

BỘ MÔN HỆ THỐNG THÔNG TIN – KHOA CÔNG NGHỆ THÔNG TIN
ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH, ĐẠI HỌC QUỐC GIA TP HCM

PHÂN TÍCH DỮ LIỆU ỨNG DỤNG



4.0

Sinh viên thực hiện: 18120636 – Trần Ngọc Tuấn

18120213 – Võ Đại Nam

18120217 – Nguyễn Trần Ái Nguyên

18120437 – Ngô Thị Thùy Linh

GV phụ trách: Hồ Bảo Quốc – Hồ Thị Hoàng Vy



BẢNG THÔNG TIN CHI TIẾT NHÓM

Mã nhóm:	PTDLUD_06			
Số lượng:	4			
MSSV	Họ tên	Email	Điện thoại	Nhóm trưởng
18120636	Trần Ngọc Tuấn	X
18120213	Võ Đại Nam			
18120217	Nguyễn Trần Ái Nguyên			
18120437	Ngô Thị Thùy Linh			

Bảng phân công & đánh giá hoàn thành công việc		
Công việc thực hiện	Người thực hiện	Mức độ hoàn thành
Tìm hiểu vấn đề, Mô tả trường dữ liệu, tiền xử lý dữ liệu, xây dựng mô hình, training và đánh giá mô hình liên quan đến dự đoán lương của 1 nhân viên dựa vào công việc, level tổ chức, tình trạng hôn nhân, giới tính, số năm làm việc, Rate lương tương ứng.	Trần Ngọc Tuấn	100%
Tìm hiểu vấn đề, Mô tả trường dữ liệu, tiền xử lý dữ liệu, xây dựng mô hình, đánh giá mô hình liên quan đến dự đoán doanh số sản phẩm bán ra ở năm tiếp theo	Võ Đại Nam	100%
Tìm hiểu vấn đề, Mô tả trường dữ liệu, tiền xử lý dữ liệu, xây dựng mô hình, training và đánh giá mô hình liên quan đến dự đoán 1 khách hàng sẽ mua sản phẩm X hay không?	Nguyễn Trần Ái Nguyên	100%
Tìm hiểu vấn đề, Mô tả trường dữ liệu, tiền xử lý dữ liệu, xây dựng mô hình, training và đánh giá mô hình liên quan đến thuật toán gom cụm để phân cụm khách hàng	Ngô Thị Thùy Linh	100%



DÀNG THÔNG TIN CHI TIẾT NHƯƠM

MỤC LỤC

Yêu cầu của Đồ án/Bài tập	4
Mô tả	4
Yêu cầu:	5
Kết quả	5
Câu 1: Dự đoán doanh số sản phẩm bán ra ở năm tiếp theo	5
Tìm hiểu vấn đề:	5
Mô tả trường dữ liệu	5
Khai phá dữ liệu:	6
4. Tiền xử lý dữ liệu:	11
5. Xây dựng mô hình:	11
6. Đánh giá và nhận xét kết quả	12
Câu 2: Dự đoán 1 khách hàng sẽ mua sản phẩm Bike hay không?	15
Tìm hiểu vấn đề:	15
Mô tả trường dữ liệu	15
Khai phá dữ liệu:	16
Describe Data:	16
Visualize Data:	17
4. Tiền xử lý dữ liệu:	21
5. Xây dựng mô hình:	21
6. Đánh giá và nhận xét kết quả	22
Câu 3: Sử dụng thuật toán gom cụm để phân cụm khách hàng	25
Tìm hiểu vấn đề	25
Mô tả trường dữ liệu	25
Khai phá dữ liệu	25



Hiện xử lý dữ liệu	34
Thuật toán gom cụm để phân cụm khách hàng	34
Câu 4: Dự đoán mức lương của 1 nhân viên dựa vào công việc, level tổ chức, tình trạng hôn nhân, giới tính, số năm làm việc, Rate lương tương ứng	40
Tìm hiểu vấn đề:	40
Mô tả trường dữ liệu	41
Khai phá dữ liệu	41
Tiền xử lý dữ liệu	49
Xây dựng mô hình	51
Training mô hình và Đánh giá mô hình	53
Link Colab và tài liệu tham khảo	55
Link Colab:	55
Tài liệu tham khảo:	55

A. Yêu cầu của Đồ án/Bài tập

1. Mô tả

Dataset: Adventurework 2012

Sử dụng file .bak để khôi phục cơ sở dữ liệu mẫu vào phiên bản SQL Server của bạn. Adventure Works Cycles là một công ty sản xuất lớn, đa quốc gia chuyên sản xuất và phân phối xe đạp bằng kim loại và composite cho các thị trường thương mại ở Bắc Mỹ, Châu Âu và Châu Á. Trụ sở chính của Adventureworks Cycles là Bothell, Washington, quy mô 500 công nhân. Ngoài ra, Adventure Works Cycles còn có một số đội bán hàng trên toàn cơ sở thị trường của mình thuộc các khu vực (region). Adventureworks Cycles hiện muốn mở rộng thị phần của mình bằng cách nhằm mục tiêu quảng cáo đến những khách hàng tốt nhất của mình, mở rộng tính khả dụng của sản phẩm thông qua trang Web bên ngoài và giảm chi phí bán hàng bằng cách giảm chi phí sản xuất.



2. Yêu cầu:

- 1- Dự đoán doanh số sản phẩm bán ra cho năm tiếp theo
- 2- Dự đoán 1 khách hàng sẽ mua sản phẩm X hay không?
- 3- Sử dụng thuật toán gom cụm để phân cụm khách hàng
- 4- Dự đoán lương của 1 nhân viên.

B. Kết quả

Câu 1: Dự đoán doanh số sản phẩm bán ra ở năm tiếp theo

1. Tìm hiểu vấn đề:

- Để dự đoán doanh số thì đầu tiên cần phải hiểu được doanh số bán hàng phụ thuộc vào những trường dữ liệu như sau (số lượng sản phẩm bán ra, giá sản phẩm,...)
- Dựa vào các yếu tố trên, tìm hiểu ý nghĩa của trường dữ liệu tương ứng thông qua [dataedo](#). Nhóm thấy cột LineTotal là Tổng sale của 1 sản phẩm bán ra trong 1 năm $\text{LineTotal} = \text{UnitPrice} * (1 - \text{UnitPriceDiscount}) * \text{OrderQty}$ nên dựa vào đó để dự đoán doanh thu của 1 sản phẩm sau đó tính tổng doanh thu các sản phẩm qua các năm.
- Chọn các trường dữ liệu sau:

- + Product ID
- + OrderQty
- + Year
- + UnitPriceDiscount
- + LineTotal
- + UnitPrice

- Với cột Year ta sẽ xử lý dữ liệu trong SQL:
 - + Lấy cột OrderDate trong bảng [Sales.SalesOrderHeader]
 - + Sử dụng hàm Year(OrderDate) để lấy năm tương ứng
 - + Inner join dữ liệu lại và ta có được cột Year

2. Mô tả trường dữ liệu

Tên cột	Mô tả	Kiểu dữ liệu
---------	-------	--------------



Product ID	ID định danh của một sản phẩm	int
OrderQty	Số lượng mua của sản phẩm	int
Year	Năm bán của sản phẩm đó	Year
UnitPriceDiscount	Tỷ lệ giảm giá của sản phẩm đó	float
UnitPrice	Giá gốc của sản phẩm	float
LineTotal	Tổng Sale của 1 sản phẩm	float

3. Khai phá dữ liệu:

a. Describe Data:

```
#Explore data
df.describe(include='all')
```

	OrderQty	UnitPrice	UnitPriceDiscount	Year	LineTotal
count	121317.000000	121317.000000	121317.000000	121317.000000	121317.000000
mean	2.266080	465.093496	0.002826	2013.034768	905.449207
std	2.491323	751.885081	0.024811	0.820866	1693.417389
min	1.000000	1.328200	0.000000	2011.000000	1.374000
25%	1.000000	21.490000	0.000000	2013.000000	24.990000
50%	1.000000	49.990000	0.000000	2013.000000	134.982000
75%	3.000000	602.346000	0.000000	2014.000000	1120.490000
max	44.000000	3578.270000	0.400000	2014.000000	27893.619000

Nhận xét:



▼ OrderQty

Độ lệch chuẩn bằng 2.49. Dữ liệu số lượng đặt hàng chủ yếu là 1. Min (1) cách Max (44). Vật range của dữ liệu ngắn [1; 44]

UnitPrice

Độ lệch chuẩn cao 751.885, có nghĩa là đơn giá sản phẩm thay đổi nhiều giữa các sản phẩm. Min (1.328) cách Max (3578.27) xa. Vật range của dữ liệu rộng [1.328; 3578.27]

UnitPriceDiscount

Độ lệch chuẩn chỉ bằng 0.0248 chứng tỏ mức giảm giá quanh (0.0028). Min (0) cách Max (0.4) rất gần. Vật range của dữ liệu rất ngắn [0; 0.4]

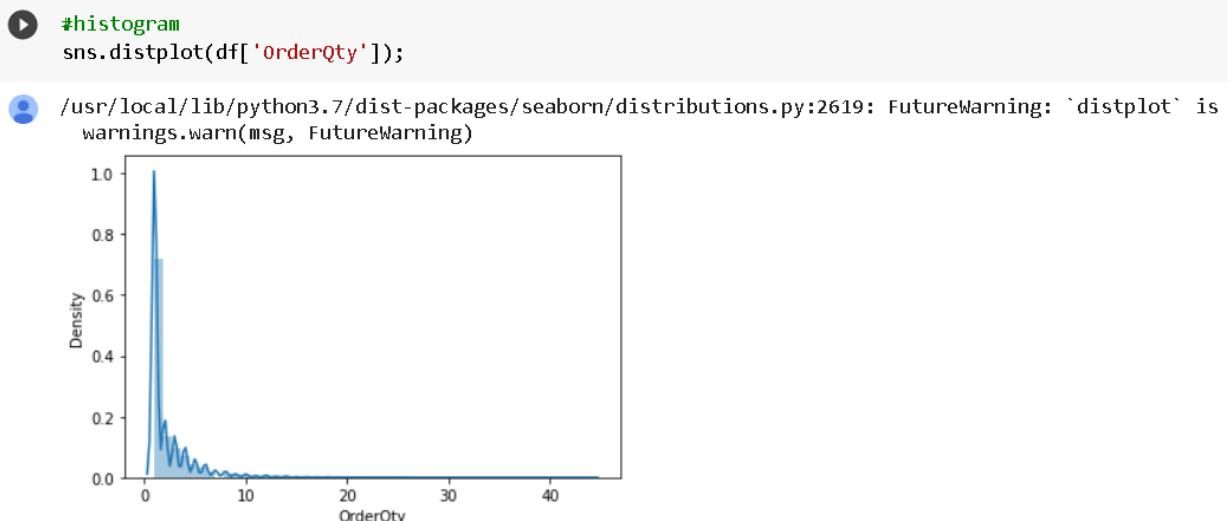
Year

Độ lệch chuẩn 0.82 các năm cách nhau 1 năm và quanh năm 2013. Min(2011) và max(2014), Vật range của dữ liệu rất ngắn [2011; 2014]

LineTotal

Độ lệch chuẩn cao 1693.417, có nghĩa là doanh thu của 1 sản phẩm trong năm thay đổi nhiều. Min (1.374) cách Max (27893.619) xa. Vật range của dữ liệu rộng [1.374; 27893.619]

b. Visualize Data:



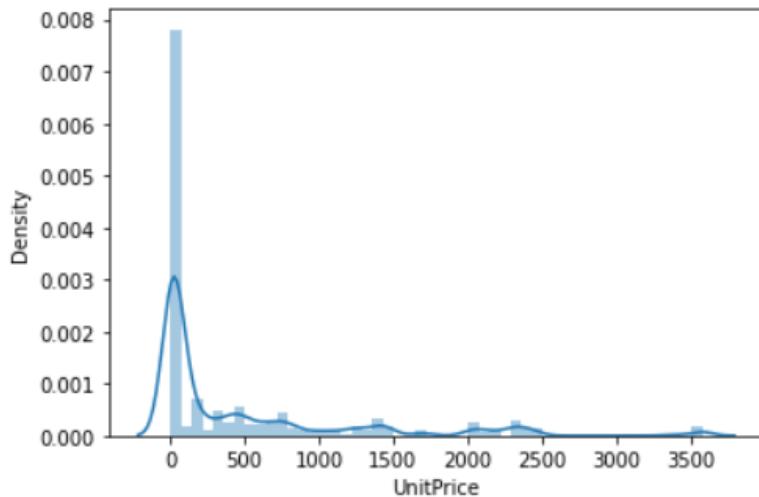
Dữ liệu biểu đồ phân phối lệch trái. Không chênh khỏi phân phối chuẩn.

Nhận xét: Ta thấy rằng số lượng đặt hàng của mỗi sản phẩm sẽ thường xuyên dao động trong khoảng 0-5 sản phẩm (trong đó việc đặt hàng tầm 1-2

san phẩm chiếm tỉ lệ cao nhất). Ta cũng thấy được rằng có 1 số đơn hàng có số lượng đơn từ 30-40

▶ sns.distplot(df['UnitPrice']);

● /usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: warnings.warn(msg, FutureWarning)

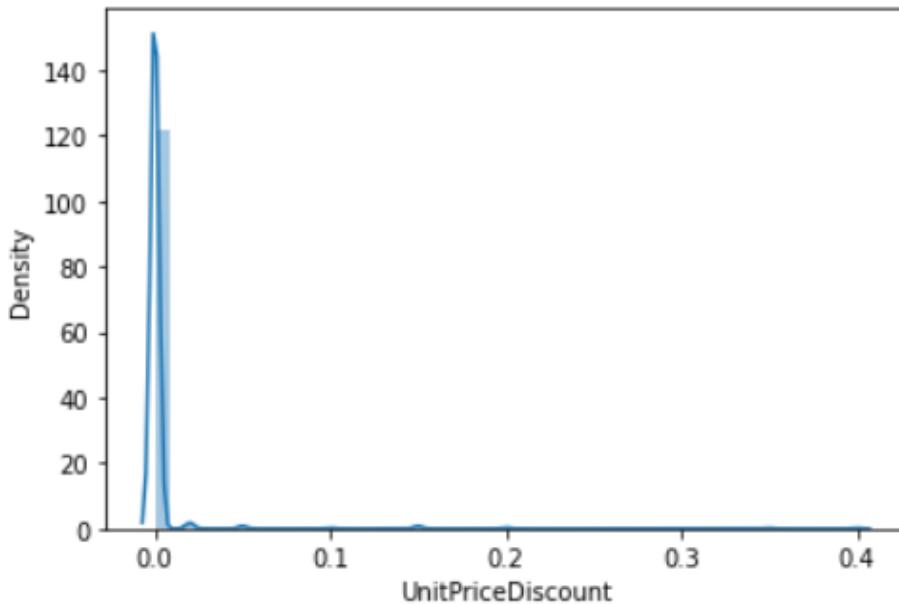


[] Dữ liệu biểu đồ phân phối lệch trái.

