

BÁO CÁO CUỐI KÌ MÔN KHAI THÁC DỮ LIỆU

CHỦ ĐỀ

DỰ ĐOÁN KHẢ NĂNG CHẤP DỨT HỢP ĐỒNG CỦA NHÂN VIÊN

GVHD :

ThS. Nguyễn Hồ Duy Trí

TS. Cao Thị Nhạn

LỚP : IS252.N11.TMCL



THÀNH VIÊN

Nguyễn Cao Khoa

Bùi Ngọc Thành

Lê Đình Quốc Huy

Trần Minh Tốt





NỘI DUNG

- 01** TỔNG QUAN DỮ LIỆU
- 02** XỬ LÝ DỮ LIỆU
- 03** VISUALIZATION
- 04** CÂN BẰNG DỮ LIỆU
- 05** THỰC HIỆN THUẬT TOÁN
- 06** SO SÁNH THUẬT TOÁN
- 07** XÂY DỰNG PHẦN MỀM DỰ ĐOÁN

TỔNG QUAN DỮ LIỆU

01

01. TỔNG QUAN DỮ LIỆU

- Nguồn dữ liệu : [Modeling the Business Cost of Retention | Kaggle](#)
- Tác giả: James Tollefson
- Tổng số dòng dữ liệu: 1470
- Tổng số thuộc tính: 35
- Đặc điểm tập dữ liệu: Đa biến
- Đặc điểm số thuộc tính: ký tự, số thực, số nguyên
- Giá trị bị mất: không có

01. TỔNG QUAN DỮ LIỆU

WA_Fn-UseC_-HR-Employee-Attrition.csv (227.98 kB)

Detail



Compact

Column

10 of 35 columns

About this file

It contains employee attrition data

# Age	Attrition	BusinessTravel	# DailyRate	Department
Númerica - Discreta	Categórica	Categórica	Númerica - Discreta	Categórica
	<div><div>true</div><div>0 0%</div></div> <div><div>false</div><div>0 0%</div></div>	<div><div>Travel_Rarely</div><div>71%</div></div> <div><div>Travel_Frequently</div><div>19%</div></div> <div><div>Other (150)</div><div>10%</div></div>		<div><div>Research & Develo...</div><div>65%</div></div> <div><div>Sales</div><div>30%</div></div> <div><div>Other (63)</div><div>4%</div></div>
41	Yes	Travel_Rarely	1102	Sales
49	No	Travel_Frequently	279	Research & Development
37	Yes	Travel_Rarely	1373	Research & Development
33	No	Travel_Frequently	1392	Research &

01. TỔNG QUAN DỮ LIỆU

STT	Thuộc tính	Ý nghĩa	Loại	Giá trị
1	Age	Tuổi của nhân viên	Ordinal	Từ 18 - 60
2	Attrition	Sự hao hụt	Nominal	Yes - No
3	Business Travel	Mật độ công tác	Nominal	Travel Rarely, Travel Frequently, Non Travel
4	Daily Rate	Tiền công mỗi ngày	Ordinal	102 - 1499\$
5	Department	Phòng ban	Nominal	Research & Development, Sales, Human Resources
6	Distance From Home	Khoảng cách đi làm	Ordinal	1 - 29km
7	Education	Trình độ học vấn	Nominal	1 - 5
8	Education Field	Chuyên môn	Nominal	Life Science, Medical, Marketing, Technical Degree, Other
9	Employee Count	Số lượng nhân viên	Ordinal	1

01. TỔNG QUAN DỮ LIỆU

STT	Thuộc tính	Ý nghĩa	Loại	Giá trị
10	Employee Number	Số thứ tự nhân viên	Ordinal	1 - 2068
11	Environment Sastifaction	Môi trường làm việc	Ordinal	1 - 5
12	Gender	Giới tính	Nominal	Male - Female
13	Hourly Rate	Tiền công mỗi giờ	Ordinal	30 - 100\$
14	Job Involment	Độ năng suất trong công việc	Ordinal	1 - 4
15	Job Level	Trình độ công việc	Ordinal	1 - 5
16	Job Role	Vai trò công việc	Nominal	Sale Executive, Research Scientist, Laboratory Technician, Manufacturing Director, Healthcare Respentative
17	Job Satisfaction	Độ hài lòng công việc	Ordinal	1 - 4

01. TỔNG QUAN DỮ LIỆU

STT	Thuộc tính	Ý nghĩa	Loại	Giá trị
18	Marial Status	Tình trạng hôn nhân	Nominal	Marriedm - Single - Divorced
19	Monthly Income	Thu nhập mỗi tháng	Ordinal	1009 - 20000\$
20	Monthly Rate	Tiền lương mỗi tháng	Ordinal	2094 - 27000\$
21	NumCompanies Worked	Số công ty từng làm	Ordinal	0 - 9
22	Over18	Hơn 18 tuổi	Nominal	Y - N
23	OverTime	Làm thêm giờ	Nominal	Yes - No
24	Percent Salary Hike	Tăng lương theo phần trăm	Ordinal	11 -25
25	Performance Rating	Đánh giá năng suất	Ordinal	3 - 4

01. TỔNG QUAN DỮ LIỆU

STT	Thuộc tính	Ý nghĩa	Loại	Giá trị
26	Relationship Satisfaction	Sự hài lòng với các mối quan hệ	Ordinal	1 – 4
27	Standard Hours	Giờ làm việc tiêu chuẩn	Ordinal	80
28	Stock Option Level	Mức tùy chọn cổ phiếu	Ordinal	0 – 3
29	Total Working Years	Tổng số năm làm việc	Ordinal	0 – 40
30	Training Times Last Year	Số lần huấn luyện năm ngoái	Ordinal	0 – 6
31	Work Life Balance	Sự cân bằng công việc và cuộc sống	Ordinal	1 – 4
32	Years At Company	Số năm ở công ty	Ordinal	0 – 40
33	Years In Current Role	Số năm làm ở chức vụ hiện tại	Ordinal	0 - 18

01. TỔNG QUAN DỮ LIỆU

STT	Thuộc tính	Ý nghĩa	Loại	Giá trị
34	Years Since Last Promotion	Số năm kể từ lần thăng chức gần nhất	Ordinal	0 - 15
35	Years With Current Manager	Số năm làm việc với quản lý hiện tại	Ordinal	0 - 17

XỬ LÝ DỮ LIỆU

02

02. XỬ LÝ DỮ LIỆU

THÊM THƯ VIỆN

```
[ ] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
import time
from datetime import timedelta
import sklearn.metrics as metrics
from sklearn.preprocessing import LabelEncoder as LE
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

from tqdm import tqdm
from sklearn.preprocessing import LabelEncoder
from datetime import datetime
```

02. XỬ LÝ DỮ LIỆU

ĐỌC DỮ LIỆU

```
df_hr = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/HR_Employee_Attrition.csv')
df_hr
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...
...
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	2061	...
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	2062	...
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	2064	...
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	2065	...
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	1	2068	...

1470 rows x 35 columns

02. XỬ LÝ DỮ LIỆU

CHECK KIỂU DỮ LIỆU

df_hr.dtypes	
Age	int64
Attrition	object
BusinessTravel	object
DailyRate	int64
Department	object
DistanceFromHome	int64
Education	int64
EducationField	object
EmployeeCount	int64
EmployeeNumber	int64
EnvironmentSatisfaction	int64
Gender	object
HourlyRate	int64
JobInvolvement	int64
JobLevel	int64
JobRole	object
JobSatisfaction	int64
MaritalStatus	object
MonthlyIncome	int64
MonthlyRate	int64
NumCompaniesWorked	int64
Over18	object
OverTime	object
PercentSalaryHike	int64
PerformanceRating	int64
RelationshipSatisfaction	int64
StandardHours	int64
StockOptionLevel	int64
TotalWorkingYears	int64
TrainingTimesLastYear	int64
WorkLifeBalance	int64
YearsAtCompany	int64
YearsInCurrentRole	int64
YearsSinceLastPromotion	int64
YearsWithCurrManager	int64
dtype:	object

02. XỬ LÝ DỮ LIỆU

CHECK NULL

```
[ ] df_hr.isnull().sum().sort_values(ascending=False)
```

Age	0
StandardHours	0
NumCompaniesWorked	0
Over18	0
OverTime	0
PercentSalaryHike	0
PerformanceRating	0
RelationshipSatisfaction	0
StockOptionLevel	0
MonthlyIncome	0
TotalWorkingYears	0
TrainingTimesLastYear	0
WorkLifeBalance	0
YearsAtCompany	0
YearsInCurrentRole	0
YearsSinceLastPromotion	0
MonthlyRate	0
MaritalStatus	0
Attrition	0
EmployeeCount	0
BusinessTravel	0
DailyRate	0
Department	0
DistanceFromHome	0
Education	0
EducationField	0
EmployeeNumber	0
JobSatisfaction	0
EnvironmentSatisfaction	0
Gender	0
HourlyRate	0
JobInvolvement	0
JobLevel	0
JobRole	0
YearsWithCurrManager	0
dtype: int64	

02. XỬ LÝ DỮ LIỆU

THÔNG TIN DỮ LIỆU KIỂU SỐ

```
df_hr.describe(include=['int64'])
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	...
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	...
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	2.063946	...
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	1.106940	...
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	1.000000	...
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	1.000000	...
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	...
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	...
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	...

8 rows × 26 columns

02. XỬ LÝ DỮ LIỆU

THÔNG TIN DỮ LIỆU KIỂU CHUỖI

```
[ ] df_hr.describe(include=['O'])
```

	Attrition	BusinessTravel	Department	EducationField	Gender	JobRole	MaritalStatus	Over18	OverTime
count	1470	1470	1470	1470	1470	1470	1470	1470	1470
unique	2	3	3	6	2	9	3	1	2
top	No	Travel_Rarely	Research & Development	Life Sciences	Male	Sales Executive	Married	Y	No
freq	1233	1043	961	606	882	326	673	1470	1054

02. XỬ LÝ DỮ LIỆU

PHÂN LOẠI CÁC THUỘC TÍNH ĐỂ XỬ LÝ DỮ LIỆU

```
[ ] continuous_features = ['Age', 'DailyRate', 'DistanceFromHome', 'HourlyRate', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',  
                           'PercentSalaryHike', 'StandardHours', 'TotalWorkingYears', 'TrainingTimesLastYear', 'YearsAtCompany',  
                           'YearsInCurrentRole', 'YearsSinceLastPromotion', 'YearsWithCurrManager']  
categorical_features = ['Attrition', 'BusinessTravel', 'Department', 'Education', 'EducationField', 'EmployeeCount', 'EmployeeNumber',  
                        'EnvironmentSatisfaction', 'Gender', 'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',  
                        'MaritalStatus', 'Over18', 'OverTime', 'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLevel',  
                        'WorkLifeBalance']
```

Chia thuộc tính thành 2 nhóm : biến rời rạc, biến liên tục

02. XỬ LÝ DỮ LIỆU

DÙNG LABEL ENCODER ĐỂ CHUYỂN THUỘC TÍNH OBJECT SANG NUMERIC

```
[ ] le = LabelEncoder()
    l1 = []; l2 = []; text_categorical_features = []
    print('Label Encoder Transformation')
    for i in tqdm(categorical_features):
        if type(df_hr[i][0]) == str:
            text_categorical_features.append(i)
            df_hr[i] = le.fit_transform(df_hr[i])
            l1.append(list(df_hr[i].unique())); l2.append(list(le.inverse_transform(df_hr[i].unique())))
            print(i, ' : ', df_hr[i].unique(), ' = ', le.inverse_transform(df_hr[i].unique()))
```

