OLUBUNMI ELIZABETH ALBERT-ABU

MODULE NAME	DATA AND DECISION MAKING
DATE	**29TH AUGUST, 2023**

DECLARATION
I certify that this assessment submission is entirely my work and I have fully referenced and correctly cited the work of others, where required. I also confirm the contents of my submission have not been generated by a third party, or through an Artificial Intelligence generative system.

TABLE OF CONTENT

Contents

EXECUTIVE SUMMARY

This report discusses the importance of making data-driven decision making through data analytics. Employee attrition which is the departure of employees from an organization poses challenges to businesses as it affects productivity, costs and morale. To address this issue, companies should turn completely to data-driven approaches for insights into attrition patterns and factors driving employee turnover. By analysing various variables such as job satisfaction, compensation etc., decision-makers within the organization can gain a deeper understanding of factors influencing attrition.

Machine learning algorithms including Logistic Regression and K-Nearest Neighbours offers predictive analytics that helps in identifying staffs at risk of leaving. The process of data analytics includes data preprocessing, explorative data analysis, feature selection, model training and evaluation. The train-test split method and K-fold cross validation were used as techniques for accurate model evaluation.

Incorporating data analytics into attrition analysis process empowers organization to make informed decisions and create targeted strategies towards employee retention, thereby, positioning the company to ensure great working conditions, reduce turnover rates and optimize workforce management strategies.

DATA ANALYSIS REPORT ON EMPLOYEE ATTRITION IN SERVICES INDUSTRY OF UNITED KINGDOM

1.0 INTRODUCTION

In business operations, the strategic utilization of data-driven decision making has emerged as a pivotal driver of success across diverse industries. Data analytics offers a paradigm shift in addressing the complexities of diverse business problems in any country's economy.

This report delves into the profound implications of data analytics in the context of a pervasive issue within the service industry in United Kingdom (UK): employee attrition. By examining the interplay between data analytics and the challenge of employee attrition, I aim to illustrate the importance of the potential application of data-driven decision making in addressing this concern within the UK's service sector. Traditional approaches rely on generic strategies that may not account for industry-specific nuances and multifaceted factors contributing to employee turnover. However, by integrating data analytics into the decision-making process, organizations can gain actionable insights into attrition patterns, identify potential risk factors and design targeted retention strategies.

1.1 EMPLOYEE ATTRITION

Employee attrition is the phenomenon when employees voluntarily or involuntarily leave an organisation, thereby requiring replacement. Voluntary attrition occurs when employees leave an organisation on their own accord based on different factors such as finding a better job opportunity or personal reasons, while involuntary attrition occurs beyond the employee's control such as layoffs or terminations, and could be based on factors such as restructuring or downsizing. High attrition rate can disrupt team dynamics, delay service delivery or project completion, loss of skilled personnel, reduce workforce morale, increase recruitment and operational cost amongst other detrimental impacts. Addressing this business problem requires a comprehensive understanding of the underlying factors causing attrition.

1.2 EMPLOYEE ATTRITION IN THE SERVICE INDUSTRY OF UNITED KINGDOM

The service sector in the United Kingdom is a significant driver of the country's economy. It encompasses a diverse range of industries that provides intangible goods and services rather than physical products such as retail, financial, healthcare, hospitality, telecommunications and more. According to Office for National Statistics (2023), the service sector is the largest contributor which makes up approximately 80% of the United Kingdom Gross Domestic Product (GDP) and has recovered strongly from 2008 recession and COVID-19 pandemic. The services sector plays a crucial role in generating economic growth, employment and innovation. As published by House of Commons Library (2023), a key indicator for the services industry contribution to the country's Gross Value-Added economic output is it provided 83% of employment in January – March, 2023.

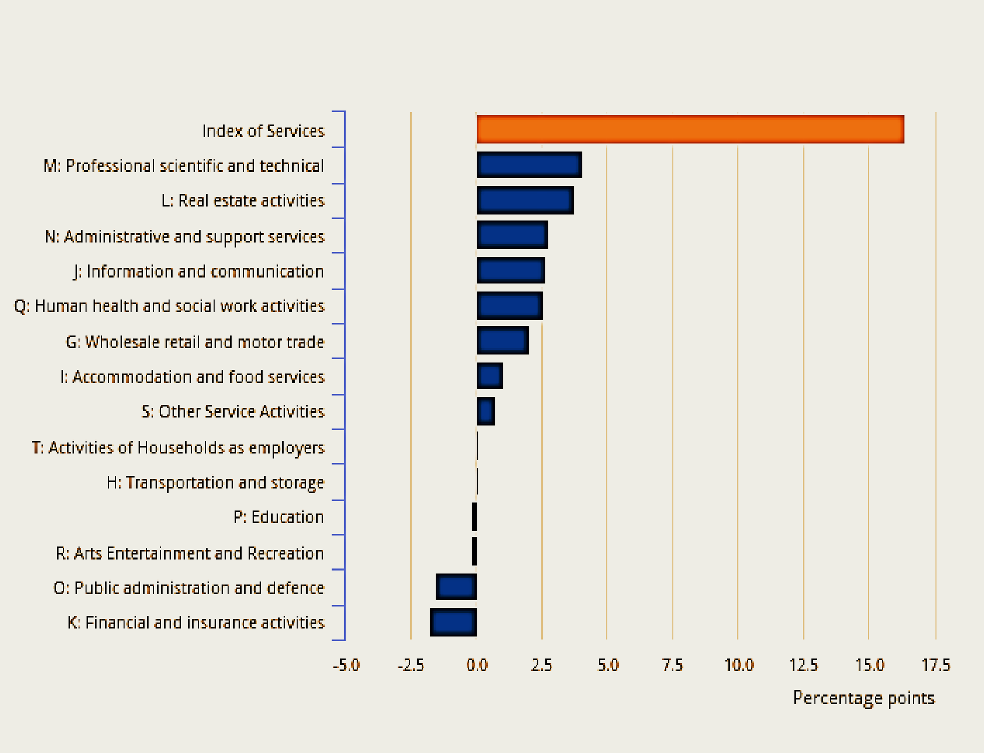


Figure 1: The index of services growth from 2008 – 2023

Source: Office for National Statistics

In recent years, high employee attrition rates were observed in the services industry due to the UK's official exit from the European Union (Brexit) in January 31, 2020 which had implications for the various aspects of the service sector such as increasing skilled worker shortages. A study by the Think Tanks Centre for European Reform and UK in a Changing Europe (2023) suggests that there are 330,000 fewer workers in the UK as a result of Brexit. Sectors such as transport, hospitality and retail were gravely affected. Furthermore, COVID-19 pandemic also led to complex changes in employment dynamics. Chris Stokel-Walker (2020) said "Employee Attrition can be a difficult challenge for most businesses, but the coronavirus pandemic and the chaos it has wrought on workplaces compounds issues of staff retention." The pandemic led to work-life balance evaluation by employees which made some resign. Some others were laid-off due to downsizing, widespread adoption of remote work and effects of rampant business closures.

2.0 EMPLOYEE ATTRITION AS AN EXCELLENT DATA-DRIVEN DECISION-MAKING PROBLEM

Some of the ways employee attrition proves to be an excellent data-driven decision-making problem are outlined below:

1. **Descriptive analytics:** by benchmarking attrition rates to industry standards and competitors, organizations can measure their performance and identify areas of improvement.
2. **Prescriptive analytics:** evidence-based decision-making ensures that actions are based on objective information on attrition rather than assumptions, leading to more effective strategies or outcome.
3. **Predictive analytics:** with the providence of historical data, organizations can develop predictive models that identifies attrition. This knowledge can help to identify and implement retention strategies such as offering growth opportunities, conducive working conditions and more.
4. **Cost and productivity implications:** reduced productivity during the re-hiring phase can diminish the company's overall efficiency. By analysing data relating to attrition, patterns can be identified and preventive measures can be established to mitigate its impact.
5. **Legal and Compliance considerations:** potential disparities that can raise legal and compliance concerns. This can be identified and proactively addressed.

3.0 ANALYZING HUMAN RESOURCE DATA FOR IMPROVED WORKFORCE MANAGEMENT

Human Resource (HR) analytics is a data-driven approach to managing human resources. It involves obtaining and analysing data related to employees to derive insight and make informed decisions.

I will be using the HR Analytics dataset created in July 2023 by Bhanupratap Biswas, a Data Scientist and Kaggle Master. This dataset explores the application of HR Analytics in a hypothetical medium-sized organisation called Tech Solutions Inc. and showcases its benefit in improving workforce management. The company specializes in software development and has a diverse workforce across different departments including Sales, Research & Development and Human Resources. The dataset was created with the objective to understand the factors influencing employee attrition, identify key predictors of employee performance and develop strategies to improve employee retention.

