Employee Attrition Analysis and Prediction

HR Analytics: Predicting and Preventing Employee Attrition

Index:

Preprocessing

For this Columns:- **job roles, performance, tenure, and exit interviews**

- **Data Collection & Preprocessing:** Gather relevant employee data, including demographics, job roles, performance, tenure, and exit interviews. Clean and preprocess the data to handle missing values, outliers, and categorical variables.

1. Tenure-Related Columns (Employment Duration & Experience):

These columns measure how long an employee has been with the company or in a specific role:

- `TotalWorkingYears` → Total years of work experience.
- `YearsAtCompany` → Number of years at the current company.
- `YearsInCurrentRole` → Number of years in the current job role.
- `YearsSinceLastPromotion` → Years since the last promotion.
- `YearsWithCurrManager` → Years working with the current manager.

2. Exit Interview-Related Columns (Employee Attrition & Resignation Factors):

These columns indicate whether an employee has left the company and potential reasons for leaving:

- `Attrition` → Whether the employee has left the company (`Yes/No`).
- `JobSatisfaction` → Job satisfaction level (low satisfaction might indicate exit reasons).
- `EnvironmentSatisfaction` → Satisfaction with the work environment.
- `JobInvolvement` → Employee engagement level.
- `WorkLifeBalance` → Work-life balance rating.
- `OverTime` → Employees working overtime might be more prone to leaving.
- `NumCompaniesWorked` → Employees with multiple past employers might have a higher attrition risk.

No Columns Have Missing Values

If Missing Values Have

1) Check Columns have Normal Distribution if not a Normal Distribution Use Median else Use Mean

2) Check the Columns have Outliers if Outliers Have Use Median Otherwise Use Mean

3) Categorical Columns have missing values use Mode

4) Times Series Columns have missing values use ffill and bfill

5) volunterly missed a data or confidential data means use custom value to fill

Handling Outliers:

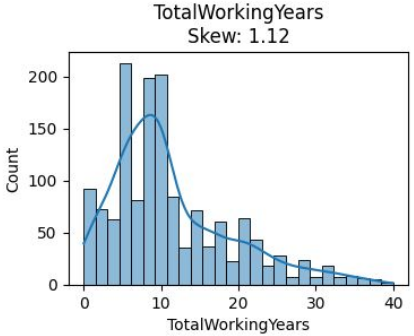
A. Based on Data Type & Distribution

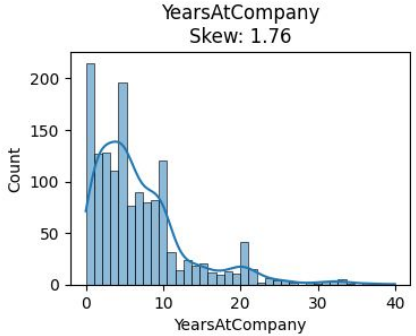
Data Type	Suggested Method
Normally Distributed	Winsorization (Capping) or Z-score filtering
Highly Skewed Data	Log/Square Root transformation
Categorical Data	No outlier treatment needed
Extreme Outliers	Trimming (removal)

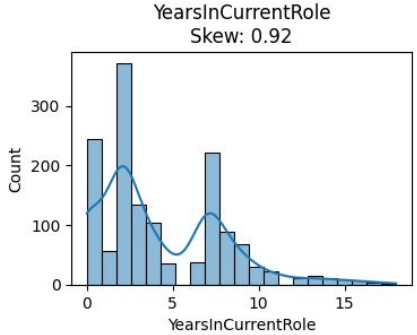
B. Based on Business Impact

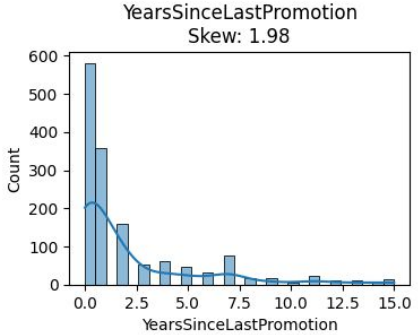
Scenario	Suggested Method
Critical outliers impact business (e.g., salary, age, fraud detection)	Winsorization (keep all values)
Outliers are errors/noisy data	Remove outliers
Outliers hold important insights (e.g., fraud, anomalies)	Keep outliers for model learning

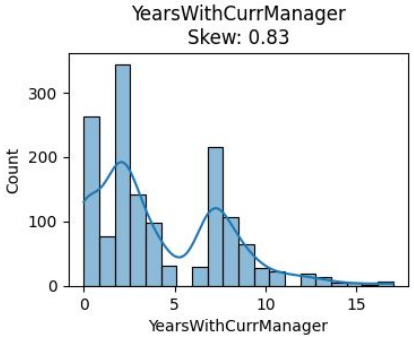
- **If you want to reduce skewness but keep all data → Log or Square Root Transformation (Recommended)**
- **If extreme values are important but need capping → Winsorization**
- **If outliers are errors and should be removed → IQR Filtering**





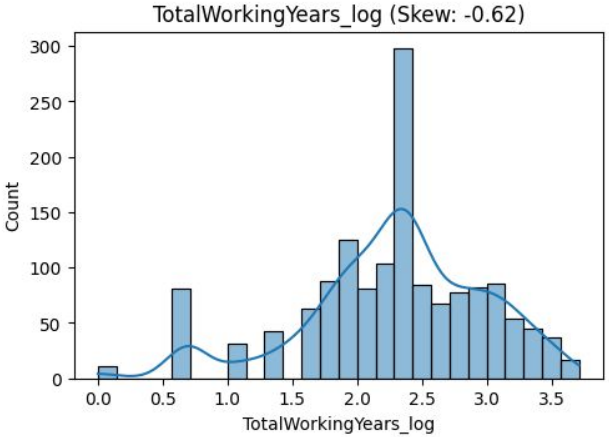


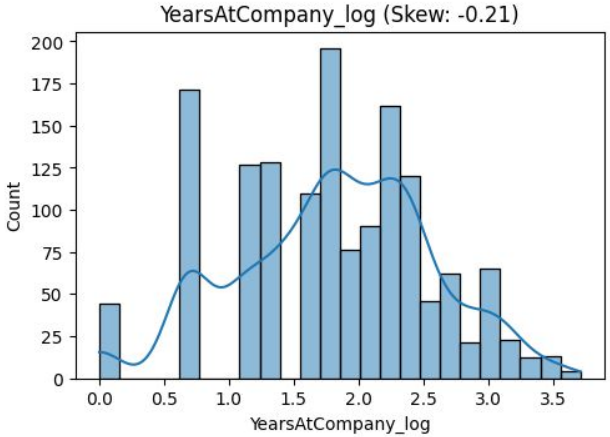


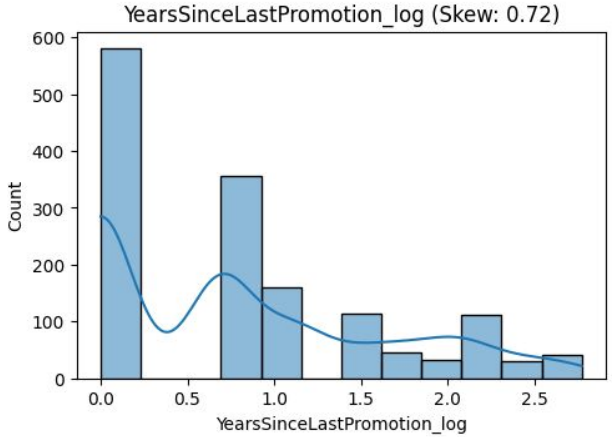


Column	Skewness	Suggested Transformation
TotalWorkingYears	1.12	Log or Square Root
YearsAtCompany	1.76	Log or Box-Cox
YearsInCurrentRole	0.92	No major transformation needed (near-normal)
YearsSinceLastPromotion	1.98	Log, Square Root, or Box-Cox
YearsWithCurrManager	0.83	No major transformation needed

```
df_pre_tenure['TotalWorkingYears_log'] = np.log1p(df_pre_tenure['TotalWorkingYears'])
df_pre_tenure['YearsAtCompany_log'] = np.log1p(df_pre_tenure['YearsAtCompany'])
df_pre_tenure['YearsSinceLastPromotion_log'] = np.log1p(df_pre_tenure['YearsSinceLastPromotion'])
```

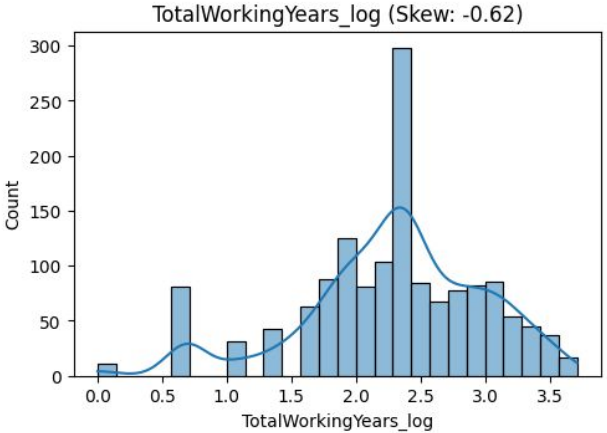


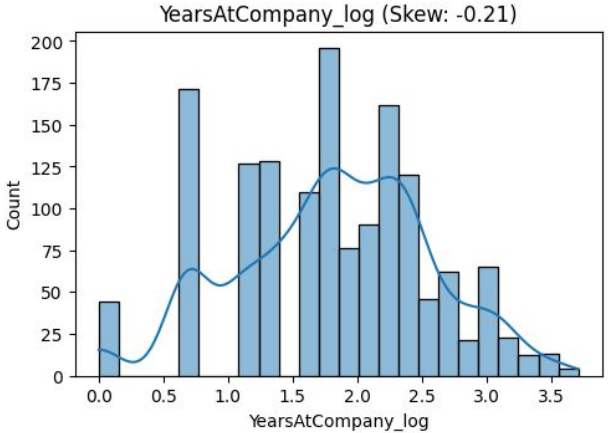


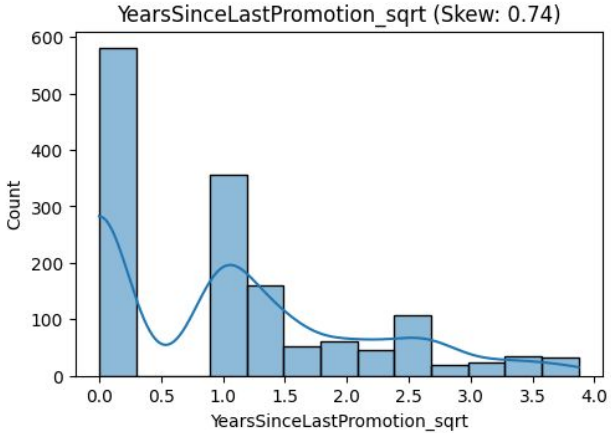


- **TotalWorkingYears_log:** Skew reduced to **-0.62** (almost normal, good).
- **YearsAtCompany_log:** Skew reduced to **-0.21** (very close to normal, good).
- **YearsSinceLastPromotion_log:** Skew reduced to **0.72** (still slightly skewed but much better).

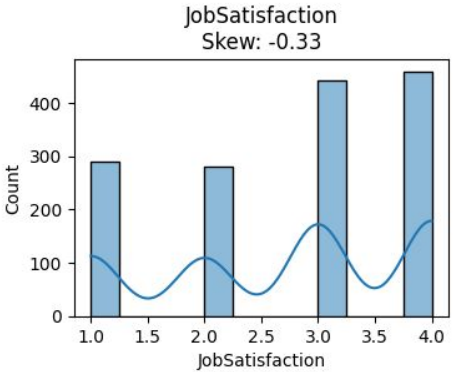
```
df_pre_tenure['YearsSinceLastPromotion_sqrt'] = np.sqrt(df_pre_tenure['YearsSinceLastPromotion'])

# Check new skewness
print(df_pre_tenure['YearsSinceLastPromotion_sqrt'].skew())
```

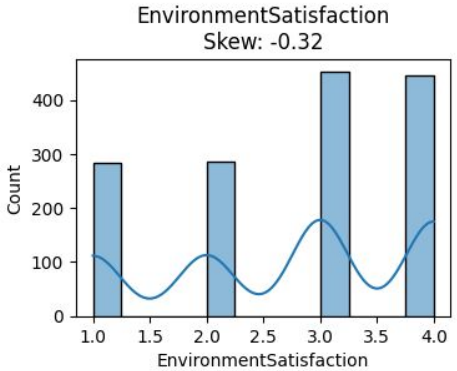


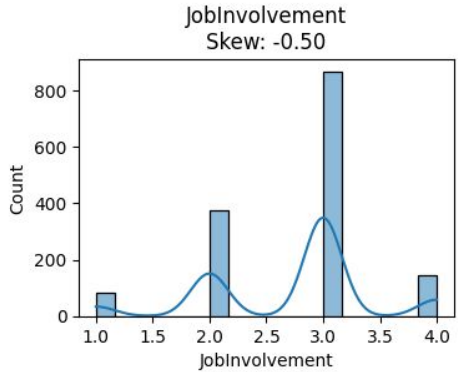


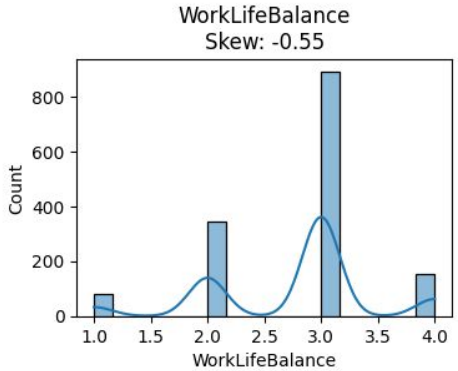


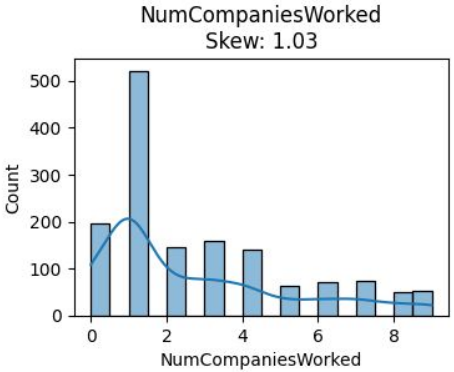
Years since last promotion log outlier treatment better than sqrt treatment











- **JobSatisfaction, EnvironmentSatisfaction, JobInvolvement, and WorkLifeBalance** have skew values between **-0.33 and -0.55**, which are relatively small. No transformation is needed here.
- **NumCompaniesWorked has a skew of 1.03**, which indicates moderate right skewness.

