

I. 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

```
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 đường: ướt, tuyết, đóng băng, lũ
Speed Limit : 60 mph , 70 mph Giới hạn tốc độ: ~96.6km/h, ~112.6km/h
<u>Output</u>
Accident Severity : 0 = Extremely serious, 1 = Serious, 2 = unserious

2. LỰA CHỌN TÍNH NĂNG

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 đường 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 chiếm giữ, 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 kỳ nghỉ. 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, 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 mình đã 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 – no 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	Wednesday
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_Manoevre  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

II. TRỰC QUAN HOÁ DỮ LIỆU

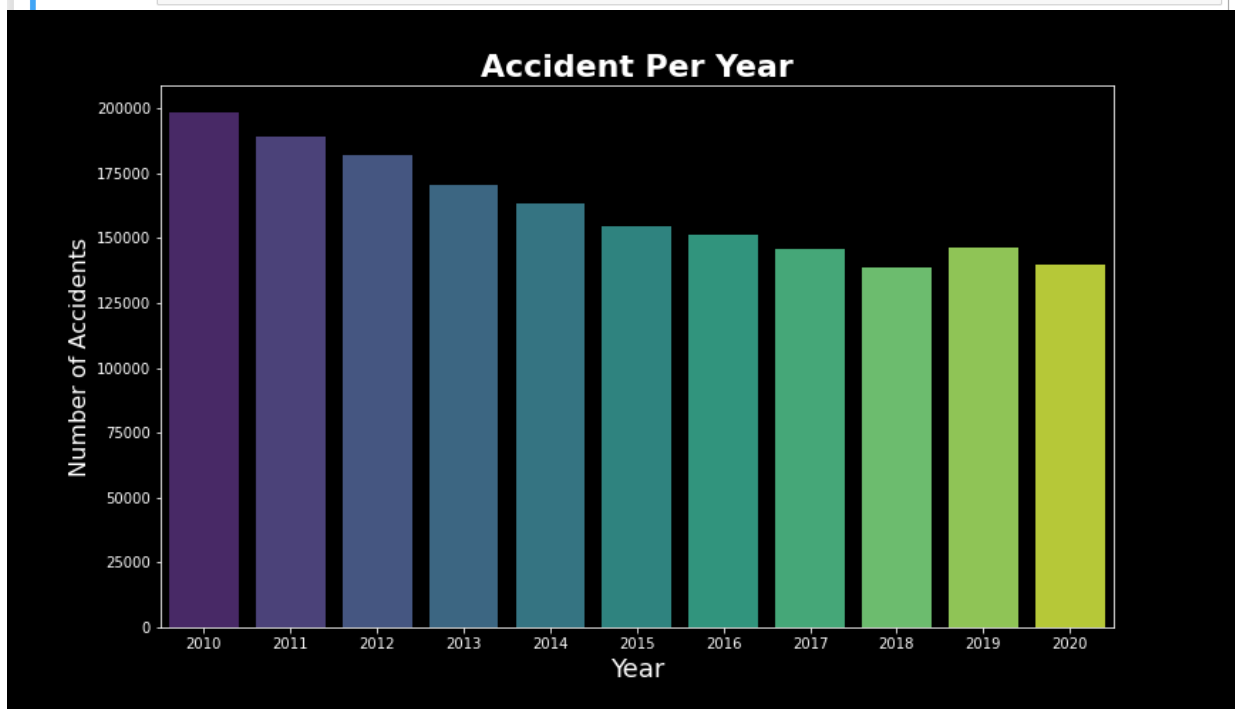
Điều đầu tiên chúng ta có thể làm là tìm hiểu về thời gian xảy ra tai nạn để đề phòng và độ tuổi của một số tài xế có liên quan đến vụ tai nạn. Tìm hiểu xem mức độ tai nạn của các năm có sự thay đổi theo hướng nào.

- 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()
```

