ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐH KHOA HỌC TỰ NHIÊN

NHẬP MÔN KHOA HỌC DỮ LIỆU

BÁO CÁO CUỐI KÌ

Giảng viên giảng dạy: Th.S Hà Văn Thảo

Lớp: 20KDL1

Thứ: 4 Tiết 6-7-8

Nội dung: Road Accident Prediction and Classification

Thành viên:

- **Quách Phong Dương** — **MSSV: 20280022**
- **Nguyễn Huy Hoàng** — **MSSV: 20280038**
- **Nguyễn Nhật Đô** — **MSSV: 20280017**
- **Trần Thanh Hùng** — **MSSV: 20280035**

MỤC LỤC

I. Giới thiệu

Theo thống kê tử vong do Tổ chức Y tế Thế giới công bố, số vụ tai nạn giao thông xảy ra hàng năm trên thế giới rất đáng báo động. Tai nạn giao thông giết chết 1,35 triệu người mỗi năm và 50 triệu người bị thương. Khoảng 3.300 người thiệt mạng và 137.000 người bị thương mỗi ngày. Thiệt hại trực tiếp về kinh tế lên tới 43 tỷ USD, Tai nạn giao thông thường xuyên xảy ra sẽ đe dọa trực tiếp đến tính mạng con người và an toàn về mặt tài sản, ngoài ra với hàng triệu người khác bị thương nặng và phải sống với những hậu quả xấu về sức khỏe lâu dài.

Trên toàn cầu, tai nạn giao thông là nguyên nhân hàng đầu gây tử vong ở lứa tuổi thanh niên và trung niên và cũng là nguyên nhân chính gây tử vong ở những người từ 15–35 tuổi. Thương tích giao thông hiện được ước tính là nguyên nhân gây tử vong đứng hàng thứ tám ở mọi lứa tuổi các nhóm trên toàn cầu, và được dự đoán là nguyên nhân gây tử vong đứng hàng thứ bảy đến năm 2030.

Dự báo tai nạn giao thông là một trong những lĩnh vực nghiên cứu quan trọng nhất trong lĩnh vực an toàn giao thông. Các vụ tai nạn giao thông đường bộ xảy ra chủ yếu do đặc điểm hình học của đường, luồng giao thông, đặc điểm của người điều khiển phương tiện và môi trường đường bộ. Nhiều nghiên cứu đã được thực hiện để dự đoán tần suất tai nạn và phân tích các đặc điểm của tai nạn giao thông, bao gồm các nghiên cứu về xác định vị trí nguy hiểm / điểm nóng, phân tích mức độ thương tật do tai nạn và phân tích thời gian xảy ra tai nạn. Một số nghiên cứu tập trung vào hình thức của tai nạn. Các yếu tố khác bao gồm thời tiết và điều kiện ánh sáng của đoạn đường.

II. Ứng dụng của Máy Học

Tận dụng các công cụ và tất cả thông tin sẵn có ngày nay, một phân tích để dự đoán tai nạn giao thông và mức độ nghiêm trọng của nó sẽ tạo ra một phát triển đột phá để giảm thiểu sự cố tai nạn giao thông. Phân tích một loạt các yếu tố, bao gồm điều kiện thời tiết, thành thị hay nông thôn, loại đường và ánh sáng cùng những yếu tố khác để đưa ra dự đoán chính xác

về mức độ nghiêm trọng của các vụ tai nạn có thể được thực hiện. Do đó, các mối quan hệ này thường dẫn đến các sự cố giao thông nghiêm trọng có thể giúp xác định các vụ tai nạn nghiêm trọng.

Các đặc điểm thông tin này có thể được sử dụng bởi các dịch vụ khẩn cấp như Y Tế, Cảnh Sát Giao Thông, Cứu Hoả… , để gửi chính xác yêu cầu về nhân viên và thiết bị đến nơi xảy ra tai nạn, để lại nhiều nguồn nhân lực hơn sẵn sàng cho các tai nạn xảy ra. Hơn nữa, các tình huống có thể xảy ra tai nạn nghiêm trọng có thể được cảnh báo cho các bệnh viện gần đó để có thể chuẩn bị đầy đủ các thiết bị y tế để sẵn sàng cấp cứu cho các trường hợp nghiêm trọng.

Do đó, an toàn giao thông phải là mối quan tâm hàng đầu đối với chính phủ, chính quyền địa phương và các công ty tư nhân đầu tư vào các công nghệ có thể giúp giảm thiểu tai nạn và cải thiện an toàn lái xe tổng thể

Dữ liệu có thể góp phần xác định khả năng xảy ra tai nạn tiềm ẩn có thể bao gồm thông tin về các vụ tai nạn trước đó, chẳng hạn như đường bộ điều kiện, điều kiện thời tiết, thời gian và địa điểm chính xác của vụ tai nạn, loại phương tiện tham gia vụ tai nạn, thông tin về những người sử dụng liên quan đến vụ tai nạn và tất nhiên là mức độ nghiêm trọng của vụ tai nạn. Dự án này nhằm mục đích dự báo mức độ nghiêm trọng của tai nạn với thông tin được lấy từ các nhân chứng cung cấp về các dịch vụ khẩn cấp.

Các chính phủ cần hết sức quan tâm đến những dự đoán chính xác về mức độ nghiêm trọng của một vụ tai nạn, để giảm thời gian đến nơi tai nạn và sử dụng hiệu quả hơn các nguồn lực, và do đó giảm thiểu được một lượng đáng kể số người chết mỗi năm. Những người khác quan tâm có thể là các công ty tư nhân đầu tư vào các công nghệ nhằm cải thiện độ an toàn của đường bộ

III. DỮ LIỆU

1. NGUỒN DỮ LIỆU

Dữ liệu được lấy từ trang web của chính phủ www.data.gov.uk. Dữ liệu bao gồm tất cả các loại va chạm xe từ năm 2010 đến năm 2020. Mỗi cột của tập dữ liệu đều ở định dạng số. Tài liệu hỗ trợ để hiểu từng loại số trong tập dữ liệu về tai nạn được cung cấp trên trang web www.data.gov.uk.

1.1 KHAI BÁO THƯ VIỆN

Việc khai báo thư viện là một phần không thể thiếu. Với các thư viện quen thuộc như Pandas, Numpy, Matplotlib và mỗi khi sử dụng đồ thị, chúng em sẽ chọn màu chủ đạo là đen để nhìn rõ hơn các điểm trong đồ thị

Entrée [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('dark_background')
%matplotlib inline
import seaborn as sns

import scipy.stats as stats
from scipy.stats import chi2_contingency

import warnings
warnings.filterwarnings('ignore')
```

1.2 ĐỌC DỮ LIỆU

Read dataset

Entrée [2]:
```
accidents=pd.read_csv("accidents_final.csv",index_col='Accident_Index')
casualties=pd.read_csv("casualties_final.csv",index_col='Accident_Index')
vehicles=pd.read_csv("vehicles_final.csv",index_col='Accident_Index')
```

Tiến hành với việc đọc dữ liệu của 3 tập tin (accidents, casualties, vehicles) Nhưng 3 tập tin có một cột Accident_Index là mã tai nạn, mỗi tai nạn sẽ có chung 1 mã riêng biệt

Table 1. *Các yếu tố quan trọng*

Day_of_Week :Numeric: 1 for Sunday, 2 for Monday, and so on.
Latitude and Longitude (Vĩ độ và Kinh độ)
Light_Conditions : Day, night, street lights or not. **Điều kiện ánh sáng: (ban ngày, ban đêm hoặc không có ánh sáng)**
Weather_Conditions: Wind, rain, snow, fog. **Điều kiện thời tiết:** gió, mưa, tuyết, sương mù.
Vehicle Type: Pedal cycle, Motorcycle, Car **Loại Phương tiện:** xe đạp, xe máy, xe hơi
Road_Surface_Conditions :Wet, snow, ice, flood. **Điều kiện mặt đương**: ướt, tuyết, đóng băng, lũ
Speed Limit : 60 mph , 70 mph **Giới hạn tốc độ:** ~96.6km/h, ~112.6km/h
Output
Accident Severity : 0 = Extremely serious, 1 = Serious, 2 = unserious

2. LỰA CHỌN BIẾN HỒI QUY

Dữ liệu được chia thành 3 bộ dữ liệu khác nhau, bao gồm tất cả các vụ tai nạn được ghi lại ở Anh từ năm 2010 đến năm 2020. Bộ dữ liệu đặc điểm chứa thông tin về thời gian, địa điểm, loại va chạm, điều kiện thời tiết và ánh sáng và loại giao lộ nơi nó xảy ra.

Tập dữ liệu địa điểm có các chi tiết cụ thể như độ dốc, hình dạng và loại đường, chế độ giao thông, điều kiện bề mặt và cơ sở hạ tầng. Trên tập dữ liệu người dùng, có thể tìm thấy vị trí do người sử dụng phương tiện đang đứng, thông tin về những người sử dụng có liên quan trong tai nạn, lý do đi du lịch, mức độ nghiêm trọng của tai nạn, việc sử dụng an toàn thiết bị và thông tin về người đi bộ. Bộ dữ liệu xe chứa lưu lượng và loại phương tiện, và ngày lễ có nhãn các vụ tai nạn xảy ra trong một ngày lễ. Tất cả 3 bộ dữ liệu đều có chung số nhận dạng vụ tai nạn.

Một phân tích ban đầu về dữ liệu đã được thực hiện để lựa chọn các tính năng liên quan cho vấn đề cụ thể này, nhóm em có sử dụng thuật toán tìm những tính năng quan trọng cho máy học bằng cách sử dụng thuật toán Chi – Bình Phương (Chi - Square).

Entrée [59]:
```
class ChiSquare:
    def __init__(self, dataframe):
        self.df = dataframe
        self.p = None #P-Value
        self.chi2 = None #Chi Test Statistic
        self.dof = None

