PROBLEM STATEMENT:

IBM faces a critical challenge with employee attrition. Understanding its drivers—job satisfaction, work-life balance, compensation, career growth, and organizational culture—is paramount. Through data analytics and predictive modelling, IBM aims to forecast attrition risks and implement targeted retention strategies. These include enhancing engagement initiatives, refining performance management, revising compensation, fostering a positive work environment, and improving leadership. Continuous evaluation ensures effectiveness, enabling iterative improvements. By addressing these factors, IBM endeavors to reduce attrition, nurture a stable, motivated workforce, and sustain its competitive edge.

INTRODUCTION:

- Using machine algorithms to predict employee attrition offers a comprehensive scope, involving data analysis, prediction modelling, and risk identification. By analysing historical data, these algorithms identify patterns and influential factors contributing to attrition, enabling organizations to predict future turnover rates and identify high-risk employees. This insight allows for proactive intervention and targeted retention strategies to retain valuable talent. Continuous refinement of predictive models ensures ongoing effectiveness in mitigating attrition and fostering a stable workforce.

OBJECTIVES:

Data Acquisition:

- We , likely won't have access to real IBM employee data due to privacy concerns. However, we can leverage publicly available datasets like the IBM HR Attrition dataset on Kaggle https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset.

Data Splitting and Preparation:

- The training set will be used to build the model, and the testing set will be used to evaluate its accuracy in predicting employee attrition.
- Explore and understand the features in the dataset. These might include factors like:
- Employee demographics (age, gender, department)
- Job details (job satisfaction, work-life balance, years with company)
- Performance metrics (salary, promotions, trainings received)

Data Preprocessing:

- Handle missing values, outliers, and any data inconsistencies through appropriate preprocessing techniques to ensure the dataset's quality.

Machine Learning Model Building:

- Apply various machine learning algorithms typically used for classification tasks, such as:
- Logistic Regression: A classic algorithm for predicting binary outcomes (employee leaving or staying)
- Decision Trees: Easy to interpret and understand, providing insights into key factors influencing attrition.

Evaluation and Feature Engineering:

- Train each model on the training set and evaluate its performance on the testing set using metrics like accuracy, precision, and recall.

Learning Outcomes:

- Gain experience with Python libraries like NumPy and scikit-learn for data manipulation and machine learning model development.
- Understand the process of building and evaluating machine learning models for real-world business problems like employee retention.

Solution Approach:

- **Logistic Regression**:
  Predicts the probability of an employee leaving based on their features, allowing for classification into high or low-risk categories. Analysing the model's coefficients will reveal which factors have the strongest positive or negative influence on employee retention.

- **K-Nearest Neighbors (KNN):**
  Classifies employees based on the similarity of their features to those who previously left IBM. Here, choosing the optimal K value and potentially scaling features are crucial.

- **Decision Tree:**
  Creates a tree-like structure that recursively splits the data based on the most important factors influencing employee departure. This method offers valuable insights into the key drivers of attrition.

- **Gradient Boosting Classifier:**
  Builds an ensemble of decision trees, where each tree focuses on correcting the errors of the previous one. This ensemble approach can lead to a robust and accurate model.

- **Random Forest:**

  Similar to gradient boosting, it builds an ensemble of decision trees but introduces randomness in feature selection at each split. This helps prevent overfitting and improve generalizability across different departments or demographics

- **Support Vector Machine (SVM):**

  Working: SVMs aim to find a hyperplane in the feature space that best separates the data points representing employees who left

(positive class) from those who stayed (negative class). This hyperplane maximizes the margin between the classes, leading to a robust decision boundary.
Considerations: Choosing the right kernel function (e.g., linear, radial basis) is crucial for SVM performance in employee attrition prediction. Feature scaling might also be necessary.

- **Neural Network:**

  Working: Neural networks are inspired by the human brain and consist of interconnected layers of artificial neurons. These neurons learn complex patterns from the data to predict employee attrition.
  Considerations: Neural networks can be powerful but require careful tuning of hyperparameters (e.g., number of layers, neurons per layer) to avoid overfitting and achieve optimal performance.

- **XGBoost (Extreme Gradient Boosting):**

  Working: XGBoost is an ensemble learning technique that builds a series of decision trees sequentially. Each tree focuses on correcting the errors of the previous one, resulting in a highly accurate model for predicting employee attrition.Benefits: XGBoost offers built-in regularization to prevent overfitting and handles missing values effectively. It can also be interpretable to some extent, providing insights into the factors influencing employee departure.

Timeline:

- **Phase 1: Planning (Days 1-3)**

  - Day 1: Define project goals and scope (focus on predicting employee attrition).
  - Day 2: Identify relevant employee data (features) and target variable (employee leaving or staying).
  - Day 3: Develop a basic workflow for data processing, model building, and evaluation.

Phase 2: Design (Days 4-7)

- Day 4: Download and explore the chosen employee dataset.
- Day 5: Analyze data distribution (e.g., histograms, boxplots) to identify potential issues like missing values or outliers.
- Day 6: Plan for data cleaning and pre-processing steps.
- Day 7: Choose machine learning algorithms to evaluate (e.g., Logistic Regression, Random Forest, XGBoost).

Phase 3: Develop (Days 8-10)

- Model Training (Days 8-9):
- Day 8: Pre-process data (handle missing values, categorical encoding, feature scaling if necessary).
- Day 9: Split data into training and testing sets. Train various machine learning models on the training set.
- Model Testing (Day 10):
- Evaluate model performance using metrics like accuracy, precision, recall, and F1-score on the testing set.
- Compare model performance and choose the best performing model for attrition prediction.

Phase 4: System Enhancement, Deployment, Release (Days 11-15)

- Enhancement (Days 11-12):
- Day 11: Refine the chosen model based on evaluation results (e.g., hyperparameter tuning).
- Day 12: Implement feature engineering techniques (create new features) to potentially improve model performance.
- Deployment (Days 13-14):
- Day 13: Develop a basic Flask application to deploy the model in a controlled environment.
- Day 14: Perform end-to-end testing of the deployed application.

- Release (Day 15):
- Document the project and prepare user instructions.
- Release the deployed application to a limited group of users for initial feedback

Dataset Overview:

1. Age
2. Attrition
3. Business Travel
4. Daily Rate
5. Department
6. Distance From Home
7. Education
8. Education Field
9. Employee Count

11. Gender
12. Hourly Rate
13. Employee Number
14. Hourly Rate
15. Job Involvement
16. Job Level
17. Job Role
18. Job Satisfaction
19. Marital Status
20. Monthly Income
21. Monthly Rate
22. Num Companies Worked
23. Over18
24. Over Time
25. Percent Salary Hike
26. Performance Rating
27. Relationship Satisfaction
28. Standard Hours
29. Stock Option Level
30. Total Working Years
31. Training Times Last Year

32. Work Life Balance
33. Years At Company
34. Years In Current Role
35. YearsSinceLastPromotion
36. Years With CurrManager

1) Importing necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
```

from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score
 # Update the path accordingly

2) Loading the dataset

```
from google.colab import files
uploaded = files.upload()
df = pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
```

3) Data Exploration

```
# Display basic information and the first few rows of the dataset
print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
None
   Age Attrition     BusinessTravel  DailyRate              Department
\
0   41       Yes      Travel_Rarely       1102                   Sales
1   49        No  Travel_Frequently        279  Research & Development
2   37       Yes      Travel_Rarely       1373  Research & Development
3   33        No  Travel_Frequently       1392  Research & Development
4   27        No      Travel_Rarely        591  Research & Development

   DistanceFromHome  Education EducationField  EmployeeCount
EmployeeNumber  \
0                 1          2  Life Sciences              1
1
1                 8          1  Life Sciences              1
2
2                 2          2          Other              1
4
3                 3          4  Life Sciences              1
5
4                 2          1        Medical              1
7

   ...  RelationshipSatisfaction  StandardHours  StockOptionLevel  \
0  ...                         1             80                 0
1  ...                         4             80                 1
2  ...                         2             80                 0
3  ...                         3             80                 0
4  ...                         4             80                 1

   TotalWorkingYears  TrainingTimesLastYear WorkLifeBalance
YearsAtCompany  \
0                  8                      0               1
6
1                 10                      3               3
10
2                  7                      3               3
0
3                  8                      3               3
8
4                  6                      3               3
2

  YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager
```

```
0                  4                        0                     5
1                  7                        1                     7
2                  0                        0                     0
3                  7                        3                     0
4                  2                        2                     2

[5 rows x 35 columns]
```

S"mmaíQ statistics roí →ı"mcíical rcat"ícs:

