```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import seaborn as sns

import os
for dirname, _, filenames in os.walk('/content/Accidents0515.csv'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


df1=pd.read_csv('/content/Accidents0515.csv',low_memory=False)
df2=pd.read_csv('/content/Accidents0515.csv',low_memory=False)
df3=pd.read_csv('/content/Accidents0515.csv',low_memory=False)
df = pd.concat([df1,df2,df3], axis=0)


df.head()
```

ce	Accident_Severity	Number_of_Vehicles	Number_of_Casualties	Date	...	Pede
1	2	1	1	04/01/2005	...	
1	3	1	1	05/01/2005	...	
1	3	2	1	06/01/2005	...	
1	3	1	1	07/01/2005	...	
1	3	1	1	10/01/2005	...	

```
df.head()
```

	Accident_Index	Location_Easting_OSGR	Location_Northing_OSGR	Longitude	Latitu
0	200501BS00001	525680.0	178240.0	-0.191170	51.4890
1	200501BS00002	524170.0	181650.0	-0.211708	51.5200
2	200501BS00003	524520.0	182240.0	-0.206458	51.5253
3	200501BS00004	526900.0	177530.0	-0.173862	51.4824
4	200501BS00005	528060.0	179040.0	-0.156618	51.4957

5 rows × 32 columns

```
df.columns
```

```
Index(['Accident_Index', 'Location_Easting_OSGR', 'Location_Northing_OSGR',
       'Longitude', 'Latitude', 'Police_Force', 'Accident_Severity',
       'Number_of_Vehicles', 'Number_of_Casualties', 'Date', 'Day_of_Week',
       'Time', 'Local_Authority_(District)', 'Local_Authority_(Highway)',
       '1st_Road_Class', '1st_Road_Number', 'Road_Type', 'Speed_limit',
       'Junction_Detail', 'Junction_Control', '2nd_Road_Class',
       '2nd_Road_Number', 'Pedestrian_Crossing-Human_Control',
       'Pedestrian_Crossing-Physical_Facilities', 'Light_Conditions',
       'Weather_Conditions', 'Road_Surface_Conditions',
       'Special_Conditions_at_Site', 'Carriageway_Hazards',
       'Urban_or_Rural_Area', 'Did_Police_Officer_Attend_Scene_of_Accident',
       'LSOA_of_Accident_Location'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 115680 entries, 0 to 38559
Data columns (total 32 columns):
 #   Column                                       Non-Null Count   Dtype
---  ------                                       --------------   -----
 0   Accident_Index                               115680 non-null  object
 1   Location_Easting_OSGR                        115641 non-null  float64
 2   Location_Northing_OSGR                       115641 non-null  float64
 3   Longitude                                    115641 non-null  float64
 4   Latitude                                     115641 non-null  float64
 5   Police_Force                                 115680 non-null  int64
 6   Accident_Severity                            115680 non-null  int64
 7   Number_of_Vehicles                           115680 non-null  int64
```

```
 8   Number_of_Casualties                         115680 non-null  int64
 9   Date                                         115680 non-null  object
 10  Day_of_Week                                  115680 non-null  int64
 11  Time                                         115677 non-null  object
 12  Local_Authority_(District)                   115680 non-null  int64
 13  Local_Authority_(Highway)                    115680 non-null  object
 14  1st_Road_Class                               115680 non-null  int64
 15  1st_Road_Number                              115680 non-null  int64
 16  Road_Type                                    115680 non-null  int64
 17  Speed_limit                                  115680 non-null  int64
 18  Junction_Detail                              115680 non-null  int64
 19  Junction_Control                             115680 non-null  int64
 20  2nd_Road_Class                               115680 non-null  int64
 21  2nd_Road_Number                              115680 non-null  int64
 22  Pedestrian_Crossing-Human_Control            115680 non-null  int64
 23  Pedestrian_Crossing-Physical_Facilities      115680 non-null  int64
 24  Light_Conditions                             115680 non-null  int64
 25  Weather_Conditions                           115680 non-null  int64
 26  Road_Surface_Conditions                      115680 non-null  int64
 27  Special_Conditions_at_Site                   115680 non-null  int64
 28  Carriageway_Hazards                          115680 non-null  int64
 29  Urban_or_Rural_Area                          115680 non-null  int64
 30  Did_Police_Officer_Attend_Scene_of_Accident  115680 non-null  int64
 31  LSOA_of_Accident_Location                    115461 non-null  object
dtypes: float64(4), int64(23), object(5)
memory usage: 29.1+ MB
```

