Employee Attrition Analysis

Introduction

Employees are the most valuable assets of a company. Extraordinary and reliable employees could sensitively catch the business opportunity, put forth creative ideas about the future direction of the company and complete tasks efficiently. The company which could maintain most extraordinary employees would probably win the game. This is especially true for technology companies which need skillful and creative employees to develop amazing products. However, nowadays, high employee attrition rate perplexes managers and they wonder how they are doing to maintain employees and in which direction they can work hard to improve employees' loyalty.

Consequently, we use IBM HR Analytics Employee Attrition & Performance dataset to predict attrition of employees based on multiple features of employees. Also, we will find what factors influence employee attrition most and what managers can do to improve employees' loyalty and satisfaction. We use exploratory data analysis, data visualization, logistic regression and machine learning methods to research on this topic.

Project Objectives

The project will answer two questions: 1) What would employee attrition probability be given features of employees? 2) What factors influence employee attrition most and what can the manager do to improve employees' satisfaction and loyalty?

To resolve the first problem, the manager can put practical data into our constructed model and the model will predict attrition status of employees.

To answer the second one, we will give the most significant factors affecting attrition rate after combining findings of EDA, regression and classification. Also, some suggestions will be given towards these findings to maintain employees.

Data Source：

IBM HR Analytics Employee Attrition & Performance dataset is from Kaggle. It is a fictional dataset created by IBM data scientists targeting to uncover factors leading to employee attrition. The data have 1470 observations and 35 variables. There are some interesting and important features affecting employee attrition, such as distance from home to workplace, employees' environment satisfaction, job satisfaction and work life balance. In logistic regression, we convert categorical variables to dummy variables. And when applying machine learning methods, a test dataset is randomly chosen whose ratio of attrition is the same as that of training datasets.

Exploratory Data Analysis

Exploratory data analysis (EDA) is a systematic way to explore our data from various perspectives, helping us gain a holistic picture of the whole data frame. Meanwhile, EDA can also be an effective method to acquire useful information for certain questions. Let's first get an overview of our dataset.

1. General Information

Before we get into the deep visualizations, we need to make sure how our data looks. This will better help us to make further plans on the data exploration process throughout the whole project.

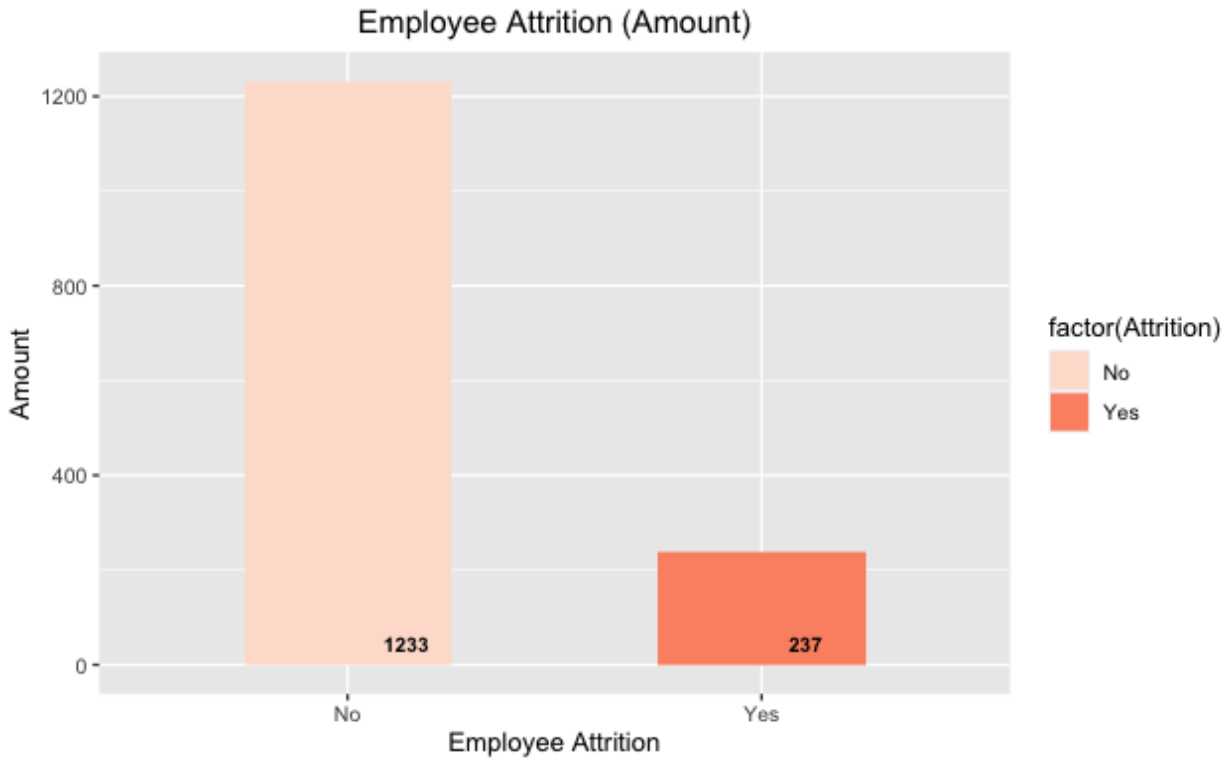
Starting with the general information, our data frame consists of 1470 observations of 35 variables. There Are only 2 kinds of data types: integers and factors. Since the data cleaning process has been conducted previously, there's no missing data in our dataset.

All the variables can be categorized into 2 types: categorical variable and numerical variable. According to this, some detailed information are listed below:

1) Categorical variables: Attrition, BusinessTravel, Department, EducationField, Gender, JobRole, MaritalStatus, Over18, OverTime.

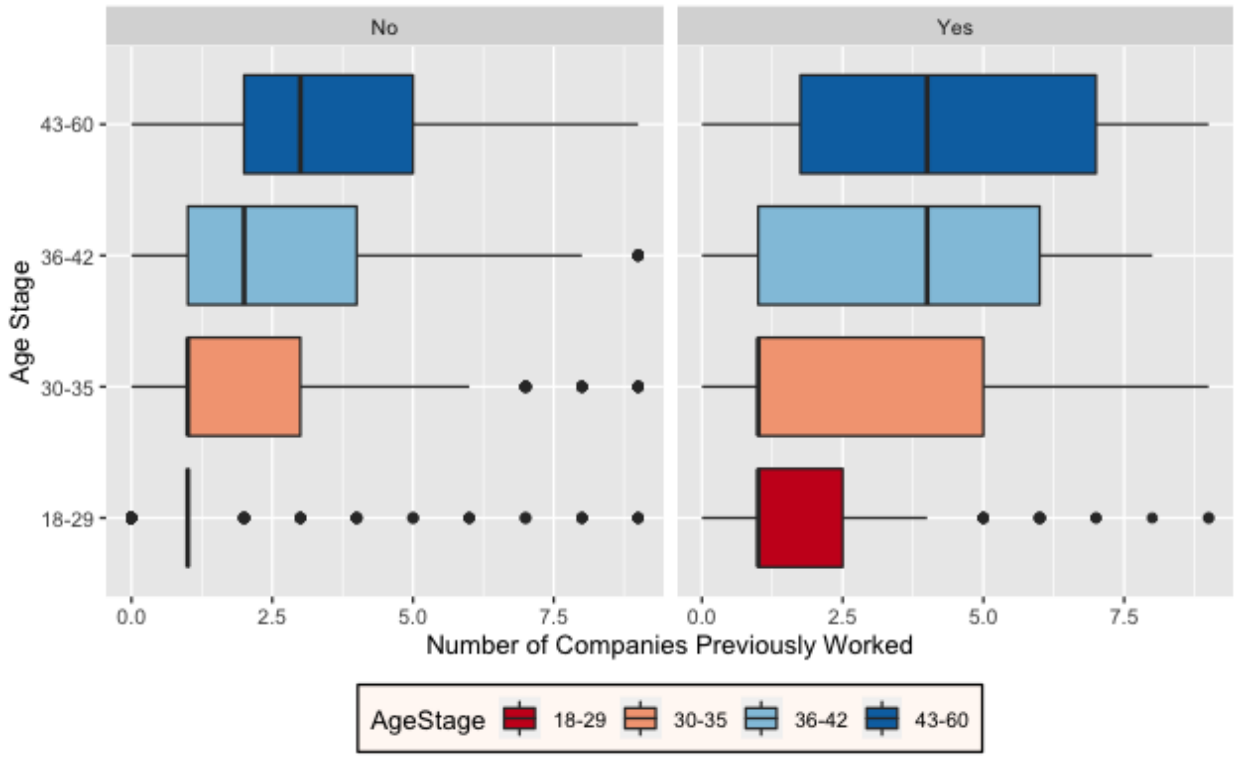
2) Numerical variables: Age, DailyRate, DistanceFromHome, Education, EmployeeCount, EmployeeNumber, EnvironmentSatisfaction, HourlyRate, JobInvolvement, JobLevel, JobSatisfaction, MonthlyIncome, MonthlyRate, NumCompaniesWorked, PercentSalaryHike, PerformanceRating, RelationshipSatisfaction, StandardHours, StockOptionLevel, TotalWorkingYears, TrainingTimesLastYear, WorkLifeBalance, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, YearsWithCurrManager.

According to the objectives of our project, Attrition will be the label in our dataset. Therefore, we want to know the distribution of our label. By building the bar plot below, we find that 1233 employees didn't quit their job in this company, while 237 employees decided to leave. Knowing that we are dealing with an imbalanced dataset will help us determine what will be the best approach to implement our predictive model in the near future.