Nhận xét: Ta thấy rằng những món hàng có giá tiền thấp tầm 1-100 đô có tần suất xuất hiện nhiều. Trong khi các món hàng có giá trị cao từ 1000 đô trở lên thì số lượng người mua vô cùng ít

```
[ ] sns.distplot(df[ 'UnitPriceDiscount']);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distribution
warnings.warn(msg, FutureWarning)
```

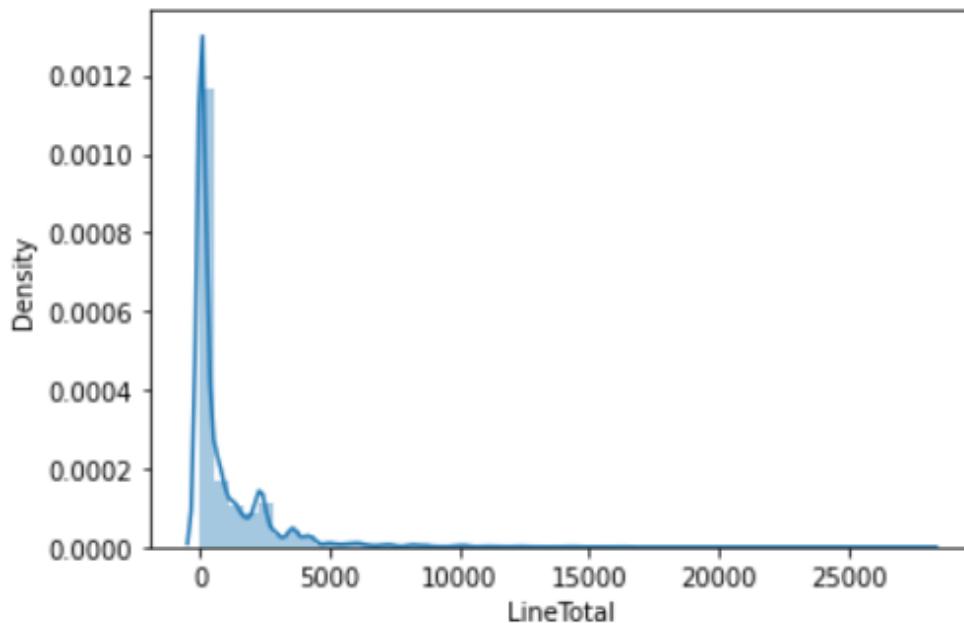


```
[ ] Dữ liệu biểu đồ phân phối lệch trái.
```

Nhận xét: Dù có áp dụng các khuyến mãi nhưng ta thấy rõ rằng các sản phẩm được mua trong dataset không bị ảnh hưởng quá nhiều bởi giảm giá

```
[ ] sns.distplot(df['LineTotal']);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distribution.py:200: FutureWarning:  
  warnings.warn(msg, FutureWarning)
```



Dữ liệu biểu đồ phân phối lệch trái.

Nhận xét: Vì dựa trên những yếu tố như số lượng sản phẩm bán ra, giá cả nên ta có thể thấy rõ ràng nguyên nhân vì sao mà tổng doanh thu của mỗi sản phẩm nằm trong mức 0-2000 đô, thế nhưng vẫn có những sản phẩm cho doanh thu lên tới 25000 đô

4. Tiên xử lý dữ liệu:

```
# %%
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121317 entries, 0 to 121316
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
--- 
 0   OrderQty      121317 non-null  int64  
 1   UnitPrice     121317 non-null  float64
 2   UnitPriceDiscount 121317 non-null  float64
 3   Year          121317 non-null  int64  
 4   LineTotal     121317 non-null  float64
dtypes: float64(3), int64(2)
memory usage: 4.6 MB
```

Có 3 kiểu dữ liệu float, 2 kiểu int, và dữ liệu có 121317 entries, không có cột nào chứa dữ liệu null

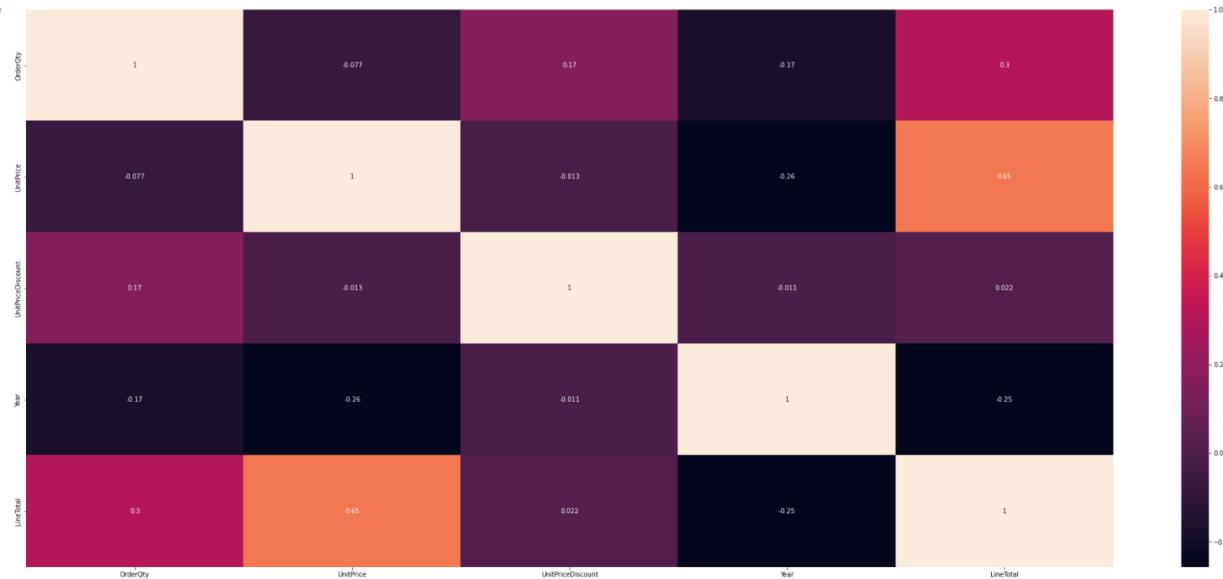
Vì dữ liệu liên quan đến doanh thu và giá sản phẩm, số lượng đặt hàng, % giảm giá theo năm không có giới hạn nên không có kiểm tra Outlier, Dữ liệu đặt hàng có thể trùng nhau nên không xóa duplicate. Vì các trường dữ liệu có non-null count như nhau nên dataset cũng không có dữ liệu null

5. Xây dựng mô hình:

Xác định biến độc lập và biến phụ thuộc:

- + Biến phụ thuộc: LineTotal
- + Biến độc lập: Các biến còn lại

Kiểm tra độ tương quan của mô hình:



Dựa vào head map trên ta thấy được độ tương quan của Year và UnitPriceDiscount với LineTotal vô cùng thấp nên ta sẽ loại bỏ hai cột này

Vậy lúc này ta sẽ có các trường dữ liệu còn lại như sau:

- + Biến độc lập: OrderQty, UnitPrice
- + Biến phụ thuộc: LineTotal

Để xử lý mô hình với đa biến độc lập thì em chọn sử dụng OLS để xây dựng công thức dự đoán

6. Đánh giá và nhận xét kết quả

Mô hình OLS:



```
# %%  
import statsmodels.api as sm  
Y=df.LineTotal  
X=df.drop(columns=["LineTotal","UnitPriceDiscount"])  
X=sm.add_constant(X)  
model=sm.OLS(Y,X)  
results=model.fit()  
print(results.summary())
```



```
OLS Regression Results  
=====
```

Dep. Variable:	LineTotal	R-squared:	0.543
Model:	OLS	Adj. R-squared:	0.543
Method:	Least Squares	F-statistic:	7.208e+04
Date:	Thu, 20 Jan 2022	Prob (F-statistic):	0.00
Time:	03:05:07	Log-Likelihood:	-1.0266e+06
No. Observations:	121317	AIC:	2.053e+06
Df Residuals:	121314	BIC:	2.053e+06
Df Model:	2		
Covariance Type:	nonrobust		

```
=====
```

	coef	std err	t	P> t	[0.025	0.975]
const	-347.8552	4.989	-69.722	0.000	-357.634	-338.077
OrderQty	241.8577	1.323	182.785	0.000	239.264	244.451
UnitPrice	1.5163	0.004	345.856	0.000	1.508	1.525

```
=====
```

Omnibus:	107027.381	Durbin-Watson:	1.721
Prob(Omnibus):	0.000	Jarque-Bera (JB):	6867702.074
Skew:	3.985	Prob(JB):	0.00
Kurtosis:	38.988	Cond. No.	1.36e+03

```
=====
```

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.36e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Công thức dự đoán và test kết quả:



```
# %% [markdown]
# Công thức dự đoán tính tổng sale của sản phẩm trong 1 năm như sau :
#  $y(\text{tổng sale của 1 sản phẩm}) = -347,86 + 242*\text{OrderQty} + 1,5163*\text{UnitPrice}$ 

# %%
data={
    'const':[1,1,1,1],
    'OrderQty':[2,3,4,5],
    'UnitPrice':[1000,2000,3000,4000]
}
x_new=pd.DataFrame(data,columns=['const','OrderQty','UnitPrice'])
y_pred=results.predict(x_new)
print(y_pred.sum())
```

17157.89483236385

Câu 2: Dự đoán 1 khách hàng sẽ mua sản phẩm Bike hay không?

1. Tìm hiểu vấn đề:

- Đề dự đoán 1 khách hàng sẽ mua sản phẩm bike hay không?
- Xác định input: MaritalStatus, YearlyIncome, Gender, TotalChildren, NumberChildrenAtHome, Education, Occupation, HomeOwnerFlag, NumberCarsOwned
- Xác định Predict: Bike Buyer
- Xác định key: Business Entity ID

2. Mô tả trường dữ liệu

Tên cột	Mô tả	Kiểu dữ liệu
Business Entity ID	ID định danh của một khách hàng	int
MaritalStatus	Tình trạng hôn nhân của khách hàng	nvarchar(255)
YearlyIncome	Thu nhập hàng năm của khách hàng	nvarchar(255)
Gender	Giới tính của khách hàng	nvarchar(255)
TotalChildren	Tổng số trẻ em của khách hàng	float
NumberChildrenAtHome	Số trẻ em ở nhà của khách hàng	float
Education	Trình độ văn hóa của khách hàng	nvarchar(255)
Occupation	Nghề nghiệp của khách hàng	nvarchar(255)
HomeOwnerFlag	Khách hàng có sở hữu nhà hay không	bit
NumberCarsOwned	Số xe của khách hàng	float



4. Khai phá dữ liệu:

a. Describe Data:

df.describe(include='all')												
	BusinessEntityID	MaritalStatus	YearlyIncome	Gender	TotalChildren	NumberChildrenAtHome	Education	Occupation	HomeOwnerFlag	NumberCarsOwned	BikeBuyer	
count	19972.000000		18484	18484	18484	18484.000000		18484	18484	18484.000000	18484.000000	19972.000000
unique	NaN		2	5	2	NaN		5	5	NaN	NaN	NaN
top	NaN	M	25001-50000	M	NaN	NaN	Bachelors	Professional	NaN	NaN	NaN	NaN
freq	NaN	10011	5704	9351	NaN	NaN	NaN	5356	5520	NaN	NaN	NaN
mean	10763.079411		NaN	NaN	NaN	1.844352		1.004058	NaN	NaN	0.676369	1.502705
std	5814.133272		NaN	NaN	NaN	1.612408		1.522660	NaN	NaN	0.467874	1.138394
min	1.000000		NaN	NaN	NaN	0.000000		0.000000	NaN	NaN	0.000000	0.000000
25%	5798.750000		NaN	NaN	NaN	0.000000		0.000000	NaN	NaN	0.000000	0.000000
50%	10791.500000		NaN	NaN	NaN	2.000000		0.000000	NaN	NaN	1.000000	2.000000
75%	15784.250000		NaN	NaN	NaN	3.000000		2.000000	NaN	NaN	1.000000	2.000000
max	20777.000000		NaN	NaN	NaN	5.000000		5.000000	NaN	NaN	1.000000	4.000000

