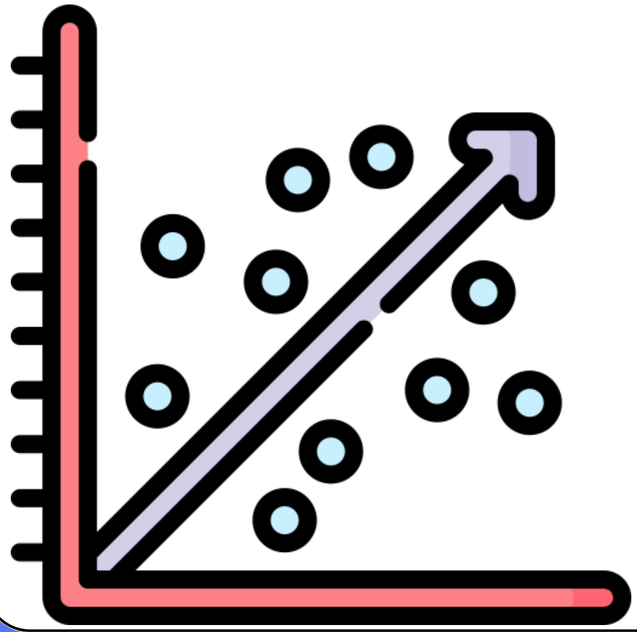
Institute of Technology of Cambodia

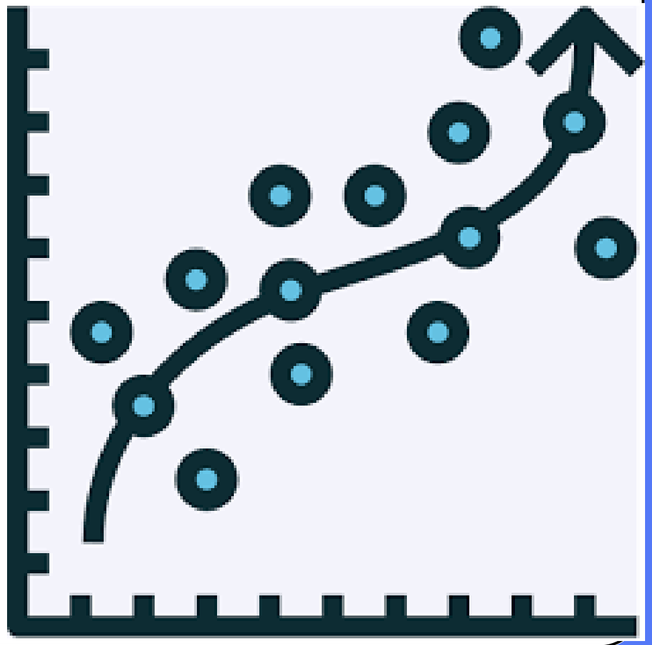
Department of Applied Mathematics and Statistics

Statistical analysis:

GROUP 10: RA005



IBM HR Analytics Employee Attrition & Performance



Lectures:

DR.PHAUK SOKHHEY
(COURSE)

MR.NHHIM MALAI
(TD)

TEAM MEMBERs

VEN VANNUTH
e20201651

SENG VATHANAK
e20200463

MEN CHANCHHORPORN
e20201146

PEAN CHHINGER
e20201339

HONG KIMLENG
e20200766

CHEA MAKARA
e20201131

CONTENTS

CONTENTS

In a work environment, Employee Attrition describes an unanticipated attrition of the workforce. The causes of this decline are all unavoidable reasons such as retirement, resignation, employee loss of work capacity or sudden death. Companies with high workforce attrition rates often face the risk of abusing internal resources.

Realizing, there are many different reasons for the high labor force attrition rate in enterprises. Eg:

- Working conditions are not guaranteed.
- Salary is too low.
- The job does not match your interests.
- There is no future for career development.
- Unable to balance work and life.
- Lack of proper recognition and appreciation for employees from managers.

Workforce attrition rate is an important indicator in human resource management, which can indicate outstanding issues that need to be addressed. A low engagement rate shows that the company is on the right track. On the contrary, a high attrition rate is something no company wants.

Therefore, this project selects this topic and dataset for the purpose of:

- Discover the factors that affect employee attrition and then take measures to reduce this rate.
- Build a machine learning model based on employee factors to predict whether that employee is likely to attrition or not?

1.1. DATASET STRUCTURE

This dataset provides a comprehensive and diverse analysis of an organization's employees, focusing on areas such as employee attrition, personal and work-related factors, and resources. main. Includes more than 35 attributes and their meanings are as follows:

STT	Attribute Name	Meaning
1	Age	Employee's age
2	Gender	Employee's Gender
3	BusinessTravel	Frequency of employees' business trips
4	DailyRate	Daily salary rate for employees
5	Department	Office of employees
6	DistanceFromHome	Distance from home in miles to work
7	Education	Level of education achieved by staff
8	EducationField	Employee's field of study
9	EmployeeCount	Total number of employees in the organization
10	EmployeeNumber	A unique identifier for each employee record
11	EnvironmentSatisfaction	Employee satisfaction with their working environment
12	HourlyRate	Hourly rate for employees
13	JobInvolvement	Level of involvement required for the employee's job
14	JobLevel	Employee's level of work
15	JobRole	The role of employees in the organization
16	JobSatisfaction	Employee satisfaction with their work
17	MaritalStatus	Employee's marital status
18	MonthlyIncome	Employee's monthly income
19	MonthlyRate	Monthly salary rate for employees
20	NumCompaniesWorked	Number of companies the employee worked for
21	Over18	Whether the employee is over 18 years old
22	OverTime	Do employees work overtime
23	PercentSalaryHike	Salary increase rate for employees
24	PerformanceRating	The performance rating of the employee
25	RelationshipSatisfaction	Employee satisfaction with their relationships
26	StandardHours	Standard working hours for employees
27	StockOptionLevel	Employee stock option level
28	TotalWorkingYears	Total number of years the employee has worked
29	TrainingTimesLastYear	Number of times employees were taken to training in the last year
30	WorkLifeBalance	Employees' perception of their work-life balance
31	YearsAtCompany	Number of years employees have been with the company
32	YearsInCurrentRole	Number of years the employee has been in their current role
33	YearsSinceLastPromotion	Number of years since employee's last promotion
34	YearsWithCurrManager	Number of years an employee has been with their current manager
35	Attrition	Does the employee leave the organization

2.DATA DISCOVERY

Read data

```
df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv"
                 , sep = ',', header = 0)

df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome
0	41	Yes	Travel_Rarely	1102	Sales	1
1	49	No	Travel_Frequently	279	Research & Development	8
2	37	Yes	Travel_Rarely	1373	Research & Development	2
3	33	No	Travel_Frequently	1392	Research & Development	3
4	27	No	Travel_Rarely	591	Research & Development	2

duplicate lines

```
have_duplicate_rows = df.duplicated().any()
have_duplicate_rows
```

```
False
```

Percentage of missing values and descriptive statistics

```
describe = df.describe()
missing_rates = df[describe.columns].isna().mean()
missing_rates.name = 'missing_rate'
describe = pd.concat([describe, missing_rates.to_frame().T])
describe
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000
missing_rate	0.000000	0.000000	0.000000	0.000000	0.0	0.000000

9 rows × 26 columns

2.1.DATA TYPE OF EACH COLUMN

```
col_dtype = df.dtypes
col_dtype
```

```
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
```

LIST COLUMNS OF TYPE NUMERICAL AND CATEGORY.

```
cat_coulmns  = df.select_dtypes(['object']).columns
num_coulmns  = df.select_dtypes(['number']).columns
print(cat_coulmns)
print(num_coulmns)
```

```
Index(['Attrition', 'BusinessTravel', 'Department', 'EducationField', 'Gender',
       'JobRole', 'MaritalStatus', 'Over18', 'OverTime'],
      dtype='object')
Index(['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'],
      dtype='object')
```