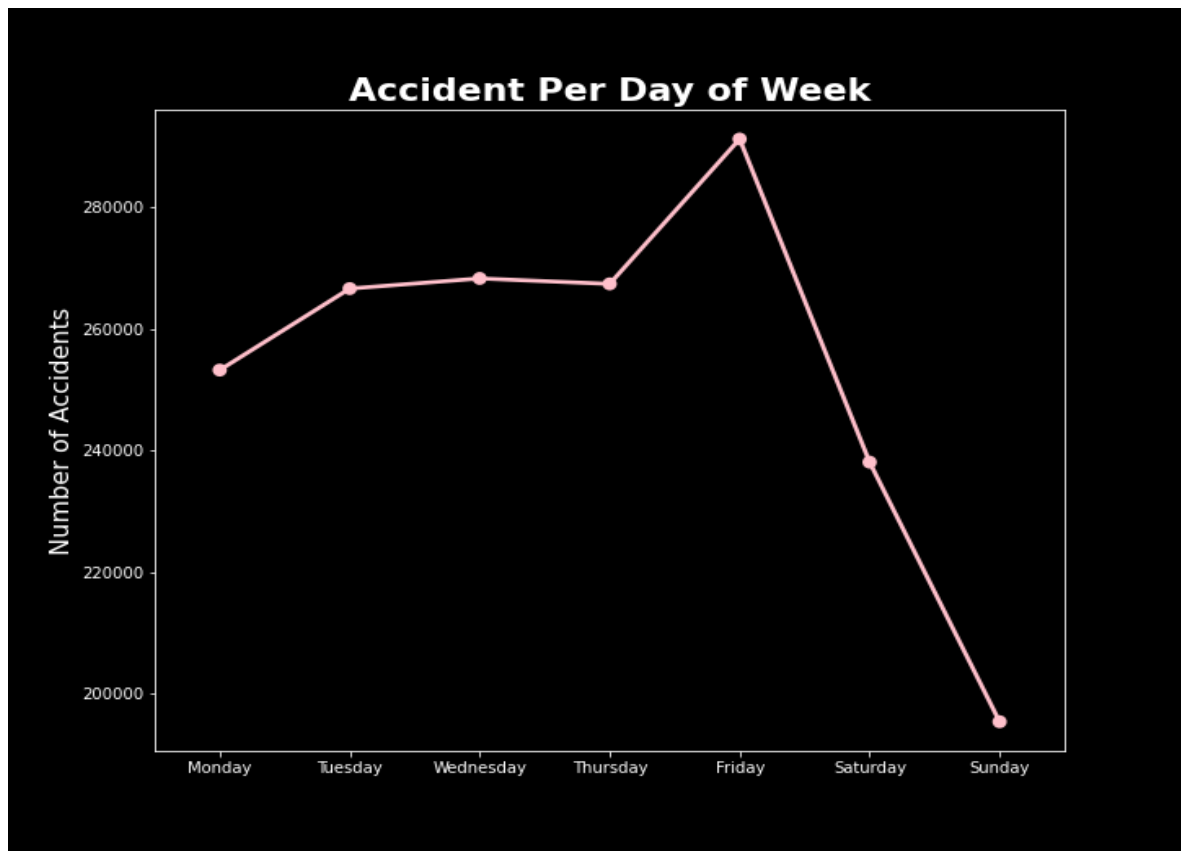
```
Entrée [119]: plt.style.use('dark_background')
plt.figure(figsize=(12,7))
sns.barplot(AccidentsPerYear.index,AccidentsPerYear.values,palette="viridis")
plt.title("Accident Per Year",fontsize=22,fontweight="bold")
plt.xlabel("Year",fontsize=18)
plt.ylabel("Number of Accidents",fontsize=16)
plt.savefig('ACCIDENT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

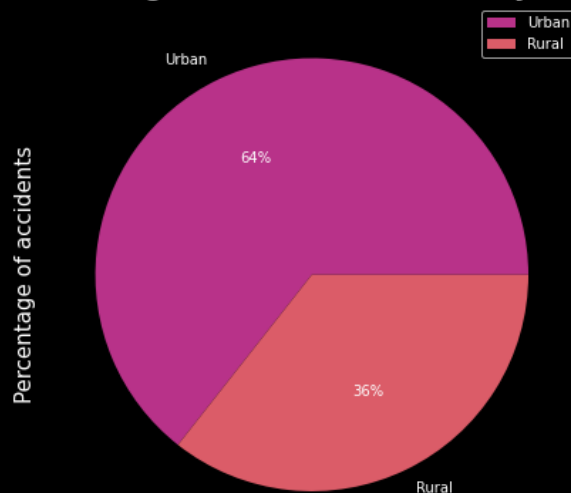
        PerRural=RuralArea*1.0/SumOfArea*1.0*100

        PerOther=OtherArea*1.0/SumOfArea*1.0*100

        print("Percentage of accidents occur in urban areas is {:.0f}%".format(PerUrban))
        print("Percentage of accidents occur in rural areas is {:.0f}%".format(PerRural))
        print("Percentage of accidents occur in other areas is {:.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='%0.0f%%')
        plt.ylabel('Percentage of accidents',fontsize=15)
        plt.title("Percentage of accidents occurred by Area",fontsize=18,fontweight="bold")
        plt.legend()
        plt.savefig('PERCENTAGE OF ACCIDENTS.png')
        plt.show()

