EMPLOYEE ATTRITION PREDICTION AND HR ANALYTICS

A project report submitted to ICT Academy of Kerala

in partial fulfillment of the requirements

for the certification of

CERTIFIED SPECIALIST

IN

DATA SCIENCE & ANALYTICS

submitted by

Team 3

Members

ADINANDA TK

ARSHAD ARIF M

PRASANTH KV

SREERADHA M

ICT ACADEMY OF KERALA

THIRUVANANTHAPURAM, KERALA, INDIA

Feb 2023

List of Figures

List of Abbreviations

AI	Artificial Intelligence
CSS	Cascading Style Sheets
HR	Human Resource
HTML	Hyper Text Markup Language
KNN	K Nearest Neighbours
K-FOLD	K number of Folds

Table of Contents

Abstract

Employee attrition, the voluntary departure of employees from an organization, poses significant challenges for businesses, impacting productivity, continuity, and overall performance. In this project, we aim to develop a system that can accurately predict employee attrition within an organization or a team and utilize data-driven decision-making processes to identify employees at risk of leaving their positions. To achieve this, we explore the application of HR analytics using a dataset of 1400 rows and 29 columns, encompassing various employee-related attributes such as age, job role, job satisfaction, work-life balance, and performance rating. Our main goal is to build and validate a predictive model capable of identifying potential attrition risks within the workforce. Leveraging machine learning algorithms, including logistic regression, decision trees, random forests, and gradient boosting, we conduct data pre-processing, feature engineering, model training, and performance evaluation. The project's outcomes offer valuable insights empowering HR professionals to proactively identify employees at higher attrition risk, enabling the implementation of targeted retention strategies, such as personalized development plans and improved work-life balance initiatives. The integration of employee attrition prediction and HR analytics holds immense potential in enhancing workforce management, fostering organizational growth, and ensuring overall success.

1. Problem Definition

1.1 Overview

The HR attrition prediction and analytics project aims to address the challenge of employee turnover within an organization. By leveraging data-driven approaches, the project seeks to analyze various factors, such as employee demographics, job satisfaction, work-life balance, and performance ratings, to identify patterns and trends associated with attrition. The goal is to develop a predictive model that can anticipate the likelihood of employees leaving the company, enabling proactive HR strategies. This project holds the potential to provide valuable insights into the key drivers of attrition, enabling the organization to implement targeted retention initiatives, improve workforce planning, and enhance overall employee satisfaction and engagement.

1.2 Problem Statement

The HR attrition prediction and analytics project addresses a critical concern faced by organizations worldwide: the high cost and disruption caused by employee turnover. The project aims to harness the power of data analytics to understand the underlying factors leading to attrition. By examining a comprehensive dataset including variables such as age, job level, job roles, job satisfaction, and work-life balance, the project seeks to uncover patterns, correlations, and predictive indicators of attrition. The primary goal is to develop an accurate and robust predictive model that can forecast the likelihood of an employee leaving the company. This model could become an invaluable tool for HR departments, enabling them to proactively identify at-risk employees, implement targeted retention strategies, and ultimately reduce attrition rates. The insights gained from this project have the potential to revolutionize HR practices, leading to a more stable and motivated workforce.

2. Introduction

Employee attrition, the process of employees voluntarily leaving an organization, has significant implications for businesses, leading to productivity loss, increased recruitment costs, and potential disruptions to company operations. Consequently, the ability to predict employee attrition can be a valuable asset for Human Resources (HR) departments in proactively addressing turnover challenges. In recent years, the advent of HR analytics has enabled organizations to leverage data-driven approaches to predict attrition and make informed decisions in managing their workforce. In this project, we explore the application of HR analytics to predict employee attrition using a dataset consisting of 1400 rows and 29 columns. The dataset contains various employee-related attributes such as age, business travel, department, distance from home, education, education field, gender, job level, job role, marital status, monthly income, number of companies worked, age limit, percent salary hike, standard hours, stock option level, total working years, training times last year, years at the company, years since the last promotion, years with current manager, environment satisfaction, job satisfaction, work-life balance, job involvement, and performance rating.

Our project aims to build and validate a predictive model that can identify potential attrition risks within the workforce. To achieve this, we will employ various machine learning algorithms, such as logistic regression, decision trees, random forests, and gradient boosting, to analyze the dataset. The analysis will involve data preprocessing, feature engineering, model training, and performance evaluation using appropriate metrics.

The outcomes of this study can offer valuable insights to HR professionals, enabling them to proactively identify employees at a higher risk of attrition. With this predictive knowledge, HR departments can implement targeted retention strategies, such as personalized development plans, competitive benefits, and improved work-life balance initiatives, to increase employee satisfaction and reduce attrition rates. Ultimately, the integration of employee attrition prediction and HR analytics can contribute to a more stable and productive workforce, fostering organizational growth and success

3. Literature Survey

[1] IBM HR Analytics Employee Attrition & Performance

The HR attrition prediction and HR analytics project is a comprehensive effort aimed at addressing the critical issue of employee attrition within the organization. High turnover rates not only disrupt operational continuity but also lead to significant financial and knowledge loss. The project seeks to utilize advanced data analysis techniques to uncover underlying factors contributing to employee attrition. By examining a wide range of variables such as age, job role, career development, compensation, and job satisfaction, the project aims to identify correlations and trends that can shed light on why employees leave. The project's primary objective is to develop a robust predictive model that can accurately forecast the likelihood of an employee leaving the company. This model will empower HR professionals and management to take proactive measures to mitigate attrition by designing targeted retention strategies and addressing potential pain points identified in the analysis.

Furthermore, the project will provide insights into the most critical features influencing attrition. Are certain departments experiencing higher attrition? Are there specific job roles with elevated turnover rates? Do employees at certain career levels face more attrition risk? These questions and more will be explored to gain a nuanced understanding of the dynamics behind attrition. Ultimately, the project's outcome will not only enable the organization to retain valuable talent but also enhance workforce planning, improve employee satisfaction, optimize talent acquisition, and bolster overall organizational performance.