Nhận xét:

column BAD là cột định danh

column MaritalStatus, YearlyIncome, Gender, Education, Occupation là dữ liệu String nên không thể mô tả

column TotalChildren

```
[152] # Độ lệch chuẩn chỉ bằng 1.612408 có nghĩa là có số trẻ thấp hơn điểm trung bình nhiều và cũng có số trẻ lớn hơn điểm trung bình nhiều  
# mean(0) < mode(2) < median(3). Vậy dữ liệu phân phối phân phối lệch phải  
# Min (0) cách Max (5) . Vậy range của dữ liệu rất ngắn [0; 5]
```

column NumberChildrenAtHome

```
[ ] # Độ lệch chuẩn bằng 1.522660 có nghĩa là có số trẻ tại nhà thấp hơn điểm trung bình nhiều và cũng có số trẻ tại nhà lớn hơn điểm trung bình nhiều  
# Mode (0) = median (0) < mean (2). Vậy có nghĩa dữ liệu phân phối lệch phải  
# Min (0) cách Max (5) . Vậy range của dữ liệu rất ngắn [0; 5]
```

column HomeOwnerFlag

```
[ ] # Độ lệch chuẩn bằng 0.467874 chứng tỏ số lượng khách có sở hữu nhà tập trung quanh điểm trung bình (0.676369). Vậy có thể cho là phần lớn khách có nhà  
# Mode (0) < median (1) = mean (1). Vậy dữ liệu phân phối lệch phải  
# Min (0) cách Max (1) . Vậy range của dữ liệu [0; 1]
```

column NumberCarsOwned

```
[ ] # Độ lệch chuẩn bằng 1.138394 có nghĩa là có số xe sở hữu hiện tại thấp hơn điểm trung bình và cũng có số xe sở hữu hiện tại lớn hơn điểm trung bình  
# Mode (1) < median (2) = mean (2). Vậy dữ liệu phân phối lệch phải  
# Min (0) cách Max (4) . Vậy range của dữ liệu [0;4]
```

column BikeBuyer

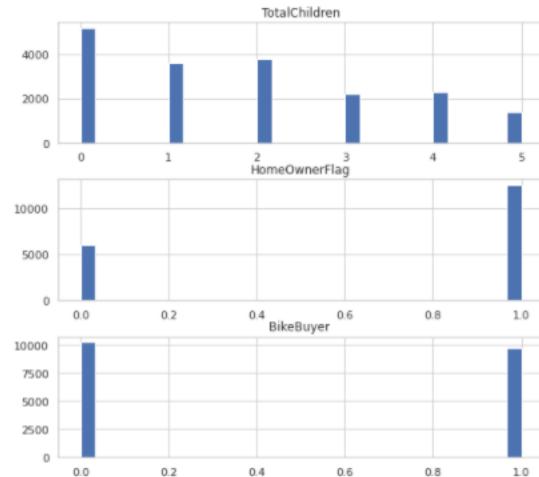
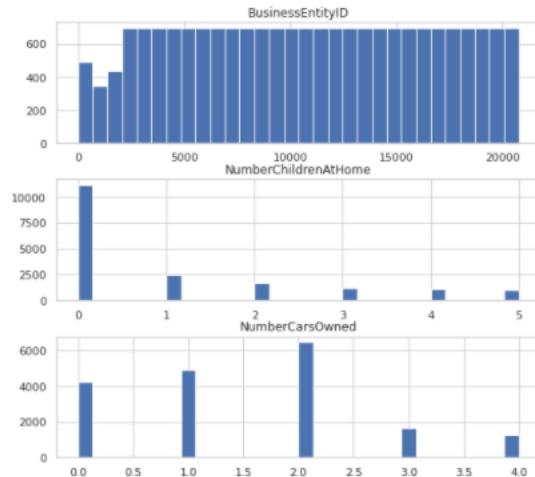
```
[ ] # Độ lệch chuẩn bằng 0.499834 chứng tỏ số lượng khách mua xe đạp tập trung quanh điểm trung bình (0.486631). Vậy có thể cho phần lớn là khách mua xe đạp  
# Mode (0) = median (0) < mean (1). Vậy dữ liệu phân phối lệch phải  
# Min (0) cách Max (1) . Vậy range của dữ liệu [0; 1]
```

b. Visualize Data:

```
[295] # Histogramas das variáveis explicativas numéricas
numeric_feats = [c for c in df.columns if df[c].dtype != 'object' and c not in ['BikeBuyer']]
df_numeric_feats = df[numeric_feats]
```

```
df_numeric_feats.hist(figsize=(20,8), bins=30)
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f895784c550>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f895781fb50>,
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f89577e1190>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f895774c090>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f89577103d0>]],  

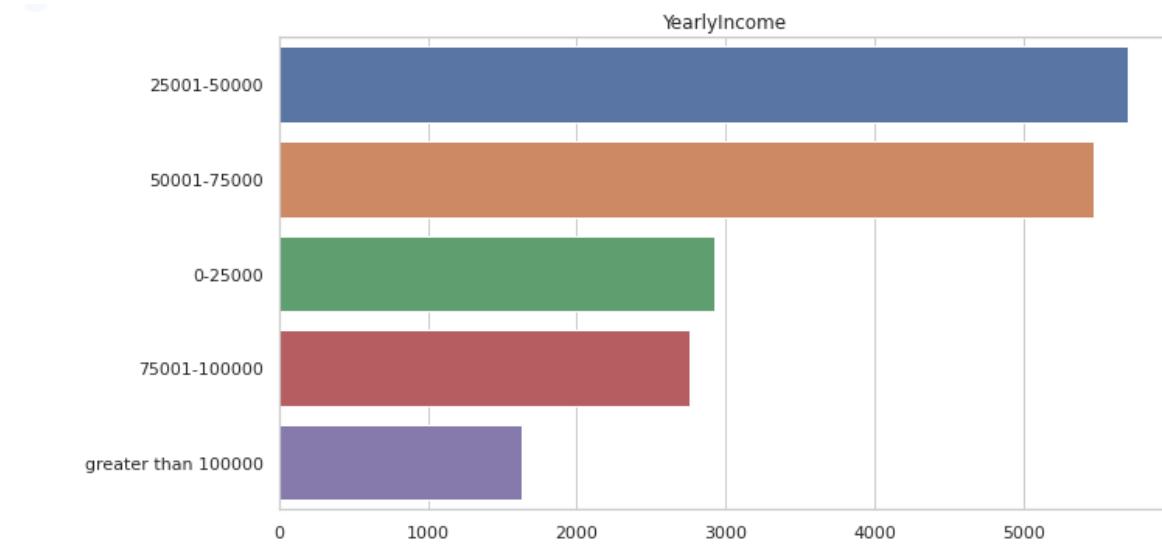
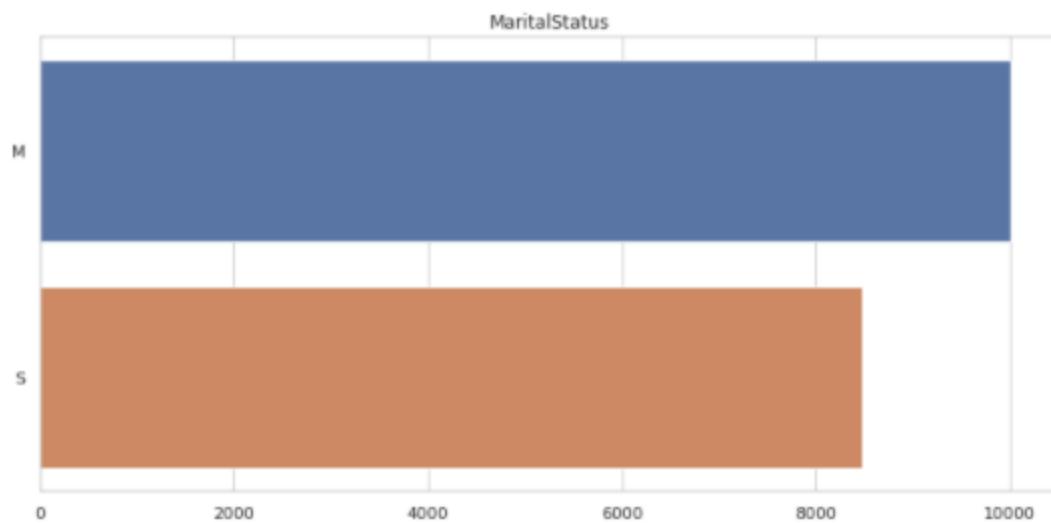
      dtype=object)
```

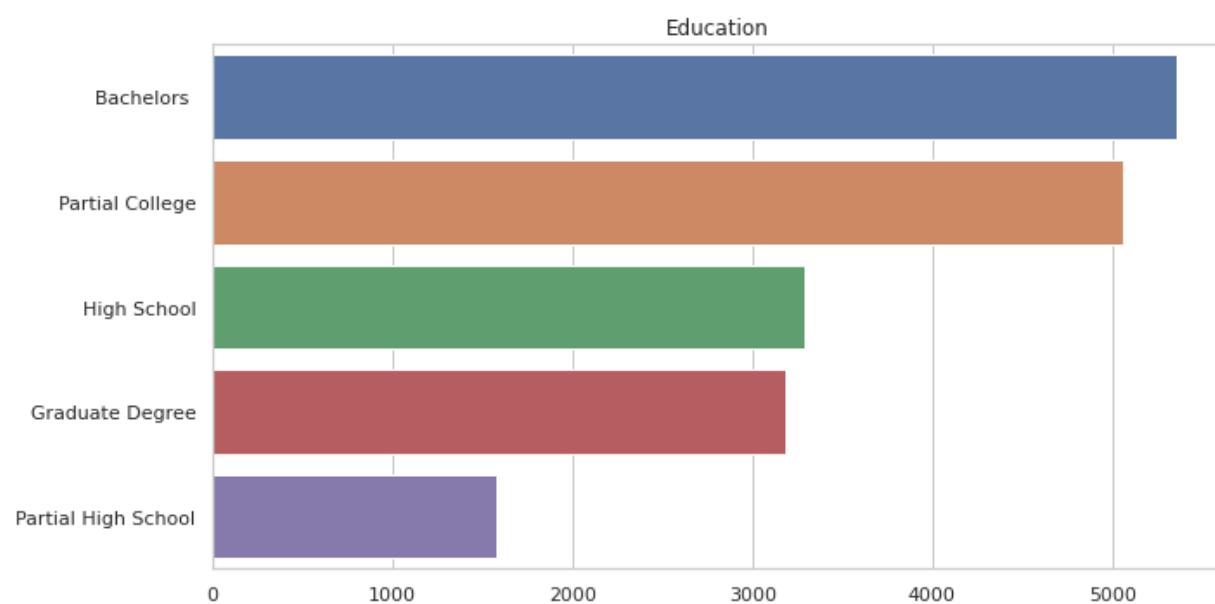
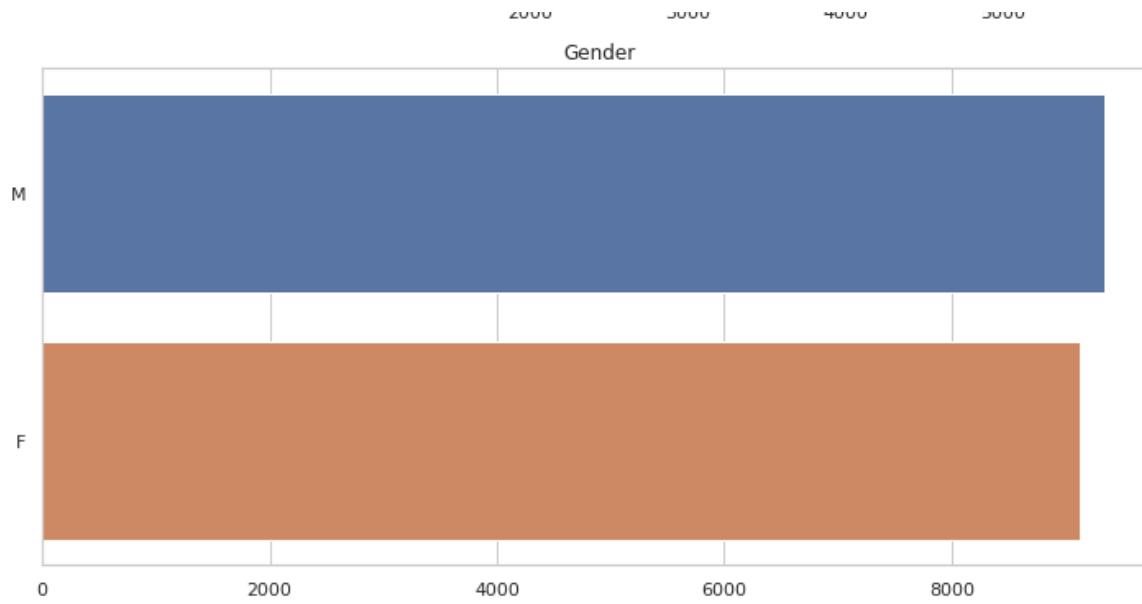


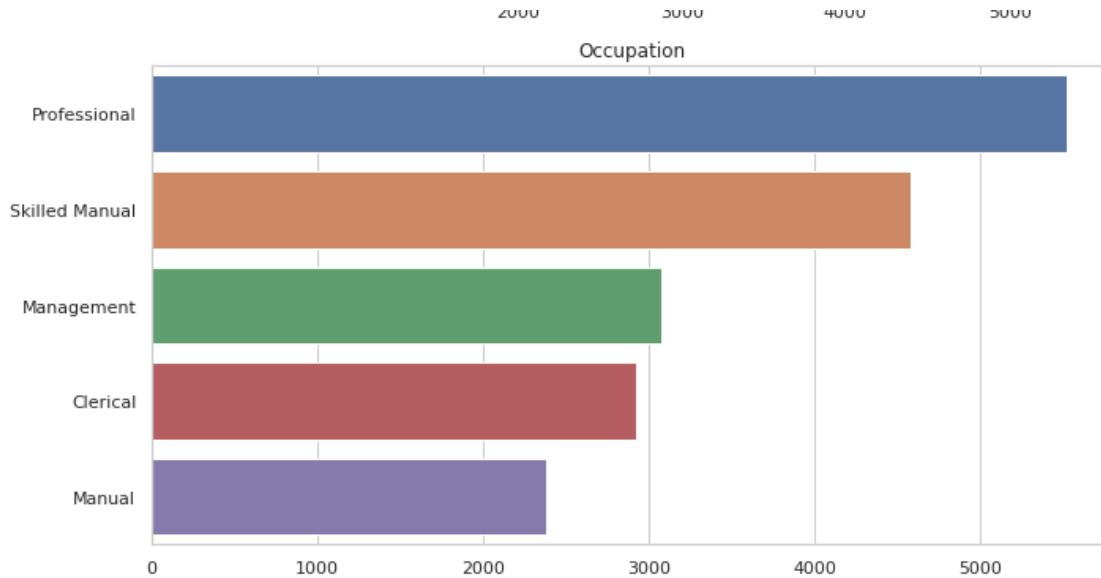
TotalChildren, NumberChildrenAtHome số trẻ em phân phối không đồng đều. Phần lớn khong có trẻ, trẻ tại nhà
 HomeOwerFlag phần lớn khách có nhà

NumberCarOwned số xe khách sở hữu phân phối không đồng đều. Phần lớn khách có xe

BikeBuyer số khách mua và không mua xấp xỉ nha. Khách không mua nhiều hơn khách mua