```

Percentage of accidents occurred by Area



Urban là đô thị, Rural là nông thôn

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

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

```

Entrée [29]: #setups for adding frequencies to visualizations
dfttotal= float(len(accidents))
nstotal= float(len(unserious_accident))
settotal= float(len(serious_accident))
esettotal=float(len(extreme_serious_accident))

Entrée [30]: spl_order=[30,40,50,60,70,80]
spl_order2=[30,40,50,60,70,80]
spl_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=spl_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=spl_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/settotal*100),
             ha="center",fontSize=12)
ax3 = sns.countplot("Accident_Severity", hue="Speed_limit", hue_order=spl_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/esettotal*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
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

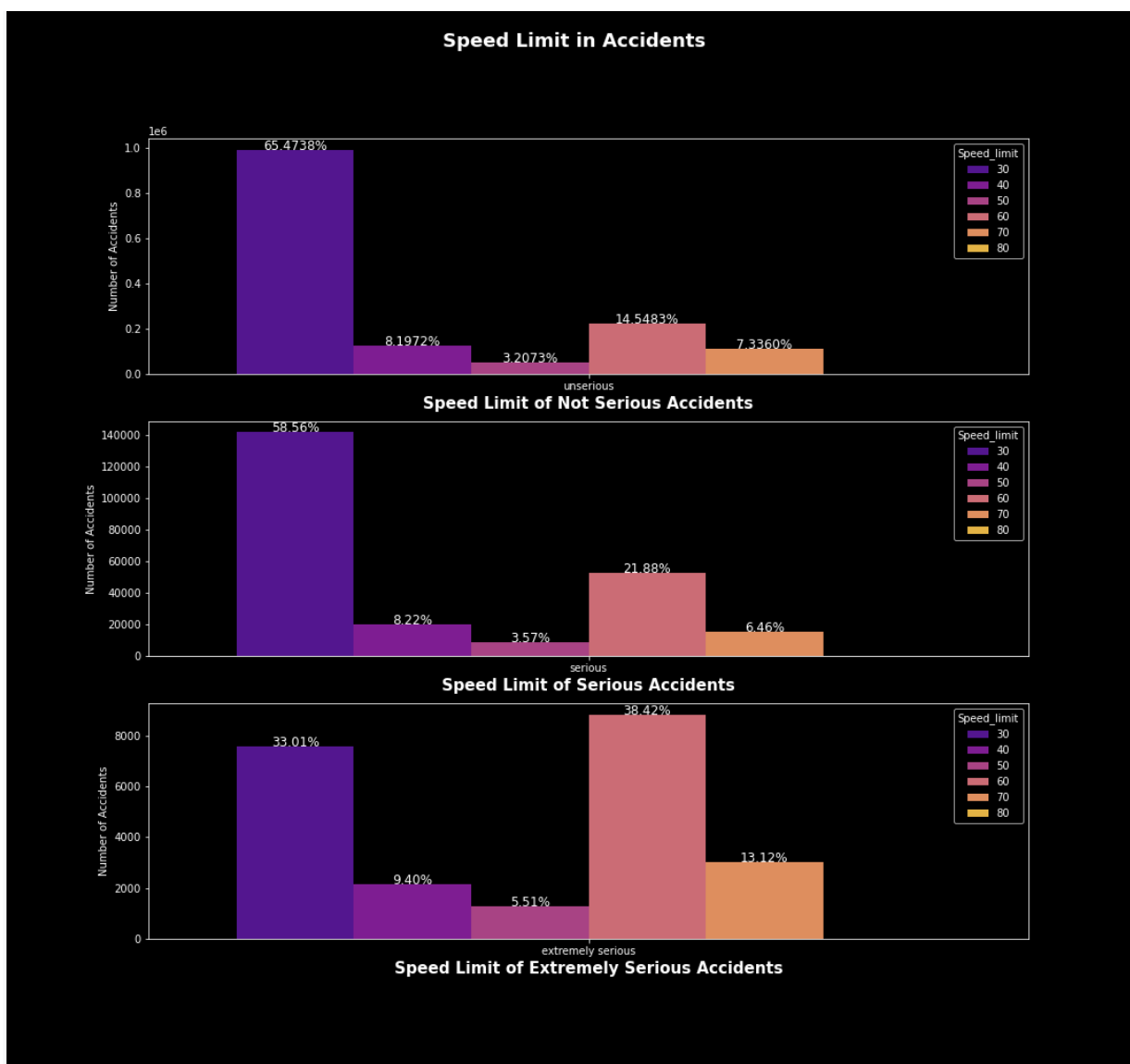
(Accident Severity: 0 = Extremely serious

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 nổi tiếp là ở tốc độ 60 với tỷ lệ là 14,5% cho thấy rằng tỉ trọng ở mức không nghiêm trọng nó 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.

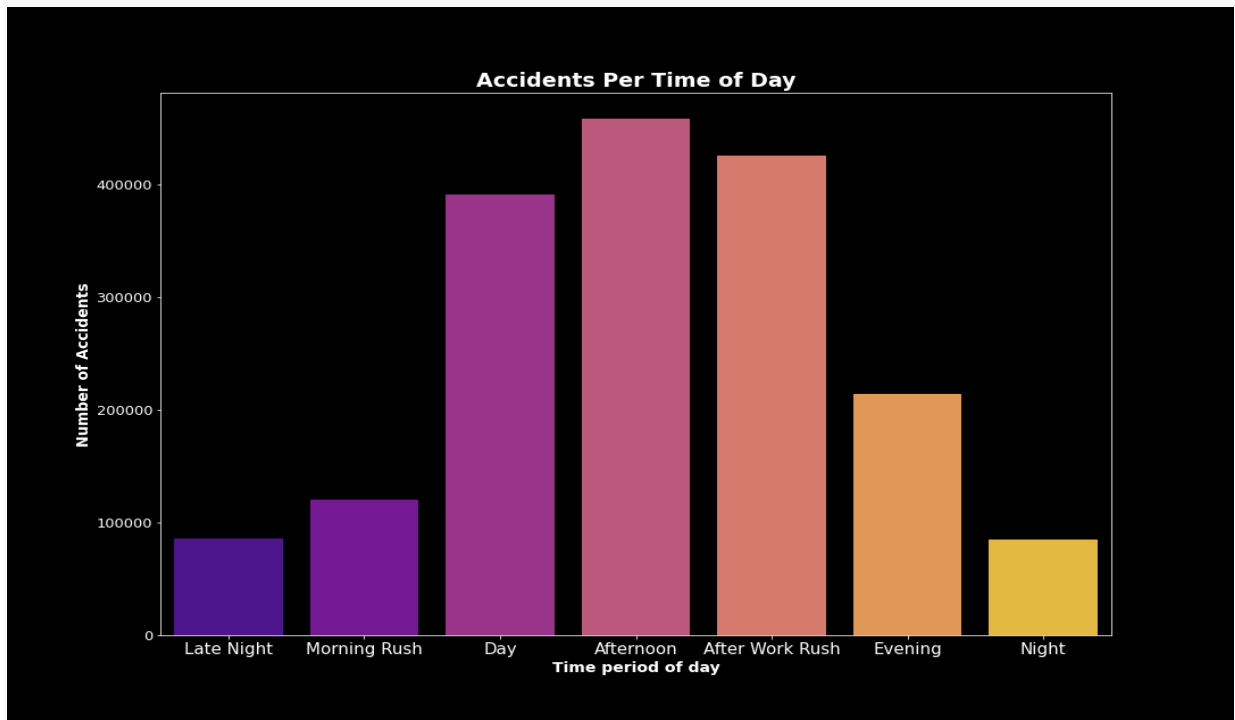
Tiếp theo, đồ thị thứ 2 (Serious Accidents) mức độ nghiêm trọng trong tai nạn, đa số các vụ tai nạn ở giới hạn tốc độ 30 vẫn khá cao nhưng dường như bị mất ưu