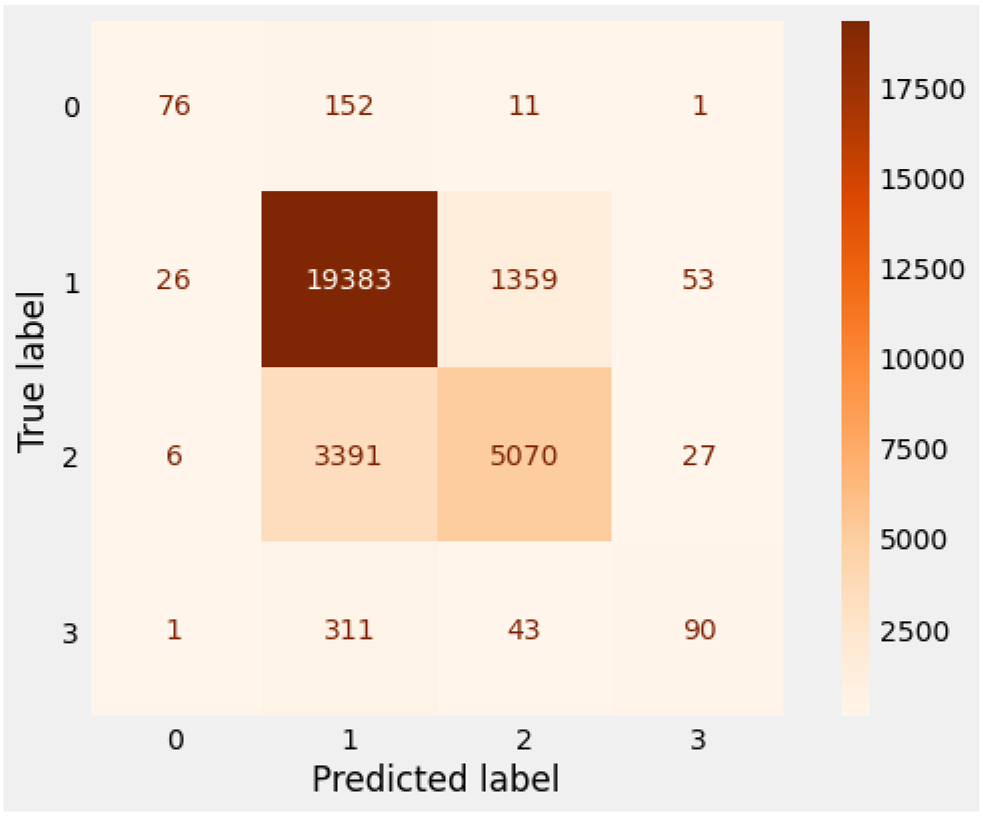


```
ROC AUC: 1.0000

Average Accuracy:         0.7791
Accuracy SD:              0.0040
```