3.1 VARIABLES WITHIN THE DATASET

In this dataset, there are different variables which will help in my analysis:

1. **Categorical variables:** these variables represent categories. The categorical nominal variables are department, job role, education, gender and marital status.
2. **Numerical variables:** these variables are expressed as either **continuous** – within a certain range but with infinite possible values, or as **discrete** – distinct values within a defined range. The continuous numerical variables are age, hourly rate, monthly income, years at company, years in current role, years with current manager and years since last promotion. The discrete numerical variables are job satisfaction and standard hours.
3. **Binary variables:** these are a type of categorical variables with only two variables as options. The binary variables are attrition and the 'over 18' question.
4. **Derived variable:** these are variables calculated from other variables in the dataset. 'Percent salary hike' is the only derived variable.

3.2 KAGGLE

This dataset was gotten from Kaggle, a data science competition platform and online community of data scientists and machine learning enthusiasts under Google LLC (Wikipedia). Kaggle is the world's largest data science community with powerful tools and resources to help you achieve your data science goals.

3.3 DATA CLEANING WITH PYTHON

To properly conduct my analysis, it is necessary to explore my dataset and clean it to ensure accurate predictions are made.

- First, I imported my google drive into my Colaboratory.

```
[ ] #DATA EXPLORATION
```

```
from google.colab import drive
drive.mount('/content/drive')
#My Google Drive was mounted
```

```
Mounted at /content/drive
```

```
[ ] from google.colab import drive
drive.mount('/content/drive')
path = "/content/HR-Employee Attrition.csv"
#My dataset csv file has been created as a path in this code
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ] import pandas as pd
print (path)
path ="/content/drive/My Drive/DDM/HR-Employee Attrition.csv"
df=pd.read_csv(path)
```

```
/content/HR-Employee Attrition.csv
```

- Next, I identified the attributes of the dataset.

```
df.describe
```

```
1466   39        No  Research & Development        Medical    Male
1467   27        No  Research & Development  Life Sciences    Male
1468   49        No                   Sales        Medical    Male
1469   34        No  Research & Development        Medical    Male

      HourlyRate                  JobRole  JobSatisfaction MaritalStatus  \
0           94.0          Sales Executive                4        Single
1           61.0       Research Scientist                2       Married
2           92.0    Laboratory Technician                3        Single
3           56.0       Research Scientist                3       Married
4           40.0    Laboratory Technician                2       Married
...          ...                      ...              ...           ...
1465        41.0    Laboratory Technician                4       Married
1466        42.0  Healthcare Representative              1       Married
1467        87.0   Manufacturing Director                2       Married
1468        63.0          Sales Executive                2       Married
1469        82.0    Laboratory Technician                3       Married

      MonthlyIncome  Over18  PercentSalaryHike  StandardHours  \
0              5993       Y                 11             80
1              5130       Y                 23             80
2              2090       Y                 15             80
3              2909       Y                 11             80
4              3468       Y                 12             80
...             ...     ...                ...            ...
1465           2571       Y                 17             80
1466           9991       Y                 15             80
```

```
[7] df.dtypes
```

```
Age                          int64
Attrition                   object
Department                  object
EducationField              object
Gender                      object
 HourlyRate                float64
JobRole                     object
JobSatisfaction              int64
MaritalStatus               object
 MonthlyIncome               int64
Over18                      object
PercentSalaryHike            int64
StandardHours                int64
YearsAtCompany               int64
YearsInCurrentRole           int64
YearsSinceLastPromotion      int64
YearsWithCurrManager         int64
dtype: object
```

- Below is the statistical summary of numerical columns.

```
df.describe()
```

1 to 8 of 8 entries Filter

Age	HourlyRate	JobSatisfaction	MonthlyIncome	PercentSalaryHike	StandardHours	YearsAtCompany	YearsInCurrentRole	YearsSinceLastPromotion	YearsWithCurrManager
1470.0	1468.0	1470.0	1470.0	1470.0	1470.0	1470.0	1470.0	1470.0	1470.0
36.923809523809524	65.891689373297	2.7285714285714286	6502.931292517007	15.209523809523809	80.0	7.0081632653061225	4.229251700680272	2.1877551020408164	4.12312925170068
9.135373489136732	20.34326718704395	1.1028461230547204	4707.956783097994	3.6599377165396407	0.0	6.126525152403569	3.623137034670628	3.222430279137967	3.5681361205404376
18.0	30.0	1.0	1009.0	11.0	80.0	0.0	0.0	0.0	0.0
30.0	48.0	2.0	2911.0	12.0	80.0	3.0	2.0	0.0	2.0
36.0	66.0	3.0	4919.0	14.0	80.0	5.0	3.0	1.0	3.0
43.0	84.0	4.0	8379.0	18.0	80.0	9.0	7.0	3.0	7.0
60.0	100.0	4.0	19999.0	25.0	80.0	40.0	18.0	15.0	17.0

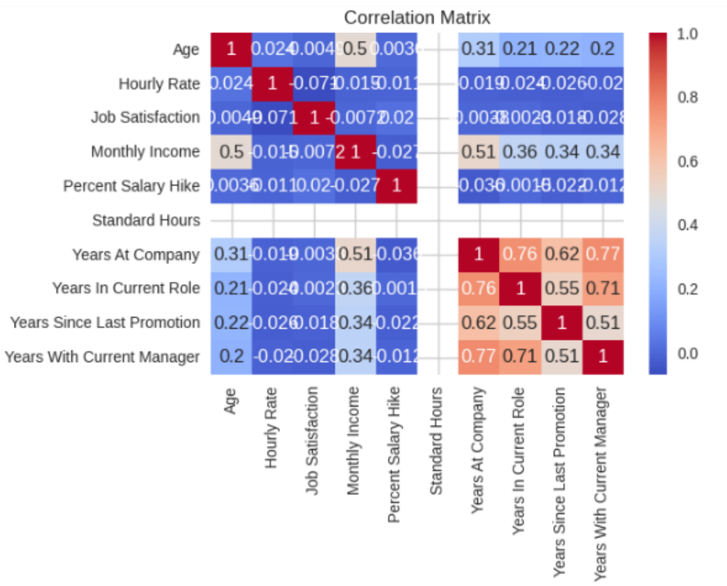
```
[9] df.head()
```

1 to 5 of 5 entries Filter

index	Age	Attrition	Department	EducationField	Gender	HourlyRate	JobRole	JobSatisfaction	MaritalStatus	MonthlyIncome	Over18	PercentSalaryHike	StandardHours	YearsAtCompany	YearsInCurrentRol
0	41	Yes	Sales	Life Sciences	Female	94.0	Sales Executive	4	Single	5993	Y	11	80	6	
1	49	No	Research & Development	Life Sciences	Male	61.0	Research Scientist	2	Married	5130	Y	23	80	10	
2	37	Yes	Research & Development	Other	Male	92.0	Laboratory Technician	3	Single	2090	Y	15	80	0	
3	33	No	Research & Development	Life Sciences	Female	56.0	Research Scientist	3	Married	2909	Y	11	80	8	
4	27	No	Research & Development	Medical	Male	40.0	Laboratory Technician	2	Married	3468	Y	12	80	2	

- I plotted the correlation matrix as a heatmap to help identify strength of linear relationship between pairs of variables in the dataset. The values range from -1 to 1 where -1 indicates a perfect negative correlation, 0 indicates no linear correlation and 1 indicates a perfect linear correlation.

```
[24] import matplotlib.pyplot as plt
     import seaborn as sns
     import matplotlib
     import plotly.express as px
     corr_matrix = df.corr()

     # I plottted the correlation matrix as a heatmap
     plt.figure(figsize=(12, 8))
     sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
     plt.title("Correlation Matrix")
     plt.show()
```



- Now, I will identify any missing data in my dataset. I have identified 2 missing data each under the 'Gender' and 'Hourly Rate' columns.

```
df.isnull().sum()
#In this code, I have identified missing data
```

```
Age                        0
Attrition                  0
Department                 0
EducationField             0
Gender                     2
 HourlyRate                2
JobRole                    0
JobSatisfaction            0
MaritalStatus              0
 MonthlyIncome             0
Over18                     0
PercentSalaryHike          0
StandardHours              0
YearsAtCompany             0
YearsInCurrentRole         0
YearsSinceLastPromotion    0
YearsWithCurrManager       0
dtype: int64
```

```
[12] df[df['Gender'].isnull()]
```

1 to 2 of 2 entries Filter

index	Age	Attrition	Department	EducationField	Gender	HourlyRate	JobRole	JobSatisfaction	MaritalStatus	MonthlyIncome	Over18	PercentSalaryHike	StandardHours	YearsAtCompany	YearsInCurrentRol
1	49	No	Research & Development	Life Sciences	NaN	61.0	Research Scientist	2	Married	5130	Y	23	80	10	
1450	35	No	Human Resources	Life Sciences	NaN	31.0	Human Resources	4	Single	8837	Y	16	80	9	

```
df[df[' HourlyRate  '].isnull()]
```