```
df.dtypes
```

```
Accident_Index                                object
Location_Easting_OSGR                        float64
Location_Northing_OSGR                       float64
Longitude                                    float64
Latitude                                     float64
Police_Force                                   int64
Accident_Severity                              int64
Number_of_Vehicles                             int64
Number_of_Casualties                           int64
Date                                          object
Day_of_Week                                    int64
Time                                          object
Local_Authority_(District)                     int64
Local_Authority_(Highway)                     object
1st_Road_Class                                 int64
1st_Road_Number                                int64
Road_Type                                      int64
Speed_limit                                    int64
Junction_Detail                                int64
Junction_Control                               int64
2nd_Road_Class                                 int64
2nd_Road_Number                                int64
Pedestrian_Crossing-Human_Control              int64
Pedestrian_Crossing-Physical_Facilities        int64
Light_Conditions                               int64
Weather_Conditions                             int64
Road_Surface_Conditions                        int64
Special_Conditions_at_Site                     int64
Carriageway_Hazards                            int64
Urban_or_Rural_Area                            int64
Did_Police_Officer_Attend_Scene_of_Accident    int64
LSOA_of_Accident_Location                     object
dtype: object
```

```
df.isnull().sum()
```

```
Accident_Index                                0
Location_Easting_OSGR                        39
Location_Northing_OSGR                       39
Longitude                                    39
Latitude                                     39
Police_Force                                  0
Accident_Severity                             0
Number_of_Vehicles                            0
Number_of_Casualties                          0
Date                                          0
Day_of_Week                                   0
Time                                          3
Local_Authority_(District)                    0
Local_Authority_(Highway)                     0
1st_Road_Class                                0
1st_Road_Number                               0
Road_Type                                     0
Speed_limit                                   0
Junction_Detail                               0
Junction_Control                              0
2nd_Road_Class                                0
2nd_Road_Number                               0
Pedestrian_Crossing-Human_Control             0
```

```
Pedestrian_Crossing-Physical_Facilities          0
Light_Conditions                                 0
Weather_Conditions                               0
Road_Surface_Conditions                          0
Special_Conditions_at_Site                       0
Carriageway_Hazards                              0
Urban_or_Rural_Area                              0
Did_Police_Officer_Attend_Scene_of_Accident      0
LSOA_of_Accident_Location                      219
dtype: int64
```

```
df = df.drop(['Accident_Index', 'Location_Easting_OSGR', 'Location_Northing_OSGR','Longitude',
  'Latitude', 'Police_Force','Date','Local_Authority_(District)','Local_Authority_(Highway)',
  '1st_Road_Class', '1st_Road_Number', 'Speed_limit','Junction_Detail', 'Junction_Control', '2nd_Road_Class',
  '2nd_Road_Number', 'Pedestrian_Crossing-Human_Control', 'Pedestrian_Crossing-Physical_Facilities',
  'Special_Conditions_at_Site', 'Carriageway_Hazards','Did_Police_Officer_Attend_Scene_of_Accident',
  'LSOA_of_Accident_Location'], axis=1)
```

```
df.head()
```

	Accident_Severity	Number_of_Vehicles	Number_of_Casualties	Day_of_Week	Time	Road_Type	Light_Conditions	Weather_Conditions
0	2	1	1	3	17:42	6	1	2
1	3	1	1	4	17:36	3	4	1
2	3	2	1	5	00:15	6	4	1
3	3	1	1	6	10:35	6	1	1
4	3	1	1	2	21:13	6	7	1

```
df.isnull().sum()
```

```
Accident_Severity          0
Number_of_Vehicles         0
Number_of_Casualties       0
Day_of_Week                0
Time                       3
Road_Type                  0
Light_Conditions           0
Weather_Conditions         0
Road_Surface_Conditions    0
Urban_or_Rural_Area        0
dtype: int64
```

```
df.dropna(inplace=True)
```