        self.dfObserved = None
        self.dfExpected = None

    def _print_chisquare_result(self, colX, alpha):
        result = ""
        if self.p<alpha:
            result="The column {0} is IMPORTANT for Prediction".format(colX)
        else:
            result="The column {0} is NOT an important predictor. (Discard {0} from model)".format(colX)

        print(result)

    def TestIndependence(self,colX,colY, alpha=0.05):
        X = self.df[colX].astype(str)
        Y = self.df[colY].astype(str)

        self.dfObserved = pd.crosstab(Y,X)
        chi2, p, dof, expected = stats.chi2_contingency(self.dfObserved.values)
        self.p = p
        self.chi2 = chi2
        self.dof = dof

        self.dfExpected = pd.DataFrame(expected, columns=self.dfObserved.columns,
                                       index = self.dfObserved.index)

        self._print_chisquare_result(colX,alpha)

#Initialize ChiSquare Class
cT = ChiSquare(df)
```

Link tham khảo : https://www.hackdeploy.com/chi-square-feature-selection-in-python/

Entrée [61]:

```
cT = ChiSquare(df_ML)
testColumns = ['Longitude', 'Latitude', 'Police_Force',
       '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',
       'Year', 'Hour', 'Unnamed: 0', 'Vehicle_Reference', 'Vehicle_Type',
       'Towing_and_Articulation', 'Vehicle_Manoeuvre',
       'Vehicle_Location-Restricted_Lane', 'Junction_Location',
       'Skidding_and_Overturning', 'Hit_Object_in_Carriageway',
       'Vehicle_Leaving_Carriageway', 'Hit_Object_off_Carriageway',
       '1st_Point_of_Impact', 'Was_Vehicle_Left_Hand_Drive?',
       'Journey_Purpose_of_Driver', 'Sex_of_Driver', 'Age_of_Driver',
       'Engine_Capacity_(CC)', 'Propulsion_Code', 'Age_of_Vehicle',
       'Driver_IMD_Decile', 'Driver_Home_Area_Type']
for col in testColumns:
    cT.TestIndependence(colX=col,colY="Accident_Severity",alpha=0.02)
```

```
The column Longitude is IMPORTANT for Prediction
The column Latitude is IMPORTANT for Prediction
The column Police_Force is IMPORTANT for Prediction
The column Number_of_Vehicles is IMPORTANT for Prediction
The column Number_of_Casualties is IMPORTANT for Prediction
The column Date is IMPORTANT for Prediction
The column Day_of_Week is IMPORTANT for Prediction
The column Time is IMPORTANT for Prediction
The column Local_Authority_(District) is IMPORTANT for Prediction
The column Local_Authority_(Highway) is IMPORTANT for Prediction
The column 1st_Road_Class is IMPORTANT for Prediction
The column 1st_Road_Number is IMPORTANT for Prediction
The column Road_Type is IMPORTANT for Prediction
The column Speed_limit is IMPORTANT for Prediction
The column Junction_Detail is IMPORTANT for Prediction
The column Junction_Control is IMPORTANT for Prediction
The column 2nd_Road_Class is IMPORTANT for Prediction
The column 2nd_Road_Number is IMPORTANT for Prediction
The column Pedestrian_Crossing-Human_Control is IMPORTANT for Prediction
The column Pedestrian_Crossing-Physical_Facilities is IMPORTANT for Prediction
The column Light_Conditions is IMPORTANT for Prediction
The column Weather_Conditions is IMPORTANT for Prediction
The column Road_Surface_Conditions is IMPORTANT for Prediction
The column Special_Conditions_at_Site is IMPORTANT for Prediction
The column Carriageway_Hazards is IMPORTANT for Prediction
The column Urban_or_Rural_Area is IMPORTANT for Prediction
The column Did_Police_Officer_Attend_Scene_of_Accident is IMPORTANT for Prediction
The column Year is IMPORTANT for Prediction
The column Hour is IMPORTANT for Prediction
The column Unnamed: 0 is IMPORTANT for Prediction
The column Vehicle_Reference is IMPORTANT for Prediction
```

Thuật toán cho ra kết quả khả quan hầu như các thuộc tính đều quan trọng cho phần dự đoán. Vì thế chúng em sẽ chọn ra các thuộc tính phù hợp và lọc ra 11 thuộc tính quan trọng nhất cho dự đoán bằng thuật toán SelectKBest:

Thuật toán tìm ra các trường tốt nhất để dự đoán

Entrée [90]:

```
from sklearn.feature_selection import SelectKBest, mutual_info_regression
#Select top 11 features based on mutual info regression
selector = SelectKBest(mutual_info_regression, k = 11)
selector.fit(X, y)
X.columns[selector.get_support()]
```

Out[90]:

```
Index(['Day_of_Week', 'Road_Type', 'Speed_limit', 'Weather_Conditions',
       'Urban_or_Rural_Area', 'Did_Police_Officer_Attend_Scene_of_Accident',
       'Vehicle_Type', 'Sex_of_Driver', 'Age_of_Driver', 'Age_of_Vehicle',
       'Engine_Capacity_(CC)'],
      dtype='object')
```

3. Ý NGHĨA CỦA CÁC YẾU TỐ PHÂN LOẠI THỜI TIẾT VÀ ÁNH SÁNG

Table 2. *Weather Conditions*

Giá trị	Ý nghĩa
1	Fine no high winds
2	Raining no high winds
3	Snowing no high winds
4	Fine + high winds
5	Raining + high winds
6	Snowing + high winds
7	Fog or mist
8	Other
9	Unknown
-1	Data missing or out of range

Table 4. *Light Conditions*

Giá trị	Ý nghĩa
1	Daylight
4	Darkness – lights lit
5	Darkness – lights unlit
6	Darkness – mo lighting
7	Darkness – lighting unknown
-1	Data missing or out of range

Road Surface Conditions and Gender of Driver and Vehicle Type

Table 4. *Road Conditions*

Giá trị	Ý nghĩa
1	Dry
2	Wet or damp
3	Snow
4	Frost or ice
5	Flood over 3cm, deep
6	Oil or diesel
7	Mud
-1	Data missing or out of range

Table 5. *Gender*

Giá trị	Ý nghĩa
1	Male
4	Female
5	Not known
-1	Data missing

Table 6. *Vehicle Type*

Giá trị	Ý nghĩa
1	Pedal cycle
2	Motorcycle 50cc and under
3	Motorcycle 125cc and under
4	Motorcycle over 125cc and up to 500cc
5	Motorcycle over 500cc
8	Taxi/Private hire car
9	Car
10	Minibus (8-16 passenger seats)
11	Bus or coach (17 or more pass seats)
16	Ridden horse
17	Agricultural vehicle
18	Tram
19	Van/Goods 3.5 tonnes mgw or under
20	Goods over 3.5L and under 7.5t
21	Goods 7.5 tonnes mgw and over
22	Mobility scooter
23	Electric motorcycle
90	Other vehicle
97	Motorcycle – unknown cc
98	Goods vehicle – unknown weight
-1	Data missing or out of range

Table 7. *Day of Week*

Giá trị	Ý nghĩa
1	Sunday
2	Monday
3	Tuesday
4	Wednesdday
5	Thursday
6	Friday
7	Saturday

4. LÀM SẠCH DỮ LIỆU

Làm sạch dữ liệu là quá trình cung cấp một định dạng thích hợp cho dữ liệu để phân tích thêm

```
Entrée [7]: for col in casualties.columns:
                if(casualties.dtypes[col]!='O'):
                    mean=casualties[col].mean(skipna=True)
                    casualties[col]=casualties[col].mask(casualties[col] == -1,round(mean))
```