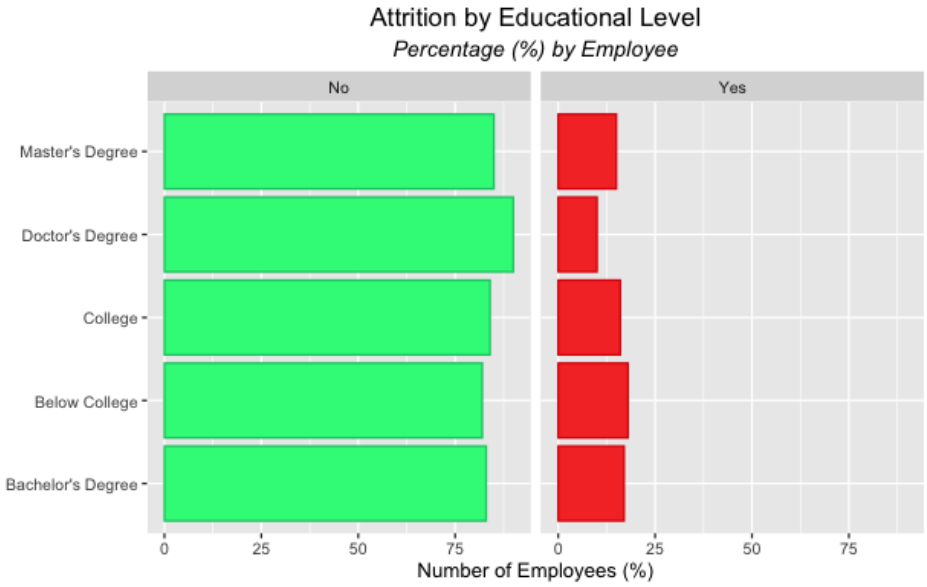


3. analysis by age and education

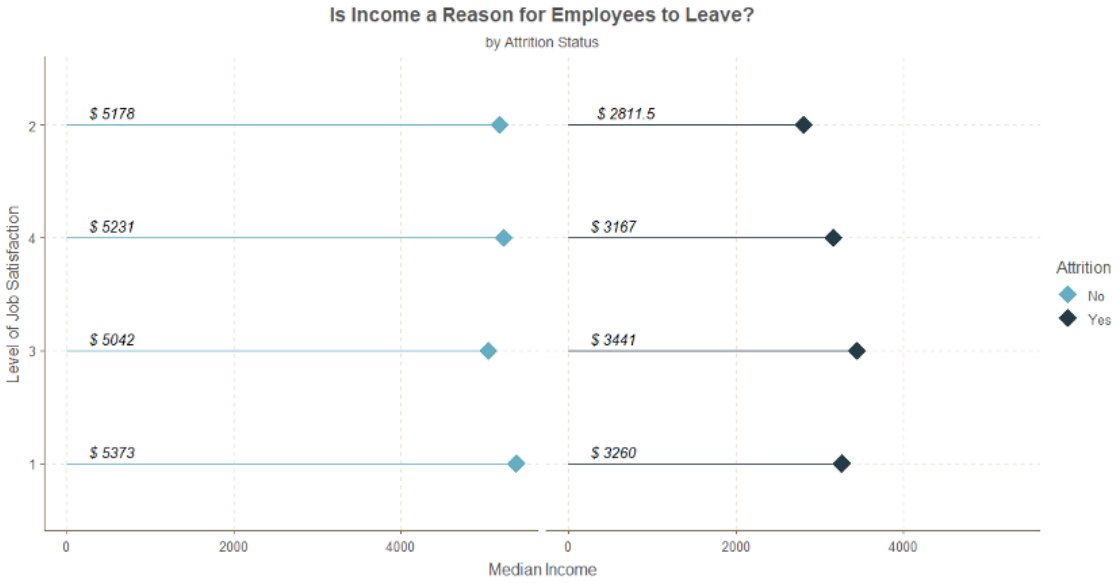
As we all know, different generations have their own peculiarities and preferences, will this have an effect on their decisions on job choosing? To explore this question, we first classify all the employees into 4 different groups according to their age stages. We want to know that if the turnover rate varies with different age stages.



From the image above, we can see that employees from 42 to 60 had a higher number of companies previously worked at. However, this may result from their longer time of working. Employees from 18 to 29 are newly at work, so that explains why the number of companies they previously worked at is relatively lower. But this number is expected to increase as the years pass by.



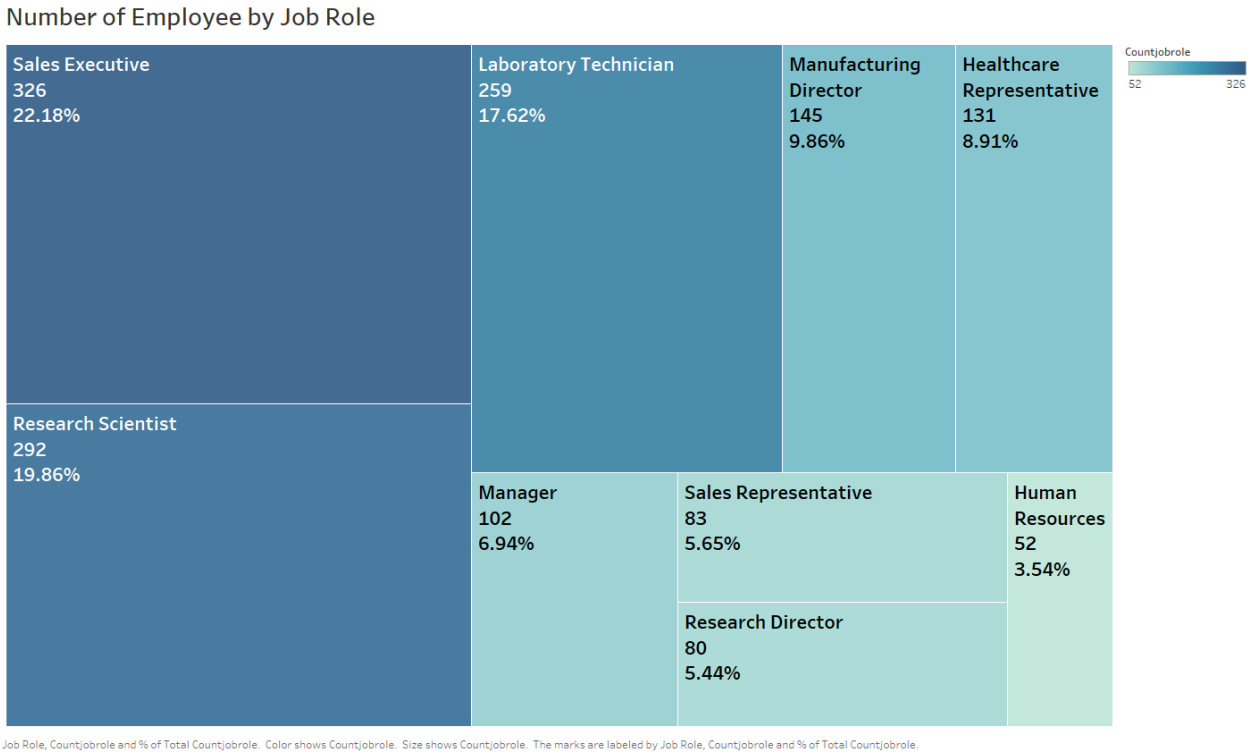
4. The impact of income towards attrition report



Next we would like to explore the relationship between the salary and attrition from the perspective of level of job satisfaction. By making two bar charts above, we can observe that the lower the job satisfaction, the wider the gap by attrition status in the levels of income. As for the reasons, I think when doing their favorite jobs, employees with more satisfaction about their jobs, might pay more attention to the sense of achievement rather than salary. The income is not the main reason which they work for.

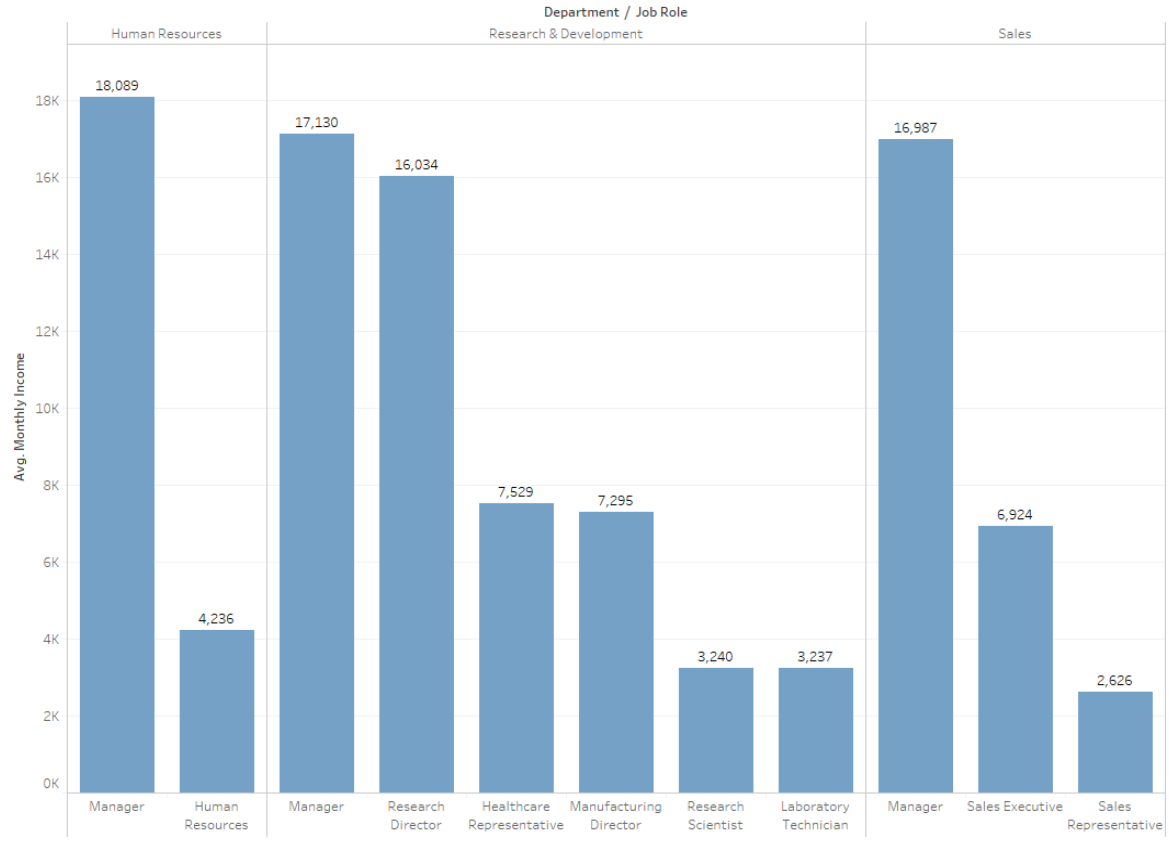
5. Working Environment

Working environment is also an important factor for employees to leave. In order to better explore the relationship between the attrition and working conditions, we first take a brief look at the employee job and department structure. According to the graph below, the company has 9 job roles: Sales executive, research scientist, laboratory technician, manager, manufacturing director, sales representative,research director, health representative, and human resource. Sales executives, research scientists, laboratory technicians consist the major group of the company, which is about 58%.