thế hơn so với đồ thị đầu, còn ở tốc độ 60 thì đang bắt đầu chiếm tỉ lệ cao hơn có thể thấy rằng các vụ tai nạn có mức độ nghiêm trọng đang dần 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, tốc độ 80 tăng lên mức 2.0, 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

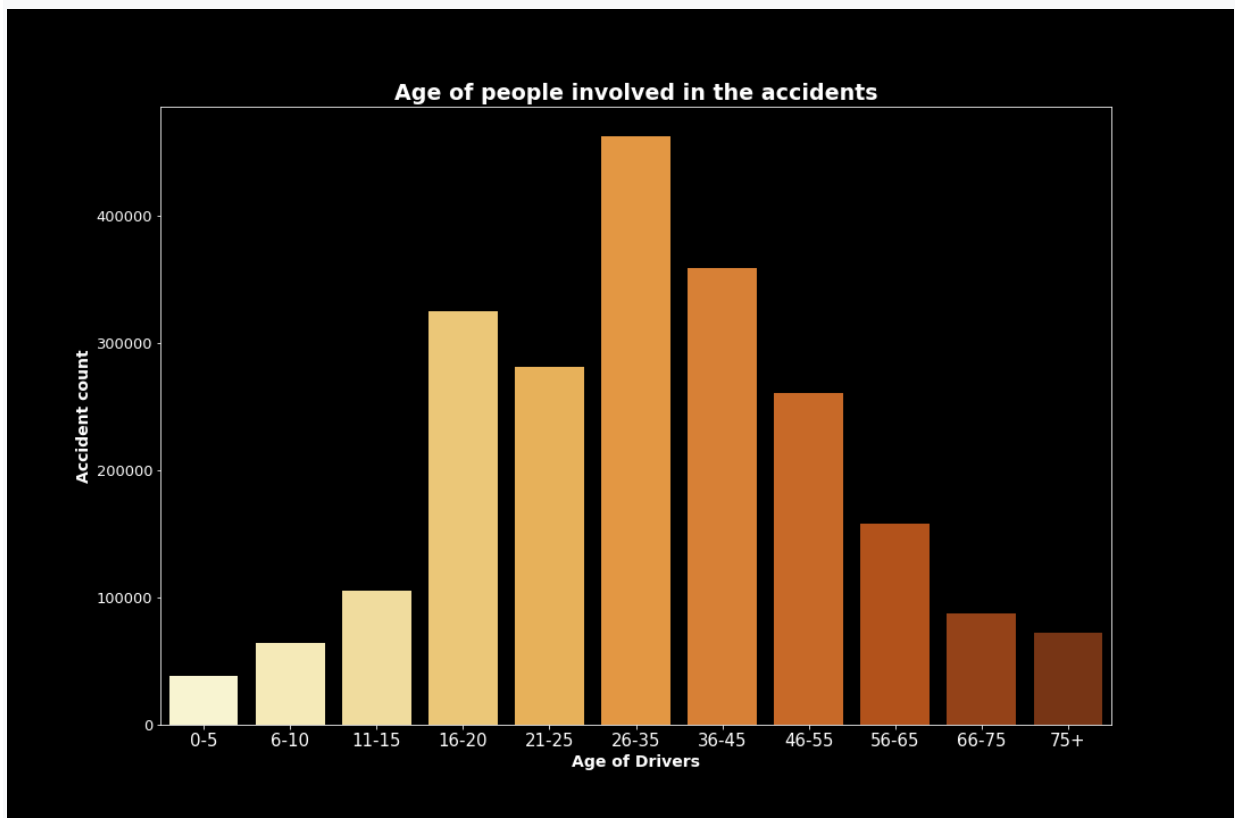
xe ở các tuyến đường xảy ra thường xuyên khi đây là thời điểm tan làm của mọi người và tan học của các em học sinh.

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.

III. CHUẨN HOÁ DỮ LIỆU

Đầu tiên Sử dụng phương pháp Join (nối) để kết hợp các tệp 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 đây 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
19   Light_Conditions    float64
```