```
Entrée [10]: for col in vehicles.columns:
                 if(vehicles.dtypes[col]!='O'):
                     mean=vehicles[col].mean(skipna=True)
                     vehicles[col]=vehicles[col].mask(vehicles[col] == -1,round(mean))
```

```
Entrée [13]: for col in accidents.columns:
                 if(accidents.dtypes[col]!='O'):
                     mean=accidents[col].mean(skipna=True)
                     accidents[col]=accidents[col].mask(accidents[col] == -1,round(mean))
```

Trong tập dữ liệu cụ thể này, có hai loại giá trị bị thiếu '-1' và 'Nan'. Chúng em sẽ xem xét từng cột với tổng giá trị còn thiếu. Chúng em sẽ duyệt xem từng thuộc tính mà thuộc tính nào không là object thì sẽ thay bằng trung bình của cột đó

```
Entrée [11]: accidents.drop_duplicates(inplace=True)
             accidents.dropna(inplace=True)
```

```
Entrée [9]: vehicles.isnull().sum()

Out[9]: Unnamed: 0                      0
        Vehicle_Reference               0
        Vehicle_Type                    0
        Towing_and_Articulation         0
        Vehicle_Manoeuvre               0
        Vehicle_Location-Restricted_Lane  0
        Junction_Location               0
        Skidding_and_Overturning        0
        Hit_Object_in_Carriageway       0
        Vehicle_Leaving_Carriageway     0
        Hit_Object_off_Carriageway      0
        1st_Point_of_Impact             0
        Was_Vehicle_Left_Hand_Drive?    0
        Journey_Purpose_of_Driver       0
        Sex_of_Driver                   0
        Age_of_Driver                   0
        Engine_Capacity_(CC)            0
        Propulsion_Code                 0
        Age_of_Vehicle                  0
        Driver_IMD_Decile               0
        Driver_Home_Area_Type           0
        dtype: int64
```

Ta thử kiểm tra lại bằng cách tính tổng xem có giá trị Null trong các cột hay không?

Thì kết quả cho ra là 0 của từng cột vì vậy ta khẳng định rằng dữ liệu đã được làm sạch

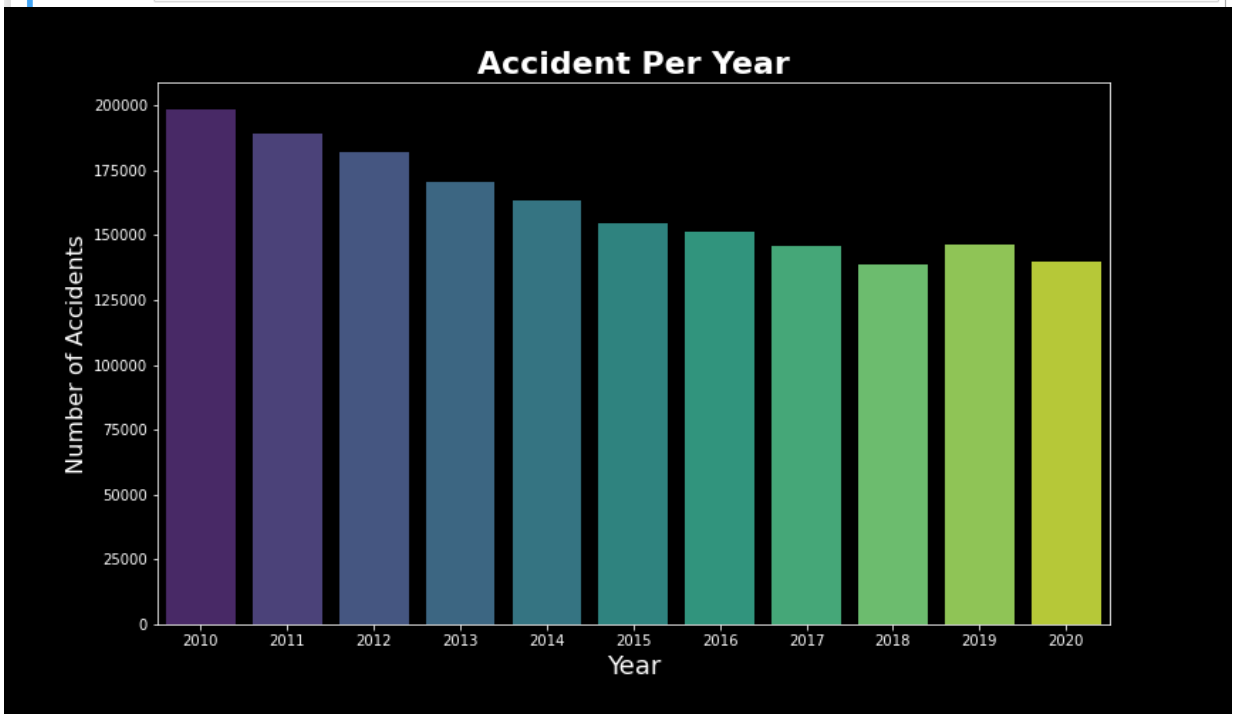
IV. TRỰC QUAN HOÁ DỮ LIỆU

Điều đầu tiên chúng ta có thể làm là tìm hiểu về thời gian xảy ra tai nạn để đề phòng và độ tuổi của một số tài xế có liên quan đến vụ tai nạn. Tìm hiểu xem mức độ tai nạn của các năm có sự thay đổi theo hướng nào.

- Tìm ra số vụ tai nạn theo từng năm
- Tìm số vụ tai nạn vào các ngày trong tuần
- Tìm hiểu về mức độ tai nạn ở thành thị và nông thôn
- Tìm hiểu về các vụ tai nạn xảy ra ở các đường có giới hạn tốc độ
- Tìm hiểu về số vụ tai nạn theo giờ trong ngày
- Tìm hiểu về tuổi của người lái xe trong các vụ tai nạn

1. SỐ VỤ TAI NẠN THEO TỪNG NĂM

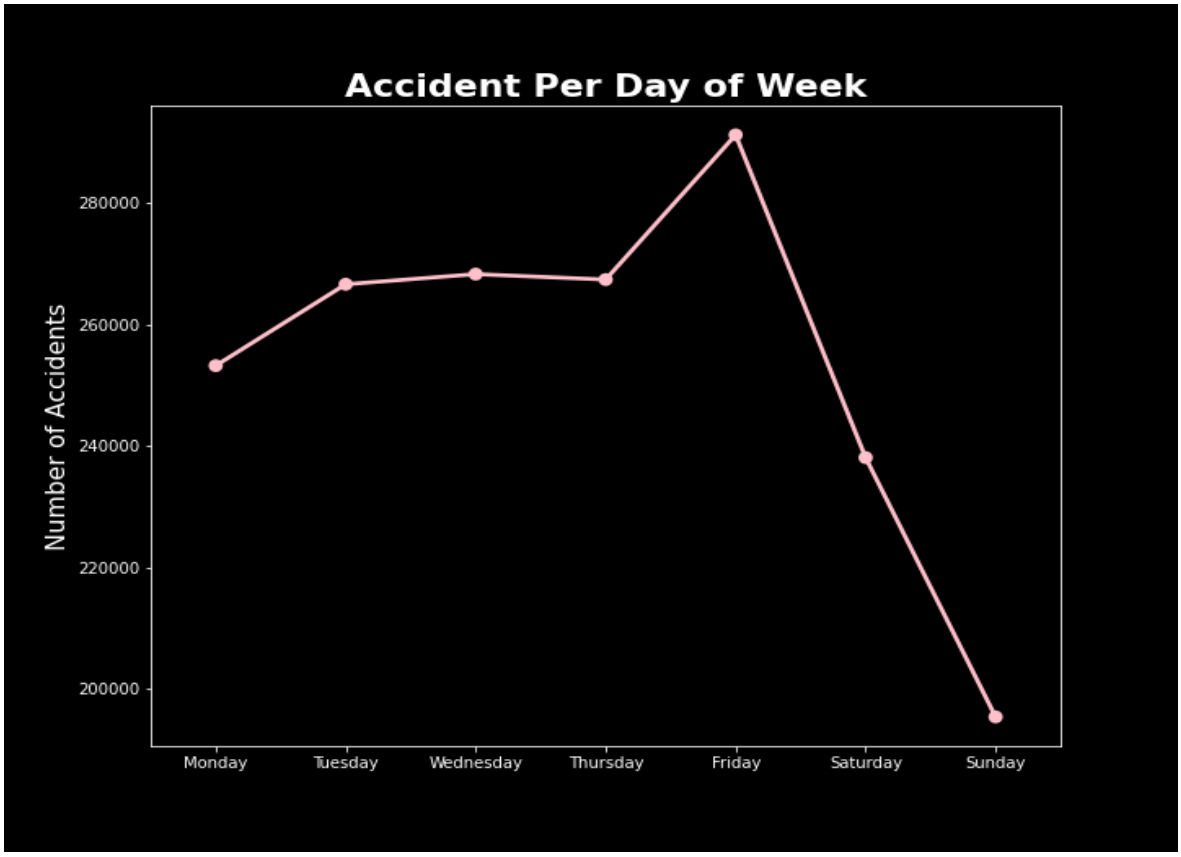
```
Entrée [20]: AccidentsPerYear=accidents.groupby(["Year"])["index"].count()