CONSIDER THE VALUE SET OF CATEGORICAL ATTRIBUTES

```
num_cols = list(set(df._get_numeric_data()))
cat_cols = list(set(df.columns) - set(df._get_numeric_data()))
```

```
for col in cat_cols:
    print('Unique values of ', col, set(df[col]))
```

```
Unique values of  Over18 {'Y'}
Unique values of  Gender {'Female', 'Male'}
Unique values of  BusinessTravel {'Travel_Frequently', 'Travel_Rarely', 'Non-Travel'}
Unique values of  MaritalStatus {'Divorced', 'Married', 'Single'}
Unique values of  Department {'Research & Development', 'Sales', 'Human Resources'}
Unique values of  OverTime {'Yes', 'No'}
Unique values of  EducationField {'Technical Degree', 'Human Resources', 'Marketing', 'Other', 'Life Sciences', 'Medical'}
Unique values of  JobRole {'Manufacturing Director', 'Research Scientist', 'Sales Executive', 'Manager', 'Healthcare Represe
ntative', 'Laboratory Technician', 'Research Director', 'Human Resources', 'Sales Representative'}
Unique values of  Attrition {'Yes', 'No'}
```

CONSIDER THE VALUE DISTRIBUTION OF NUMERIC DATA COLUMNS

```
num_cols_info_df = df[num_cols].describe()
num_cols_info_df
```

	MonthlyRate	MonthlyIncome	EnvironmentSatisfaction	StandardHours	YearsSinceLastPromotion	TrainingTimesLastYear	WorkLifeBalance
count	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000
mean	14313.103401	6502.931293	2.721769	80.0	2.187755	2.799320	2.761224
std	7117.786044	4707.956783	1.093082	0.0	3.222430	1.289271	0.706476
min	2094.000000	1009.000000	1.000000	80.0	0.000000	0.000000	1.000000
25%	8047.000000	2911.000000	2.000000	80.0	0.000000	2.000000	2.000000
50%	14235.500000	4919.000000	3.000000	80.0	1.000000	3.000000	3.000000
75%	20461.500000	8379.000000	4.000000	80.0	3.000000	3.000000	3.000000
max	26999.000000	19999.000000	4.000000	80.0	15.000000	6.000000	4.000000

8 rows × 26 columns

CONSIDER THE VALUE DISTRIBUTION OF NON-NUMERIC DATA COLUMNS

Calculate the number of different values using the nunique() method.

```
1 n_values_df = pd.DataFrame({'n_values': df[cat_cols].nunique()})
2 n_values_df
```

	n_values
Over18	1
Gender	2
BusinessTravel	3
MaritalStatus	3
Department	3
OverTime	2
EducationField	6
JobRole	9
Attrition	2

Calculate the ratio of each value using the `value_counts()` method.

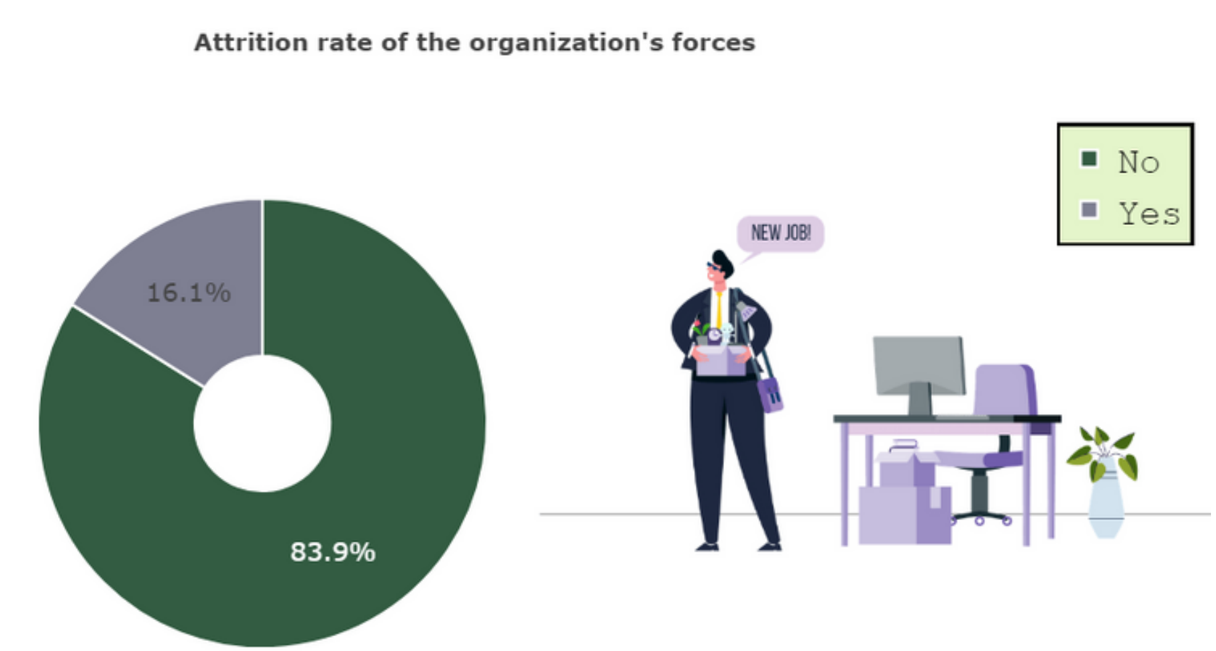
```
1 value_ratios_dict = {}
2 for col in cat_cols:
3     value_ratios_dict[col] = dict(df[col].value_counts(normalize=True) * 100)
4 value_ratios_df = pd.DataFrame({'value_ratios': value_ratios_dict})
5 value_ratios_df = value_ratios_df.transpose()[cat_cols]
6 value_ratios_df
7
```

	Over18	Gender	BusinessTravel	MaritalStatus	Department	OverTime
value_ratios	{'Y': 100.0}	{'Male': 60.0, 'Female': 40.0}	{'Travel_Rarely': 70.95238095238095, 'Travel_F...	{'Married': 45.78231292517007, 'Single': 31.97...	{'Research & Development': 65.37414965986395, ...	{'No': 71.70068027210884, 'Yes': 28.2993197278...

CONTENTS

3. DISCOVERING RELATIONSHIPS IN DATA

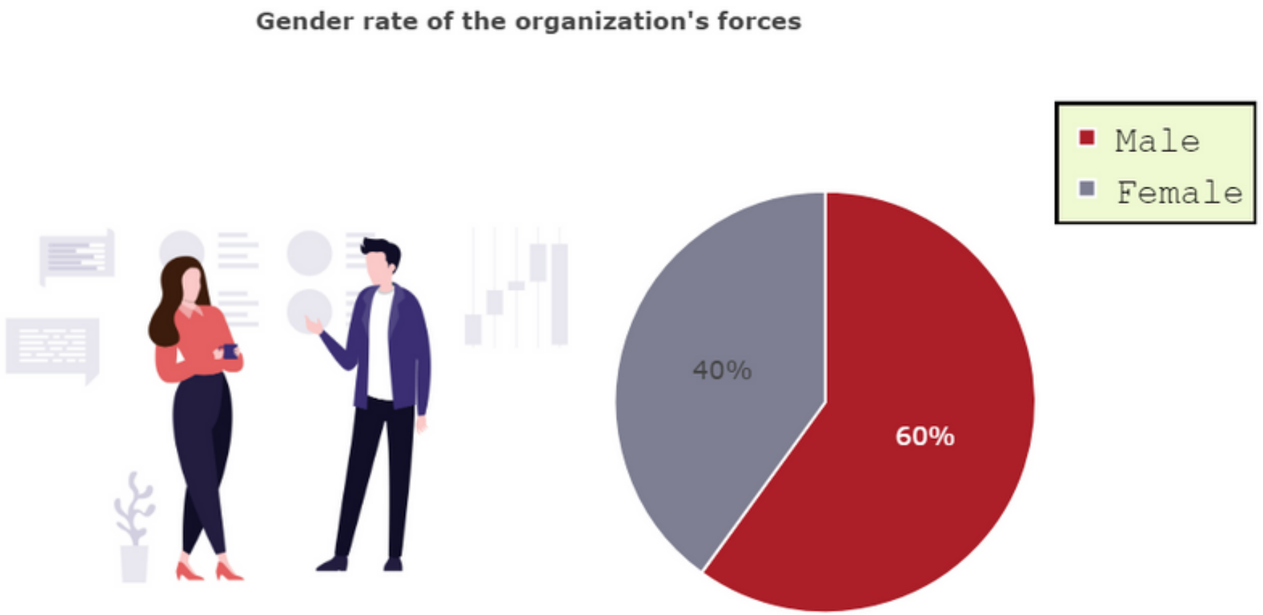
Consider the rate of employees leaving the organization or not.



- The employee turnover rate of this organization is 16.1%. And according to experts in the field of human resources, the human resource consumption rate of each enterprise from 4% to 6% is a stable level.