Label Encoder Transformation
100%|██████████| 20/20 [00:00<00:00, 106.96it/s]Attrition : [1 0] = ['Yes' 'No']
BusinessTravel : [2 1 0] = ['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']
Department : [2 1 0] = ['Sales' 'Research & Development' 'Human Resources']
EducationField : [1 4 3 2 5 0] = ['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical Degree'
'Human Resources']
Gender : [0 1] = ['Female' 'Male']
JobRole : [7 6 2 4 0 3 8 5 1] = ['Sales Executive' 'Research Scientist' 'Laboratory Technician'
'Manufacturing Director' 'Healthcare Representative' 'Manager'
'Sales Representative' 'Research Director' 'Human Resources']
MaritalStatus : [2 1 0] = ['Single' 'Married' 'Divorced']
Over18 : [0] = ['Y']
OverTime : [1 0] = ['Yes' 'No']

02. XỬ LÝ DỮ LIỆU

DỮ LIỆU TRƯỚC VÀ SAU KHI ENCODING

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction	StandardHours
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1	80
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4	80
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2	80
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	3	80
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	4	80

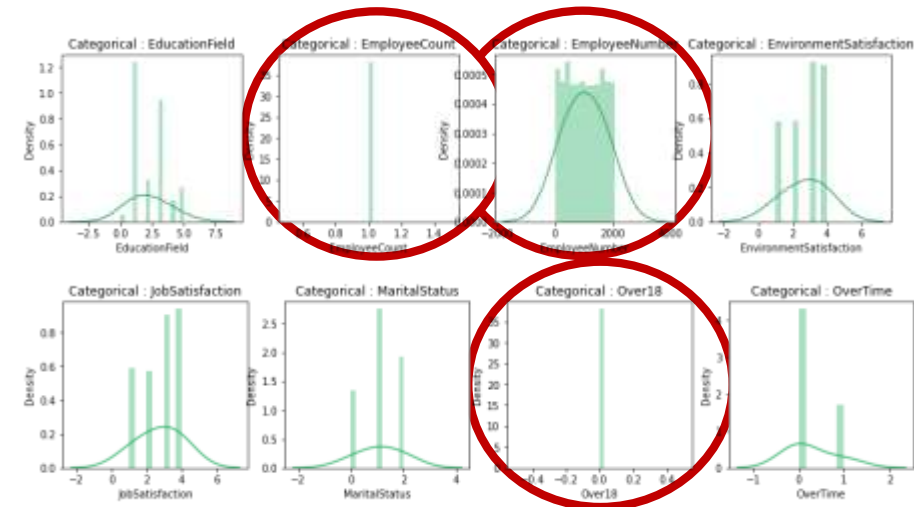
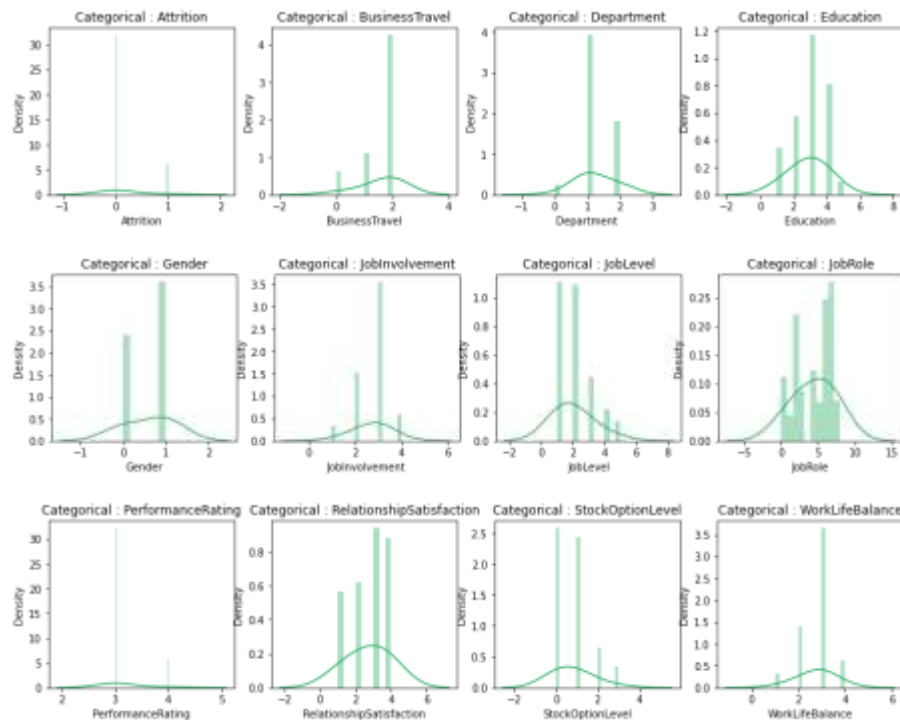
Trước

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction	StandardHours
0	41	1	2	1102	2	1	2	1	1	1	...	1	80
1	49	0	1	279	1	8	1	1	1	2	...	4	80
2	37	1	2	1373	1	2	2	4	1	4	...	2	80
3	33	0	1	1392	1	3	4	1	1	5	...	3	80
4	27	0	2	591	1	2	1	3	1	7	...	4	80

Sau

02. XỬ LÝ DỮ LIỆU

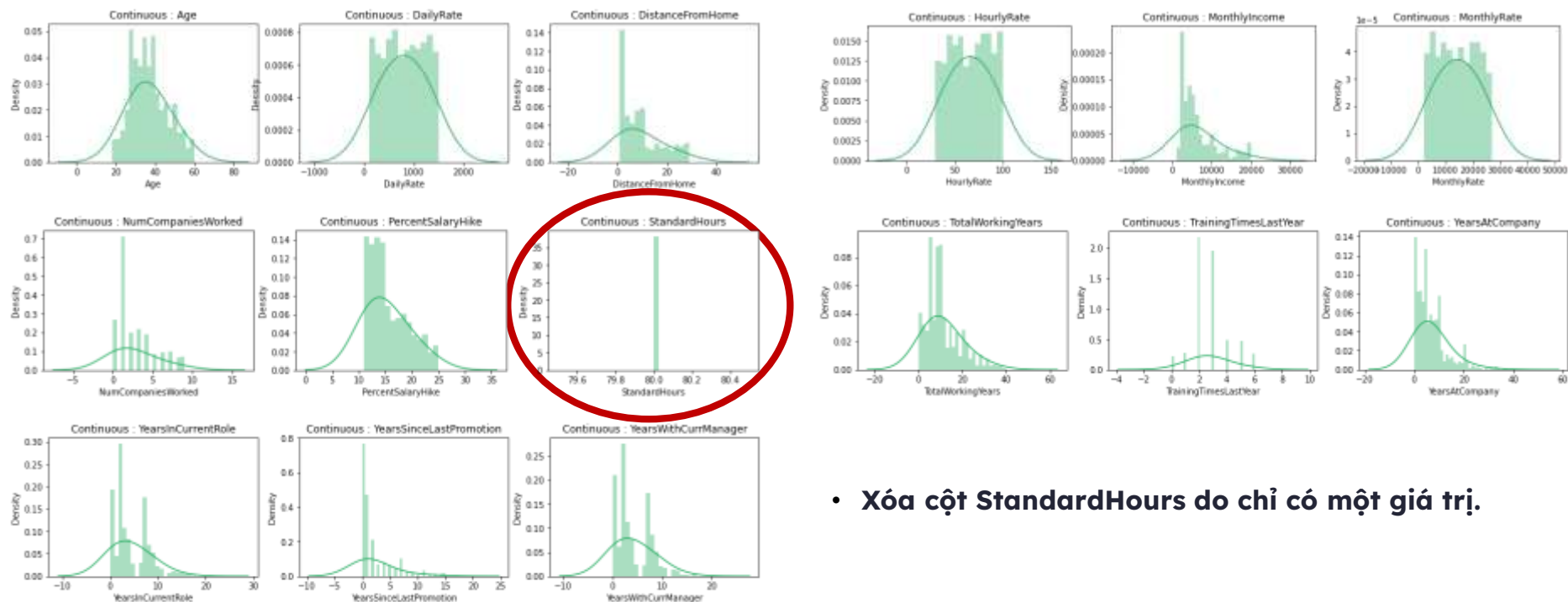
TỔNG QUAN GIÁ TRỊ CỦA DỮ LIỆU RỜI RẠC



- Xóa cột Over18 và EmployeeCount do chỉ có một giá trị.
- Xóa cột EmployeeNumber do chỉ là những con số tự nhiên không cần thiết.

02. XỬ LÝ DỮ LIỆU

TỔNG QUAN GIÁ TRỊ CỦA DỮ LIỆU LIÊN TỤC



- Xóa cột StandardHours do chỉ có một giá trị.

VISUALIZATION



03

03. VISUALIZATION

PHÂN LOẠI CÁC THUỘC TÍNH ĐỂ TRỰC QUAN HÓA DỮ LIỆU

```
[ ] # Thông tin cơ bản của nhân viên
gr1 = ['Age', 'Gender', 'MaritalStatus', 'Education',
       'DistanceFromHome', 'TotalWorkingYears', 'NumCompaniesWorked']

# Thông tin công việc của nhân viên
gr2 = ['EducationField', 'Department', 'JobLevel', 'JobRole',
       'JobInvolvement', 'OverTime', 'JobSatisfaction']

# Thông tin giữa nhân viên và công ty
gr3 = ['YearsAtCompany', 'YearsInCurrentRole', 'YearsWithCurrManager',
       'YearsSinceLastPromotion', 'TrainingTimesLastYear', 'WorkLifeBalance']

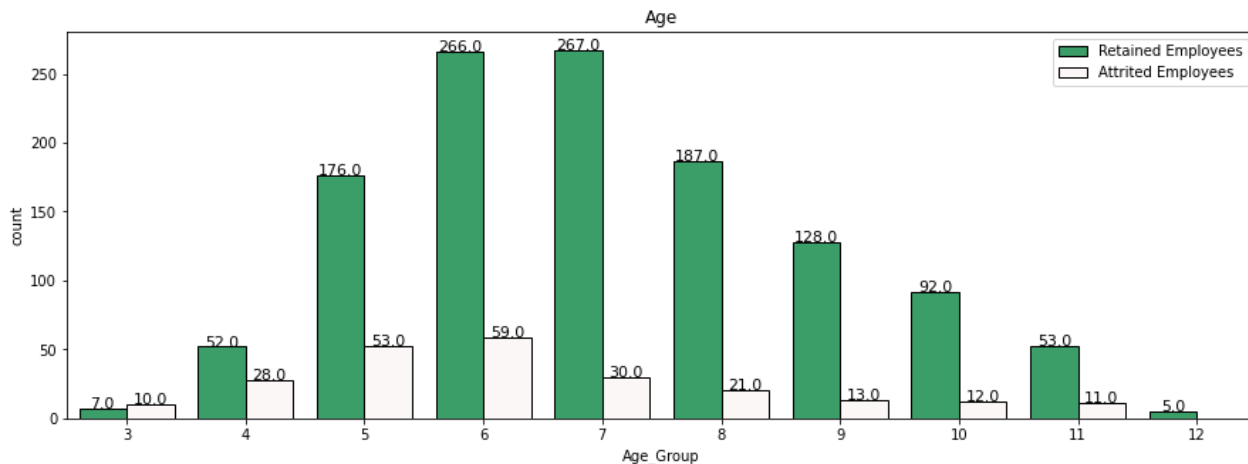
# Thông tin của công ty
gr4 = ['PercentSalaryHike', 'StockOptionLevel', 'BusinessTravel',
       'PerformanceRating', 'EnvironmentSatisfaction', 'RelationshipSatisfaction']

# Tài chính
gr5 = ['MonthlyIncome', 'HourlyRate', 'DailyRate', 'MonthlyRate']

df = pd.DataFrame()
df['Attrition'] = df_hr['Attrition']
```