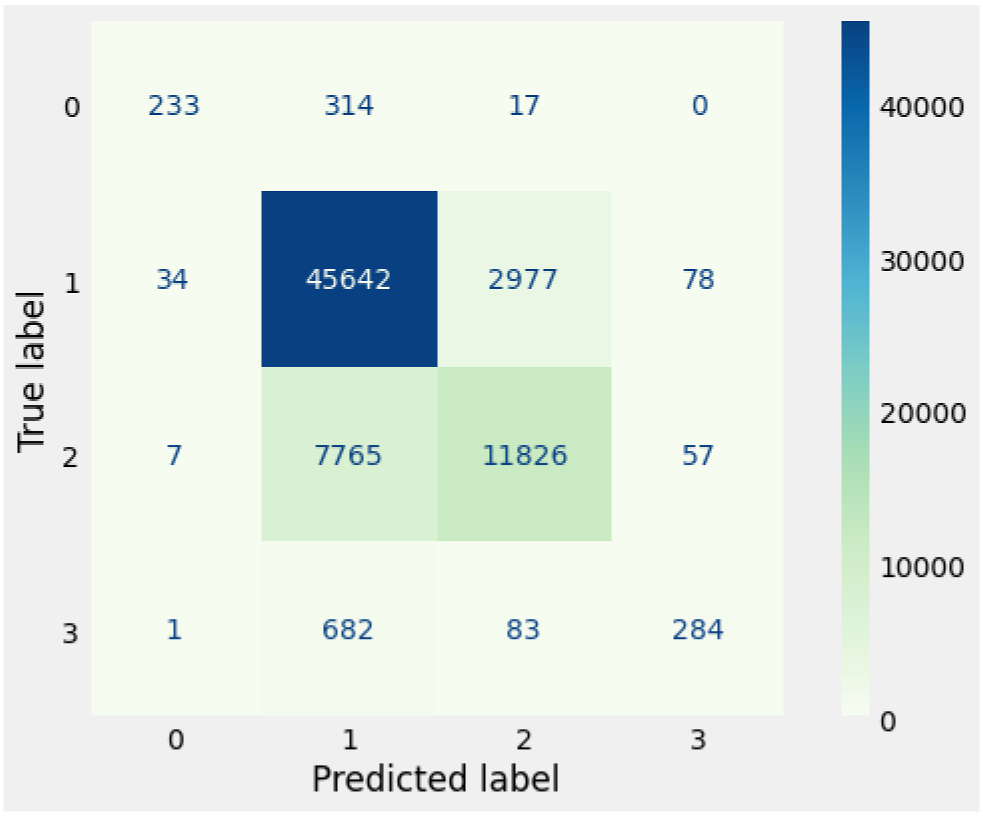
Gradient Boosting

50
```
gradient_classifier = classify(GradientBoostingClassifier)
```

```
Test Result:

Accuracy score: 0.8206

Error rate: 0.1794

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.70      0.32      0.44       240
           1       0.83      0.93      0.88     20821
           2       0.78      0.60      0.68      8494
           3       0.53      0.20      0.29       445

    accuracy                           0.82     30000
   macro avg       0.71      0.51      0.57     30000
weighted avg       0.81      0.82      0.81     30000


Confusion Matrix:
 [[   76   152    11     1]
 [   26 19383  1359    53]
 [    6  3391  5070    27]
 [    1   311    43    90]]
```



```
ROC AUC: 0.6947

Average Accuracy:        0.8169
Accuracy SD:             0.0073
================================================================================
Train Result:

Accuracy score: 0.8284

Error rate: 0.1716

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.41      0.56       564
           1       0.84      0.94      0.89     48731
           2       0.79      0.60      0.68     19655
           3       0.68      0.27      0.39      1050

    accuracy                           0.83     70000
   macro avg       0.79      0.56      0.63     70000
weighted avg       0.82      0.83      0.82     70000


Confusion Matrix:
 [[  233   314    17     0]
 [   34 45642  2977    78]
 [    7  7765 11826    57]
 [    1   682    83   284]]
```

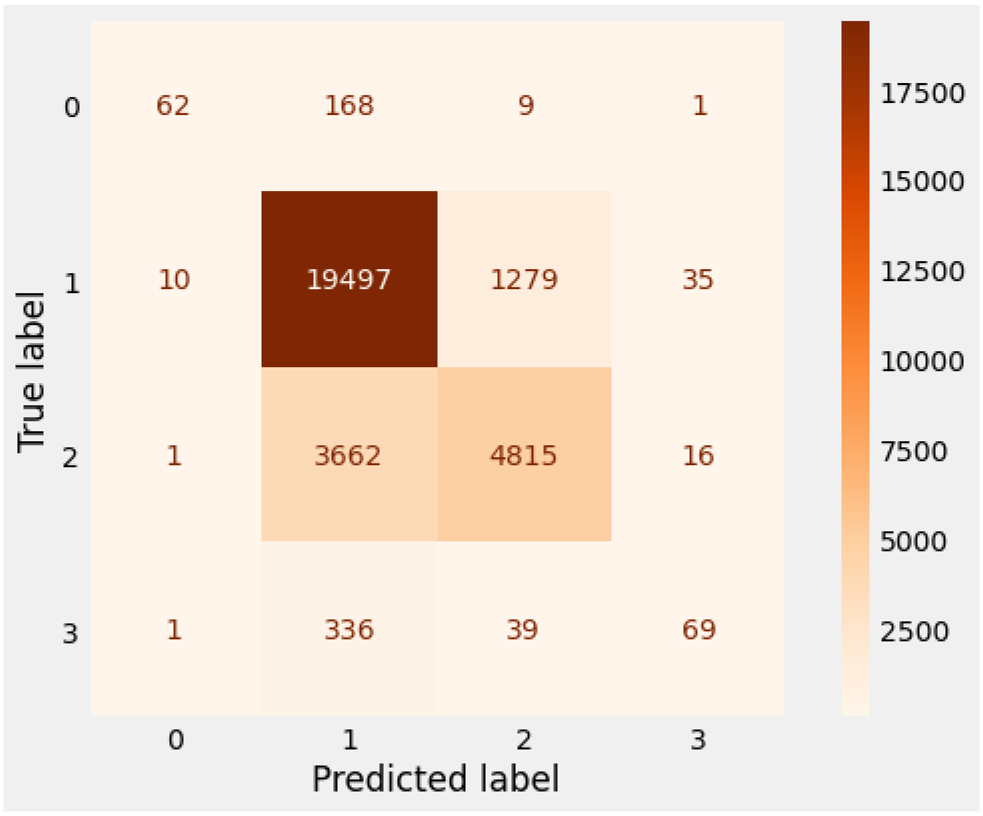


```
ROC AUC: 0.7183

Average Accuracy:         0.8228
Accuracy SD:              0.0023
```