Attach all Columns

```
[ ]  1 print(df_pre1.shape, df_pre_tenure.shape, df_pre_exit_interviews.shape)
     2 print(df_pre1.index.equals(df_pre_tenure.index))
     3 print(df_pre1.index.equals(df_pre_exit_interviews.index))
```

```
(1244, 2) (1470, 5) (1470, 7)
False
False
```

Index Not Matched

So now Used Inner Join

```
[ ]  1 df_prep_in = pd.concat([df_pre1, df_pre_tenure, df_pre_exit_interviews], axis=1, join='inner')
```

```
[ ]    1 df_prep_in.shape
```

```
(1244, 14)
```

Extreme Outliers was removed

Scenario	Use Log2	Use Sqrt	Use Winsorization
Highly skewed data	✅ Best choice	❌ Not strong enough	❌ Doesn't help
Moderate skew	❌ Too aggressive	✅ Better choice	✅ Can help
Large outliers	✅ Works well	❌ Not effective	✅ Helps (truncates extreme values)
Small outliers	❌ Over-compresses	✅ Handles them better	✅ Best option
Preserve original values	❌ Changes scale	❌ Changes scale	✅ Keeps range
Use in linear regression	✅ Good for normality	✅ Good if mild skew	✅ Helps remove influence of extreme values

2nd-preprocessing-steps

Encoding Steps - Sperate Numerical and Categorical Columns

```
[ ]  1 # Separate numerical and categorical columns
     2 numerical_cols = df.select_dtypes(include=['number']).columns
     3 categorical_cols = df.select_dtypes(exclude=['number']).columns
     4
     5 print("Numerical Columns:", numerical_cols.tolist())
     6 print("Categorical Columns:", categorical_cols.tolist())
```

```
Numerical Columns: ['Unnamed: 0', 'Age', 'DailyRate', 'DistanceFromHome', 'Education', 'EmployeeCount', 'EmployeeNumber', 'HourlyRate', 'JobLevel', '
Categorical Columns: ['BusinessTravel', 'Department', 'EducationField', 'Gender', 'MaritalStatus', 'Over18', 'JobRole', 'Attrition', 'OverTime']
```

```
[ ]  1 categorical_cols.shape
```

```
(9,)
```

```
[ ]  1 categorical_cols.tolist()
```

```
['BusinessTravel',
 'Department',
 'EducationField',
 'Gender',
 'MaritalStatus',
 'Over18',
 'JobRole',
 'Attrition',
 'OverTime']
```

Column	Type	Best Encoding	Why?
BusinessTravel (Rarely, Frequently, Non-Travel)	**Ordinal**	✅ **Ordinal Encoding**	"Non-Travel" < "Rarely" < "Frequently" (Travel frequency has order).
Department (Sales, R&D, HR)	**Nominal**	✅ **One-Hot Encoding**	No order among departments.
EducationField (Life Sciences, Marketing, Medical, etc.)	**Nominal**	✅ **One-Hot Encoding**	No ranking between education fields.
Gender (Male, Female)	**Nominal**	✅ **Label Encoding (0/1)**	Only two categories, so label encoding is sufficient.
MaritalStatus (Single, Married, Divorced)	**Nominal**	✅ **One-Hot Encoding**	No strict ranking (Single ≠ Married ≠ Divorced).
Over18 (Y, N)	**Nominal**	✅ **Label Encoding (0/1)**	Only two categories.
JobRole (Manager, Sales Rep, etc.)	**Nominal**	✅ **One-Hot Encoding**	No order between job roles.
Attrition (Yes, No)	**Nominal**	✅ **Label Encoding (0/1)**	Binary classification target variable.
OverTime (Yes, No)	**Nominal**	✅ **Label Encoding (0/1)**	Binary category.

Encoding Type	Columns
Ordinal Encoding	`BusinessTravel`
One-Hot Encoding	`Department`, `EducationField`, `MaritalStatus`, `JobRole`
Label Encoding (0/1)	`Gender`, `Over18`, `Attrition`, `OverTime`

JobRole Dataset has 9 unique value to avoid high cause of dimensionality use Freqency Encoding

```
[ ]  1 df['JobRole_FreqEncoded'] = df['JobRole'].map(df['JobRole'].value_counts())
```

```
[ ]  1 df['JobRole_FreqEncoded'].value_counts()
```

JobRole_FreqEncoded	count
285	285
243	243
217	217
118	118

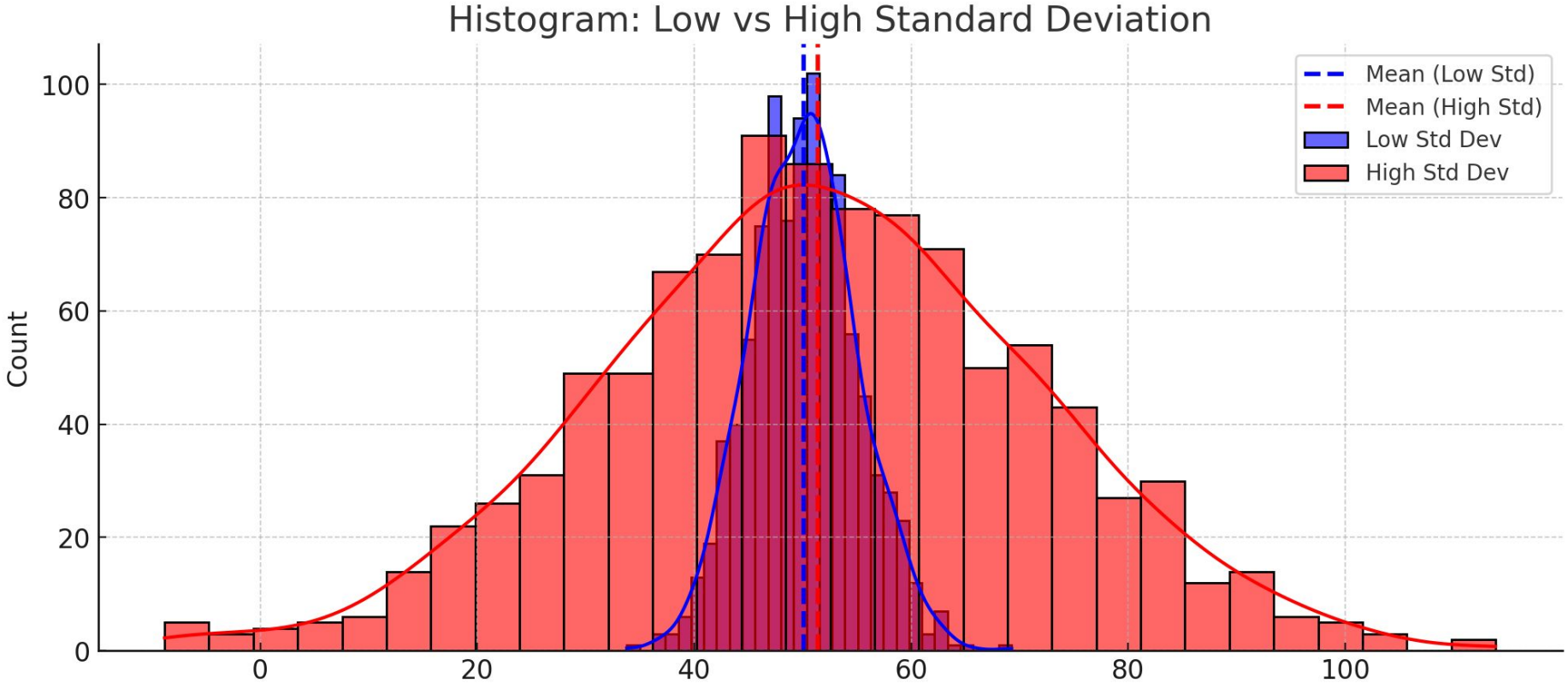
EDA - Exploratory Data Analysis

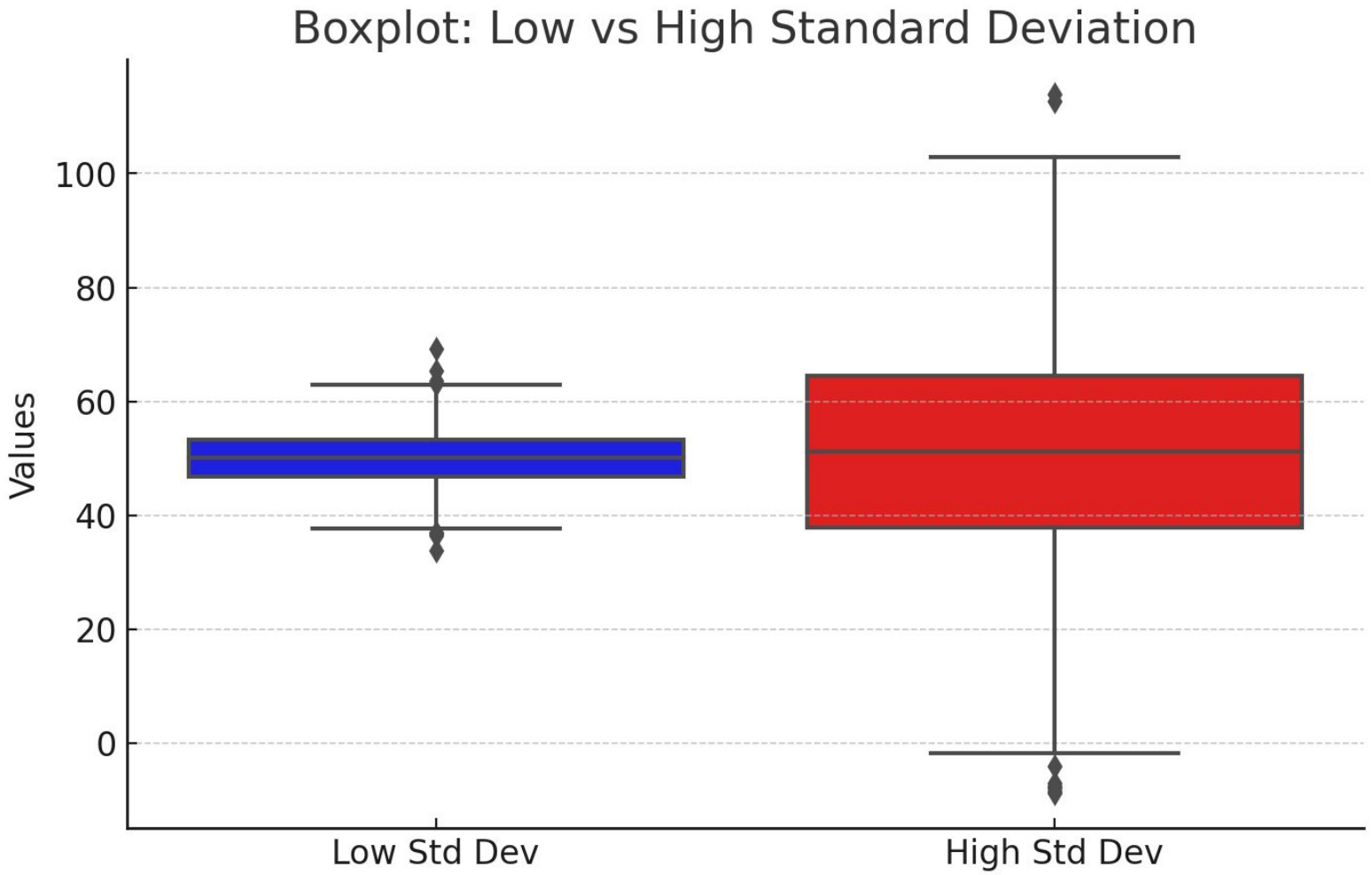
```
1 df.describe()  # Shows count, mean, std, min, max, etc.
```

	Age	BusinessTravel	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	Gender	HourlyRate	
count	1244.000000	1244.000000	1244.000000	1244.000000	1244.000000	1244.0	1244.000000	1244.000000	1244.000000	12
mean	36.916399	1.086013	802.404341	9.098875	2.923633	1.0	1030.087621	0.602894	65.909968	
std	9.100021	0.527442	402.877088	7.995717	1.016258	0.0	596.246903	0.489495	20.278748	
min	18.000000	0.000000	102.000000	1.000000	1.000000	1.0	1.000000	0.000000	30.000000	
25%	30.000000	1.000000	466.750000	2.000000	2.000000	1.0	497.750000	0.000000	48.000000	
50%	36.000000	1.000000	804.500000	7.000000	3.000000	1.0	1029.500000	1.000000	66.000000	
75%	43.000000	1.000000	1154.750000	14.000000	4.000000	1.0	1553.500000	1.000000	84.000000	
max	60.000000	2.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	1.000000	100.000000	

8 rows × 33 columns

Statistics of Dataset - Analysing Total Counts, Average Values, Standard Deviation [Low Standard Deviation Means by Data not widely Spread - High Standard Deviation Means by Data Widely Spread that means maximum and minimum value was high] and look 25,50 and 75 Percentile can understand the IQR


Histogram: Low vs High Standard Deviation
Mean (Low Std)
Mean (High Std)
Low Std Dev
High Std Dev
Count
0
20
40
60
80
100
0
20
40
60
80
100


Boxplot: Low vs High Standard Deviation
Values
100
80
60
40
20
0
Low Std Dev
High Std Dev

- The **Low Std Dev** data has a smaller IQR (box size) and shorter whiskers.
- The **High Std Dev** data has a larger spread, showing greater variability.

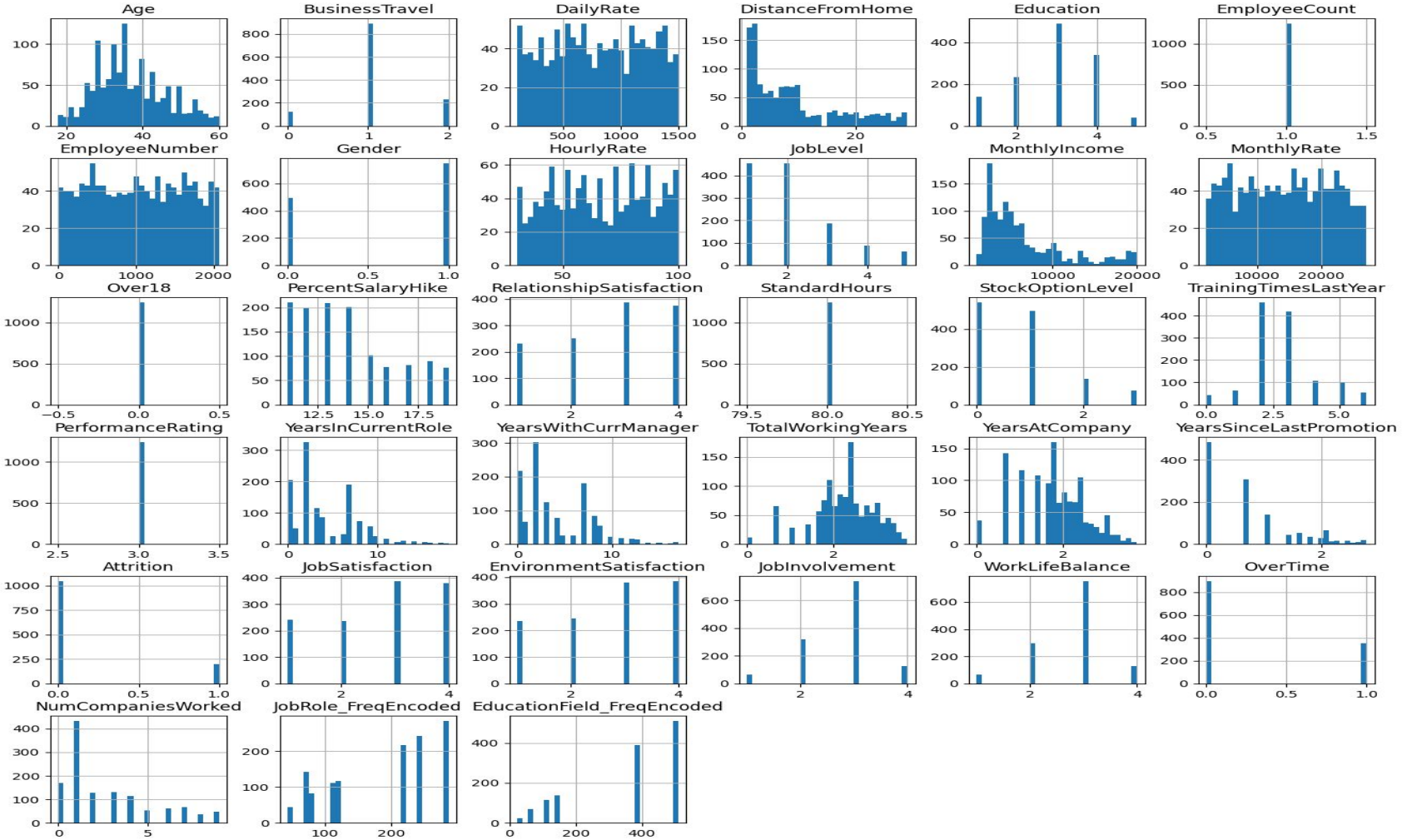
The higher standard deviation indicates a wider distribution, meaning the data points vary more