1 to 2 of 2 entries Filter

index	Age	Attrition	Department	EducationField	Gender	HourlyRate	JobRole	JobSatisfaction	MaritalStatus	MonthlyIncome	Over18	PercentSalaryHike	StandardHours	YearsAtCompany	YearsInCurrentRol
1260	32	No	Research & Development	Technical Degree	Male	NaN	Research Scientist	2	Single	2718	Y	14	80	7	
1383	36	No	Research & Development	Life Sciences	Male	NaN	Laboratory Technician	2	Married	2810	Y	22	80	5	

- First, the attributes headings will be cleaned by spacing out the words where necessary.

```
[14] #DATA CLEANING CODES BELOW
```

```
[15] df.rename(columns = {'EducationField':'Education Field',' HourlyRate  ':'Hourly Rate','JobRole':'Job Role', 'JobSatisfaction':'Job Satisfaction',
' MonthlyIncome  ':'Monthly Income','MaritalStatus':'Marital Status','Over18':'Over 18','PercentSalaryHike':'Percent Salary Hike','StandardHours':'Standard Hours','YearsAtCompany':'
```

- Next, I will clean the gender data by filling in the missing information from pre-determined total of males and females by their marital status. I should have 588 females and 882 males.

```
[16] df['Gender'].fillna('missing',inplace=True)
```

```
[17] df.at[1,'Gender']='Male'
     df.at[1450,'Gender']='Female'
```

```
[18] df.loc[[1,1450]]
```

1 to 2 of 2 entries Filter

index	Age	Attrition	Department	Education Field	Gender	Hourly Rate	Job Role	Job Satisfaction	Marital Status	Monthly Income	Over 18	Percent Salary Hike	Standard Hours	Years At Company	Years In Cur
1	49	No	Research & Development	Life Sciences	Male	61.0	Research Scientist	2	Married	5130	Y	23	80	10	
1450	35	No	Human Resources	Life Sciences	Female	31.0	Human Resources	4	Single	8837	Y	16	80	9	

- The numerical data will be cleaned using interpolation.

```
[19] df['Hourly Rate'].fillna(df['Hourly Rate'].interpolate(),inplace=True)
```

```
df.loc[[1260,1383]]
```

1 to 2 of 2 entries Filter

index	Age	Attrition	Department	Education Field	Gender	Hourly Rate	Job Role	Job Satisfaction	Marital Status	Monthly Income	Over 18	Percent Salary Hike	Standard Hours	Years At Company	Years In Cur
1260	32	No	Research & Development	Technical Degree	Male	59.0	Research Scientist	2	Single	2718	Y	14	80	7	
1383	36	No	Research & Development	Life Sciences	Male	40.0	Laboratory Technician	2	Married	2810	Y	22	80	5	

- Now, I will confirm that my dataset is clean.

```
df.isnull().sum()
#In this code, it is seen that my dataset for Classification has been cleaned and ready for preliminary visualization.
```

```
Age                             0
Attrition                       0
Department                      0
Education Field                 0
Gender                          0
Hourly Rate                     0
Job Role                        0
Job Satisfaction                0
Marital Status                  0
Monthly Income                  0
Over 18                         0
Percent Salary Hike             0
Standard Hours                  0
Years At Company                0
Years In Current Role           0
Years Since Last Promotion      0
Years With Current Manager      0
dtype: int64
```

4.0 WHY AND HOW EMPLOYEE ATTRITION ANALYSIS WILL ADD VALUE TO THE SERVICES INDUSTRY IN UNITED KINGDOM

Employee attrition analysis will allow organizations within the services industry to make informed decisions that positively impact their workforce, culture and goals. By using predictive analysis, organizations can implement tailored strategies that encourages employee satisfaction, improve retention rates and create a stable work environment. Some of the benefits of data-driven decision making in the context of employee attrition are: early identification of root causes of attrition, effective retention strategies, resource allocation, cost saving, feedback for managers, long-term workforce planning.

5.0 DATA ANALYSIS STEPS WITH PYTHON

In the deployment of my solution, I will follow the steps below:

1. **Data exploration:** I sourced extensively on repositories like Statista and Kaggle until I found the HR Analytics dataset by Bhanupratap Biswas.
2. **Data cleaning and preprocessing:** I will handle missing data, outliers and inconsistencies as well as convert data types to appropriate formats.
3. **Preliminary visualization:** I will explore the dataset to understand its structure, distribution and characteristics with visualization libraries such as Seaborn and Plotly.
4. **Feature selection:** I will select the features that enhance the predictive power of my data.
5. **Data analysis:** I will use machine learning algorithms to make my analysis using libraries like NumPy, Scikit-learn and Pandas.
6. **Model building and evaluation:** I will be using the train-test-split to evaluate my models using appropriate metrics to measure their performance.
7. **Insight generation:** I will interpret my analysis and derive recommendations based on the result achieved.

6.0 DATA ANALYTICS TASKS

I will be performing descriptive and exploratory data analytics, data cleaning and preprocessing, feature selection, predictive analytics, prescriptive analytics, clustering and anomaly detection by model evaluation and validation. I will be utilizing K-Nearest Neighbour and Logistic Regression as my models.

6.1 K-NEAREST NEIGHBOR

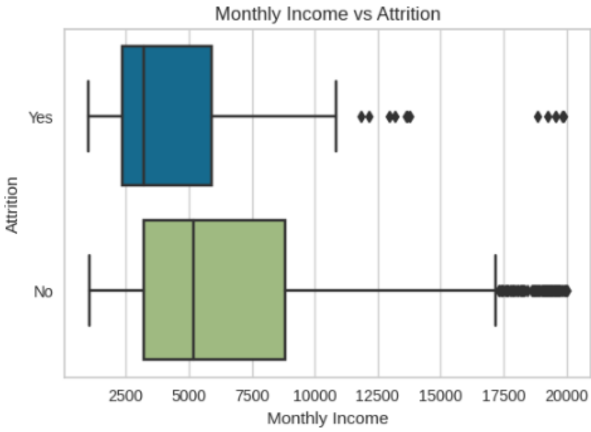
This is a machine learning algorithm used for classification and regression tasks. A value of 'K', which represents the nearest elements in the selected feature variable, and the right distance metric will be chosen. Before I carry out any of the selected tasks, my data needs to be organised with target selection as 'Y' and feature(s) selection denoted with 'X'. See code below:

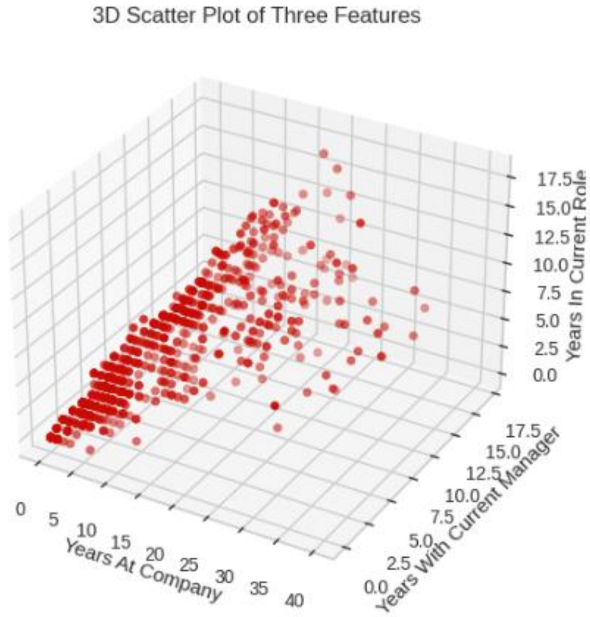
```
#DATA ORGANIZATION
df_target_data = df[['Attrition']]
df_feature_data = df[['Monthly Income', 'Percent Salary Hike', 'Years At Company', 'Years With Current Manager', 'Years In Current Role']]

x = df_feature_data
y = df_target_data

print(x)
print(y)
```