4. Exploratory Data Analysis

1. COLUMNS WITH NULL VALUES

- **NumCompaniesWorked**
- **EnviornmentSatisfaction**
- **JobSatisfaction**
- **WorkLifeBalance**

2. GRAPHS

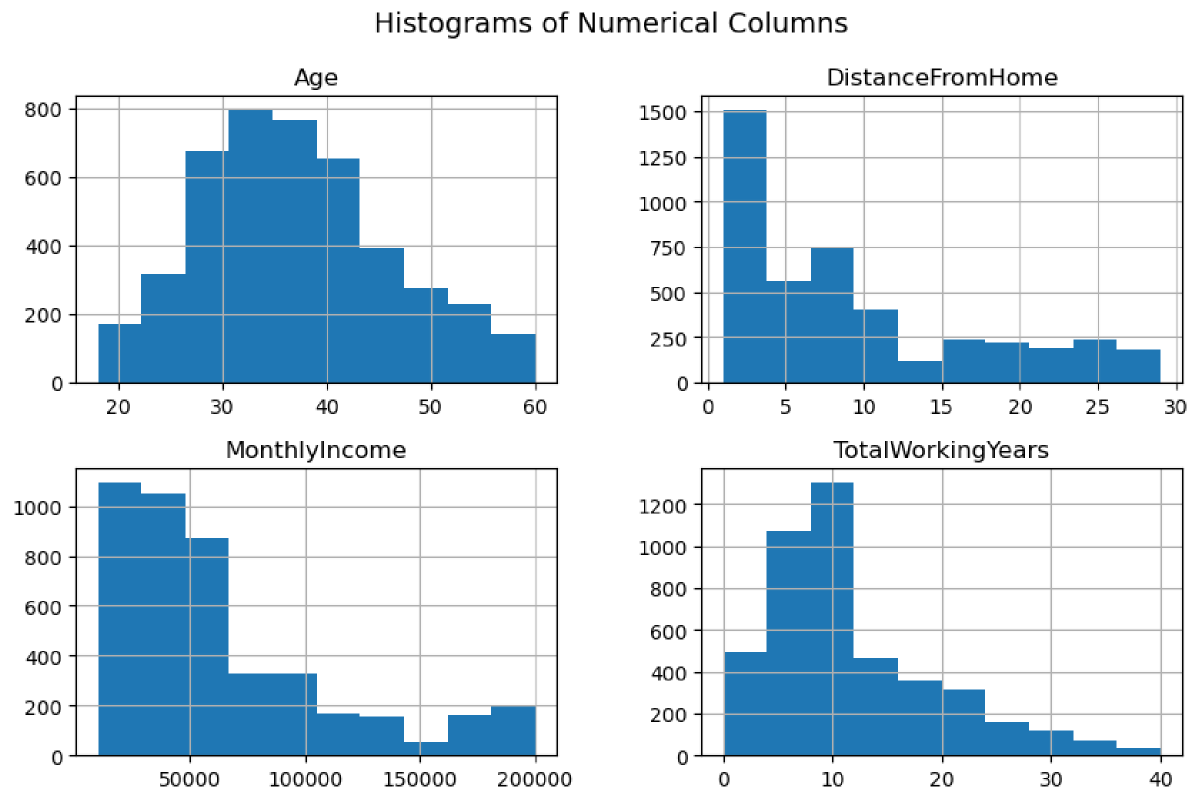

Figure 1: Histograms

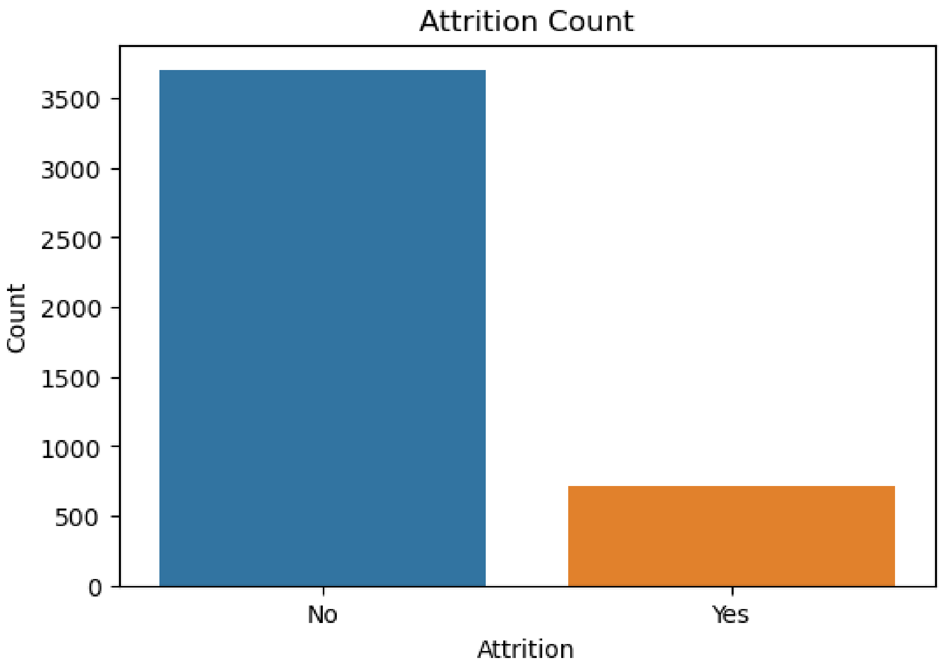

Figure 2: Bar plot for Attrition

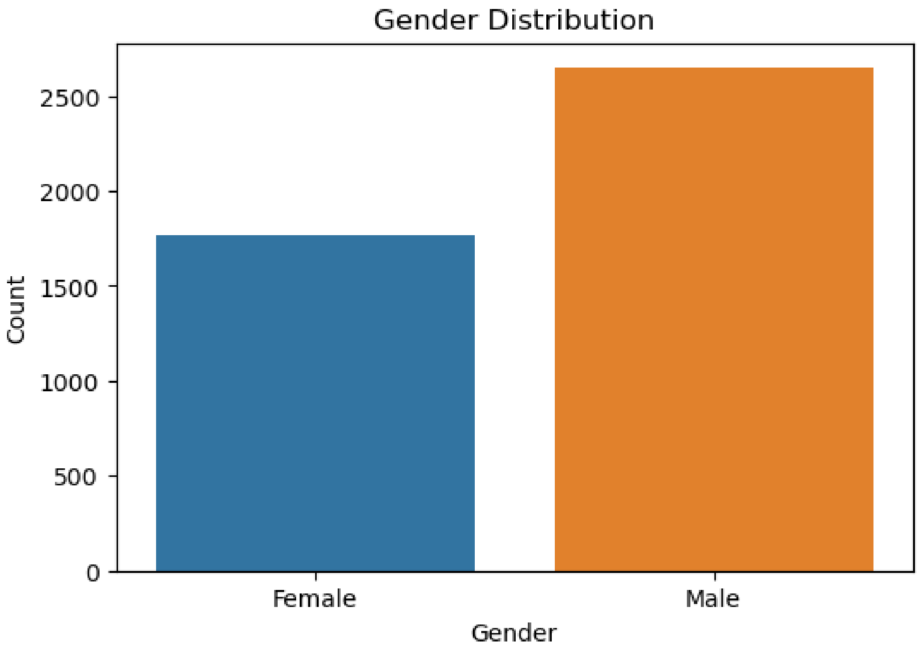

Figure 3: Bar plot for gender

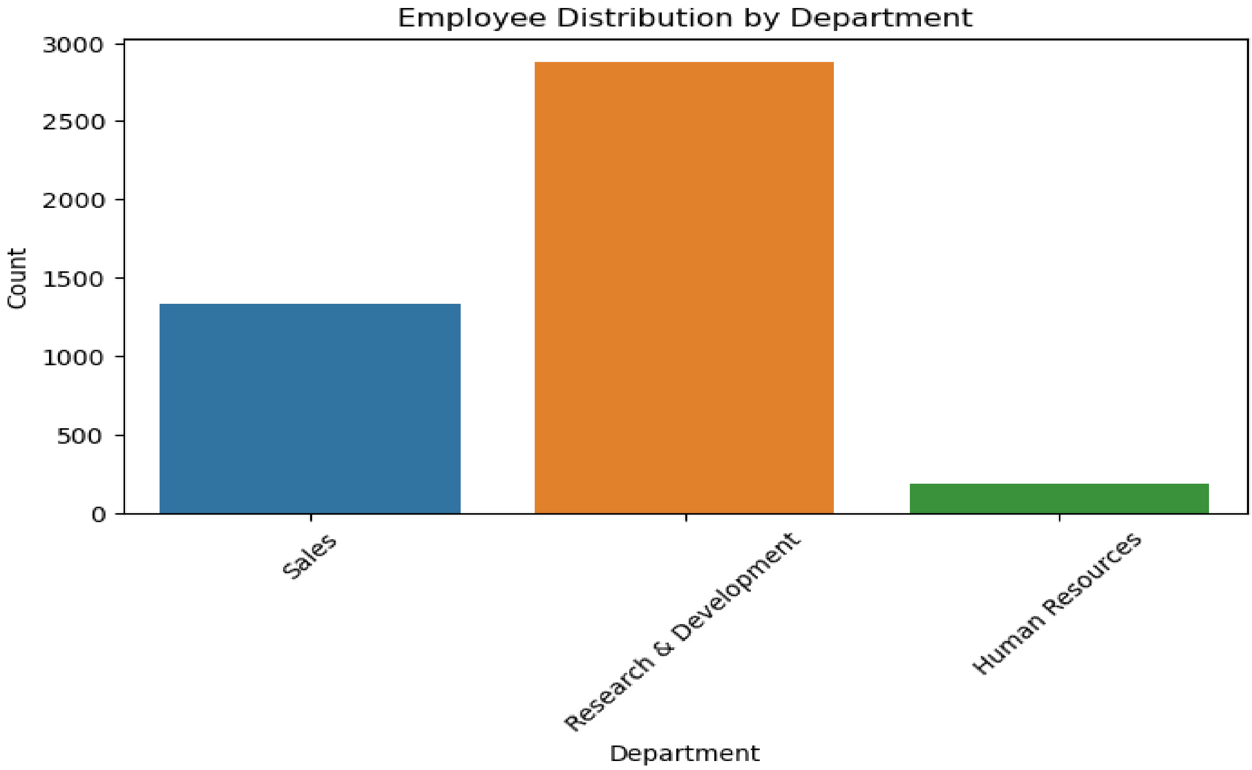

Figure 4: Bar Plots of department

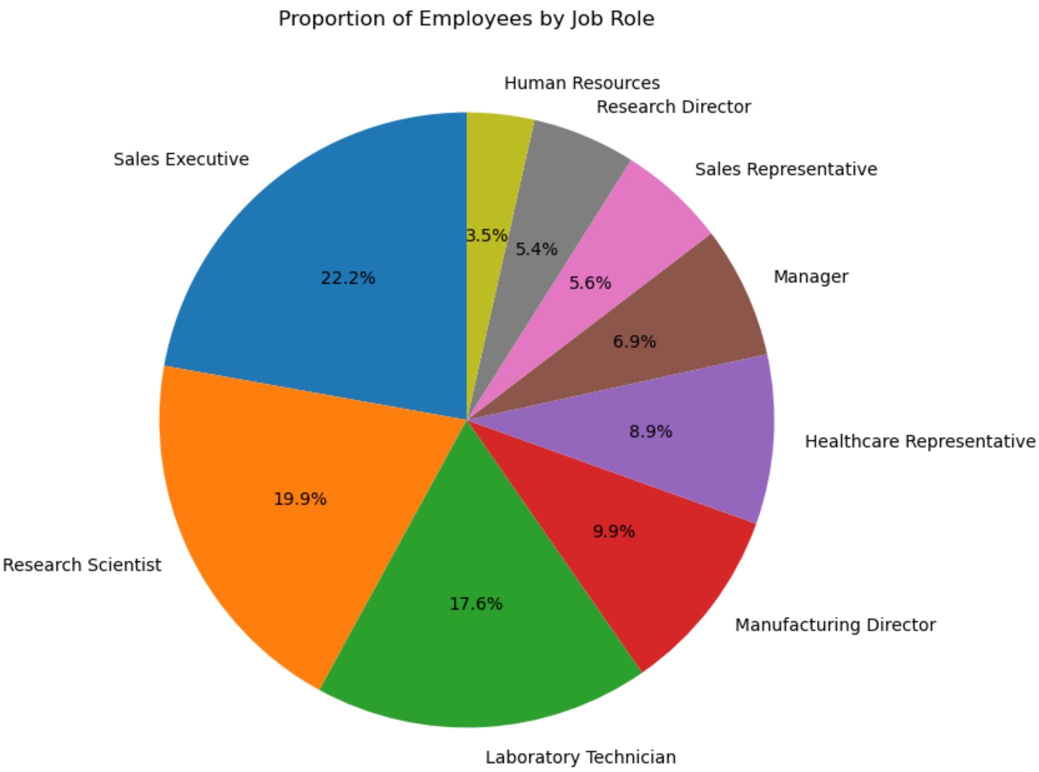

Figure 5: Proportion of employees by job role

Outier detection

```
In [50]:
def detect_outliers_zscore(df):
    z_scores = np.abs((df - df.mean()) / df.std())
    return z_scores > 3  # Adjust the threshold (e.g., 3) based on your dataset and r

outliers_zscore = df.apply(detect_outliers_zscore)
print(outliers_zscore)
```

```