```
print(df.describe())
```

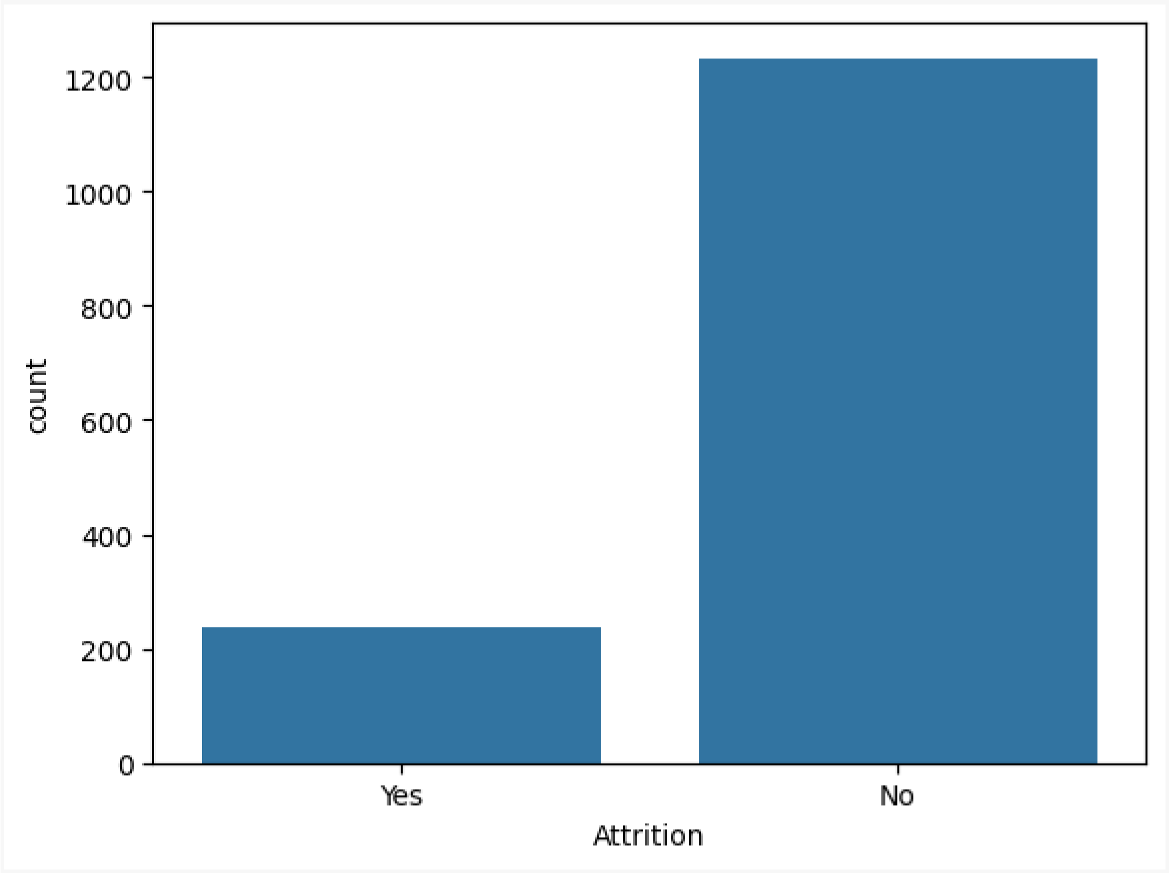
```
 Age     DailyRate  DistanceFromHome    Education  EmployeeCount  \
count  1470.000000  1470.000000       1470.000000  1470.000000
1470.0
mean     36.923810   802.485714          9.192517     2.912925
1.0
std       9.135373   403.509100          8.106864     1.024165
0.0
min      18.000000   102.000000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

       EmployeeNumber  EnvironmentSatisfaction   HourlyRate
JobInvolvement  \
count     1470.000000              1470.000000  1470.000000
1470.000000
mean      1024.865306                 2.721769    65.891156
2.729932
std        602.024335                 1.093082    20.329428
0.711561
min          1.000000                 1.000000    30.000000
1.000000
25%        491.250000                 2.000000    48.000000
2.000000
50%       1020.500000                 3.000000    66.000000
3.000000
75%       1555.750000                 4.000000    83.750000
3.000000
max       2068.000000                 4.000000   100.000000
4.000000

          JobLevel  ...  RelationshipSatisfaction  StandardHours  \
count  1470.000000  ...              1470.000000        1470.0
mean      2.063946  ...                 2.712245          80.0
std       1.106940  ...                 1.081209           0.0
min       1.000000  ...                 1.000000          80.0
25%       1.000000  ...                 2.000000          80.0
50%       2.000000  ...                 3.000000          80.0
75%       3.000000  ...                 4.000000          80.0
max       5.000000  ...                 4.000000          80.0

       StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
count       1470.000000        1470.000000            1470.000000
mean           0.793878          11.279592               2.799320
std            0.852077           7.780782               1.289271
min            0.000000           0.000000               0.000000
25%            0.000000           6.000000               2.000000
50%            1.000000          10.000000               3.000000
75%            1.000000          15.000000               3.000000
max            3.000000          40.000000               6.000000

       WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
count      1470.000000     1470.000000         1470.000000
mean          2.761224        7.008163            4.229252
std           0.706476        6.126525            3.623137
min           1.000000        0.000000            0.000000
25%           2.000000        3.000000            2.000000
50%           3.000000        5.000000            3.000000
75%           3.000000        9.000000            7.000000
max           4.000000       40.000000           18.000000

       YearsSinceLastPromotion  YearsWithCurrManager
count              1470.000000           1470.000000
mean                  2.187755              4.123129
std                   3.222430              3.568136
min                   0.000000              0.000000
25%                   0.000000              2.000000
50%                   1.000000              3.000000
75%                   3.000000              7.000000
max                  15.000000             17.000000

[8 rows x 26 columns]
```

4) Data Visualization

1. Co"→ıt plot roí tkc taígct :aíiablc 'Attíitio→ı':