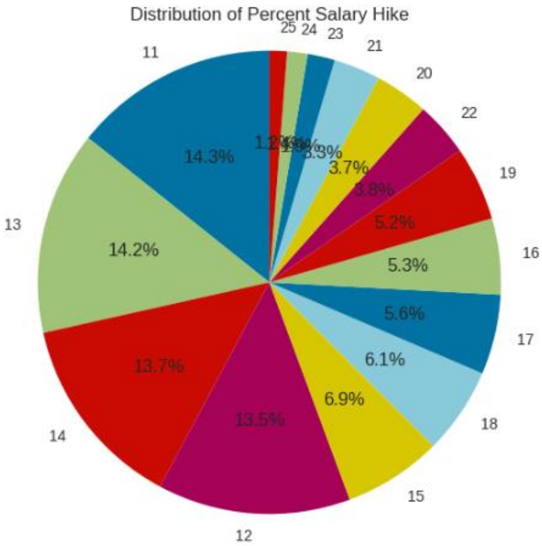
```
      Monthly Income  Percent Salary Hike  Years At Company  \
0               5993                   11                 6
1               5130                   23                10
2               2090                   15                 0
3               2909                   11                 8
4               3468                   12                 2
...              ...                  ...               ...
1465            2571                   17                 5
1466            9991                   15                 7
1467            6142                   20                 6
1468            5390                   14                 9
1469            4404                   12                 4

      Years With Current Manager  Years In Current Role
0                              5                      4
1                              7                      7
2                              0                      0
3                              0                      7
4                              2                      2
...                          ...                    ...
```

- Next, I visualized my target and features variables against each other using libraries like Seaborn, Axes3D, Matplotlib and Plotly.

```
# Plotting with Seaborn
plt.figure(figsize=(10, 6))
sns.boxplot(y='Attrition', x='Monthly Income', data=df)
plt.title('Monthly Income vs Attrition')
plt.show()

# Plotting with Axes3D
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
years_at_company = df['Years At Company']
years_with_current_manager = df['Years With Current Manager']
years_in_current_role = df['Years In Current Role']

ax.scatter(years_at_company, years_with_current_manager, years_in_current_role, c='r', marker='o')
ax.set_xlabel('Years At Company')
ax.set_ylabel('Years With Current Manager')
ax.set_zlabel('Years In Current Role')
ax.set_title('3D Scatter Plot of Three Features')

# Plot Percent Salary Hike as a Pie Chart
department_counts = df['Percent Salary Hike'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(department_counts, labels=department_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Percent Salary Hike')
plt.axis('equal')  # Equal aspect ratio ensures that the pie chart is drawn as a circle.
plt.show()
```


Monthly Income vs Attrition
Attrition
Yes
No
2500
5000
7500
10000
12500
15000
17500
20000
Monthly Income


3D Scatter Plot of Three Features
Years At Company
Years With Current Manager
Years In Current Role


Distribution of Percent Salary Hike
11
14.3%
13
14.2%
14
13.7%
12
13.5%
15
6.9%
18
6.1%
17
5.6%
16
5.3%
19
5.2%
22
3.8%
20
3.7%
21
3.3%
23
24
25

- Next, I split my data into 80% for training my model and 20% for testing the performance. I used the Decision Tree Classifier to help determine the best split. I chose a random state of 42 to ensure consistency of the split and stratified my dataset to ensure stable class distribution. I also converted my categorical variables using LabelEncoder.

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

# Split the data into training and test sets
# Given the class imbalance of my dataset, stratify is used to ensure class distribution is preserved during the split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42, stratify=y)

# Convert categorical features using one-hot encoding
x_train_encoded = pd.get_dummies(x_train)
x_test_encoded = pd.get_dummies(x_test)

# My target variable column contains strings 'No' and 'Yes'. I will encode them to become floats '0' and '1' respectively
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.transform(y_test)
# Now y_train and y_test contains 0 for 'No' and 1 for 'Yes'
```

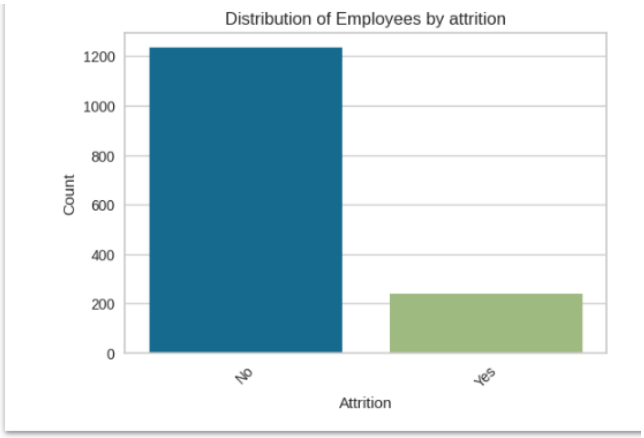
```
train_scores, test_scores = list(), list()
values = [i for i in range(1, 21)]
# Model fitting
for i in values:
    model = DecisionTreeClassifier(max_depth=i)
    model.fit(x_train_encoded, y_train)

    train_yhat = model.predict(x_train_encoded)
    train_acc = accuracy_score(y_train, train_yhat)
    train_scores.append(train_acc)

    test_yhat = model.predict(x_test_encoded)
    test_acc = accuracy_score(y_test, test_yhat)
    test_scores.append(test_acc)

    print('>%d, train: %.3f, test: %.3f' % (i, train_acc, test_acc))

plt.plot(values, train_scores, '-o', label='Train')
plt.plot(values, test_scores, '-o', label='Test')
plt.xlabel('Max Depth')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
>1, train: 0.838, test: 0.840
>2, train: 0.843, test: 0.796
>3, train: 0.848, test: 0.810
>4, train: 0.854, test: 0.789
>5, train: 0.861, test: 0.813
```

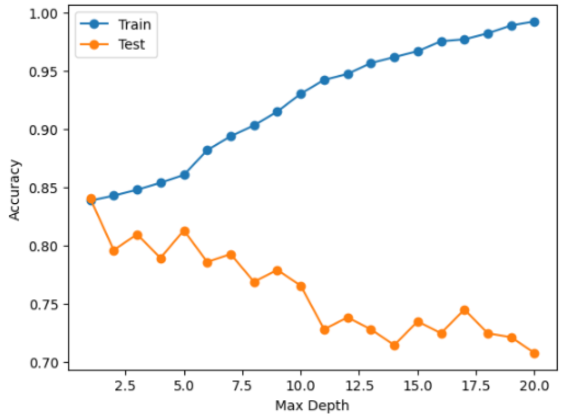
- I also conducted a multivariate analysis of attrition on this dataset.

```
#Attrition analysis: univariate, bivariate and multivariate
import matplotlib.pyplot as plt
import seaborn as sns

##Attrition frequency
Attrition_counts = df['Attrition'].value_counts()
print(Attrition_counts)

##Visualizing attrition in a bar plot
plt.figure(figsize=(8, 6))
sns.barplot(x=Attrition_counts.index, y=Attrition_counts.values)
plt.title('Distribution of Employees by attrition')
plt.xlabel('Attrition')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

```
No     1233
Yes     237
Name: Attrition, dtype: int64
```



- I used the Decision Tree Classifier to help identify the best value of 'K' that will give my model a better accuracy score. From the diagram below, the best max depth or 'K' for approximately 82% accuracy is 3.



- Now, I will use KNN to make predictions on the dataset. As a result, an array of 'Yes' and 'No' was ascertained.

```
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors=3)
y = np.ravel(y)
knn.fit(x,y)
knn.predict(x)
```

```
array(['No', 'No', 'Yes', ..., 'No', 'No', 'No'], dtype=object)
```

- I will conduct further predictive analysis on a new data I will create with the code below:

```
[40] new_data = pd.DataFrame({'Monthly Income': [1000.0, 9965.0, 13877.0, 17986.0, 20100.0],
                            'Percent Salary Hike': [-10.0, -5.0, 0.0, 5.0, 10.0],
                            'Years At Company': [6.0, 11.0, 19.0, 41.0, 49.0],
                            'Years With Current Manager': [1.0, 5.0, 10.0, 19.0, 23.0],
                            'Years In Current Role': [2.0, 7.0, 15.0, 26.0, 30.0]})
```

- Here is the result:

```
knn.predict(new_data)
```

```
array(['No', 'No', 'No', 'No', 'No'], dtype=object)
```

6.2 LOGISTIC REGRESSION

Logistic regression is a statistical classification algorithm used for predicting the probability of a binary outcome based on one or more predictor variable. I will now use Logistic Regression to make predictions on this dataset with the code below:

```
from sklearn.linear_model import LogisticRegression
Logreg=LogisticRegression()
import numpy as np
y = np.ravel(y)
logreg = LogisticRegression(max_iter=200)
logreg.fit(x,y)
logreg.predict(x)
```

```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

- Now, I will make predictions with the new data I created previously

```
new_data = pd.DataFrame({'Monthly Income': [1000.0, 9965.0, 13877.0, 17986.0, 20100.0],
                         'Percent Salary Hike': [-10.0, -5.0, 0.0, 5.0, 10.0],
                         'Years At Company': [6.0, 11.0, 19.0, 41.0, 49.0],
                         'Years With Current Manager': [1.0, 5.0, 10.0, 19.0, 23.0],
                         'Years In Current Role': [2.0, 7.0, 15.0, 26.0, 30.0]})
```