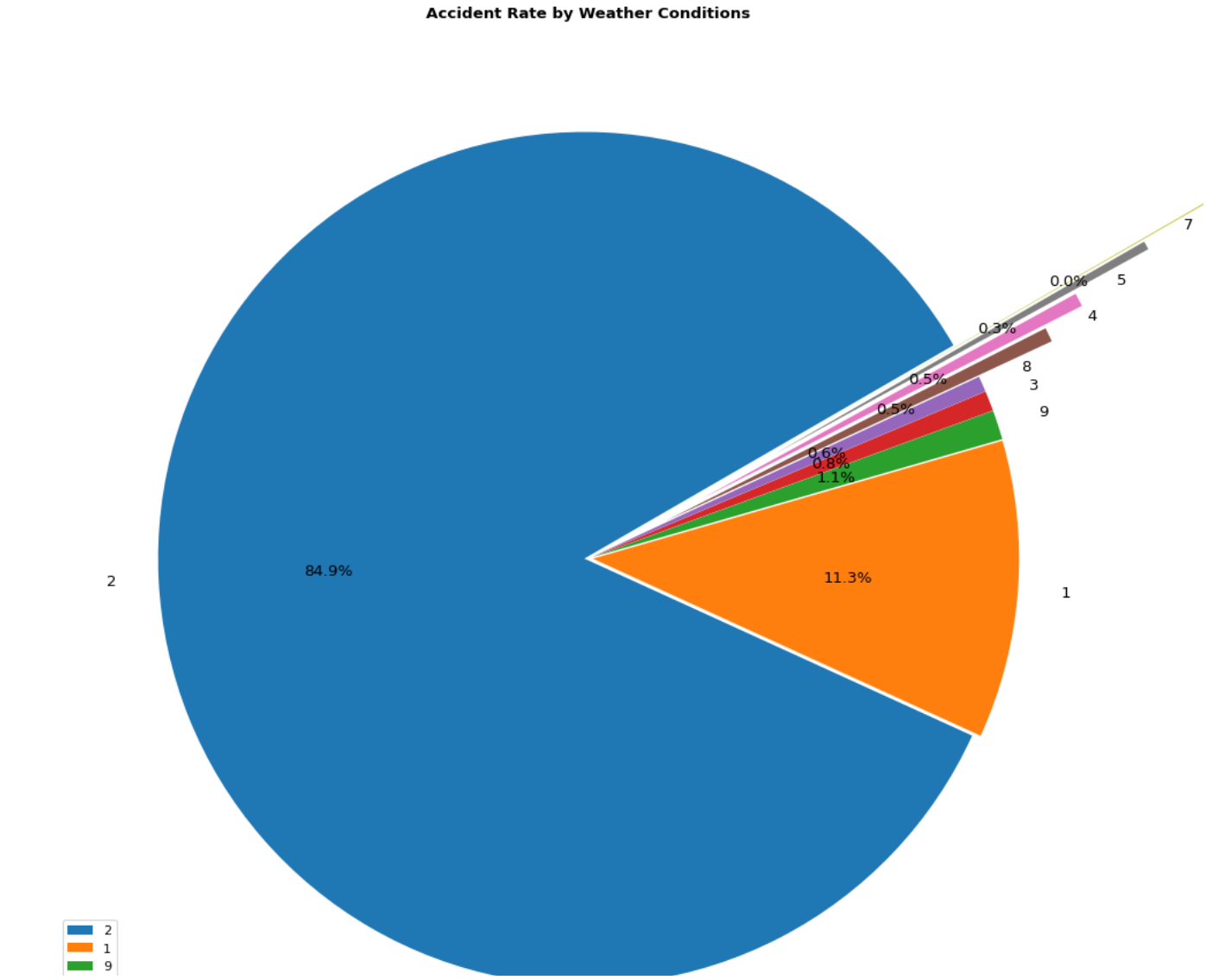
```
df.isnull().sum()
```

```
Accident_Severity          0
Number_of_Vehicles         0
Number_of_Casualties       0
Day_of_Week                0
Time                       0
Road_Type                  0
Light_Conditions           0
Weather_Conditions         0
Road_Surface_Conditions    0
Urban_or_Rural_Area        0
dtype: int64
```

What is the weather condition that causes the most traffic accidents?

```
weather_cond = df["Weather_Conditions"].value_counts()
weather_cond_arr = df["Weather_Conditions"].unique()
weather_num_acc_arr = weather_cond.values

figure(figsize=(15, 15), dpi=80)
plt.pie(weather_num_acc_arr, labels = weather_cond_arr,colors = sns.color_palette(),startangle = 30,textprops={'size': 'large'},explode=
plt.legend(loc ="lower left")
plt.title("Accident Rate by Weather Conditions",weight="bold")
plt.show()
```