```
sns.countplot(x='Attrition', data=df)
```

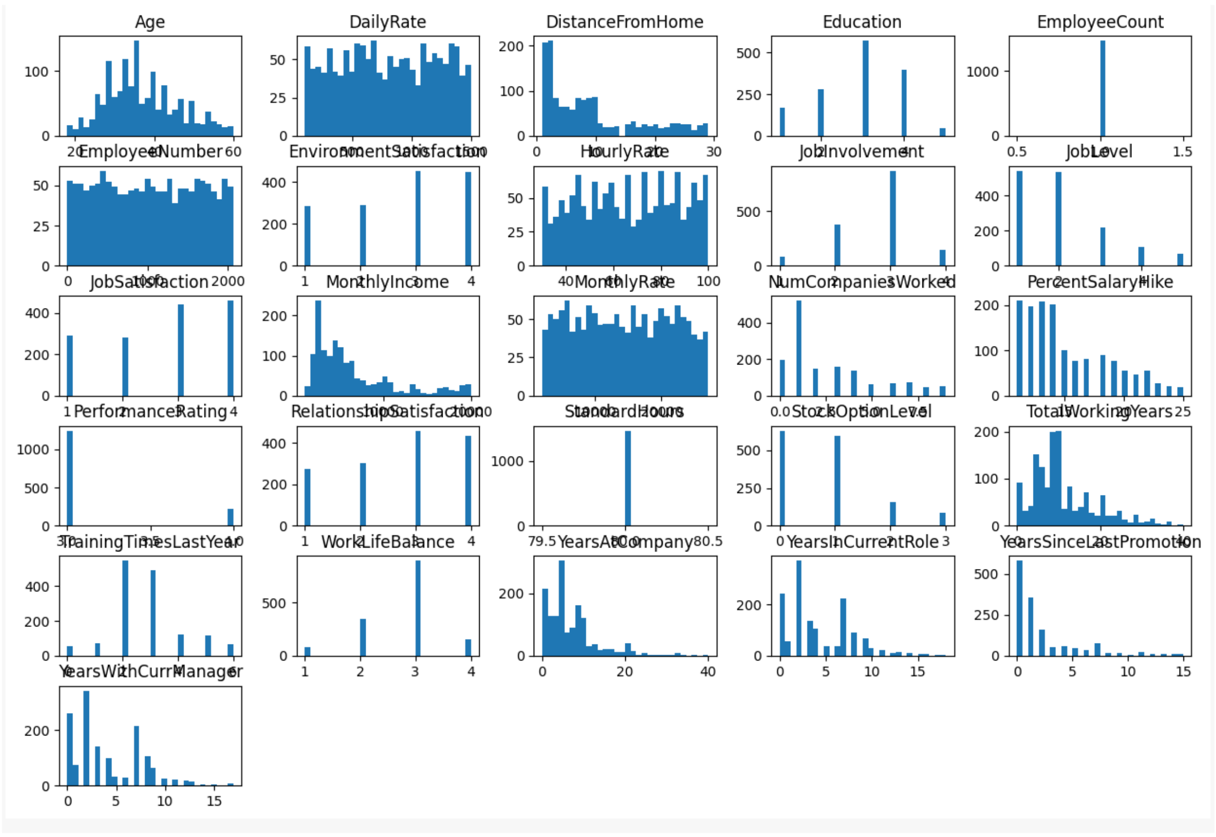
```
plt.show()
```



The count plot illustrates the distribution of employee attrition within the dataset. It is evident that the number of current employees ('No' attrition) significantly surpasses the number of employees who have left the company ('Yes' attrition). Such a distribution suggests that the dataset is imbalanced with respect to the target variable, which is an important characteristic to consider when developing predictive models, as it may influence model performance and will likely require specific techniques to handle the imbalance during the modeling phase.

2. Vis"alizc tkc distíib"tio→ı or →ı"mcíic rcat"ícs

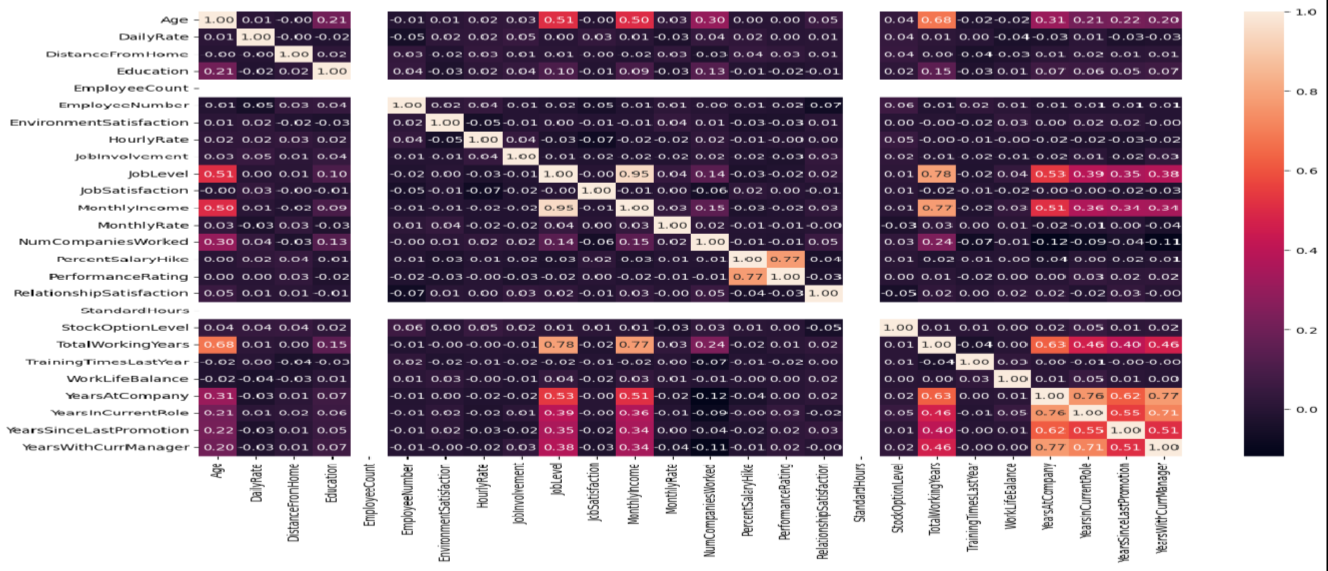
```
df.hist(bins=30, figsize=(15, 10), grid=False)
plt.show()
```



The histogram plots reveal the underlying distributions of the numeric features in the dataset. Features related to employee satisfaction and ratings tend to show a multi-modal distribution, likely reflecting the discrete nature of survey responses. Salary-related features, such as MonthlyIncome and PercentSalaryHike, along with tenure-related features, like TotalWorkingYears and YearsAtCompany, exhibit right-skewed distributions. This skewness indicates that a smaller proportion of the workforce has very high salaries or has been with the company for an extended period. The uniform distributions of DailyRate and HourlyRate suggest variability in compensation that does not concentrate around a particular figure. Additionally, some features like EmployeeCount may not vary across the workforce, indicating they might not provide discriminative power in predictive modeling of employee attrition.

«.Coííclatio→ı matíix kcatmap

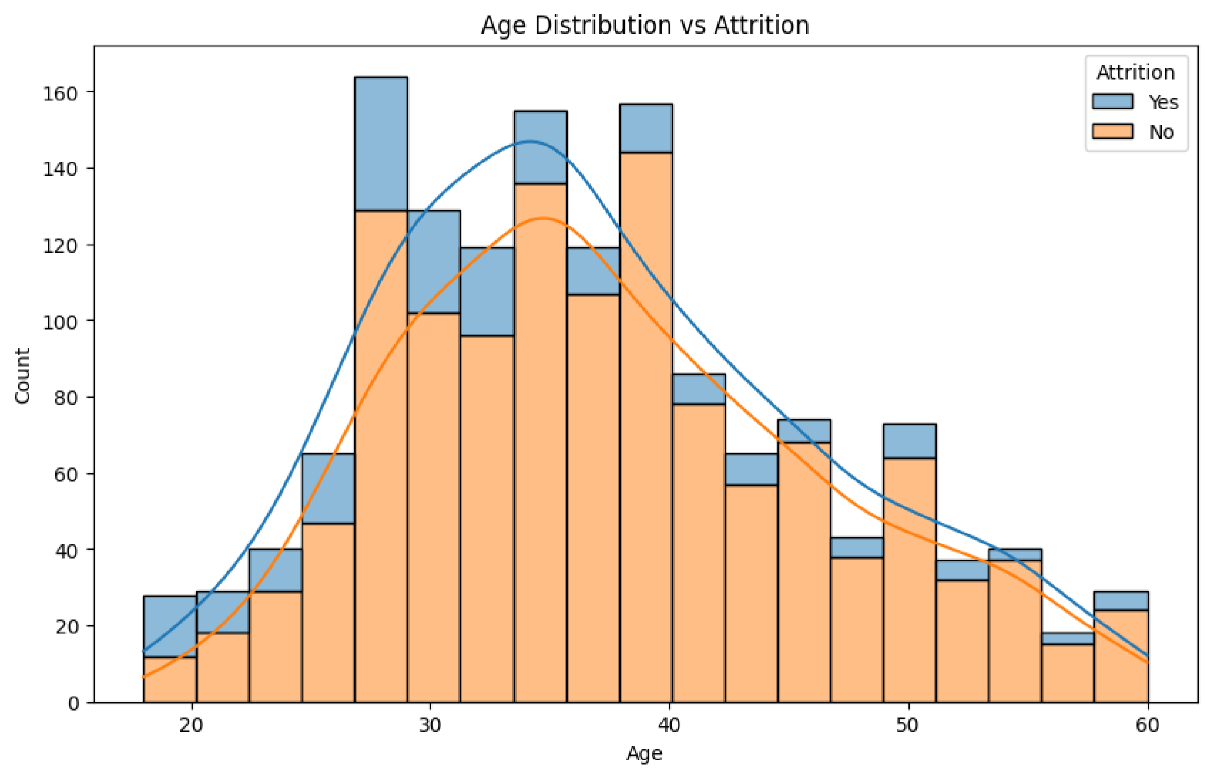
```
plt.figure(figsize=(15, 10))
sns.heatmap(df.corr(), annot=True, fmt=".2f")
plt.show()
```



The correlation matrix heatmap provides a visual representation of the relationship strength between numerical features. The intensity of the colors corresponds to the magnitude of the correlation coefficient, where warmer colors denote higher positive correlations and cooler colors indicate negative correlations. Diagonal elements are maximally correlated as they represent the correlation of each variable with itself. Notable correlations such as those between TotalWorkingYears and JobLevel suggest a relationship where employees with more years of experience are in higher job positions. Such insights can be valuable for predictive modeling and hypothesis generation. It's also important to consider these correlations for feature selection to mitigate potential multicollinearity issues in machine learning models.

«.ExploíatoíQ Kata A→ıalQsis

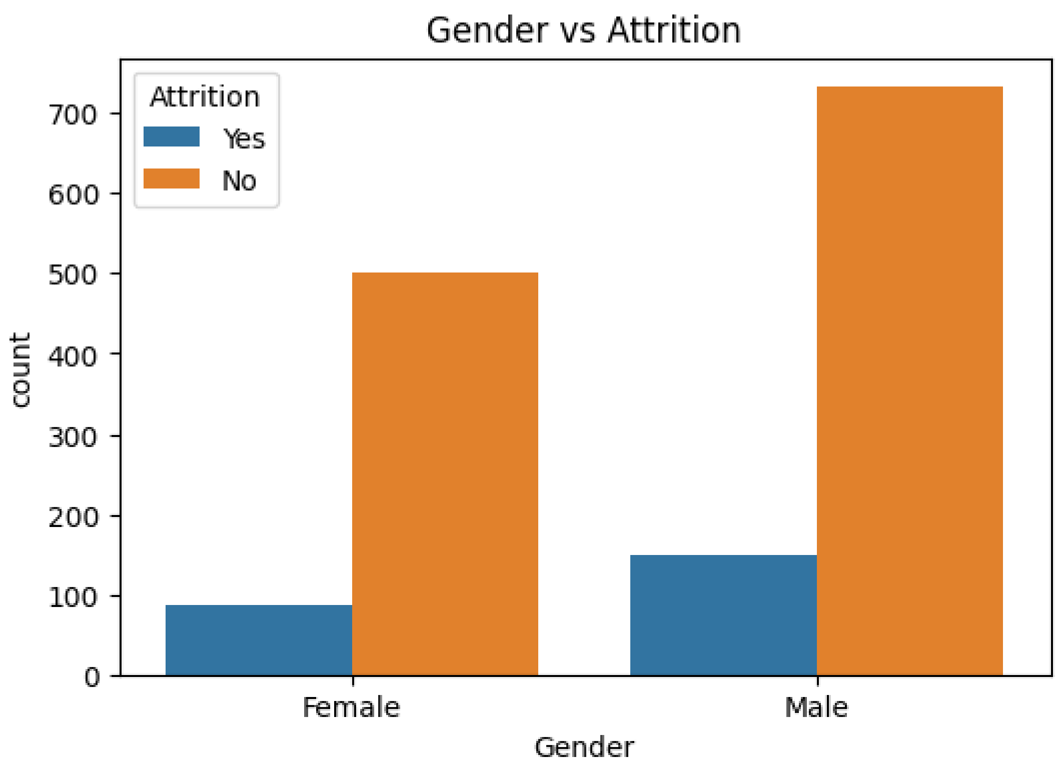
```
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='Gender', hue='Attrition')
plt.title('Gender vs Attrition')
plt.show()
```



The visualization provides a comparative view of the age distribution among current and former employees. The stacked histogram, complemented by the KDE curves, suggests a higher attrition rate among younger employees, while older employees show a higher retention rate. This pattern could indicate that age is an influential factor in employee turnover, with potential implications for HR policies and retention strategies. The density estimation curves help to highlight the differences in age distributions beyond the specific bin choices of the histogram, providing a smoother overview of the underlying age-related trends in attrition.

Gender vs Attrition

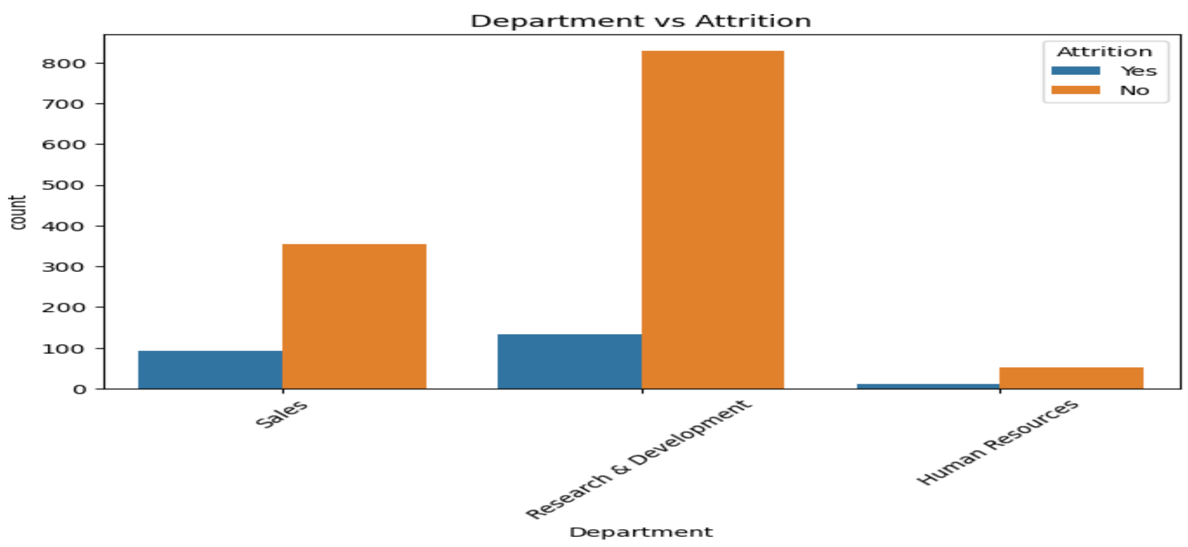
```
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='Gender', hue='Attrition')
plt.title('Gender vs Attrition')
plt.show()
```



The visualization provides a comparative view of the age distribution among current and former employees. The stacked histogram, complemented by the KDE curves, suggests a higher attrition rate among younger employees, while older employees show a higher retention rate. This pattern could indicate that age is an influential factor in employee turnover, with potential implications for HR policies and retention strategies. The density estimation curves help to highlight the differences in age distributions beyond the specific bin choices of the histogram, providing a smoother overview of the underlying age-related trends in attrition.

Department vs Attrition.