```
predicted_class = logreg.predict(new_data)

print("Predicted Class:", predicted_class)
```

```
Predicted Class: ['Yes' 'No' 'No' 'No' 'No']
```

It is observed that both models perform well for predictive analytics.

7.0 TARGET VARIABLE

Employee attrition is my target variable in the chosen dataset. Attrition, in this analysis, will be interpreted as a binary function of ‘Yes’- meaning attrition will occur and ‘No’ – which means attrition will not take place, all based on certain elements of my independent variables also known as features.

7.1 FEATURES FOR ANALYSIS

The selection of feature(s) is very crucial to the successful prediction of the target variable. After several feature importance analysis, I discovered that the following features have a stronger correlation to attrition: monthly income, percent salary hike, years at company, years in current role and years with current manager. These are the features I will combine for my analysis.

8.0 TRAINING DATA

I have used the train-test-split technique to evaluate the performance of my models. Due to the class imbalance of my dataset, it has been divided into the training set (80%) which is used to train my model and the test set (20%) which will be used to test the performance of my models. The purpose of the split is to simulate how well my model will generalize to new unseen data.

9.0 MODEL EVALUATION

Model evaluation methods helps assess how well machine learning algorithms perform on unseen data. I have used Accuracy, Confusion Matrix and K-fold Cross Validation for my model evaluation.

- **ACCURACY**

  Accuracy is a metric used to measure the proportion of correctly predicted instances among all instances in the dataset. Below are the codes used to evaluate KNN and Logistic Regression accuracy score on this dataset:

```
#MODEL EVALUATION
#LOGISTIC REGRESSION
```

```
y_pred=logreg.predict(x)
y_pred
```

```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

```
from sklearn import metrics
metrics.accuracy_score(y, y_pred)
```

```
0.8387755102040816
```

```
#MODEL EVALUATION
#KNN
```

```
knn.fit(x,np.ravel(y))
y_pred_knn3= knn.predict(x)
metrics.accuracy_score(y,y_pred_knn3)
#For KNN = 3, accuracy =87.48%
```

```
0.8748299319727891
```

- **CONFUSION MATRIX**

  This evaluation technique is used to determine the number of correct and incorrect predictions made by the model on a classification task. The following report was derived:

  Prediction accuracy of approximately 77% was achieved

  True Positive- the number of instances that were predicted as positive is 10

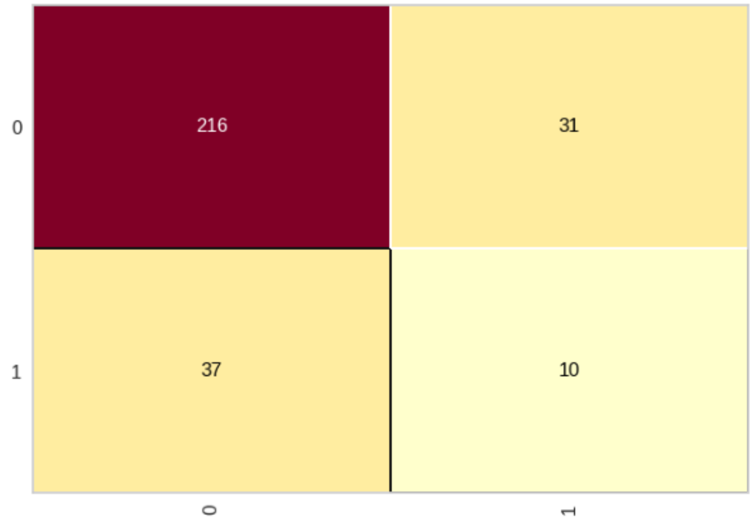
  True Negative- the number of instances that were predicted as negative is 216

  False Positive- the number of instances that were incorrectly predicted as positive is 31

  False Negative- the number of instances that were incorrectly predicted as negative is 37

```
from yellowbrick.classifier import ConfusionMatrix
knn.fit(x_train, y_train)

# Create the Confusion Matrix
cm = ConfusionMatrix(knn)
cm.score(x_test, y_test)
```

```
0.7687074829931972
```



- **K-FOLD VALIDATION**

This is a technique used to assess the performance of machine learning models on unseen data by partitioning the dataset into subsets for training and testing, also known as folds. In this analysis, I chose ‘5’ folds and the results for KNN is shown below with a mean accuracy of 81.70%.

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold

model = KNeighborsClassifier()
num_folds = 5
# I have created a k-fold cross-validation object
kfold = KFold(n_splits=num_folds, shuffle=True, random_state=42)
scores = cross_val_score(model, x, y, cv=kfold, scoring='accuracy')
for fold, score in enumerate(scores, start=1):
    print(f'Fold {fold} Accuracy: {score:.4f}')

# Calculate the mean and standard deviation of the accuracy scores
mean_accuracy = scores.mean()
std_accuracy = scores.std()
print(f'Mean Accuracy: {mean_accuracy:.4f}')
print(f'Standard Deviation of Accuracy: {std_accuracy:.4f}')
```

```
Fold 1 Accuracy: 0.8435
Fold 2 Accuracy: 0.8299
Fold 3 Accuracy: 0.7993
Fold 4 Accuracy: 0.8265
Fold 5 Accuracy: 0.7857
Mean Accuracy: 0.8170
Standard Deviation of Accuracy: 0.0212
```

- For Logistic Regression, the number of folds remains the same with a mean accuracy of 83.88%.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
num_folds = 5
# I have created a k-fold cross-validation object
kfold = KFold(n_splits=num_folds, shuffle=True, random_state=42)
scores = cross_val_score(model, x, y, cv=kfold, scoring='accuracy')
for fold, score in enumerate(scores, start=1):
    print(f'Fold {fold} Accuracy: {score:.4f}')

# Calculate the mean and standard deviation of the accuracy scores
mean_accuracy = scores.mean()
std_accuracy = scores.std()
print(f'Mean Accuracy: {mean_accuracy:.4f}')
print(f'Standard Deviation of Accuracy: {std_accuracy:.4f}')
```

```
Fold 1 Accuracy: 0.8673
Fold 2 Accuracy: 0.8571
Fold 3 Accuracy: 0.8265
Fold 4 Accuracy: 0.8401
Fold 5 Accuracy: 0.8027
Mean Accuracy: 0.8388
Standard Deviation of Accuracy: 0.0228
```

10.0 CONCLUSION

In conclusion, the application of data analytics in optimizing workforce management has been established in the analysis above. As observed, high employee turnover was observed in the sales department especially amongst junior-level staff and a large bulk of Research Scientists and Laboratory Technicians expressed lower job satisfaction. This can be solved be providing better career growth opportunities and fostering a culture of open communication and feedback.

Whilst it is key to understand the role certain variables plays in maintaining employee satisfaction, it is equally important to devise effective futuristic strategies to keep staff fulfilled. The integration of machine learning algorithms such as those used in this report proves valuable in predicting employee attrition as well as prescribing effective strategies to ensure staff retention.

11.0 RECOMMENDATION

Building from the insight in this report, the following recommendations are proposed for companies who will employ the use of data analytics in employee attrition analysis:

1. Ensure accurate and comprehensive employee data are collected and updated regularly. A more robust data set will deliver better analysis accuracy.
2. Prioritize selecting the best feature variables that has significant correlation to employee attrition. This will make your model more efficient.
3. Leverage on your predictions to implement proactive intervention strategies such as creative retention programs.
4. Use data-driven insight to address identified factors influencing attrition.
5. Ensure continuous monitoring of attrition trends and make timely adjustment to meet evolving workforce dynamics.

By embracing these recommendations, any organization within the UK's services sector and beyond can embrace the power of data to mitigate the effects of employee attrition as well as foster a work environment that ensures career fulfilment and company loyalty.

REFERENCES

Bhanupratap Biswas (July 2023). HR Analytics dataset: Case Study. Available at: https://www.kaggle.com/datasets/bhanupratapbiswas/hr-analytics-case-study Accessed on: 1st August, 2023.

Chris Stokel-Walker (December 2020). "Why you might be about to lose vital talent." Available at: https://www.raconteur.net/hr/talent-management/employee-attrition-covid Accessed on: 22nd August, 2023.

Dharshini David (January 2023) "What Impact has Brexit had on the UK Economy?". Available at: https://www.bbc.co.uk/news/business-64450882 Accessed on: 22nd August, 2023.

Financial Times (May 2021). Brexit. Available at: https://www.ft.com/content/20a626ab-d221-43e6-9990-bfd1e1ff132d Accessed on: 22nd August, 2023.

Henry Duquemin, Gemma Rabbaiotti, Iolo Tomlinson et al (2019). Office for National Statistics: Services Sector, UK: 2008 – 2018. Available at: https://www.ons.gov.uk/economy/economicoutputandproductivity/output/articles/servicessectoruk/2008to2018. Accessed on: 22nd August, 2023.