Entrée [77]: `df[col]`

Out[77]:

	Day_of_Week	Road_Type	Speed_limit	Light_Conditions	Weather_Conditions	Road_Surface_Conditions	Urban_or_Rural_Area	Did_Police_Office
Accident_Index								
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 TÍNH NĂNG)

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      0.5      0.      ... 0.22680412 0.07476636 0.01895038]
 [0.2      0.33333333 0.11111111 ... 0.17525773 0.04672897 0.01123022]
 [0.      0.33333333 0.11111111 ... 0.69072165 0.01869159 0.0198304 ]
 ...
 [0.8      1.      0.11111111 ... 0.21649485 0.03738318 0.00124002]
 [0.2      0.33333333 0.11111111 ... 0.20618557 0.08411215 0.01387028]
 [0.2      0.83333333 0.11111111 ... 0.32989691 0.03738318 0.01411028]]
```

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

Giá trị được normalize theo công thức sau:

$$y = (x - \min) / (\max - \min)$$

PLAINTEXT

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

Entrée [95]: `X = pd.DataFrame(X, columns = [col])`

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	0.500000	0.000000	0.166667	1.0	0.0	0.082474	
1	0.2	0.333333	0.111111	0.000000	1.0	0.0	0.082474	
2	0.0	0.333333	0.111111	0.000000	0.5	0.0	0.082474	
3	0.2	0.833333	0.111111	0.666667	0.5	0.0	0.082474	
4	0.8	0.500000	0.111111	0.333333	1.0	0.0	0.185567	
...	
499995	0.2	0.333333	0.111111	0.333333	1.0	0.0	0.185567	
499996	0.2	0.333333	0.111111	0.000000	1.0	0.0	0.082474	
499997	0.8	1.000000	0.111111	0.166667	0.5	0.0	0.020619	
499998	0.2	0.333333	0.111111	0.500000	1.0	0.0	0.082474	
499999	0.2	0.833333	0.111111	0.833333	0.5	0.0	0.195876	

500000 rows × 11 columns

IV. 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 [109]: from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(multi_class='multinomial')
# Fit the model on the training data.
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

Evaluating Model