Data Distribution & Skewness

Check Distribution of Numerical Features

```
[ ]
```

```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df.hist(figsize=(16, 16), bins=30)
plt.show()
```


Age
BusinessTravel
DailyRate
DistanceFromHome
Education
EmployeeCount
EmployeeNumber
Gender
HourlyRate
JobLevel
MonthlyIncome
MonthlyRate
Over18
PercentSalaryHike
RelationshipSatisfaction
StandardHours
StockOptionLevel
TrainingTimesLastYear
PerformanceRating
YearsInCurrentRole
YearsWithCurrManager
TotalWorkingYears
YearsAtCompany
YearsSinceLastPromotion
Attrition
JobSatisfaction
EnvironmentSatisfaction
JobInvolvement
WorkLifeBalance
OverTime
NumCompaniesWorked
JobRole_FreqEncoded
EducationField_FreqEncoded

Check Skewness

```
[ ]  1 print(df.select_dtypes(include=[np.number]).skew())
```

```
Age                          0.406708
BusinessTravel               0.093246
DailyRate                   -0.012011
DistanceFromHome             0.973138
Education                   -0.311965
EmployeeCount                0.000000
EmployeeNumber               0.009786
Gender                      -0.421085
HourlyRate                  -0.021131
JobLevel                     1.028196
MonthlyIncome                1.370812
MonthlyRate                 -0.005088
Over18                       0.000000
PercentSalaryHike            0.592151
RelationshipSatisfaction    -0.321198
StandardHours                0.000000
StockOptionLevel             0.970566
TrainingTimesLastYear        0.555063
PerformanceRating            0.000000
YearsInCurrentRole           0.939105
YearsWithCurrManager         0.876310
TotalWorkingYears           -0.628512
YearsAtCompany              -0.180691
YearsSinceLastPromotion      0.718075
Attrition                    1.849274
JobSatisfaction             -0.333759
EnvironmentSatisfaction     -0.333946
JobInvolvement              -0.486684
WorkLifeBalance             -0.533231
OverTime                     0.969269
NumCompaniesWorked           1.015517
JobRole_FreqEncoded         -0.298590
EducationField_FreqEncoded  -0.740541
dtype: float64
```

✅ **Approximately Normal (-0.5 to 0.5):**

- **Age** (0.41)
- **BusinessTravel** (0.09)
- **DailyRate** (-0.01)
- **Education** (-0.31)
- **EmployeeCount** (0.00)
- **EmployeeNumber** (0.01)
- **Gender** (-0.42)
- **HourlyRate** (-0.02)
- **MonthlyRate** (-0.01)
- **RelationshipSatisfaction** (-0.32)
- **PerformanceRating** (0.00)
- **YearsAtCompany** (-0.18)
- **JobSatisfaction** (-0.33)
- **EnvironmentSatisfaction** (-0.33)
- **JobInvolvement** (-0.49)
- **JobRole_FreqEncoded** (-0.29)

📉 **Moderately Skewed (-1 to -0.5 or 0.5 to 1):**

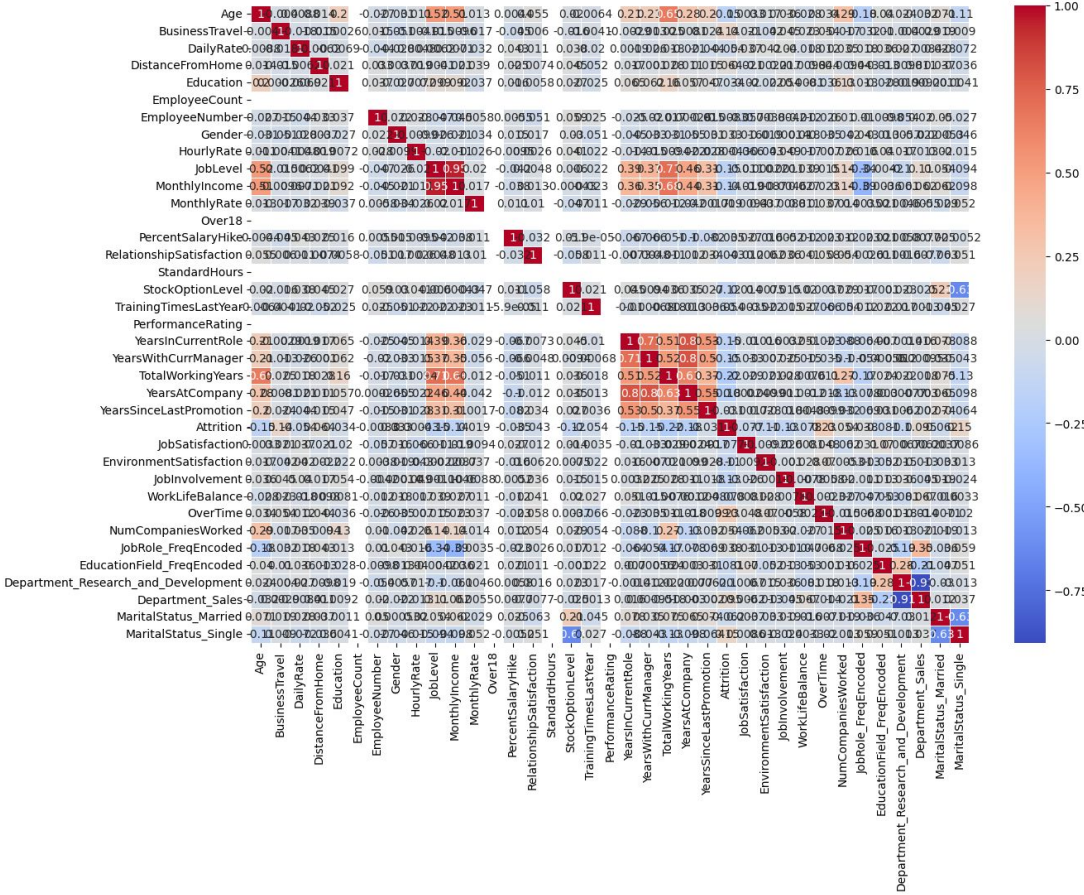
- **DistanceFromHome** (0.97)
- **PercentSalaryHike** (0.59)
- **StockOptionLevel** (0.97)
- **TrainingTimesLastYear** (0.55)
- **YearsInCurrentRole** (0.94)
- **YearsWithCurrManager** (0.88)
- **TotalWorkingYears** (-0.63)
- **YearsSinceLastPromotion** (0.71)
- **WorkLifeBalance** (-0.53)
- **OverTime** (0.97)
- **EducationField_FreqEncoded** (-0.74)

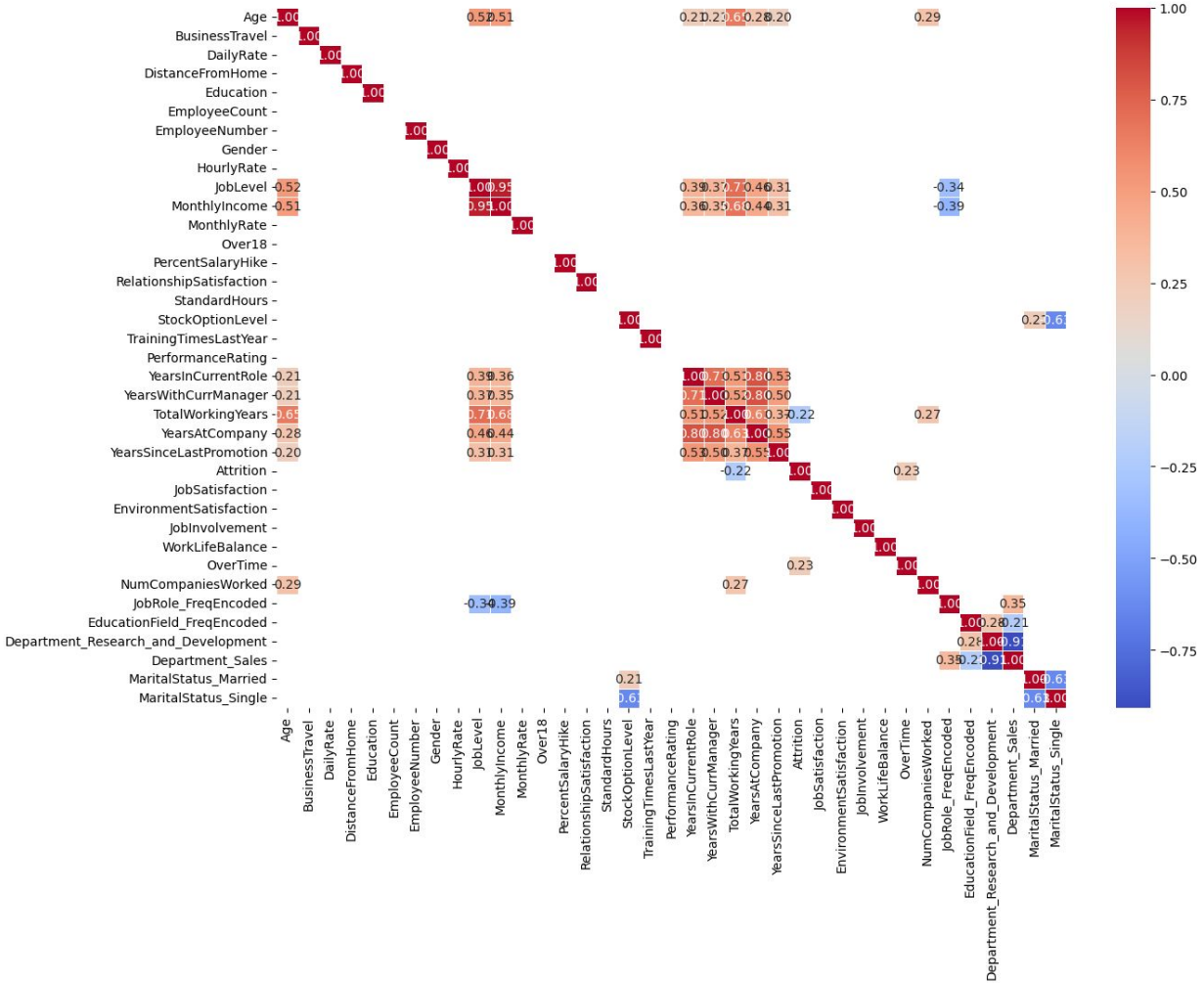
🚨 **Highly Skewed (>|1|):**

- **JobLevel** (1.03)
- **MonthlyIncome** (1.37)
- **Attrition** (1.85)
- **NumCompaniesWorked** (1.02)

Check Correlations & Feature Relationships

Correlation Heatmap




Age
BusinessTravel
DailyRate
DistanceFromHome
Education
EmployeeCount
EmployeeNumber
Gender
HourlyRate
JobLevel
MonthlyIncome
MonthlyRate
Over18
PercentSalaryHike
RelationshipSatisfaction
StandardHours
StockOptionLevel
TrainingTimesLastYear
PerformanceRating
YearsInCurrentRole
YearsWithCurrManager
TotalWorkingYears
YearsAtCompany
YearsSinceLastPromotion
Attrition
JobSatisfaction
EnvironmentSatisfaction
JobInvolvement
WorkLifeBalance
OverTime
NumCompaniesWorked
JobRole_FreqEncoded
EducationField_FreqEncoded
Department_Research_and_Development
Department_Sales
MaritalStatus_Married
MaritalStatus_Single
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Display Strong Correlated Columns

```
corr_pairs
```

	Feature_1	Feature_2	Correlation
379	MonthlyIncome	JobLevel	0.949365
343	JobLevel	MonthlyIncome	0.949365
834	YearsAtCompany	YearsWithCurrManager	0.803513
762	YearsWithCurrManager	YearsAtCompany	0.803513
725	YearsInCurrentRole	YearsAtCompany	0.800686
...	...	...	...
1331	MaritalStatus_Married	MaritalStatus_Single	-0.627702
628	StockOptionLevel	MaritalStatus_Single	-0.634064
1348	MaritalStatus_Single	StockOptionLevel	-0.634064
1291	Department_Sales	Department_Research_and_Development	-0.906566
1255	Department_Research_and_Development	Department_Sales	-0.906566

82 rows × 3 columns

🔹 **Categorical Feature Correlations:**

- **Marital Status (Single)** ↔ **Department (Sales)** (~0.30)
- **JobRole Encoding** ↔ **EducationField Encoding** (~-0.34)
- **OverTime** ↔ **Attrition** (~0.23)

🔹 **Continuous Feature Correlations with Categorical Features:**

- **EducationField Encoding** shows some correlation with **Job Role**.
- **OverTime** and **Attrition** show a weak correlation (~0.23), meaning employees working overtime might have higher attrition.