Entrée [119]: plt.style.use('dark_background')
plt.figure(figsize=(12,7))
sns.barplot(AccidentsPerYear.index,AccidentsPerYear.values,palette="viridis")
plt.title("Accident Per Year",fontsize=22,fontweight="bold")
plt.xlabel("Year",fontsize=18)
plt.ylabel("Number of Accidents",fontsize=16)
plt.savefig('ACCIDENT PER YEAR.png')
plt.show()
```



Hình ảnh cho thấy số vụ tai nạn giao thông giảm dần qua các năm từ 2010 đến 2020, cho thấy được mức độ an toàn đã có sự cải thiện theo từng năm, nhưng số vụ tai nạn vẫn còn khá cao

2. SỐ VỤ TAI NẠN VÀO CÁC NGÀY TRONG TUẦN

```
Entrée [120]: plt.style.use('dark_background')
              plt.figure(figsize=(10,8))
              sns.pointplot(AccidentPerDayOfWeek.index,AccidentPerDayOfWeek.values,color="pink",linewidth=4,markers='o', markersize=4)
              plt.title("Accident Per Day of Week",fontsize=22,fontweight="bold")
              plt.xlabel("")
              plt.ylabel("Number of Accidents",fontsize=16)
              plt.savefig('ACCIDENT PER Day of Week.png')
              plt.show()
```



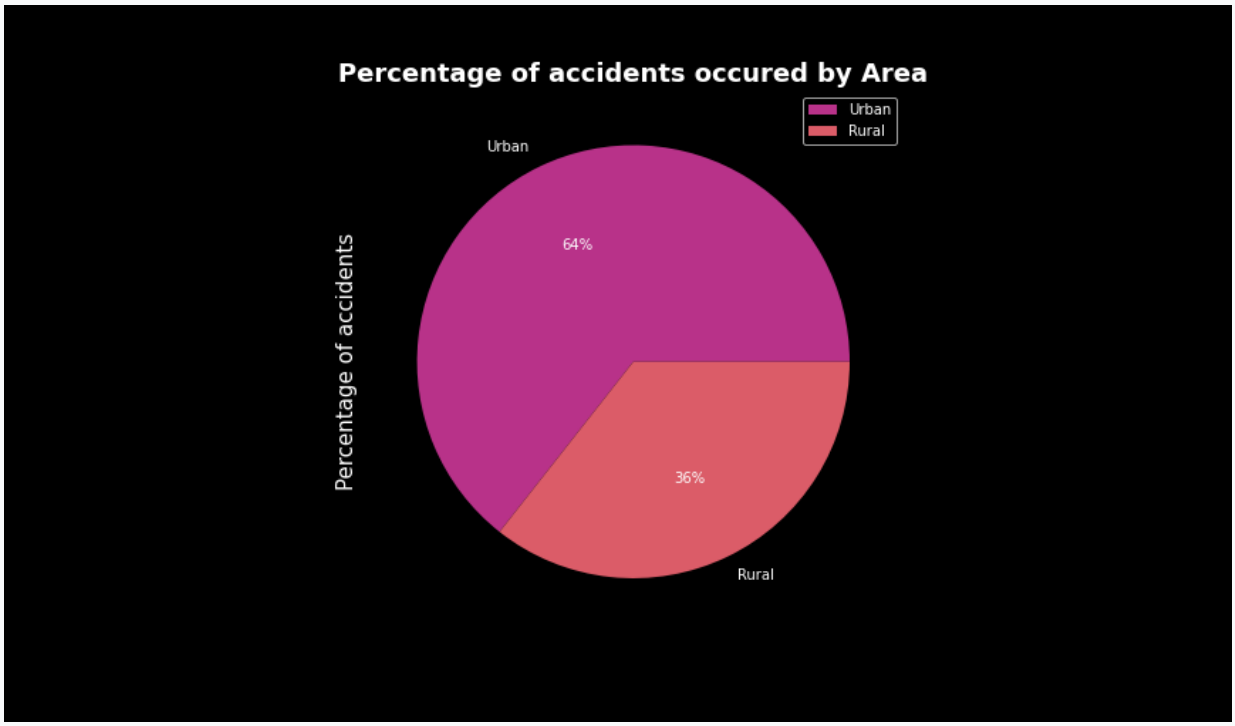
Các vụ tai nạn trong tuần dường như đi theo chiều hướng lên vào các ngày hành chính, cũng dễ hiểu khi vào những ngày này chúng ta thường hay đi làm và đi học nhiều, sẽ dễ gây ra tai nạn hơn. Vào thứ 6 đồ thị cao nhất có thể thấy đây là thời điểm trong tuần mọi người ra ngoài nhiều hơn vì thường đa số các doanh nghiệp lớn nhỏ, họ sẽ vận chuyển hàng hoá vào thời điểm này nhiều hơn các ngày trong tuần.

Thấp nhất là ngày chủ nhật vì đơn giản mà nói thì đa số vào cuối tuần mọi người sẽ có xu hướng nghỉ ngơi ở nhà sau một tuần dài làm việc, họ sẽ có chủ trương nghỉ dưỡng hơn là đi đâu đó vào ngày cuối tuần.

3. MỨC ĐỘ TAI NẠN Ở THÀNH THỊ VÀ NÔNG THÔN

```
Entrée [25]: accidents["Urban_or_Rural_Area"].value_counts()

    Out[25]: Urban    1146322
             Rural     633974
             Other         36
             Name: Urban_or_Rural_Area, dtype: int64
```

```
Entrée [121]: UrbanArea=len(accidents[accidents["Urban_or_Rural_Area"]=="Urban"])
              RuralArea=len(accidents[accidents["Urban_or_Rural_Area"]=="Rural"])
              OtherArea=len(accidents[accidents["Urban_or_Rural_Area"]=="Other"])
              SumOfArea=UrbanArea+RuralArea+OtherArea
              PerUrban=UrbanArea*1.0/SumOfArea*1.0*100

              PerRural=RuralArea*1.0/SumOfArea*1.0*100

              PerOther=OtherArea*1.0/SumOfArea*1.0*100

              print("Percentage of accidents occur in urban areas is {0:.0f}%".format(PerUrban))
              print("Percentage of accidents occur in rural areas is {0:.0f}%".format(PerRural))
              print("Percentage of accidents occur in other areas is {0:.0f}%".format(PerOther))
              labels = ['Urban', 'Rural']
              y = [PerUrban, PerRural]
              plt.figure(figsize=(12,7))
              colors = sns.color_palette('plasma')[2:4]
              plt.pie(y, labels = labels, colors = colors, autopct='%.0f%%')
              plt.ylabel('Percentage of accidents',fontsize=15)
              plt.title("Percentage of accidents occured by Area",fontsize=18,fontweight="bold")
              plt.legend()
              plt.savefig('PERCENTAGE OF ACCIDENTS.png')
              plt.show()
```



Urban là đô thị, Rural là nông thôn

Dễ dàng kết luận khi đời sống ở đô thị họ tấp nập hơn, xe cộ lưu thông đông hơn, sự va chạm vào nhau nhiều hơn nên dễ gây ra tai nạn hơn so với ở nông thôn.

4. TAI NẠN XẢY RA Ở ĐƯỜNG CÓ GIỚI HẠN TỐC ĐỘ

Entrée [29]:
```
#setups for adding frequencies to visualizations
dftotal= float(len(accidents))
nstotal= float(len(unserious_accident))
setotal= float(len(serious_accident))
esetotal=float(len(extreme_serious_accident))
```

Entrée [30]:
```
splt_order=[30,40,50,60,70,80]
splt_order2=[30,40,50,60,70,80]
splt_order3=[30,40,50,60,70,80]
fig, ax =plt.subplots(nrows=3, ncols=1,figsize = (15,14))