      Unnamed: 0    Age  Attrition  BusinessTravel  Department  \
0          False  False      False           False       False
1          False  False      False           False       False
2          False  False      False           False       False
3          False  False      False           False       False
4          False  False      False           False       False
...          ...    ...        ...             ...         ...
4405       False  False      False           False       False
4406       False  False      False           False       False
4407       False  False      False           False       False
4408       False  False      False           False       False
4409       False  False      False           False       False

      DistanceFromHome  Education  EducationField  EmployeeCount  Gender  ...  \
0                False      False           False          False   False  ...
1                False      False           False          False   False  ...
2                False      False           False          False   False  ...
3                False      False           False          False   False  ...
4                False      False           False          False   False  ...
```

Figure 6: Outlier Detection

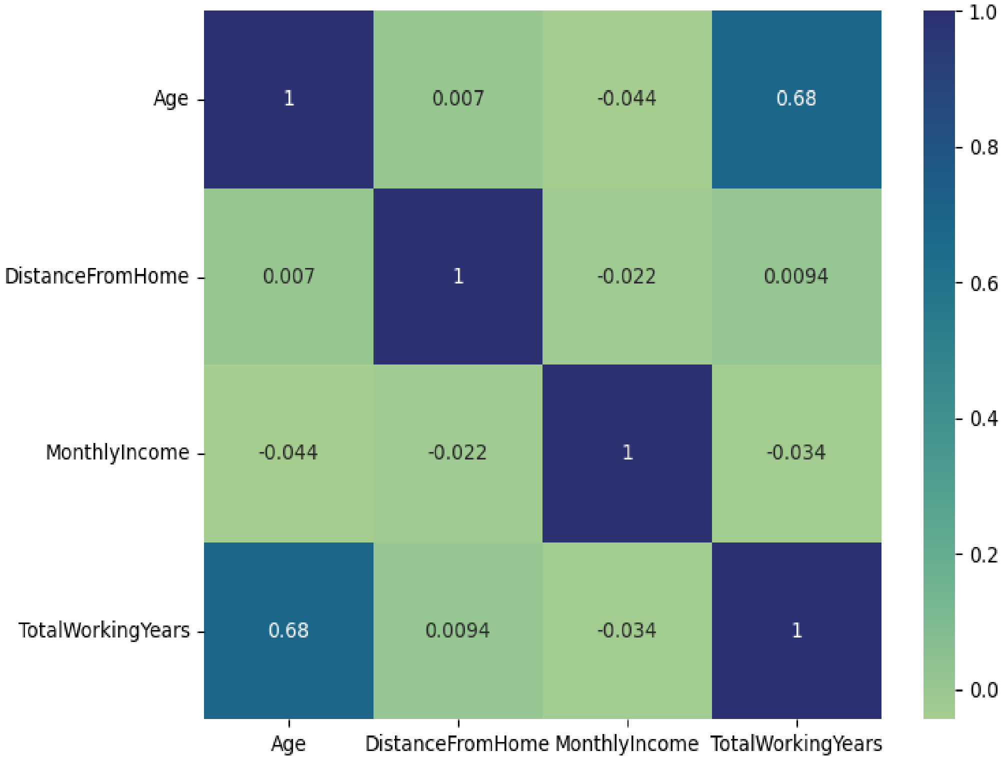

Figure 7: Correlation Heatmap

3. BOX PLOTS

A box plot, also known as a box-and-whisker plot, is a graphical representation of the distribution of a dataset. It provides a visual summary of key statistical measures, such as the median, quartiles, and potential outliers, in a concise and informative manner. Box plots are commonly used to display the spread and central tendency of a dataset, making it easier to understand its overall characteristics.