Johannes Ledolter (2013). Data Mining and Business Analytics with R. Oxford: Wiley.

John Allcoat (2023). Office for National Statistics: Services Sector, UK: June 2023. Available at:https://www.ons.gov.uk/economy/economicoutputandproductivity/output/bulletins/indexofservices/latest . Accessed on: 22nd August, 2023.

Lloyd. Chris J. (2011). Data-Driven Business Decisions. Oxford: Wiley.

Marr Bernard (2016). Big data in practice how 45 successful companies used big data analytics to deliver extraordinary results. West Sussex: Wiley.

Sarkar, Deepayan (2008). Lattice multivariate data visualization with R. New York: Springer.

United Kingdom Parliament: House of Commons Library. Service Industries: Key Economic Indicators (August 2023). Available at: https://commonslibrary.parliament.uk/research-briefings/sn02786/ . Accessed on: 22nd August, 2023.

APPENDIX

APPENDIX 1: DATA EXPLORATION

```
[ ] #DATA EXPLORATION
```

```
from google.colab import drive
drive.mount('/content/drive')
#My Google Drive was mounted
```

```
Mounted at /content/drive
```

```
[ ] from google.colab import drive
drive.mount('/content/drive')
path = "/content/HR-Employee Attrition.csv"
#My dataset csv file has been created as a path in this code
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ] import pandas as pd
print (path)
path ="/content/drive/My Drive/DDM/HR-Employee Attrition.csv"
df=pd.read_csv(path)
```

```
/content/HR-Employee Attrition.csv
```

```
df.describe
```

```
1466   39        No  Research & Development         Medical    Male
1467   27        No  Research & Development   Life Sciences    Male
1468   49        No                   Sales         Medical    Male
1469   34        No  Research & Development         Medical    Male

       HourlyRate                    JobRole  JobSatisfaction MaritalStatus  \
0            94.0            Sales Executive                4        Single
1            61.0         Research Scientist                2       Married
2            92.0      Laboratory Technician                3        Single
3            56.0         Research Scientist                3       Married
4            40.0      Laboratory Technician                2       Married
...           ...                        ...              ...           ...
1465         41.0      Laboratory Technician                4       Married
1466         42.0  Healthcare Representative                1       Married
1467         87.0     Manufacturing Director                2       Married
1468         63.0            Sales Executive                2       Married
1469         82.0      Laboratory Technician                3       Married

       MonthlyIncome  Over18  PercentSalaryHike  StandardHours  \
0               5993       Y                 11             80
1               5130       Y                 23             80
2               2090       Y                 15             80
3               2909       Y                 11             80
4               3468       Y                 12             80
...              ...     ...                ...            ...
1465            2571       Y                 17             80
1466            9991       Y                 15             80
```

```
[7] df.dtypes
```

```
Age                         int64
Attrition                  object
Department                 object
EducationField             object
Gender                     object
 HourlyRate               float64
JobRole                    object
JobSatisfaction             int64
MaritalStatus              object
 MonthlyIncome              int64
Over18                     object
PercentSalaryHike           int64
StandardHours               int64
YearsAtCompany              int64
YearsInCurrentRole          int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
dtype: object
```

```
df.describe()
```

1 to 8 of 8 entries Filter

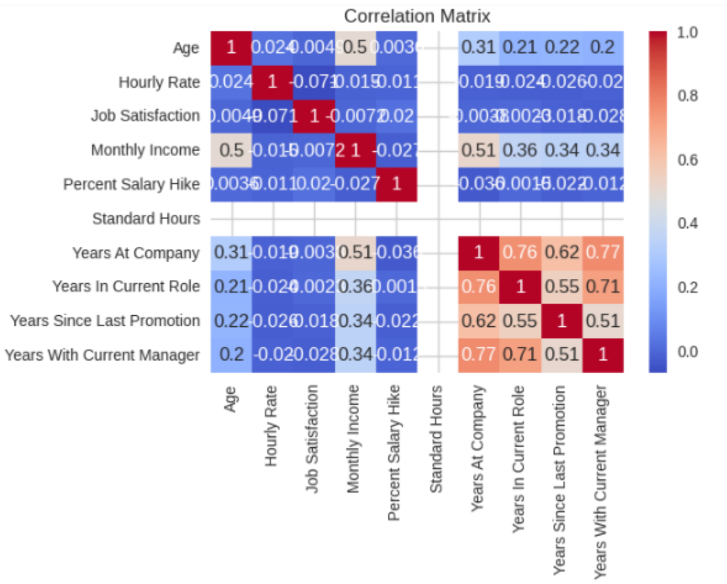
Age	HourlyRate	JobSatisfaction	MonthlyIncome	PercentSalaryHike	StandardHours	YearsAtCompany	YearsInCurrentRole	YearsSinceLastPromotion	YearsWithCurrManager
1470.0	1468.0	1470.0	1470.0	1470.0	1470.0	1470.0	1470.0	1470.0	1470.0
36.923809523809524	65.891689373297	2.7285714285714286	6502.931292517007	15.209523809523809	80.0	7.0081632653061225	4.229251700680272	2.1877551020408164	4.12312925170068
9.135373489136732	20.34326718704395	1.1028461230547204	4707.956783097994	3.6599377165396407	0.0	6.126525152403569	3.623137034670628	3.222430279137967	3.5681361205404376
18.0	30.0	1.0	1009.0	11.0	80.0	0.0	0.0	0.0	0.0
30.0	48.0	2.0	2911.0	12.0	80.0	3.0	2.0	0.0	2.0
36.0	66.0	3.0	4919.0	14.0	80.0	5.0	3.0	1.0	3.0
43.0	84.0	4.0	8379.0	18.0	80.0	9.0	7.0	3.0	7.0
60.0	100.0	4.0	19999.0	25.0	80.0	40.0	18.0	15.0	17.0

```
[9] df.head()
```

1 to 5 of 5 entries Filter

index	Age	Attrition	Department	EducationField	Gender	HourlyRate	JobRole	JobSatisfaction	MaritalStatus	MonthlyIncome	Over18	PercentSalaryHike	StandardHours	YearsAtCompany	YearsInCurrentRol
0	41	Yes	Sales	Life Sciences	Female	94.0	Sales Executive	4	Single	5993	Y	11	80	6	
1	49	No	Research & Development	Life Sciences	Male	61.0	Research Scientist	2	Married	5130	Y	23	80	10	
2	37	Yes	Research & Development	Other	Male	92.0	Laboratory Technician	3	Single	2090	Y	15	80	0	
3	33	No	Research & Development	Life Sciences	Female	56.0	Research Scientist	3	Married	2909	Y	11	80	8	
4	27	No	Research & Development	Medical	Male	40.0	Laboratory Technician	2	Married	3468	Y	12	80	2	

```
[24] import matplotlib.pyplot as plt
     import seaborn as sns
     import matplotlib
     import plotly.express as px
     corr_matrix = df.corr()

     # I plottted the correlation matrix as a heatmap
     plt.figure(figsize=(12, 8))
     sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
     plt.title("Correlation Matrix")
     plt.show()
```



APPENDIX 2: DATA CLEANING WITH PYTHON

IDENTIFICATION OF MISSING DATA

```
df.isnull().sum()
#In this code, I have identified missing data
```

```
Age                        0
Attrition                  0
Department                 0
EducationField             0
Gender                     2
 HourlyRate                2
JobRole                    0
JobSatisfaction            0
MaritalStatus              0
 MonthlyIncome             0
Over18                     0
PercentSalaryHike          0
StandardHours              0
YearsAtCompany             0
YearsInCurrentRole         0
YearsSinceLastPromotion    0
YearsWithCurrManager       0
dtype: int64
```

```
[12] df[df['Gender'].isnull()]
```

1 to 2 of 2 entries Filter

index	Age	Attrition	Department	EducationField	Gender	HourlyRate	JobRole	JobSatisfaction	MaritalStatus	MonthlyIncome	Over18	PercentSalaryHike	StandardHours	YearsAtCompany	YearsInCurrentRol
1	49	No	Research & Development	Life Sciences	NaN	61.0	Research Scientist	2	Married	5130	Y	23	80	10	
1450	35	No	Human Resources	Life Sciences	NaN	31.0	Human Resources	4	Single	8837	Y	16	80	9	

```
df[df[' HourlyRate  '].isnull()]
```

1 to 2 of 2 entries Filter

index	Age	Attrition	Department	EducationField	Gender	HourlyRate	JobRole	JobSatisfaction	MaritalStatus	MonthlyIncome	Over18	PercentSalaryHike	StandardHours	YearsAtCompany	YearsInCurrentRol
1260	32	No	Research & Development	Technical Degree	Male	NaN	Research Scientist	2	Single	2718	Y	14	80	7	
1383	36	No	Research & Development	Life Sciences	Male	NaN	Laboratory Technician	2	Married	2810	Y	22	80	5	