03. VISUALIZATION

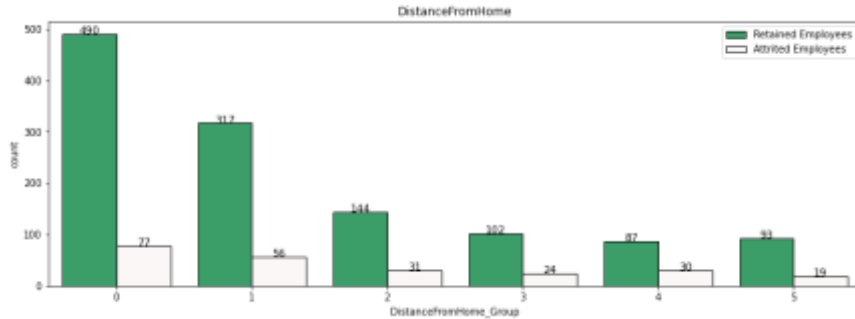
ĐÁNH GIÁ TỔNG QUAN GIỮA ĐỘ TUỔI (AGE) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)



- Sự tiêu hao xuất hiện ở hầu hết các nhóm tuổi.
- Đối với các giá trị Độ tuổi từ 30 - 34, số lượng nhân viên nghỉ việc cao nhất là 59 nhân viên.
- Độ tuổi từ 25 - 29 đứng thứ hai với 53 nhân viên nghỉ việc tại công ty.
- Các giá trị tuổi từ 20 - 24 & 35 - 40 có số lượng gần bằng nhau với 28 & 30.
- Nhân viên trên 40 tuổi có thể là những người đã được miễn nhiệm.

03. VISUALIZATION

ĐÁNH GIÁ TỔNG QUAN GIỮA KHOẢNG CÁCH ĐI LÀM (DistanceFromHome) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)

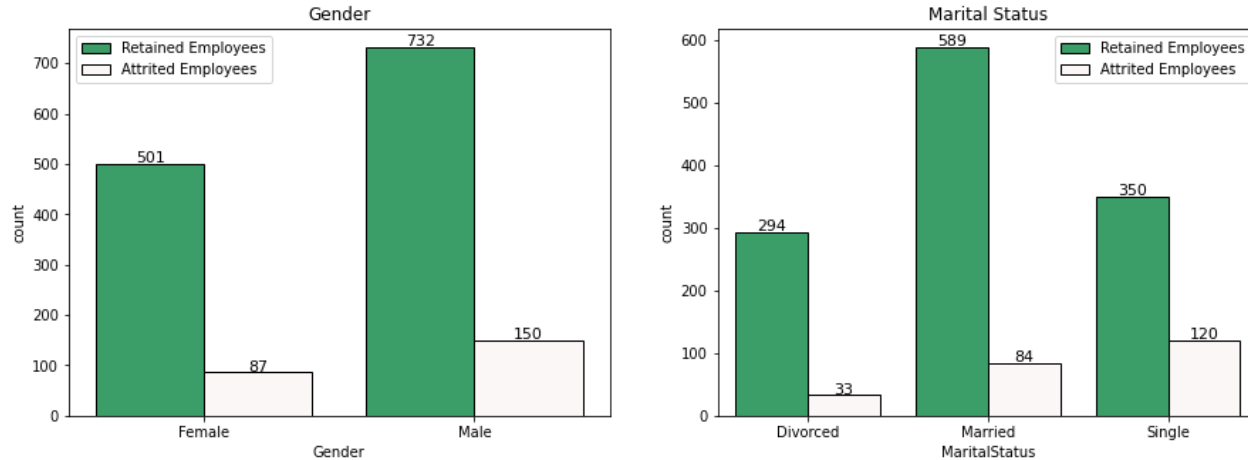


- Xét biểu đồ cột thì có thể thấy nhân viên ở nhóm ở gần công ty nhất (nhóm 0) có số lượng nhân viên rời đi nhiều nhất.
- Tuy nhiên khi kiểm tra tỷ lệ phần trăm tiêu hao thì ta có thể thấy nhân viên sống trong khoảng cách từ 0 - 4 ít bị tiêu hao nhất.
- Khi giá trị của DistanceFromHome tăng lên, tỷ lệ tiêu hao của nhân viên cũng tăng lên.



03. VISUALIZATION

ĐÁNH GIÁ TỔNG QUAN GIỮA GIỚI TÍNH, TRÌNH TRẠNG HÔN NHÂN (Gender, MaritalStatus) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)

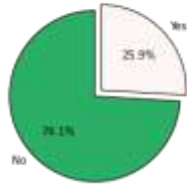


- Số lượng nhân viên Nam bị sa thải nhiều hơn số lượng nhân viên Nữ.
- Nhân viên độc thân có số lượng tiêu hao nhiều nhất.
- Nhân viên đã kết hôn chiếm vị trí thứ 2.
- Đã ly hôn đứng ở vị trí cuối cùng.

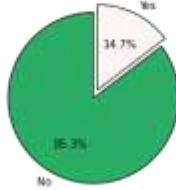
03. VISUALIZATION

ĐÁNH GIÁ TỔNG QUAN GIỮA CHUYÊN MÔN (EducationField) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)

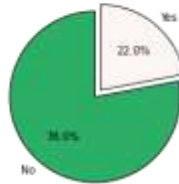
EducationField : Human Resources



EducationField : Life Sciences



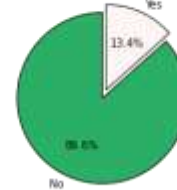
EducationField : Marketing



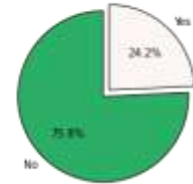
EducationField : Medical



EducationField : Other



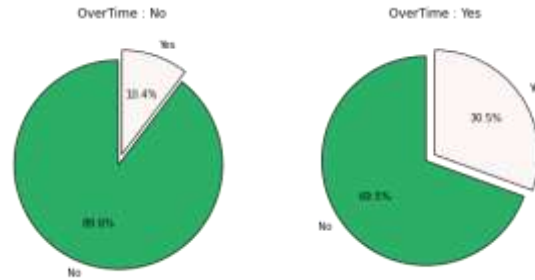
EducationField : Technical Degree



- Chúng ta có thể thấy rằng những nhân viên có Education Field là Human Resources, Technical Degree & Marketing có cơ hội bị loại bỏ cao hơn.

03. VISUALIZATION

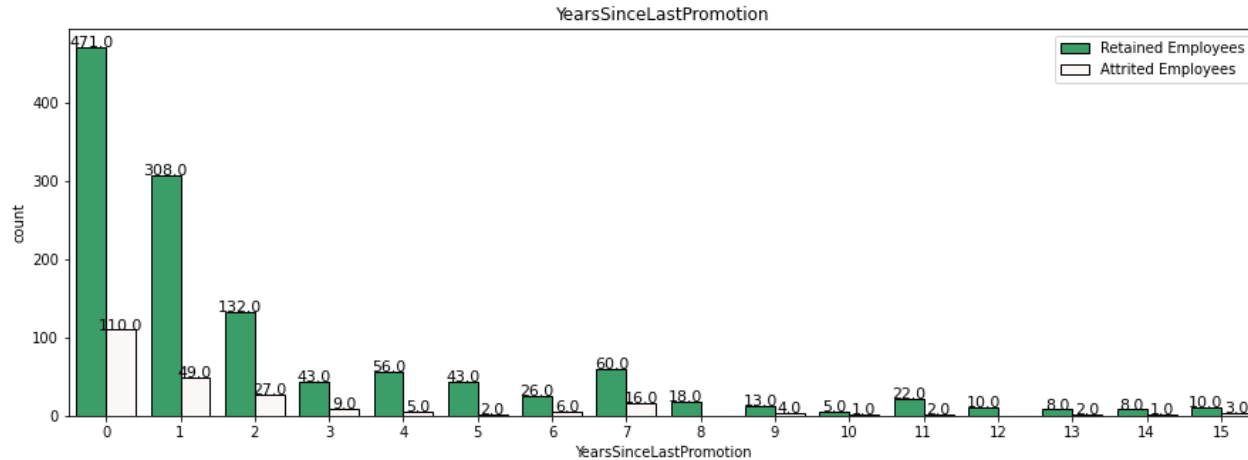
ĐÁNH GIÁ TỔNG QUAN GIỮA LÀM THÊM GIỜ (OverTime) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)



- Chúng ta có thể thấy rằng những người làm việc quá giờ sẽ có khả năng rời bỏ công ty cao do áp lực.
- Tỷ lệ nghỉ việc là 30%, tức là cao gấp 3 lần so với những nhân viên không làm việc ngoài giờ.

03. VISUALIZATION

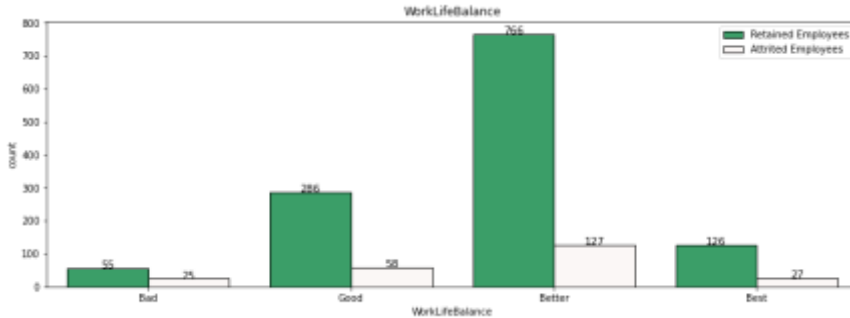
ĐÁNH GIÁ TỔNG QUAN GIỮA SỐ NĂM LÀM VIỆC KỂ TỪ KHI ĐƯỢC THĂNG CHỨC (YearsSinceLastPromotion) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)



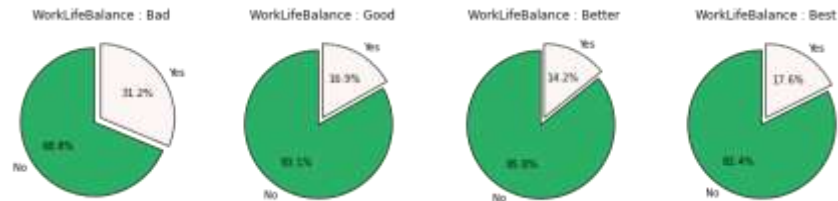
- Chúng ta có thể thấy rằng có một số lượng lớn các trường hợp tiêu hao cho giá trị năm là 0 => đó có thể là giá trị đa số đại diện cho những người mới vào công ty.
- 1 & 2 năm kể từ lần thăng chức cuối cùng cũng ghi nhận một số lượng đáng kể các trường hợp nhân viên rời đi.
- 7 năm kể từ lần thăng chức cuối cùng cũng có khá nhiều trường hợp nhân viên rời đi.