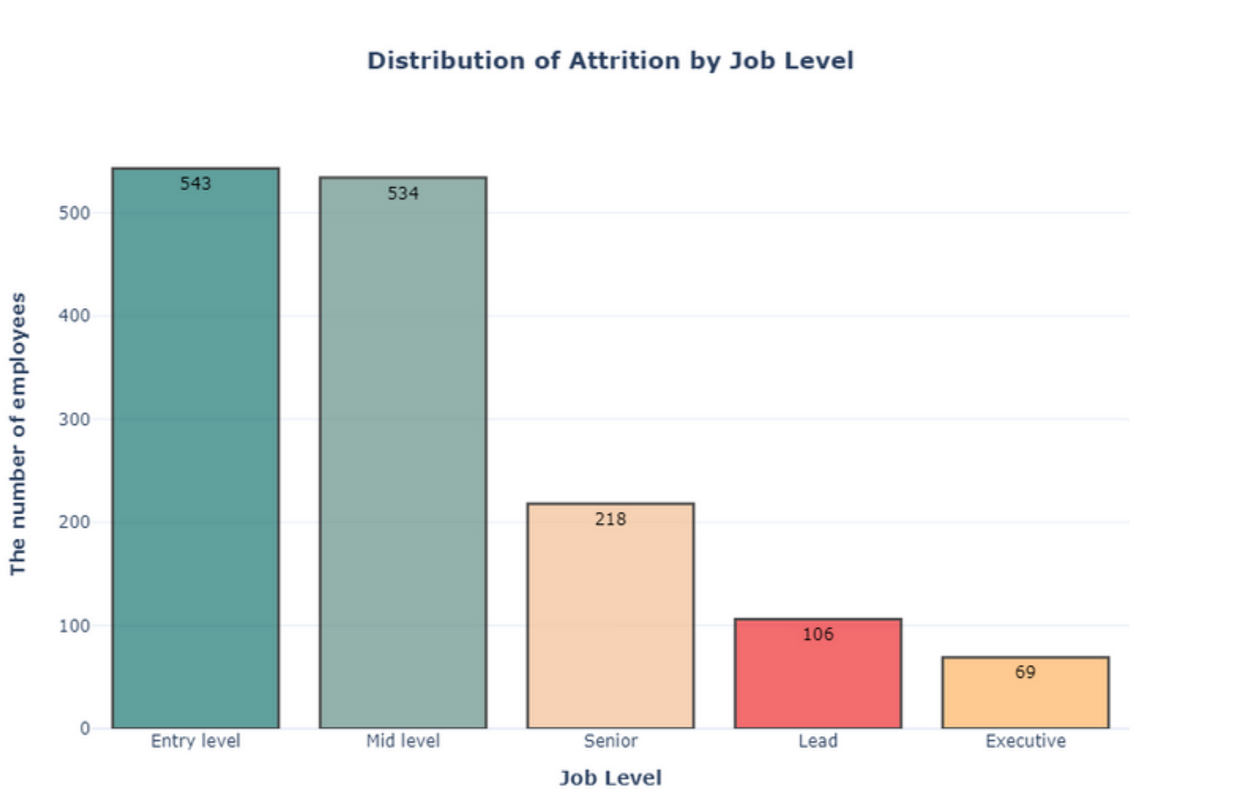
=> The turnover rate of this organization is at a dangerous level. Therefore, the organization should take measures to reduce this ratio.

Ratio of men and women in the organization



The number of male employees in the company accounts for a higher proportion than female employees (more than 20%).

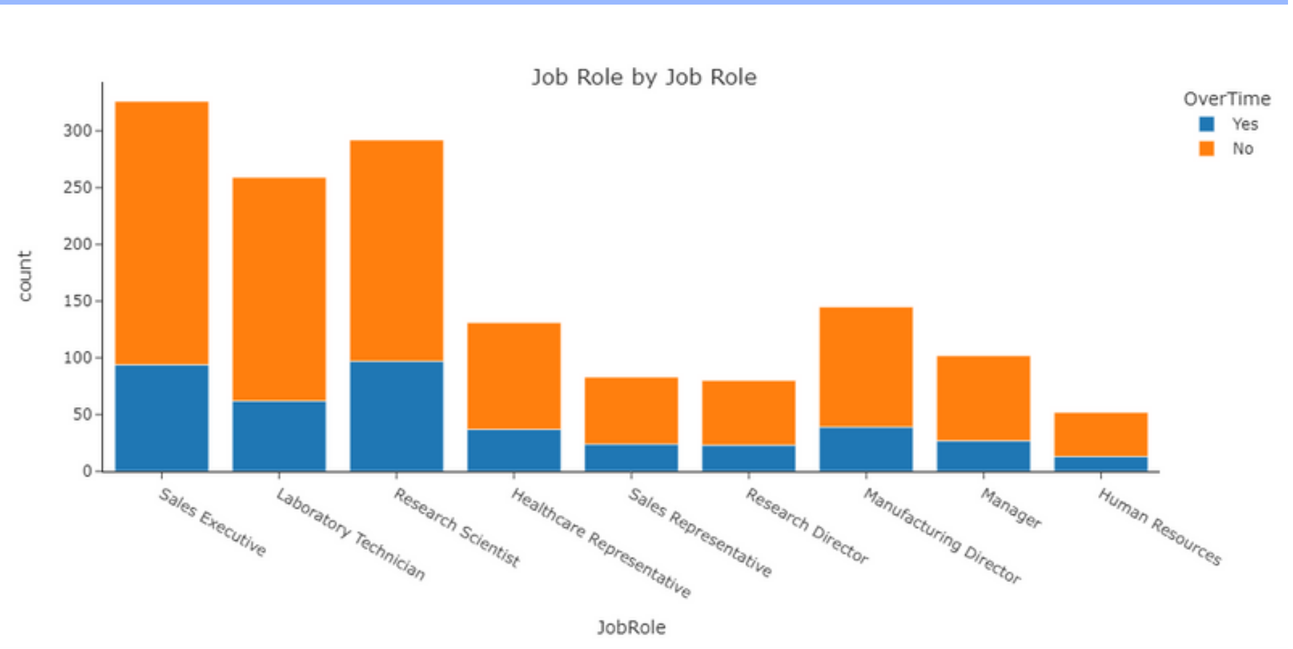
Percentage of levels in work



Based on the bar chart above we can see:
- Employees at Level 1 (Entry level) have a very high rate of leaving the company (60%). They are usually very young people.
- Employees at Level 2 (Middle Level) have a relatively high turnover rate (21%)
- Employees who achieve Level 4 (Lead) and 5 (Executive) have a very low turnover rate.

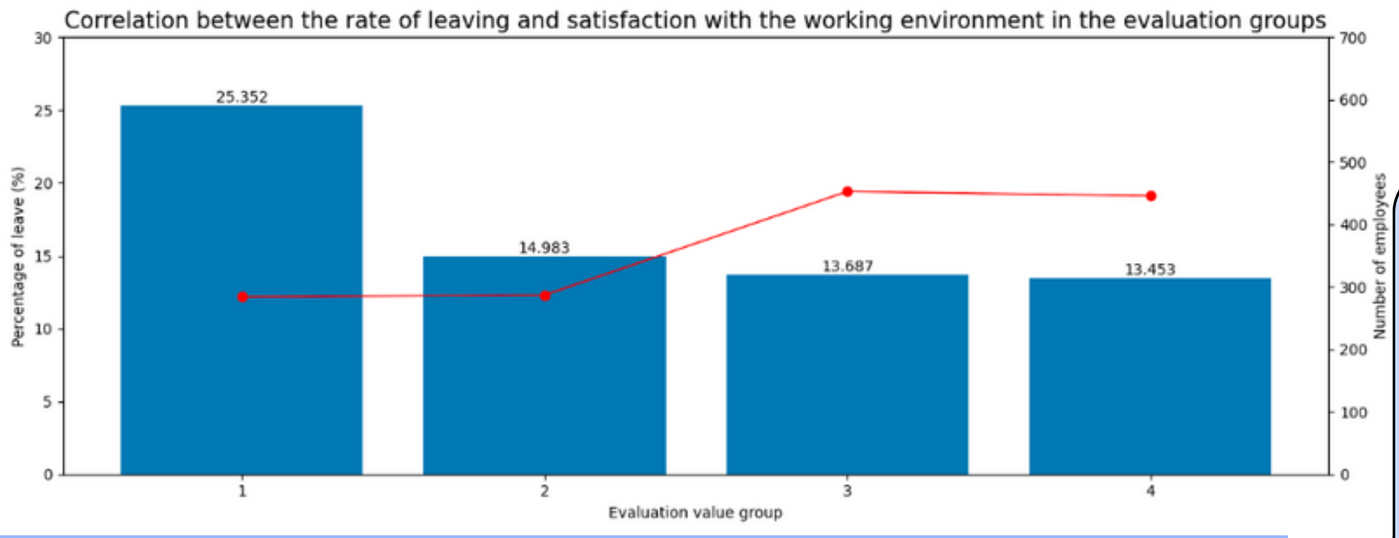
=> Young employees who have just joined the company have a very high ability to "jump".