ax1 =sns.countplot("Accident_Severity", hue="Speed_limit", hue_order=splt_order,
              palette="plasma", data=unserious_accident, ax=ax[0])
for p in ax1.patches:
    height = p.get_height()
    ax1.text(p.get_x()+p.get_width()/2.,
            height + 4,
            '{:1.4f}%'.format(height/nstotal*100),
            ha="center",fontsize=12)
ax2 =  sns.countplot("Accident_Severity", hue="Speed_limit", hue_order=splt_order2,
               palette="plasma", data=serious_accident, ax=ax[1])
for p in ax2.patches:
    height = p.get_height()
    ax2.text(p.get_x()+p.get_width()/2.,
            height + 4,
            '{:1.2f}%'.format(height/setotal*100),
            ha="center",fontsize=12)
ax3 =  sns.countplot("Accident_Severity", hue="Speed_limit", hue_order=splt_order3,
               palette="plasma", data=extreme_serious_accident, ax=ax[2])
for p in ax3.patches:
    height = p.get_height()
    ax3.text(p.get_x()+p.get_width()/2.,
            height + 4,
            '{:1.2f}%'.format(height/esetotal*100),
            ha="center",fontsize=12)
fig.suptitle("Speed Limit in Accidents", fontsize=18, fontweight="bold")
ax1.set_xlabel('Speed Limit of Not Serious Accidents', fontsize=15, fontweight="bold")
ax2.set_xlabel('Speed Limit of Serious Accidents', fontsize=15, fontweight="bold")
ax3.set_xlabel('Speed Limit of Extremely Serious Accidents', fontsize=15, fontweight="bold")
ax1.set_ylabel("Number of Accidents",fontsize=10)
ax2.set_ylabel("Number of Accidents",fontsize=10)
ax3.set_ylabel("Number of Accidents",fontsize=10)
fig.show()
plt.savefig('Speed Limit in Accidents.png')
```

```
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
```

Đoạn code giúp gộp 3 đồ thị để thấy rõ sự thay đổi theo thông số của từng mức độ nguy hiểm khác nhau. Đồ thị này ngoài thấy được tai nạn xảy ra ở các đường có giới hạn tốc độ còn cho thấy mức độ tai nạn nguy hiểm dựa theo 3 mức độ

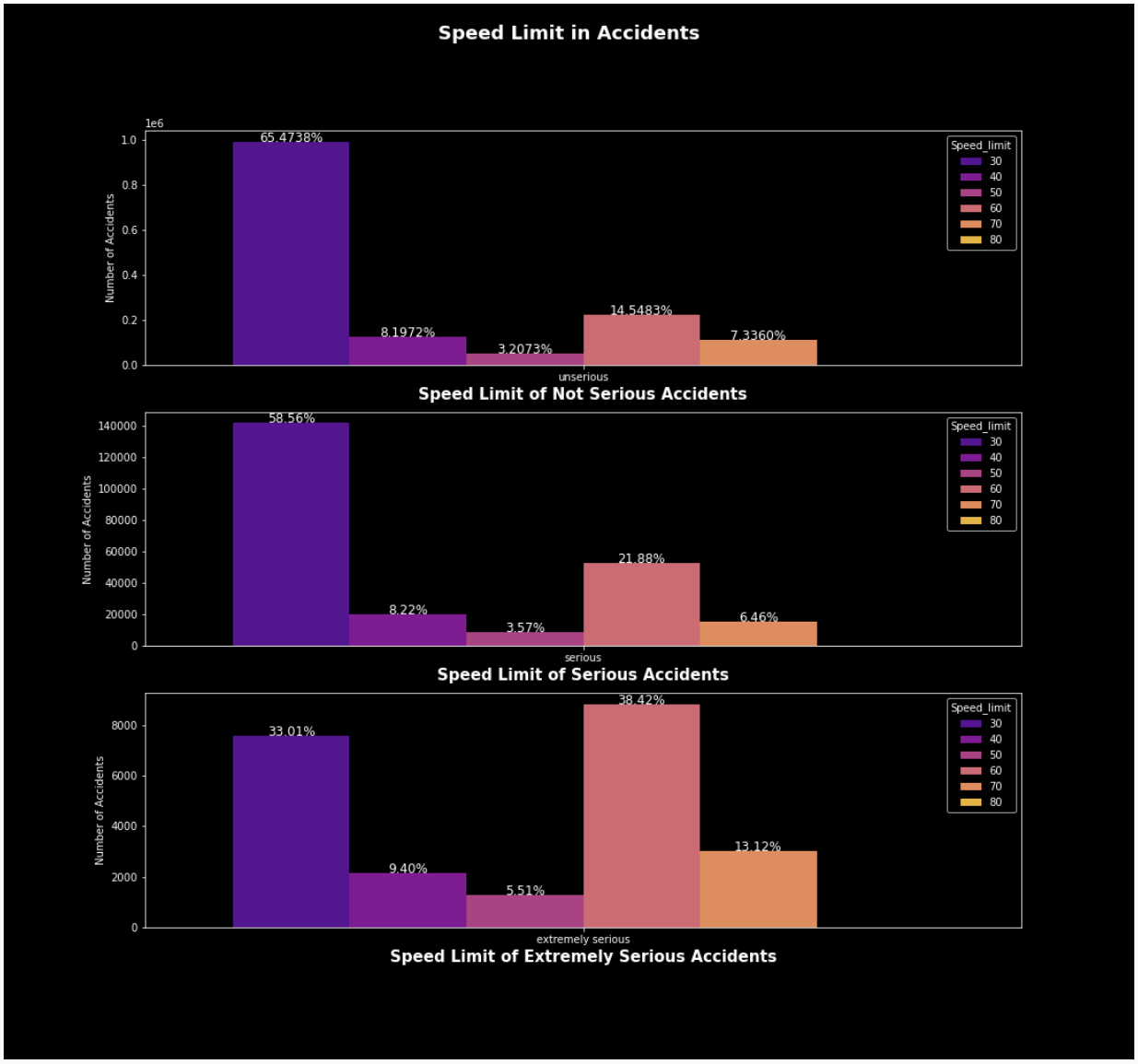
(**Accident Severity: 0 = Extremely serious**

1 = serious

2 = unserious)

Đối với đồ thị đầu tiên (Not Serious Accidents) hầu hết các vụ tai nạn xảy ra trên đường có giới hạn tốc độ là 30 với tỷ lệ là 65,4% cao nhất và nối tiếp là ở tốc độ 60 với tỷ lệ là 14,5% cho thấy rằng tỉ trọng ở mức không nghiêm trọng thường xảy ra ở một số vụ tai nạn có thể là do biển báo dừng, chuyển làn hoặc rẽ vào bãi đậu xe, v.v.

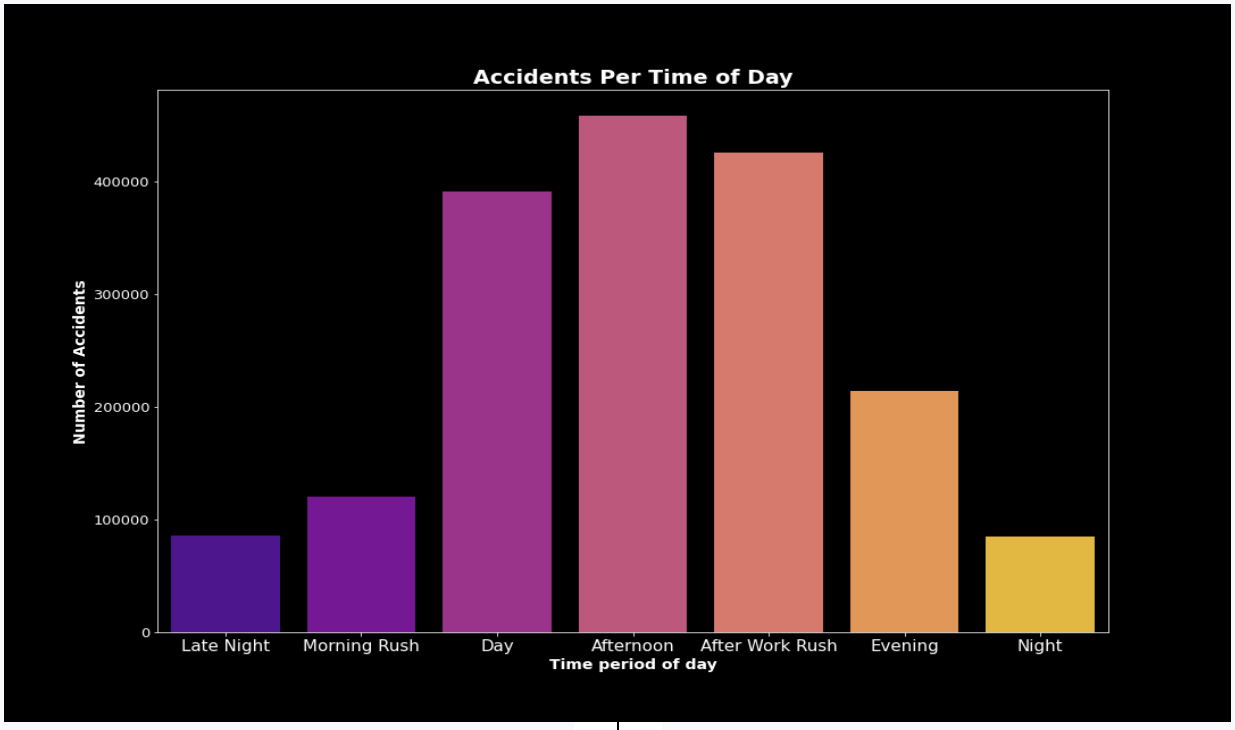
Tiếp theo, đồ thị thứ 2 (Serious Accidents) mức độ nghiêm trọng trong tai nạn, đa số các vụ tai nạn ở giới hạn tốc độ 30 vẫn khá cao nhưng dường như bị mất ưu thế hơn so với đồ thị đầu, còn ở tốc độ 60 thì đang bắt đầu chiếm tỉ lệ cao hơn có thể thấy rằng các vụ tai nạn có mức độ nghiêm trọng đang dần chuyển sang ở các đường có giới hạn là 60. Có thể thấy mức độ nghiêm trọng của tai nạn ảnh hưởng nhiều đến giới hạn tốc độ theo một tỷ lệ thuận.



Đồ thị thứ 3 (Extremely Serious Accidents) mức độ Cực kì nghiêm trọng, giới hạn tốc độ 30 đã giảm khá nhiều so với 2 đồ thị trước nhưng các giới hạn tốc độ khác tăng lên một cách đột biến, ở các tốc độ 50 60 tăng tận 1.5 lần, ở tốc độ 80 tăng lên gấp đôi, có thể thấy rõ là có sự tỷ lệ thuận trong mức độ tai nạn và giới hạn tốc độ, điều đó càng làm cho chúng ta dễ dàng khẳng định và rút ra bài học khi tham gia an toàn giao thông là: "***Tăng tốc độ là tăng mức độ tai nạn***"

5. SỐ VỤ TAI NẠN THEO TỪNG GIỜ TRONG NGÀY