APPENDIX 3: DATA CLEANING

```
[14] #DATA CLEANING CODES BELOW
```

```
[15] df.rename(columns = {'EducationField':'Education Field',' HourlyRate  ':'Hourly Rate','JobRole':'Job Role', 'JobSatisfaction':'Job Satisfaction',
' MonthlyIncome  ':'Monthly Income','MaritalStatus':'Marital Status','Over18':'Over 18','PercentSalaryHike':'Percent Salary Hike','StandardHours':'Standard Hours','YearsAtCompany':'
```

```
[16] df['Gender'].fillna('missing',inplace=True)
```

```
[17] df.at[1,'Gender']='Male'
     df.at[1450,'Gender']='Female'
```

```
[18] df.loc[[1,1450]]
```

1 to 2 of 2 entries Filter

index	Age	Attrition	Department	Education Field	Gender	Hourly Rate	Job Role	Job Satisfaction	Marital Status	Monthly Income	Over 18	Percent Salary Hike	Standard Hours	Years At Company	Years In Cur
1	49	No	Research & Development	Life Sciences	Male	61.0	Research Scientist	2	Married	5130	Y	23	80	10	
1450	35	No	Human Resources	Life Sciences	Female	31.0	Human Resources	4	Single	8837	Y	16	80	9	

```
[19] df['Hourly Rate'].fillna(df['Hourly Rate'].interpolate(),inplace=True)
```

```
df.loc[[1260,1383]]
```

1 to 2 of 2 entries Filter

index	Age	Attrition	Department	Education Field	Gender	Hourly Rate	Job Role	Job Satisfaction	Marital Status	Monthly Income	Over 18	Percent Salary Hike	Standard Hours	Years At Company	Years In Cur
1260	32	No	Research & Development	Technical Degree	Male	59.0	Research Scientist	2	Single	2718	Y	14	80	7	
1383	36	No	Research & Development	Life Sciences	Male	40.0	Laboratory Technician	2	Married	2810	Y	22	80	5	

```
df.isnull().sum()
#In this code, it is seen that my dataset for Classification has been cleaned and ready for preliminary visualization.
```

```
Age                         0
Attrition                   0
Department                  0
Education Field             0
Gender                      0
Hourly Rate                 0
Job Role                    0
Job Satisfaction            0
Marital Status              0
Monthly Income              0
Over 18                     0
Percent Salary Hike         0
Standard Hours              0
Years At Company            0
Years In Current Role       0
Years Since Last Promotion  0
Years With Current Manager  0
dtype: int64
```

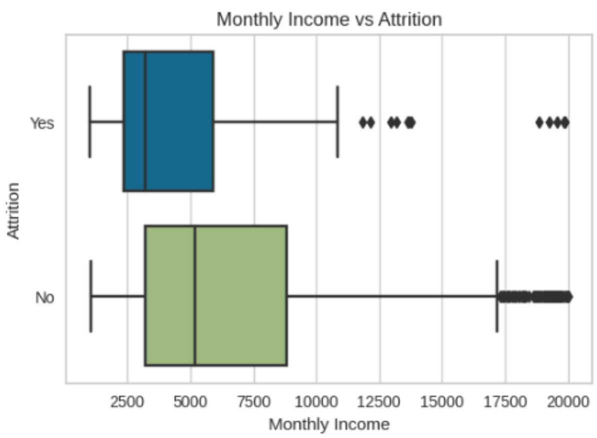
APPENDIX 4: TARGET AND FEATURE SELECTION

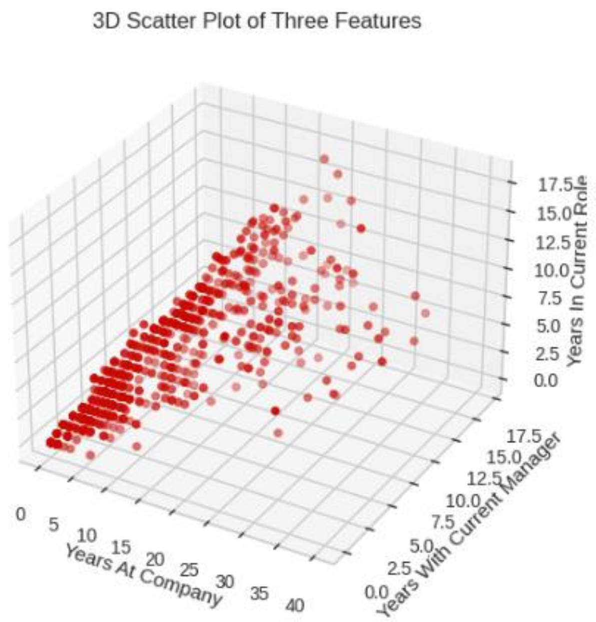
```
#DATA ORGANIZATION
df_target_data = df[['Attrition']]
df_feature_data = df[['Monthly Income', 'Percent Salary Hike', 'Years At Company', 'Years With Current Manager', 'Years In Current Role']]

x = df_feature_data
y = df_target_data

print(x)
print(y)
```

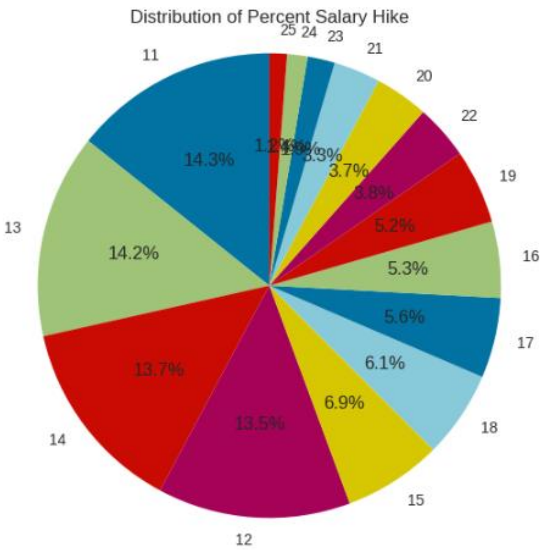
```
      Monthly Income  Percent Salary Hike  Years At Company  \
0               5993                   11                 6
1               5130                   23                10
2               2090                   15                 0
3               2909                   11                 8
4               3468                   12                 2
...              ...                  ...               ...
1465            2571                   17                 5
1466            9991                   15                 7
1467            6142                   20                 6
1468            5390                   14                 9
1469            4404                   12                 4

      Years With Current Manager  Years In Current Role
0                              5                      4
1                              7                      7
2                              0                      0
3                              0                      7
4                              2                      2
...                          ...                    ...
```

APPENDIX 5: PRELIMINARY VISUALIZATION

```
# Plotting with Seaborn
plt.figure(figsize=(10, 6))
sns.boxplot(y='Attrition', x='Monthly Income', data=df)
plt.title('Monthly Income vs Attrition')
plt.show()

# Plotting with Axes3D
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
years_at_company = df['Years At Company']
years_with_current_manager = df['Years With Current Manager']
years_in_current_role = df['Years In Current Role']

ax.scatter(years_at_company, years_with_current_manager, years_in_current_role, c='r', marker='o')
ax.set_xlabel('Years At Company')
ax.set_ylabel('Years With Current Manager')
ax.set_zlabel('Years In Current Role')
ax.set_title('3D Scatter Plot of Three Features')

# Plot Percent Salary Hike as a Pie Chart
department_counts = df['Percent Salary Hike'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(department_counts, labels=department_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Percent Salary Hike')
plt.axis('equal')  # Equal aspect ratio ensures that the pie chart is drawn as a circle.
plt.show()
```







APPENDIX 6: MODEL BUILDING WITH TRAIN-TEST SPLIT

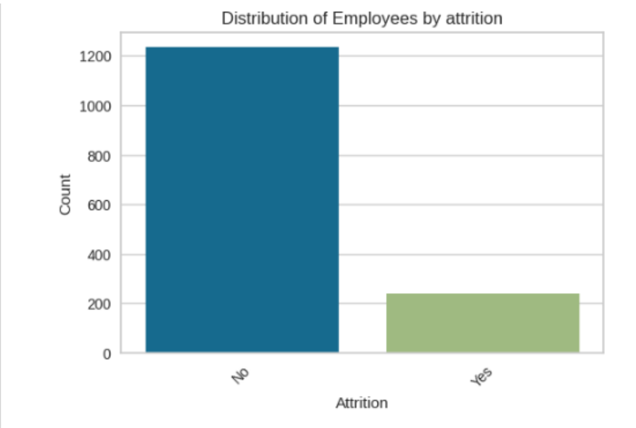
```python
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

# Split the data into training and test sets
# Given the class imbalance of my dataset, stratify is used to ensure class distribution is preserved during the split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42, stratify=y)

# Convert categorical features using one-hot encoding
x_train_encoded = pd.get_dummies(x_train)
x_test_encoded = pd.get_dummies(x_test)

# My target variable column contains strings 'No' and 'Yes'. I will encode them to become floats '0' and '1' respectively
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.transform(y_test)
# Now y_train and y_test contains 0 for 'No' and 1 for 'Yes'
```