```
Entrée [110]: 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	86.31				
		precision	recall	f1-score	support
	0.0	0.000000	0.000000	0.000000	1238
	1.0	0.000000	0.000000	0.000000	12452
	2.0	0.863100	1.000000	0.926520	86310
accuracy				0.863100	100000
macro avg		0.287700	0.333333	0.308840	100000
weighted avg		0.744942	0.863100	0.799680	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 86,31% 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 13,69% 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 [111]: `from sklearn.tree import DecisionTreeClassifier`

Entrée [112]: `dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred2 = dt.predict(X_test)`

3. Evaluating Model

Entrée [113]: `acc_decision_tree1 = round(accuracy_score(y_pred2, y_test) * 100, 2)
sk_report = ClassificationReport(
 digits=6,
 y_true=y_test,
 y_pred=y_pred2)
print("Accuracy", acc_decision_tree1)
print(sk_report)`

```
Accuracy 76.43
      precision    recall  f1-score   support

      0.0   0.039947   0.048465   0.043796       1238
      1.0   0.157970   0.167764   0.162720      12452
      2.0   0.871039   0.860584   0.865780       86310

 accuracy          0.764260      100000
  macro avg   0.356319   0.358938   0.357432      100000
 weighted avg   0.771959   0.764260   0.768059      100000
```

Entrée [114]: `pd.DataFrame(y_pred2).value_counts()`

Out[114]: `2.0 85274
1.0 13224
0.0 1502
dtype: int64`

Entrée [115]: `y_test.value_counts()`

Out[115]: `2.0 86310
1.0 12452
0.0 1238
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.

4. 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 là khả năng tốt 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.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]:
```

	y	y_predict
Accident_Index		
20064100A0011	2.0	0.0
201414A175814	2.0	2.0
200801PL60757	1.0	1.0
200643W044016	2.0	2.0
2005350409305	2.0	2.0

Dự đoán ở mức độ 2 (Không nghiêm trọng) chiếm 86,7% , recall là 97,5% 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 2,5%

Đánh giá mô hình

```
Entrée [107]: sk_report = classification_report(
    digits=6,
    y_true=y_test,
    y_pred=Y_pred)
print("Accuracy" , acc_random_forest1)
print(sk_report)
```

```
Accuracy 84.89
```

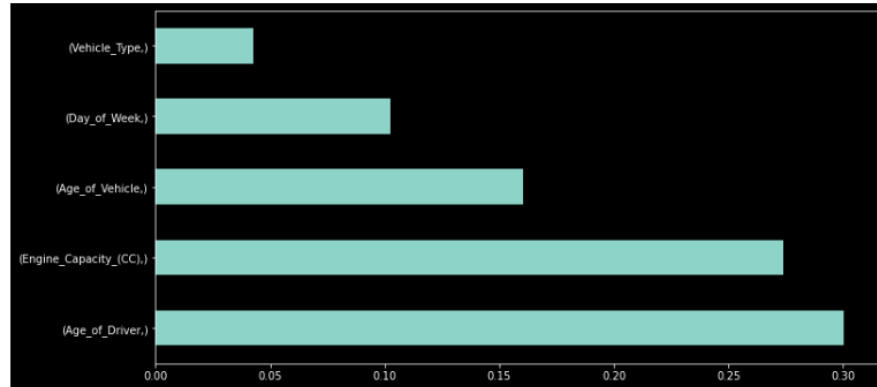
	precision	recall	f1-score	support
0.0	0.038961	0.004847	0.008621	1238
1.0	0.244604	0.054610	0.089286	12452
2.0	0.867461	0.975565	0.918343	86310
accuracy			0.848870	100000
macro avg	0.383676	0.345007	0.338750	100000
weighted avg	0.779646	0.848870	0.803846	100000

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_Driver**) 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.

V. TỔNG KẾT

THUẬT TOÁN	ĐIỂM
Logistic Regression	86.31
Decision Tree	76.43
Random Forest	84.89

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.