MaritalStatus: phần lớn khách kết hôn

YearlyIncome: phần lớn khách có thu nhập trung bình

Gender: giới tính xấp xỉ nhau

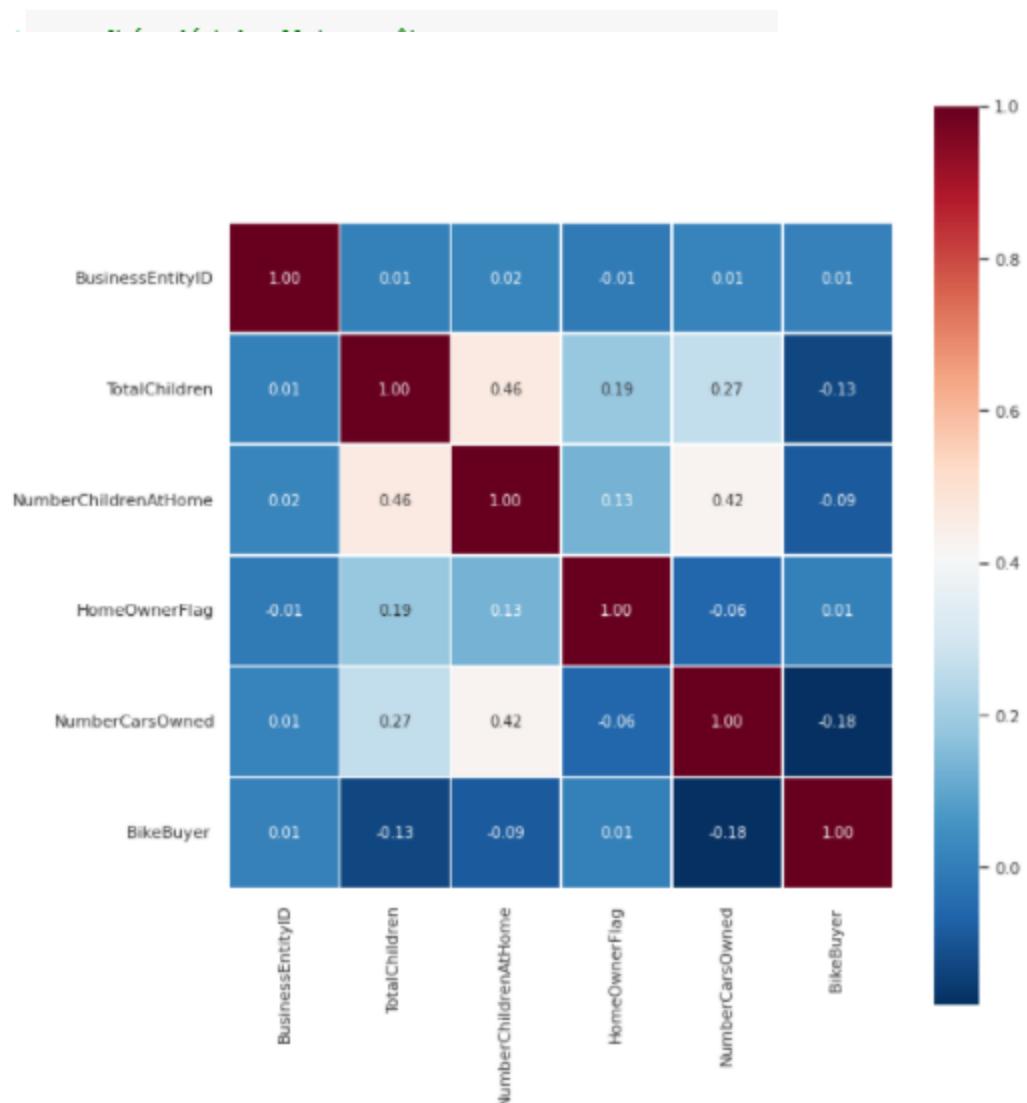
Education: phần lớn khách có mức học thức đại học/ cao đẳng

Occupation: phần lớn khách có công việc chuyên môn

```
[299] df.isnull().sum()
```

```
BusinessEntityID          0
MaritalStatus        1488
YearlyIncome         1488
Gender              1488
TotalChildren       1488
NumberChildrenAtHome 1488
Education            1488
Occupation           1488
HomeOwnerFlag        1488
NumberCarsOwned      1488
BikeBuyer             0
dtype: int64
```

Chỉ có 2 cột BikeBuyer và BusinessEntityID không có giá trị NULL





4. Tiến xу lý dữ liệu:

```
[300] df=df.dropna()
```

```
[301] df.isnull().sum()
```

```
BusinessEntityID      0
MaritalStatus        0
YearlyIncome         0
Gender               0
TotalChildren        0
NumberChildrenAtHome 0
Education             0
Occupation           0
HomeOwnerFlag        0
NumberCarsOwned      0
BikeBuyer            0
dtype: int64
```

```
[292] sum(df.duplicated())
```

```
0
```

=> Không có dữ liệu trùng

```
[302] df.head()
```

	BusinessEntityID	Maritalstatus	YearlyIncome	Gender	TotalChildren	NumberChildrenAtHome	Education	Occupation	HomeOwnerFlag	NumberCarsOwned	BikeBuyer
0	14116	S	0-25000	F	2.0	2.0	High School	Manual	1.0	0.0	1
1	14173	S	25001-50000	M	4.0	2.0	High School	Skilled Manual	1.0	2.0	0
2	14175	S	25001-50000	M	1.0	0.0	Bachelors	Clerical	0.0	1.0	0
3	14182	S	25001-50000	F	2.0	0.0	Partial College	Skilled Manual	1.0	2.0	1
4	14209	S	0-25000	F	0.0	0.0	High School	Manual	0.0	2.0	1

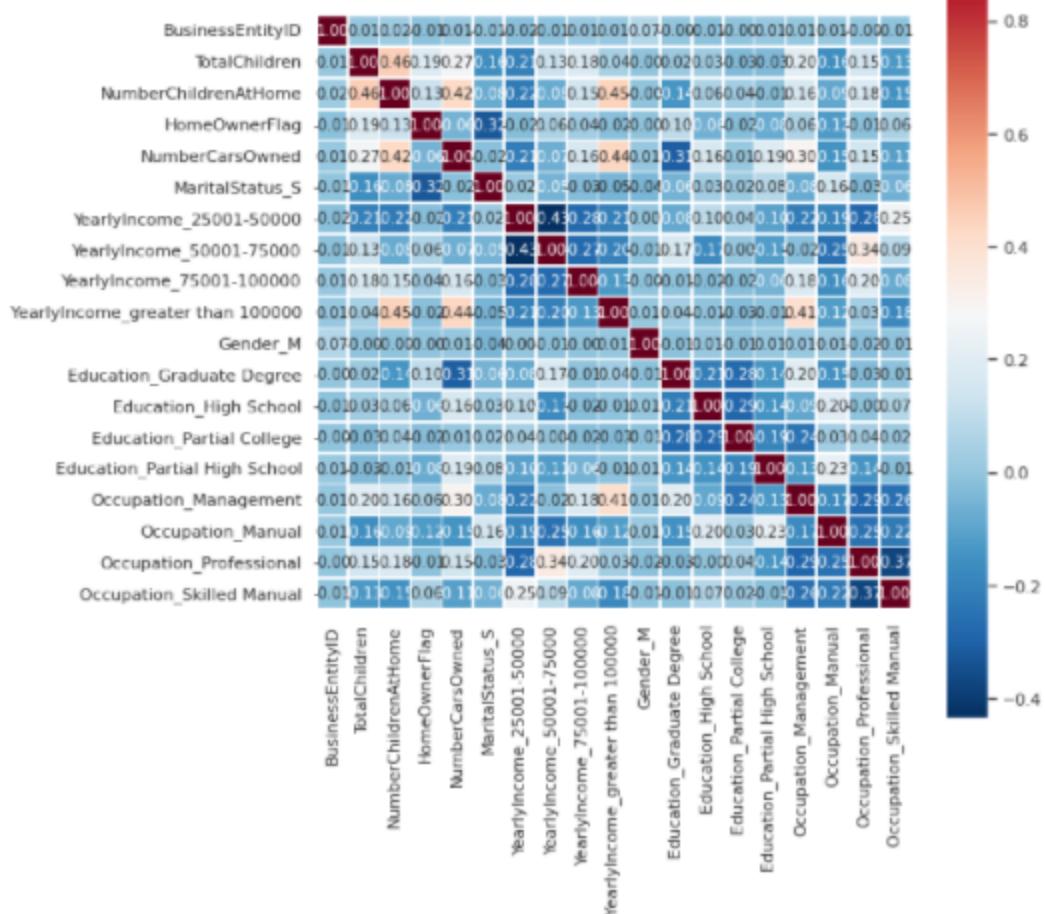
5. Xây dựng mô hình:

Xác định biến độc lập và biến phụ thuộc:

- + Biến phụ thuộc: BikeBuyer
- + Biến độc lập: Các biến còn lại

Kiểm tra độ tương quan của mô hình:

L*





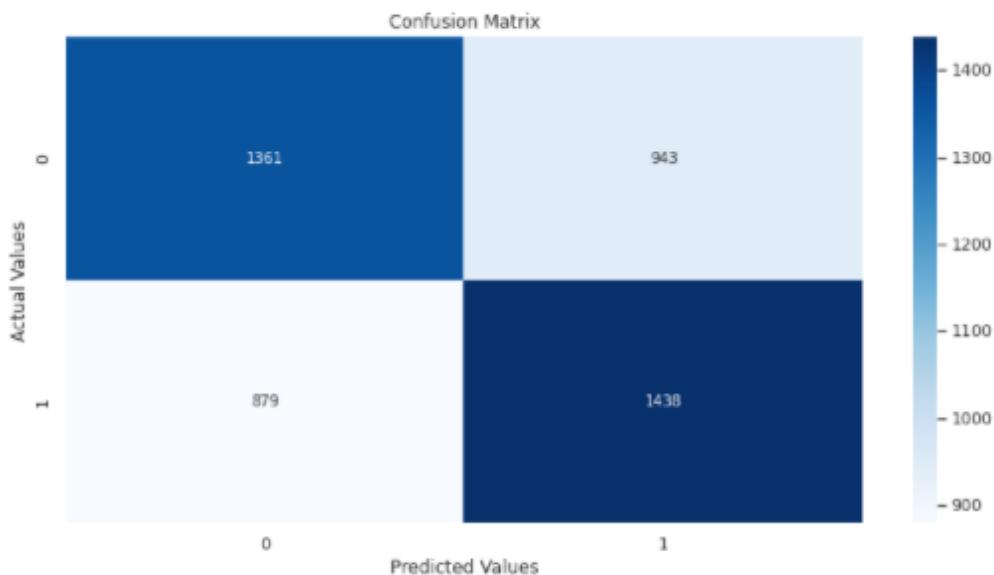
```
[313] # Đào tạo mô hình hồi quy logistic cơ bản với gói đào tạo
lm = LogisticRegression(solver='lbfgs',random_state=0)
lm.fit(x_train, y_train)
y_pred = lm.predict(x_test)
y_pred
print("intercept ")
print(lm.intercept_)
print("")
print("coefficients ")
print(lm.coef_)

intercept
[0.25610801]

coefficients
[[ -3.24766105e-06 -1.77019063e-01  4.61592700e-02  2.30408786e-01
 -2.94854892e-01  3.42725830e-01  5.88695957e-02  1.89259819e-01
 1.03943692e-01  2.16635840e-01  3.31107708e-02  5.05399482e-02
 -1.00574641e-01  5.45771101e-02 -2.32124559e-01  1.06133879e-01
 -1.99935802e-01  1.85915407e-01  2.73418557e-03]]
```

6. Đánh giá và nhận xét kết quả

```
[ ] from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
cm=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(12,6))
plt.title("Confusion Matrix")
sns.heatmap(cm, annot=True,fmt='d', cmap='Blues')
plt.ylabel("Actual Values")
plt.xlabel("Predicted Values")
plt.savefig('confusion_matrix.png')
```



Tạo báo phân loại

```
[ ] print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.61	0.59	0.60	2304
1	0.60	0.62	0.61	2317
accuracy			0.61	4621
macro avg	0.61	0.61	0.61	4621
weighted avg	0.61	0.61	0.61	4621

Nhận xét:

accuracy xấp xỉ 0.6 → Mức độ dự đoán nợ xấu chính xác khá thấp

precision xấp xỉ 0.65 → mức độ báo động nhầm khá cao

recall xấp xỉ 0.6 → mức độ nhận nhầm, sai sót cao

F1 xấp xỉ 0,6 → mức độ cân bằng giữa recall và precision khá cao

▼ Helper code to visualize tree

```
[ ] def tree_graph_to_png(tree, feature_names, png_file_to_save):
    tree_str = export_graphviz(tree, feature_names=feature_names,
                               filled=True, out_file=None)
    graph = pydotplus.graph_from_dot_data(tree_str)
    graph.write_png(png_file_to_save)
```

▼ Train tree and get predictions

```
[ ] # Instantiate a DecisionTreeClassifier 'dt' with a maximum depth of 6
dt = DecisionTreeClassifier(max_depth=6, random_state=1)
```

```
# Fit dt to the training set
dt.fit(x_train, y_train)
```

```
# Predict test set labels
y_pred = dt.predict(x_test)
print(y_pred[0:5])
```

```
[0 0 1 0 1]
```

```
[ ] # Predict test set labels
y_pred = dt.predict(x_test)
```

```
# Compute test set accuracy
acc = accuracy_score(y_test, y_pred)
print("Test set accuracy: {:.2f}".format(acc))
```

```
Test set accuracy: 0.67
```

Mức độ dự đoán chính xác theo Cây quyết định cao hơn mô hình OLS



Câu 3: Sử dụng thuật toán gom cụm để phân cụm khách hàng

1. Tìm hiểu vấn đề

- Tìm hiểu ý nghĩa và vị trí của các trường dữ liệu tương ứng theo yêu cầu từ [dataedo](#). Nhóm chọn dựa vào nhân khẩu học để phân cụm khách hàng từ views **Sales.vPersonDemographics**
- Chọn các trường dữ liệu sau:
 - + BusinessEntityID
 - + TotalPurchaseYTD
 - + BirthDate
 - + YearlyIncome
 - + Gender
- Với cột BirthDate ta xử lý trong SQL: ta lấy năm sinh trong BirthDate để tính tuổi của khách hàng.

2. Mô tả trường dữ liệu

Tên cột	Mô tả	Kiểu dữ liệu
BusinessEntityID	Mã định danh khách hàng mua sắm tại công ty	int
TotalPurchaseYTD	Tổng doanh thu YTD (year to date)	money
BirthDate	năm sinh	datetime
YearlyIncome	thu nhập hằng năm	nvarchar
Gender	giới tính	char(1) (Female : F Male : M)

3. Khai phá dữ liệu

Đọc dữ liệu:



```
df1 = pd.read_csv('cau3.csv')
df1.head(None)
```

	BusinessEntityID	TotalPurchaseYTD	DateFirstPurchase	BrithYear	MaritalStatus	YearlyIncome	Gender
0	1	0.00		NaN	NaN	NaN	NaN
1	2	0.00		NaN	NaN	NaN	NaN
2	3	0.00		NaN	NaN	NaN	NaN
3	4	0.00		NaN	NaN	NaN	NaN
4	5	0.00		NaN	NaN	NaN	NaN
...
1895	2701	-24.99	00:00:0	1958.0	S	0-25000	M
1896	2702	2375.08	00:00:0	1960.0	M	50001-75000	F
1897	2703	76.95	00:00:0	1939.0	M	greater than 100000	F
1898	2704	2307.98	00:00:0	1950.0	M	25001-50000	F
1899	2705	-539.99	00:00:0	1967.0	M	25001-50000	M

1900 rows × 13 columns