First of all, we would love to know who earns the highest salary. According to the histogram of the average monthly salary, managers of different departments earn the highest salary among all employees. In addition, research directors also earn relatively higher salaries than other employees, while sales representatives earn less among all employees.

Avg Monthly Salary of Job Roles

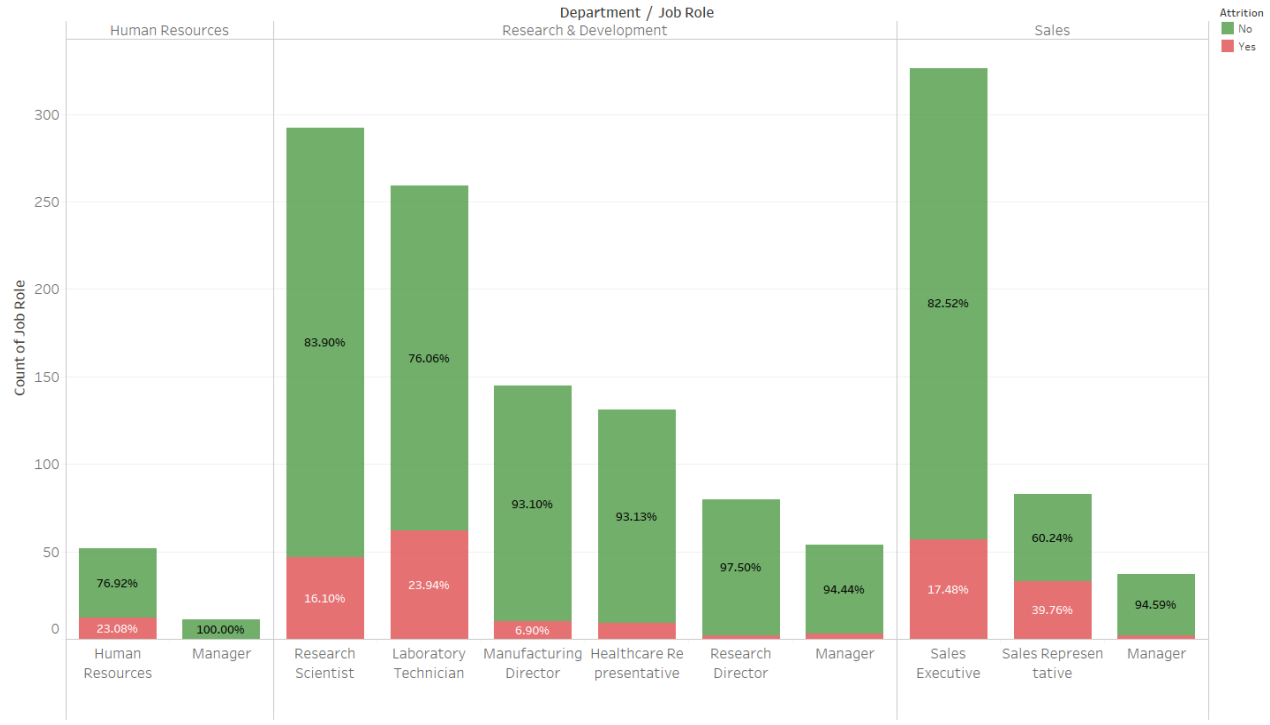


Average of Monthly Income for each Job Role broken down by Department. The marks are labeled by average of Monthly Income.

After the overview of the department, position, and salary, we are also interested in what is the attrition rate among all departments and positions. According to the histogram *Attrition percentage by job role*, human resources, research scientists,laboratory technicians, sales executives, sales representatives have the higher rate to have an attrition among all employees.

Several questions could be asked: Do employees satisfy with their working environment? Should the company offer promotions and who should be offered promotions? Should the company offer training sessions? Should the company offer housing reimbursement to decrease the working distance? Do employees balance their work and life?

Attrition Percentage by Job Role



Count of Job Role for each Job Role broken down by Department. Color shows details about Attrition. The marks are labeled by % of Total Count of Attrition.

Working Environment Satisfaction

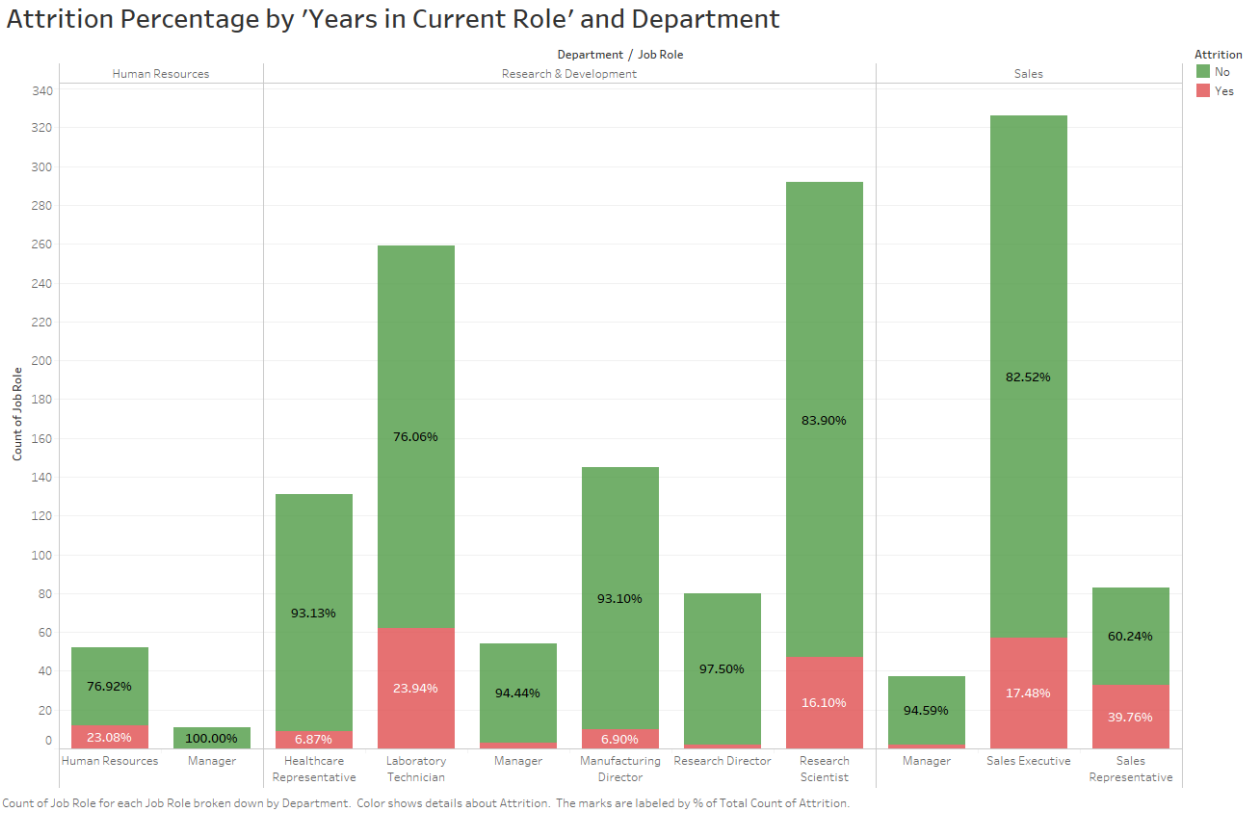
The graph provided below indicates that the average environment satisfaction score by job role. For those groups who are more easily to leave (human resources, research scientists, laboratory technicians,sales executives, sales representatives), the average disparity between attrition Yes/No groups is large. The working environment of the laboratory technicians and human resource could be improved if the company wants to make them stay. For research directors, the environment seems not to be the reason for leaving. For managers, the working environment seems to be important.

Working Environment Satisfaction By Job Role

The trend of average of Environment Satisfaction for Job Role. Color shows details about Attrition.

Promotions and Attritions

If one worked in one position for a long period of time, he or she may be tired of the job and has an intention to have a job-hopping. The graph below shows that people who are in the current position for 0-4 years among all departments are more likely to have an attrition. Therefore, human resources representatives could think of providing promotions, training, and job switching for those new employees in order to prevent unexpected job-hopping.



Training and Attritions

Training is also crucial to both company and employees. Does the reason for employees' leaving relate to training provided by the company?

According to the graph below, the training session seems to be important for research directors. There is a big disparity between average training time last year among people who had attrition and those who have not. Therefore, the HR representatives should provide more training sessions to research directors to prevent unexpected job-hopping. As for research scientists, training times seem not to be an important contributor for them to leave. As for other job positions, training times may also need to be increased.

Training Times Last Year By Job Roles

The trend of average of Training Times Last Year for Job Role. Color shows details about Attrition.

Should Company offer reimbursement?