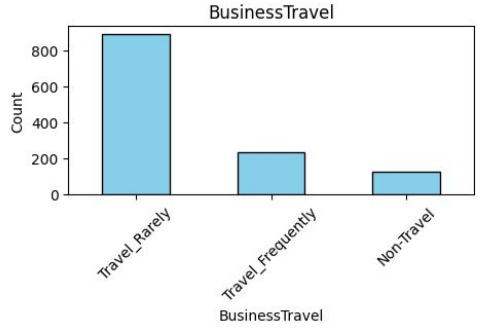
1) Univariate

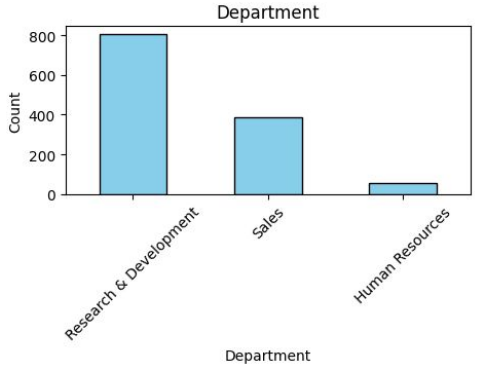
study structure of dataset on Individual Columns

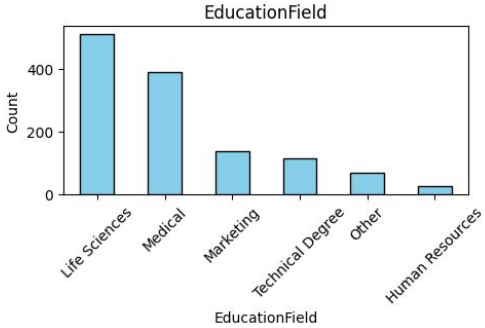
Both Categorical and Numerical Columns

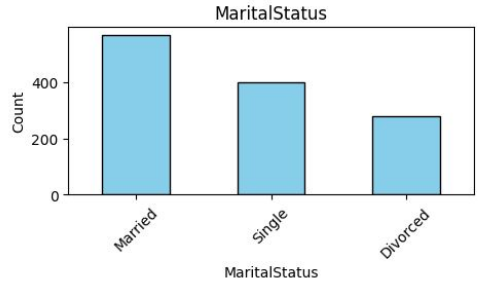
- **Categorical Variables:**
  - `BusinessTravel`
  - `Department`
  - `EducationField`
  - `MaritalStatus`
  - `JobRole`
  - `OverTime`
  - `Attrition`

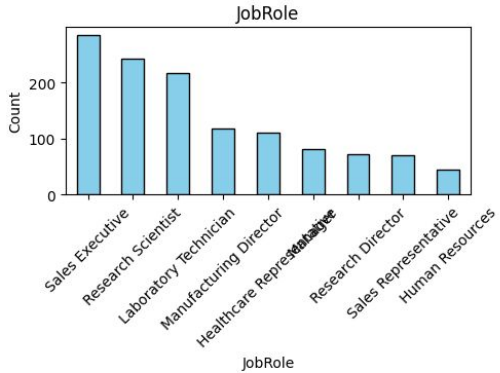
Bar charts, Pie charts → for categorical variables

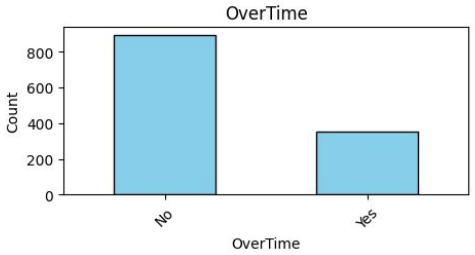

BusinessTravel
Count
0
200
400
600
800
Travel_Rarely
Travel_Frequently
Non-Travel
BusinessTravel

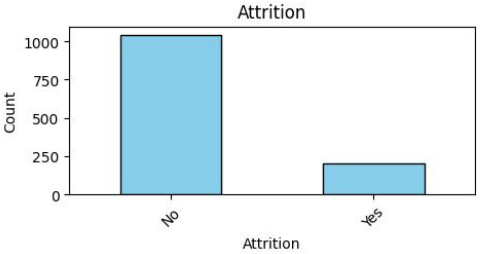

Department
Count
0
200
400
600
800
Research & Development
Sales
Human Resources
Department


EducationField
Count
0
200
400
Life Sciences
Medical
Marketing
Technical Degree
Other
Human Resources
EducationField


MaritalStatus
Count
0
200
400
Married
Single
Divorced
MaritalStatus


JobRole
Count
0
100
200
Sales Executive
Research Scientist
Laboratory Technician
Manufacturing Director
Research Director
Sales Representative
Human Resources
JobRole


OverTime
Count
0
200
400
600
800
No
Yes
OverTime

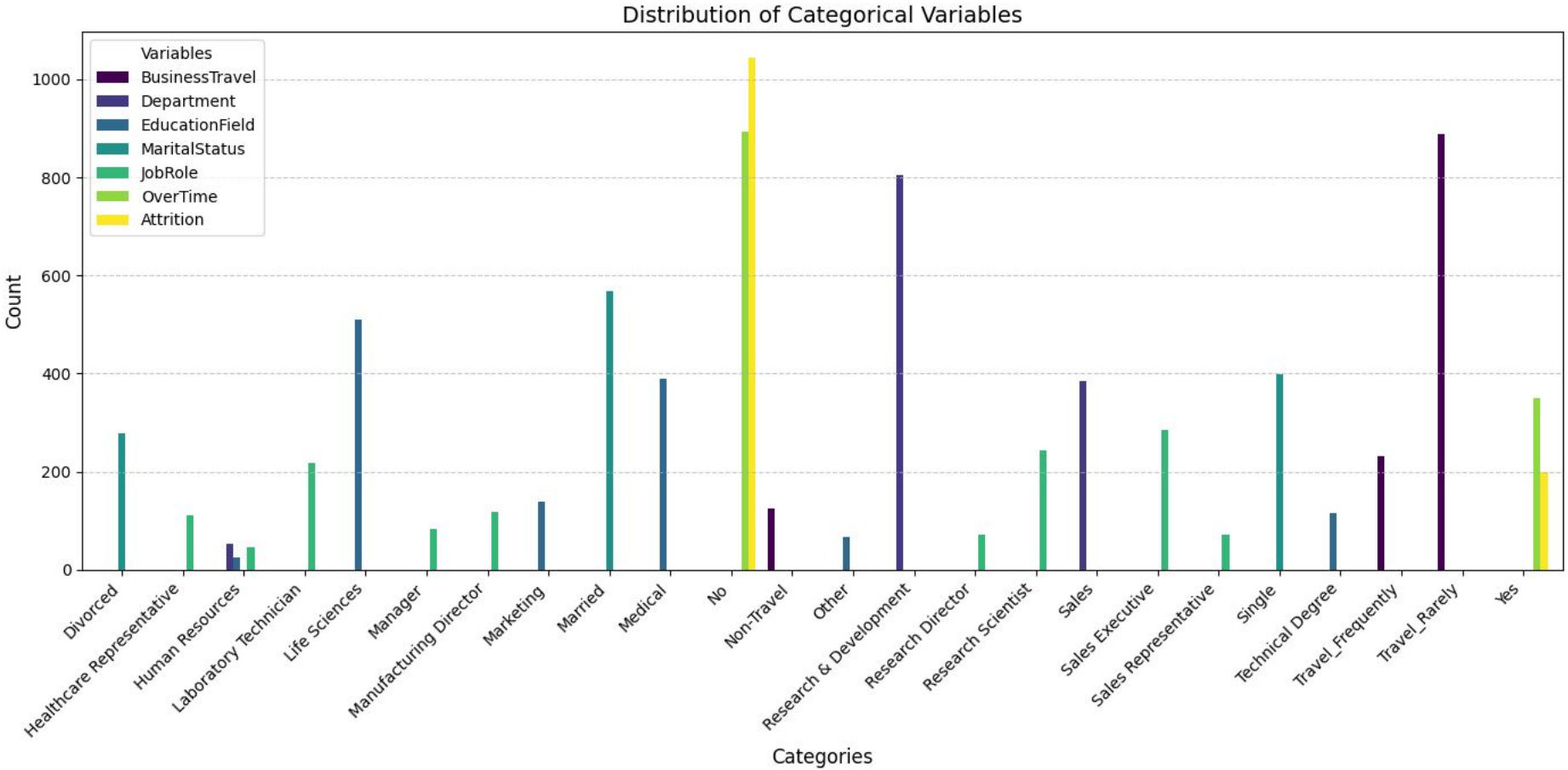

Attrition
Count
0
250
500
750
1000
No
Yes
Attrition

1. **Business Travel**:
    - The majority of employees travel **rarely** for business.
    - A smaller proportion travel frequently or do not travel at all.
2. **Department**:
    - Most employees work in **Research & Development**, followed by **Sales**.
    - The **Human Resources** department has the fewest employees.
3. **Education Field**:
    - A large proportion of employees have an educational background in **Life Sciences** and **Medical fields**.
    - Fewer employees come from **Marketing, Technical, or Human Resources** education fields.
4. **Marital Status**:
    - The highest number of employees are **Married**, followed by **Single** employees.
    - The least common marital status is **Divorced**.

5. **Job Role**:
    - **Sales Executive** and **Research Scientist** roles are the most common.
    - Other roles, such as **Healthcare Representative** and **Human Resources**, have fewer employees.
6. **Overtime**:
    - The majority of employees **do not** work overtime.
    - A smaller proportion **does** work overtime.
7. **Attrition (Employee Turnover)**:
    - Most employees **stay** with the company (Attrition = "No").
    - Only a small proportion of employees **leave** (Attrition = "Yes").

Possible Business Insights:

- Since **most employees do not work overtime**, work-life balance may be a priority.
- The high concentration in **Research & Development** suggests the company is **R&D-focused**.
- Attrition is relatively low, which may indicate **good employee retention strategies**.
- Understanding why certain job roles have **higher attrition** can help improve retention strategies.


Distribution of Categorical Variables
Variables
BusinessTravel
Department
EducationField
MaritalStatus
JobRole
OverTime
Attrition
Count
0
200
400
600
800
1000
Divorced
Healthcare Representative
Human Resources
Laboratory Technician
Life Sciences
Manager
Manufacturing Director
Marketing
Married
Medical
No
Non-Travel
Other
Research & Development
Research Director
Research Scientist
Sales
Sales Executive
Sales Representative
Single
Technical Degree
Travel_Frequently
Travel_Rarely
Yes
Categories

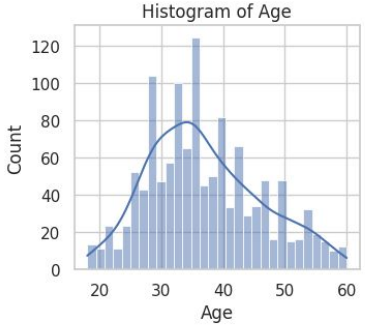
Comparisons & Observations from the Chart:

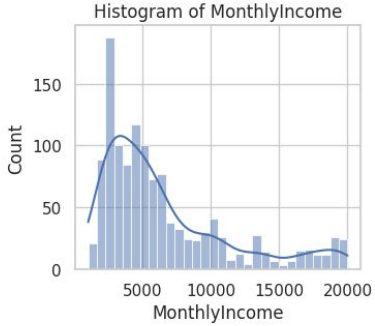
1. **Business Travel & Attrition:**
    - Employees who travel **rarely** have **lower attrition**.
    - Employees with **no travel** have **high attrition**, suggesting that lack of travel might be linked to dissatisfaction or career stagnation.
    - Employees who **travel frequently** show **moderate attrition**, meaning frequent travel might contribute to burnout but isn't the main factor.
2. **Overtime & Attrition:**
    - Employees who **do not work overtime** are the majority.
    - Employees who **work overtime** show **higher attrition**, indicating potential burnout or dissatisfaction.

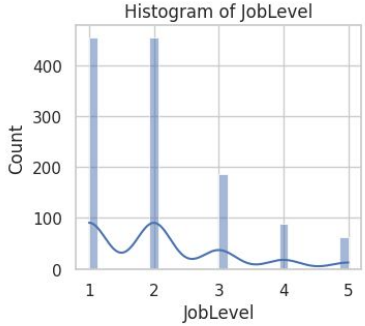
3. **Department & Attrition:**
    - The **Research & Development** department has the most employees, but its attrition levels appear moderate.
    - **Sales employees** show some attrition, possibly due to performance-based pressures.
    - **Human Resources** has fewer employees, making attrition trends harder to analyze.
4. **Education Field & Attrition:**
    - Employees from **Life Sciences and Medical backgrounds** are the largest group.
    - Attrition seems to be **spread across all education fields** without a clear dominant category.
5. **Marital Status & Attrition:**
    - **Single employees** have **higher attrition** compared to married or divorced employees.
    - Married employees seem **more stable**, possibly due to financial or family commitments.

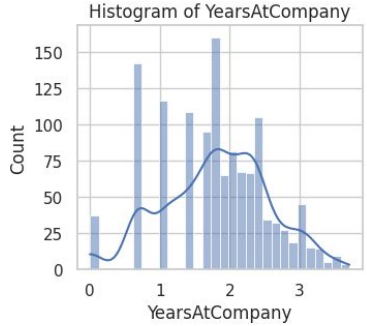
- **Numerical Variables:**
  - `Age`
  - `MonthlyIncome`
  - `JobLevel`
  - `YearsAtCompany`
  - `TotalWorkingYears`
  - `YearsWithCurrManager`
  - `YearsInCurrentRole`
  - `PerformanceRating`
  - `JobSatisfaction`
  - `WorkLifeBalance`

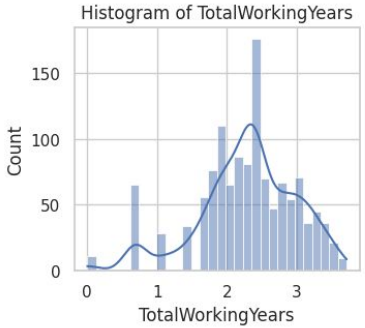
- **Histograms, Boxplots** → for numerical variables

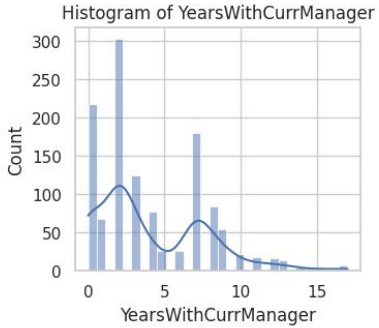

Histogram of Age
Count
Age

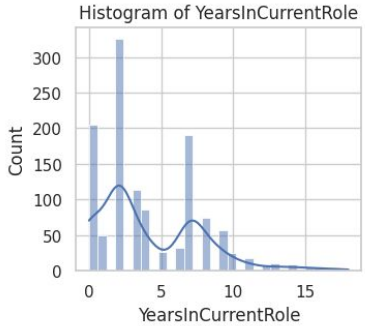

Histogram of MonthlyIncome
Count
MonthlyIncome

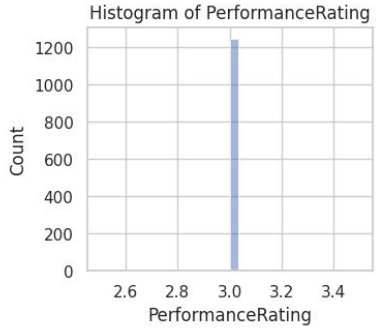

Histogram of JobLevel
Count
JobLevel

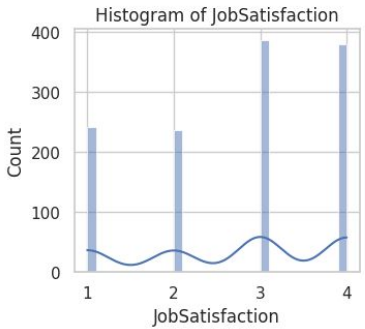

Histogram of YearsAtCompany
Count
YearsAtCompany

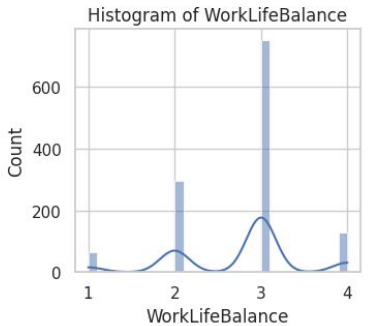

Histogram of TotalWorkingYears
Count
TotalWorkingYears


Histogram of YearsWithCurrManager
Count
YearsWithCurrManager


Histogram of YearsInCurrentRole
Count
YearsInCurrentRole


Histogram of PerformanceRating
Count
PerformanceRating


Histogram of JobSatisfaction
Count
JobSatisfaction


Histogram of WorkLifeBalance
Count
WorkLifeBalance

Insights from the Histograms:

1. **Age Distribution:**
   - Right-skewed distribution with most employees aged between 25-40.
   - Few employees above 50.
2. **Monthly Income:**
   - Highly right-skewed, with most employees earning below 10,000.
   - A few high earners create a long tail.
3. **Job Level:**
   - Most employees are at **Job Level 1 or 2**.
   - Fewer employees at higher job levels (3-5).