Distribution of working overtime in job roles



- The 3 positions with the most personnel are: Sale Executive, Research Scientist and Laboratory Technician.
- The rate of working OverTime is nearly 2 times that of the group that does not work overtime. The trend of working OT is increasingly popular.

Relationship between leaving rate and job satisfaction



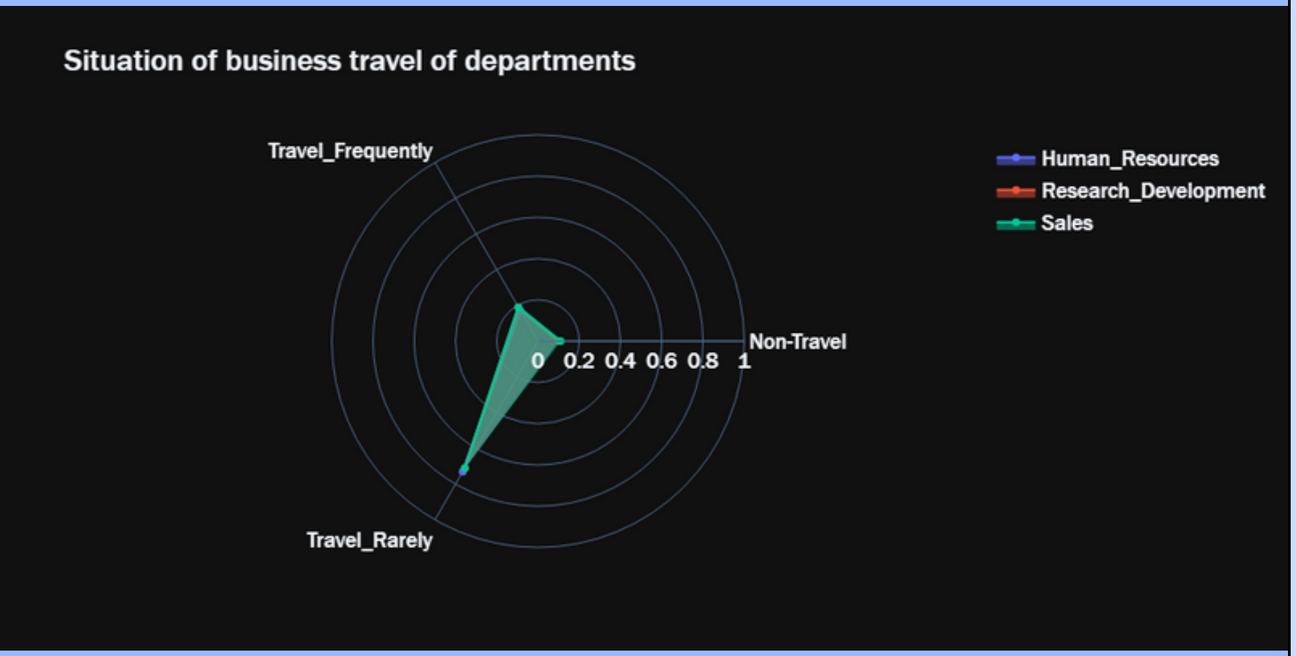
- The number of employees who are satisfied with the working environment is also very high, as is the job satisfaction.
- The average rate of leaving is higher in the low-rated groups than in the high-value groups. This is also very consistent with reality.
- From here, it shows that besides salary, the working environment factor is also an important factor that determines whether employees really stick around for a long time or not.

How does the average salary change according to the number of years working at the company?

- During the period of working with the company from 0-35 years, the average salary of employees increases gradually over time. This proves that when working for a long time, that employee will have more work experience and be a loyal employee to the company. Should have a decent salary is worth it.

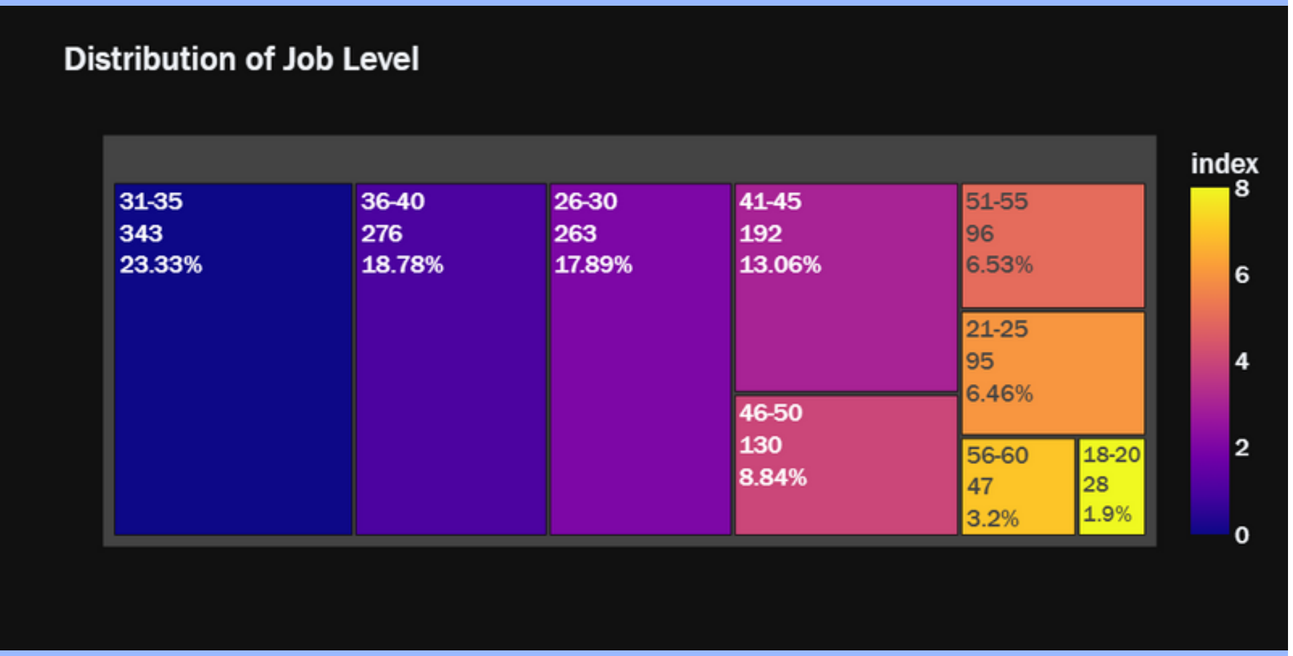
- However, the salary for working period from 35-40 years has decreased but not too big. The reason for this decline may be that when they have been with the company for 35-40 years, these employees are nearing retirement age, so their productivity decreases or they have ceded senior positions. for young leaders,...

Degree of business travel in each department



Based on the above Radar chart, we can see:
- The triangles are almost stacked, showing that the level of business travel in the departments is quite similar.
- The rate of 'Travel_Rarely' (rarely traveling for business) is the highest and 'Non-Travel' (never traveling for business) is the lowest in all departments. This ratio is quite common in companies, because it is common to go to work or to belong to high-ranking positions in the work, and the higher the rank, the smaller the number of employees (as evidenced in the figure below). chart 1).