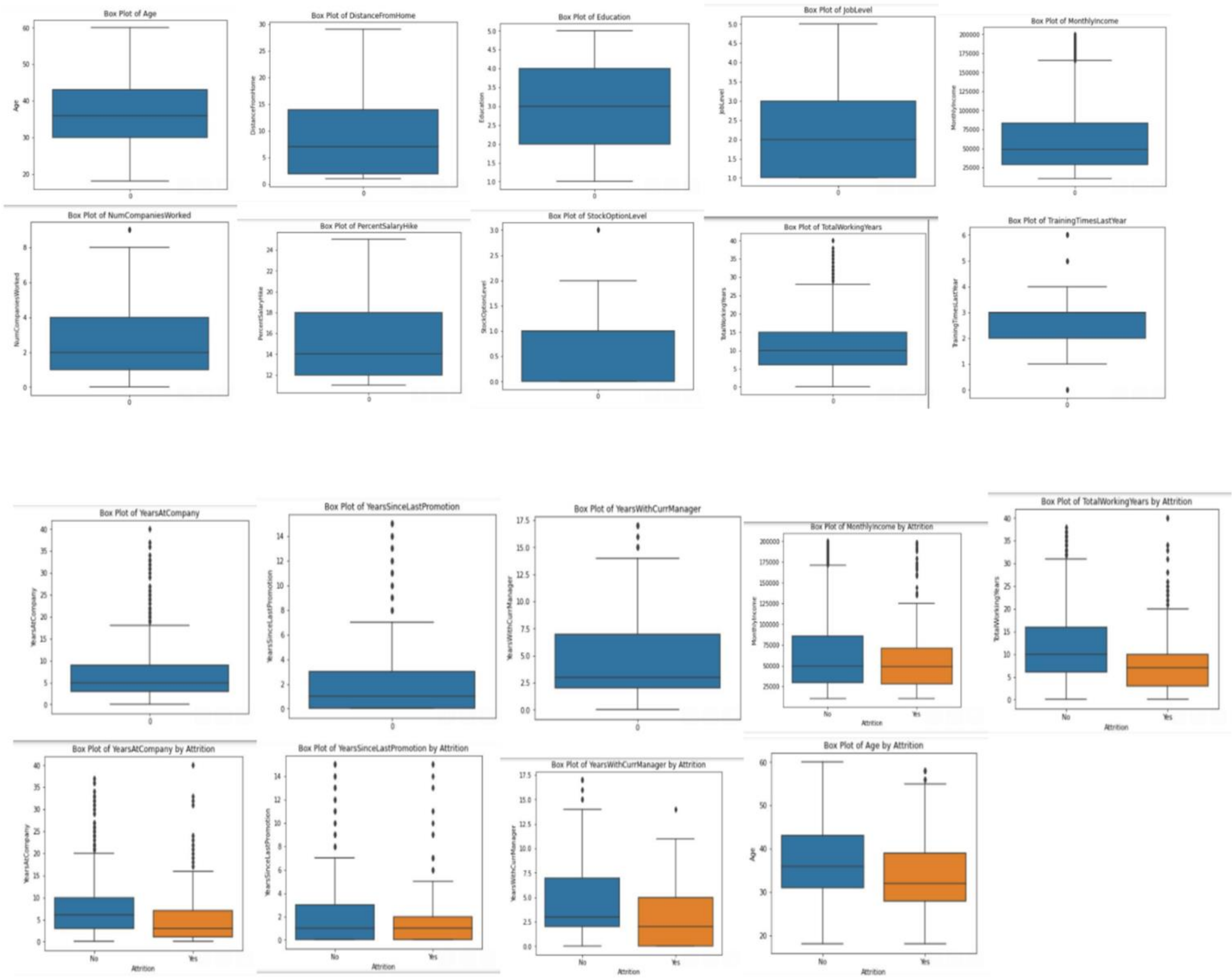

Figure 8 : Box plots

5. Data Preprocessing

1. FILLING MSSING VALUES

- Identify the numerical columns in the DataFrame data that have missing values, based on the list of column names or indices provided in num_col_with_missing_values.
- Calculate the mean value for each of these numerical columns.
- Replace the missing values in these columns with their respective mean values.

```
data.isna().sum()
```

```
Unnamed: 0                  0
Age                         0
Attrition                   0
BusinessTravel              0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
Gender                      0
JobLevel                    0
JobRole                     0
MaritalStatus               0
MonthlyIncome               0
NumCompaniesWorked         19
Over18                      0
PercentSalaryHike           0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           9
TrainingTimesLastYear       0
YearsAtCompany              0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
EnvironmentSatisfaction    25
JobSatisfaction            20
WorkLifeBalance            38
JobInvolvement              0
PerformanceRating           0
dtype: int64
```

```
#Columns with missing values
num_col_with_missing_values = ['NumCompaniesWorked', 'EnvironmentSatisfaction', 'JobSatisfaction', 'WorkLifeBalance']
```

```
#Filling missing values
data[num_col_with_missing_values] = data[num_col_with_missing_values].fillna(data[num_col_with_missing_values].mean())
```

```
data.isna().sum()
```

```
Unnamed: 0                  0
Age                         0
Attrition                   0
BusinessTravel              0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
Gender                      0
JobLevel                    0
JobRole                     0
MaritalStatus               0
MonthlyIncome               0
NumCompaniesWorked          0
Over18                      0
PercentSalaryHike           0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           9
TrainingTimesLastYear       0
YearsAtCompany              0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
EnvironmentSatisfaction     0
JobSatisfaction             0
WorkLifeBalance             0
JobInvolvement              0
PerformanceRating           0
dtype: int64
```

2. SCALING AND NORMALIZATION

```
data[numerical_columns] = scaler.fit_transform(data[numerical_columns])
```

```
data.head()
```

	Unnamed: 0	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Gender	...	TotalWorkingYears	TrainingTimesLastYear	YearsAtCompany	YearsSinceLastPromotion	YearsWithCurrManager	EnvironmentSatisfaction	JobSatisfaction	Wor
0	0	0.785714	No	Travel_Rarely	Sales	0.178571	0.25	Life Sciences	0.0	Female	...	0.025	1.000000	0.025	0.000000	0.000000	0.666667	1.000000	
1	1	0.309524	Yes	Travel_Frequently	Research & Development	0.321429	0.00	Life Sciences	0.0	Female	...	0.150	0.500000	0.125	0.066667	0.235294	0.666667	0.333333	
2	2	0.333333	No	Travel_Frequently	Research & Development	0.571429	0.75	Other	0.0	Male	...	0.125	0.333333	0.125	0.000000	0.176471	0.333333	0.333333	
3	3	0.476190	No	Non-Travel	Research & Development	0.035714	1.00	Life Sciences	0.0	Male	...	0.325	0.833333	0.200	0.466667	0.294118	1.000000	1.000000	
4	4	0.333333	No	Travel_Rarely	Research & Development	0.321429	0.00	Medical	0.0	Male	...	0.225	0.333333	0.150	0.000000	0.235294	1.000000	0.000000	