```
Entrée [45]: plt.style.use('dark_background')
             plt.figure(figsize=(15,10))
             sns.barplot(accidentsPerHour.index,accidentsPerHour.values, order=indexDay, palette='plasma')
             plt.title("Accidents Per Time of Day",fontsize=19,fontweight="bold")
             plt.xlabel("Time period of day", fontsize=14, fontweight="bold")
             plt.ylabel("\nNumber of Accidents", fontsize=14, fontweight="bold")
             plt.xticks(fontsize=15)
             plt.yticks(fontsize=13)
             plt.show()
```



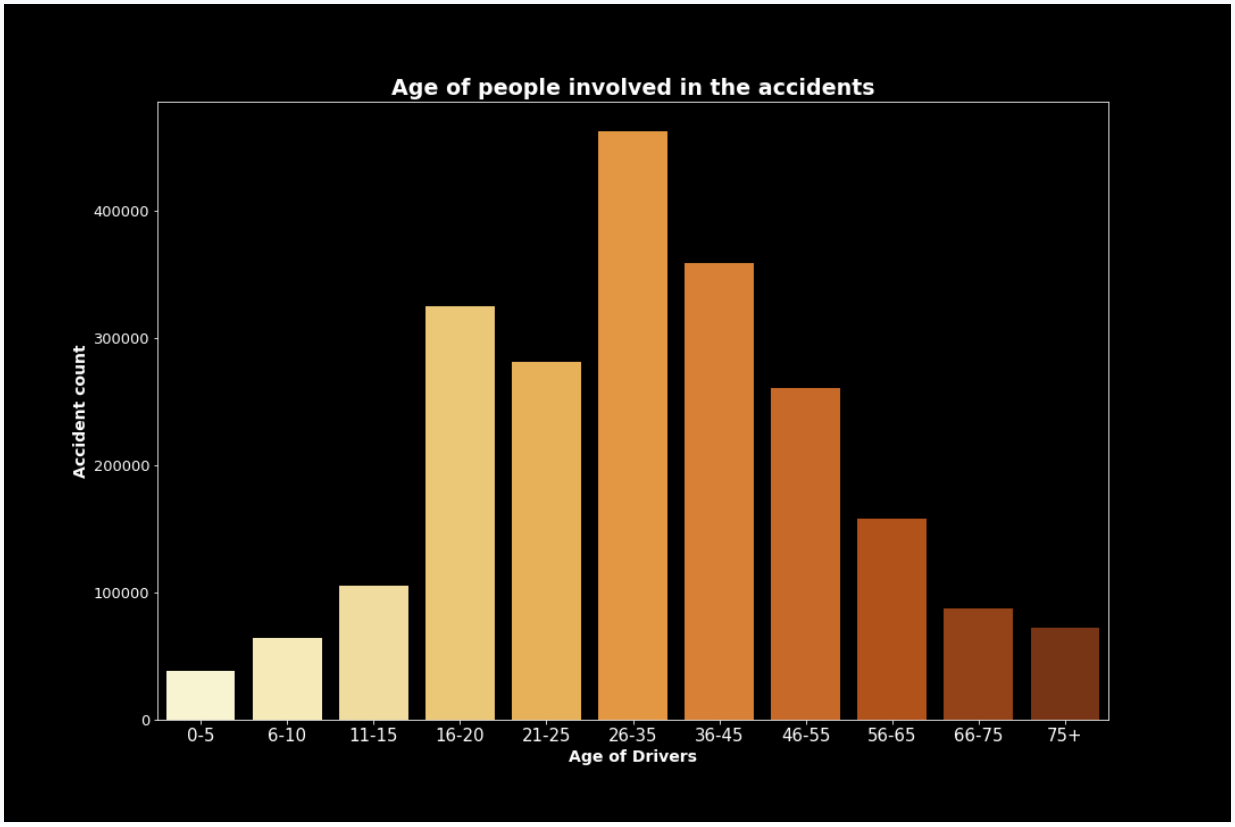
Vụ tai nạn xảy ra ở mức cao nhất là khoảng (Afternoon) 12 đến 15 giờ, đây là thời điểm việc lưu thông ở các tuyến đường khá là đông đúc và thời tiết nóng bức làm cho mọi người lưu thông xe cộ có suy nghĩ đi nhanh nhất có thể (Tốc độ tăng cao). Và tiếp theo là khoảng (After Work Rush) từ 16 đến 18 giờ, đây là khoảng thời gian kẹt xe ở các tuyến đường xảy ra thường xuyên khi đây là thời điểm tan làm của mọi người và tan học của các em học sinh.

Tại hai thời gian Late Night và Night thì đây là thời gian mọi người đang yên giấc nên 2 cột này thể hiện thấp nhất là hoàn toàn đúng.

Thời gian	Tên gọi
[0:4]	Late Night
[5:8)	Morning Rush
[8:11]	Day
[12:15]	Afternoon
[16:18]	After Work Rush
[19:21]	Evening
[21:23]	Night

6. TUỔI LIÊN QUAN ĐẾN VỤ TAI NẠN

Entrée [55]:
```
plt.style.use('dark_background')
plt.figure(figsize=(15,10))
sns.barplot(Age_Band.index,Age_Band.values, order=AgeBand, palette='YlOrBr')
plt.title("Age of people involved in the accidents",fontsize=19,fontweight="bold")
plt.xlabel("Age of Drivers", fontsize=14, fontweight="bold")
plt.ylabel("\nAccident count", fontsize=14, fontweight="bold")
plt.xticks(fontsize=15)
plt.yticks(fontsize=13)
plt.savefig('Age of people involved in the accidents.png')
plt.show()
```



Đây là sự thật rất thú vị về tập dữ liệu này. Hầu hết các tài xế trong độ tuổi khoảng 25 đến 35 có liên quan đến vụ tai nạn. và từ 36 đến 45 tuổi đứng vị trí thứ 2 cho thấy hầu như ở các vụ tai nạn đang nghiêng về phía tuổi trung niên nhiều hơn. So với ta dự đoán ban đầu là ở tuổi thanh niên, nhưng đồ thị cho biết ở mức 16 đến 20 tuổi chỉ đứng vị trí thứ 3.

V. CHUẨN HOÁ DỮ LIỆU

Đầu tiên Sử dụng phương pháp Join (nối) để kết hợp các file tai nạn và phương tiện vì chúng có cùng khóa chính Accident_Index.