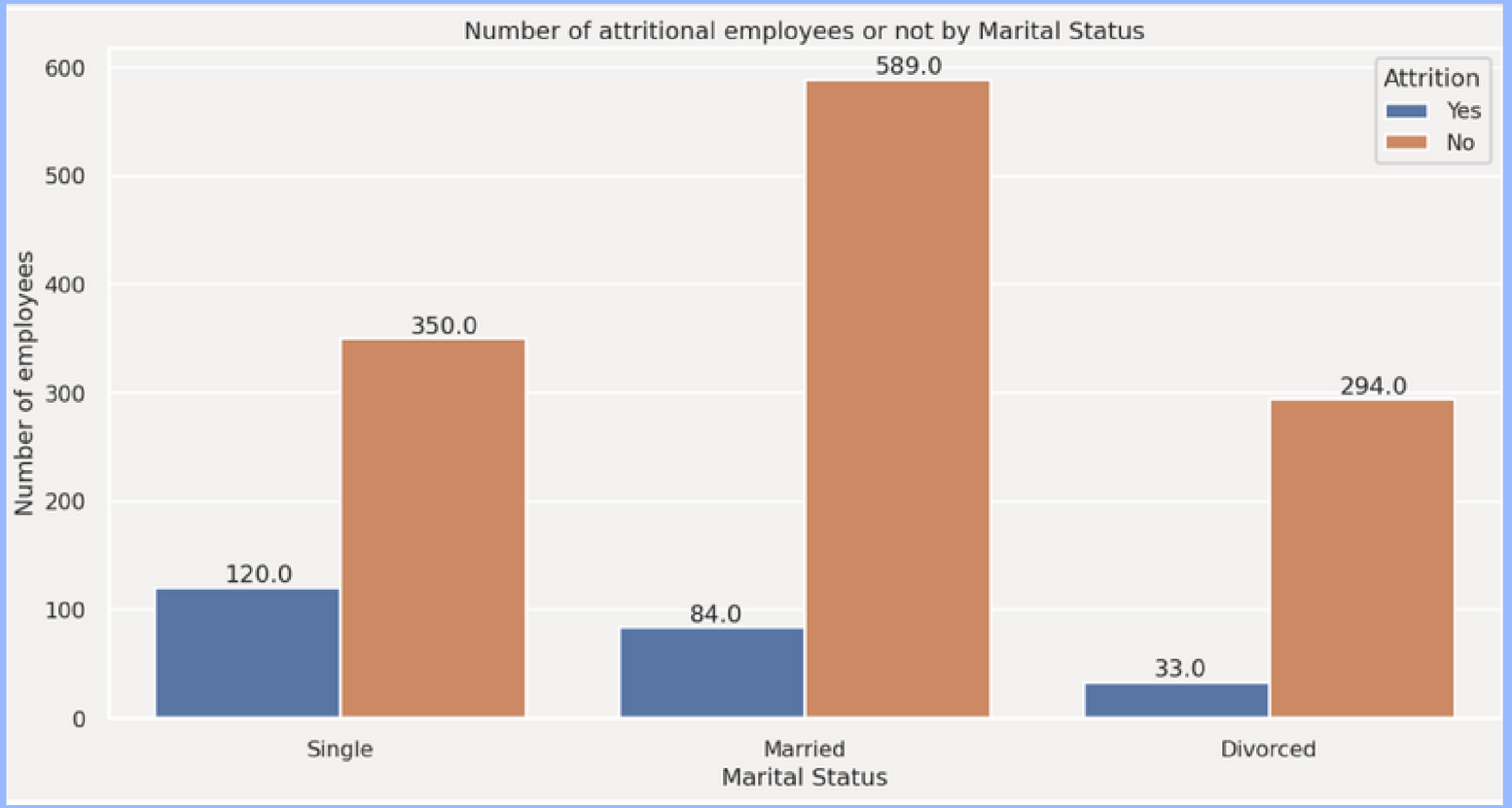
Percentage of employees by age in the company



- The number of employees aged between 31-35 accounts for the highest proportion (23%)
- Employees between the ages of 26-30 and 36-40 account for a high percentage (18%)
- The number of employees aged 18-20 accounts for the lowest percentage (2%).
- Employees between the ages of 56-60 are also quite low (3%)

=> This is a reasonable distribution because the age groups with the highest percentage are the most productive age groups.

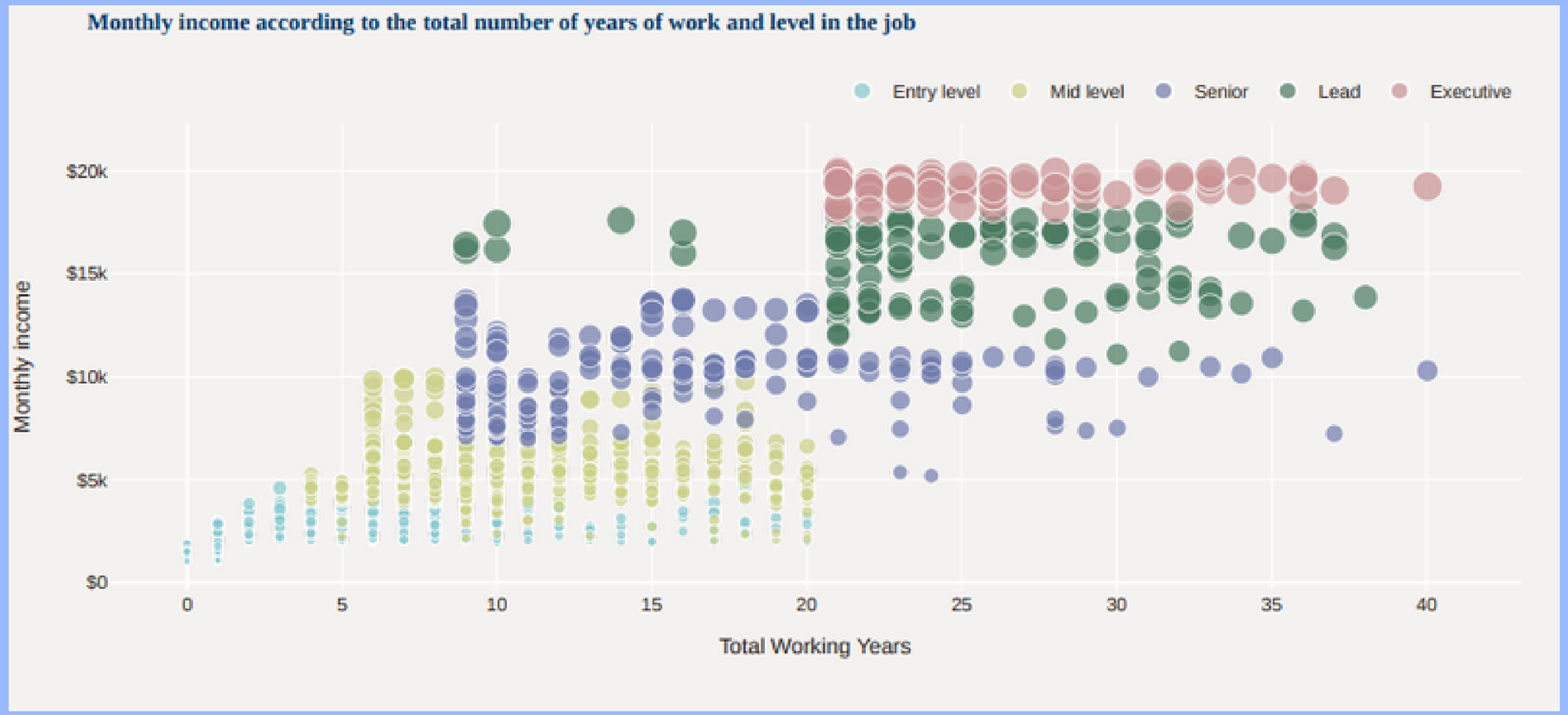
NUMBER OF ATTRITIONAL EMPLOYEES OR NOT BY MARITAL STATUS



Based on the compound column chart, we have the following comments:

- The percentage of employees who are married accounts for the largest number, this is completely appropriate when the age of employees in this company is from 26-40 accounting for about 60% (Figure 9), this is the appropriate age to set up a company. family.
- The rate of single employees, although ranked second, has the highest turnover rate. This can be seen that young and free people (not having to worry about family) tend to "jump work" to experience many different working environments and find jobs that match their interests.