03. VISUALIZATION

ĐÁNH GIÁ TỔNG QUAN GIỮA CÂN BẰNG CUỘC SỐNG (WorkLifeBalance) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)

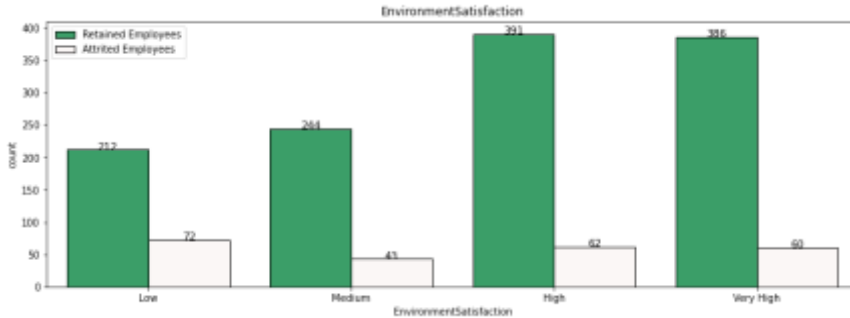


- Đúng như dự đoán, WorkLifeBalance không tốt đã dẫn đến tỷ lệ tiêu hao lớn là 31,2%.
- Đáng ngạc nhiên là Best WorkLifeBalance có giá trị phần trăm tiêu hao cao thứ hai.

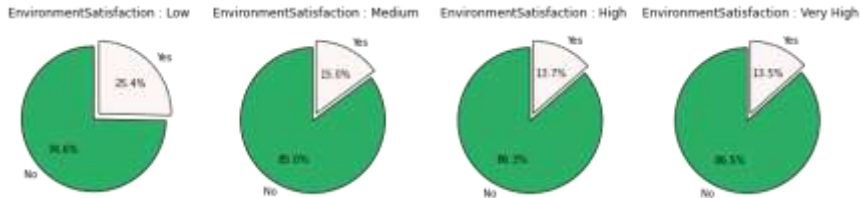


03. VISUALIZATION

ĐÁNH GIÁ TỔNG QUAN GIỮA ĐỘ HÀI LÒNG MÔI TRƯỜNG LÀM VIỆC (EnvironmentSatisfaction) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)

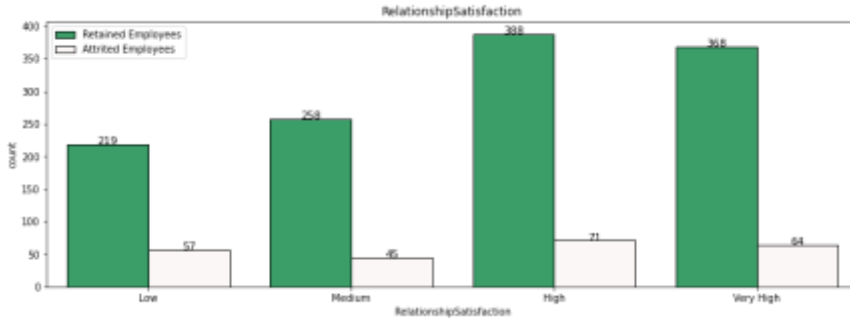


- Các giá trị Mức độ hài lòng về môi trường Cao & Rất cao đã được ghi nhận nhiều lần nhất.
- Đúng như dự đoán, chúng có tỷ lệ tiêu hao thấp so với Mức độ hài lòng về môi trường thấp và trung bình.
- Tỷ lệ tiêu hao được cải thiện khi Sự hài lòng về Môi trường được cải thiện

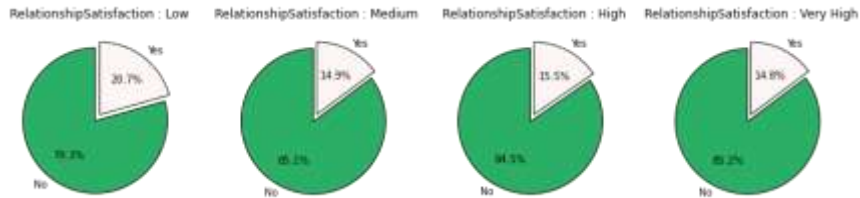


03. VISUALIZATION

ĐÁNH GIÁ TỔNG QUAN GIỮA ĐỘ HÀI LÒNG CÁC MỐI QUAN HỆ (RelationshipSatisfaction) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)

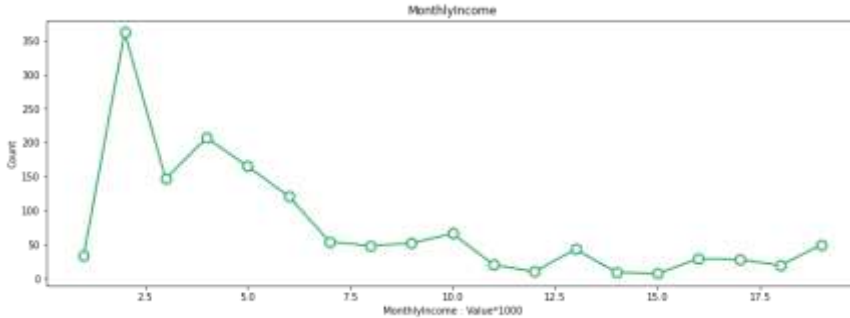


- Biểu đồ ở trên về Sự hài lòng về Mối quan hệ rất giống với Sự hài lòng về Môi trường làm việc.
- Khi các giá trị của Sự hài lòng về Mối quan hệ được cải thiện, tỷ lệ tiêu hao sẽ giảm.

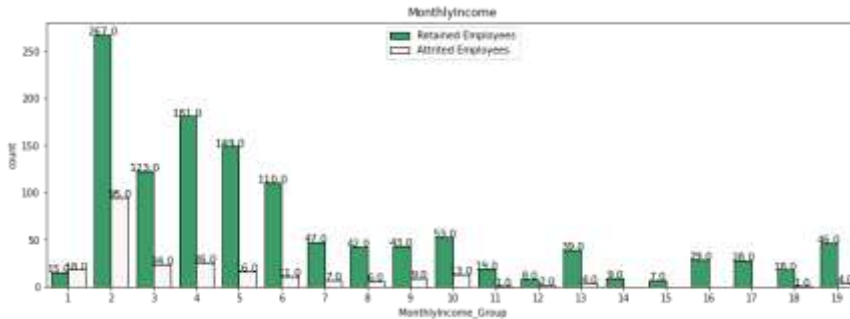


03. VISUALIZATION

ĐÁNH GIÁ TỔNG QUAN GIỮA THU NHẬP HÀNG THÁNG (MonthlyIncome) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)

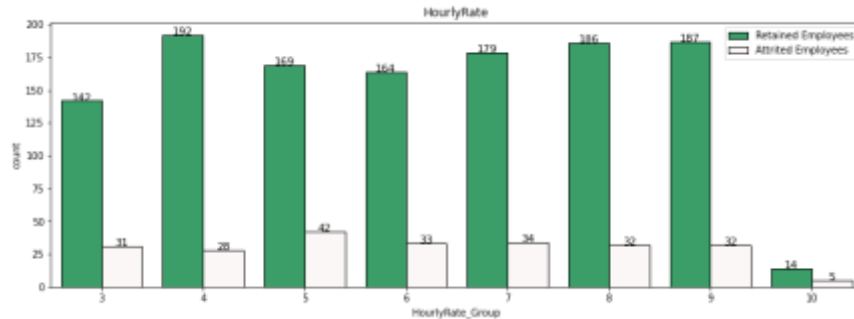
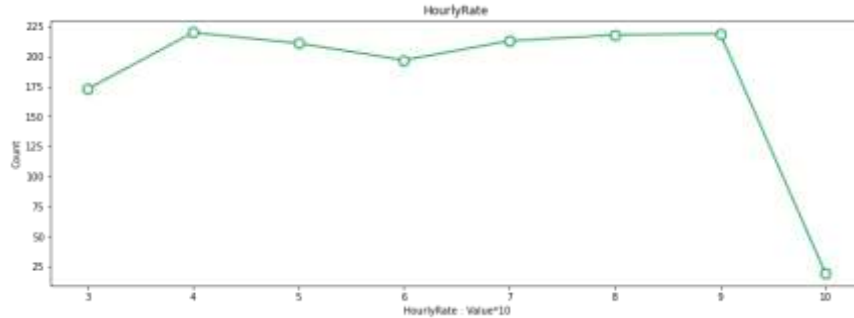


- Biểu đồ nêu bật sự suy giảm tổng thể về số lượng các giá trị.
- Các giá trị Thu nhập hàng tháng trong khoảng 1000 - 2000 hiện diện với số lượng lớn.
- Các giá trị trong khoảng 3000 - 4000 đứng thứ hai với hơn 200 giá trị có trong phạm vi này.



03. VISUALIZATION

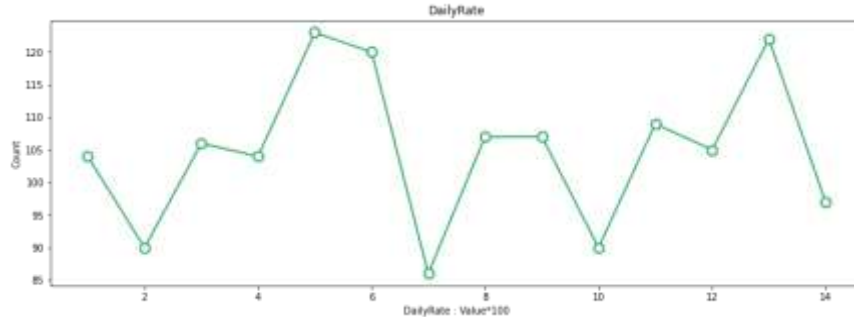
ĐÁNH GIÁ TỔNG QUAN GIỮA TIỀN LƯƠNG THEO GIỜ (HourlyRate) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)



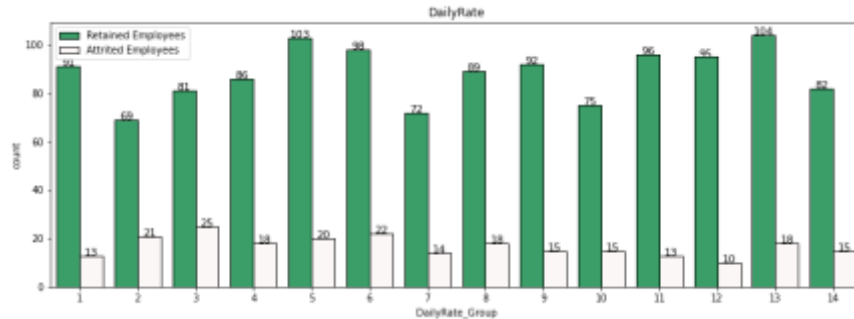
- Đối với HourlyRate, các giá trị từ 30 - 100 xuất hiện với tổng số hơn 175+ mỗi giá trị.
- Tỷ lệ tiêu hao của các giá trị này cũng thấp và rất gần nhau.
- Đối với HourlyRate lớn hơn 100, có rất ít giá trị và do đó mức tiêu hao cũng cao.

03. VISUALIZATION

ĐÁNH GIÁ TỔNG QUAN GIỮA TIỀN LƯƠNG THEO NGÀY (DailyRate) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)

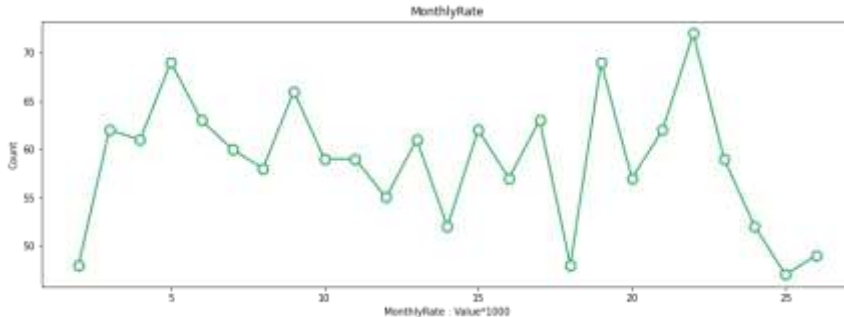


- Số lượng nhân viên tiêu hao gần nhau. Có sự sụt giảm nhất định về số lượng giá trị.
- Các giá trị từ 600 - 700 có số lượng thấp nhất.