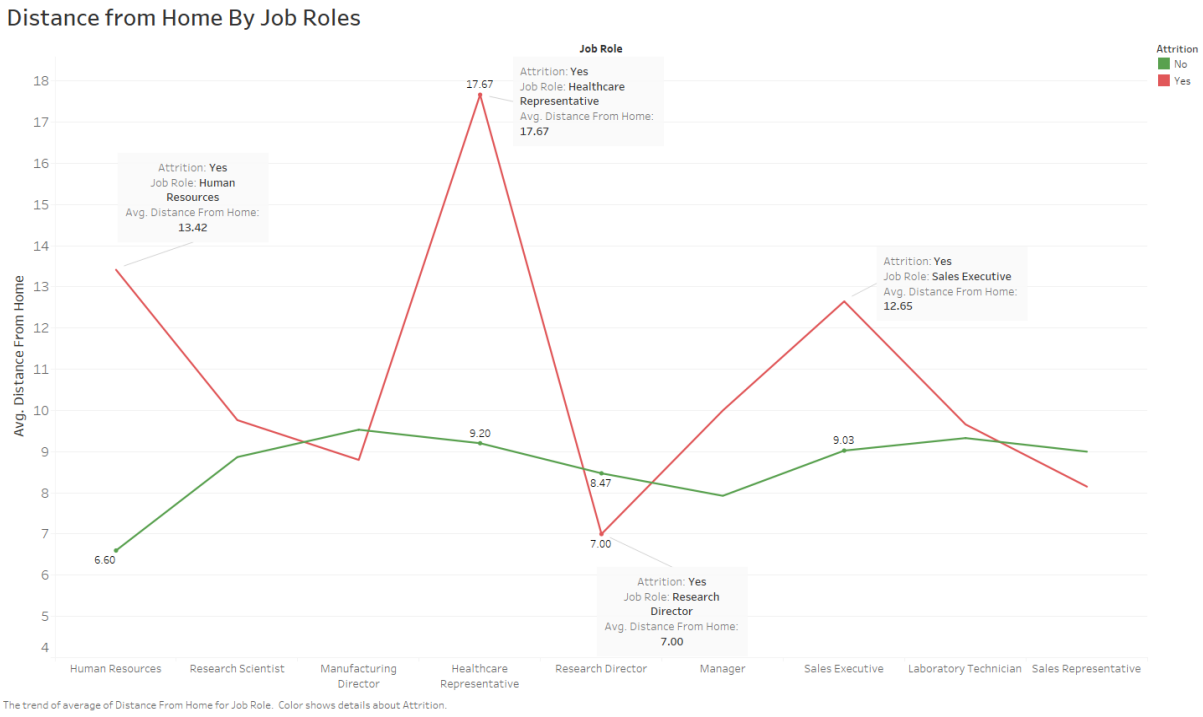
Working distances may also be an important factor for people to leave. The table below shows that the average distance from home is about the same for each position.

Job Role	
Healthcare Representative	9.786
Human Resources	8.173
Laboratory Technician	9.409
Manager	8.029
Manufacturing Director	9.483
Research Director	8.438
Research Scientist	9.014
Sales Executive	9.660
Sales Representative	8.663

Average of Distance From Home broken down by Job Role.

As for healthcare representatives, human resource, and sales executives, the company

should offer some housing reimbursement to prevent its employees from leaving. However, the working distance may not be a great contributor for the research director to leave. The company may think about housing reimbursement if they want to make their employees stay.

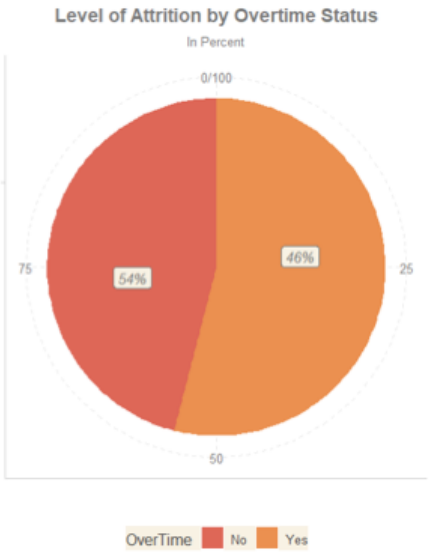


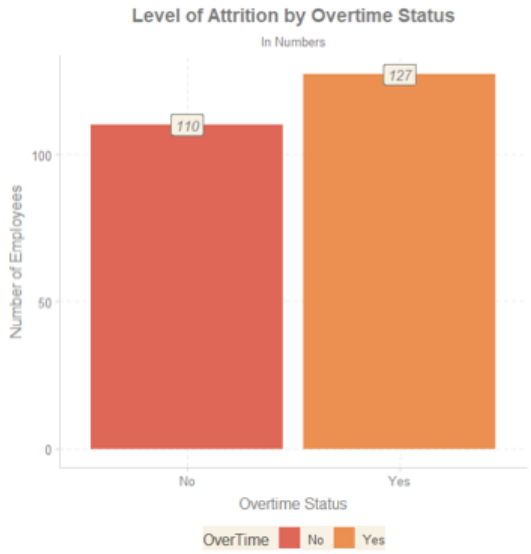
Work-life balance and attrition

Work-life balance is an essential part of one's happiness and satisfaction. The overtime hours may decrease workers' We make a fan charts and histograms for level of attrition by overtime status. Then we find that over 54% of workers who left the organization worked overtime and 46% of employees have regular work time. So it seems that there is no obvious relationship between working late and attrition.

IV. The Impact of Income towards Attrition

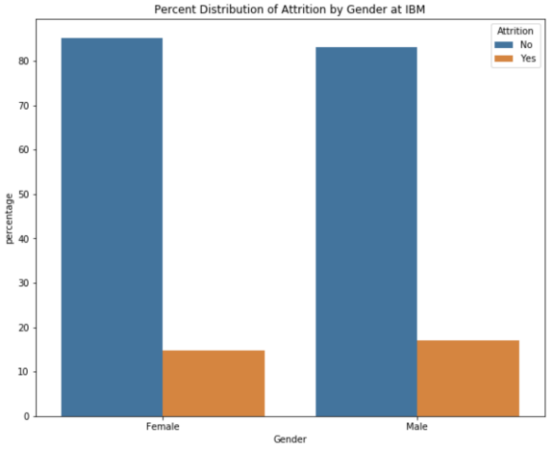
e) Level of Attrition by Overtime Status

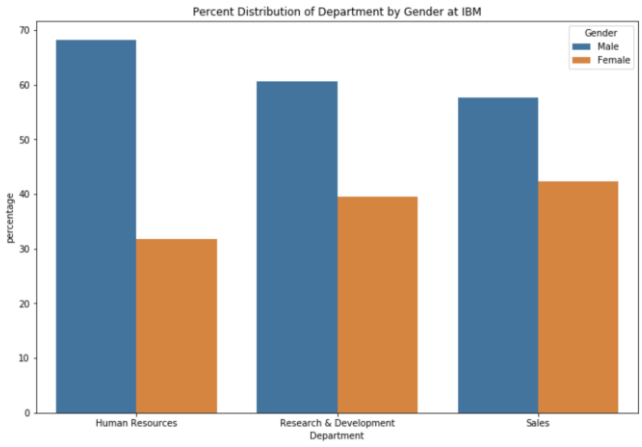




Gender Analysis

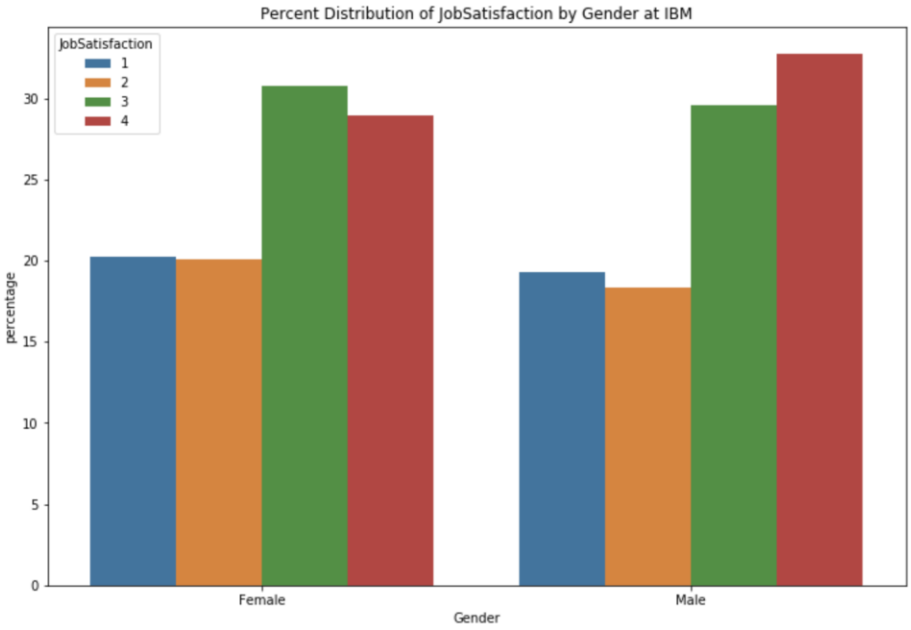
For gender analysis, we first take a look at the attrition distribution by gender, we can not find a significant difference between male and female. But we found there have significant gender imbalance phenomenon, from the graph, we can see for the HR department, which has almost 70% of employees are male, and only 30% of employees are female.Besides this, for the RDD and sales department, their gender distribution is similar, with a proportion of 60% are male employees, and 40% being female employees.



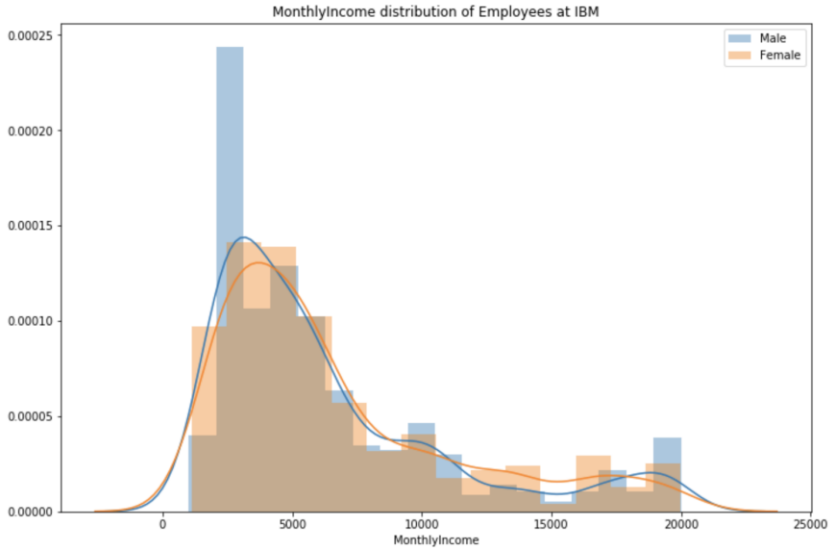


So we want to see if there are gender inequality issues in this company. From this graph, we can find the job satisfaction distribution between male and females are similar. Also, most