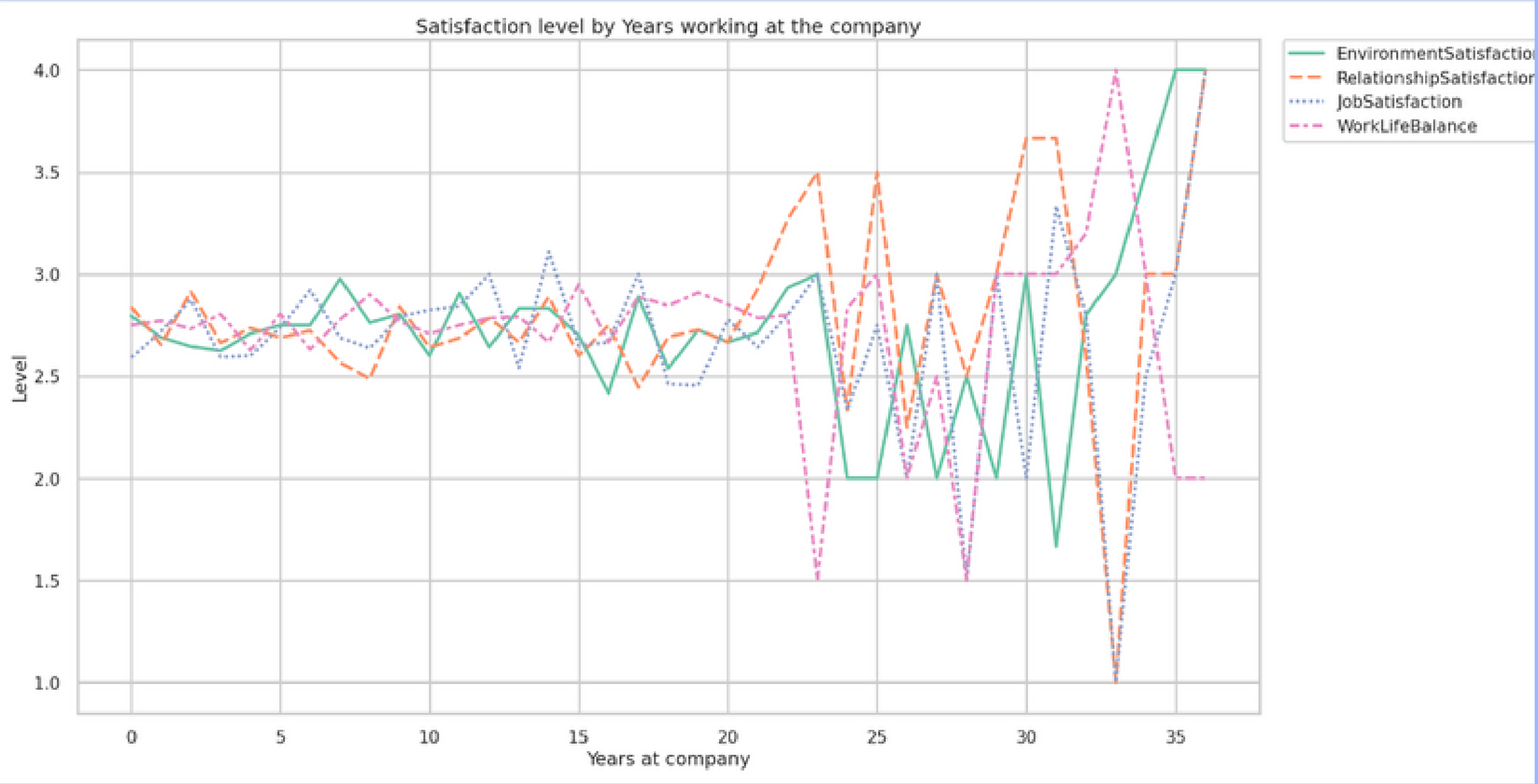
Monthly income according to the total number of years of work and level in the job



Based on the scatter plot above, we can see:

- The higher the rank in the job, the more monthly income tends to be.
- Levels such as: Entry level, Mid level have the number of working years in the range of 0 - 20 years. While to be able to be in the highest position of Executive, it is necessary to work for more than 20 years, even over 35 years.

Satisfaction level by Years working at the company



- There is a clear difference between employee satisfaction by age, which is clearly separated at the 20th working year.
- Group under 20 years old: less volatile, fluctuates at 2.5-3.
- Group over 20 years old:

- More volatility. The more you stick with the company, the more you can access different aspects of the job, the more diverse the needs that people require at work.

- Work-life balance often tends to be the opposite of the rest of the characteristics. This shows job satisfaction in 3 aspects (environment, relationship, work) that are often parallel and related to each other. In addition, satisfactionis also a great motivation for employees, they tend to spend more time on work. Since then, the time for other life activities is also reduced, affecting work-life balance.

-The more attached to the company, the more obvious the influence. Even then the employees already have working experience in that company

Monthly income by Job Roles and Attrition rate

- Salary distribution of different positions. In which the highest are Manager and Research Director, these are management positions. These are also the 2 groups with the lowest number of "regret" employees.
- Human Resources positions have a small number, but the salary range is also quite wide, showing that although this profession has few employees, the salary also varies among individuals.
- When dividing about Attrition rating, we see a clear grouping with 2 positions: Human Resources and Sales Representative. Those who vote Yes are in the lower salary group, so it can be seen that salary has a lot of influence on employees' evaluation of jobs in these two industries. In the remaining industries, Attrition vote does not have much difference in salary ranks (the dots are quite mixed together).

4.MACHINE LEARNING MODELS

4.1.THE PROBLEM POSED

Problem posed:

Predict whether employees are at risk of leaving the company they are working for?

General introduction

- In machine learning, supervised learning is a group of popular algorithms in this field and one of the important problems of supervised learning is the classification problem.
- There are two common types of classification: binary classification and multiclassification, and in the problem that the group poses, this is a binary classification problem: from the input attributes of an employee such as the total number of years of work, salary level. , satisfaction with the working environment, ... predicts whether that employee is at risk of leaving the current company (whether the company loses employees or not) (0: No / 1: Yes).
- The team will create a logistic regression model for the binary classification problem.

4.2. DATA PREPROCESSING

Encoding catalog attributes in numerical form

```
#filter out category attributes
cat_cols=df_copy.select_dtypes(exclude=['int32','int64','float32','float64'])
cat_cols.head()
```

	Attrition	BusinessTravel	Department	EducationField	Gender	JobRole	MaritalStatus	Over18	OverTime
0	Yes	Travel_Rarely	Sales	Life Sciences	Female	Sales Executive	Single	Y	Yes
1	No	Travel_Frequently	Research & Development	Life Sciences	Male	Research Scientist	Married	Y	No
2	Yes	Travel_Rarely	Research & Development	Other	Male	Laboratory Technician	Single	Y	Yes
3	No	Travel_Frequently	Research & Development	Life Sciences	Female	Research Scientist	Married	Y	Yes
4	No	Travel_Rarely	Research & Development	Medical	Male	Laboratory Technician	Married	Y	No

```
#count the number of distinct values in each of the above category attributes
count_uvalue=[cat_cols[c].nunique() for c in list(cat_cols.columns)]
count_uvalue
```

```
[2, 3, 3, 6, 2, 9, 3, 1, 2]
```

A few comments are as follows:

- The Over18 column contains only 1 type of value, 'Y', which proves that this column will not affect the results of the model training.
- There are 3 properties, Attrition, Gender, and OverTime, which have 2 different types of values, so with these properties we can encode by assigning labels 0/1 to them.
- For attributes with more than 2 value types, we will highlight with a one-hot vector, the reason for using one-hot without using ordinal or label to avoid bias caused by encoding into the values. large value if the number of values is large.

```
#remove attribute 'Over18'
df_copy.drop(['Over18'],axis=1, inplace=True)
```

```
#convert category attributes with only 2 distinct values to numeric by assigning labels
from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
df_copy['Attrition']=label_encoder.fit_transform(df['Attrition'])
df_copy['OverTime']=label_encoder.fit_transform(df['OverTime'])
df_copy['Gender']=label_encoder.fit_transform(df['Gender'])
```

```
#convert category attributes with more than 2 distinct values to numeric using one-hot vector
df_copy=pd.get_dummies(df_copy, columns=['BusinessTravel', 'Department', 'EducationField',
                                         'JobRole', 'MaritalStatus'])
```

```
df_copy.head()
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	Gender	HourlyRate	...	JobRole_l
0	41	1	1102	1	2	1	1	2	0	94	...	
1	49	0	279	8	1	1	2	3	1	61	...	
2	37	1	1373	2	2	1	4	4	1	92	...	
3	33	0	1392	3	4	1	5	4	0	56	...	
4	27	0	591	2	1	1	7	1	1	40	...	

5 rows × 53 columns

Eliminate the attributes that do not make sense for the problem

```
#delete 'StandardHours'
df_copy.drop(['StandardHours'],axis=1,inplace=True)
```

Option 1:

- The most rudimentary way that can be thought of as a way of eliminating meaningfully immediately visible attributes that are not necessary for the classification problem: The 'Over18' attribute example is removed above because they only carry a value of 'yes', understandable when the dataset is provided by IBM, based in the US, and the US only allows work when 18 years old or older. In addition, when looking at the table of meanings of the attributes that the team provided in the data collection section, the column 'StandardHours' will also make no sense for the problem because this column carries the same value for all employees. the standard working hours they have to go to work, can delete this attribute.