```
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='Department', hue='Attrition')
plt.title('Department vs Attrition')
plt.xticks(rotation=45)
plt.show()
```

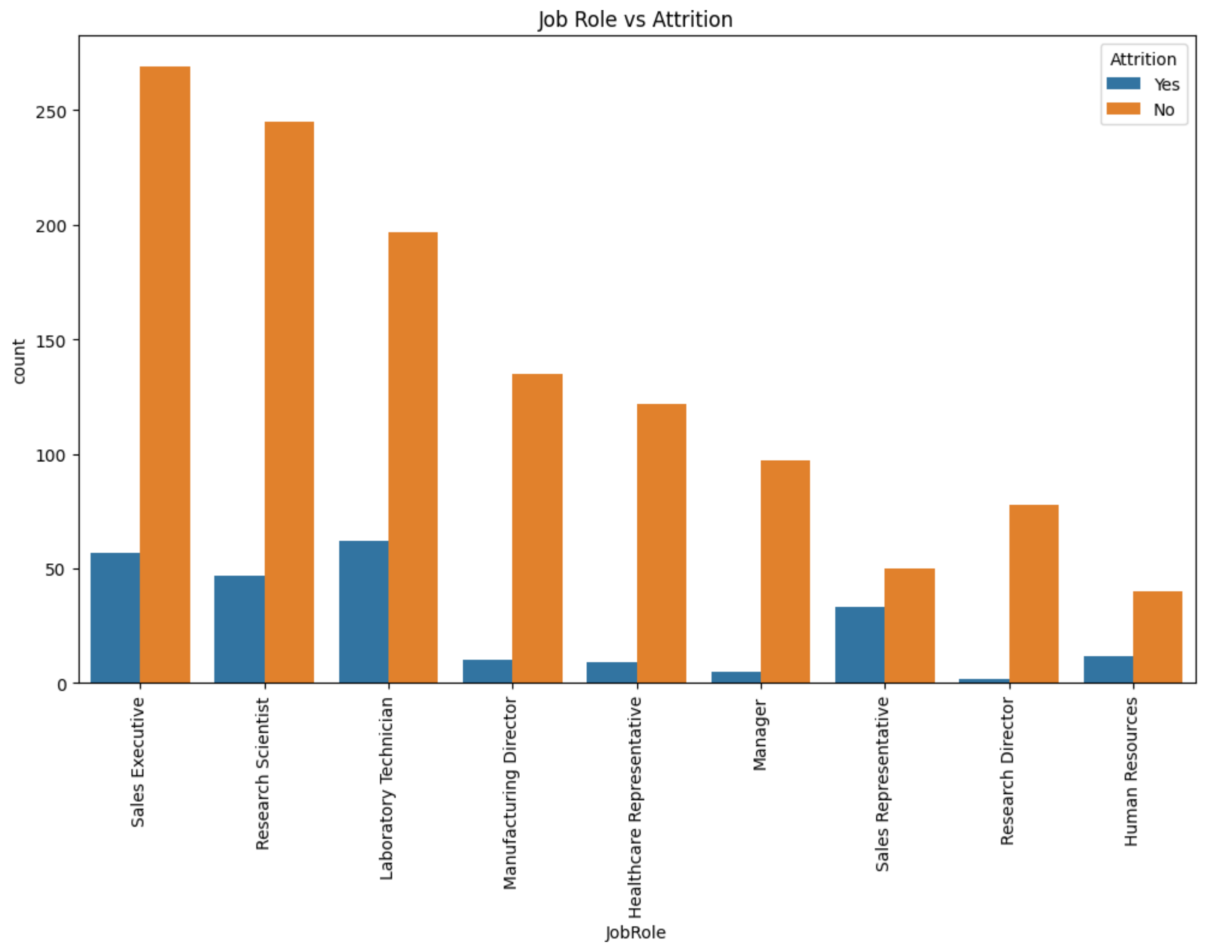


The Department vs Attrition bar chart elucidates the distribution of employee attrition across different departments within the organization. The Research & Development department, being the largest, exhibits the highest counts of both retained and departed employees. Sales, while smaller in size, also shows a considerable attrition count. Human Resources has the lowest overall count, consistent with its smaller department size. When examining the proportion of attrition, it is crucial to consider the relative department sizes as well as the absolute counts to gain a true understanding of attrition patterns within each department.

Job vs Attrition

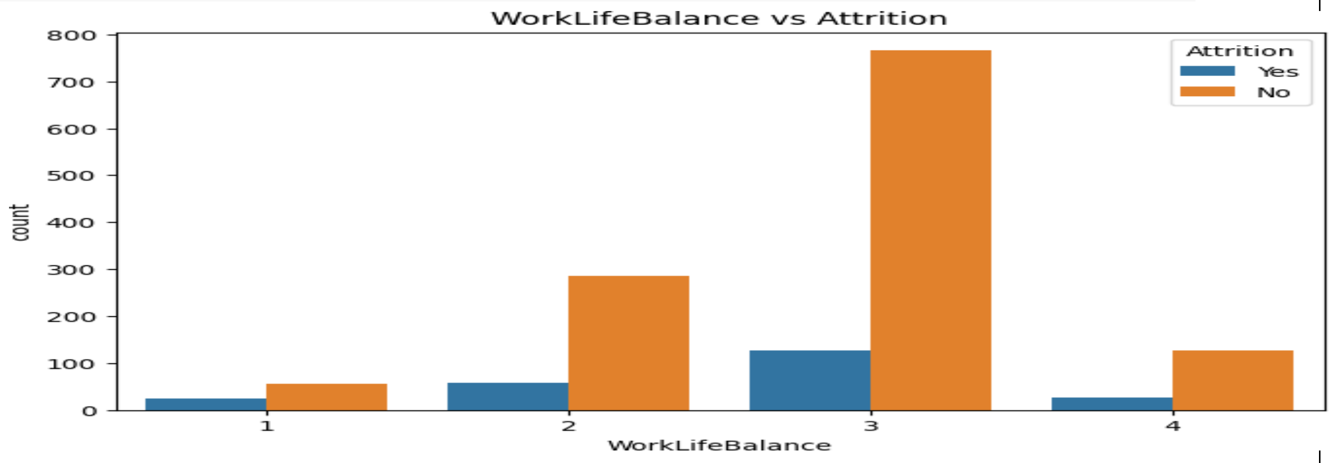
```
plt.figure(figsize=(12, 7))
sns.countplot(data=df, x='JobRole', hue='Attrition')
plt.title('Job Role vs Attrition')
plt.xticks(rotation=90)
plt.show()
```

The Job Role vs Attrition bar chart delineates the attrition rate across various job roles within the company. While roles like Sales Executive and Research Scientist have a higher absolute number of employees staying and leaving, the Laboratory Technician role stands out with a relatively high attrition rate compared to its size. In contrast, leadership roles such as Managers and Research Directors exhibit lower attrition rates, which aligns with expectations that higher job roles tend to have greater job stability. The data suggests that attrition is not uniformly distributed across job roles, and certain positions may require more attention to understand and address the underlying causes of turnover.



WorkLifeBalance vs Attrition

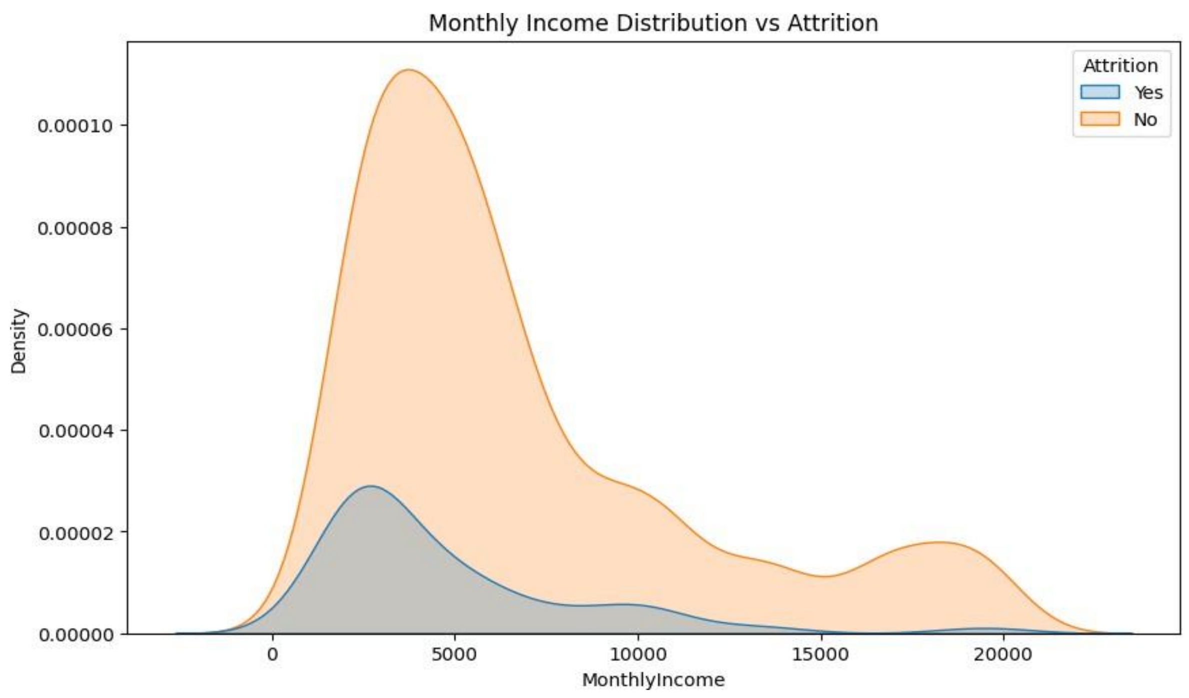
```
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='WorkLifeBalance', hue='Attrition')
plt.title('WorkLifeBalance vs Attrition')
plt.show()
```



The WorkLifeBalance vs Attrition chart depicts the correlation between employees' contentment with work-life balance and their decision to stay with or leave the company. A discernible trend shows that employees who report lower work-life balance ratings tend to leave the company at higher rates. Conversely, the highest work-life balance rating (4) corresponds with the lowest attrition, suggesting that employees who are most satisfied with their work-life integration are more inclined to remain at the company. This pattern underscores the importance of work-life balance as a factor in employee retention strategies

Monthly Income Distribution vs Attrition.

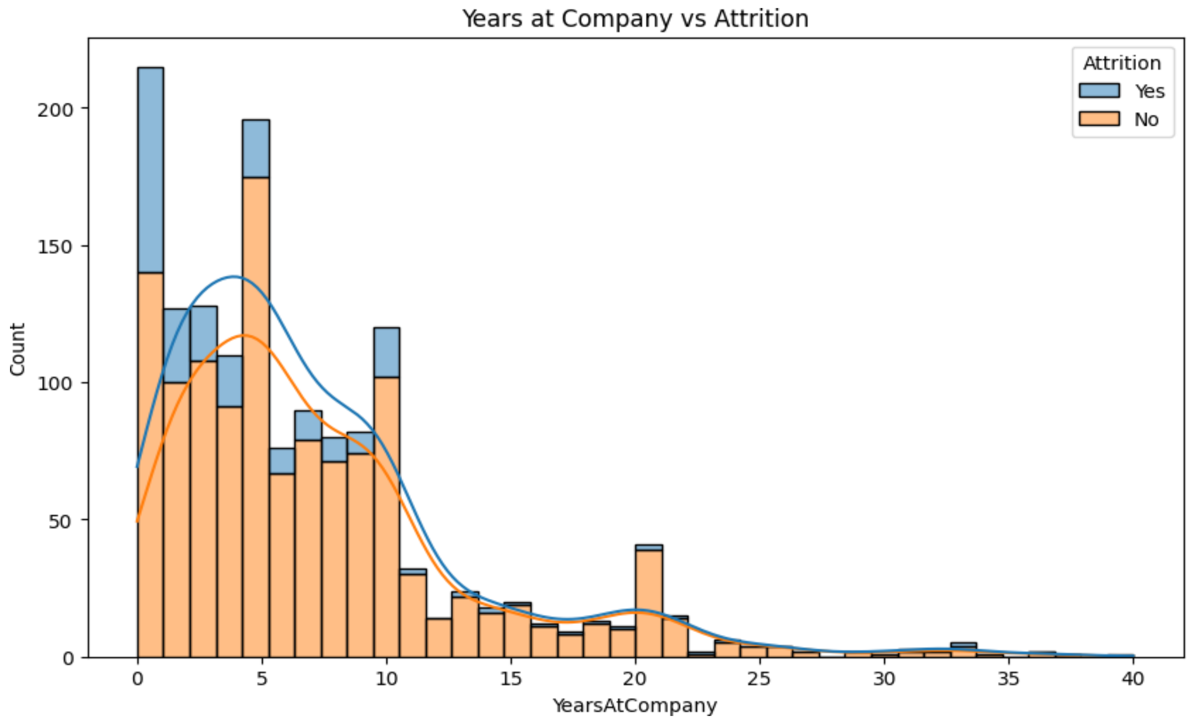
```
plt.figure(figsize=(10, 6))
sns.kdeplot(data=df, x='MonthlyIncome', hue='Attrition', fill=True)
plt.title('Monthly Income Distribution vs Attrition')
plt.show()
```



The Monthly Income Distribution vs Attrition KDE plot provides insight into the role of compensation in employee turnover. The plot reveals that employees with lower monthly incomes are more densely represented among those who have left the company, suggesting a trend where lower income may contribute to higher turnover rates. In contrast, the distribution for employees who remain with the company extends towards higher income levels, which could indicate that competitive compensation is effective for employee retention. This pattern highlights the importance of considering compensation strategies when addressing workforce stability concerns.

Years at Company vs Attrition.

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='YearsAtCompany', hue='Attrition',
multiple='stack', kde=True)
plt.title('Years at Company vs Attrition')
plt.show()
```