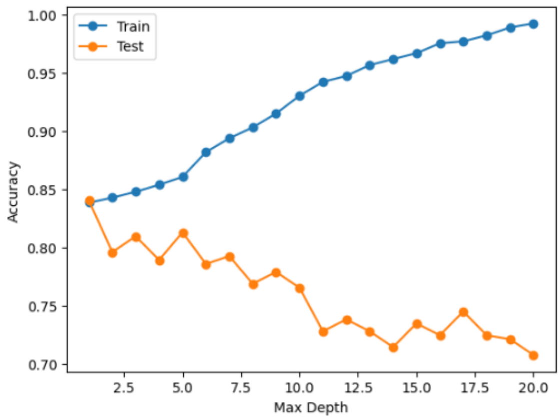
```
train_scores, test_scores = list(), list()
values = [i for i in range(1, 21)]
# Model fitting
for i in values:
    model = DecisionTreeClassifier(max_depth=i)
    model.fit(x_train_encoded, y_train)

    train_yhat = model.predict(x_train_encoded)
    train_acc = accuracy_score(y_train, train_yhat)
    train_scores.append(train_acc)

    test_yhat = model.predict(x_test_encoded)
    test_acc = accuracy_score(y_test, test_yhat)
    test_scores.append(test_acc)

    print('>%d, train: %.3f, test: %.3f' % (i, train_acc, test_acc))

plt.plot(values, train_scores, '-o', label='Train')
plt.plot(values, test_scores, '-o', label='Test')
plt.xlabel('Max Depth')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
>1, train: 0.838, test: 0.840
>2, train: 0.843, test: 0.796
>3, train: 0.848, test: 0.810
>4, train: 0.854, test: 0.789
>5, train: 0.861, test: 0.813
```

```
#Attrition analysis: univariate, bivariate and multivariate
import matplotlib.pyplot as plt
import seaborn as sns

##Attrition frequency
Attrition_counts = df['Attrition'].value_counts()
print(Attrition_counts)

##Visualizing attrition in a bar plot
plt.figure(figsize=(8, 6))
sns.barplot(x=Attrition_counts.index, y=Attrition_counts.values)
plt.title('Distribution of Employees by attrition')
plt.xlabel('Attrition')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

```
No      1233
Yes      237
Name: Attrition, dtype: int64
```





APPENDIX 7: MACHINE LEARNING WITH KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors=3)
y = np.ravel(y)
knn.fit(x,y)
knn.predict(x)
```

```
array(['No', 'No', 'Yes', ..., 'No', 'No', 'No'], dtype=object)
```

```
[40] new_data = pd.DataFrame({'Monthly Income': [1000.0, 9965.0, 13877.0, 17986.0, 20100.0],
                              'Percent Salary Hike': [-10.0, -5.0, 0.0, 5.0, 10.0],
                              'Years At Company': [6.0, 11.0, 19.0, 41.0, 49.0],
                              'Years With Current Manager': [1.0, 5.0, 10.0, 19.0, 23.0],
                              'Years In Current Role': [2.0, 7.0, 15.0, 26.0, 30.0]})
```

```
knn.predict(new_data)
```

```
array(['No', 'No', 'No', 'No', 'No'], dtype=object)
```

APPENDIX 8: MACHINE LEARNING WITH LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression
Logreg=LogisticRegression()
import numpy as np
y = np.ravel(y)
logreg = LogisticRegression(max_iter=200)
logreg.fit(x,y)
logreg.predict(x)
```

```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

```
new_data = pd.DataFrame({'Monthly Income': [1000.0, 9965.0, 13877.0, 17986.0, 20100.0],
                         'Percent Salary Hike': [-10.0, -5.0, 0.0, 5.0, 10.0],
                         'Years At Company': [6.0, 11.0, 19.0, 41.0, 49.0],
                         'Years With Current Manager': [1.0, 5.0, 10.0, 19.0, 23.0],
                         'Years In Current Role': [2.0, 7.0, 15.0, 26.0, 30.0]})
```

```
predicted_class = logreg.predict(new_data)

print("Predicted Class:", predicted_class)
```

```
Predicted Class: ['Yes' 'No' 'No' 'No' 'No']
```

APPENDIX 9: MODEL EVALUATION

- ## ACCURACY

```
#MODEL EVALUATION
#LOGISTIC REGRESSION
```

```
y_pred=logreg.predict(x)
y_pred
```

```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

```
from sklearn import metrics
metrics.accuracy_score(y, y_pred)
```

```
0.8387755102040816
```

```
#MODEL EVALUATION
#KNN
```

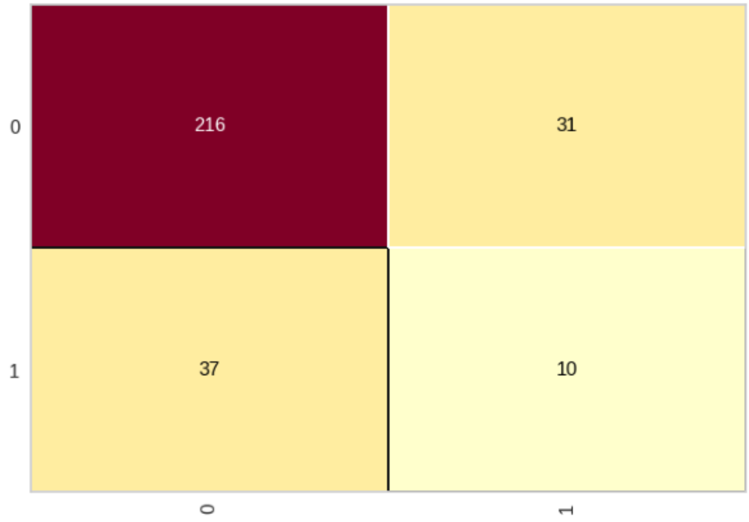
```
knn.fit(x,np.ravel(y))
y_pred_knn3= knn.predict(x)
metrics.accuracy_score(y,y_pred_knn3)
#For KNN = 3, accuracy =87.48%
```

```
0.8748299319727891
```

- ## CONFUSION MATRIX

```
from yellowbrick.classifier import ConfusionMatrix
knn.fit(x_train, y_train)

# Create the Confusion Matrix
cm = ConfusionMatrix(knn)
cm.score(x_test, y_test)
```

```
0.7687074829931972
```



- **K-FOLD VALIDATION**

```python
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold

model = KNeighborsClassifier()
num_folds = 5
# I have created a k-fold cross-validation object
kfold = KFold(n_splits=num_folds, shuffle=True, random_state=42)
scores = cross_val_score(model, x, y, cv=kfold, scoring='accuracy')
for fold, score in enumerate(scores, start=1):
    print(f'Fold {fold} Accuracy: {score:.4f}')

# Calculate the mean and standard deviation of the accuracy scores
mean_accuracy = scores.mean()
std_accuracy = scores.std()
print(f'Mean Accuracy: {mean_accuracy:.4f}')
print(f'Standard Deviation of Accuracy: {std_accuracy:.4f}')
```

```
Fold 1 Accuracy: 0.8435
Fold 2 Accuracy: 0.8299
Fold 3 Accuracy: 0.7993
Fold 4 Accuracy: 0.8265
Fold 5 Accuracy: 0.7857
Mean Accuracy: 0.8170
Standard Deviation of Accuracy: 0.0212
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
num_folds = 5
# I have created a k-fold cross-validation object
kfold = KFold(n_splits=num_folds, shuffle=True, random_state=42)
scores = cross_val_score(model, x, y, cv=kfold, scoring='accuracy')
for fold, score in enumerate(scores, start=1):
    print(f'Fold {fold} Accuracy: {score:.4f}')

# Calculate the mean and standard deviation of the accuracy scores
mean_accuracy = scores.mean()
std_accuracy = scores.std()
print(f'Mean Accuracy: {mean_accuracy:.4f}')
print(f'Standard Deviation of Accuracy: {std_accuracy:.4f}')
```

```
Fold 1 Accuracy: 0.8673
Fold 2 Accuracy: 0.8571
Fold 3 Accuracy: 0.8265
Fold 4 Accuracy: 0.8401
Fold 5 Accuracy: 0.8027
Mean Accuracy: 0.8388
Standard Deviation of Accuracy: 0.0228
```