```
1 df_copy.corr()
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	E
Age	1.000000	-0.159205	0.010661	-0.001686	0.208034	NaN	-0.010145	
Attrition	-0.159205	1.000000	-0.056652	0.077924	-0.031373	NaN	-0.010577	
DailyRate	0.010661	-0.056652	1.000000	-0.004985	-0.016806	NaN	-0.050990	
DistanceFromHome	-0.001686	0.077924	-0.004985	1.000000	0.021042	NaN	0.032916	
Education	0.208034	-0.031373	-0.016806	0.021042	1.000000	NaN	0.042070	
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
EmployeeNumber	-0.010145	-0.010577	-0.050990	0.032916	0.042070	NaN	1.000000	
EnvironmentSatisfaction	0.010146	-0.103369	0.018355	-0.016075	-0.027128	NaN	0.017621	
Gender	-0.036311	0.029453	-0.011716	-0.001851	-0.016547	NaN	0.022556	
HourlyRate	0.024287	-0.006846	0.023381	0.031131	0.016775	NaN	0.035179	
JobInvolvement	0.029820	-0.130016	0.046135	0.008783	0.042438	NaN	-0.006888	
JobLevel	0.509604	-0.169105	0.002966	0.005303	0.101589	NaN	-0.018519	

Option 2:

- The next way is to use the correlations value between each independent variable and the dependent variable: Correlation is a commonly used statistical term that refers to the degree to which two variables have a linear relationship with each other.
  - Correlation has the highest value of 1 (two variables are completely linear) and the lowest if the two variables do not have a linear relationship.
  - The team will create a dataframe named 'correlations' containing the correlations of each column in the dataset to easily comment on the degree of correlation between variables.

	feature1	feature2	correlation
0	JobLevel	MonthlyIncome	0.950300
1	JobLevel	TotalWorkingYears	0.782208
2	MonthlyIncome	TotalWorkingYears	0.772893
3	PercentSalaryHike	PerformanceRating	0.773550
4	YearsAtCompany	YearsInCurrentRole	0.758754
5	YearsAtCompany	YearsWithCurrManager	0.769212
6	YearsInCurrentRole	YearsWithCurrManager	0.714365
7	Department_Human Resources	JobRole_Human Resources	0.904983
8	Department_Sales	JobRole_Sales Executive	0.808869

- **TotalWorkingYears, JobLevel and MonthlyIncome:** Have very high correlation value. Choose to keep MonthlyIncome.
- **PercentSalaryHike and PerformanceRating:** Have a correlation value of 0.77. Choose to keep PerformanceRating.
- **YearsAtCompany, YearsInCurrentRole, and YearsWithCurrManager:** Have high correlation value. Choose to keep YearsAtCompany.
- **Department_Human Resources and JobRole_Human Resources:** Have correlation value of 0.9. Choose to keep JobRole_Human Resources.
- **Department_Sales and JobRole_Sales Executive:** Have correlation value of 0.8. Choose to keep JobRole_Sales Executive.

```
df_copy.drop(['TotalWorkingYears', 'JobLevel', 'PercentSalaryHike', 'YearsInCurrentRole',
        'YearsWithCurrManager', 'Department_Human Resources', 'Department_Sales'],axis=1,inplace=True)
```

```
df_copy.head()
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction	Gender	HourlyRate	JobInvolvement	.
0	41	1	1102	1	2	1	2	0	94	3	.
1	49	0	279	8	1	2	3	1	61	2	.
2	37	1	1373	2	2	4	4	1	92	2	.
3	33	0	1392	3	4	5	4	0	56	3	.
4	27	0	591	2	1	7	1	1	40	3	.

xpand output; double click to hide output

5 rows × 44 columns

Option 3: Using correlations between independent variables:

- The higher the correlation between the two independent variables, the more similar information they bring to the context of the problem.
- Therefore, when two independent variables have high correlation, one can be selected to apply to model training.

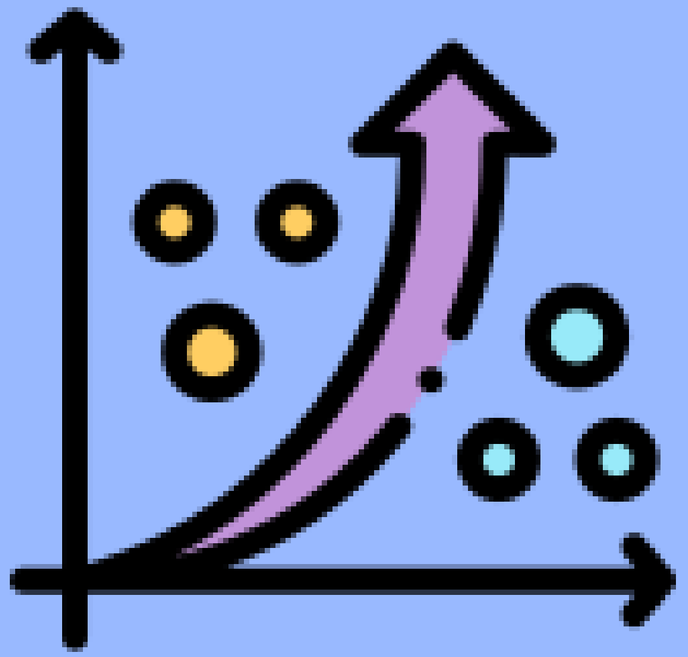
Handling missing values

```
1 df_copy.isnull().sum()
```

```
Age                                  0
Attrition                            0
DailyRate                            0
DistanceFromHome                     0
Education                            0
EmployeeNumber                       0
EnvironmentSatisfaction              0
Gender                               0
HourlyRate                           0
JobInvolvement                       0
JobSatisfaction                      0
MonthlyIncome                        0
MonthlyRate                          0
NumCompaniesWorked                   0
OverTime                             0
PerformanceRating                    0
RelationshipSatisfaction             0
StockOptionLevel                     0
TrainingTimesLastYear                0
WorkLifeBalance                      0
YearsAtCompany                       0
YearsSinceLastPromotion              0
BusinessTravel_Non-Travel            0
BusinessTravel_Travel_Frequently     0
BusinessTravel_Travel_Rarely         0
Department_Research & Development    0
EducationField_Human Resources       0
EducationField_Life Sciences         0
EducationField_Marketing             0
EducationField_Medical               0
EducationField_Other                 0
EducationField_Technical Degree      0
JobRole_Healthcare Representative    0
JobRole_Human Resources              0
JobRole_Laboratory Technician        0
JobRole_Manager                      0
JobRole_Manufacturing Director       0
JobRole_Research Director            0
JobRole_Research Scientist           0
JobRole_Sales Executive              0
JobRole_Sales Representative         0
MaritalStatus_Divorced               0
MaritalStatus_Married                0
MaritalStatus_Single                 0
dtype: int64
```

4.3. DATA PREPROCESSING

Logistic Regression for Binary Classification



Description of the problem:

- The objective of the binary classification problem is to predict the probability of belonging to one of the two classes to be classified of a dependent variable based on independent variables (also known as attributes).
- In this dataset we will try to predict probability of belonging to class 0 or 1 of variable Attrition based on other independent attributes of an employee.

Using Pipeline and Metrics used to evaluate the model

Pipeline:

- Pipeline is a tool that helps combine multiple steps of data processing and model training into a complete process.
- The steps to build a machine learning model will be sequentially in a Pipeline object (can be seen as a pipeline to lead through the steps one by one).

Pipeline saves time and optimizes model training.

Measurements:

- The measures used by the team in this lesson will be:
  - Accuracy score.
  - Precision.
  - Recall.
  - F1-score.

Implement

Create a set of input attributes and a set of target variables from the original data set.