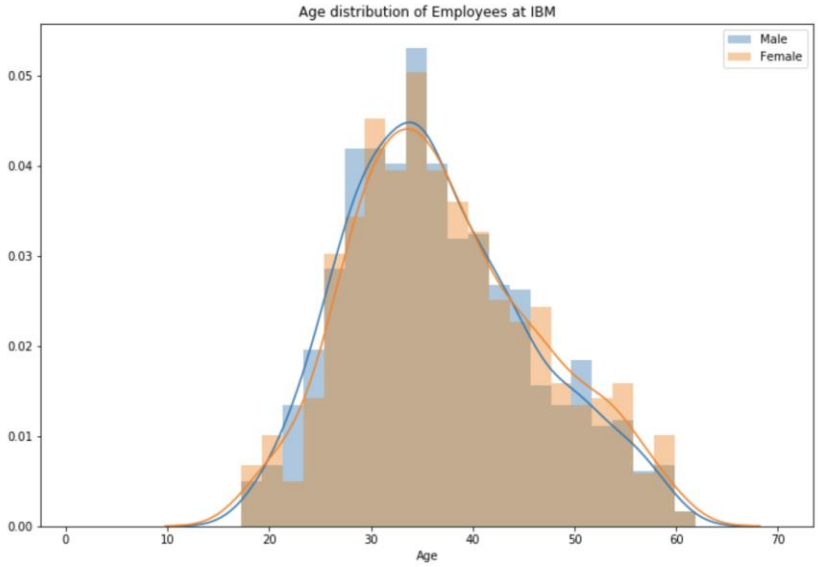
employees feel satisfied with their job. Specifically, 31% of female employees feel highly satisfied with their jobs, 29% of male employees feel satisfied with their jobs, and 40% of male employees feel unsatisfied or highly unsatisfied with their jobs. For male employees, 33% of them feel highly satisfied with their jobs, 30% of them feel satisfied with their jobs, and only 37% of them feel unsatisifed or highly unsatisfied with their jobs.



From this graph, we can see the average monthly income is similar between male and female employees, female employees have a slightly higher average monthly income than male employees, with an average income of 6687 dollars, compared to the male employees' average income of 6381 dollars. The only small difference might be the most common female income interval is 3k - 6k, and there are significantly more male in the wage interval of 2k - 3k than females.



Lastly, we want to see the age distribution between male and female employees. From the graph we can see the age distribution by gender is almost the same.



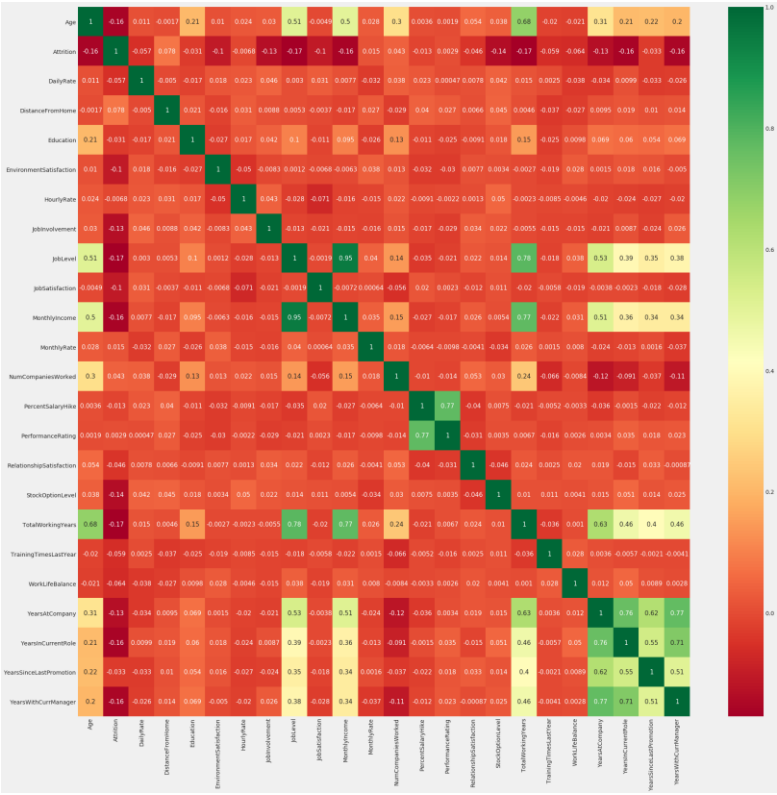
Overall, although there is a gender imbalance phenomenon. But when we look at the age, job satisfaction and income distribution by gender, it turns out that male and females have similar distributions. So the gender inequality issue may not exist in the company.

Applying Machine Learning Algorithms

- Correlation Matrix
- Data Processing
- Train different machine learning models
- Best Machine Learning model & Feature Importance

- Conclusion

Correlation Matrix



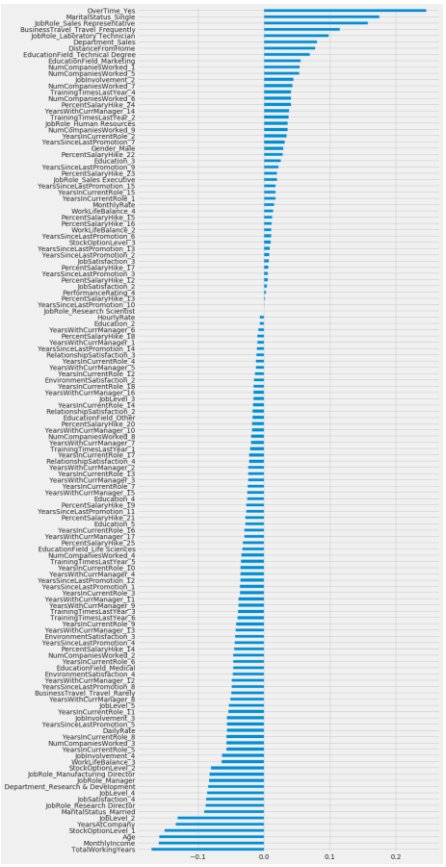
- First, we use heatmap to check the correlation matrix of different numerical variables.
- We can clearly see that the correlation between most numerical variables is not strong.

Analysis of correlation results (sample analysis):

- Monthly income is highly correlated with Job level.
- Job level is highly correlated with total working hours.
- Monthly income is highly correlated with total working hours.
- Age is also positively correlated with the Total working hours.
- Marital status and stock option level are negatively correlated

Data Processing

1.Dummy variables



- First of all, we noticed that we have many categorical variables. These variables are also important factors that affect Attrition. In order to apply these factors to machine model algorithms other than tree models, we Transform categorical data into dummies.
- And then we also check the correlation between attrition and other variables.

2. Train test split

- I personally prefer to shuffle my data before splitting.
- We split all our data into the training set and the test set at a ratio of 2:8.
- Checking that both the training and testing sets have the same label proportions.The attrition ratio of the two sets is the same

A tibble: 2 × 3

Attrition	n	pct
<fct>	<int>	<dbl>
No	987	0.84
Yes	190	0.16

A tibble: 2 × 3

Attrition	n	pct
<fct>	<int>	<dbl>
No	246	0.84
Yes	47	0.16

Applying Machine Learning

Before applying machine learning algorithms, there is an important thing to think about.

What defines success?

We have an imbalanced data, so if we predict that all our employees will stay we'll have an accuracy of 83.90%.

Train results

(*the higher the score, the better the model for all measurement*)

	Test accuracy	Train accuracy	Test Precision	Recall Score	F1 score	Confusion Matrix

Logistic Regression	85.26%	92.91%	56.00%	39.44%	46.28%	[[348 22] [43 28]]
Random Forest Classifier	83.90%	100%	50.00%	8.45%	14.46%	[[364 6] [65 6]]
Decision Trees Classifier	85.49%	94.07%	61.29%	26.76%	37.25%	[[358 12] [52 19]]
XGBoost Classifier	85.48%	88.82%	46.15%	29.51%	36.00%	[[359 21] [43 18]]

Explanation for result

(the higher the score, the better the model for all measurement)

tp--predict the positive class as a positive class (true positive)

fn--predict a positive class to a negative class (false negative)

fp--predict negative classes as positive classes (false positive)

tn--predict the negative class as a negative class (true negative)

Using the hospital to diagnose whether a patient is sick as an example, "sick" is the positive category that is concerned, and "no disease" is the negative category. This example is referred to here as example 1.

Precision:

$$P = \frac{tp}{tp + fp}$$

Recall

$$R = \frac{tp}{tp + fn}$$

F1:

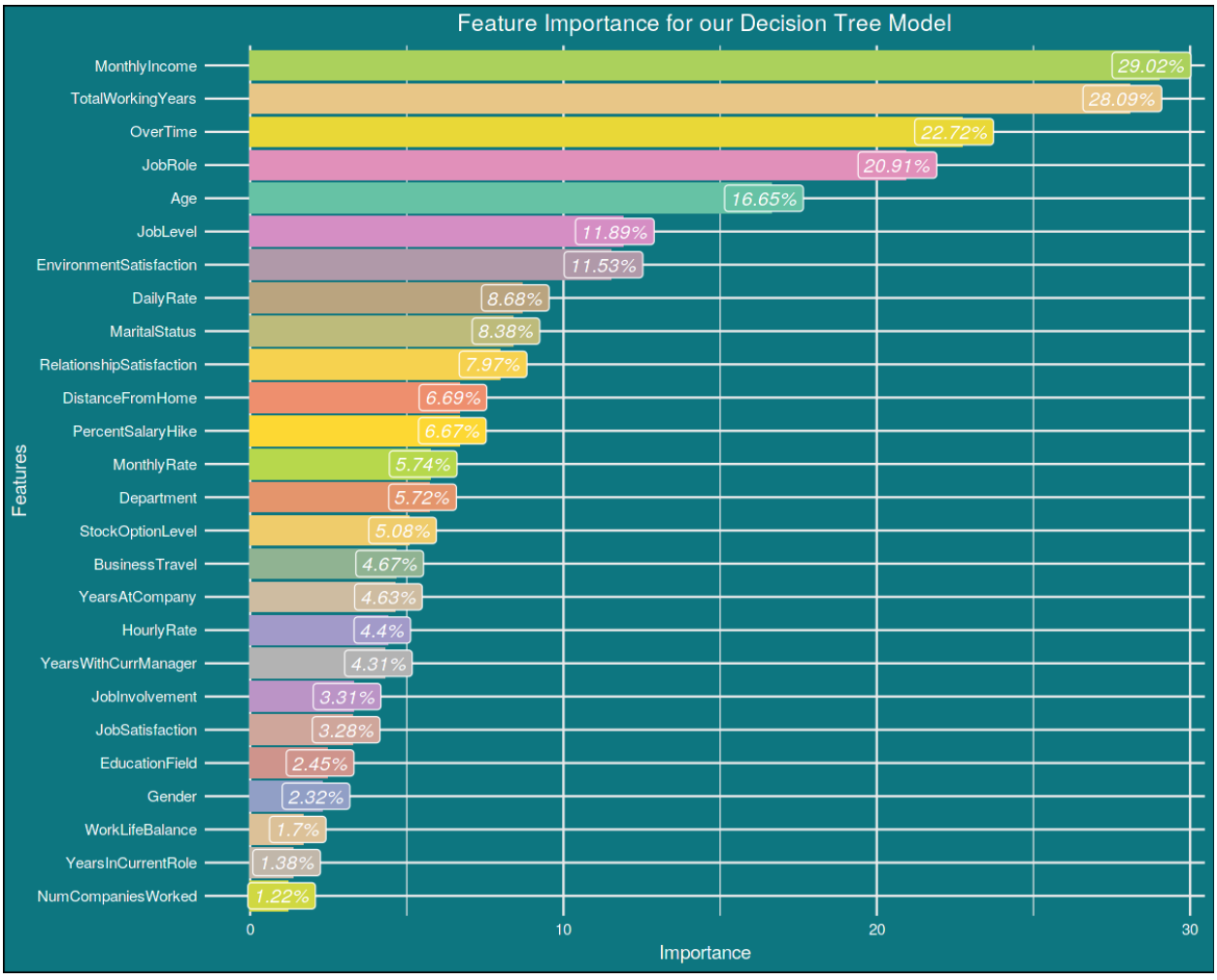
$$\frac{2}{F_1} = \frac{1}{P} + \frac{1}{R} \Rightarrow F_1 = \frac{2PR}{P + R}$$

The F1 value is the harmonic average of precision and recall.

Through training and Tuning four machine learning models, we got the following results. Among them, the highest test accuracy is the decision tree model, with an accuracy of 85.49%. (可以不说, at the same time we noticed that the training accuracy of the random forest model reached 100%, which represents overfitting)

Best Model Analysis: Decision Tree Algorithm

Combining the overall training results and considering that our original data contains more categorical variables, and the results of the tree model are more convenient for practical applications, we finally choose the decision tree model as our final model.



Top 5 Feature: Monthly Income, Total Working Years, OverTime, Job Role, ,Age

Classification Process

Conclusion

The Decision Tree Model considers these six attributes as the most important:

- **Monthly income** :As expected, Income is a huge factor as to why employees leave the organization in search for a better salary.
- **Total Working Years**: A portion might be retiring or looking for more challenges.
- **No Overtime**: This was a surprise, employees who don't have overtime are most likely to leave the organization. This could be that employees would like to have a higher amount of income or employees could feel that they are underused.
- **Job Role**: Employees who do not like their current Job Positions.
- **Job Level:** Employees who believe they deserve a higher job level are prone to leaving the organization.
- **Age**:This could also be expected, since people who are aiming to retire will leave the organization.

Appendix

Code Part 1:

Code Part 2：

Exploratory Data Analysis Part Code

I. General Information

a) Summary of our Data

```
mydata<-read.csv(file = "/Volumes/公共子集/❤️Columbia/❤️Fall 2020/5291/proj/HR-Employee-Attrition.csv",header = TRUE)
head(mydata)
str(mydata)
# 1. Dataset structure: 1470 observations of 35 variables
# 2. Data type: int, factor
summary(mydata)
# 1. Categorical variables: Attrition, BusinessTravel, Department, EducationField, Gender, JobRole, MaritalStatus, Over18, OverTime.
# 2. Numerical variables: Age, DailyRate, DistanceFromHome, etc.
```

b) Distribution of our Labels

```
library(ggplot2)
library(RColorBrewer)

# distribute "Attrition" by amount
attr_by_amount<-mydata %>% group_by(Attrition) %>% summarise(Amount=n()) %>%
  ggplot(aes(x=Attrition, y=Amount,fill=factor(Attrition)))+
  geom_bar(stat = "identity",width = 0.5)+
  geom_text(aes(x=Attrition, y=0.01, label= Amount),hjust=-0.8, vjust=-1, size=3, colour="black", fontface="bold")+
```

```
  labs(title="Employee Attrition (Amount)", x="Employee Attrition",y="Amount")+
  theme(plot.title=element_text(hjust=0.5))+
  scale_fill_brewer(palette="Reds")

print(attr_by_amount)
```

III. Analysis by Age and Education

a) Age stage

```
# create categoricals variables based on age
mydata$AgeStage<-ifelse(mydata$Age<30,"18-29",
                        ifelse(mydata$Age<36,"30-35",
                               ifelse(mydata$Age<43,"36-42","43-60")))

# density plot
age_density<-ggplot(data = mydata,aes(AgeStage))+
  geom_density(aes(fill=factor(AgeStage)),size=2)+
  labs(title = "Density")

print(age_density)

# We want to know if the turnover rate varies with different age stages
attr_dist<-mydata %>% select(AgeStage,Attrition,Age) %>%
  ggplot()+
  geom_boxplot(aes(x=Age,y=Attrition,fill=factor(AgeStage)),width=1,outlier.colour = "black",
outlier.size = 0.5, outlier.shape = 16, outlier.stroke = 2, notch=TRUE)+
  labs(title="Box plot")

print(attr_dist)

# Distribution of Number of Companies Worked by Attrition and Age
# We want to see if young people have worked in more companies than the older generation
co_num_dist<-mydata %>% select(AgeStage,Attrition,NumCompaniesWorked) %>%
  ggplot()+
  geom_boxplot(aes(x=reorder(AgeStage,NumCompaniesWorked,FUN =
median),y=NumCompaniesWorked,fill=AgeStage))+
  facet_wrap(~Attrition)+
  scale_fill_brewer(palette = "RdBu")+
  coord_flip()+
  labs(x="Age Stage", y="Number of Companies Previously Worked")+
```

```
  theme(legend.position="bottom", legend.background = element_rect(fill="#FFF9F5",size=0.5,
linetype="solid",colour ="black"))

print(co_num_dist)
```

b) Educational Level

```
library(forcats)
mydata$Education[mydata$Education==1]<-"Below College"
mydata$Education[mydata$Education==2]<-"College"
mydata$Education[mydata$Education==3]<-"Bachelor's Degree"
mydata$Education[mydata$Education==4]<-"Master's Degree"
mydata$Education[mydata$Education==5]<-"Doctor's Degree"

edu_per_dist<-mydata %>% select(Education,Attrition) %>% group_by(Education,
Attrition) %>%
summarize(n=n()) %>% mutate(pct=round(prop.table(n),2) * 100) %>%
arrange(desc(pct)) %>%
  ggplot(aes(x=fct_reorder(Education,pct), y=pct, fill=Attrition, color=Attrition))+
  geom_bar(stat = "identity")+
  facet_wrap(~Attrition)+
  coord_flip()+
  scale_fill_manual(values=c("#2EF688", "#F63A2E"))+
  scale_color_manual(values=c("#09C873","#DD1509"))+
  labs(x="", y="Number of Employees (%)", title="Attrition by Educational Level",
subtitle="Percentage (%) by Employee")+
  theme(legend.position="none", plot.title=element_text(hjust=0.5, size=14),
plot.subtitle=element_text(hjust=0.5, size=12, face="italic"))

print(edu_per_dist)
```

The impact of income towards attrition report:

```
 options(repr.plot.width=8, repr.plot.height=5)
import seaborn as sns
 ggthemr('fresh')
 df$JobSatisfaction <- as.factor(df$JobSatisfaction)
 high.inc <- df %>% select(JobSatisfaction, MonthlyIncome, Attrition) %>%
group_by(JobSatisfaction, Attrition) %>%
   summarize(med=median(MonthlyIncome)) %>%
```

```
  ggplot(aes(x=fct_reorder(JobSatisfaction, -med), y=med, color=Attrition)) +
  geom_point(size=6,shape=18) +
  geom_segment(aes(x=JobSatisfaction,
                   xend=JobSatisfaction,
                   y=0,
                   yend=med)) + facet_wrap(~Attrition) +
  labs(title="Is Income a Reason for Employees to Leave?",
       subtitle="by Attrition Status",
       y="Median Income",
       x="Level of Job Satisfaction") +
  theme(axis.text.x = element_text(angle=0, vjust=0.6),
plot.title=element_text(hjust=0.5),plot.subtitle=element_text(hjust=0.5), strip.background =
element_blank(),
        strip.text = element_blank()) +
  coord_flip()  +
  geom_text(aes(x=JobSatisfaction, y=0.01, label= paste0("$ ", round(med,2))),
            hjust=-0.5, vjust=-0.5, size=4,
            colour="black", fontface="italic",
            angle=360)
 high.inc
```