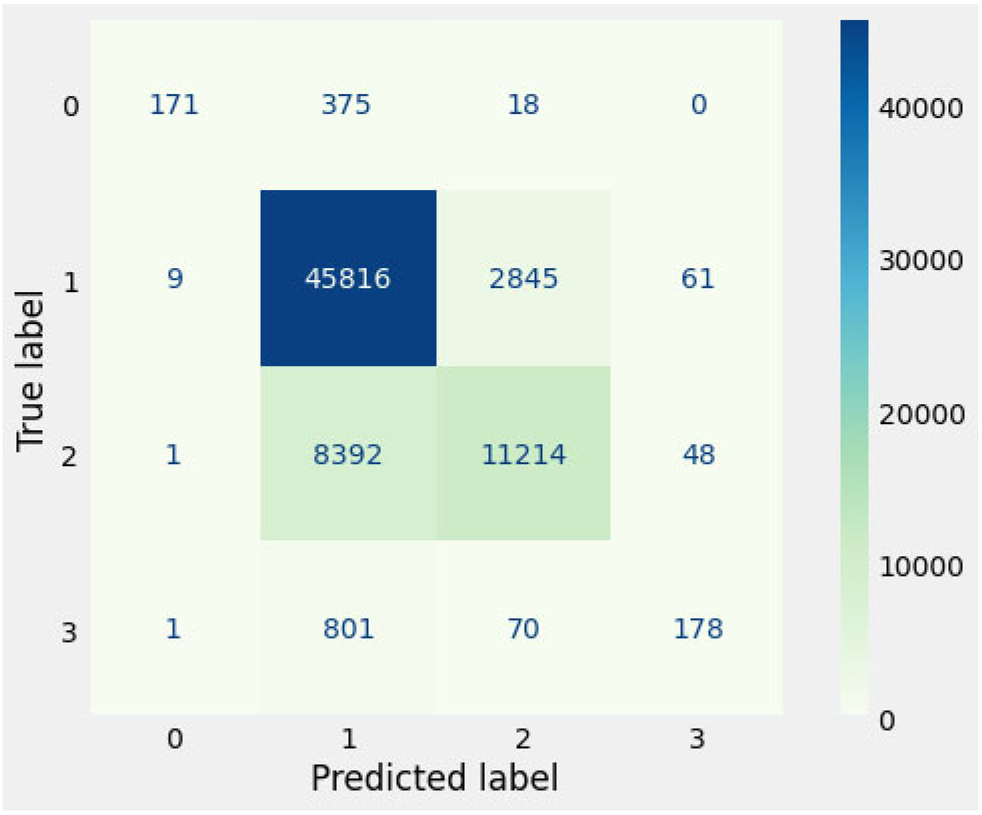
XG Boosting

51
```
xgb_classifier = classify(xgb.XGBClassifier)
```

```
Test Result:

Accuracy score: 0.8148

Error rate: 0.1852

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.84      0.26      0.39       240
           1       0.82      0.94      0.88     20821
           2       0.78      0.57      0.66      8494
           3       0.57      0.16      0.24       445

    accuracy                           0.81     30000
   macro avg       0.75      0.48      0.54     30000
weighted avg       0.81      0.81      0.80     30000


Confusion Matrix:
 [[   62   168     9     1]
 [   10 19497  1279    35]
 [    1  3662  4815    16]
 [    1   336    39    69]]
```



```
ROC AUC: 0.6749

Average Accuracy: 	 0.8126
Accuracy SD: 		 0.0072