4. **Years at Company:**
    - Right-skewed, indicating that most employees have **less than 10 years** at the company.
    - A few long-tenured employees.
5. **Total Working Years:**
    - Majority have **0-10 years** of experience.
    - Very few employees with over 20 years of experience.
6. **Years with Current Manager:**
    - Right-skewed, indicating that most employees have **less than 5 years** with their current manager.

7. **Years in Current Role:**
    - Similar pattern as "Years with Current Manager," with most employees having **less than 5 years** in their role.
8. **Performance Rating:**
    - Almost all employees have a **rating of 3**, indicating limited variance.
    - Possible issue: Performance ratings may be **biased or not diverse**.
9. **Job Satisfaction:**
    - Employees are spread across all satisfaction levels (1-4).
    - Some peaks at satisfaction levels **1 and 4**.

10. **Work-Life Balance:**

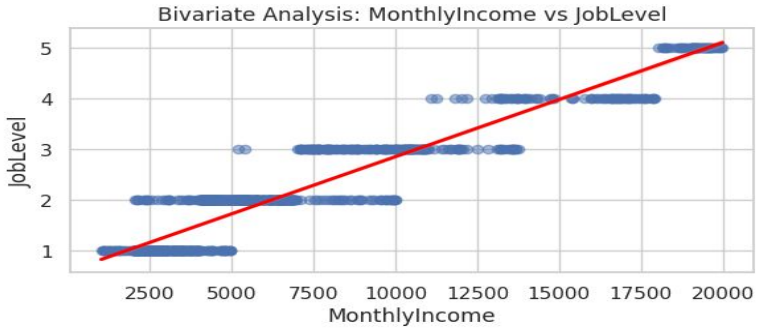
- Distribution is slightly **bimodal**, with peaks at ratings **2 and 3**.
- Few employees rate work-life balance as **1 (poor)**.

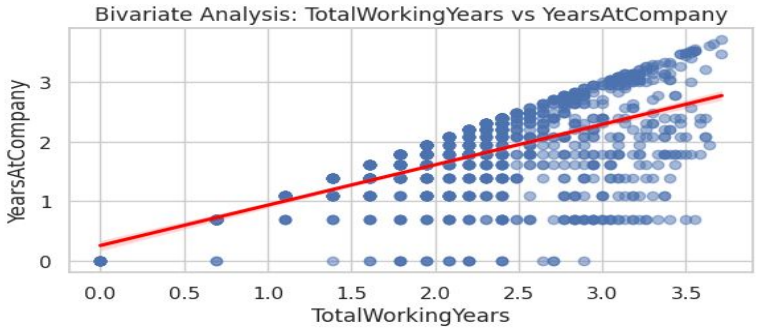
Key Takeaways:

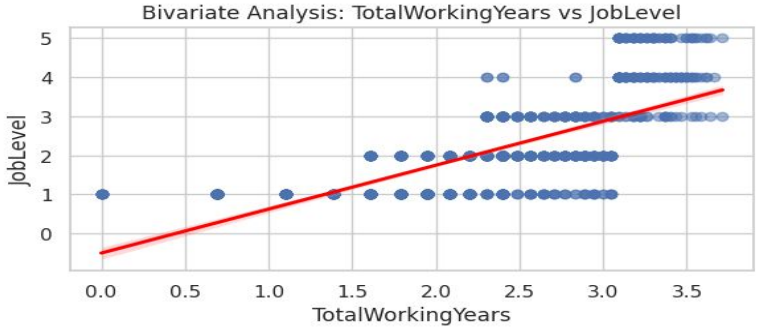
- The company has a **younger workforce** with relatively low tenure.
- There are **income disparities**, with a few high earners.
- **Performance ratings lack diversity**, indicating potential rating bias.
- **Job satisfaction is varied**, meaning some employees are very satisfied while others are unhappy.
- **Work-life balance is not extreme**, but a significant portion finds it moderate.

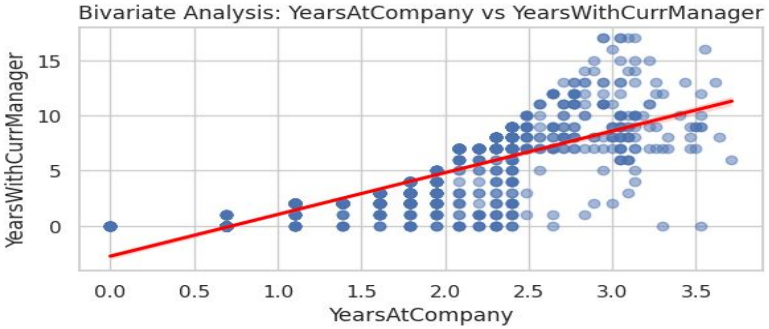
2) Bivariate Analysis

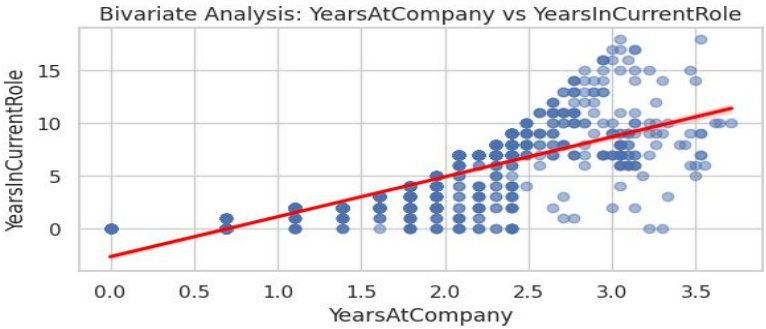
1. **Strong Positive Correlations:**
   - `MonthlyIncome` ↔ `JobLevel` (**0.99**)
   - `TotalWorkingYears` ↔ `YearsAtCompany` (**0.77**)
   - `TotalWorkingYears` ↔ `JobLevel` (**0.66**)
   - `YearsAtCompany` ↔ `YearsWithCurrManager` (**0.46**)
   - `YearsAtCompany` ↔ `YearsInCurrentRole` (**0.44**)


Bivariate Analysis: MonthlyIncome vs JobLevel
JobLevel
MonthlyIncome
1
2
3
4
5
2500
5000
7500
10000
12500
15000
17500
20000


Bivariate Analysis: TotalWorkingYears vs YearsAtCompany
YearsAtCompany
TotalWorkingYears
0
1
2
3
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5


Bivariate Analysis: TotalWorkingYears vs JobLevel
JobLevel
TotalWorkingYears
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5


Bivariate Analysis: YearsAtCompany vs YearsWithCurrManager
YearsWithCurrManager
YearsAtCompany
0
5
10
15
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

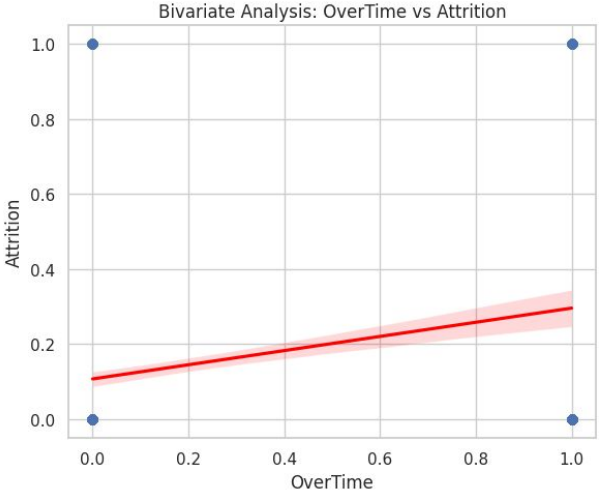

Bivariate Analysis: YearsAtCompany vs YearsInCurrentRole
YearsInCurrentRole
YearsAtCompany
0
5
10
15
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

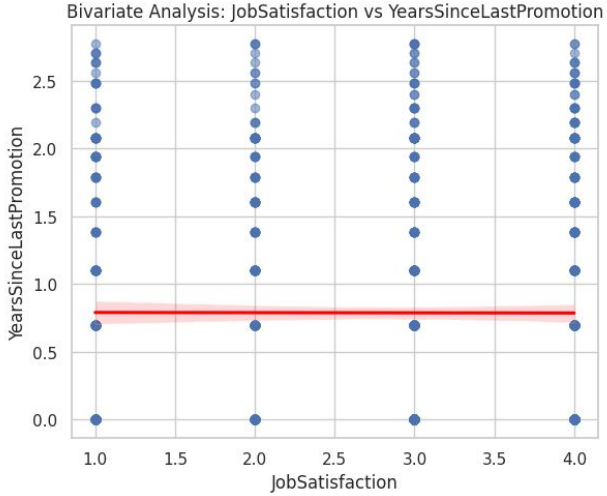
1. **MonthlyIncome vs JobLevel (Correlation: 0.99)**
    - There is a **strong positive correlation** between **MonthlyIncome** and **JobLevel**.
    - Higher job levels tend to have **higher salaries**, which is expected.
    - The **linear trend** confirms that income increases as job level increases.
2. **TotalWorkingYears vs YearsAtCompany (Correlation: 0.77)**
    - A **positive correlation** indicates that employees with more **total experience** tend to stay longer in a company.
    - Some variations suggest that **job changes** may impact years at a company.

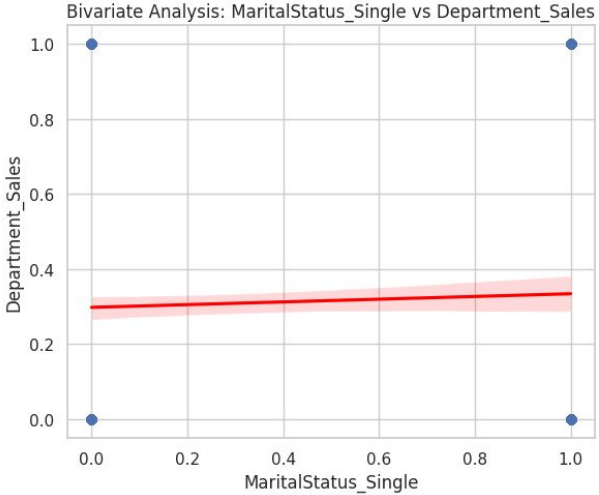
3. **TotalWorkingYears vs JobLevel (Correlation: 0.66)**
    - A **moderate positive correlation** suggests that employees with **more years of experience** tend to be in **higher job levels**.
    - However, job level is not solely determined by experience; other factors like **performance, promotions, and education** may play a role.
4. **YearsAtCompany vs YearsWithCurrManager (Correlation: 0.46)**
    - A **moderate correlation** shows that employees staying longer at a company tend to work with the same manager for a longer period.
    - Some points **deviate from the trend**, indicating **manager changes** over time.
5. **YearsAtCompany vs YearsInCurrentRole (Correlation: 0.44)**
    - A **moderate positive relationship** shows that employees who have been in the company longer also tend to remain in their **current role for longer periods**.
    - Some **outliers suggest** that certain employees change roles faster.

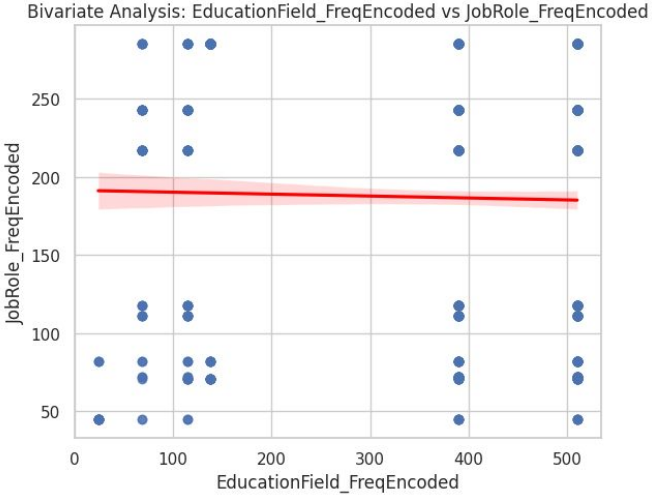
2. **Moderate Correlations:**

   - `OverTime` ↔ `Attrition` (**0.23**)
   - `JobSatisfaction` ↔ `YearsSinceLastPromotion` (**0.31**)
   - `MaritalStatus_Single` ↔ `Department_Sales` (**0.30**)
   - `EducationField_FreqEncoded` ↔ `JobRole_FreqEncoded` (**-0.34**)


Bivariate Analysis: OverTime vs Attrition
Attrition
OverTime
0.0
0.2
0.4
0.6
0.8
1.0


Bivariate Analysis: JobSatisfaction vs YearsSinceLastPromotion
YearsSinceLastPromotion
JobSatisfaction
0.0
0.5
1.0
1.5
2.0
2.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Bivariate Analysis: MaritalStatus_Single vs Department_Sales
Department_Sales
MaritalStatus_Single
0.0
0.2
0.4
0.6
0.8
1.0


Bivariate Analysis: EducationField_FreqEncoded vs JobRole_FreqEncoded
JobRole_FreqEncoded
EducationField_FreqEncoded
50
100
150
200
250
0
100
200
300
400
500

1️⃣ OverTime vs Attrition (Top-Left)

- **Positive correlation (0.23)**: Employees who **work overtime are slightly more likely to leave** (attrition is higher).
- The trend suggests that while **overtime alone doesn't strongly determine attrition**, it **could be a contributing factor** when combined with job stress or dissatisfaction.

2️⃣ JobSatisfaction vs YearsSinceLastPromotion (Top-Right)

- **Weak correlation (0.31)**: **No clear trend** between job satisfaction and years since the last promotion.
- Employees may **still be satisfied despite no promotion**, possibly due to other benefits or job stability.
- However, employees with extremely long gaps since promotion **might start feeling dissatisfied** over time.

3️⃣ MaritalStatus_Single vs Department_Sales (Bottom-Left)

- **Weak positive correlation (0.30)**: Single employees **slightly tend to work in Sales**.
- This could be due to sales jobs often requiring **frequent travel, irregular hours, or higher performance pressure**, which might be easier for single employees to manage.

4️⃣ EducationField_FreqEncoded vs JobRole_FreqEncoded (Bottom-Right)

- **Negative correlation (-0.34)**: Certain **education fields are less likely to align with specific job roles**.
- For example, employees from **technical fields might not prefer HR or administrative roles**, while business graduates might be less likely to work in engineering.
- This pattern suggests that education specialization **significantly influences job role assignments**.