The Years at Company vs Attrition chart provides a clear depiction of tenure's relation to employee attrition. Notably, there is a higher frequency of attrition among employees with shorter tenures, as demonstrated by the early peak and quick tapering off of the 'Yes' distribution. In contrast, the 'No' distribution is more spread out, reflecting that employees who stay with the company are likely to do so over many years, with a substantial number having long-term tenures. This pattern highlights the potential of tenure as a predictor of employee retention and may point to the benefits of developing strategies that encourage long-term commitment from the workforce.

ľíai→ıi→ıg a→ıd tcsti→ıg ľkc modcl

```
# Assuming 'Attrition' is the target variable and it's binary (Yes/No)
df['Attrition'] = df['Attrition'].apply(lambda x: 1 if x == 'Yes' else
0)

# Splitting dataset into features (X) and target (y)
X = df.drop('Attrition', axis=1)
y = df['Attrition']
```

```
# Encoding categorical variables and scaling numerical variables
categorical_features = [col for col in X.columns if X[col].dtype ==
'object']
numerical_features = [col for col in X.columns if col not in
categorical_features]

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Gradient Boosting": GradientBoostingClassifier(),
    "SVM": SVC(probability=True),
    "KNN": KNeighborsClassifier(),
    "Neural Network": MLPClassifier(max_iter=1000),
    "XGBoost": XGBClassifier(use_label_encoder=False,
eval_metric='logloss')
}

results = {}

for name, model in models.items():
    # Pipeline for preprocessing and model training
    pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('model', model)])

    pipeline.fit(X_train, y_train)
    predictions = pipeline.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)

    results[name] = {
        'Accuracy': accuracy,
        'Precision': precision_score(y_test, predictions),
        'Recall': recall_score(y_test, predictions),
        'F1 Score': f1_score(y_test, predictions)
    }

# Displaying performance metrics for each model
for result in results:
    print(f"{result}: {results[result]}")
```

Result:

```
Logistic Regression: {'Accuracy': 0.8945578231292517, 'Precision':
0.6428571428571429, 'Recall': 0.46153846153846156, 'F1 Score':
0.537313432835821}
Decision Tree: {'Accuracy': 0.7721088435374149, 'Precision':
0.18181818181818182, 'Recall': 0.20512820512820512, 'F1 Score':
0.1927710843373494}
Random Forest: {'Accuracy': 0.8741496598639455, 'Precision':
0.6666666666666666, 'Recall': 0.10256410256410256, 'F1 Score':
0.17777777777777778}
Gradient Boosting: {'Accuracy': 0.8809523809523809, 'Precision':
0.6666666666666666, 'Recall': 0.20512820512820512, 'F1 Score':
0.3137254901960784}
SVM: {'Accuracy': 0.891156462585034, 'Precision': 1.0, 'Recall':
0.1794871794871795, 'F1 Score': 0.30434782608695654}
KNN: {'Accuracy': 0.8639455782312925, 'Precision': 0.4444444444444444,
'Recall': 0.10256410256410256, 'F1 Score': 0.16666666666666666}
Neural Network: {'Accuracy': 0.8639455782312925, 'Precision':
0.4838709677419355, 'Recall': 0.38461538461538464, 'F1 Score':
0.42857142857142856}
XGBoost: {'Accuracy': 0.8775510204081632, 'Precision':
0.5882352941176471, 'Recall': 0.2564102564102564, 'F1 Score':
0.35714285714285715}
```

```
# Example for tuning a Random Forest Classifier
param_grid = {
    'model__n_estimators': [100, 200],
    'model__max_depth': [10, 20, None],
    'model__min_samples_split': [2, 5],
    'model__min_samples_leaf': [1, 2]
}

pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('model', RandomForestClassifier())])

grid_search = GridSearchCV(pipeline, param_grid, cv=5,
scoring='accuracy')
grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
print("Best accuracy:", grid_search.best_score_)
```