================================================================================
Train Result:

Accuracy score: 0.8197

Error rate: 0.1803

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.30      0.46       564
           1       0.83      0.94      0.88     48731
           2       0.79      0.57      0.66     19655
           3       0.62      0.17      0.27      1050

    accuracy                           0.82     70000
   macro avg       0.79      0.50      0.57     70000
weighted avg       0.82      0.82      0.81     70000


Confusion Matrix:
 [[  171   375    18     0]
 [    9 45816  2845    61]
 [    1  8392 11214    48]
 [    1   801    70   178]]
```

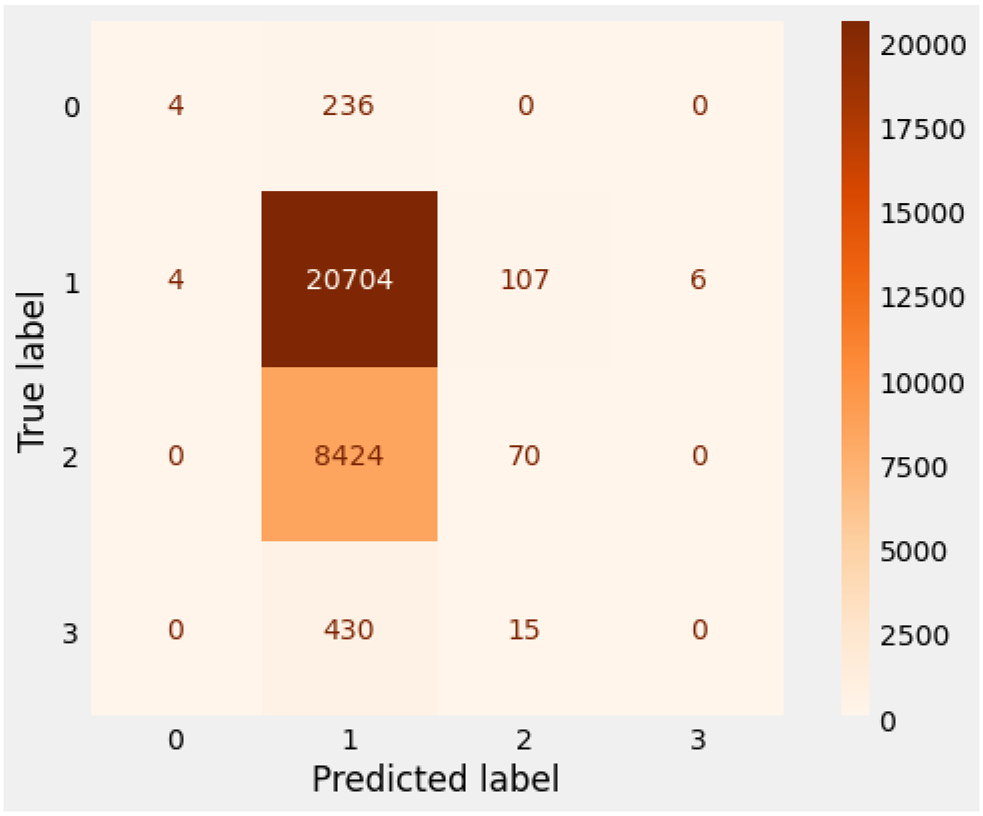


```
ROC AUC: 0.6842

Average Accuracy: 	 0.8168
Accuracy SD: 		 0.0020
```