```
1 X=np.array(df_copy.drop(['Attrition'],axis=1))
2 y=np.array(df_copy['Attrition'])
```

```
1 X
```

```
array([[  41, 1102,    1, ...,    0,    0,    1],
       [  49,  279,    8, ...,    0,    1,    0],
       [  37, 1373,    2, ...,    0,    0,    1],
       ...,
       [  27,  155,    4, ...,    0,    1,    0],
       [  49, 1023,    2, ...,    0,    1,    0],
       [  34,  628,    8, ...,    0,    1,    0]])
```

```
1 y
```

```
array([1, 0, 1, ..., 0, 0, 0])
```

Divide data set into training set and test set

```
1  X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.2)
```

```
1  print(X_train.shape)
2  print(X_test.shape)
```

```
(1176, 43)
(294, 43)
```

Model Settings

```
pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', LogisticRegression(solver="liblinear",penalty="l2",
                                      class_weight={0:1,1:1},max_iter=10000))
])

pipe.fit(X_train, y_train)
pipe.named_steps['classifier'].get_params()
```

```
{'C': 1.0,
 'class_weight': {0: 1, 1: 1},
 'dual': False,
 'fit_intercept': True,
 'intercept_scaling': 1,
 'l1_ratio': None,
 'max_iter': 10000,
 'multi_class': 'auto',
 'n_jobs': None,
 'penalty': 'l2',
 'random_state': None,
 'solver': 'liblinear',
 'tol': 0.0001,
 'verbose': 0,
 'warm_start': False}
```

Pipeline Explanation: Building a pipeline performs the following steps:

- First create a 'scaler' component that does the normalization of the data as mentioned at the end of the data preprocessing section. This component uses the StandardScaler() class provided by sklearn to normalize the value of each column in X according to the Standardization method.
- Next is to create a classifier 'classifier' to perform the classification of input samples about one of the two classes of the binary classification problem. This classifier applies the Logistic Regression algorithm that the team introduced earlier (The parameters in the LogisticRegression class will be explained in the accompanying report file).

```
#predict
predictions=pipe.predict(X_test)
predictions
```

```
array([0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0])
```

Evaluate the model: First consider the value of accuracy score.

```
acc_score=accuracy_score(predictions, y_test)
print(acc_score)
```

```
0.8775510204081632
```

Accuracy Score is quite high, but is it good enough for evaluating the effectiveness of the model?
Let's take a closer look at the other metrics.

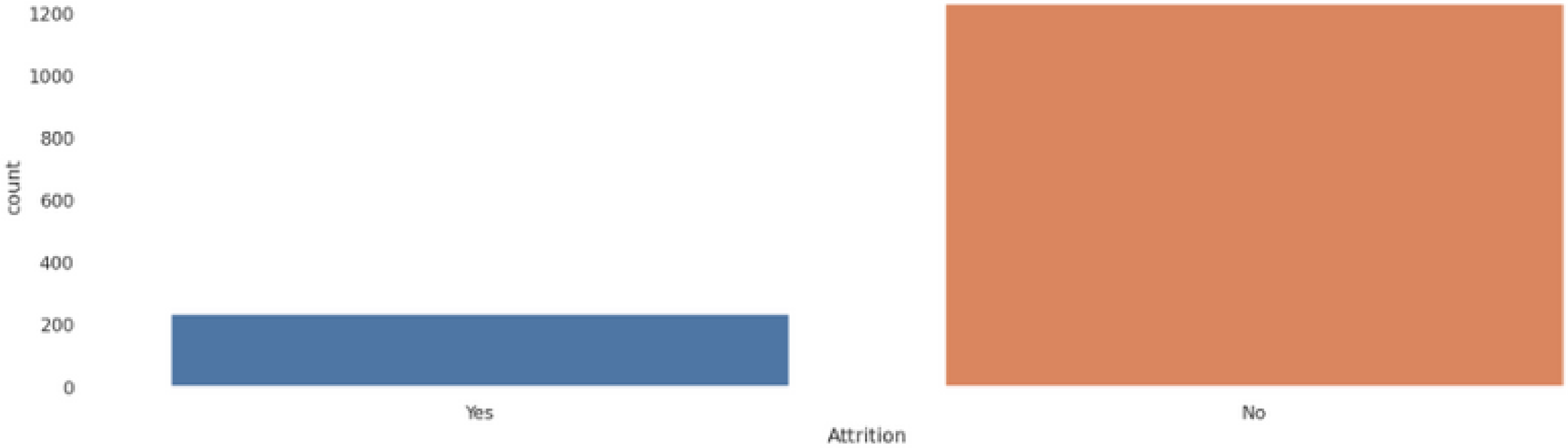
```
report=classification_report(predictions, y_test)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.97      0.89      0.93       267
           1       0.41      0.74      0.53        27

    accuracy                           0.88       294
   macro avg       0.69      0.82      0.73       294
weighted avg       0.92      0.88      0.89       294
```

- Other measures such as Precision, Recall and F1-score are all very low in class 1 than in class 0, proving that the predictive model is not good for predicting fields that actually belong to class 1 (or say it). in this context are samples that belong to the negative class).
- And this makes us think of the case of imbalanced data.

Check data balance

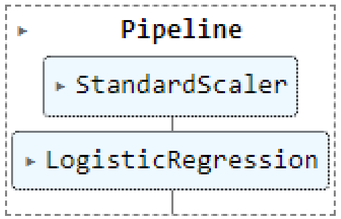
```
sns.set(rc={"axes.facecolor":"white","figure.facecolor":"white"})
sns.set_context("poster",font_scale = .7)
plt.subplots(figsize=(20,6))
sns.countplot(data=df,x=df['Attrition']);
```



- It is clear that an imbalance occurs between classes 0 and 1 when the number of elements of class 0 in this dataset is approximately 6 times the number of elements of class 1.
- This serious imbalance greatly affects the accuracy of the model.

Use weights to increase model penalty if wrong class prediction:

```
pipe2 = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', LogisticRegression(solver="liblinear",penalty="l2",
                                      class_weight={0:1,1:2},max_iter=10000))
])

pipe2.fit(X_train, y_train)
```




```
predictions2=pipe2.predict(X_test)
```

```
acc_score2=accuracy_score(predictions2, y_test)
print(acc_score2)
```

```
0.8741496598639455
```

```
report2=classification_report(predictions2, y_test)
print(report2)
```

```
              precision    recall  f1-score   support

           0       0.93      0.92      0.92       246
           1       0.61      0.62      0.62        48

    accuracy                           0.87       294
   macro avg       0.77      0.77      0.77       294
weighted avg       0.88      0.87      0.87       294
```

- Using a larger weight for class 1 has improved the precision and f1-score of class 1 significantly, showing that the overarching of the model has been increased. However, these measures have not yet reached a really good value for a classification problem.
- And the group also experimented with many different weight pairs, but the best result was {0:1, 1:2}.

Apply SMOTE algorithm

```
from imblearn.over_sampling import SMOTE
X,y=SMOTE(sampling_strategy=1, random_state=0).fit_resample(X, y)
```

```
sns.set(rc={"axes.facecolor":"white","figure.facecolor":"white"})
sns.set_context("poster",font_scale = .7)
plt.subplots(figsize=(20,6))
sns.countplot(x=y);
```

Divide data set into training set and test set

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.2)
```

```
print(X_train.shape)
print(X_test.shape)
```

```
(1972, 43)
(494, 43)
```

Model Settings

```
pipe3 = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', LogisticRegression(solver="liblinear",penalty="l2",
                                      class_weight={0:1,1:1},max_iter=10000))
])

pipe3.fit(X_train, y_train)
pipe3.named_steps['classifier'].get_params()
```

```
{'C': 1.0,
 'class_weight': {0: 1, 1: 1},
 'dual': False,
 'fit_intercept': True,
 'intercept_scaling': 1,
 'l1_ratio': None,
 'max_iter': 10000,
 'multi_class': 'auto',
 'n_jobs': None,
 'penalty': 'l2',
 'random_state': None,
 'solver': 'liblinear',
 'tol': 0.0001,
 'verbose': 0,
 'warm_start': False}
```

```
predictions3=pipe3.predict(X_test)
predictions3
```

```
array([0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0,
       0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 1, 1])
```

Model rating

```
#First look at the model's accuracy score
acc_score3=accuracy_score(predictions3, y_test)
print(acc_score3)
```

```
0.9048582995951417
```

```
#More detailed look at other metric values like F1-score or Recall score
report3=classification_report(predictions3, y_test)
print(report3)
```

```
              precision    recall  f1-score   support

           0       0.98      0.84      0.91       268
           1       0.84      0.98      0.90       226

    accuracy                           0.90       494
   macro avg       0.91      0.91      0.90       494
weighted avg       0.92      0.90      0.90       494
```

- All measures are at 90% or more, showing the effectiveness of the SMOTE algorithm on this dataset.

5.SUMMARY

- Accuracy score: 90%.
- Precision macro average: 91%.
- Recall macro average: 91%.
- F1-Score macro average: 90%.

- Recall value (for Attrition =Yes) tells us the number of employees which might leave the organization
- hoping that the company will care more about their employees and improve their job satisfaction
- they must pay more attention to human resources employees because they have very low job satisfaction

THANK YOU FOR LISTENING!