```
df3 = [df1["BusinessEntityID"], df1["TotalPurchaseYTD"],df1["BrithYear"],df1["YearlyIncome"],df1["Gender"] ]
headers = ["BusinessEntityID", "TotalPurchaseYTD", "BrithYear", "YearlyIncome", "Gender"]
df = pd.concat(df3, axis=1, keys=headers)
df
```

	BusinessEntityID	TotalPurchaseYTD	BrithYear	YearlyIncome	Gender
0	1	0.00	NaN	NaN	NaN
1	2	0.00	NaN	NaN	NaN
2	3	0.00	NaN	NaN	NaN
3	4	0.00	NaN	NaN	NaN
4	5	0.00	NaN	NaN	NaN
...
1895	2701	-24.99	1958.0	0-25000	M
1896	2702	2375.08	1960.0	50001-75000	F

+ Xử lý Cột YearlyIncome: gán các khảng giá thu nhập:

- 0 : 0-25000
- 1 : 25001 - 50000
- 2: 50001 - 75000
- 3: 75001 - 100000
- 4: greater than 100000



```
[87] #Hàm để khoảng cách nhập theo sốt 0-24 để để tính toán
df['YearlyIncome'] = df['YearlyIncome'].map({'0-25000': 0, '25001-50000': 1, '50001-75000': 2, '75001-100000': 3, 'greater than 100000' : 4 })
```

```
[88] df['YearlyIncome'] = pd.to_numeric(df['YearlyIncome'])
```

```
[89] df.describe()
```

	BusinessEntityID	TotalPurchaseYTD	BrithYear	YearlyIncome	Age	edit
count	1900.000000	1900.000000	412.000000	412.000000	412.000000	
mean	1456.754737	-2275.869914	1961.361650	1.650485	60.638350	
std	856.224889	5265.143640	11.286699	1.280273	11.286699	
min	1.000000	-48861.593900	1930.000000	0.000000	42.000000	
25%	660.500000	-2510.811200	1953.000000	1.000000	51.000000	
50%	1610.000000	0.000000	1963.000000	1.000000	59.000000	
75%	2230.250000	0.000000	1971.000000	3.000000	69.000000	
max	2705.000000	9263.661800	1980.000000	4.000000	92.000000	

+ Thêm column Age (cột tuổi) vào data:

```
[84] today = pd.to_datetime('today')
```

```
[85] year_today = today.year
```

```
[86] #Tính tuổi của khách hàng
df['Age'] = year_today - df['BrithYear']
df.to_csv('salescustomer.csv')
```



df

	BusinessEntityID	TotalPurchaseYTD	BrithYear	YearlyIncome	Gender	Age	label	edit
0	1	0.00	1961.36165	1.650485	0.0	60.63835	0	
1	2	0.00	1961.36165	1.650485	0.0	60.63835	0	
2	3	0.00	1961.36165	1.650485	0.0	60.63835	0	
3	4	0.00	1961.36165	1.650485	0.0	60.63835	0	
4	5	0.00	1961.36165	1.650485	0.0	60.63835	0	
...	

+ Mô tả:



[96] df.describe()

	BusinessEntityID	TotalPurchaseYTD	BrithYear	YearlyIncome	Gender	Age	label
count	1900.000000	1900.000000	1900.000000	1900.000000	1900.000000	1900.000000	1900.000000
mean	1456.754737	-2275.869914	1961.361650	1.650485	0.076842	60.638350	2.000526
std	856.224889	5265.143640	5.250799	0.595609	0.266411	5.250799	3.142483
min	1.000000	-48861.593900	1930.000000	0.000000	0.000000	42.000000	0.000000
25%	660.500000	-2510.811200	1961.361650	1.650485	0.000000	60.638350	0.000000
50%	1610.000000	0.000000	1961.361650	1.650485	0.000000	60.638350	0.000000
75%	2230.250000	0.000000	1961.361650	1.650485	0.000000	60.638350	4.000000
max	2705.000000	9263.661800	1980.000000	4.000000	1.000000	92.000000	9.000000

[97] df.info()

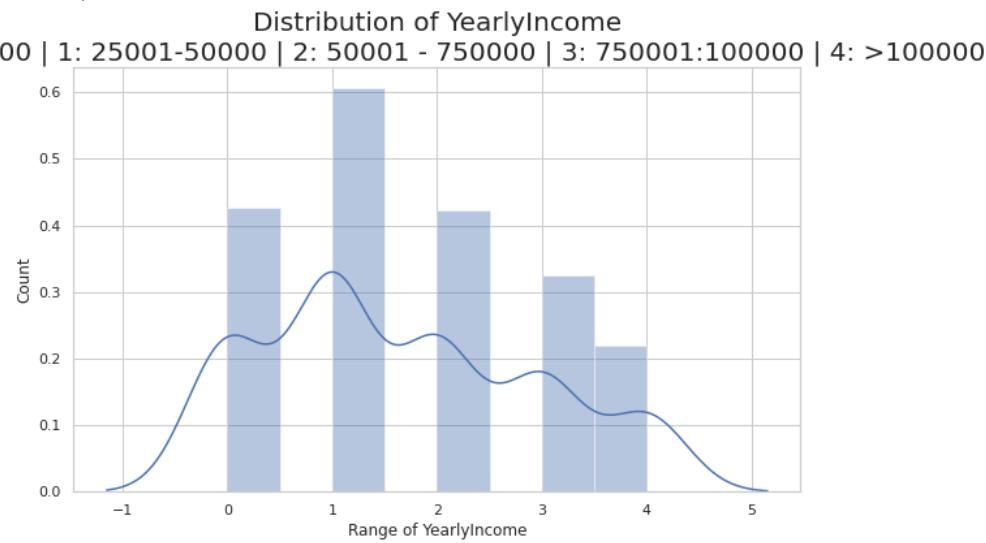
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1900 entries, 0 to 1899
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   BusinessEntityID  1900 non-null   int64  
 1   TotalPurchaseYTD  1900 non-null   float64 
 2   BrithYear         412 non-null   float64 
 3   YearlyIncome      412 non-null   float64 
 4   Gender            412 non-null   float64 
 5   Age               412 non-null   float64 
dtypes: float64(5), int64(1)
memory usage: 89.2 KB
```

- + Một số đồ thị phân phối: Age, Gender, YearlyIncome, TotalPurchaseYTD

Phân phôi thu nhập hàng năm:

```
[90] #Distribution of Annual Income
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(df['YearlyIncome'])
plt.title('Distribution of YearlyIncome'\n'0: 0-25000 | 1: 25001-50000 | 2: 50001 - 750000 | 3: 750001:100000 | 4: >100000', fontsize = 20)
plt.xlabel('Range of YearlyIncome')
plt.ylabel('Count')

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function
  warnings.warn(msg, FutureWarning)
Text(0, 0.5, 'Count')
```



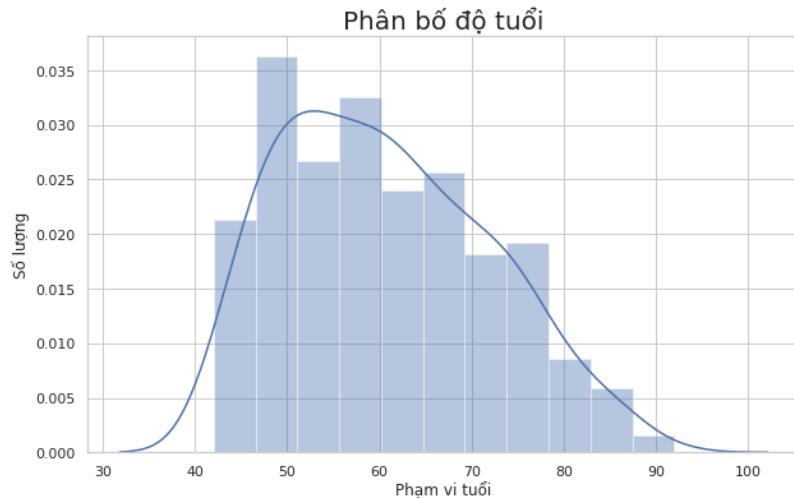
Từ đồ thị trên ta thấy:

- + Thu nhập có số lượng khách hàng cao nhất: 250001 - 50000
 - + Thu nhập có số lượng khách hàng thấp nhất: >100000
- => Các khách hàng có thu nhập 250001 - 50000 thì mua sắm nhiều.

Phân bố theo độ tuổi:

```
[91] # Phân bố độ tuổi
    plt.figure (figsize = (10, 6))
    sns.set (style = 'whitegrid')
    sns.distplot (df['Age'])
    plt.title ('Phân bố độ tuổi', fontsize = 20)
    plt.xlabel ('Phạm vi tuổi')
    plt.ylabel ('Số lượng')

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function
  warnings.warn(msg, FutureWarning)
Text(0, 0.5, 'Số lượng')
```



Bằng cách xem biểu đồ:

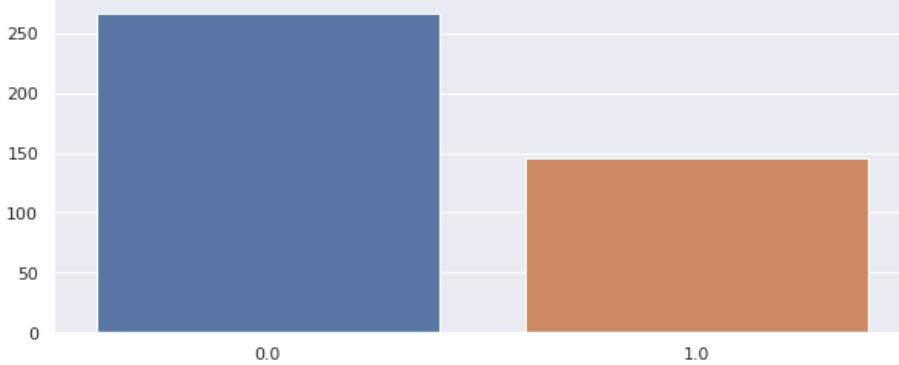
- Có nhiều khách hàng ở nhiều lứa tuổi.
- chúng ta có thể hiểu rằng độ tuổi có số lượng khách hàng cao nhất là khoảng 45-55 và số lượng khách hàng thấp nhất là từ 80-90.

Phân tích giới tính:

```
[92] df['Gender'] = df['Gender'].map({'F':0, 'M':1 })
```

```
genders = df.Gender.value_counts ()  
sns.set_style ("darkgrid")  
plt.figure (figsize = (10,4))  
sns.barplot (x = genders.index, y = genders.values)  
plt.title('Distribution of Gender'\n'0: Female 1: Male', fontsize = 20)  
plt.show ()
```

Distribution of Gender
0: Female 1: Male

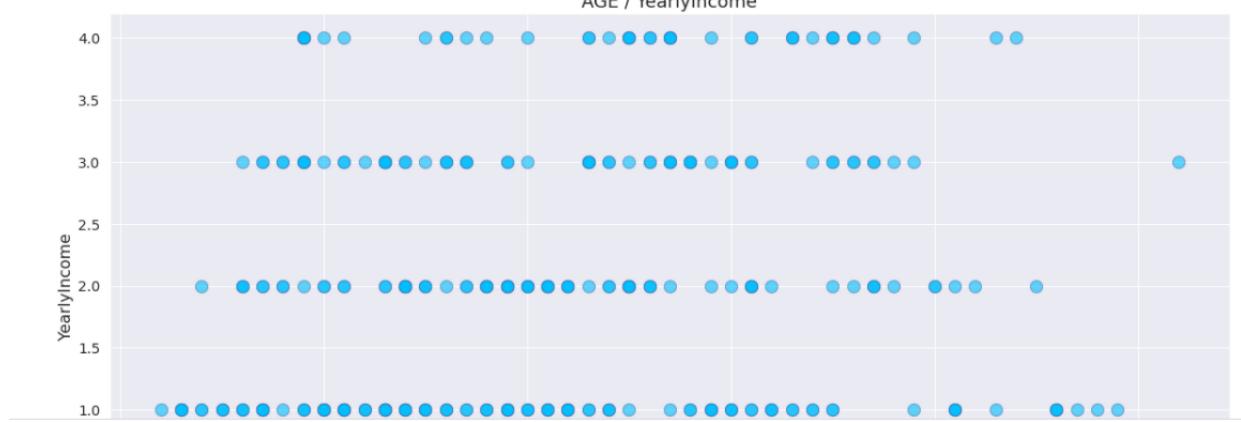


Từ đồ thị ta thấy: số lượng khách hàng Nữ nhiều hơn khách hàng Nam

```
sns.scatterplot('Age', 'YearlyIncome',
                 data = df,
                 color = 'deepskyblue',
                 s = 150,
                 alpha = 0.6,
                 edgecolor = 'b')
plt.title('AGE / YearlyIncome',
          fontsize = 18)
plt.xlabel('Age',
           fontsize = 16)
plt.ylabel('YearlyIncome',
           fontsize = 16)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)

plt.savefig('age_income.png')
plt.show()
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the FutureWarning

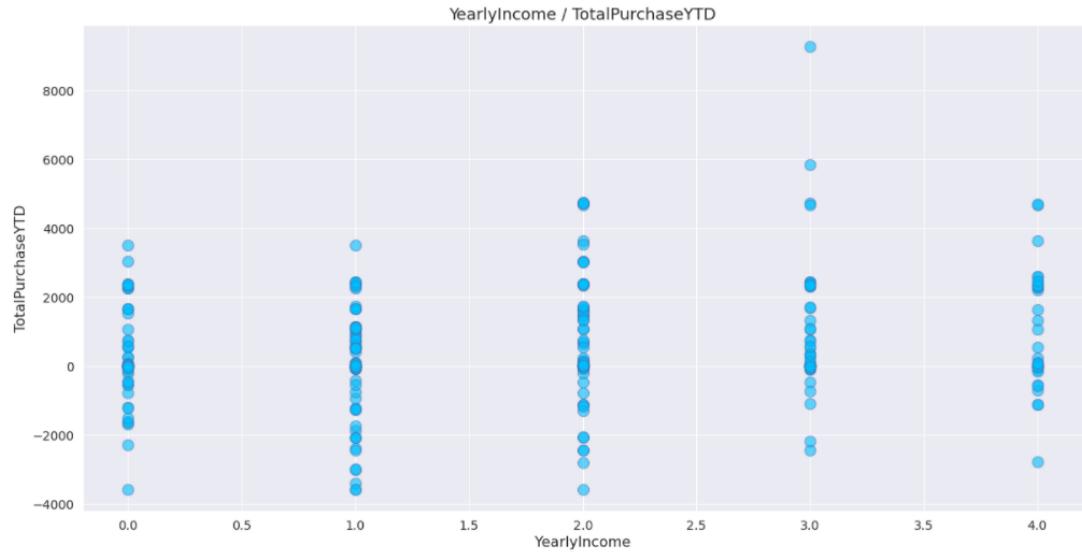


Như chúng ta thấy, khi tuổi càng cao thì thu nhập cũng tăng theo. Vì vậy, chúng ta có thể nói rằng các thuộc tính 'Tuổi' và 'Thu nhập' có mối quan hệ tuyến tính.