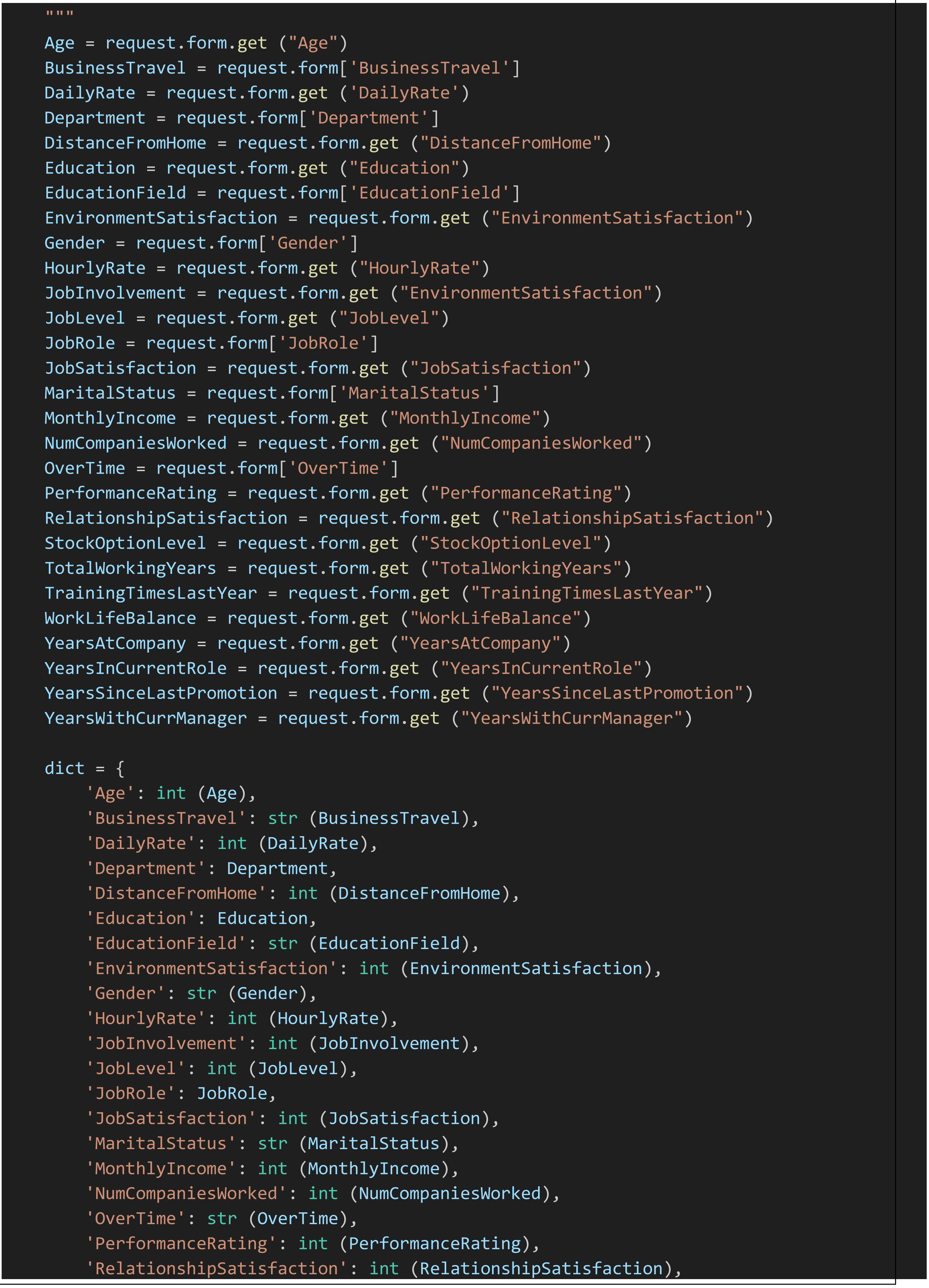
Best parameters: {'model__max_depth': 10, 'model__min_samples_leaf': 1, 'model__min_samples_split': 2, 'model__n_estimators': 100}
Best accuracy: 0.8613883880274071

```
# Train the final model (example with Random
Forest)
final_model = grid_search.best_estimator_
final_predictions = final_model.predict(X_test)

# Final evaluation
print(f"Final Model Accuracy: {accuracy_score(y_test,
final_predictions)}")
print(classification_report(y_test, final_predictions))
```

```
Final Model Accuracy: 0.8707482993197279
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       255
           1       0.57      0.10      0.17        39

    accuracy                           0.87       294
   macro avg       0.72      0.55      0.55       294
weighted avg       0.84      0.87      0.83       294
```

Backend Flask Code :

```
import numpy as np
import scipy as sp
import pandas as pd
from flask import Flask,request,jsonify,render_template
import pickle

app = Flask (__name__)
model = pickle.load (open ('C:/Users/vaish/OneDrive/Documents/internship/model.pkl','rb'))


@app.route ('/')
def home():
    return render_template ('index.html')


@app.route ('/predict',methods=['POST','GET'])
def predict():
    """
    For rendering results on HTML GUI
```