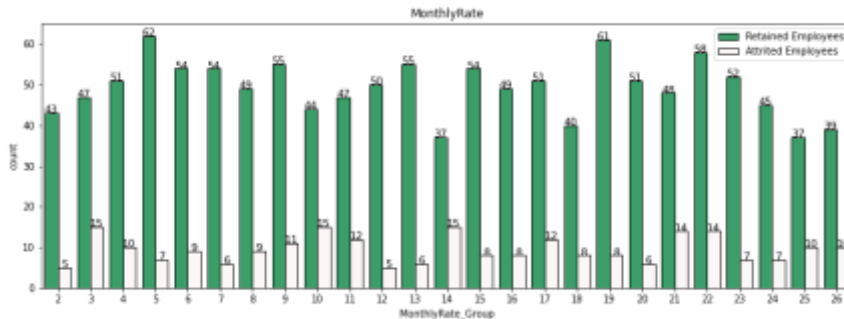


03. VISUALIZATION

ĐÁNH GIÁ TỔNG QUAN GIỮA TIỀN LƯƠNG THEO THÁNG (MonthlyRate) VỚI THUỘC TÍNH QUYẾT ĐỊNH (ATTRITION)



- Giá trị của tiêu hao nhân lực rất gần nhau. Các giá trị trong khoảng 21000 - 22000 có số lượng cao nhất.

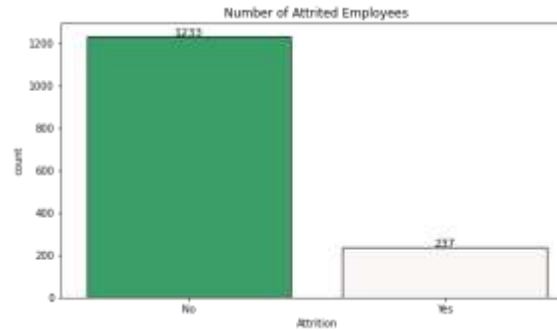


CÂN BẰNG DỮ LIỆU

04

04. CÂN BẰNG DỮ LIỆU

KIỂM TRA CÂN BẰNG CỦA DỮ LIỆU



- Bộ dữ liệu không cân bằng với xu hướng nghiêng về Nhân viên ở lại công ty theo tỷ lệ 5,2 : 1 đối với Nhân viên ở lại : Nhân viên được rời đi.
=> cân bằng tập dữ liệu bằng Phân tích SMOTE để có thể đưa ra dự đoán tốt nhất.

04. CÂN BẰNG DỮ LIỆU

DÙNG PHƯƠNG PHÁP SMOTE ĐỂ CÂN BẰNG LẠI DỮ LIỆU

```
[194] import imblearn
      from collections import Counter
      from imblearn.over_sampling import SMOTE
      from imblearn.under_sampling import RandomUnderSampler
      from imblearn.pipeline import Pipeline

[195] cols = list(df_hr.columns)
      cols.remove('Attrition')

      over = SMOTE(sampling_strategy = 0.85)
      #under = RandomUnderSampler(sampling_strategy = 0.1)
      f1 = df_hr.loc[:,cols]
      t1 = df_hr.loc[:, 'Attrition']

      steps = [('over', over)]
      pipeline = Pipeline(steps=steps)
      f1, t1 = pipeline.fit_resample(f1, t1)
      Counter(t1)

      Counter({1: 1048, 0: 1233})
```

- Cân bằng dữ liệu bằng cách tăng số mẫu của nhóm thiểu số.
- Dữ liệu sau khi cân bằng (1048 đối với nhân viên rời công ty và 1233 với nhân viên ở lại)

04. CÂN BẰNG DỮ LIỆU

DATAFRAME SAU KHI ĐÃ CÂN BẰNG

```
[ ] #DataFrame đã cân bằng
f1
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRate	...
0	41	2	1102	2	1	2	1	2	0	94	...
1	49	1	279	1	8	1	1	3	1	61	...
2	37	2	1373	1	2	2	4	4	1	92	...
3	33	1	1392	1	3	4	1	4	0	56	...
4	27	2	591	1	2	1	3	1	1	40	...
...
2276	28	2	1000	1	9	3	2	2	0	81	...
2277	42	2	980	1	1	2	1	1	0	95	...
2278	42	2	1119	1	9	4	1	3	1	92	...
2279	31	2	424	1	4	4	4	3	1	69	...
2280	27	1	311	1	5	3	1	1	0	62	...

2281 rows x 30 columns

Dữ liệu sau khi cân bằng (không có thuộc tính quyết định)

04. CÂN BẰNG DỮ LIỆU

THUỘC TÍNH QUYẾT ĐỊNH SAU KHI ĐÃ CÂN BẰNG

```
#Thuộc tính quyết định đã được tách  
t1
```

0	1
1	0
2	1
3	0
4	0
...	
2276	1
2277	1
2278	1
2279	1
2280	1

Name: Attrition, Length: 2281, dtype: int64

THỰC HIỆN THUẬT TOÁN



05

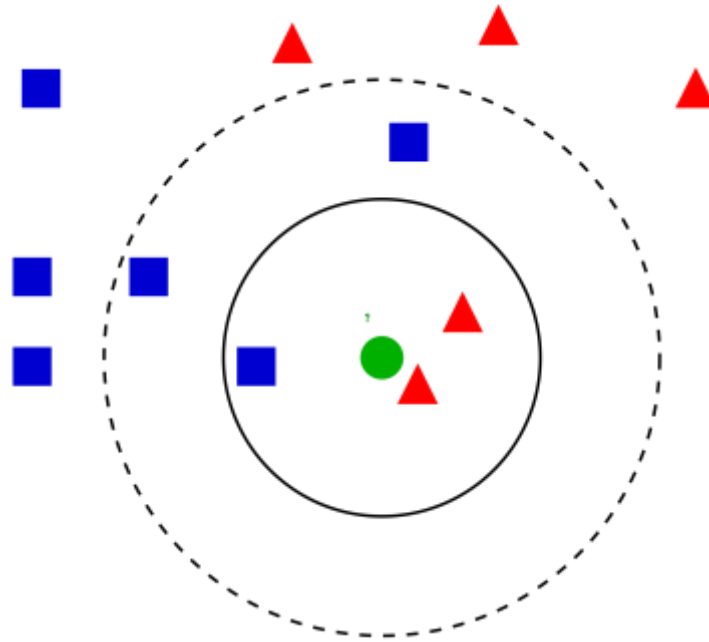
05. THỰC HIỆN THUẬT TOÁN

TÁCH DỮ LIỆU TRAIN TEST

```
[ ] # Tách dữ liệu train và test (dữ liệu train=70%, test=30%)  
    X_train, X_test, y_train, y_test = train_test_split(f1, t1, test_size=0.3, random_state=10)
```

05. THỰC HIỆN THUẬT TOÁN

K-NEAREST NEIGHBOR



05. THỰC HIỆN THUẬT TOÁN

THỰC HIỆN THUẬT TOÁN K-NEAREST NEIGHBOR

```
from sklearn.neighbors import KNeighborsClassifier
import time
from sklearn.metrics import classification_report
from datetime import timedelta
import matplotlib.pyplot as plt
import sklearn.metrics as metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

start_knn = time.time()
knn_scores = []

for i in range(1,12):
    knnc = KNeighborsClassifier(i)
    knn_pred = knnc.fit(X_train, y_train).predict(X_test)
    knn_scores.append(metrics.accuracy_score(y_test, knn_pred))
    max_knn_score = max(knn_scores)

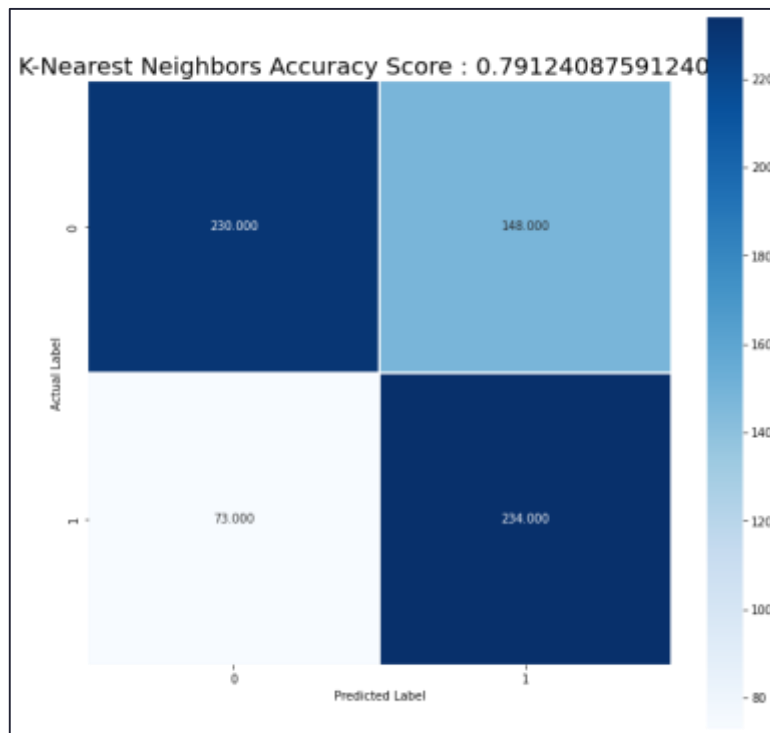
knn_score_ind = [i for i, v in enumerate(knn_scores) if v == max_knn_score]
end_knn = time.time()
times_knn = timedelta(seconds=round(end_knn - start_knn,4)).total_seconds()
print('Highest Accuracy Score : {}% with k = {}'.format(max_knn_score*100, list(map(lambda x: x + 1, knn_score_ind))))
print('Time', times_knn)
knn_score = max_knn_score
accuracies_max_knn = knn_score
print("Accuracy", accuracies_max_knn)
print("Report", metrics.classification_report(y_test, knn_pred))
```

```
□ Highest Accuracy Score : 79.12408759124088% with k = [1]
Time 0.6847
Accuracy 0.7912408759124088
```

report	precision	recall	f1-score	support
0	0.76	0.61	0.68	378
1	0.61	0.76	0.68	307
accuracy			0.68	685
macro avg	0.69	0.69	0.68	685
weighted avg	0.69	0.68	0.68	685

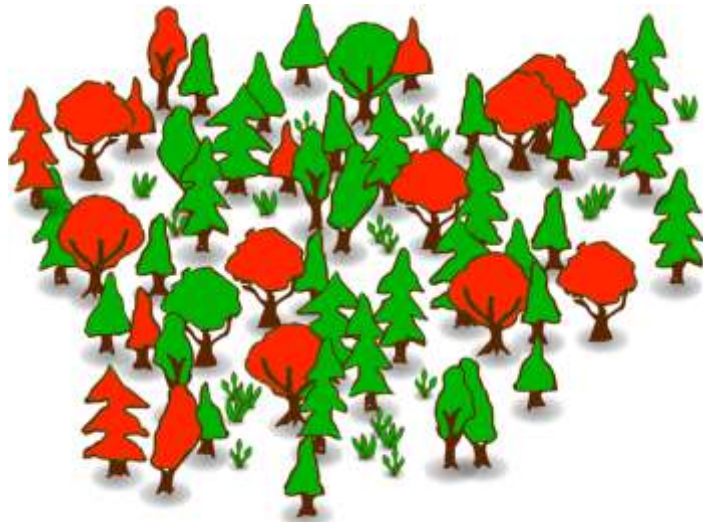
05. THỰC HIỆN THUẬT TOÁN

MA TRẬN NHẦM LẦN K-NEAREST NEIGHBOR



05. THỰC HIỆN THUẬT TOÁN

RANDOM FOREST



05. THỰC HIỆN THUẬT TOÁN

THỰC HIỆN THUẬT TOÁN RANDOM FOREST