```
fontsize = 18)
plt.xlabel('YearlyIncome',
           fontsize = 16)
plt.ylabel('TotalPurchaseYTD',
           fontsize = 16)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)

plt.savefig('YearlyIncome vs TotalPurchaseYTD.png')
plt.show()

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positions
FutureWarning
```



Như chúng ta thấy, Thu nhập càng thấp thì vỡ nợ tín dụng càng tăng khi mua sắm. Vì vậy, chúng ta có thể nói rằng các thuộc tính 'Thu nhập' và 'TotalPurchaseYDT' có mối quan hệ tuyến tính.

4. Tiến xử lý dữ liệu

```
#Thống kê các cột có giá trị null
df.isnull().sum().sort_values(ascending=False)
for column in df.columns:
    percent=df[column].isnull().mean()
    print(f'{column}:{round(percent*100,3)}')
```

BusinessEntityID:0.0
TotalPurchaseYTD:0.0
BrithYear:78.316
YearlyIncome:78.316
Gender:78.316
Age:78.316

```
[99] df['Age'].fillna(value = df['Age'].mean(),inplace=True)
df['YearlyIncome'].fillna(value = df['YearlyIncome'].mean(),inplace=True)
df['Gender'].fillna(value = df['Gender'].median(),inplace=True)
df['BrithYear'].fillna(value = df['BrithYear'].mean(),inplace=True)
```

```
100] df
```

	BusinessEntityID	TotalPurchaseYTD	BrithYear	YearlyIncome	Gender	Age	edit
0	1	0.00	1961.36165	1.650485	0.0	60.63835	
1	2	0.00	1961.36165	1.650485	0.0	60.63835	
2	3	0.00	1961.36165	1.650485	0.0	60.63835	
3	4	0.00	1961.36165	1.650485	0.0	60.63835	
4	5	0.00	1961.36165	1.650485	0.0	60.63835	
...
1895	2701	-24.99	1958.00000	0.000000	1.0	64.00000	
1896	2702	2375.08	1960.00000	2.000000	0.0	62.00000	
1897	2703	76.95	1939.00000	4.000000	0.0	83.00000	
1898	2704	2307.98	1950.00000	1.000000	0.0	72.00000	

5. Thuật toán gom cụm để phân cụm khách hàng

- Xử dụng thuật toán KMeans để gom cụm

```
from sklearn.preprocessing import StandardScaler # data normalization
from sklearn.cluster import KMeans # K-means algorithm
```

1. Tính K value

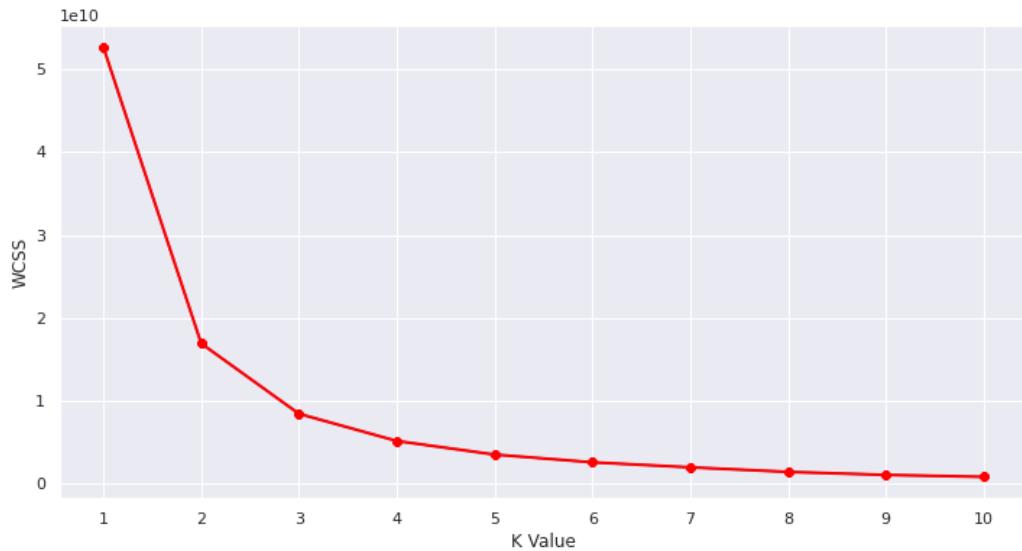
Kiểm tra giá trị để gom cụm

Ta sẽ tính K theo phương pháp khuỷu tay:

```
[101] X=df[["YearlyIncome","TotalPurchaseYTD"]]
```

```
[102] wcss=[]
for i in range(1,11):
    km=KMeans(n_clusters=i)
    km.fit(X)
    wcss.append(km.inertia_)
```

```
#The elbow curve
plt.figure(figsize=(12,6))
plt.plot(range(1,11),wcss)
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker = "8")
plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS")
plt.show()
```



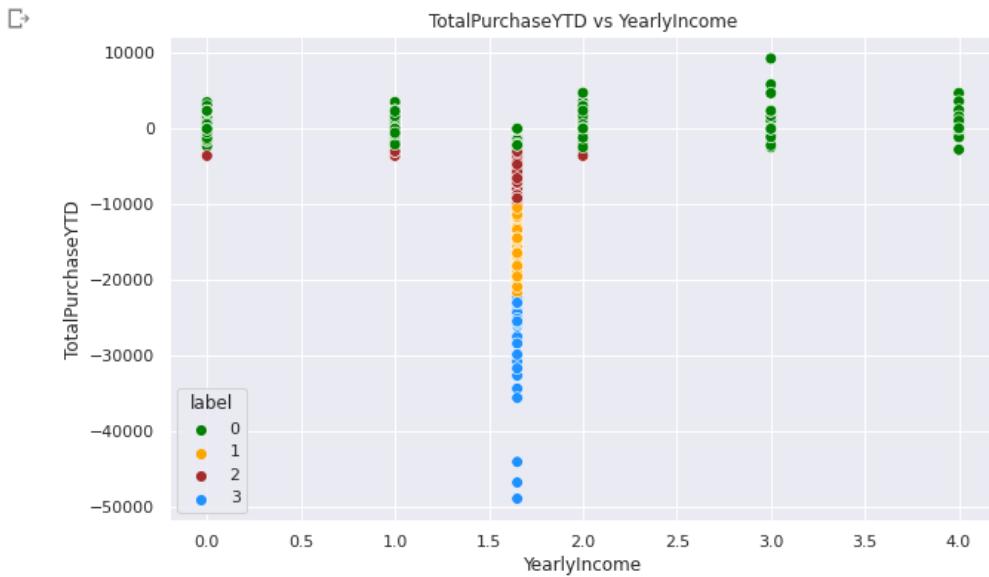
Đây được gọi là đồ thị khuỷu tay, trục x là số lượng cụm, số lượng cụm được lấy tại điểm khớp khuỷu tay. Đây là điểm mà việc tạo các cụm có liên quan nhất vì ở đây giá trị của WCSS đột ngột ngừng giảm. Ở đây trong biểu đồ, sau 4 lần giảm là nhỏ nhất, vì vậy chúng tôi lấy 4 là số cụm.

Gom cụm theo TotalPurchaseYTD với YearlyIncome

```
▶ km1=KMeans(n_clusters=4)
#Fitting the input data
km1.fit(X)
#predicting the labels of the input data
y=km1.predict(X)
#adding the labels to a column named label
df["label"] = y
#The new dataframe with the clustering done
df.head()
```

	BusinessEntityID	TotalPurchaseYTD	BrithYear	YearlyIncome	Gender	Age	label
0	1	0.0	1961.36165	1.650485	0.0	60.63835	0
1	2	0.0	1961.36165	1.650485	0.0	60.63835	0
2	3	0.0	1961.36165	1.650485	0.0	60.63835	0
3	4	0.0	1961.36165	1.650485	0.0	60.63835	0
4	5	0.0	1961.36165	1.650485	0.0	60.63835	0

```
▶ plt.figure(figsize=(10,6))
sns.scatterplot(x = 'YearlyIncome',y = 'TotalPurchaseYTD',hue="label",
                 palette=['green','orange','brown','dodgerblue'], legend='full',data = df ,s = 60 )
plt.xlabel('YearlyIncome')
plt.ylabel('TotalPurchaseYTD')
plt.title('TotalPurchaseYTD vs YearlyIncome')
plt.show()
```



Chúng ta có thể thấy rõ ràng 4 cụm khác nhau đã được hình thành từ dữ liệu.

+ Cụm red là những khách hàng có thu nhập 28000 - 50000 và nợ tín dụng từ 20000-50000

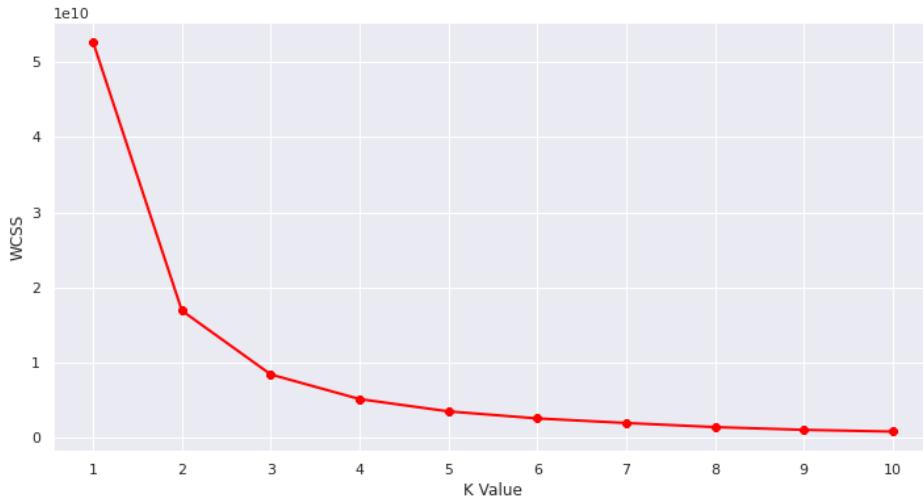
- + Cụm green là những khách hàng có điểm thu nhập 28000-50000 và nợ tín dụng từ 10000-25000
- + Cụm blue là những khách hàng có thu nhập 0-50000 và nợ tín dụng từ 5000-10000
- + Cụm màu vàng là những khách hàng có thu nhập 25000 - hơn 100000 và họ không nợ tín dụng

Gom cụm theo: Age, TotalPurchaseYTD, YearlyIncome

```
106] X2=df[["Age","YearlyIncome","TotalPurchaseYTD"]]

#Now we calculate the Within Cluster Sum of Squared Errors (WSS) for different values of k.
wcss = []
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, init="k-means++")
    kmeans.fit(X2)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(12,6))
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")
plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS")
plt.show()
```

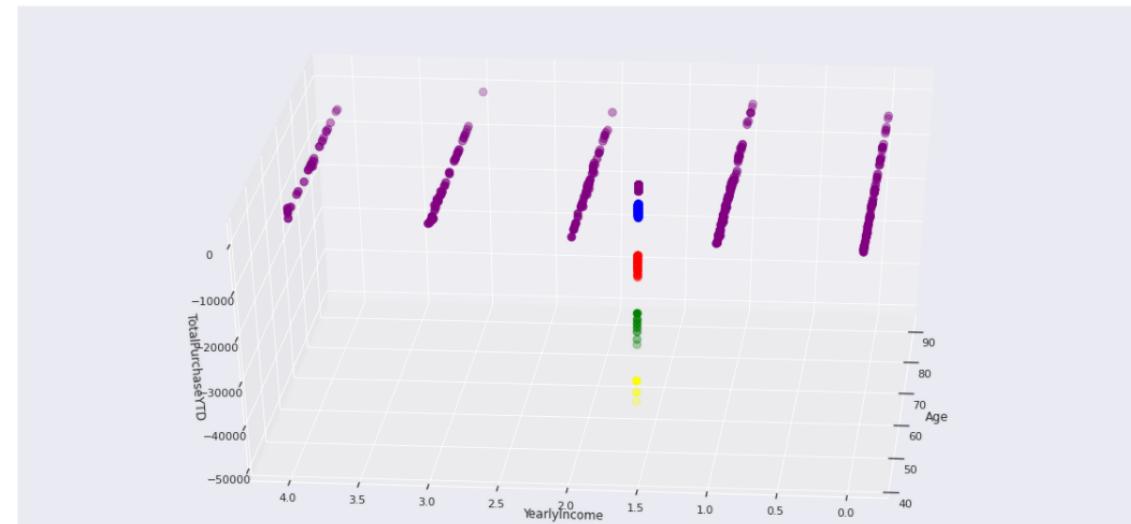


```
#0/ ] #we choose the k for which wss starts to diminish
km2 = KMeans(n_clusters=5)
y2 = km2.fit_predict(X2)
df["label"] = y2
#The data with labels
df.head()
```

	BusinessEntityID	TotalPurchaseYTD	BrithYear	YearlyIncome	Gender	Age	label	edit
0	1	0.0	1961.36165	1.650485	0.0	60.63835	0	
1	2	0.0	1961.36165	1.650485	0.0	60.63835	0	
2	3	0.0	1961.36165	1.650485	0.0	60.63835	0	
3	4	0.0	1961.36165	1.650485	0.0	60.63835	0	
4	5	0.0	1961.36165	1.650485	0.0	60.63835	0	