5 rows × 29 columns

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Attrition'] = le.fit_transform(data['Attrition'])
```

```
data.head()
```

	Unnamed: 0	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Gender	...	TotalWorkingYears	TrainingTimesLastYear	YearsAtCompany	YearsSinceLastPromotion	YearsWithCurrManager	EnvironmentSatisfaction	JobSatisfaction	Wor
0	0	0.785714	0	Travel_Rarely	Sales	0.178571	0.25	Life Sciences	0.0	Female	...	0.025	1.000000	0.025	0.000000	0.000000	0.666667	1.000000	
1	1	0.309524	1	Travel_Frequently	Research & Development	0.321429	0.00	Life Sciences	0.0	Female	...	0.150	0.500000	0.125	0.066667	0.235294	0.666667	0.333333	
2	2	0.333333	0	Travel_Frequently	Research & Development	0.571429	0.75	Other	0.0	Male	...	0.125	0.333333	0.125	0.000000	0.176471	0.333333	0.333333	
3	3	0.476190	0	Non-Travel	Research & Development	0.035714	1.00	Life Sciences	0.0	Male	...	0.325	0.833333	0.200	0.466667	0.294118	1.000000	1.000000	
4	4	0.333333	0	Travel_Rarely	Research & Development	0.321429	0.00	Medical	0.0	Male	...	0.225	0.333333	0.150	0.000000	0.235294	1.000000	0.000000	

5 rows × 29 columns

```
[ ] data.dtypes

Unnamed: 0                   int64
Age                        float64
Attrition                    int64
BusinessTravel              object
Department                  object
DistanceFromHome           float64
Education                  float64
EducationField              object
EmployeeCount              float64
Gender                      object
JobLevel                   float64
JobRole                     object
MaritalStatus               object
MonthlyIncome              float64
NumCompaniesWorked         float64
Over18                      object
PercentSalaryHike          float64
StandardHours              float64
StockOptionLevel           float64
TotalWorkingYears          float64
TrainingTimesLastYear      float64
YearsAtCompany             float64
YearsSinceLastPromotion    float64
YearsWithCurrManager       float64
EnvironmentSatisfaction    float64
JobSatisfaction            float64
WorkLifeBalance            float64
JobInvolvement             float64
PerformanceRating          float64
dtype: object
```

3. FEATURE SELECTION

```
[ ] data['Age'].max()

    0.9999999999999998
```

Since ages are of the range 30 to 55 we are assuming that no one retiers in ths dataset, we are removing the age column

```
[ ] data = data.drop('Age', axis=1)

[ ] data['BusinessTravel'].unique()

    array(['Travel_Rarely', 'Travel_Frequently', 'Non-Travel'], dtype=object)

[ ] data['Department'].unique()

    array(['Sales', 'Research & Development', 'Human Resources'], dtype=object)

    data['JobRole'].unique()

    array(['Healthcare Representative', 'Research Scientist',
           'Sales Executive', 'Human Resources', 'Research Director',
           'Laboratory Technician', 'Manufacturing Director',
           'Sales Representative', 'Manager'], dtype=object)

[ ] data['MaritalStatus'].unique()

    array(['Married', 'Single', 'Divorced'], dtype=object)

[ ] data['Over18'].unique()

    array(['Y'], dtype=object)

[ ] data['MaritalStatus'].unique()

    array(['Married', 'Single', 'Divorced'], dtype=object)

[ ] cat=['Gender','JobRole','MaritalStatus','Over18','BusinessTravel','Department']
    for i in cat:
        data[i]=le.fit_transform(data[i])
```

6 . Machine Learning Models & Flask App

1. **KNN**

```
[ ] from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.metrics import accuracy_score

[ ] # Handle missing values using imputation
    from sklearn.impute import SimpleImputer
    imputer = SimpleImputer(strategy='mean')
    data = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)

[ ] # Select features and target
    X = data.drop('Attrition', axis=1)  # Features
    y = data['Attrition']  # Target

[ ] # Split the dataset 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)

    # Initialize the MinMaxScaler
    scaler = MinMaxScaler()

[ ] # Apply Min-Max scaling to features
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

[ ] # Initialize the KNN classifier
    knn_classifier = KNeighborsClassifier(n_neighbors=5)

[ ] # Train the KNN classifier
    knn_classifier.fit(X_train_scaled, y_train)

    ▾ KNeighborsClassifier
    KNeighborsClassifier()

[ ] # Predictions on the test set
    y_pred = knn_classifier.predict(X_test_scaled)

[ ] # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)

[ ] print("Accuracy:", accuracy)

    Accuracy: 0.854875283446712
```

We got the accuracy of 0.85 in KNN Classifier

2. LOGISTIC REGRESSION

```
[ ] from sklearn.linear_model import LogisticRegression

[ ] lr = LogisticRegression()

[ ] model = LogisticRegression(max_iter=1000)  # Increase max_iter value
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

[ ] from sklearn.metrics import confusion_matrix

[ ] confusion_matrix(y_test,y_pred)

    array([[736,   5],
           [129,  12]])

[ ] from sklearn.metrics import precision_score,recall_score,f1_score,accuracy_score

[ ] print(precision_score(y_test,y_pred))

    0.7058823529411765

[ ] print(recall_score(y_test,y_pred))

    0.0851063829787234

[ ] print(f1_score(y_test,y_pred))

    0.1518987341772152

[ ] print(accuracy_score(y_test,y_pred))

    0.8480725623582767
```

We got the accuracy of 0.84 in Logistic Regression

3. DESCISION TREE

Importing the required modules

```
[ ] from sklearn.tree import DecisionTreeClassifier
    from sklearn.metrics import accuracy_score
```

Create a Decision Tree classifier with a specific random state

```
[ ] decision_tree = DecisionTreeClassifier(random_state=21)
```

Fit the classifier to your training data