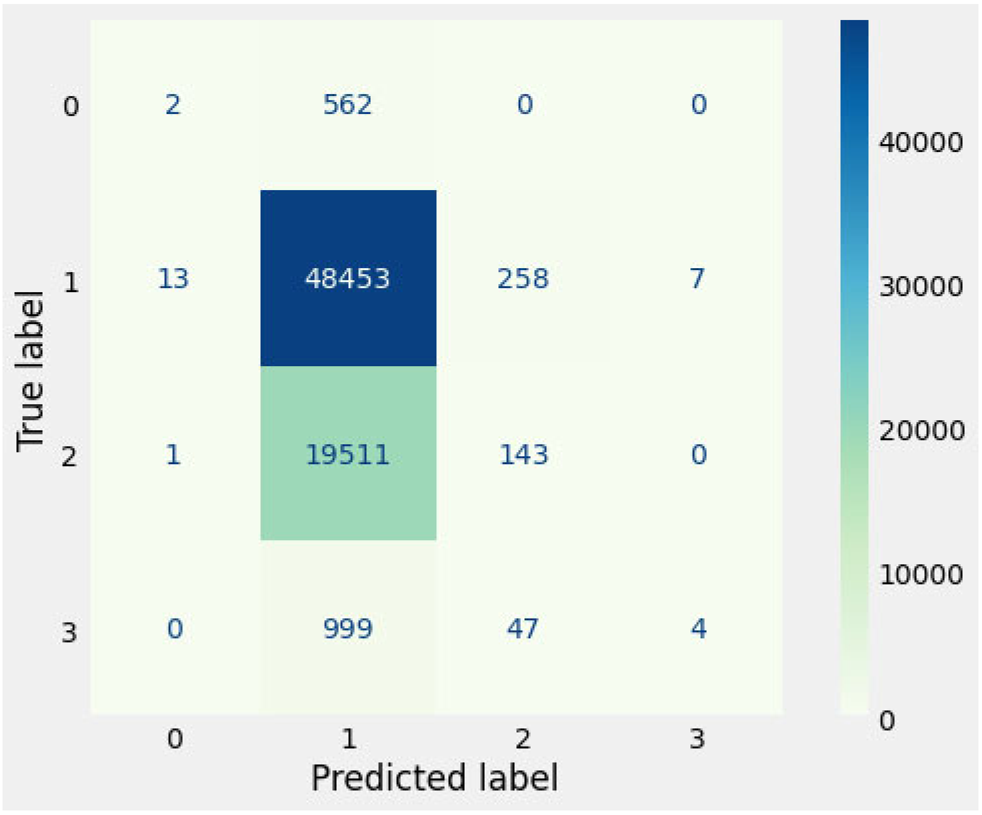
Logistic Classification

52
```
log_classifier = LogisticRegression(max_iter=100000, n_jobs = -1)

log_classifier.fit(X_train, y_train)

print_performance(log_classifier, X_train, X_test, y_train, y_test, train=False)
print("=" * 40)
print_performance(log_classifier, X_train, X_test, y_train, y_test, train=True)
```

```
Test Result:

Accuracy score: 0.6926

Error rate: 0.3074

recall of the positive class is also known as sensitivity ; recall of the negative class is specificit
Classification Report:
               precision    recall  f1-score   support

           0       0.50      0.02      0.03       240
           1       0.69      0.99      0.82     20821
           2       0.36      0.01      0.02      8494
           3       0.00      0.00      0.00       445

    accuracy                           0.69     30000
   macro avg       0.39      0.25      0.22     30000
weighted avg       0.59      0.69      0.57     30000


Confusion Matrix:
 [[    4   236     0     0]
 [    4 20704   107     6]
 [    0  8424    70     0]
 [    0   430    15      0]]
```



```
ROC AUC: 0.5029

Average Accuracy:        0.6928
Accuracy SD:             0.0015
========================================
Train Result:

Accuracy score: 0.6943

Error rate: 0.3057

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       0.12      0.00      0.01       564
           1       0.70      0.99      0.82     48731
           2       0.32      0.01      0.01     19655
           3       0.36      0.00      0.01      1050

    accuracy                           0.69     70000
   macro avg       0.38      0.25      0.21     70000
weighted avg       0.58      0.69      0.57     70000


Confusion Matrix:
 [[    2   562     0     0]
 [   13 48453   258     7]
 [    1 19511   143     0]
 [    0   999    47     4]]
```



```
ROC AUC: 0.5015

Average Accuracy:        0.6946
Accuracy SD:             0.0009
```