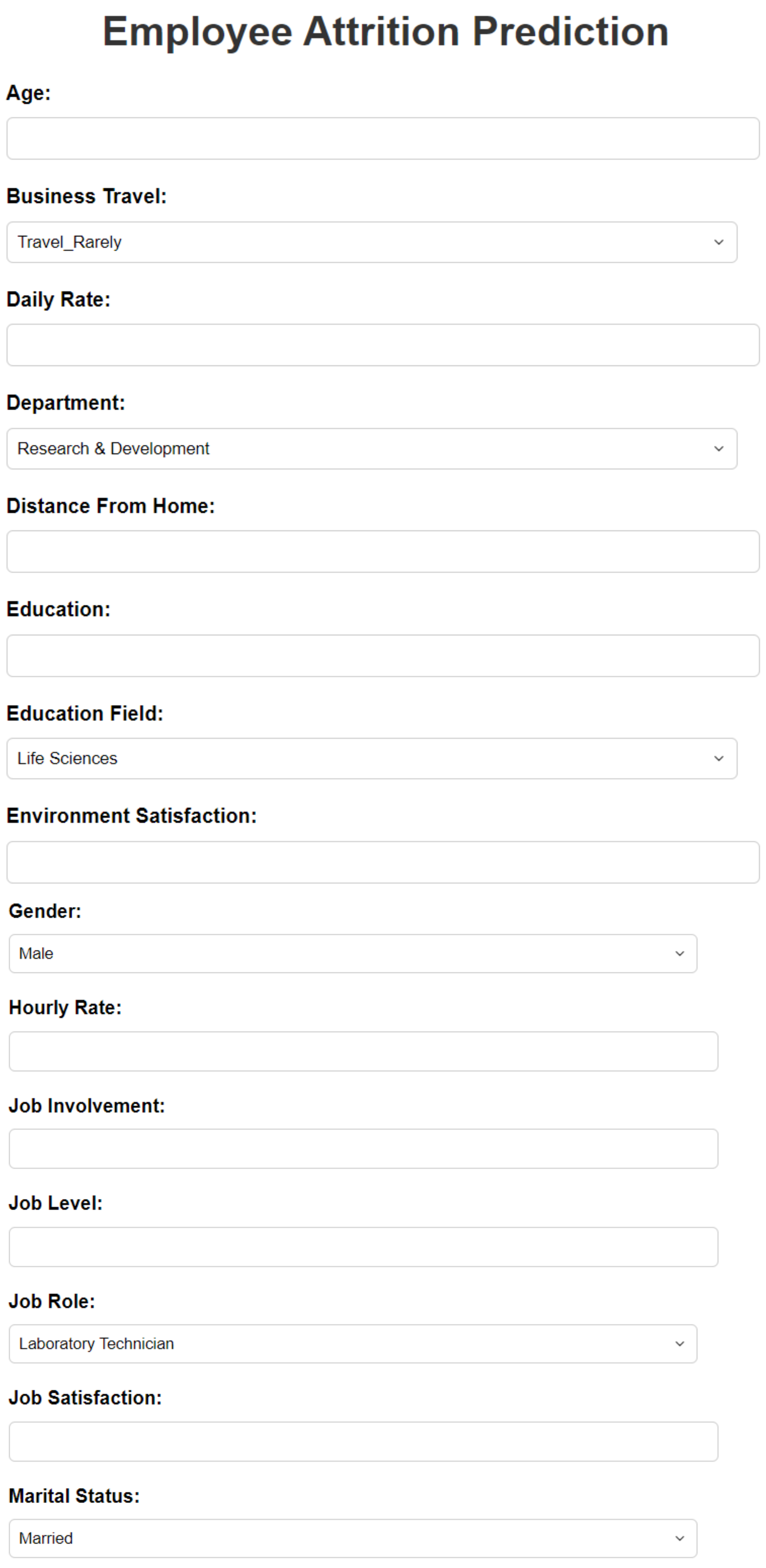
```
"""
Age = request.form.get ("Age")
BusinessTravel = request.form['BusinessTravel']
DailyRate = request.form.get ('DailyRate')
Department = request.form['Department']
DistanceFromHome = request.form.get ("DistanceFromHome")
Education = request.form.get ("Education")
EducationField = request.form['EducationField']
EnvironmentSatisfaction = request.form.get ("EnvironmentSatisfaction")
Gender = request.form['Gender']
HourlyRate = request.form.get ("HourlyRate")
JobInvolvement = request.form.get ("EnvironmentSatisfaction")
JobLevel = request.form.get ("JobLevel")
JobRole = request.form['JobRole']
JobSatisfaction = request.form.get ("JobSatisfaction")
MaritalStatus = request.form['MaritalStatus']
MonthlyIncome = request.form.get ("MonthlyIncome")
NumCompaniesWorked = request.form.get ("NumCompaniesWorked")
OverTime = request.form['OverTime']
PerformanceRating = request.form.get ("PerformanceRating")
RelationshipSatisfaction = request.form.get ("RelationshipSatisfaction")
StockOptionLevel = request.form.get ("StockOptionLevel")
TotalWorkingYears = request.form.get ("TotalWorkingYears")
TrainingTimesLastYear = request.form.get ("TrainingTimesLastYear")
WorkLifeBalance = request.form.get ("WorkLifeBalance")
YearsAtCompany = request.form.get ("YearsAtCompany")
YearsInCurrentRole = request.form.get ("YearsInCurrentRole")
YearsSinceLastPromotion = request.form.get ("YearsSinceLastPromotion")
YearsWithCurrManager = request.form.get ("YearsWithCurrManager")

dict = {
    'Age': int (Age),
    'BusinessTravel': str (BusinessTravel),
    'DailyRate': int (DailyRate),
    'Department': Department,
    'DistanceFromHome': int (DistanceFromHome),
    'Education': Education,
    'EducationField': str (EducationField),
    'EnvironmentSatisfaction': int (EnvironmentSatisfaction),
    'Gender': str (Gender),
    'HourlyRate': int (HourlyRate),
    'JobInvolvement': int (JobInvolvement),
    'JobLevel': int (JobLevel),
    'JobRole': JobRole,
    'JobSatisfaction': int (JobSatisfaction),
    'MaritalStatus': str (MaritalStatus),
    'MonthlyIncome': int (MonthlyIncome),
    'NumCompaniesWorked': int (NumCompaniesWorked),
    'OverTime': str (OverTime),
    'PerformanceRating': int (PerformanceRating),
    'RelationshipSatisfaction': int (RelationshipSatisfaction),
        'StockOptionLevel': StockOptionLevel,
        'TotalWorkingYears': int (TotalWorkingYears),
        'TrainingTimesLastYear': TrainingTimesLastYear,
        'WorkLifeBalance': int (WorkLifeBalance),
        'YearsAtCompany': int (YearsAtCompany),
        'YearsInCurrentRole': int (YearsInCurrentRole),
        'YearsSinceLastPromotion': int (YearsSinceLastPromotion),
        'YearsWithCurrManager': int (YearsWithCurrManager)
    }

    df = pd.DataFrame ([dict])

    df['Total_Satisfaction'] = (df['EnvironmentSatisfaction'] +
                                df['JobInvolvement'] +
                                df['JobSatisfaction'] +
                                df['RelationshipSatisfaction'] +
                                df['WorkLifeBalance']) / 5

    # Drop Columns
    df.drop (
        ['EnvironmentSatisfaction','JobInvolvement','JobSatisfaction','RelationshipSatisfa
ction','WorkLifeBalance'],
        axis=1,inplace=True)

    # Convert Total satisfaction into boolean
    df['Total_Satisfaction_bool'] = df['Total_Satisfaction'].apply (lambda x: 1 if x >=
2.8 else 0)
    df.drop ('Total_Satisfaction',axis=1,inplace=True)

    # It can be observed that the rate of attrition of employees below age of 35 is high
    df['Age_bool'] = df['Age'].apply (lambda x: 1 if x < 35 else 0)
    df.drop ('Age',axis=1,inplace=True)

    # It can be observed that the employees are more likey the drop the job if dailyRate
less than 800
    df['DailyRate_bool'] = df['DailyRate'].apply (lambda x: 1 if x < 800 else 0)
    df.drop ('DailyRate',axis=1,inplace=True)

    # Employees working at R&D Department have higher attrition rate
    df['Department_bool'] = df['Department'].apply (lambda x: 1 if x == 'Research &
Development' else 0)
    df.drop ('Department',axis=1,inplace=True)

    # Rate of attrition of employees is high if DistanceFromHome > 10
    df['DistanceFromHome_bool'] = df['DistanceFromHome'].apply (lambda x: 1 if x > 10 else
0)
    df.drop ('DistanceFromHome',axis=1,inplace=True)

    # Employees are more likey to drop the job if the employee is working as Laboratory
Technician
```

```
    df['JobRole_bool'] = df['JobRole'].apply (lambda x: 1 if x == 'Laboratory Technician'
else 0)
    df.drop ('JobRole',axis=1,inplace=True)

    # Employees are more likey to the drop the job if the employee's hourly rate < 65
    df['HourlyRate_bool'] = df['HourlyRate'].apply (lambda x: 1 if x < 65 else 0)
    df.drop ('HourlyRate',axis=1,inplace=True)

    # Employees are more likey to the drop the job if the employee's MonthlyIncome < 4000
    df['MonthlyIncome_bool'] = df['MonthlyIncome'].apply (lambda x: 1 if x < 4000 else 0)
    df.drop ('MonthlyIncome',axis=1,inplace=True)

    # Rate of attrition of employees is high if NumCompaniesWorked < 3
    df['NumCompaniesWorked_bool'] = df['NumCompaniesWorked'].apply (lambda x: 1 if x > 3
else 0)
    df.drop ('NumCompaniesWorked',axis=1,inplace=True)

    # Employees are more likey to the drop the job if the employee's TotalWorkingYears < 8
    df['TotalWorkingYears_bool'] = df['TotalWorkingYears'].apply (lambda x: 1 if x < 8
else 0)
    df.drop ('TotalWorkingYears',axis=1,inplace=True)

    # Employees are more likey to the drop the job if the employee's YearsAtCompany < 3
    df['YearsAtCompany_bool'] = df['YearsAtCompany'].apply (lambda x: 1 if x < 3 else 0)
    df.drop ('YearsAtCompany',axis=1,inplace=True)

    # Employees are more likey to the drop the job if the employee's YearsInCurrentRole <
3
    df['YearsInCurrentRole_bool'] = df['YearsInCurrentRole'].apply (lambda x: 1 if x < 3
else 0)
    df.drop ('YearsInCurrentRole',axis=1,inplace=True)

    # Employees are more likely to the drop the job if the employee's
YearsSinceLastPromotion < 1
    df['YearsSinceLastPromotion_bool'] = df['YearsSinceLastPromotion'].apply (lambda x: 1
if x < 1 else 0)
    df.drop ('YearsSinceLastPromotion',axis=1,inplace=True)

    # Employees are more likely to the drop the job if the employee's YearsWithCurrManager
< 1
    df['YearsWithCurrManager_bool'] = df['YearsWithCurrManager'].apply (lambda x: 1 if x <
1 else 0)
    df.drop ('YearsWithCurrManager',axis=1,inplace=True)

    # Convert Categorical to Numerical
    # Buisness Travel
    if BusinessTravel == 'Rarely':
        df['BusinessTravel_Rarely'] = 1
        df['BusinessTravel_Frequently'] = 0
        df['BusinessTravel_No_Travel'] = 0
    elif BusinessTravel == 'Frequently':
        df['BusinessTravel_Rarely'] = 0
        df['BusinessTravel_Frequently'] = 1
        df['BusinessTravel_No_Travel'] = 0
    else:
        df['BusinessTravel_Rarely'] = 0
        df['BusinessTravel_Frequently'] = 0
        df['BusinessTravel_No_Travel'] = 1
    df.drop ('BusinessTravel',axis=1,inplace=True)

    # Education
    if Education == 1:
        df['Education_1'] = 1
        df['Education_2'] = 0
        df['Education_3'] = 0
        df['Education_4'] = 0
        df['Education_5'] = 0
    elif Education == 2:
        df['Education_1'] = 0
        df['Education_2'] = 1
        df['Education_3'] = 0
        df['Education_4'] = 0
        df['Education_5'] = 0
    elif Education == 3:
        df['Education_1'] = 0
        df['Education_2'] = 0
        df['Education_3'] = 1
        df['Education_4'] = 0
        df['Education_5'] = 0
    elif Education == 4:
        df['Education_1'] = 0
        df['Education_2'] = 0
        df['Education_3'] = 0
        df['Education_4'] = 1
        df['Education_5'] = 0
    else:
        df['Education_1'] = 0
        df['Education_2'] = 0
        df['Education_3'] = 0
        df['Education_4'] = 0
        df['Education_5'] = 1
    df.drop ('Education',axis=1,inplace=True)

    # EducationField
    if EducationField == 'Life Sciences':
        df['EducationField_Life_Sciences'] = 1
        df['EducationField_Medical'] = 0
        df['EducationField_Marketing'] = 0
        df['EducationField_Technical_Degree'] = 0
        df['Education_Human_Resources'] = 0
        df['Education_Other'] = 0
    elif EducationField == 'Medical':
```

```
    df['EducationField_Life_Sciences'] = 0
    df['EducationField_Medical'] = 1
    df['EducationField_Marketing'] = 0
    df['EducationField_Technical_Degree'] = 0
    df['Education_Human_Resources'] = 0
    df['Education_Other'] = 0
elif EducationField == 'Marketing':
    df['EducationField_Life_Sciences'] = 0
    df['EducationField_Medical'] = 0
    df['EducationField_Marketing'] = 1
    df['EducationField_Technical_Degree'] = 0
    df['Education_Human_Resources'] = 0
    df['Education_Other'] = 0
elif EducationField == 'Technical Degree':
    df['EducationField_Life_Sciences'] = 0
    df['EducationField_Medical'] = 0
    df['EducationField_Marketing'] = 0
    df['EducationField_Technical_Degree'] = 1
    df['Education_Human_Resources'] = 0
    df['Education_Other'] = 0
elif EducationField == 'Human Resources':
    df['EducationField_Life_Sciences'] = 0
    df['EducationField_Medical'] = 0
    df['EducationField_Marketing'] = 0
    df['EducationField_Technical_Degree'] = 0
    df['Education_Human_Resources'] = 1
    df['Education_Other'] = 0
else:
    df['EducationField_Life_Sciences'] = 0
    df['EducationField_Medical'] = 0
    df['EducationField_Marketing'] = 0
    df['EducationField_Technical_Degree'] = 0
    df['Education_Human_Resources'] = 1
    df['Education_Other'] = 1
df.drop ('EducationField',axis=1,inplace=True)

# Gender
if Gender == 'Male':
    df['Gender_Male'] = 1
    df['Gender_Female'] = 0
else:
    df['Gender_Male'] = 0
    df['Gender_Female'] = 1
df.drop ('Gender',axis=1,inplace=True)

# Marital Status
if MaritalStatus == 'Married':
    df['MaritalStatus_Married'] = 1
    df['MaritalStatus_Single'] = 0
    df['MaritalStatus_Divorced'] = 0
elif MaritalStatus == 'Single':
```

```python
        df['MaritalStatus_Married'] = 0
        df['MaritalStatus_Single'] = 1
        df['MaritalStatus_Divorced'] = 0
    else:
        df['MaritalStatus_Married'] = 0
        df['MaritalStatus_Single'] = 0
        df['MaritalStatus_Divorced'] = 1
    df.drop ('MaritalStatus',axis=1,inplace=True)

    # Overtime
    if OverTime == 'Yes':
        df['OverTime_Yes'] = 1
        df['OverTime_No'] = 0
    else:
        df['OverTime_Yes'] = 0
        df['OverTime_No'] = 1
    df.drop ('OverTime',axis=1,inplace=True)

    # Stock Option Level
    if StockOptionLevel == 0:
        df['StockOptionLevel_0'] = 1
        df['StockOptionLevel_1'] = 0
        df['StockOptionLevel_2'] = 0
        df['StockOptionLevel_3'] = 0
    elif StockOptionLevel == 1:
        df['StockOptionLevel_0'] = 0
        df['StockOptionLevel_1'] = 1
        df['StockOptionLevel_2'] = 0
        df['StockOptionLevel_3'] = 0
    elif StockOptionLevel == 2:
        df['StockOptionLevel_0'] = 0
        df['StockOptionLevel_1'] = 0
        df['StockOptionLevel_2'] = 1
        df['StockOptionLevel_3'] = 0
    else:
        df['StockOptionLevel_0'] = 0
        df['StockOptionLevel_1'] = 0
        df['StockOptionLevel_2'] = 0
        df['StockOptionLevel_3'] = 1
    df.drop ('StockOptionLevel',axis=1,inplace=True)

    # Training Time Last Year
    if TrainingTimesLastYear == 0:
        df['TrainingTimesLastYear_0'] = 1
        df['TrainingTimesLastYear_1'] = 0
        df['TrainingTimesLastYear_2'] = 0
        df['TrainingTimesLastYear_3'] = 0
        df['TrainingTimesLastYear_4'] = 0
        df['TrainingTimesLastYear_5'] = 0
        df['TrainingTimesLastYear_6'] = 0
    elif TrainingTimesLastYear == 1:
        df['TrainingTimesLastYear_0'] = 0
        df['TrainingTimesLastYear_1'] = 1
        df['TrainingTimesLastYear_2'] = 0
        df['TrainingTimesLastYear_3'] = 0
        df['TrainingTimesLastYear_4'] = 0
        df['TrainingTimesLastYear_5'] = 0
        df['TrainingTimesLastYear_6'] = 0
    elif TrainingTimesLastYear == 2:
        df['TrainingTimesLastYear_0'] = 0
        df['TrainingTimesLastYear_1'] = 0
        df['TrainingTimesLastYear_2'] = 1
        df['TrainingTimesLastYear_3'] = 0
        df['TrainingTimesLastYear_4'] = 0
        df['TrainingTimesLastYear_5'] = 0
        df['TrainingTimesLastYear_6'] = 0
    elif TrainingTimesLastYear == 3:
        df['TrainingTimesLastYear_0'] = 0
        df['TrainingTimesLastYear_1'] = 0
        df['TrainingTimesLastYear_2'] = 0
        df['TrainingTimesLastYear_3'] = 1
        df['TrainingTimesLastYear_4'] = 0
        df['TrainingTimesLastYear_5'] = 0
        df['TrainingTimesLastYear_6'] = 0
    elif TrainingTimesLastYear == 4:
        df['TrainingTimesLastYear_0'] = 0
        df['TrainingTimesLastYear_1'] = 0
        df['TrainingTimesLastYear_2'] = 0
        df['TrainingTimesLastYear_3'] = 0
        df['TrainingTimesLastYear_4'] = 1
        df['TrainingTimesLastYear_5'] = 0
        df['TrainingTimesLastYear_6'] = 0
    elif TrainingTimesLastYear == 5:
        df['TrainingTimesLastYear_0'] = 0
        df['TrainingTimesLastYear_1'] = 0
        df['TrainingTimesLastYear_2'] = 0
        df['TrainingTimesLastYear_3'] = 0
        df['TrainingTimesLastYear_4'] = 0
        df['TrainingTimesLastYear_5'] = 1
        df['TrainingTimesLastYear_6'] = 0
    else:
        df['TrainingTimesLastYear_0'] = 0
        df['TrainingTimesLastYear_1'] = 0
        df['TrainingTimesLastYear_2'] = 0
        df['TrainingTimesLastYear_3'] = 0
        df['TrainingTimesLastYear_4'] = 0
        df['TrainingTimesLastYear_5'] = 0
        df['TrainingTimesLastYear_6'] = 1
    df.drop ('TrainingTimesLastYear',axis=1,inplace=True)

    df.to_csv ('features.csv',index=False)
    prediction = model.predict (df)



    if prediction == 0:
        return render_template ('index.html',prediction_text='Employee Might Not Leave The
Job')

    else:
        return render_template ('index.html',prediction_text='Employee Might Leave The
Job')





    # Drop Columns


    # Convert Total satisfaction into boolean

    # Convert Categorical to Numerical
    # Buisness Travel


    # Education


    # df.to_csv ('features.csv',index=False)
    print(df)




if __name__ == "__main__":
    app.run (debug=True)
```