```
[ ] # Thực hiện thuật toán Random Forest
rfc = RandomForestClassifier()
start_rf = time.time()
rf_pred = rfc.fit(X_train, y_train).predict(X_test)
end_rf = time.time()
times_rf = timedelta(seconds=round(end_rf-start_rf,4)).total_seconds()
print("time", times_rf)

rf_score = metrics.accuracy_score(y_test, rf_pred)
accuracy_rf = rf_score
print("Accuracy", accuracy_rf)
print("Report", metrics.classification_report(y_test, rf_pred))
```

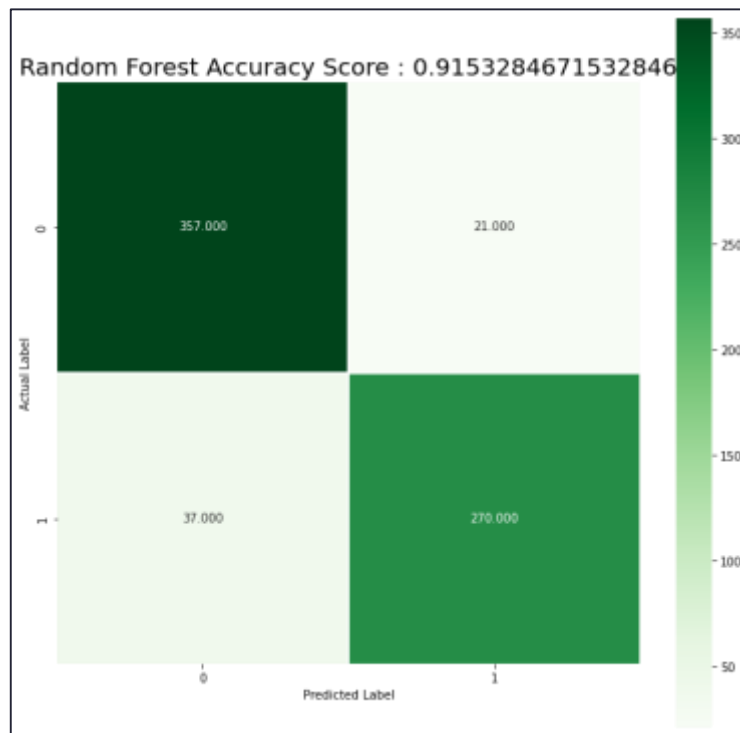
time 0.3804

Accuracy 0.9153284671532846

Report	precision	recall	f1-score	support
0	0.91	0.94	0.92	378
1	0.93	0.88	0.90	307
accuracy			0.92	685
macro avg	0.92	0.91	0.91	685
weighted avg	0.92	0.92	0.92	685

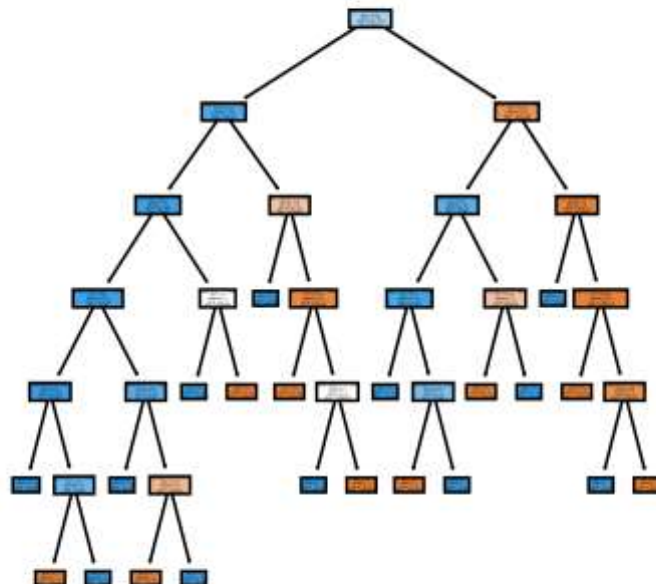
05. THỰC HIỆN THUẬT TOÁN

MA TRẬN NHẦM LẤN RANDOM FOREST



05. THỰC HIỆN THUẬT TOÁN

DECISION TREES ID3 VÀ CART



05. THỰC HIỆN THUẬT TOÁN

THỰC HIỆN THUẬT TOÁN DECISION TREES ID3

```
[ ] # Thực hiện thuật toán Decision Trees (ID3)
clf = tree.DecisionTreeClassifier(criterion="entropy", random_state=0)
start_tree = time.time()
id3_pred = clf.fit(X_train, y_train).predict(X_test)
end_tree = time.time()
times_tree_id3 = timedelta(seconds=round(end_tree - start_tree,4)).total_seconds()
print("Time decision tree (ID3)",times_tree_id3)
id3_score = metrics.accuracy_score(y_test, id3_pred)
accuracy_tree_id3 = id3_score
print("Accuracy",accuracy_tree_id3)
print("Report",metrics.classification_report(y_test,id3_pred))
```

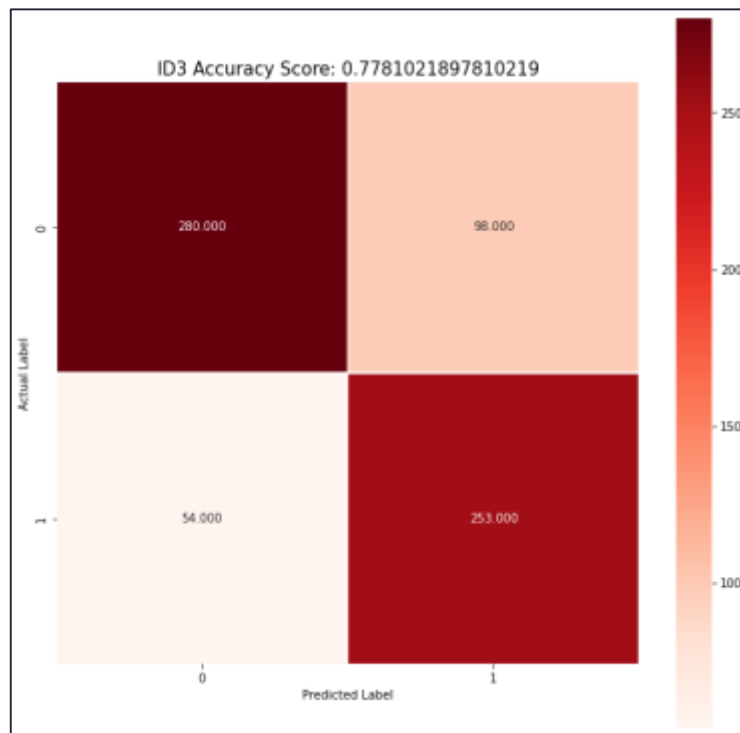
Time decision tree (ID3) 0.0295

Accuracy 0.7781021897810219

Report	precision	recall	f1-score	support
0	0.84	0.74	0.79	378
1	0.72	0.82	0.77	307
accuracy			0.78	685
macro avg	0.78	0.78	0.78	685
weighted avg	0.79	0.78	0.78	685

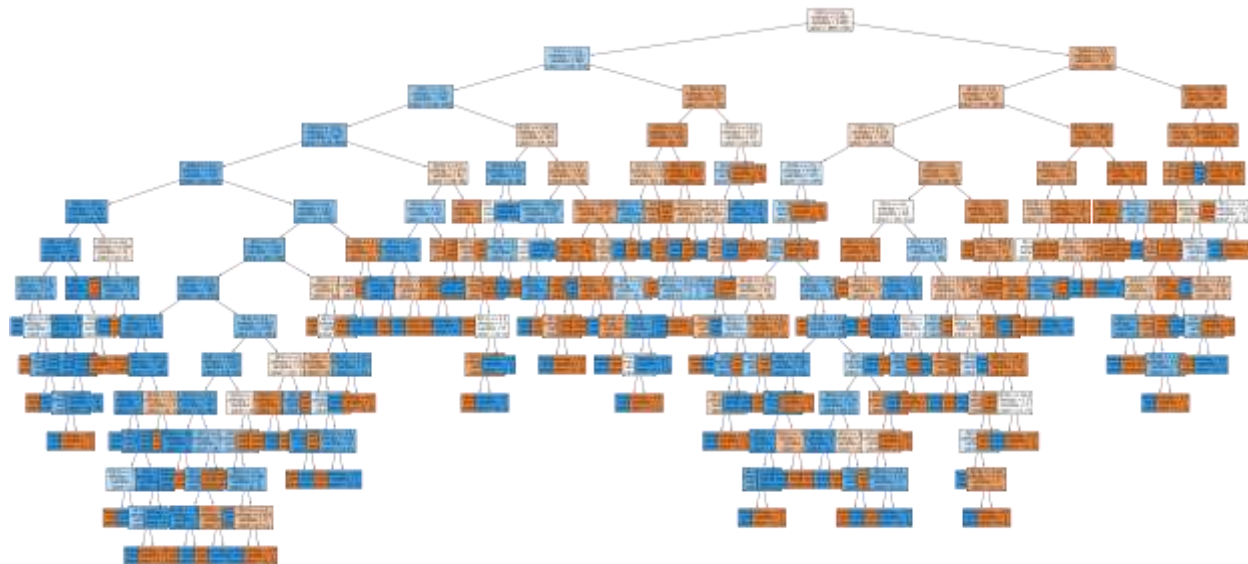
05. THỰC HIỆN THUẬT TOÁN

MA TRẬN NHẦM LÃN DECISION TREES ID3



05. THỰC HIỆN THUẬT TOÁN

VẼ CÂY DECISION TREES ID3



05. THỰC HIỆN THUẬT TOÁN

THỰC HIỆN THUẬT TOÁN DECISION TREES CART

```
[ ] # Thực hiện thuật toán Decision Trees (CART)
    clf1 = tree.DecisionTreeClassifier(criterion="gini", random_state=0)

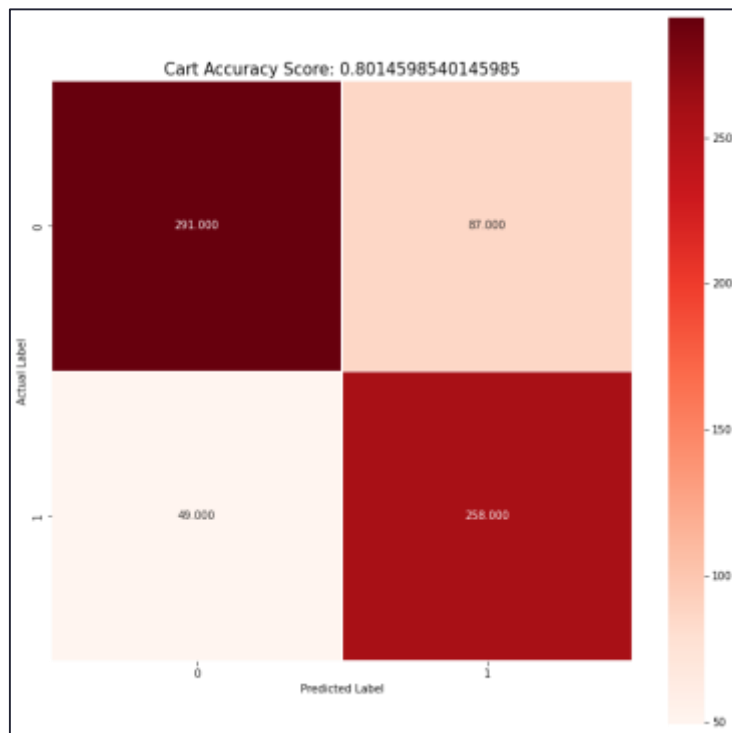
    start_cart = time.time()
    cart_pred = clf1.fit(X_train, y_train).predict(X_test)
    end_cart = time.time()
    times_tree_cart = timedelta(seconds=round(end_cart - start_cart,4)).total_seconds()
    print("Time decision tree (CART)",times_tree_cart)
    cart_score = metrics.accuracy_score(y_test, cart_pred)
    accuracy_tree_cart = cart_score
    print("Accuracy",accuracy_tree_cart)
    print("Report",metrics.classification_report(y_test,cart_pred))
```