Classification By Stacking Methods

Stacking With Random Forest, Desicion Tree and Logistic Classification

53
```
sclf = StackingClassifier(classifiers=[xgb_classifier, gradient_classifier, ada_classifier], meta_cla

sclf.fit(X_train, y_train)

print_performance(sclf, X_train, X_test, y_train, y_test, train=False)
print("=" * 40)
print_performance(sclf, X_train, X_test, y_train, y_test, train=True)
```

```
Test Result:

Accuracy score: 0.7790

Error rate: 0.2210

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
              precision    recall  f1-score   support

           0       0.40      0.43      0.41       240
           1       0.85      0.84      0.85     20821
           2       0.65      0.67      0.66      8494
           3       0.23      0.25      0.24       445

    accuracy                           0.78     30000
   macro avg       0.53      0.55      0.54     30000
weighted avg       0.78      0.78      0.78     30000
```

```
Confusion Matrix:
 [[  104   106    25     5]
 [  127 17499  2943   252]
 [   30  2695  5656   113]
 [    2   217   116   110]]

ROC AUC: 0.7121

Average Accuracy: 	 0.7560
Accuracy SD: 		 0.0051
========================================
Train Result:

Accuracy score: 1.0000

Error rate: 0.0000

recall of the positive class is also known as sensitivity ; recall of the negative class is specifici
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       564
           1       1.00      1.00      1.00     48731
           2       1.00      1.00      1.00     19655
           3       1.00      1.00      1.00      1050

    accuracy                           1.00     70000
   macro avg       1.00      1.00      1.00     70000
weighted avg       1.00      1.00      1.00     70000


Confusion Matrix:
 [[  564     0     0     0]
 [    0 48731     0     0]
 [    0     0 19655     0]
 [    0     0     0  1050]]

ROC AUC: 1.0000

Average Accuracy: 	 0.7796
Accuracy SD: 		 0.0047
```

Clustering

Checking for cluster tendency with Hopkins Score

A score between 0 and 1, a score around 0.5 express no clusterability and a score tending to 0 express a high cluster tendency.

54
```
hopkins(X,  X.shape[0])
```

54
```
0.22001508729373495
```

Defining function to find optimal number of clusters

69
```
mms = MinMaxScaler()
mms.fit(X)
data_transformed = mms.transform(X)

def find_optimal_clusters(model):
  Sum_of_squared_distances = []
  K = range(1,15)

  for k in K:
      km = model(n_clusters=k, n_jobs = -1)
      km = km.fit(data_transformed)
      Sum_of_squared_distances.append(km.inertia_)

  plt.plot(K, Sum_of_squared_distances, 'bx-')
  plt.xlabel('k')
  plt.ylabel('Sum of squared distances')
  plt.title('Elbow Method For Optimal k')

  kn = KneeLocator(range(1,15), Sum_of_squared_distances, curve='convex', direction='decreasing')
  plt.vlines(kn.knee, plt.ylim()[0], plt.ylim()[1], linestyles='dashed')

  plt.show()

  return kn.knee
```