Basic Front-End UI :



Employee Attrition Prediction

Age:

Business Travel: Travel_Rarely

Daily Rate:

Department: Research & Development

Distance From Home:

Education:

Education Field: Life Sciences

Environment Satisfaction:

Gender: Male

Hourly Rate:

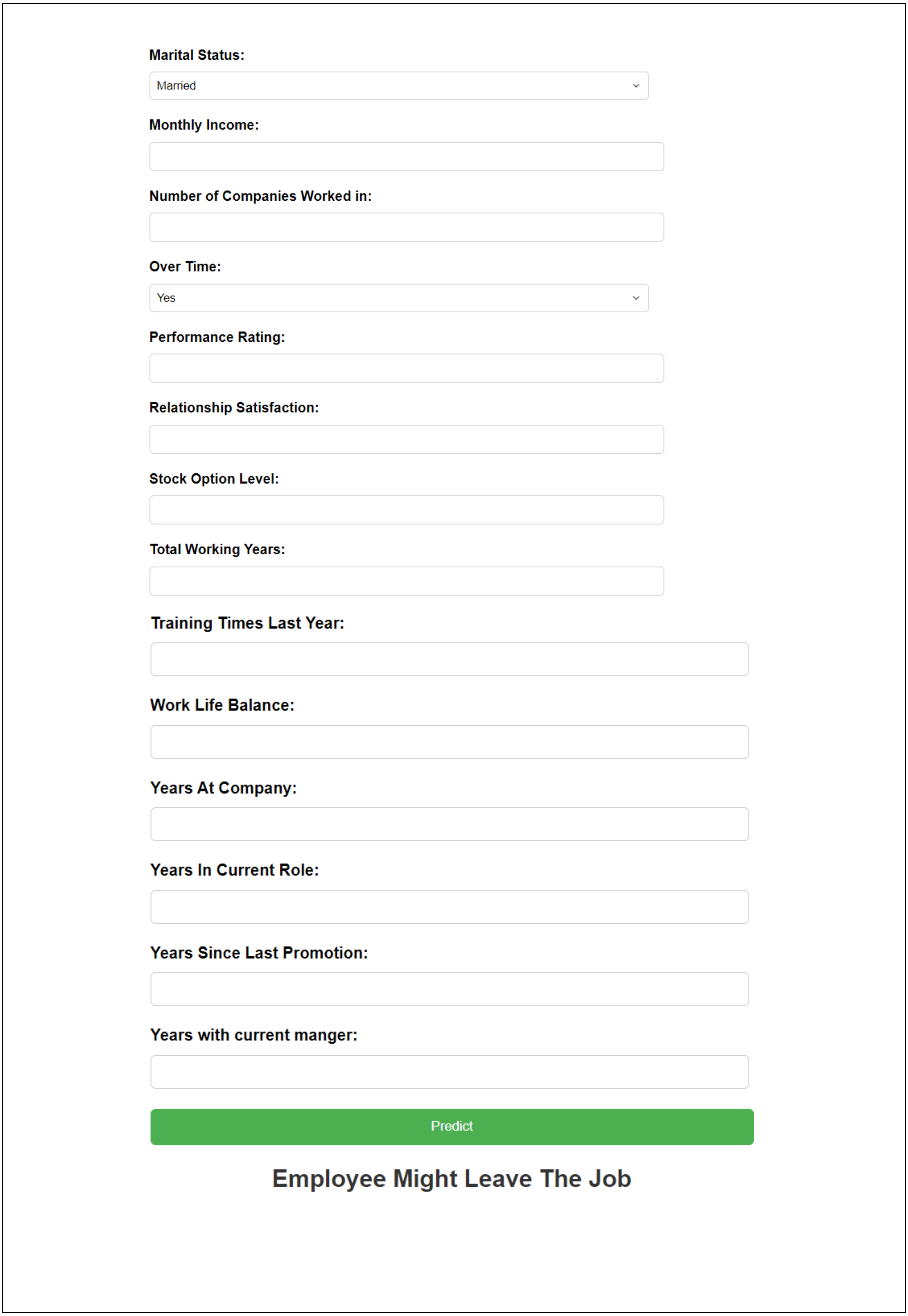
Job Involvement:

Job Level:

Job Role: Laboratory Technician

Job Satisfaction:

Marital Status: Married


Marital Status:

Married

Monthly Income:

Number of Companies Worked in:

Over Time:

Yes

Performance Rating:

Relationship Satisfaction:

Stock Option Level:

Total Working Years:

Training Times Last Year:

Work Life Balance:

Years At Company:

Years In Current Role:

Years Since Last Promotion:

Years with current manger:

Employee Might Leave The Job

CONCLUSION:

IN CONCLUSION, USING MACHINE LEARNING TO MANAGE EMPLOYEE ATTRITION IS A POWERFUL WAY FOR COMPANIES TO TACKLE TURNOVER ISSUES AND BUILD A STRONGER WORKFORCE. BY ANALYZING DATA AND PREDICTING WHICH EMPLOYEES MIGHT LEAVE, ORGANIZATIONS CAN TAKE PROACTIVE STEPS TO KEEP THEM ENGAGED AND SATISFIED. THIS INCLUDES THINGS LIKE PERSONALIZED RETENTION STRATEGIES, IMPROVING HOW THEY RECRUIT AND DEVELOP TALENT, AND CREATING A POSITIVE WORKPLACE CULTURE. NOT ONLY DOES THIS APPROACH HELP SAVE MONEY BY REDUCING THE COSTS OF TURNOVER, BUT IT ALSO FOSTERS A WORK ENVIRONMENT WHERE EMPLOYEES FEEL VALUED AND SUPPORTED. PLUS, BY MAKING SMARTER DECISIONS BASED ON DATA, COMPANIES CAN BETTER PLAN FOR THE FUTURE AND STAY COMPETITIVE IN THEIR INDUSTRY. OVERALL, EMBRACING MACHINE LEARNING IN EMPLOYEE ATTRITION MANAGEMENT LEADS TO HAPPIER EMPLOYEES, STRONGER TEAMS, AND GREATER SUCCESS FOR THE COMPANY.