```
Time decision tree (CART) 0.0367
Accuracy 0.8014598540145985
```

Report	precision	recall	f1-score	support
0	0.86	0.77	0.81	378
1	0.75	0.84	0.79	307
accuracy			0.80	685
macro avg	0.80	0.81	0.80	685
weighted avg	0.81	0.80	0.80	685

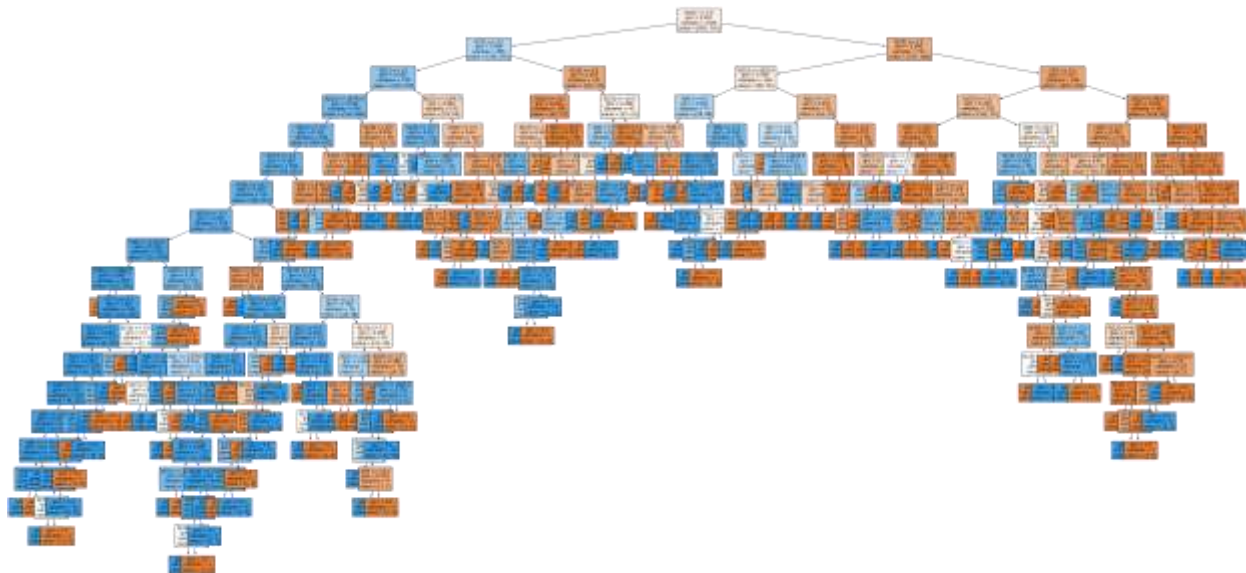
05. THỰC HIỆN THUẬT TOÁN

MA TRẬN NHẦM LẦN DECISION TREES CART



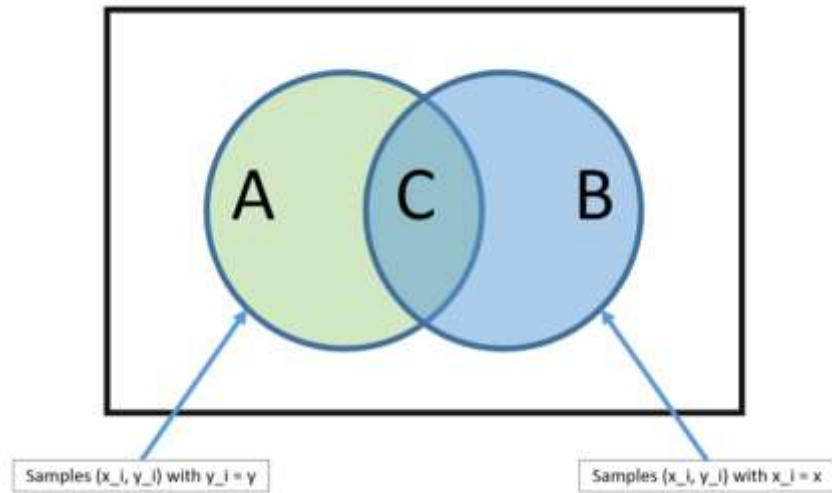
05. THỰC HIỆN THUẬT TOÁN

VẼ CÂY DECISION TREES CART



05. THỰC HIỆN THUẬT TOÁN

NAIVE BAYES



05. THỰC HIỆN THUẬT TOÁN

THỰC HIỆN THUẬT TOÁN NAIVE BAYES

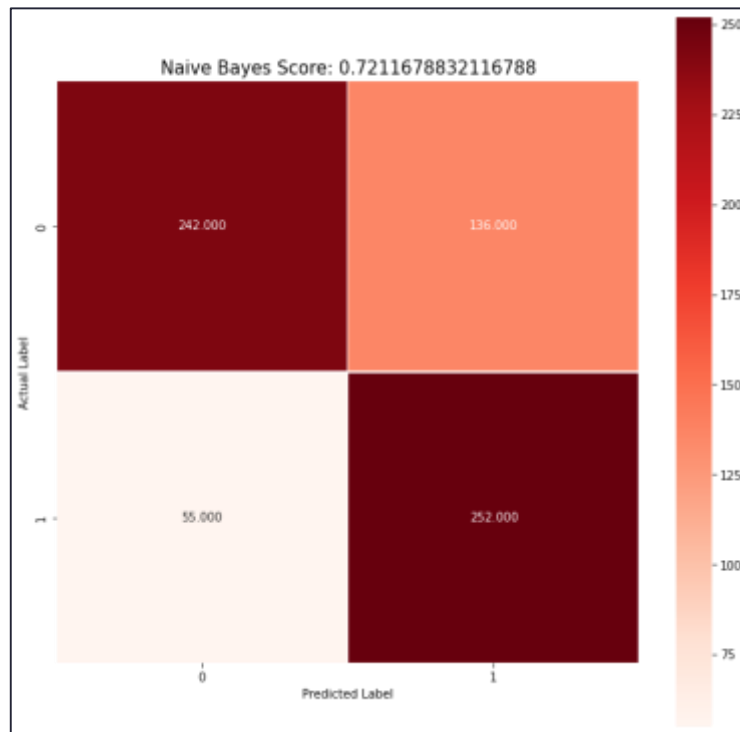
```
[ ] # Thực hiện thuật toán Naive Bayes
nv = GaussianNB()
start_nv = time.time()
nv_pred = nv.fit(X_train, y_train).predict(X_test)
end_nv = time.time()
times_nv = timedelta(seconds=round(end_nv - start_nv,4)).total_seconds()
print("Time Naive Bayes",times_nv)
nv_score = metrics.accuracy_score(y_test, nv_pred)
accuracy_nv = nv_score
print("Accuracy",accuracy_nv)
print("Report",metrics.classification_report(y_test,nv_pred))
```

```
Time Naive Bayes 0.0076
Accuracy 0.7211678832116788
```

Report:	precision	recall	f1-score	support
0	0.81	0.64	0.72	378
1	0.65	0.82	0.73	307
accuracy			0.72	685
macro avg	0.73	0.73	0.72	685
weighted avg	0.74	0.72	0.72	685

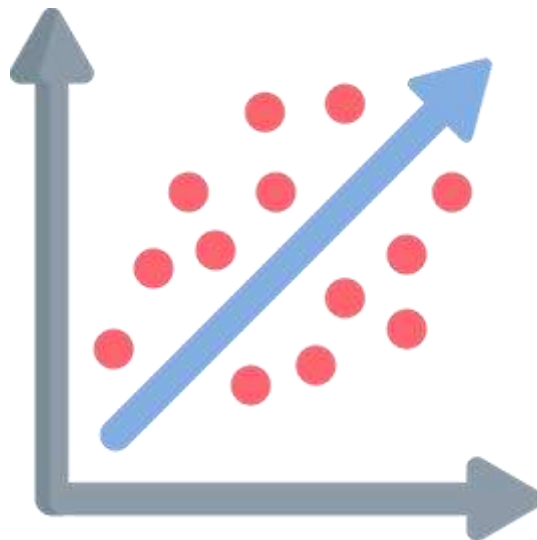
05. THỰC HIỆN THUẬT TOÁN

MA TRẬN NHẦM LẦN NAIVE BAYES



05. THỰC HIỆN THUẬT TOÁN

LOGISTIC REGRESSION



05. THỰC HIỆN THUẬT TOÁN

THỰC HIỆN THUẬT TOÁN LOGISTIC REGRESSION

```
[ ] # Thực hiện thuật toán Logistic Regression
lr_score = metrics.accuracy_score(y_test, lr_pred)
accuracies_logistic_regression=lr_score
print("Accuracy", lr_score)
print("Report", metrics.classification_report(y_test, lr_pred, labels=np.unique(lr_pred)))
```

Accuracy 0.7766423357664234

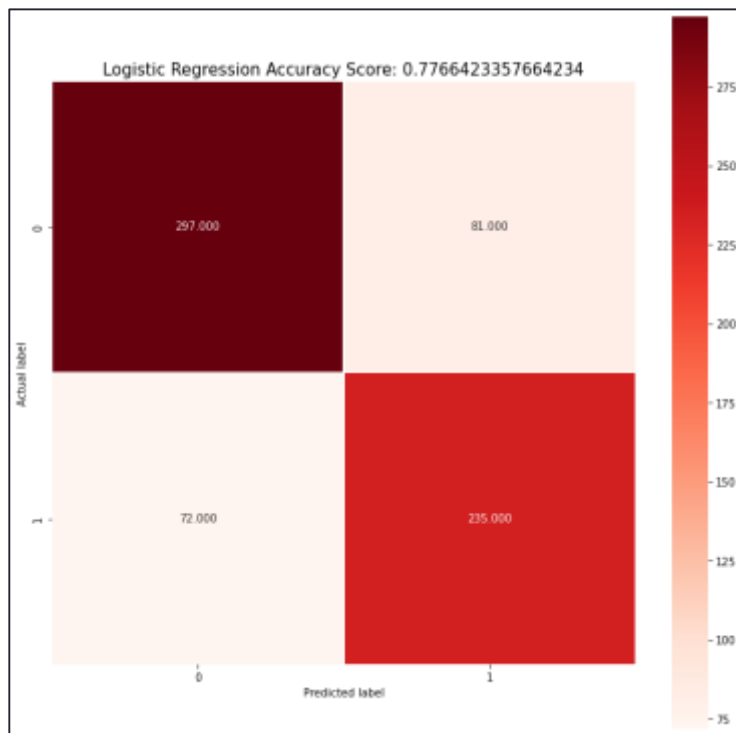
Report	precision	recall	f1-score	support
0	0.80	0.79	0.80	378
1	0.74	0.77	0.75	307
accuracy			0.78	685
macro avg	0.77	0.78	0.77	685
weighted avg	0.78	0.78	0.78	685