What is the Difference Between Urban and Rural Area Traffic Accidents Risks?

```
df["Urban_or_Rural_Area"].value_counts()
plt.style.use('fivethirtyeight')

sns.countplot(x="Urban_or_Rural_Area",data=df)
plt.ylabel("Number of Accidents")
plt.xlabel("Urban(1) or Rural(2) Area")
plt.title("Urban Area vs Rural Area")

plt.show()
```

Urban Area vs Rural Area

Columns we need to edit: Road_Type, Weather_Conditions, Road_Surface_Conditions, Light_Conditions

```
df["Road_Type"]=[1 if each == "Single carriageway" else 2
                         if each == "Dual carriageway" else 3
                         if each == "One way street" else 4
                         if each == "Roundabout" else 5
                         if each == "Slip road" else 6
                         for each in df["Road_Type"]]
```

```
df["Road_Type"].value_counts()
```

```
6    115677
Name: Road_Type, dtype: int64
```

```
df["Weather_Conditions"]=[1 if each == "Raining without high winds" else 2
                         if each == "Fine without high winds" else 3
                         if each == "Unknown" else 4
                         if each == "Snowing without high winds" else 5
                         if each == "Other" else 6
                         if each == "Fine with high winds" else 7
                         if each == "Raining with high winds" else 8
                         if each == "Fog or mist" else 9
                         for each in df["Weather_Conditions"]]
```

```
df["Weather_Conditions"].value_counts()
```

```
9    115677
Name: Weather_Conditions, dtype: int64
```

```
df["Road_Surface_Conditions"]=[1 if each == "Wet/Damp" else 2
                         if each == "Dry" else 3
                         if each == "Frost/Ice" else 4
                         if each == "Snow" else 5
                         for each in df["Road_Surface_Conditions"]]
```

```
df["Road_Surface_Conditions"].value_counts()
```

```
5    115677
Name: Road_Surface_Conditions, dtype: int64
```

```
df["Light_Conditions"]=[1 if each == "Daylight: Street light present" else 2
                         if each == "Darkness: Street lights present and lit" else 3
                         if each == "Darkness: Street lighting unknown" else 4
                         if each == "Darkness: Street lights present but unlit" else 5
                         if each == "Darkeness: No street lighting" else 6
                         for each in df["Light_Conditions"]]
```

```
df["Light_Conditions"].value_counts()
```

```
6    115677
Name: Light_Conditions, dtype: int64
```

```
df.drop(["Time"], axis=1, inplace=True)
```

```
df.head()
```

	Accident_Severity	Number_of_Vehicles	Number_of_Casualties	Day_of_Week	Road_Type	Light_Conditions	Weather_Conditions	Road_
0	2	1	1	3	6	6	9	
1	3	1	1	4	6	6	9	
2	3	2	1	5	6	6	9	
3	3	1	1	6	6	6	9	
4	3	1	1	2	6	6	9	

Model Building

```
X = df.drop(["Accident_Severity"],axis=1).values
y = df["Accident_Severity"].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20, random_state=0)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((92541, 8), (23136, 8), (92541,), (23136,))
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Light Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)
```

```
▾ GradientBoostingClassifier
GradientBoostingClassifier()
```

```
GradientBoostingClassifierScore = gb.score(X_test,y_test)
print("Accuracy obtained by Gradient Boosting Classifier model:",GradientBoostingClassifierScore*100)
```

```
Accuracy obtained by Gradient Boosting Classifier model: 86.60096818810513
```

```
from sklearn import metrics

print('Training score: ', regressor.score(X_train, y_train))
print('Testing score: ', regressor.score(X_test, y_test))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Training score:  0.04424257955005917
Testing score:   0.04328769366758489
Root Mean Squared Error: 0.37137320865301926
```

Dession Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
DecisionTreeClassifierScore = dtc.score(X_test,y_test)
print("Accuracy obtained by Decision Tree Classifier model:",DecisionTreeClassifierScore*100)
```

```
Accuracy obtained by Decision Tree Classifier model: 86.6441908713693
```

```
from sklearn import metrics

print('Training score: ', regressor.score(X_train, y_train))
print('Testing score:  ', regressor.score(X_test, y_test))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Training score:  0.04424257955005917
Testing score:   0.04328769366758489
Root Mean Squared Error: 0.37137320865301926
```