3) Multivariate Analysis

Best Combinations for Multivariate Analysis:

- **Attrition Analysis:**

  `Attrition` **vs.** `OverTime`, `JobSatisfaction`, `WorkLifeBalance`

  (*To see if employees with overtime are more likely to leave.*)

- **Income & Experience Relationship:**

  `MonthlyIncome` **vs.** `JobLevel`, `YearsAtCompany`, `TotalWorkingYears`

  (*To check how experience affects salary.*)

- **Job Role & Education Impact:**

  `EducationField_FreqEncoded` **vs.** `JobRole_FreqEncoded`, `JobSatisfaction`

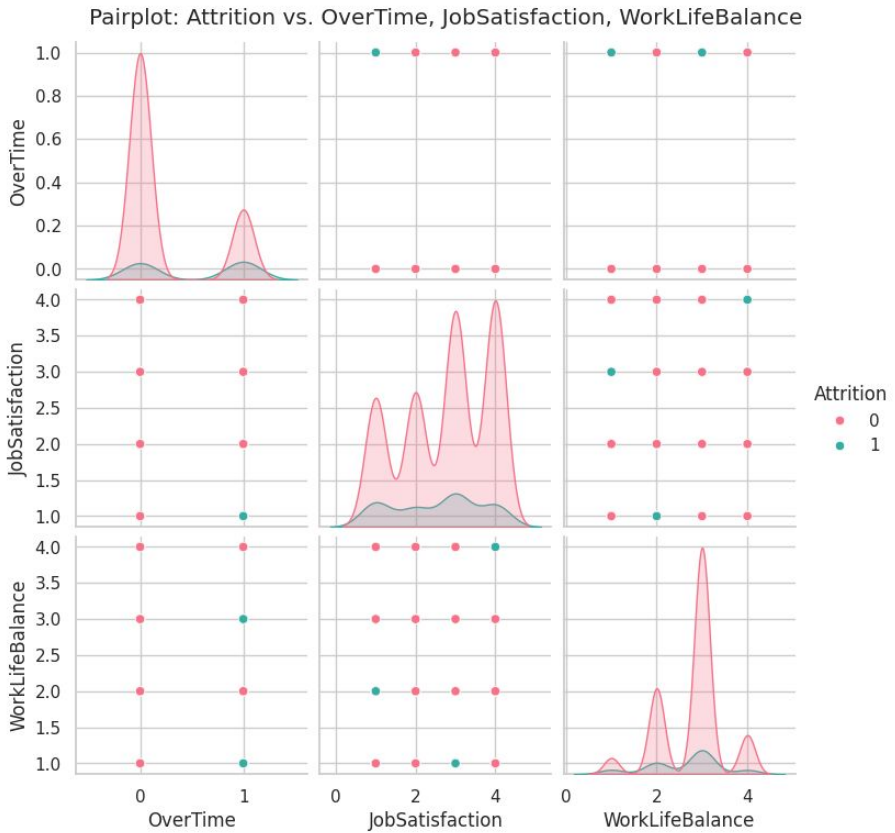
  (*To see if different education fields lead to different job roles.*)

- **Marital Status & Department:**

  `MaritalStatus_Single` **vs.** `Department_Sales`, `Attrition`

  (*To check if single employees are more in sales.*)

i) Attrition Analysis:

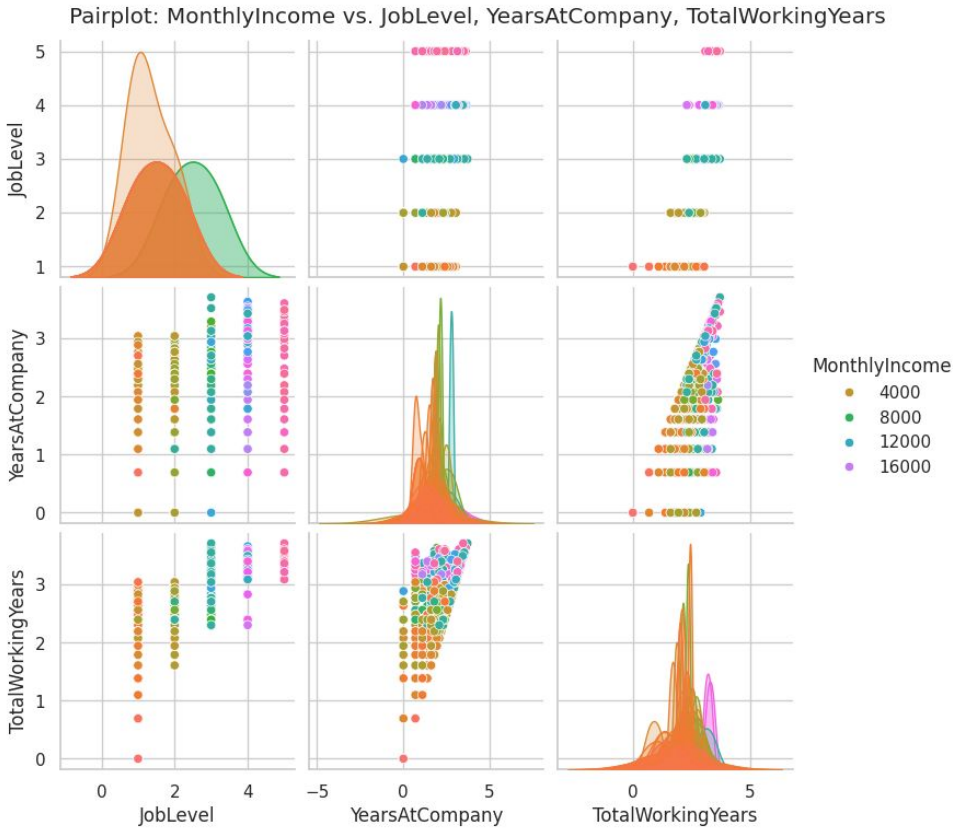


✅ **OverTime seems to have a stronger impact on Attrition** – employees who work overtime are more likely to leave.

✅ **JobSatisfaction does not directly determine attrition** – employees leave even with high satisfaction.

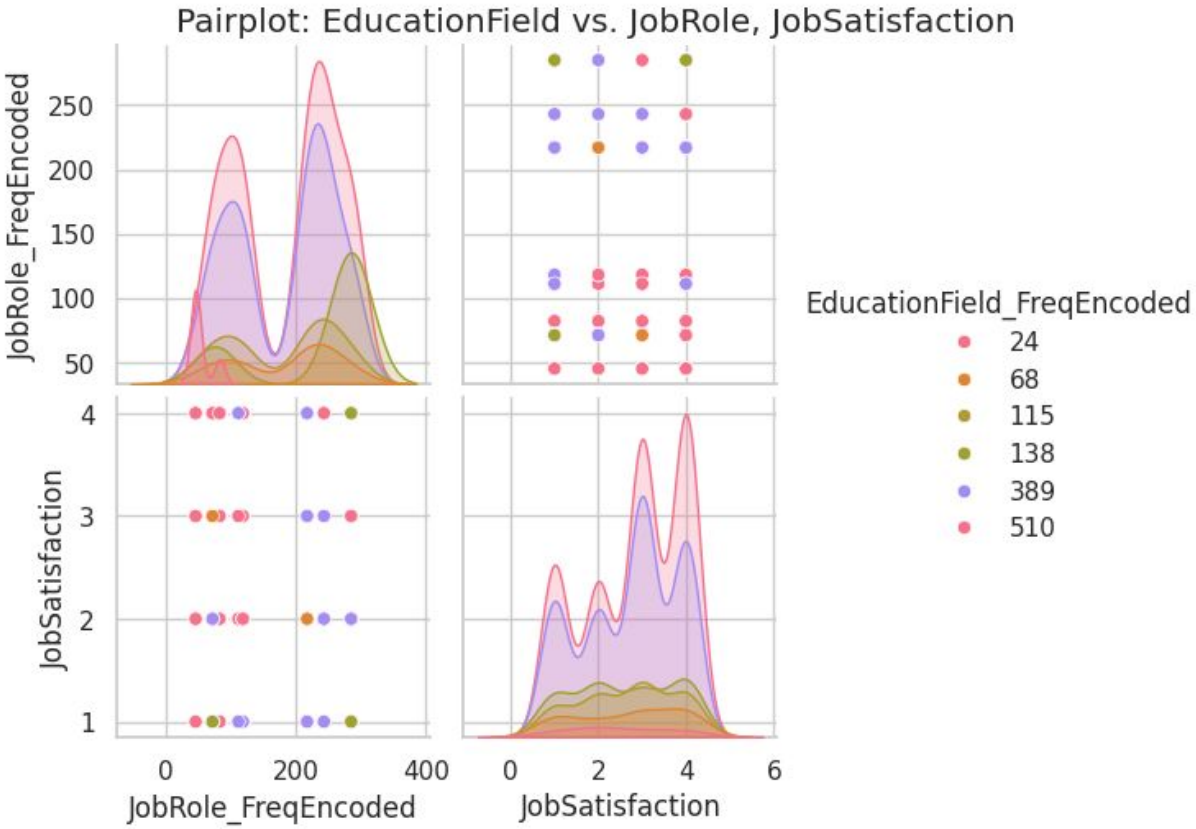
✅ **WorkLifeBalance is an important factor** – poor balance may contribute to higher attrition.

ii) Income & Experience Relationship:



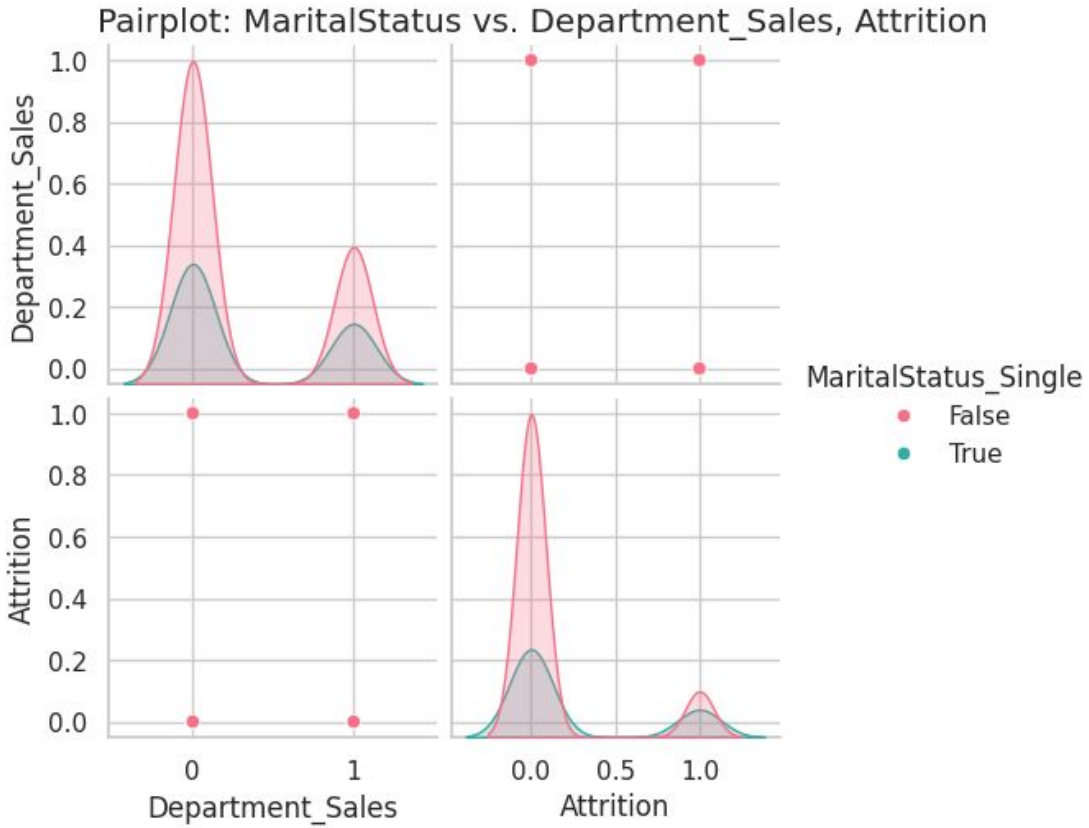
- **Retention Strategy:** Since tenure alone does not guarantee higher pay, HR should ensure employees see clear paths to **salary growth through promotions or skill enhancements**.
- **Salary Adjustments:** If some employees have been in the company for long but earn less, a **review of salary increments based on years of experience** may help with **retention**.
- **Training & Upskilling:** Since **TotalWorkingYears** has a stronger impact on income than just company tenure, employees should be encouraged to **develop expertise and move up the career ladder rather than just staying longer**.

3) Job Role & Education Impact:



- **Job Role Optimization:** If certain education fields are more dominant in some job roles, companies can use this insight to **streamline hiring** and career path planning.
- **Job Satisfaction Improvement:** Since some roles have **diverse satisfaction levels**, HR can analyze those roles further to **identify pain points** affecting employee happiness.
- **Education Field Alignment:** If specific education fields result in **higher satisfaction in particular roles**, companies can refine **recruitment strategies** to match education backgrounds with the right roles.

4) Marital Status & Department



Feature Engineering

1. **Tenure Category** (`Tenure_Category`): Groups employees based on `YearsAtCompany`
    - **New**: < 2 years
    - **Intermediate**: 2 - 5 years
    - **Experienced**: 5 - 10 years
    - **Veteran**: > 10 years
2. **Performance Score** (`Performance_Score`): Weighted combination of:
    - `PerformanceRating` (50%)
    - `JobSatisfaction` (30%)
    - `WorkLifeBalance` (20%)

3. **Engagement Score** (`Engagement_Score`): Weighted combination of:
    - `OverTime` (40%)
    - `WorkLifeBalance` (20%)
    - `JobInvolvement` (20%)
    - `TrainingTimesLastYear` (20%)

```python
# Function to categorize tenure
def tenure_category(years):
    if years < 2:
        return 'New'
    elif years < 5:
        return 'Intermediate'
    elif years < 10:
        return 'Experienced'
    else:
        return 'Veteran'

# Apply tenure category if column exists
if 'YearsAtCompany' in df.columns:
    df['Tenure_Category'] = df['YearsAtCompany'].apply(tenure_category)

# Calculate Performance Score (Weighted) if required columns exist
if {'PerformanceRating', 'JobSatisfaction', 'WorkLifeBalance'}.issubset(df.columns):
    df['Performance_Score'] = (df['PerformanceRating'] * 0.5) + \
                              (df['JobSatisfaction'] * 0.3) + \
                              (df['WorkLifeBalance'] * 0.2)

# Calculate Engagement Score (Weighted) if required columns exist
if {'OverTime', 'WorkLifeBalance', 'JobInvolvement', 'TrainingTimesLastYear'}.issubset(df.columns):
    df['Engagement_Score'] = (df['OverTime'] * 0.4) + \
                             (df['WorkLifeBalance'] * 0.2) + \
                             (df['JobInvolvement'] * 0.2) + \
                             (df['TrainingTimesLastYear'] * 0.2)

```

Feature Selection

For 1st Prediction [Attrition Prediction]

Method	Type of Data	Strength	Weakness
Correlation Heatmap	Continuous vs. Continuous	Easy to interpret	Doesn't work for categorical data
Chi-Square Test	Categorical vs. Categorical	Good for categorical data	Doesn't work for continuous data
ANOVA (F-test)	Continuous vs. Categorical	Finds significant numerical features	Assumes normality in data
RFE (Recursive Feature Elimination)	All types	Works with any ML model	Computationally expensive

MI (Mutual Information) is Type of Filter Method : Combine of Chi2 & ANOVA
Both Categorical and Numerical

🚀 Best Feature Selection Strategy

1. **If features & target are categorical** → Use **Chi-Square Test**.
2. **If features are continuous & target is categorical** → Use **ANOVA F-test**.
3. **If all features are continuous** → Use **Correlation Heatmap**.
4. **If you want model-based selection** → Use **RFE (Recursive Feature Elimination)** with a model like Random Forest.

```python
# 1️⃣ Chi-Square Test for Categorical Features
chi2_scores, chi2_pvalues = chi2(X_categorical, y)
chi2_results = pd.DataFrame({"Feature": categorical_features, "Chi2 Score": chi2_scores, "p-value": chi2_pv
chi2_results = chi2_results.sort_values(by="Chi2 Score", ascending=False)

# 2️⃣ ANOVA (F-Test) for Numerical Features
f_scores, f_pvalues = f_classif(X_numerical, y)
anova_results = pd.DataFrame({"Feature": numerical_features, "F Score": f_scores, "p-value": f_pvalues})
anova_results = anova_results.sort_values(by="F Score", ascending=False)

# 3️⃣ Mutual Information (MI) for Both
mi_scores = mutual_info_classif(X, y)
mi_results = pd.DataFrame({"Feature": X.columns, "MI Score": mi_scores})
mi_results = mi_results.sort_values(by="MI Score", ascending=False)

# 4️⃣ Recursive Feature Elimination (RFE) using Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rfe = RFE(rf_model, n_features_to_select=10)  # Selecting top 10 features
rfe.fit(X, y)
rfe_results = pd.DataFrame({"Feature": X.columns, "RFE Ranking": rfe.ranking_})
rfe_results = rfe_results.sort_values(by="RFE Ranking", ascending=True)

# Display top features from each method
chi2_results.head(10), anova_results.head(10), mi_results.head(10), rfe_results.head(10)