```
[ ] # Check thời gian thực hiện thuật toán
lr = LogisticRegression(max_iter=3300)
start_lr = time.time()
lr_pred = lr.fit(X_train, y_train).predict(X_test)
end_lr = time.time()
times_lr = timedelta(seconds=round(end_lr - start_lr,4)).total_seconds()
print("Time Logistic Regression (lr)",times_lr)
```

Time Logistic Regression (lr) 2.296

05. THỰC HIỆN THUẬT TOÁN

MA TRẬN NHẦM LẦN LOGISTIC REGRESSION



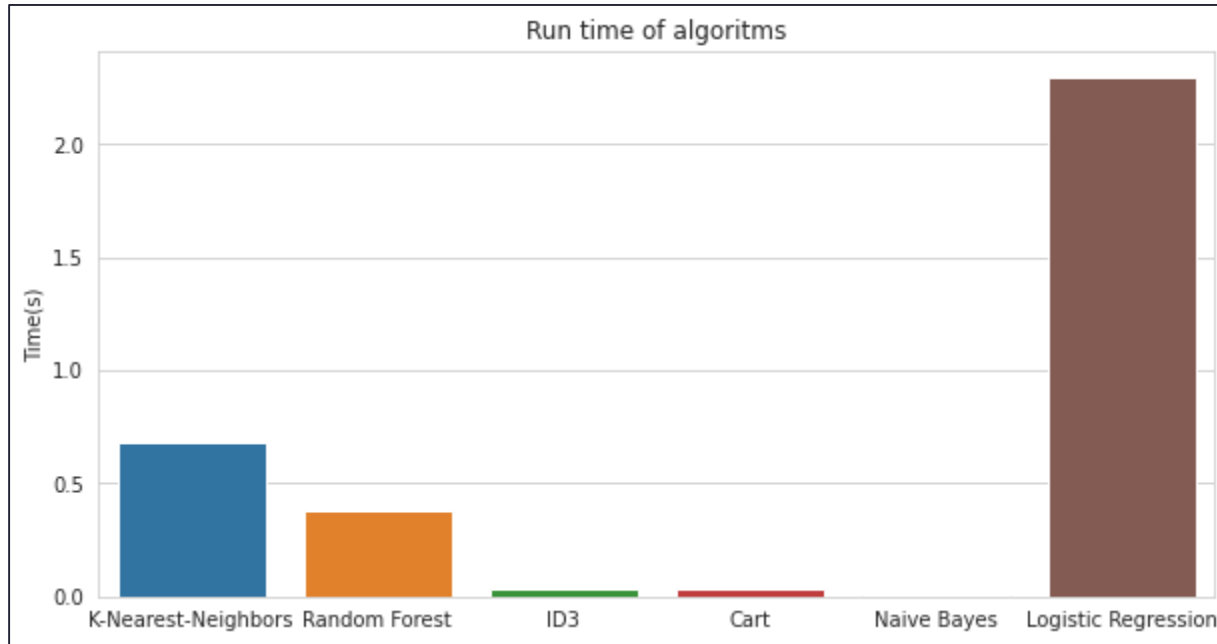
SO SÁNH THUẬT TOÁN



06

06. SO SÁNH THUẬT TOÁN

THỜI GIAN CHẠY CÁC THUẬT TOÁN



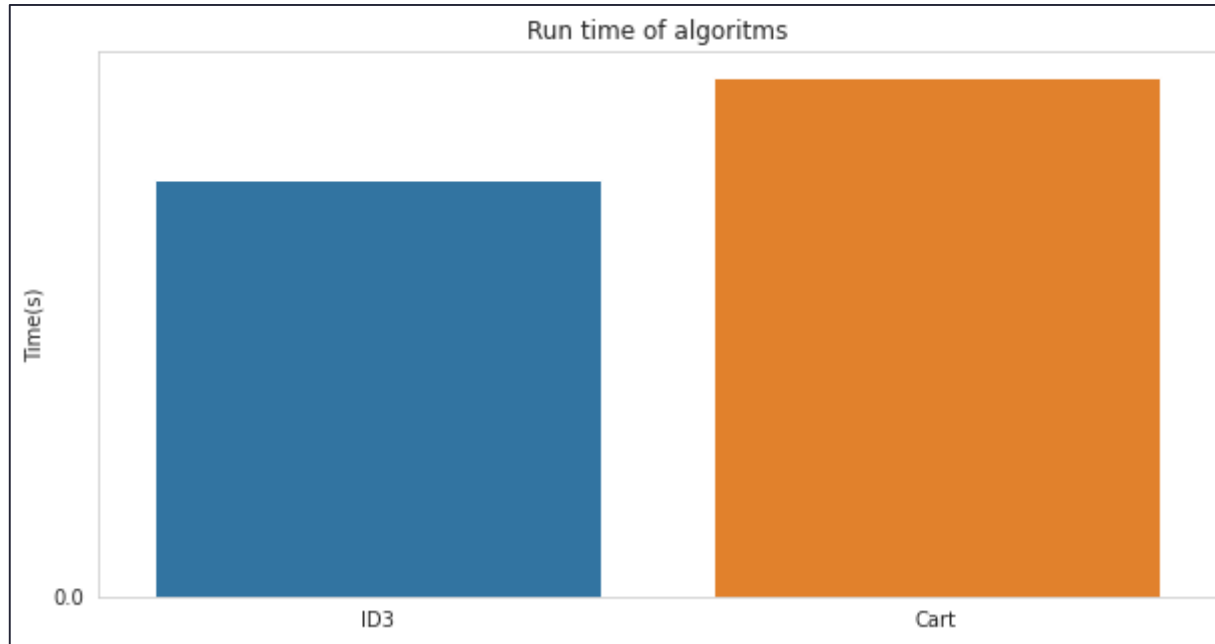
Xấp xỉ nhau

Thấp
nhất

Cao
nhất

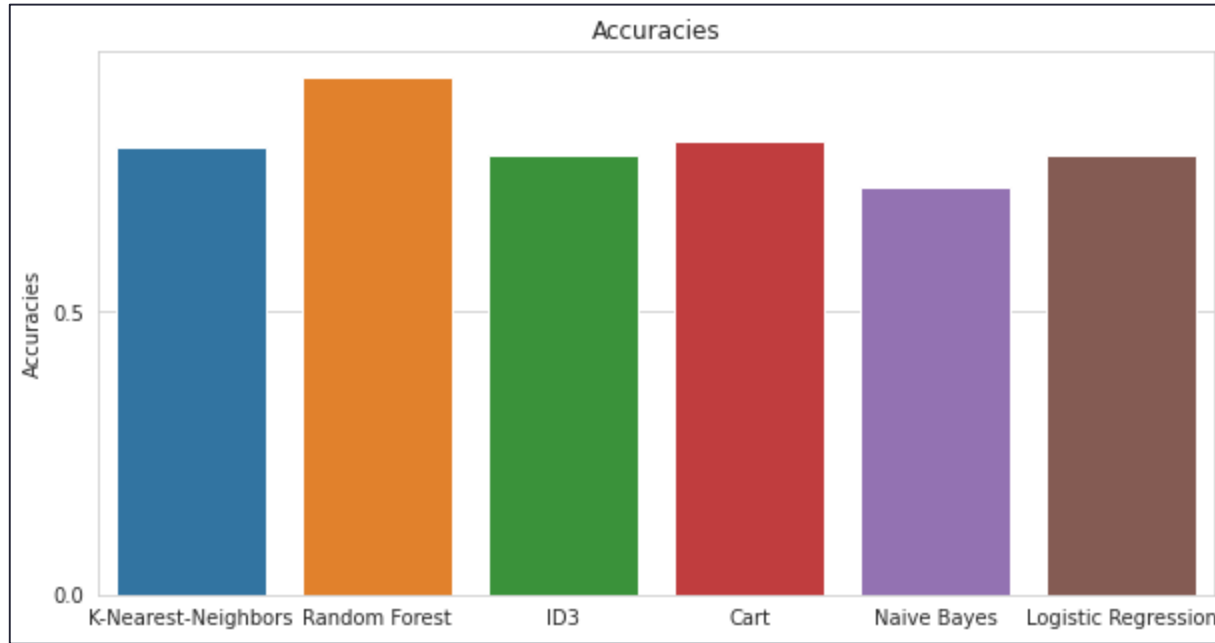
06. SO SÁNH THUẬT TOÁN

THỜI GIAN CHẠY CÁC THUẬT TOÁN



06. SO SÁNH THUẬT TOÁN

ĐỘ CHÍNH XÁC CÁC THUẬT TOÁN



**Cao
nhất**

**Thấp
nhất**

Chọn Random Forest để làm thuật toán cho phần mềm dự đoán

XÂY DỰNG PHẦN MỀM DỰ ĐOÁN

07

07. XÂY DỰNG PHẦN MỀM DỰ ĐOÁN

TÌM THUỘC TÍNH CÓ ĐỘ TIN CẬY CAO ĐỂ LÀM THUỘC TÍNH DỰ ĐOÁN

```
[149] # tìm thuộc tính có độ tin cậy cao để chọn làm thuộc tính dự đoán
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
clf = RandomForestClassifier()
clf.fit(X_train,y_train)
feature_imp = pd.Series(clf.feature_importances_,index=fi.columns).sort_values(ascending=False)
feature_imp
```

StockOptionLevel	0.061648
MonthlyIncome	0.060173
JobInvolvement	0.051810
MonthlyRate	0.046458
DailyRate	0.045678
JobSatisfaction	0.045609
TotalWorkingYears	0.045508
Age	0.043518
YearsWithCurrManager	0.043151
HourlyRate	0.043019
EnvironmentSatisfaction	0.042857
DistanceFromHome	0.040013
JobLevel	0.038941
YearsAtCompany	0.036078
WorkLifeBalance	0.034596
BusinessTravel	0.032164
YearsInCurrentRole	0.031592
RelationshipSatisfaction	0.031057
TrainingTimesLastYear	0.030547
Education	0.029696
PercentSalaryHike	0.024672
NumCompaniesWorked	0.023902
YearsSinceLastPromotion	0.021506
JobRole	0.021091
EducationField	0.019574
MaritalStatus	0.016134

Chọn 5 thuộc tính cao nhất :

- **StockOptionLevel**
- **MonthlyIncome**
- **JobInvolvement**
- **MonthlyRate**
- **DailyRate**

07. XÂY DỰNG PHẦN MỀM DỰ ĐOÁN

TÌM THUỘC TÍNH CÓ ĐỘ TIN CẬY CAO ĐỂ LÀM THUỘC TÍNH DỰ ĐOÁN

```
df_hr[["MonthlyIncome", "StockOptionLevel", "MonthlyRate", "DailyRate", "JobInvolvement"]].head(10)
```

	MonthlyIncome	StockOptionLevel	MonthlyRate	DailyRate	JobInvolvement
0	5993	0	19479	1102	3
1	5130	1	24907	279	2
2	2090	0	2396	1373	2
3	2909	0	23159	1392	3
4	3468	1	16632	591	3
5	3068	0	11864	1005	3
6	2670	3	9964	1324	4
7	2693	1	13335	1358	3
8	9526	0	6787	216	2
9	5237	2	16577	1299	3

CHỌN DÒNG 0 VÀ 3 ĐỂ THỬ APP

**CẢM ƠN THẦY VÀ CÁC
BẠN ĐÃ LẮNG NGHE**