```
Entrée [57]: df_ML=accidents.join(vehicles, how='outer')

Entrée [58]: df=df_ML
```

Khi Nối xong ta cập nhật lại thành df, sau đấy ta dùng thuật toán **Ordinal encoding:** Để chuyển dữ liệu thành dạng số:

Ví dụ: Sex_of_Driver (Male or Female) sẽ được chuyển sang dạng số là (1 và 0)

```
Entrée [73]: from sklearn.preprocessing import OrdinalEncoder
ord_enc = OrdinalEncoder()
df["Accident_Severity"] = ord_enc.fit_transform(df[["Accident_Severity"]])
df["Day_of_Week"] = ord_enc.fit_transform(df[["Day_of_Week"]])
df["Local_Authority_(Highway)"] = ord_enc.fit_transform(df[["Local_Authority_(Highway)"]])
df["Road_Type"] = ord_enc.fit_transform(df[["Road_Type"]])
df["Light_Conditions"] = ord_enc.fit_transform(df[["Light_Conditions"]])
df["Weather_Conditions"] = ord_enc.fit_transform(df[["Weather_Conditions"]])
df["Road_Surface_Conditions"] = ord_enc.fit_transform(df[["Road_Surface_Conditions"]])
df["Sex_of_Driver"] = ord_enc.fit_transform(df[["Sex_of_Driver"]])
df["Urban_or_Rural_Area"] = ord_enc.fit_transform(df[["Urban_or_Rural_Area"]])
df["Was_Vehicle_Left_Hand_Drive?"] = ord_enc.fit_transform(df[["Was_Vehicle_Left_Hand_Drive?"]])
```

Dữ liệu đã chuẩn hoá thành dạng float64 với các con số cụ thể.

```
Entrée [74]: df.info()
```

```
 0   Longitude                                    float64
 1   Latitude                                     float64
 2   Police_Force                                 float64
 3   Accident_Severity                            float64
 4   Number_of_Vehicles                           float64
 5   Number_of_Casualties                         float64
 6   Day_of_Week                                  float64
 7   Local_Authority_(District)                   float64
 8   Local_Authority_(Highway)                    float64
 9   1st_Road_Class                               float64
 10  1st_Road_Number                              float64
 11  Road_Type                                    float64
 12  Speed_limit                                  float64
 13  Junction_Detail                              float64
 14  Junction_Control                             float64
 15  2nd_Road_Class                               float64
 16  2nd_Road_Number                              float64
 17  Pedestrian_Crossing-Human_Control            float64
 18  Pedestrian_Crossing-Physical_Facilities      float64
```

Entrée [77]:
```
df[col]
```

Out[77]:

Accident_Index	Day_of_Week	Road_Type	Speed_limit	Light_Conditions	Weather_Conditions	Road_Surface_Conditions	Urban_or_Rural_Area	Did_Police_Offic
200501BS00001	5.0	1.0	30.0	4.0	6.0	1.0	2.0	
200501BS00002	6.0	4.0	30.0	1.0	1.0	0.0	2.0	
200501BS00003	4.0	1.0	30.0	1.0	1.0	0.0	2.0	
200501BS00003	4.0	1.0	30.0	1.0	1.0	0.0	2.0	
200501BS00004	0.0	1.0	30.0	4.0	1.0	0.0	2.0	
...	...	...	...	...	...	...	...	...
2014984138414	6.0	1.0	60.0	3.0	6.0	1.0	1.0	
2014984138414	6.0	1.0	60.0	3.0	6.0	1.0	1.0	
2014984138414	6.0	1.0	60.0	3.0	6.0	1.0	1.0	
2014984139614	6.0	1.0	60.0	4.0	1.0	1.0	1.0	
2014984139614	6.0	1.0	60.0	4.0	1.0	1.0	1.0	

2980888 rows × 14 columns

Tiếp theo ta sẽ lựa chọn các trường (tính năng) để phục vụ cho việc dự đoán (Ở mục **2. LỰA CHỌN BIẾN HỒI QUY)**

Phương pháp Scaler:

Với việc các dữ liệu còn những cột chứa các giá trị số quá lớn, sẽ làm sai lệch một phần nào đó vào việc dự đoán (ví dụ Speed_limit và Age_of_driven ...)

Entrée [93]:
```
from numpy import asarray
from sklearn.preprocessing import MinMaxScaler
# define min max scaler
scaler = MinMaxScaler()
# transform data
X = scaler.fit_transform(X)
print(X)
```

```
[[0.8        1.         0.11111111 ... 0.45918367 0.00909091 0.02964119]
 [0.2        0.5        0.11111111 ... 0.30612245 0.11818182 0.01329053]
 [0.2        0.83333333 0.11111111 ... 0.71428571 0.         0.01557062]
 ...
 [0.2        0.33333333 0.11111111 ... 0.51020408 0.         0.09357374]
 [0.2        0.33333333 0.11111111 ... 0.39795918 0.03636364 0.01409056]
 [0.2        0.33333333 0.11111111 ... 0.54081633 0.         0.01385055]]
```

Vì thế việc sử dụng thuật toán MinMaxScaler theo công thức:

Giá trị được normalize theo công thức sau:

PLAINTEXT
```
y = (x - min) / (max - min)
```

Theo công thức này từ đây các giá trị ở mỗi cột sẽ quy về các giá trị từ (0 đến 1)

Out[95]:

	Road_Type	Speed_limit	Weather_Conditions	Day_of_Week	Urban_or_Rural_Area	Did_Police_Officer_Attend_Scene_of_Accident	Vehicle_Type	Sex_of_D
0	0.8	1.000000	0.111111	0.000000	0.5	0.0	0.082474	
1	0.2	0.500000	0.111111	0.666667	1.0	0.0	0.082474	
2	0.2	0.833333	0.111111	0.166667	0.5	0.0	0.082474	
3	0.2	0.333333	0.111111	0.333333	1.0	0.0	0.082474	
4	0.2	0.333333	0.111111	0.500000	1.0	0.0	0.082474	
...	...	...	...	...	...	...	...	...
499995	0.2	0.333333	0.111111	0.000000	1.0	0.0	0.082474	
499996	0.2	0.833333	0.111111	0.000000	0.5	0.0	0.082474	
499997	0.2	0.333333	0.111111	0.000000	1.0	0.5	0.103093	
499998	0.2	0.333333	0.111111	0.166667	1.0	0.0	0.000000	
499999	0.2	0.333333	0.111111	0.000000	1.0	0.0	0.082474	

500000 rows × 11 columns

VI. XÂY DỰNG MODEL MÁY HỌC (DỰ ĐOÁN)

Tách dữ liệu thành dữ liệu Train và Test

X là dữ liệu đầu vào và Y là nhãn lớp.

20% dữ liệu dành cho test và 80% dành cho train.

```
Entrée [96]: from sklearn.model_selection import train_test_split
             X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=99)

Entrée [97]: X_train.shape
    Out[97]: (400000, 11)

Entrée [98]: X_test.shape
    Out[98]: (100000, 11)

Entrée [99]: y_train.shape
    Out[99]: (400000,)

Entrée [100]: y_test.shape
   Out[100]: (100000,)
```

Việc tách dữ liệu sẽ giúp cho máy học có tính công bằng hơn khi dự đoán, nếu ta train hết 100% thì sẽ không có dữ liệu để ta dự đoán và nếu Training quá thấp thì việc dự đoán sẽ không chính xác .

Thuật toán và so sánh

1. **Decision Tree**
2. **Random Forest**
3. **Logistic Regression**

1. Logistic Regression

Logistic Regression

Entrée [108]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(multi_class='multinomial')
# Fit the model on the trainng data.
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

Evaluating Model

Entrée [109]:
```
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=y_pred)
print("Accuracy", round(accuracy_score(y_pred, y_test)*100,2))
print(sk_report)
```

```
Accuracy 86.4
              precision    recall  f1-score   support

         0.0   0.000000  0.000000  0.000000      1263
         1.0   0.000000  0.000000  0.000000     12340
         2.0   0.863969  0.999988  0.927016     86397

    accuracy                       0.863960    100000
   macro avg   0.287990  0.333329  0.309005    100000
weighted avg   0.746443  0.863960  0.800914    100000
```

Như chúng ta có thể thấy rằng hồi quy logistic đã làm khá tốt về số lượng. Khi nhìn về mặt dự đoán mức độ 2 thì việc bỏ sót nạn nhân dường như là 0%, còn về phần dự đoán thì chiếm 86,4% cho thấy nếu xét về 100.000 người bị tai nạn thì hầu như dự đoán 100% số người bị thương ở mức 2, còn 13,6% là dự đoán sai khi khả năng cao là người này bị ở mức độ 1 và 0 nhưng thuật toán không ra được kết quả.

2. Decision Tree

Decision Tree

Entrée [110]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred2 = dt.predict(X_test)
```

Đánh giá mô hình

Evaluating Model

Entrée [111]:
```
acc_decision_tree1 = round(accuracy_score(y_pred2, y_test) * 100, 2)
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=y_pred2)
print("Accuracy", acc_decision_tree1)
print(sk_report)
```

```
Accuracy 76.67
              precision    recall  f1-score   support

         0.0   0.034623  0.040380  0.037281      1263
         1.0   0.157931  0.166288  0.162002     12340
         2.0   0.871805  0.863097  0.867429     86397

    accuracy                       0.766720    100000
   macro avg   0.354787  0.356589  0.355571    100000
weighted avg   0.773140  0.766720  0.769895    100000
```

Entrée [112]:
```
pd.DataFrame(y_pred2).value_counts()
```

Out[112]:
```
2.0    85534
1.0    12993
0.0     1473
dtype: int64
```

Entrée [113]:
```
y_test.value_counts()
```

Out[113]:
```
2.0    86397
1.0    12340
0.0     1263
Name: Accident_Severity, dtype: int64
```

Xét so với thuật toán hồi quy logistic ta nhận ra rằng Cây quyết định làm tốt hơn nhiều. Thuật toán hồi quy Logistic chỉ dự đoán hầu như ở mức độ tai nạn nhẹ (Unserious) nhưng so với Cây quyết định, nó dự đoán được thương tích nghiêm trọng (Serious) và cực kì nghiêm trọng (Extremely Serious) đây là một tín hiệu tích cực. So về điểm thì mức độ dự đoán chính xác thấp hơn so với thuật toán khác vì thuật toán khác dự đoán phần lớn tai nạn nhẹ và những con số đó thực sự cao trong bộ dữ liệu.