```
#3D Plot as we did the clustering on the basis of 3 input features
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df.Age[df.label == 0], df["YearlyIncome"][df.label == 0], df["TotalPurchaseYTD"][df.label == 0], c='purple', s=60)
ax.scatter(df.Age[df.label == 1], df["YearlyIncome"][df.label == 1], df["TotalPurchaseYTD"][df.label == 1], c='red', s=60)
ax.scatter(df.Age[df.label == 2], df["YearlyIncome"][df.label == 2], df["TotalPurchaseYTD"][df.label == 2], c='blue', s=60)
ax.scatter(df.Age[df.label == 3], df["YearlyIncome"][df.label == 3], df["TotalPurchaseYTD"][df.label == 3], c='yellow', s=60)
ax.scatter(df.Age[df.label == 4], df["YearlyIncome"][df.label == 4], df["TotalPurchaseYTD"][df.label == 4], c='green', s=60)

ax.view_init(35, 185)
plt.xlabel("Age")
plt.ylabel("YearlyIncome")
ax.set_zlabel('TotalPurchaseYTD')
plt.show()
```



- Cụm purple: thu nhập cao , chi tiêu mua sắm ít và tuổi già
- Cụm yellow: thu nhập trung bình, chi tiêu mua sắm trung bình và tuổi già, không nợ thẻ tín dụng
- Cụm green: thu nhập trung bình, nợ thẻ tín dụng trung bình và tuổi trung niên

- Cụm blue: thu nhập trung bình, nợ thẻ tín dụng cao và tuổi trẻ
- Cụm red: thu nhập trung bình, nợ thẻ tín dụng trung bình và tuổi già.

ID của khách hàng của từng cụm:

```
[109] cust1=df[df["label"]==1]
      print('Number of customer in 1st group=', len(cust1))
      print('They are -', cust1["BusinessEntityID"].values)
      print("-----")
      cust2=df[df["label"]==2]
      print('Number of customer in 2nd group=', len(cust2))
      print('They are -', cust2["BusinessEntityID"].values)
      print("-----")
      cust3=df[df["label"]==3]
      print('Number of customer in 3rd group=', len(cust3))
      print('They are -', cust3["BusinessEntityID"].values)
      cust4=df[df["label"]==3]
      print('Number of customer in 4rd group=', len(cust4))
      print('They are -', cust4["BusinessEntityID"].values)
      cust5=df[df["label"]==4]
      print('Number of customer in 5rd group=', len(cust5))
      print('They are -', cust5["BusinessEntityID"].values)
```



⇒ Number of customer in 1st group= 39
They are - [297 343 431 475 497 559 561 569 591 611 647 653 667 703
803 831 855 927 1017 1051 1173 1205 1265 1269 1295 1303 1309 1311
1357 1361 1373 1423 1429 1835 1841 1843 1915 1979 1983]

Number of customer in 2nd group= 146
They are - [299 315 319 331 337 345 349 361 369 373 375 389 411 427
449 451 453 479 487 495 503 519 521 535 537 541 551 563
573 579 585 589 615 617 619 627 629 635 637 639 657 659
687 695 699 701 719 731 733 739 759 761 765 771 779 787
791 793 795 801 821 825 837 847 863 909 913 933 937 965
973 985 993 1011 1025 1027 1037 1049 1055 1083 1089 1095 1111 1113
1125 1129 1133 1135 1137 1155 1159 1169 1183 1191 1197 1227 1233 1271
1273 1277 1305 1307 1317 1319 1321 1327 1353 1355 1363 1365 1367 1371
1375 1377 1409 1411 1439 1457 1471 1475 1489 1803 1811 1815 1851 1855
1861 1865 1867 1869 1877 1885 1887 1893 1897 1899 1917 1933 1935 1937
1957 1959 1963 1981 1987 1993]

Number of customer in 3rd group= 146
They are - [299 315 319 331 337 345 349 361 369 373 375 389 411 427
449 451 453 479 487 495 503 519 521 535 537 541 551 563
573 579 585 589 615 617 619 627 629 635 637 639 657 659
687 695 699 701 719 731 733 739 759 761 765 771 779 787
791 793 795 801 821 825 837 847 863 909 913 933 937 965
973 985 993 1011 1025 1027 1037 1049 1055 1083 1089 1095 1111 1113
1125 1129 1133 1135 1137 1155 1159 1169 1183 1191 1197 1227 1233 1271
1273 1277 1305 1307 1317 1319 1321 1327 1353 1355 1363 1365 1367 1371
1375 1377 1409 1411 1439 1457 1471 1475 1489 1803 1811 1815 1851 1855
1861 1865 1867 1869 1877 1885 1887 1893 1897 1899 1917 1933 1935 1937
1957 1959 1963 1981 1987 1993]

Number of customer in 4rd group= 3

They are - [317 813 1031]

Number of customer in 5rd group= 7

They are - [517 597 815 969 1181 1331 1969]

Câu 4: Dự đoán mức lương của 1 nhân viên dựa vào công việc, level tổ chức, tình trạng hôn nhân, giới tính, số năm làm việc, Rate lương tương ứng

1. Tìm hiểu vấn đề:

- Tìm hiểu ý nghĩa và vị trí của các trường dữ liệu tương ứng theo yêu cầu từ [dataedo](#). Nhóm sẽ chọn các trường ảnh hưởng đến tính lương của một nhân viên
- Chọn các trường dữ liệu sau:
 - + BusinessEntityID
 - + Organization Level
 - + Marital Status
 - + Gender
 - + Rate
 - + Job Title



- Với cột năm kinh nghiệm và lương thì ta sẽ xử lý trong SQL như sau:

- + Năm kinh nghiệm = StartDate - EndDate (trong bảng HumanResources.Employee Department History)
- + Lương = Rate * [Số h làm việc trong tháng]

2. Mô tả trường dữ liệu

Tên cột	Mô tả	Kiểu dữ liệu
BusinessEntityID	Mã định danh của nhân viên	int
Organization Level	Level của tổ chức mà nhân viên đó thuộc về	int
Marital Status	Tình trạng hôn nhân	Char(1) (M - Married, S - Single)
Gender	Giới tính	Char(1) (M - Male, F - Female)
Rate	Tỷ lệ lương theo giờ	float
Năm kinh nghiệm	Số năm nhân viên đó làm công việc	int
Lương	Lương của nhân viên	float

3. Khai phá dữ liệu

Đọc dữ liệu:

Cau 4: Dự đoán lương của nhân viên theo các trường nghề nghiệp, tình trạng hôn nhân, cấp độ trong công ty, giới tính, tiền lương, Số năm làm việc, rate lương

```
[66] #Cau4:
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from datetime import date, timedelta, datetime
from google.colab import files
uploaded = files.upload()
df=pd.read_csv('Cau4.csv')
df = df.iloc[:, 1:]
df.head()
```

Choose Files No file chosen

	BusinessEntityID	JobTitle	MaritalStatus	OrganizationLevel	Gender	TienLuong	SoNamLamViec	Rate
0	1	Chief Executive Officer	S	NaN	M	28112.0000	13	125.5000
1	2	Vice President of Engineering	S	1.0	F	14215.3760	14	63.4615
2	3	Engineering Manager	M	2.0	M	9692.3008	15	43.2692
3	4	Senior Tool Designer	S	3.0	M	6685.5488	12	8.6200
4	4	Senior Tool Designer	S	3.0	M	6685.5488	12	8.6200

Mô tả:

```
[67] #Explore data
df.describe(include='all')
```

	BusinessEntityID	JobTitle	MaritalStatus	OrganizationLevel	Gender	TienLuong	SoNamLamViec	Rate
count	334.000000		334	334	333.000000	334	334.000000	334.000000
unique	NaN		67	2	NaN	2	NaN	NaN
top	NaN	Production Technician - WC30	S	NaN	M	NaN	NaN	NaN
freq	NaN		41	168	NaN	237	NaN	NaN
mean	147.793413		NaN	NaN	3.441441	NaN	4383.862975	12.784431
std	84.625739		NaN	NaN	0.836134	NaN	2878.961818	1.196189
min	1.000000		NaN	NaN	1.000000	NaN	2128.000000	9.000000
25%	74.250000		NaN	NaN	3.000000	NaN	2640.000000	12.000000
50%	157.500000		NaN	NaN	4.000000	NaN	3360.000000	13.000000
75%	224.000000		NaN	NaN	4.000000	NaN	5465.381600	13.000000
max	290.000000		NaN	NaN	4.000000	NaN	28112.000000	125.500000

- cột Organization Level

Độ lệch chuẩn chỉ bằng 0.836 chứng tỏ mức độ trong tổ chức quanh (3.44). Min (1) cách Max (4) rất gần. Vậy range của dữ liệu rất ngắn [1; 4]

- Cột tiền lương



Độ lệch chuẩn bằng 28/8.96 (lớn) có nghĩa là có số tiền lương theo các nhân viên thay đổi nhiều Min (2128) cách Max (28112) xa. Vậy range của dữ liệu rất rộng [2128; 28112]

- Cột số năm làm việc

Độ lệch chuẩn thấp 1.196 chứng tỏ số năm làm việc chênh lệch không nhiều quanh (12 năm). Min (9) cách Max (16) gần. Vậy range của dữ liệu ngắn [9; 16]

- Cột rate

Độ lệch chuẩn bằng 12.496 chứng tỏ mức lương nhân viên trong công ty thay đổi nhiều. Min (6.5) cách Max (125.5). Vậy range của dữ liệu đoạn [6.5; 125.5]



df.info()

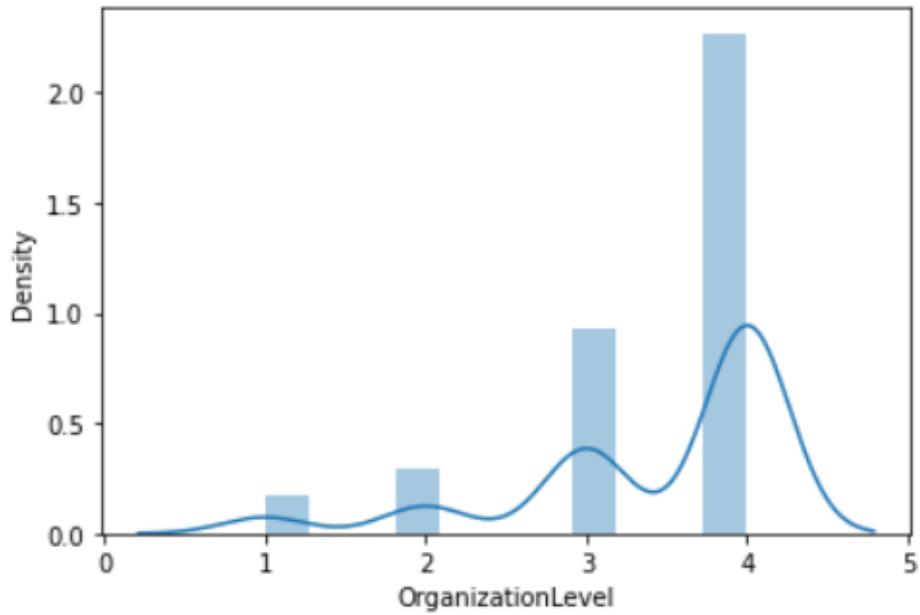
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 334 entries, 0 to 333
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   BusinessEntityID  334 non-null    int64  
 1   JobTitle          334 non-null    object  
 2   MaritalStatus     334 non-null    object  
 3   OrganizationLevel 333 non-null    float64 
 4   Gender            334 non-null    object  
 5   TienLuong         334 non-null    float64 
 6   SoNamLamViec     334 non-null    int64  
 7   Rate              334 non-null    float64 
dtypes: float64(3), int64(2), object(3)
memory usage: 21.0+ KB
```

Có 3 kiểu dữ liệu float, 2 kiểu int, 3 kiểu object, và dữ liệu có 334 entries, riêng Organization Level có 1 dòng null.

[69] #histogram

```
sns.distplot(df[ 'OrganizationLevel']);
```

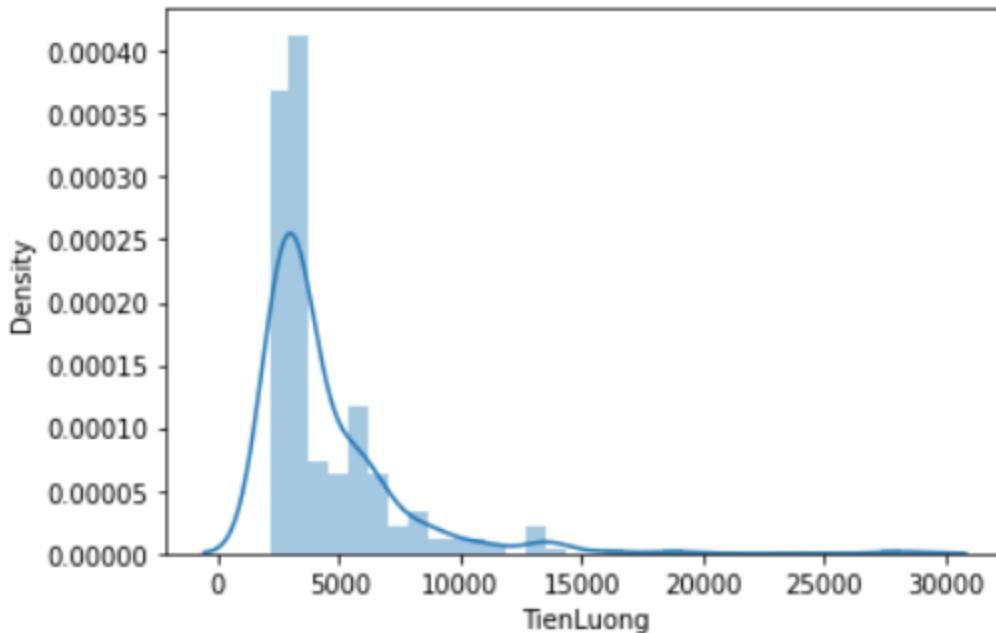
```
↳ /usr/local/lib/python3.7/dist-packages/seaborn/distributions.  
    warnings.warn(msg, FutureWarning)
```



Dữ liệu biểu đồ phân phối lệch phải. Cấp độ trong công ty nhiều nhất ở mức 4.
Mode<Median<Mean

```
[70] sns.distplot(df['TienLuong']);
```

```
↳ /usr/local/lib/python3.7/dist-packages/seaborn/distributi
    warnings.warn(msg, FutureWarning)
```

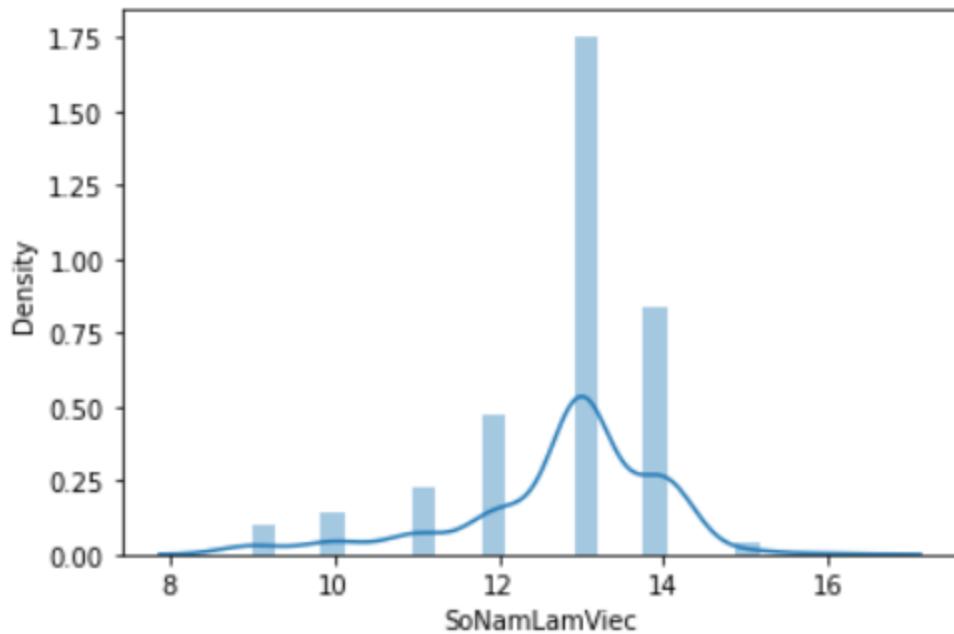


Dữ liệu biểu đồ phân phối lệch trái. Tiền lương nhân viên nằm trong khoảng 3000-5000
đô, thế nhưng vẫn có những nhân viên có tiền lương lên tới 30000 đô.