Work-life balance and attrition:

```
df %>% select(OverTime, Attrition) %>% filter(Attrition == "Yes") %>% group_by(Attrition,
OverTime) %>%
  summarize(n=n()) %>% mutate(pct=round(prop.table(n),2) * 100)
```

```
 options(repr.plot.width=10, repr.plot.height=5)
 ggthemr('pale')
```

```
overtime_percent <- df %>% select(OverTime, Attrition) %>% filter(Attrition == "Yes") %>%
group_by(Attrition, OverTime) %>%
  summarize(n=n()) %>% mutate(pct=round(prop.table(n),2) * 100) %>%
  ggplot(aes(x="", y=pct, fill=OverTime)) +
  geom_bar(width = 1, stat = "identity") + coord_polar("y", start=0) +
  geom_label(aes(label = paste0(pct, "%")), fill="#f8f2e4", colour = "gray47", position =
position_stack(vjust = 0.5),  fontface = "italic")+
```

```
  theme(legend.position="bottom", strip.background = element_blank(), strip.text.x =
element_blank(),
        plot.title=element_text(hjust=0.5, color="gray47",size = 14),
plot.subtitle=element_text(hjust=0.5,color="gray47",size = 10),
        axis.text.x=element_text(colour="gray47"), axis.text.y=element_text(colour="gray47"),
        axis.title=element_text(colour="gray47"),
        legend.background = element_rect(fill="#f8f2e4",
                                         size=0.5, linetype="solid", colour ="#f8f2e4")) +
  labs(title="Level of Attrition by Overtime Status", subtitle="In Percent", x="", y="")

 overtime_number <- df %>% select(OverTime, Attrition) %>% filter(Attrition == "Yes") %>%
group_by(Attrition, OverTime) %>%
  summarize(n=n()) %>% mutate(pct=round(prop.table(n),2) * 100) %>%
  ggplot(aes(x=OverTime, y=n, fill=OverTime)) + geom_bar(stat="identity") +
  geom_label(aes(label=paste0(n)), fill="#f8f2e4", colour = "gray47", fontface = "italic") +
  labs(title="Level of Attrition by Overtime Status", subtitle="In Numbers", x="Overtime
Status", y="Number of Employees") +
  theme(legend.position="bottom", strip.background = element_blank(), strip.text.x =
element_blank(),
        plot.title=element_text(hjust=0.5, color="gray47",size = 14),
plot.subtitle=element_text(hjust=0.5,color="gray47",size = 10),
        axis.text.x=element_text(colour="gray47"), axis.text.y=element_text(colour="gray47"),
        axis.title=element_text(colour="gray47"),
        legend.background = element_rect(fill="#f8f2e4",
                                         size=0.5, linetype="solid",
                                         colour ="#f8f2e4"))

 plot_grid(overtime_percent, overtime_number)
```

Gender Analysis

```
ibm = pd.read_csv("/Users/yingruili/Desktop/employee.csv")
ibm.head()
plt.figure(figsize=(8,8))
plt.title('Percent Distribution by Gender at IBM')
sns.countplot(x="Gender", data=ibm)
print(ibm["Gender"].value_counts())
print(ibm['Gender'].value_counts().Male/ibm['Gender'].count())
print(ibm['Gender'].value_counts().Female/ibm['Gender'].count())
attrition_counts = (ibm.groupby(['Gender'])['Attrition']
                    .value_counts(normalize=True)
```

```
                .rename('percentage')
                .mul(100)
                .reset_index())
print(attrition_counts)
plt.figure(figsize=(10,8))
plt.title('Percent Distribution of Attrition by Gender at IBM')
sns.barplot(y="percentage", x="Gender", hue="Attrition", data=attrition_counts)
department_counts = (ibm.groupby(['Department'])['Gender']
                .value_counts(normalize=True)
                .rename('percentage')
                .mul(100)
                .reset_index())
print(department_counts)
plt.figure(figsize=(12,8))
plt.title('Percent Distribution of Department by Gender at IBM')
sns.barplot(x="Department", y="percentage", hue="Gender", data=department_counts)
BusinessTravel_counts = (ibm.groupby(['BusinessTravel'])['Gender']
                .value_counts(normalize=True)
                .rename('percentage')
                .mul(100)
                .reset_index())
print(BusinessTravel_counts)
plt.figure(figsize=(12,8))
plt.title('Percent Distribution of BusinessTravel by Gender at IBM')
sns.barplot(x="BusinessTravel", y="percentage", hue="Gender", data=BusinessTravel_counts)
BusinessTravel_counts2 = (ibm.groupby(['Gender'])['BusinessTravel']
                .value_counts(normalize=True)
                .rename('percentage')
                .mul(100)
                .reset_index())
print(BusinessTravel_counts2)
plt.figure(figsize=(12,8))
plt.title('Percent Distribution of BusinessTravel by Gender at IBM')
sns.barplot(y="percentage", x="Gender", hue="BusinessTravel", data=BusinessTravel_counts2)
JobSatisfaction_counts = (ibm.groupby(['Gender'])['JobSatisfaction']
                .value_counts(normalize=True)
                .rename('percentage')
                .mul(100)
                .reset_index())
print(JobSatisfaction_counts)
```

```
plt.figure(figsize=(12,8))
plt.title('Percent Distribution of JobSatisfaction by Gender at IBM')
sns.barplot(y="percentage", x="Gender", hue="JobSatisfaction", data=JobSatisfaction_counts)
ibm.groupby("Gender")["MonthlyIncome"].describe()
plt.figure(figsize=(12,8))
plt.title('MonthlyIncome distribution of Employees at IBM')
sns.distplot(ibm.MonthlyIncome[ibm.Gender == 'Male'])
sns.distplot(ibm.MonthlyIncome[ibm.Gender == 'Female'])
plt.legend(['Male','Female'])
ibm.groupby("Gender")["Age"].describe()
plt.figure(figsize=(12,8))
plt.title('Age distribution of Employees at IBM')
sns.distplot(ibm.Age[ibm.Gender == 'Male'], bins = np.linspace(1,70,35))
sns.distplot(ibm.Age[ibm.Gender == 'Female'], bins = np.linspace(1,70,35))
plt.legend(['Male','Female'])
```

Code Part 3:

Model Part Code:

Data Processing & Applying Machine Learning(By python)

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

%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")

pd.set_option("display.float_format", "{:.2f}".format)
pd.set_option("display.max_columns", 80)
pd.set_option("display.max_rows", 80)

df = pd.read_csv("/kaggle/input/ibm-hr-analytics-attrition-dataset/WA_Fn-UseC_-HR-Employee-Attrition.csv")
```

```
df.head()
df.describe()

for column in df.columns:
    print(f"{column}: Number of unique values {df[column].nunique()}")
    print("=====================================================")

df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis="columns",
inplace=True)
```

Categorical Features

```
object_col = []
for column in df.columns:
    if df[column].dtype == object and len(df[column].unique()) <= 30:
        object_col.append(column)
        print(f"{column} : {df[column].unique()}")
        print(df[column].value_counts())
        print("====================================")
object_col.remove('Attrition')

len(object_col)

from sklearn.preprocessing import LabelEncoder

label = LabelEncoder()
df["Attrition"] = label.fit_transform(df.Attrition)
```

Numerical Features

```
disc_col = []
for column in df.columns:
    if df[column].dtypes != object and df[column].nunique() < 30:
        print(f"{column} : {df[column].unique()}")
        disc_col.append(column)
        print("====================================")
disc_col.remove('Attrition')

cont_col = []
for column in df.columns:
```

```
    if df[column].dtypes != object and df[column].nunique() > 30:
        print(f"{column} : Minimum: {df[column].min()}, Maximum: {df[column].max()}")
        cont_col.append(column)
        print("==================================")
```

3. Correlation Matrix

```
plt.figure(figsize=(30, 30))
sns.heatmap(df.corr(), annot=True, cmap="RdYlGn", annot_kws={"size":15})
col = df.corr().nlargest(20, "Attrition").Attrition.index
plt.figure(figsize=(15, 15))
sns.heatmap(df[col].corr(), annot=True, cmap="RdYlGn", annot_kws={"size":10})

df.drop('Attrition', axis=1).corrwith(df.Attrition).plot(kind='barh', figsize=(10, 7))
```

Analysis of correlation results (sample analysis):

- Monthly income is highly correlated with Job level.
- Job level is highly correlated with total working hours.
- Monthly income is highly correlated with total working hours.
- Age is also positively correlated with the Total working hours.
- Marital status and stock option level are negatively correlated

4. Data Processing

```
# Transform categorical data into dummies
dummy_col = [column for column in df.drop('Attrition', axis=1).columns if
df[column].nunique() < 20]
data = pd.get_dummies(df, columns=dummy_col, drop_first=True, dtype='uint8')
data.info()

print(data.shape)

# Remove duplicate Features
data = data.T.drop_duplicates()
data = data.T

# Remove Duplicate Rows
data.drop_duplicates(inplace=True)

print(data.shape)
```

```
data.shape

data.drop('Attrition', axis=1).corrwith(data.Attrition).sort_values().plot(kind='barh', figsize=(10,
30))

feature_correlation = data.drop('Attrition', axis=1).corrwith(data.Attrition).sort_values()
model_col = feature_correlation[np.abs(feature_correlation) > 0.02].index
len(model_col)
```

5. Applying machine learning algorithms

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X = data.drop('Attrition', axis=1)
y = data.Attrition

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42,
                                    stratify=y)

scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)
X_std = scaler.transform(X)

def feature_imp(df, model):
    fi = pd.DataFrame()
    fi["feature"] = df.columns
    fi["importance"] = model.feature_importances_
    return fi.sort_values(by="importance", ascending=False)
```

What defines success?

We have an imbalanced data, so if we predict that all our employees will stay we'll have an accuracy of 83.90%.

```
y_test.value_counts()[0] / y_test.shape[0]

stay = (y_train.value_counts()[0] / y_train.shape)[0]
leave = (y_train.value_counts()[1] / y_train.shape)[0]
```

```
print("===============TRAIN================")
print(f"Staying Rate: {stay * 100:.2f}%")
print(f"Leaving Rate: {leave * 100 :.2f}%")

stay = (y_test.value_counts()[0] / y_test.shape)[0]
leave = (y_test.value_counts()[1] / y_test.shape)[0]

print("===============TEST================")
print(f"Staying Rate: {stay * 100:.2f}%")
print(f"Leaving Rate: {leave * 100 :.2f}%")

from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score,
f1_score

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        print("Train Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
        print("_____________________________________________")
        print("Classification Report:", end='')
        print(f"\tPrecision Score: {precision_score(y_train, pred) * 100:.2f}%")
        print(f"\t\t\tRecall Score: {recall_score(y_train, pred) * 100:.2f}%")
        print(f"\t\t\tF1 score: {f1_score(y_train, pred) * 100:.2f}%")
        print("_____________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

    elif train==False:
        pred = clf.predict(X_test)
        print("Test Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
        print("_____________________________________________")
        print("Classification Report:", end='')
        print(f"\tPrecision Score: {precision_score(y_test, pred) * 100:.2f}%")
        print(f"\t\t\tRecall Score: {recall_score(y_test, pred) * 100:.2f}%")
        print(f"\t\t\tF1 score: {f1_score(y_test, pred) * 100:.2f}%")
        print("_____________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")
```