3. Random Forest

Thuật toán Random Forest là một sự nâng cấp của cây quyết định. Nó dường như khắc phục nhược điểm của cây quyết định. Giúp cải thiện rất tốt khả năng dự báo của cây quyết định

Đặc điểm của cây quyết định là không có tính ổn định: Một thay đổi nhỏ trong bộ mẫu có thể làm thay đổi lớn đến kết quả rẽ nhánh của cây => Phương sai lớn

Random Forest

Entrée [101]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=200)
random_forest.fit(X_train,y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_test, y_test)
acc_random_forest1 = round(random_forest.score(X_test, y_test) * 100, 2)
```

Entrée [102]:
```
pd.DataFrame({"y":y_test,"y_predict":Y_pred}).head(5)
```

Out[102]:

Accident_Index	y	y_predict
200714A093407	2.0	2.0
201201XD80975	2.0	2.0
2010210003697	2.0	2.0
201020S017650	0.0	2.0
2009130904609	2.0	2.0

Dự đoán ở mức độ 2 (Không nghiêm trọng) chiếm 84,88% , recall là 97,4% cho thấy việc dự đoán của mô hình khá tốt và hầu như việc bỏ sót của những người ở mức độ 2 rất thấp, chỉ chiếm 2,6%

Đánh giá mô hình

```
Entrée [106]: sk_report = classification_report(
                  digits=6,
                  y_true=y_test,
                  y_pred=Y_pred)
              print("Accuracy" , acc_random_forest1)
              print(sk_report)
```

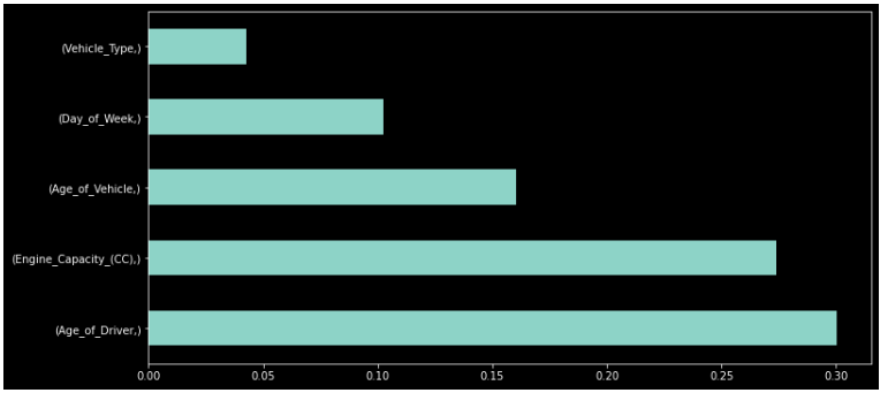
```
Accuracy 84.88
              precision    recall  f1-score   support

         0.0   0.053030  0.005542  0.010036      1263
         1.0   0.241821  0.058104  0.093695     12340
         2.0   0.868415  0.974015  0.918189     86397

    accuracy                       0.848760    100000
   macro avg   0.387755  0.345887  0.340640    100000
weighted avg   0.780795  0.848760  0.804976    100000
```

Random Forest cũng như cây quyết định có thể dự đoán được 3 mức độ khác nhau, làm cho mô hình càng thêm tính thực tế hơn, Random Forest còn cho thấy sự cải thiện về mức độ điểm khi đánh giá 3 mức độ cao hơn so với cây quyết định khá nhiều, có thể nói Random Forest là mô hình có dự đoán khá tốt về cả 3 mức độ.

Các thuộc tính ảnh hưởng mạnh đến dự đoán

```
Entrée [108]: plt.figure(figsize=(12,6))
              feat_importances = pd.Series(random_forest.feature_importances_, index=X.columns)
              feat_importances.nlargest(5).plot(kind='barh')
```

```
Out[108]: <AxesSubplot:>
```



Đồ thị nêu rõ cho ta thấy sự ảnh hưởng của (**Age_of_Driven)** là tuổi của người lái xe, và tiếp theo là động cơ của xe, 2 cột này được mô hình đánh giá cao nhất và được cho là quan trọng nhất đối với mô hình **Random Forest** này.

4. Hyper Parameter

Entrée [114]:
```
n_estimators = [5,20,50,100] # number of trees in the random forest
max_features = ['auto', 'sqrt'] # number of features in consideration at every split
max_depth = [4,5,6,8] # maximum number of levels allowed in each decision tree
bootstrap = [True, False] # method used to sample data points

random_grid = {'n_estimators': n_estimators,

'max_features': max_features,

'max_depth': max_depth,

'bootstrap': bootstrap}
```

Entrée [115]:
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
```

Entrée [116]:
```
from sklearn.model_selection import RandomizedSearchCV
rf_random = RandomizedSearchCV(estimator = rf,param_distributions = random_grid,
               n_iter = 100, cv = 5, verbose=2, random_state=35, n_jobs = -1)
```

Entrée [117]:
```
rf_random.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 64 candidates, totalling 320 fits
```

Out[117]:
```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_iter=100,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [4, 5, 6, 8],
                                        'max_features': ['auto', 'sqrt'],
                                        'n_estimators': [5, 20, 50, 100]},
                   random_state=35, verbose=2)
```

Tìm ra Hyper Parameter tốt nhất cho model:

Entrée [118]:
```
print ('Random grid: ', random_grid, '\n')
# print the best parameters
print ('Best Parameters: ', rf_random.best_params_, ' \n')
```

```
Random grid:  {'n_estimators': [5, 20, 50, 100], 'max_features': ['auto', 'sqrt'], 'max_depth': [4, 5, 6, 8], 'bootstrap': [Tru
e, False]}

Best Parameters:  {'n_estimators': 100, 'max_features': 'auto', 'max_depth': 8, 'bootstrap': True}
```

Chạy model **Random Forest** với **Hyper Parameter**

Entrée [119]:
```
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=100,max_features='auto',max_depth=8,bootstrap=True)
random_forest.fit(X_train,y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_test, y_test)
acc_random_forest1 = round(random_forest.score(X_test, y_test) * 100, 2)
```

Entrée [120]:
```
sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=Y_pred)
print("Accuracy" , acc_random_forest1)
print(sk_report)
```

```
Accuracy 86.4
              precision    recall  f1-score   support

         0.0   0.000000  0.000000  0.000000      1263
         1.0   0.416667  0.000405  0.000810     12340
         2.0   0.864014  0.999931  0.927017     86397

    accuracy                       0.863960    100000
   macro avg   0.426893  0.333445  0.309275    100000
weighted avg   0.797899  0.863960  0.801014    100000
```

Hyper Parameter có vẻ đã cải thiện được điểm dự đoán của Random Forest 86.4%

VII. TỔNG KẾT

THUẬT TOÁN	ĐIỂM
Logistic Regression	**86.4**
Decision Tree	**76.67**
Random Forest	**84.88**
Hyper Parameter (Random Forest)	**86.4**

Dự án này nhằm mục đích sử dụng các kỹ thuật phân loại Học máy để dự đoán mức độ nghiêm trọng của tai nạn tại bất kỳ vị trí cụ thể nào.

Học máy đã cho phép chúng tôi phân tích dữ liệu có ý nghĩa để cung cấp các giải pháp với độ chính xác cao hơn so với con người.

Với nhiều nguồn lực hơn, dự đoán và cảnh báo liên tục có thể được gửi đến cảnh sát ở mọi địa điểm trong khoảng thời gian thường xuyên để thực hiện các biện pháp ngăn chặn.