Mode>Median>Mean

✓ [71] sns.distplot(df['SoNamLamViec']);
0s

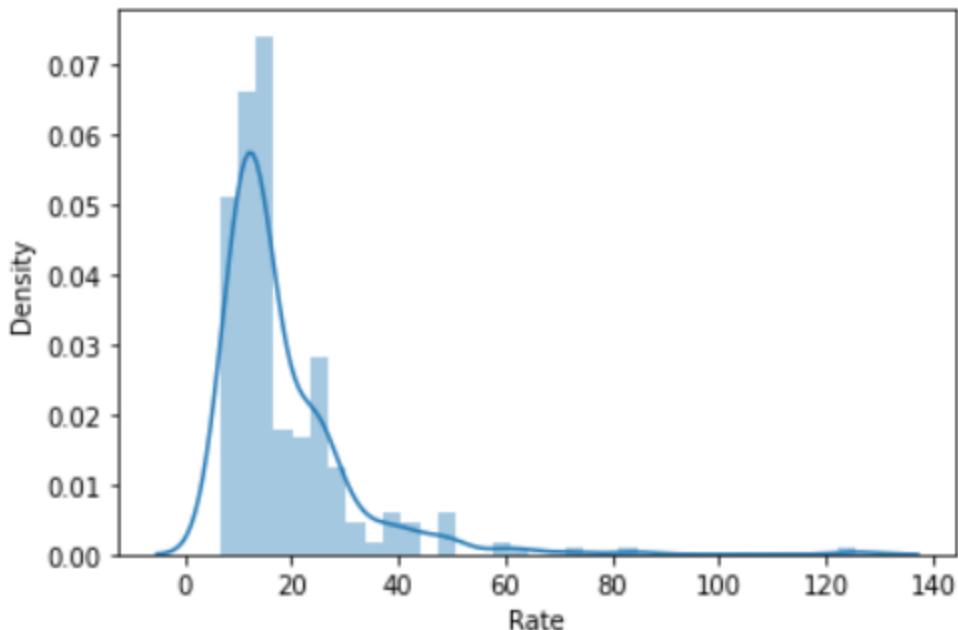
```
/usr/local/lib/python3.7/dist-packages/seaborn/distribut:  
warnings.warn(msg, FutureWarning)
```



Dữ liệu biểu đồ phân phối lệch phải. Năm làm việc của nhân viên nhiều nhất ở 13 năm.
Mode<Median<Mean

```
[1]: sns.distplot(df[ 'Rate' ]);
```

```
[2]: /usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:50: FutureWarning:  
  warnings.warn(msg, FutureWarning)
```



Dữ liệu biểu đồ phân phối lệch trái. Mức lương của nhân viên nhiều trong khoảng 8-20 đô/giờ, có nhân viên lên gần đến 140 đô/giờ. Mode>Median>Mean

4. Tiết xử lý dữ liệu



```
#clean data  
df.isnull().sum()
```

```
BusinessEntityID      0  
JobTitle             0  
MaritalStatus        0  
OrganizationLevel    1  
Gender               0  
TienLuong            0  
SoNamLamViec         0  
Rate                 0  
dtype: int64
```

Hầu hết không có dữ liệu null. Chỉ có cột Organization Level chứa 1 dòng null.

✓
0s



```
df=df.fillna(0)
```

Vì cột Organization Level dạng int và các dữ liệu quanh từ đoạn [1:4] nên điền giá trị 0 vào ô null.

✓
0s



```
df.duplicated().sum()
```

↪ 18

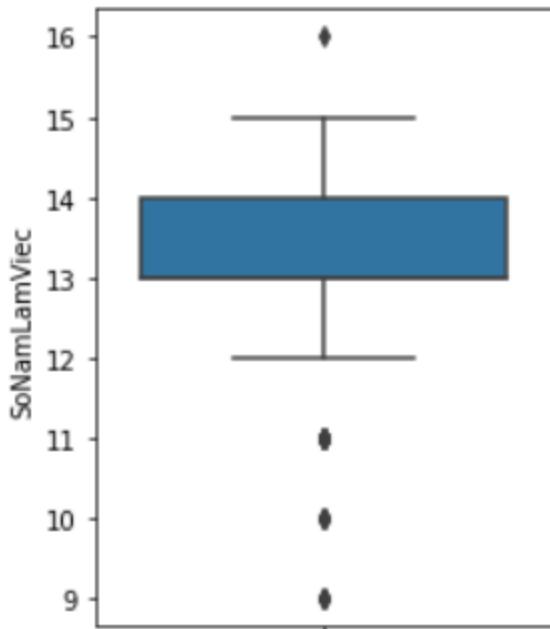
Kiểm tra giá trị trùng thì tổng giá trị trùng là 18 dòng. Xóa các giá trị trùng

✓ [77] df=df.drop_duplicates()
0s df.duplicated().sum()

0

[▶] import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(15,10))
plt.subplot(2,4,1)
sns.boxplot(y='SoNamLamViec',data=df)

⇨ <matplotlib.axes._subplots.AxesSubplot at 0x7f648209d550>



Kiểm tra outlier cột số năm làm việc thì thấy 4 giá trị bị outlier nhưng vẫn hợp lý nên không xử lý outlier. Vì cột BusinessEntityID không ảnh hưởng đến việc dự đoán tiền lương nhân viên nên drop cột BusinessEntityID.



5. Xây dựng mô hình

Ở đây nhóm chọn cột phụ thuộc là TienLuong và các cột còn lại là độc lập.



```
x = df.drop(columns=["TienLuong"]) #independ  
y = df["TienLuong"] #depend
```

Trước hết xử lý các cột dạng string để xây dựng mô hình

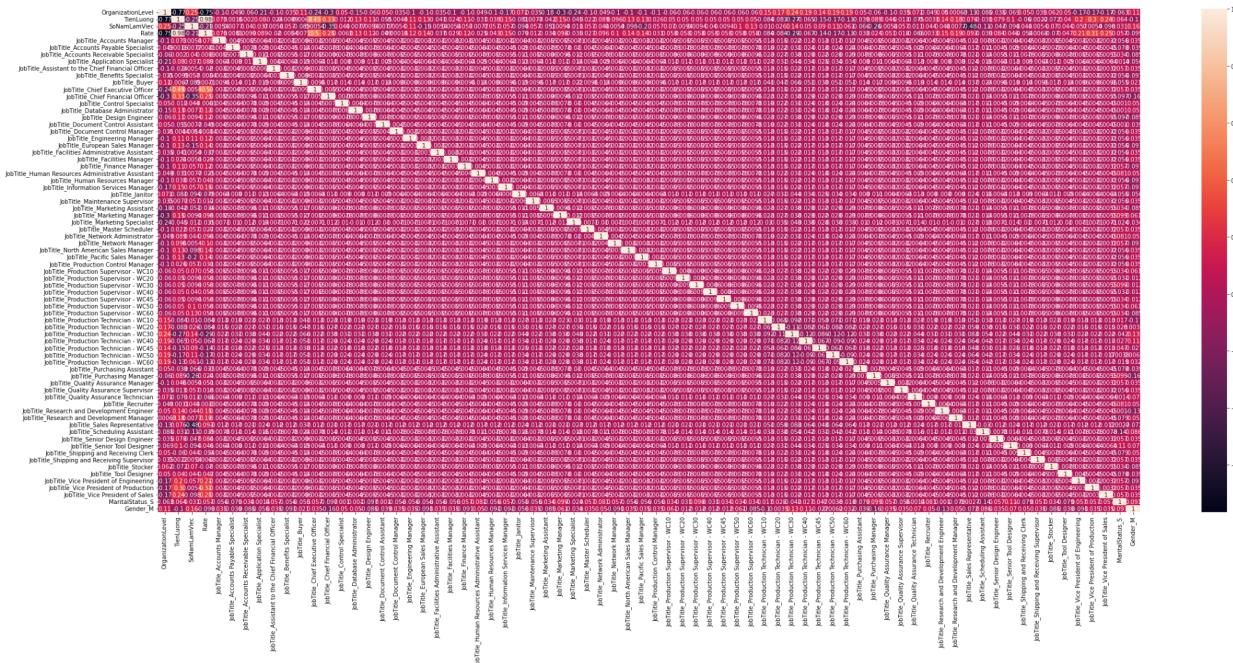
```
[80] #doi qua dang cac du lieu dang string qua dang int  
df = pd.get_dummies(df,drop_first=True)  
df.head()
```

Xây dựng biểu đồ tương quan:

✓
34s



```
plt.figure(figsize = (38,16))  
sns.heatmap(df.corr(), annot = True)  
plt.savefig('heatmap.png')  
plt.show()
```



Vì theo biểu đồ tương quan cột Organization Level có độ tương quan thấp so với Tiền lương nên nhóm drop cột Organization Level.

Xây dựng mô hình Chia dữ liệu train & test: 80:20.

```
#Chia dữ liệu train & test: 80:20.
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=1/4,random_state=0)
```

✓ [85] #Xây dựng model
model = LogisticRegression()



6. Training mô hình và Đánh giá mô hình



```
#Tạo standard x scaler
scaler = preprocessing.StandardScaler().fit(x_train)
x_train_scaled = scaler.transform(x_train)
```

Bạn sẽ gọi phương thức fit_transform với đầu vào là train_X_df để tính các giá trị từ tập huấn luyện (ví dụ, tienluong ở bước thêm và xóa cột, mean và mode ở bước xử lý giá trị thiểu, mean và độ lệch chuẩn ở bước chuẩn hóa) và đồng thời tiền xử lý train_X_df, kết quả trả về sẽ là train_X_df sau khi đã tiền xử lý, là một mảng Numpy, bạn đặt tên là train_X_scaled.

Training model cho tập train



```
#Training the model
model.fit(x_train_scaled, y_train.astype('int'))
```

LogisticRegression()

Đánh giá mô hình tập train



```
#đánh giá mô hình trên training set và xem kết quả:
train_acc = model.score(x_train_scaled, y_train.astype('int'))
print("The Accuracy for Training Set is {}".format(train_acc*100))
```

⇒ The Accuracy for Training Set is 100.0

Độ chính xác hơn 100%. Tốt, nhưng độ chính xác tập train không hữu ích, độ chính xác tập test mới thực sự thành công.

Tạo y predict

```
#Tạo y predict
lm = LogisticRegression(solver='lbfgs',random_state=0)
lm.fit(x_train, y_train.astype('int'))
y_pred = lm.predict(x_test)
y_pred

/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
extra_warning_msg=LOGISTIC_SOLVER_CONVERGENCE_MSG,
array([ 3600, 3230, 3228, 2400, 4092, 5600, 2640, 5600, 5169,
       3228, 3600, 3584, 5600, 2640, 3360, 2400, 2400, 3360,
       2280, 5600, 3230, 3360, 5600, 5600, 3584, 2280, 3600,
      3015, 2640, 5600, 5600, 3230, 2280, 3228, 5600, 4092,
      2640, 9692, 2280, 5600, 5169, 5600, 3600, 2280, 3360,
      3600, 3600, 5169, 2280, 2640, 3360, 2369, 3600, 8400,
      2220, 10774, 3360, 4256, 2640, 2988, 2640, 2400, 3600,
      2280, 5600, 2988, 4092, 2988, 2640, 3228, 5169, 2988,
      2280, 2640, 2640, 3600, 3600, 2280, 5600])
```



#đánh giá mô hình trên test set và xem kết quả:

```
test_acc = accuracy_score(y_test.astype('int'), y_pred)
print("The Accuracy for Test Set is {}".format(test_acc*100))
```

The Accuracy for Test Set is 78.48101265822784

Mức độ tin cậy của tập train(100%) > tập Test(78.48%) khả năng overfitting thấp. Mức độ tin cậy dự đoán TienLuong theo các trường được xấp xỉ 78.48% cũng khá tốt.

✓
0s



```
from sklearn.metrics import r2_score
from sklearn import metrics
test_acc = metrics.r2_score(y_test.astype('int'), y_pred)
print("The R2_score for Test Set is {}".format(test_acc*100))
```

The R2_score for Test Set is 91.31645704123143

Vậy mô hình hồi quy tuyến tính đang được thống kê sẽ phù hợp với dữ liệu ở mức 91.316457%



C. Link Colab và tài liệu tham khảo

1. Link Colab:

câu 4:

[DoAnTH#2.ipynb - Colaboratory \(google.com\)](#)

câu 2:

[https://colab.research.google.com/drive/1XmxFd19Tbsc80yzMx28BqNZ92DgTkwZg?usp=sharing](#)

câu 1:

[https://colab.research.google.com/drive/1ApI1Sx9dNzNTteIOjQsfC79pmY0UDdAm?fbclid=IwAR0vfx8Rh_0eJdBvMtaG3Gqbpt_ByqLkjCLMaVZZnyP_PSgdhYzphK0J4Zc#scrollTo=bFIoCnLBnN72&uniquifier=1](#)

câu 3:

link

2. Tài liệu tham khảo:

[AdventureWorks \(dataedo.com\)](#)

[AdventureWorks sample databases - SQL Server | Microsoft Docs](#)