```
decision_tree.fit(X_train, y_train)

    DecisionTreeClassifier
DecisionTreeClassifier(random_state=21)
```

Make the predictions

```
[ ] y_pred = decision_tree.predict(X_test)
```

Calculate and print the accuracy of the model

```
[ ] accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy:", accuracy)

    Accuracy: 0.9841269841269841
```

Descision tree accuracy = 0.98

4. RANDOM FOREST

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score
```

```
random_forest = RandomForestClassifier(random_state=21)
random_forest.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=21)
```

```
[ ] y_pred = random_forest.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    Accuracy: 0.99
```

```
accuracy

0.9875283446712018
```

We got the accuracy of 0.9875 in Random Forest

Hyperparameter Tuning

A. Using Manual Hyperparameter Tuning

```
[ ] hyp_classifier = RandomForestClassifier(n_estimators = 300,criterion = 'entropy',max_features = 'sqrt',min_samples_leaf=10,random_state=100).fit(X_train,y_train)
```

```
[ ] y_pred_2 = hyp_classifier.predict(X_test)
    print('Accuracy:',accuracy_score(y_test,y_pred_2))

    Accuracy: 0.8594104308390023
```

We got the accuracy of 0.85

B. Using RandomizedSearchCV

```
[ ] from sklearn.model_selection import RandomizedSearchCV
```

```
[ ] np.linspace(start=100,stop=500,num=30)

    array([100.        , 113.79310345, 127.5862069 , 141.37931034,
           155.17241379, 168.96551724, 182.75862069, 196.55172414,
           210.34482759, 224.13793103, 237.93103448, 251.72413793,
           265.51724138, 279.31034483, 293.10344828, 306.89655172,
           320.68965517, 334.48275862, 348.27586207, 362.06896552,
           375.86206897, 389.65517241, 403.44827586, 417.24137931,
           431.03448276, 444.82758621, 458.62068966, 472.4137931 ,
           486.20689655, 500.        ])
```

```
[ ] n_estimators = [int(x) for x in np.linspace(start=200,stop=2000,num=10)]
    max_features = ['auto','sqrt','log2']
    max_depth = [int(x) for x in np.linspace(start=10,stop=1000,num=10)]
    min_samples_split = [2,5,10,14]
    min_samples_leaf = [1,2,4,6,8]
    random_grid ={'n_estimators':n_estimators,
                  'max_features' :max_features,
                  'max_depth' :max_depth,
                  'min_samples_split':min_samples_split,
                  'min_samples_leaf':min_samples_leaf,
                  'criterion' : ['entropy','gini','log_loss']}
```

```
[ ] rf = RandomForestClassifier()
```

```
[ ] rf_randomCV = RandomizedSearchCV(estimator=rf,param_distributions=random_grid,n_iter=100,cv=3,verbose=2,random_state=100,n_jobs=-1)
```

```
[ ] #fit the randomized model
    rf_randomCV.fit(X_train,y_train)

    Fitting 3 folds for each of 100 candidates, totalling 300 fits
    ▸ RandomizedSearchCV
    ▸ estimator: RandomForestClassifier
        ▸ RandomForestClassifier
```

```
[ ] rf_randomCV.best_params_

    {'n_estimators': 1400,
     'min_samples_split': 2,
     'min_samples_leaf': 1,
     'max_features': 'sqrt',
     'max_depth': 120,
     'criterion': 'entropy'}
```

```
rf_randomCV.best_estimator_

    ▾ RandomForestClassifier
    RandomForestClassifier(criterion='entropy', max_depth=120, n_estimators=1400)
```

```
[ ] best_random = rf_randomCV.best_estimator_
```

```
[ ] y_pred_3 = best_random.predict(X_test)
    print('Accuracy:',accuracy_score(y_pred_3,y_pred_3))

    Accuracy: 1.0
```

We got the accuracy of 0.99

5. PERFORMING K-FOLD CROSS VALIDATION (Random Forest)

```python
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score

kf = KFold(n_splits=5, shuffle=True, random_state=42)
accuracy_scores = []
for fold_num, (train_index, test_index) in enumerate(kf.split(X), start=1):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Fit the hyperparameter-tuned model on the training data
    best_random_forest.fit(X_train, y_train)

    # Predict on the test data
    y_pred = best_random_forest.predict(X_test)

    # Calculate accuracy and store it
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

    print(f"Fold {fold_num} - Accuracy: {accuracy}")
```

```
Fold 1 - Accuracy: 0.9920634920634921
Fold 2 - Accuracy: 0.9954648526077098
Fold 3 - Accuracy: 0.9931972789115646
Fold 4 - Accuracy: 0.9920634920634921
Fold 5 - Accuracy: 0.9931972789115646
```

6 . DEPLOYING FLASK APP

web.py

```python
from flask import Flask, render_template, request
import pickle
import numpy as np

app = Flask(__name__)

# Load the trained model
with open(r'C:\Users\USER\Desktop\DSA course\HR AV hacakthon\Web app\best_random_forest_model1.pkl',
    model = pickle.load(model_file)

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

@app.route('/predict', methods=['POST'])
def predict():
    features = [
        'BusinessTravel', 'Department', 'DistanceFromHome', 'Education',
        'EducationField', 'EmployeeCount', 'Gender', 'JobLevel', 'JobRole',
        'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked',
        'PercentSalaryHike', 'StandardHours', 'StockOptionLevel',
        'TotalWorkingYears', 'TrainingTimesLastYear', 'YearsAtCompany',
        'YearsSinceLastPromotion', 'YearsWithCurrManager',
        'EnvironmentSatisfaction', 'JobSatisfaction', 'WorkLifeBalance',
        'JobInvolvement', 'PerformanceRating'
    ]

    input_data = [request.form[feature] for feature in features]
    input_data = np.array(input_data, dtype=float).reshape(1, -1)

    prediction = model.predict(input_data)
    if prediction[0] == 0:
        result = "No"
    else:
        result = "Yes"

    return render_template('result.html', prediction_text=f"Predicted Attrition: {result}")

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

Here we export the Random Forest Classifier which gives 0.99 accuracy as a pickle file and loads that into the flask app

7 . HTML & CSS