```

Analyzing Which Feature Selection is Best

```
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    # Store result
    accuracy_results[method] = accuracy

# Display accuracy results
accuracy_results

```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_sag.py:348: ConvergenceWarning: The max_iter was
  warnings.warn(
{'Chi-Square': 0.8313253012048193,
 'ANOVA': 0.8393574297188755,
 'Mutual Information': 0.8313253012048193,
 'RFE': 0.8313253012048193}
```

Method	Best For	Handles Categorical?	Handles Numerical?	Captures Non-Linear?
Chi-Square (`chi2`)	Categorical features only	✅ Yes	❌ No	❌ No
ANOVA F-Test (`f_classif`)	Continuous numerical features	❌ No	✅ Yes	❌ No
Mutual Information (`mutual_info_classif`)	Both categorical & numerical	✅ Yes	✅ Yes	✅ Yes
RFE (Recursive Feature Elimination)	Model-based feature selection	✅ Yes	✅ Yes	✅ Yes

Move to Mutual Information or RFE is best Compare to Correlation Check (Heatmap)

Method	Description	Example Techniques
Filter	Uses statistical measures to select features **before model training**.	Correlation heatmaps, Chi-Square test, ANOVA, Mutual Information
Wrapper	Uses a machine learning model to **iteratively select the best subset of features**.	Recursive Feature Elimination (RFE), Forward/Backward Selection

```
Top 14 Features Most Correlated with Attrition:
OverTime                    0.231314
TotalWorkingYears           0.221856
YearsAtCompany              0.184940
JobLevel                    0.152849
Age                         0.152598
YearsWithCurrManager        0.152104
YearsInCurrentRole          0.151308
MaritalStatus_Single        0.145502
MonthlyIncome               0.144153
BusinessTravel              0.136136
JobInvolvement              0.129409
Tenure_Category             0.123146
StockOptionLevel            0.121294
EnvironmentSatisfaction     0.108631
Name: Attrition, dtype: float64
```

```
1 df['Attrition'].value_counts()
```

Attrition	count
0	1044
1	200

dtype: int64

Attrition - 0 is mean by no wearing or stress

Attrition - 1 is mean by is high and afftected by wearing is low

1. **If the goal is just to predict Attrition (Yes/No) with high accuracy**

   ✅ **Focus on Accuracy** (Ensure it's >80%)

   ❌ No need to focus on Precision, Recall, or F1-score

2. **If the goal is to filter or confirm more "Attrition = Yes" cases (Minimize False Positives)**

   ✅ **Focus on Precision** (Actual Positives & Predicted Positives should match)

   ❌ No need to focus on Recall

3. **If the goal is to find as many "Attrition = Yes" cases as possible (Minimize False Negatives)**

   ✅ **Focus on Recall** (Catch all Attrition cases)

   ❌ No need to focus on Precision

4. **If the goal is to **predict Attrition (Yes/No) while balancing Precision & Recall**

   ✅ **Focus on F1-score** (Because it considers both Precision & Recall)

Feature Selection Based on RFE Method

```
[ ]  1 features = ['Age', 'DailyRate', 'DistanceFromHome', 'EmployeeNumber',
     2             'MonthlyIncome', 'MonthlyRate', 'HourlyRate', 'YearsAtCompany','TotalWorkingYears','TotalWorkingYears','Performance_Score']
```

```
[ ]  1 ##########################################################################################################
```

1) Logistic Regression - rfe based

```
Model Performance Metrics:
Train Accuracy: 0.5889
Test Accuracy: 0.5823
Precision: 0.2288
Recall: 0.6750
Train Cross-Entropy Loss: 0.6558
Test Cross-Entropy Loss: 0.6601

Confusion Matrix:
[[118  91]
 [ 13  27]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.56      0.69       209
           1       0.23      0.68      0.34        40

    accuracy                           0.58       249
   macro avg       0.56      0.62      0.52       249
weighted avg       0.79      0.58      0.64       249
```

Cross Validation for Logistic Regression - rfe Based

```
  n_iter_i = _check_optimize_result(
Stratified K-Fold Cross-Validation Results:
Average Accuracy: 0.5876
Average Precision: 0.2110
Average Recall: 0.5700
Average Cross-Entropy Loss: 0.6525
```

RFE (Wrapper Method) Based Feature Selection Give Low Accuracy

```
[ ]  1 feature = df[['OverTime','TotalWorkingYears','YearsAtCompany','JobLevel','Age','YearsWithCurrManager','YearsInCurrentRole','MaritalStatus_Marr
     2 target = df['Attrition']
```

2) Logistic Regression - Heatmap Correlation Based

```
Model Performance Metrics:
Train Accuracy: 0.7075
Test Accuracy: 0.7149
Precision: 0.3258
Recall: 0.7250
Train Cross-Entropy Loss: 0.5728
Test Cross-Entropy Loss: 0.5850

Confusion Matrix:
[[149  60]
 [ 11  29]]

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.71      0.81       209
           1       0.33      0.72      0.45        40

    accuracy                           0.71       249
   macro avg       0.63      0.72      0.63       249
weighted avg       0.83      0.71      0.75       249
```

Correlation Based Feature Selection Compare to Better than RFE Based Feature Selection

3) Logistic Regression - mi (Mutual Information both chi2 and ANOVA)

```
[ ]  1 feature_mi = df[['MonthlyIncome','TotalWorkingYears','OverTime','YearsAtCompany','Age','JobLevel','Department_Sales','Tenure_Category','JobRol
     2 target_mi = df['Attrition']
```

```
[ ]  1 feature_mi.columns
```

```
Index(['MonthlyIncome', 'TotalWorkingYears', 'OverTime', 'YearsAtCompany',
       'Age', 'JobLevel', 'Department_Sales', 'Tenure_Category',
       'JobRole_FreqEncoded', 'StockOptionLevel', 'EducationField_FreqEncoded',
       'JobInvolvement', 'YearsWithCurrManager', 'DailyRate'],
      dtype='object')
```

```
Model Performance Metrics:
Train Accuracy: 0.7296
Test Accuracy: 0.7108
Precision: 0.3049
Recall: 0.6250
Train Cross-Entropy Loss: 0.5726
Test Cross-Entropy Loss: 0.5775

Confusion Matrix:
[[152  57]
 [ 15  25]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.73      0.81       209
           1       0.30      0.62      0.41        40

    accuracy                           0.71       249
   macro avg       0.61      0.68      0.61       249
weighted avg       0.81      0.71      0.74       249
```

- **Balanced Accuracy & Loss: Mutual Information** gives the highest accuracy and lowest loss.
- **Best Recall: Heatmap-based selection** captures more true positives (high recall).
- **Worst Performance: RFE** has the lowest accuracy and highest loss, making it the least effective.

Final Recommendation

- If **overall accuracy & stability** are important → **Mutual Information**
- If **capturing more attrition cases (higher recall)** is critical → **Heatmap-based selection**

Mutual Information Based Feature Selection is Best

1) Random Forest for Mutual Information (combaine of chi2 & ANOVA) - Filter Method

```
Model Performance Metrics:
Train Accuracy: 1.0000
Test Accuracy: 0.8394
Precision: 0.5000
Recall: 0.1750
Train Cross-Entropy Loss: 0.0933
Test Cross-Entropy Loss: 0.5380

Confusion Matrix:
[[202   7]
 [ 33   7]]

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.97      0.91       209
           1       0.50      0.17      0.26        40

    accuracy                           0.84       249
   macro avg       0.68      0.57      0.58       249
weighted avg       0.80      0.84      0.81       249
```

After Using Random Forest Test Accuracy was Increased But Overfitting Happen

Cross Validation for Random Forest

```
Stratified K-Fold Cross-Validation Results:
Average Train Accuracy: 1.0000
Average Test Accuracy: 0.8416
Average Precision: 0.5259
Average Recall: 0.1450
Average Cross-Entropy Loss: 0.4704
```

After Using Cross Validation Stratified K-fold - 2 splits but still Overfitt not prevent

class weight concentrated :- 0: 1, 1: 3

Hyperparameter tuning to find best params for our model

```
[ ]
```

```python
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier

# Define parameter grid
param_grid = {
    'n_estimators': [50, 100, 200, 300],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 3, 5],
    'max_features': ['sqrt', 'log2'],
    'class_weight': [{0: 1, 1: 2}, {0: 1, 1: 3}, 'balanced']
}

# Initialize model
rf = RandomForestClassifier(random_state=42)
```

```
Model Performance Metrics:
Train Accuracy: 0.8643
Test Accuracy: 0.8193
Precision: 0.4510
Recall: 0.5750
Train Cross-Entropy Loss: 0.4400
Test Cross-Entropy Loss: 0.5008

Confusion Matrix:
[[181  28]
 [ 17  23]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.87      0.89       209
           1       0.45      0.57      0.51        40

    accuracy                           0.82       249
   macro avg       0.68      0.72      0.70       249
weighted avg       0.84      0.82      0.83       249
```

class 0 has high precision, recall and f1 score reason my dataset has imbalanced that why showing huge difference of class 0 and class 1

1. **Macro Average**
    - Takes the average of precision, recall, and F1-score across all classes **without considering class imbalance**.
    - Each class contributes equally, even if one class is much smaller than the other.
    - Good for analyzing **model performance across all classes equally**.
2. **Weighted Average**
    - Takes the **class distribution into account**, meaning larger classes contribute more to the final score.
    - If your dataset is imbalanced, **weighted avg gives a more realistic performance measure**.
    - Useful when you **care about overall accuracy** rather than focusing on minority classes.

Check AUC and ROC Curve

```
Best Threshold (Youden's J): 0.4422
Best Threshold (F1-score - ROC curve): 0.5195
Best Threshold (F1-score - Precision-Recall curve): 0.5195
```

Which Threshold to Use?

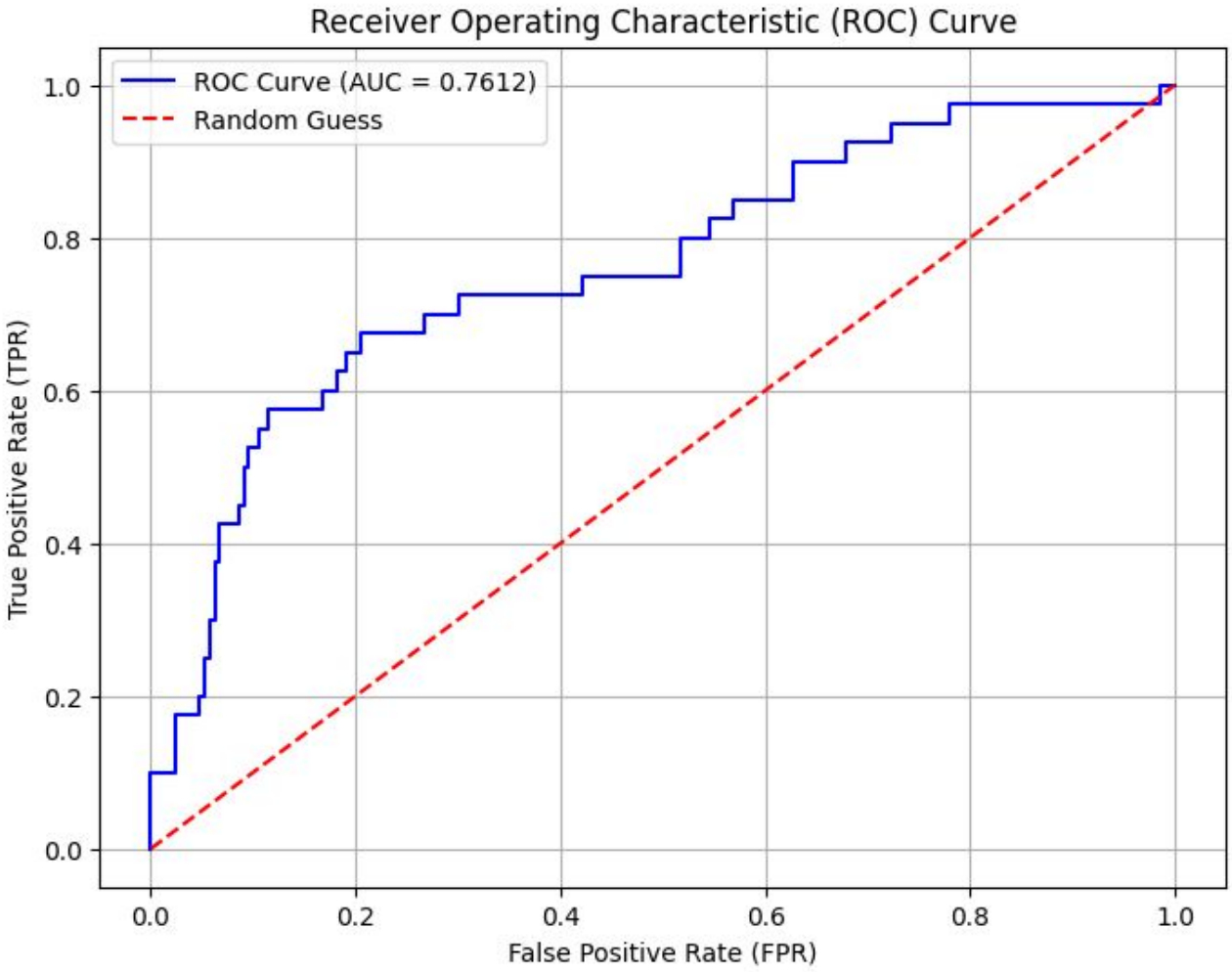
- **For balanced datasets:** Use **Youden's J** (Default ROC-based threshold).
- **For imbalanced datasets:** Use **F1-score from Precision-Recall curve**.

My Dataset has Imbalanced so Focus on F1-Score - precision-Recall Curve - 0.5255

Final Decision:

✅ Go with the hyperparameter-tuned Random Forest (AUC = 0.76, Accuracy = 81.93%)

🚫 Skip Stratified K-Fold if it reduces performance


Receiver Operating Characteristic (ROC) Curve
ROC Curve (AUC = 0.7612)
Random Guess
True Positive Rate (TPR)
False Positive Rate (FPR)
0.0
0.2
0.4
0.6
0.8
1.0

```
Using Threshold = 0.5195
Accuracy: 0.8353
Precision: 0.4894
Recall: 0.5750
F1 Score: 0.5287
```

- **General Interpretation of AUC-ROC Scores**:
  - **0.5 → 0.6** → Poor (close to random guessing)
  - **0.6 → 0.7** → Fair
  - **0.7 → 0.8** → Good ✅ *(Your Model)*
  - **0.8 → 0.9** → Very Good
  - **0.9+** → Excellent (Rare for imbalanced datasets without overfitting)

AUC-ROC is useful, but not always a must. Since your dataset is imbalanced, you should focus more on **Precision-Recall AUC** and **F1-score** rather than AUC-ROC. However, **showing the ROC curve is still helpful to confirm how well your model distinguishes classes.**

Save this [Hyerparameter Tuned Random Forest]model