Defining function to cluster

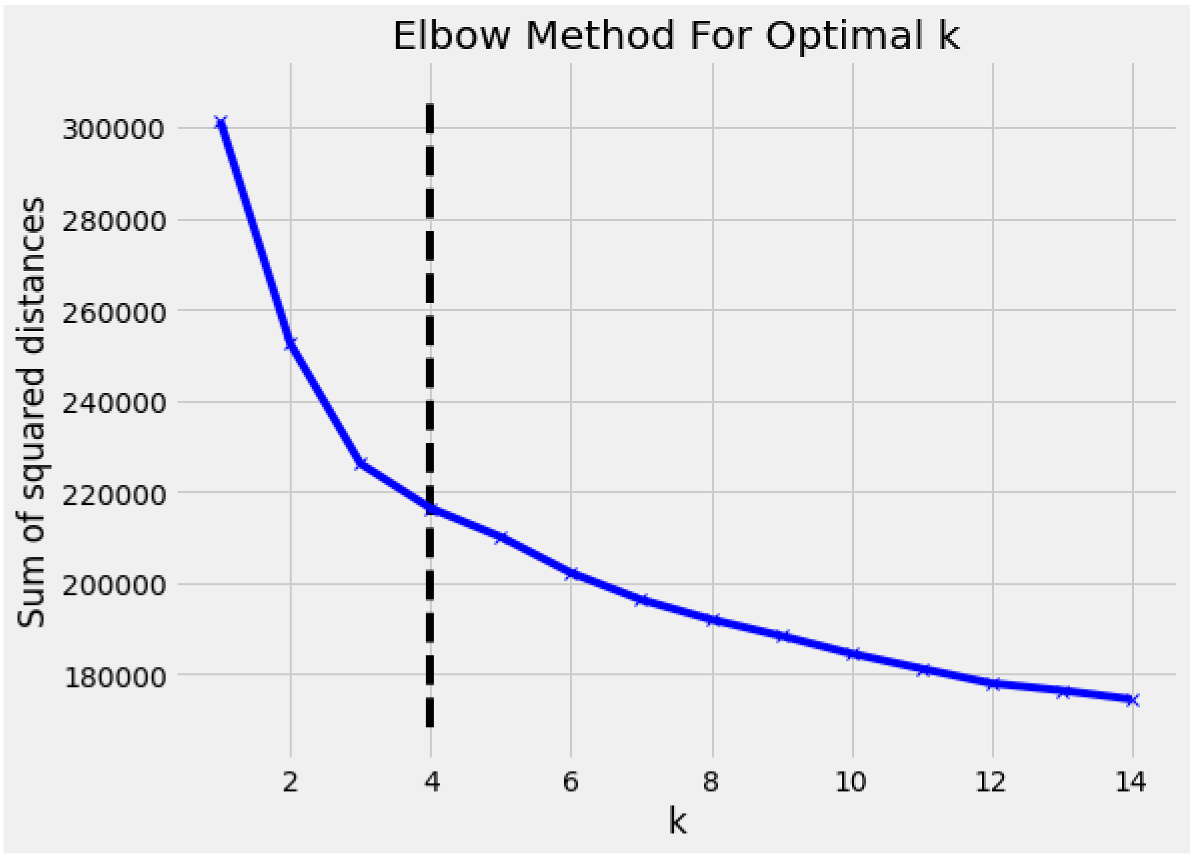
67
```
def cluster(model):
  model = model(random_state = 40, n_jobs = -1, n_clusters = find_optimal_clusters(model))

  model.fit(X)

  print_clusters_performance(model)

  return model
```

Defining clustering performance function

68
```
def print_clusters_performance(model):
  res = model.predict(X)

  print("Adjusted Rand Score : {0:4f}\n".format(metrics.adjusted_rand_score(y, res)))
  print("Adjusted Mutual Information : {0:4f}\n".format(metrics.adjusted_mutual_info_score(y, res)))
  print("Homogeneity Score : {0:4f}\n".format(metrics.homogeneity_score(y, res)))
  print("Completeness Score : {0:4f}\n".format(metrics.completeness_score(y, res)))
  print("V Measure Score : {0:4f}\n".format(metrics.v_measure_score(y, res)))
  print("Fowlkes Mallows Score : {0:4f}\n".format(metrics.fowlkes_mallows_score(y, res)))
  print("Contingency Matrix : {}\n".format(metrics.cluster.contingency_matrix(y, res)))

  print("Silhouette Score : {0:4f}\n".format(metrics.silhouette_score(X, res)))
  print("Harabasz Score : {0:4f}\n".format(metrics.calinski_harabasz_score(X, res)))
  print("Davies Bouldin Score : {0:4f}\n".format(metrics.davies_bouldin_score(X, res)))
```

K Means

70
```
kmeans = cluster(KMeans)
```



```
Adjusted Rand Score : 0.004366

Adjusted Mutual Information : 0.011712

Homogeneity Score : 0.017263

Completeness Score : 0.008911

V Measure Score : 0.011755

Fowlkes Mallows Score : 0.381594

Contingency Matrix : [[  414    14    70   306]
 [13768 19404 16629 19751]
 [ 5544  9691  7634  5280]
 [  298   230   756   211]]

Silhouette Score : 0.224719

Harabasz Score : 30240.771680

Davies Bouldin Score : 1.334234
```

K Mediods

71
```
from sklearn.cluster import DBSCAN

db = DBSCAN()

db.fit(X)
```

71
```
DBSCAN(algorithm='auto', eps=0.5, leaf_size=30, metric='euclidean',
       metric_params=None, min_samples=5, n_jobs=None, p=None)
```

72
```
metrics.v_measure_score(y, db.labels_)
```

72
```
1.4053685471137497e-15
```

73
```
metrics.adjusted_rand_score(y, db.labels_)
```

73
```
0.0
```

74
```
metrics.cluster.contingency_matrix(y, db.labels_)
```

74
```
array([[  804],
       [69552],
       [28149],
       [ 1495]])
```