5. 1. Logistic Regression

```
from sklearn.linear_model import LogisticRegression

lr_classifier = LogisticRegression(solver='liblinear', penalty='l1')
lr_classifier.fit(X_train_std, y_train)

print_score(lr_classifier, X_train_std, y_train, X_test_std, y_test, train=True)
print_score(lr_classifier, X_train_std, y_train, X_test_std, y_test, train=False)
```

5. 2. Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

rand_forest = RandomForestClassifier(n_estimators=1200,
#                                      bootstrap=False,
#                                      class_weight={0:stay, 1:leave}
                                     )
rand_forest.fit(X_train, y_train)

print_score(rand_forest, X_train, y_train, X_test, y_test, train=True)
print_score(rand_forest, X_train, y_train, X_test, y_test, train=False)

df = feature_imp(X, rand_forest)[:40]
df.set_index('feature', inplace=True)
df.plot(kind='barh', figsize=(10, 10))
plt.title('Feature Importance according to Random Forest')
```

5. 3. Support Vector Machine

```
from sklearn.svm import SVC

svc = SVC(kernel='linear')
svc.fit(X_train_std, y_train)

print_score(svc, X_train_std, y_train, X_test_std, y_test, train=True)
print_score(svc, X_train_std, y_train, X_test_std, y_test, train=False)
```

5. 4. XGBoost Classifier

```
from xgboost import XGBClassifier

xgb_clf = XGBClassifier()
xgb_clf.fit(X_train, y_train)

print_score(xgb_clf, X_train, y_train, X_test, y_test, train=True)
print_score(xgb_clf, X_train, y_train, X_test, y_test, train=False)

df = feature_imp(X, xgb_clf)[:35]
df.set_index('feature', inplace=True)
df.plot(kind='barh', figsize=(10, 8))
plt.title('Feature Importance according to XGBoost')
```

6. Balance the dataset

```
X = data.drop('Attrition', axis=1)
y = data.Attrition

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Concatinating X_train and y_train
df = pd.concat([pd.DataFrame(X_train), pd.DataFrame(y_train)], axis=1)
df.head()

from sklearn.utils import resample

minority_class = df[df.Attrition == 1]
majority_class = df[df.Attrition == 0]

majority_downsample = resample(majority_class, replace=False,
                               n_samples=minority_class.shape[0],
                               random_state=42)

df_2 = pd.concat([majority_downsample, minority_class])
df_2.Attrition.value_counts()

X_train = df_2.drop('Attrition', axis=1)
y_train = df_2.Attrition
```

```
scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)
X_std = scaler.transform(X)
```

6. 1. Logistic Regression

```
lr_classifier = LogisticRegression(solver='liblinear', penalty='l1')
lr_classifier.fit(X_train_std, y_train)

print_score(lr_classifier, X_train_std, y_train, X_test_std, y_test, train=True)
print_score(lr_classifier, X_train_std, y_train, X_test_std, y_test, train=False)
```

6. 2. Random Forest Classifier

```
rand_forest = RandomForestClassifier(n_estimators=1500,
                                     bootstrap=True,
                                     oob_score=True
                                    )
rand_forest.fit(X_train, y_train)

print_score(rand_forest, X_train, y_train, X_test, y_test, train=True)
print_score(rand_forest, X_train, y_train, X_test, y_test, train=False)
```

6. 3. Support Vector Machine

```
svc = SVC(kernel='linear')
svc.fit(X_train_std, y_train)

print_score(svc, X_train_std, y_train, X_test_std, y_test, train=True)
print_score(svc, X_train_std, y_train, X_test_std, y_test, train=False)
```

6. 4. XGBoost classifier

```
xgb_clf = XGBClassifier()
xgb_clf.fit(X_train, y_train)

print_score(xgb_clf, X_train, y_train, X_test, y_test, train=True)
```

print_score(xgb_clf, X_train, y_train, X_test, y_test, train=False)

Data Processing & Decision Tree Model (By R)

```
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(skimr))
suppressPackageStartupMessages(library(GGally))
suppressPackageStartupMessages(library(plotly))
suppressPackageStartupMessages(library(viridis))
suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library(randomForest))
suppressPackageStartupMessages(library(e1071))
suppressPackageStartupMessages(library(rpart))
suppressPackageStartupMessages(library(xgboost))
suppressPackageStartupMessages(library(h2o))
suppressPackageStartupMessages(library(ggcorrplot))
suppressPackageStartupMessages(library(rpart.plot))
suppressPackageStartupMessages(library(corrgram))
suppressPackageStartupMessages(library(lightgbm))
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(ggthemes))
suppressPackageStartupMessages(library(psych))
suppressPackageStartupMessages(library(scales))
suppressPackageStartupMessages(library(treemap))
suppressPackageStartupMessages(library(treemapify))
suppressPackageStartupMessages(library(repr))
suppressPackageStartupMessages(library(cowplot))
suppressPackageStartupMessages(library(magrittr))
suppressPackageStartupMessages(library(ggpubr))
suppressPackageStartupMessages(library(RColorBrewer))
suppressPackageStartupMessages(library(plotrix))
suppressPackageStartupMessages(library(ggrepel))
suppressPackageStartupMessages(library(forcats))
suppressPackageStartupMessages(library(reshape2))
suppressPackageStartupMessages(library(caTools))
suppressPackageStartupMessages(library(tree))
suppressPackageStartupMessages(library(rattle))

options(repr.plot.width=8, repr.plot.height=6)
```

```
options(warn=-1)

df <- read.csv("../input/WA_Fn-UseC_-HR-Employee-Attrition.csv")
head(df)

# This will be used for training and testing.
original_df <- df

# Using an insightful summary with skim and kable
df %>% glimpse()
```

Splitting the Data

```
set.seed(142)
# # I personally prefer to shuffle my data before splitting.
original_df <- original_df[sample(nrow(original_df)),]

# Let's encode the ordinal variables
original_df$BusinessTravel = factor(original_df$BusinessTravel,
                  levels = c('Travel_Frequently', 'Travel_Rarely', 'Non-Travel'),
                  labels = c(1, 2, 3))



# Changing the datatype from integer to factors from the ordinal variables.
cols <- c("Education", "EnvironmentSatisfaction", "JobInvolvement", "JobLevel",
      "JobSatisfaction", "PerformanceRating", "RelationshipSatisfaction",
      "StockOptionLevel", "TrainingTimesLastYear", "WorkLifeBalance")

original_df[cols] <- lapply(original_df[cols], factor)

# Delete unecessary columns
cols <- c("Over18", "EmployeeNumber", "EmployeeCount")

original_df[cols] <- NULL


# Splitting our data
trainIndex <- createDataPartition(original_df$Attrition, p=0.8,
                        list=FALSE, times=1)
```

```
train <- original_df[trainIndex,]
test <- original_df[-trainIndex,]



# Checking that both the training and testing sets have the same label proportions.
prop_train <- train %>% select(Attrition) %>% group_by(Attrition) %>%
summarize(n=n()) %>%
mutate(pct=round(prop.table(n), 2))

prop_test <- test %>% select(Attrition) %>% group_by(Attrition) %>%
summarize(n=n()) %>%
mutate(pct=round(prop.table(n), 2))

prop_train
prop_test

options(repr.plot.width=10, repr.plot.height=8)

rpart.tree <- rpart(Attrition ~ ., data=train)
plot(rpart.tree, uniform=TRUE, branch=0.6, margin=0.05)
text(rpart.tree, all=TRUE, use.n=TRUE)
title("Training Set's Classification Tree")

# Complicated DecisionTree, Is there a way to determine variable importance?
var_imp <- data.frame(rpart.tree$variable.importance)
var_imp$features <- rownames(var_imp)
var_imp <- var_imp[, c(2, 1)]
var_imp$importance <- round(var_imp$rpart.tree.variable.importance, 2)
var_imp$rpart.tree.variable.importance <- NULL

colorCount <- length(unique(var_imp$features))
feature_importance <- var_imp %>%
ggplot(aes(x=reorder(features, importance), y=importance, fill=features)) +
geom_bar(stat='identity') + coord_flip() +
  theme_minimal() + theme(legend.position="none", strip.background = element_blank(),
strip.text.x = element_blank(),
    plot.title=element_text(hjust=0.5, color="white"),
plot.subtitle=element_text(color="white"),
```

```
plot.background=element_rect(fill="#0D7680"),
                                        axis.text.x=element_text(colour="white"),
axis.text.y=element_text(colour="white"),
                                        axis.title=element_text(colour="white"),
    legend.background = element_rect(fill="#FFF9F5",
                        size=0.5, linetype="solid",
                        colour ="black")) + scale_fill_manual(values =
colorRampPalette(brewer.pal(24, "Set2"))(colorCount)) +
geom_label(aes(label=paste0(importance, "%")), colour = "white", fontface = "italic",
hjust=0.6) +
labs(title="Feature Importance for our Decision Tree Model", x="Features",
y="Importance")

feature_importance

options(repr.plot.width=8, repr.plot.height=6)

predictions <- predict(rpart.tree, test, type="class")
conf_df <- data.frame(table(test$Attrition, predictions))

ggplot(data =  conf_df, mapping = aes(x = predictions, y = Var1)) +
  geom_tile(aes(fill = Freq), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Freq)), vjust = 1) +
  scale_fill_gradient(low = "#F3F781", high = "#58FA82") +
  theme_economist() + theme(legend.position="none", strip.background = element_blank(),
strip.text.x = element_blank(),
    plot.title=element_text(hjust=0.5, color="white"),
plot.subtitle=element_text(color="white"),
plot.background=element_rect(fill="#0D7680"),
                                        axis.text.x=element_text(colour="white"),
axis.text.y=element_text(colour="white"),
                                        axis.title=element_text(colour="white"),
    legend.background = element_rect(fill="#FFF9F5",
                        size=0.5, linetype="solid",
                        colour ="black")) +
labs(title="Confusion Matrix", y="Attrition Status", x="Predictions")

# Pruning reduces the size of decision trees by removing parts of the tree that do not
provide power to classify instances
```

```
prune.rpart.tree <- prune(rpart.tree, cp=0.02) # pruning the tree
plot(prune.rpart.tree, uniform=TRUE, branch=0.6)
text(prune.rpart.tree, all=TRUE, use.n=TRUE)

library(partykit)

rparty.tree <- as.party(rpart.tree)
rparty.tree
```