```html
</style>
<body>
  <h1>Predict Attrition</h1>
  <form action="{{ url_for('predict') }}" method="POST">

    <label for="BusinessTravel">Business Travel  [2=Rarely ,1=Frequently ,0=No travel]</label>
    <input type="number" name="BusinessTravel" required><br><br>

    <label for="Department">Department  [ 2=Sales, 1=R&D, 0=HR]</label>
    <input type="text" name="Department" required><br><br>

    <label for="DistanceFromHome">DistanceFromHome [in Km's]</label>
    <input type="number" name="DistanceFromHome" required><br><br>

    <label for="Education">Education [1,2,3,4,5]</label>
    <input type="number" name="Education" required><br><br>

    <label for="EducationField">EducationField ['Healthcare Representative': 1, 'R&D Scientist': 2,
    <input type="number" name="EducationField" required><br><br>

    <label for="Gender">Gender [Female : 0, Male 1]</label>
    <input type="number" name="Gender" required><br><br>
```

A Snippet of HTML code for index page

```html
</style>
<body>
  <h1>Prediction Result</h1>
  <p>{{ prediction_text }}</p>
  <h1>
<br>
<br>
<br>
<br>
<br>

  </h1>
  <h3>
    Project Team <br>
    - Adinanda TK <br>
    - Arshad Arif <br>
    - Prasanth KV <br>
    - Sreeradha M <br>
  </h3>
  <h3>
    Project made as a part of the CERTIFIED SPECIALIST IN DATA SCIENCE & ANALYTICS COURSE
  </h3>
  <h4>
    The model is trained on Random Forest Classification and has an overall accuracy of 0.9920634920634921
  </h4>
```

A snippet of HTML code for the result page

7. Result

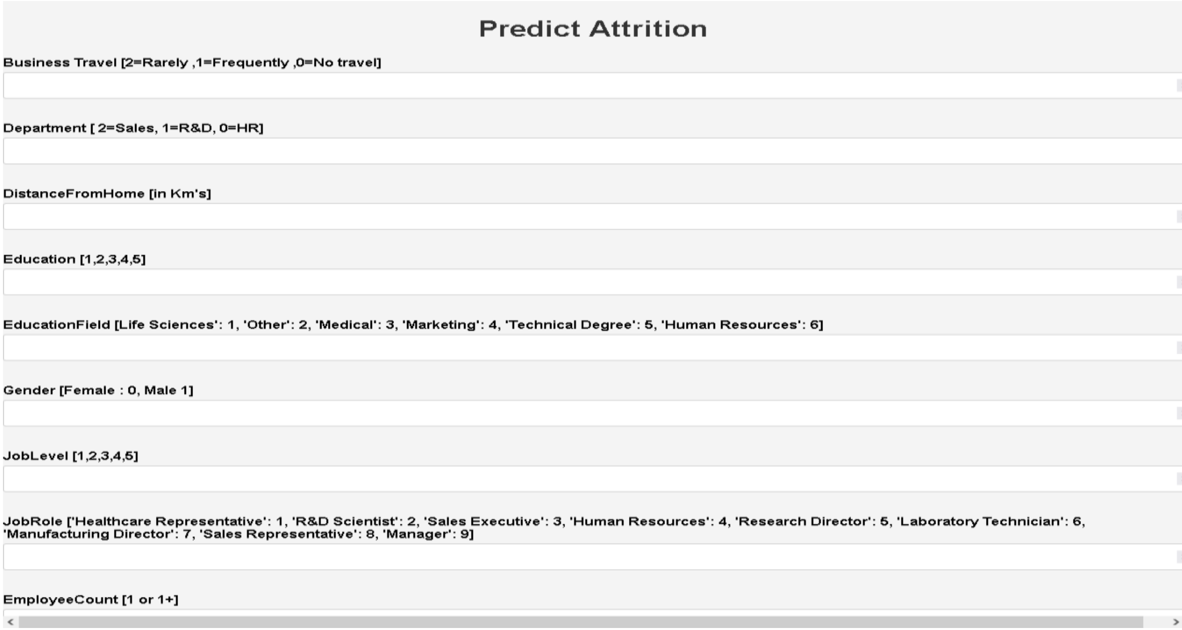

Fig 9: Index.html 1

Fig 10 : Index.html 2

Fig 11: Result page

Published website: https://hrattritionteam3.pythonanywhere.com/

Test Cases	Expected output	Output that the model predicted
Test case 1	No	No
Test case 2	Yes	No
Test case 3	No	No
Test case 4	Yes	No
Test case 5	No	No
Test case 6	No	No
Test case 7	No	No
Test case 8	No	No
Test case 9	Yes	No
Test case 10	No	No

Table 1: Predictions on External Data

Test cases from the dataset	Expected output	Output that the model predicted
Test case 1	Yes	Yes
Test case 2	No	No
Test case 3	No	No
Test case 4	Yes	Yes
Test case 5	No	No
Test case 6	Yes	Yes
Test case 7	No	No
Test case 8	No	No
Test case 9	Yes	Yes
Test case 10	Yes	Yes

Table 2: Predictions on Internal data

The model is specifically trained on a company specific dataset hence the values like *JobSatisfaction*, *WorkLifeBalance* etc are company specific , hence we conclude that the model gives very good accuracy on internal data inputs and does not work as intended and has a bias towards ‘NO’ when tested with external data.

8. Conclusion

The developed system for predicting employee attrition and employing data-driven decision-making offers valuable insights and proactive measures to tackle retention challenges within the organization or team.
By leveraging historical employee data, the system enables the implementation of targeted strategies that can reduce attrition rates and improve overall workforce management, leading to a more stable and productive work environment.

References

[1] https://www.geeksforgeeks.org/ibm-hr-analytics-employee-attrition-performance-using-knn/

[2] https://ieee-dataport.org/documents/ibm-hr-analytics-employee-attrition-performance