```python
import joblib

# Save the trained Random Forest model
joblib.dump(model, "random_forest_best_1st.pkl")

# Save the feature list
joblib.dump(feature_mi, "random_forest_features_1st.pkl")

print("Model and features saved successfully!")
```

```
Model and features saved successfully!
```

Future Prediction - Attrition

1st Future Prediction

Loading the Saved Model and Features

```
[ ]
```

```python
import joblib

# Load the trained Random Forest model
loaded_model = joblib.load("random_forest_best_1st.pkl")

# Load the feature list
loaded_features = joblib.load("random_forest_features_1st.pkl")

print("Model and features loaded successfully!")

```

```
Model and features loaded successfully!
```

```
[ ]    1 loaded_features
```

```
['MonthlyIncome',
 'TotalWorkingYears',
 'OverTime',
 'YearsAtCompany',
 'Age',
 'JobLevel',
 'Department_Research_and_Development',
 'Department_Sales',
 'Tenure_Category',
 'JobRole_FreqEncoded',
 'StockOptionLevel',
 'EducationField_FreqEncoded',
 'JobInvolvement']
```

Future Prediction

```
[ ]
```

```python
import numpy as np

# Corrected input data with encoded categorical values
input_data = np.array([[5993, 2.197225, 1, 1.945910, 41, 2, 0, 1, 0, 285, 0, 510, 3]])  # Encoded values

# Reshape if predicting a single instance
input_data = input_data.reshape(1, -1)

# Make the prediction
prediction = loaded_model.predict(input_data)

print("Predicted Output:", prediction)
```

```
Predicted Output: [1]
```

```
[ ]
```

```python
import numpy as np

# Corrected input data with encoded categorical values
input_data = np.array([[3468, 1.945910, 0, 1.098612, 27, 1, 1, 0, 0, 217, 1, 389, 3]])  # Encoded values

# Reshape if predicting a single instance
input_data = input_data.reshape(1, -1)

# Make the prediction
prediction = loaded_model.predict(input_data)

print("Predicted Output:", prediction)
```

```
Predicted Output: [0]
```

Encoded and Scaled Value will decoded and unscaled for user handle

```
[ ]
# Department ->  onehotencoding - >Department_Research_and_Development, Department_Sales
# Tenure_Category -> Feature Engineering -> based on yearatcompany
# JobRole_FreqEncoded -> Frequency Encoding -> JobRole
# EducationField_FreqEncoded -> Frequency Encoding -> EducationField

```

```
[ ]
# # Select features and target
# feature_raw = ['MonthlyIncome','TotalWorkingYears','OverTime','YearsAtCompany',
#                'Age','JobLevel','Department_Research_and_Development','Department_Sales',
#                'Tenure_Category','JobRole_FreqEncoded','StockOptionLevel','EducationField_FreqEncoded',
#                'JobInvolvement']
```

```
[ ]
# MonthlyIncome -> same no changes //
# TotalWorkingYears -> Log Transformation
# OverTime -> ordinal encoding
# YearsAtCompany -> Log Transformation
# Age -> Same No Changes //
# JobLevel -> Same No Changes //
# Department -> One Hot Encoding
# Tenure_Category -> Feature Engineering -> based on yearatcompany
# JobRole_FreqEncoded -> Frequency Encoding
# StockOptionLevel -> Same No Changes //
# EducationField_FreqEncoded   -> Frequency Encoding
# JobInvolvement -> Same No Changes //
```

1) Log Transformation (TotalWorkingYears) Model Save and Load

```python
from sklearn.preprocessing import FunctionTransformer
import joblib

# Create a transformer for log transformation
log_transformer = FunctionTransformer(np.log1p, validate=True)

# Fit and transform data
X_raw['TotalWorkingYears_log'] = log_transformer.fit_transform(X_raw[['TotalWorkingYears']])

# Save the transformer
joblib.dump(log_transformer, "log_transformer_TotalWorkingYears.pkl")
print("Log transformer saved!")
```

Step 1: Save the Fitted LabelEncoder

```
[ ]
```

```python
from sklearn.preprocessing import LabelEncoder
import joblib

# Define the columns to encode
cols_to_encode = ['OverTime']

# Initialize and fit LabelEncoder
label_encoders = {}  # Dictionary to store label encoders

for col in cols_to_encode:
    le = LabelEncoder()
    X_raw[col] = le.fit_transform(X_raw[col])  # Transform data
    label_encoders[col] = le  # Store the fitted encoder

# Save the label encoders dictionary
joblib.dump(label_encoders, "label_encoders.pkl")
print("Label encoders saved successfully!")
```

```
Label encoders saved successfully!
<ipython-input-34-f48fdfd0659c>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

4) Step 1: Save the OHE Column Structure

```
[ ]
```

```python
import pandas as pd
import joblib

# Apply One-Hot Encoding (OHE) to 'Department'
X_raw = pd.get_dummies(X_raw, columns=['Department'], drop_first=True)

# Save the OHE column names (important for consistency)
ohe_columns = X_raw.columns.tolist()
joblib.dump(ohe_columns, "ohe_columns.pkl")

print("OHE column structure saved successfully!")
```

```
OHE column structure saved successfully!
```

5) Step 1: Save the Frequency Encoding Mapping

```
[ ]
```

```python
import pandas as pd
import joblib

# Create frequency encoding mapping
freq_encoding_map = X_raw2['JobRole'].value_counts().to_dict()

# Apply frequency encoding
X_raw2['JobRole_FreqEncoded'] = X_raw2['JobRole'].map(freq_encoding_map)

# Save the frequency encoding mapping
joblib.dump(freq_encoding_map, "jobrole_freq_encoding.pkl")

print("Frequency encoding mapping saved successfully!")

```

```
Frequency encoding mapping saved successfully!
```

Final Future Prediction

```python
# Load saved transformations & model
log_transformer_total = joblib.load("log_transformer_TotalWorkingYears.pkl")
log_transformer_year = joblib.load("log_transformer_YearsAtCompany.pkl")
label_encoders = joblib.load("label_encoders.pkl")
ohe_columns = joblib.load("ohe_columns.pkl")
freq_encoding_map_job = joblib.load("jobrole_freq_encoding.pkl")
freq_encoding_map_edu = joblib.load("EducationField_freq_encoding.pkl")

# Load trained Random Forest model
loaded_model = joblib.load("random_forest_best_1st.pkl")
```

```python
# ✅ Example new input data
new_data = pd.DataFrame({
    'MonthlyIncome': [5993],
    'TotalWorkingYears': [8],
    'OverTime': ['Yes'],  # Categorical
    'YearsAtCompany': [6],
    'Age': [41],
    'JobLevel': [2],
    'Department': ['Sales'],  # Categorical
    'JobRole': ['Sales Executive'],  # Categorical
    'StockOptionLevel': [0],
    'EducationField': ['Life Sciences'],  # Categorical
    'JobInvolvement': [3]
})
```

```
54 new_data['TotalWorkingYears'] = log_transformer_total.transform(new_data[['TotalWorkingYears']])
55 new_data['YearsAtCompany'] = log_transformer_year.transform(new_data[['YearsAtCompany']])
56
57 # Tenure category transformation
58 new_data['Tenure_Category'] = new_data['YearsAtCompany'].apply(tenure_category)
59
60 # ✅ FIX: Encode `Tenure_Category` properly
61 if 'Tenure_Category' in label_encoders:
62     # If label encoding was used
63     new_data['Tenure_Category'] = label_encoders['Tenure_Category'].transform(new_data['Tenure_Category'])
64 else:
65     # If one-hot encoding was used
66     new_data = pd.get_dummies(new_data, columns=['Tenure_Category'], drop_first=True)
67
```

```
100
101 if prediction[0] == 1:
102     print("Predicted Output: [1] The employee is likely to leave (Attrition: Yes).")
103 else:
104     print("Predicted Output: [0] The employee is likely to stay (Attrition: No).")
```

```
Predicted Output: [1] The employee is likely to leave (Attrition: Yes).
```

Streamlit Application

Employee Attrition Prediction

Enter employee details to predict if they will leave or stay.

Monthly Income

2426

Total Working Years

6

OverTime

No

Years At Company

5

Stock Option Level

1

Education Field

Medical

Job Involvement

4

Predict Attrition

Prediction Result

The employee is likely to stay (Attrition: No).

Feature Selection

For 2nd Prediction [Monthly Income Prediction]

Use Filter and wrapper Method for Feature Selection for Regression Model

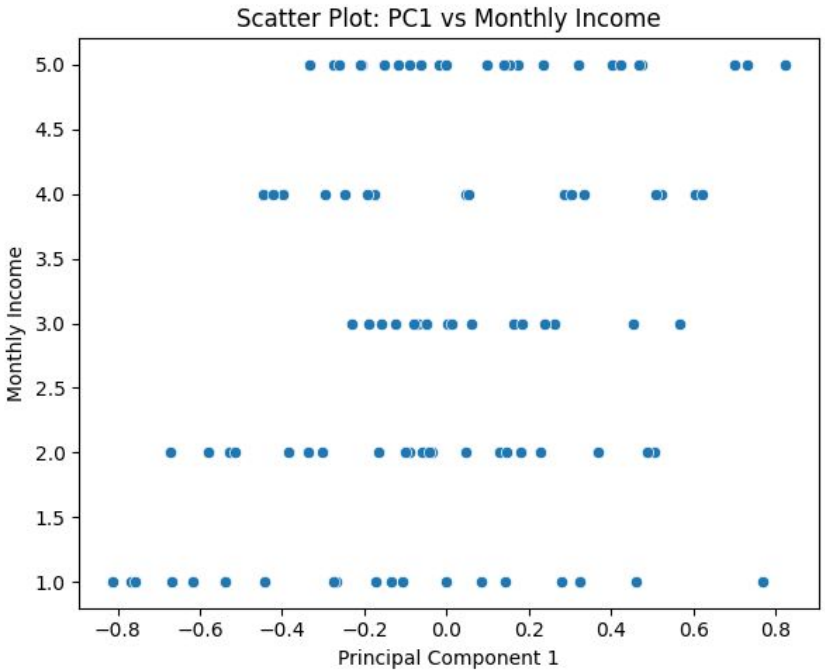
```
40 # Display performance results
41 regression_results
42
```

```
{'ANOVA': {'MSE': 7930444.111897583, 'R^2': 0.5608576910376343},
 'Mutual Information': {'MSE': 1074246.0936991968, 'R^2': 0.9405144398819818},
 'RFE': {'MSE': 1056550.9861008034, 'R^2': 0.9414942930022425}}
```

ANOVA Rejected because of Poor Accuracy

MI (Filter Method) and **RFE** (Wrapper Method) - and **RFE** have low Error value and high Accuracy Compare to **MI**

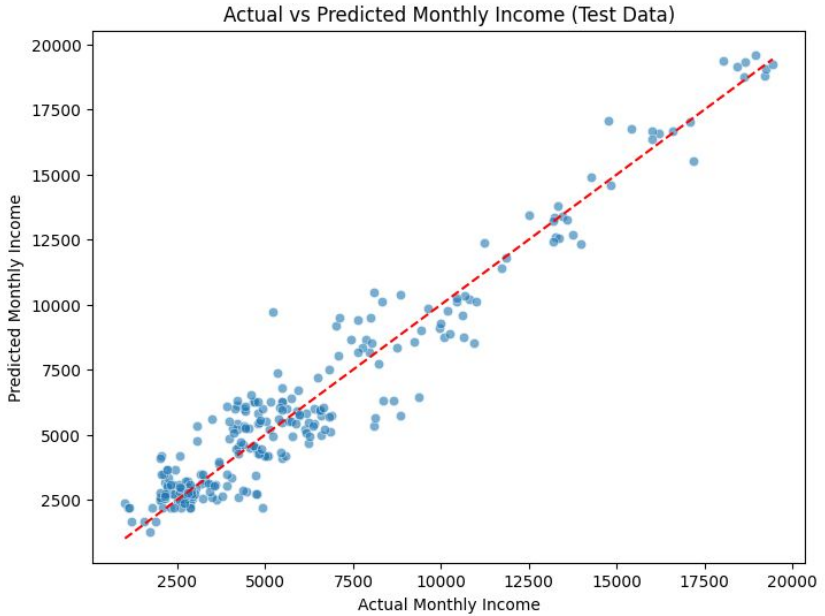
Check Line Plot for Target Columns to find the Problem Type based on PCA



1) Linear Regression Based on Correlation Heatmap (Filter Method)

```
### Model Performance Metrics ###
Train MSE: 2056779.8266, Test MSE: 2137514.7372
Train MAE: 1094.6984, Test MAE: 1132.1273
Train RMSE: 1434.1478, Test RMSE: 1462.0242
Train R²: 0.9114, Test R²: 0.8816
Train Adjusted R²: 0.9111, Test Adjusted R²: 0.8797

The model is GENERALIZING WELL. Performance is balanced between training and testing data.
```

Random Forest Regression



```
### Model Performance Metrics ###
Train MSE: 454536.9473, Test MSE: 1227385.1676
Train MAE: 471.2028, Test MAE: 858.8457
Train RMSE: 674.1936, Test RMSE: 1107.8742
Train R²: 0.9804, Test R²: 0.9320
Train Adjusted R²: 0.9803, Test Adjusted R²: 0.9309

The model is GENERALIZING WELL. Performance is balanced between training and testing data.
```

Random Forest is better compare to Linear because of error value are low and high accuracy

```
[ ]  1 ##########################################################################################
```

Finally i select Feature Based on Heatmap based Correlation (Because of Low MSE Values and High Accuracy Get and with Meaningful Features Compare to RFE

Loading the Saved Model

```
[ ]
```

```python
import joblib

# Load the trained Random Forest model
loaded_model = joblib.load("random_forest_regressor_model.pkl")

print("Model and features loaded successfully!")

```

```
Model and features loaded successfully!
```

Prediction

```python
import numpy as np

# Corrected input data with encoded categorical values
input_data = np.array([[1, 2.197225, 32, 2.079442, 217, 7, 2, 6, 1.386294]])  # Encoded values

# Reshape if predicting a single instance
input_data = input_data.reshape(1, -1)

# Make the prediction
prediction = loaded_model.predict(input_data)

print("Predicted Output:", prediction)
```

```
Predicted Output: [3068.]
```

```python
import numpy as np

# Corrected input data with encoded categorical values
input_data = np.array([[2, 2.197225, 41, 1.945910, 285, 4, 1, 5, 0.000000]])  # Encoded values

# Reshape if predicting a single instance
input_data = input_data.reshape(1, -1)

# Make the prediction
prediction = loaded_model.predict(input_data)

print("Predicted Output:", prediction)
```

```
Predicted Output: [5993.]
```

Prediction Working Perfectly

Now Using Transform Method To Decode the Encode Value for User Handel

Transform Work Based between Humans and Machine - Decode and Encode

Streamlit

Employee Monthly Income Prediction

Enter employee details to predict their monthly income.

Job Level

2 − +

Total Working Years

8 − +

Age

41 − +

Years at Company

6 − +

Years in Current Role

4 − +

Years with Current Manager

5 − +

Years Since Last Promotion

0 − +

Predict Monthly Income

Predicted Monthly Income: $5,993.00

Prediction Work Perfectly

Thank You