Project 2: Employee Attrition Prediction

1. Introduction

Employee attrition, also known as employee turnover, is a significant challenge faced by organizations across various industries. It refers to the voluntary or involuntary departure of employees from a company. High attrition rates can have detrimental effects on an organization's productivity, morale, and overall performance. It leads to the loss of valuable talent, disruption in operations, and additional costs associated with hiring and training new employees.

Attrition can occur due to various reasons, including job dissatisfaction, lack of growth opportunities, poor work-life balance, better prospects elsewhere, or personal reasons. Understanding the factors that contribute to employee attrition is crucial for organizations to take proactive measures and implement effective retention strategies.

The goal of this project is to develop a machine learning model that can predict employee attrition based on various factors such as age, job satisfaction, work-life balance, performance ratings, and more. By accurately identifying employees who are at risk of leaving the organization, companies can take targeted actions to address their concerns and retain valuable talent.

Machine learning techniques, particularly supervised learning algorithms, can be employed to analyse historical employee data and identify patterns or relationships between various features and attrition outcomes. By training models on this data, organizations can gain insights into the factors that influence employee attrition and make data-driven decisions to improve employee retention.

The development of an effective employee attrition prediction model can provide organizations with a competitive advantage by reducing turnover costs, minimizing operational disruptions, and fostering a more engaged and satisfied workforce. It can also assist in proactive workforce planning and resource allocation, ensuring that the organization has the necessary talent and expertise to meet its strategic objectives.

In this project, we explore the application of machine learning techniques to predict employee attrition using a real-world dataset. The report outlines the methodology, data preprocessing steps, model selection, and evaluation techniques employed, as well as the results and insights gained from the analysis.

2. Project Prerequisites

Before diving into the implementation details of the employee attrition prediction project, it is essential to ensure that the necessary prerequisites are met. This section outlines the software, libraries, and tools required for the successful execution of the project.

Software Requirements

- Python (version 3.6 or higher): Python is the programming language used for this project. It is recommended to have Python installed on your system or use a Python distribution like Anaconda, which includes Python and several pre-installed packages.
- Integrated Development Environment (IDE) (e.g., PyCharm, Visual Studio Code, Spyder): An IDE provides a convenient environment for writing, debugging, and executing Python code. While not strictly required, an IDE can greatly enhance the development experience.

Python Libraries and Packages

The following Python libraries and packages are required for this project:

- NumPy: A fundamental package for scientific computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions.
- Pandas: A powerful data manipulation and analysis library for Python, providing data structures and data analysis tools for working with structured (tabular, multidimensional, potentially heterogeneous) and time series data.
- Matplotlib: A plotting library for creating static, animated, and interactive visualizations in Python.
- Seaborn: A data visualization library based on Matplotlib, providing a high-level interface for drawing attractive and informative statistical graphics.
- Scikit-learn: A machine learning library for Python, featuring various classification, regression, and clustering algorithms, as well as tools for model evaluation and data preprocessing.

These libraries can be installed individually using pip, the Python package installer, or through Python distribution platforms like Anaconda, which provide pre-built packages for easy installation.

Data

The project utilizes the IBM HR Analytics Employee Attrition dataset, which contains information about employees, including their personal details, job roles, satisfaction levels, and attrition status (whether they have left the company or not). The dataset should be downloaded and placed in an accessible location for the project code to access and preprocess it. With the necessary software, libraries, and data in place, you are ready to proceed with the implementation of the employee attrition prediction project.

3. Steps to build the project

1. Import Required Libraries

Import packages

```
[ ] ##Importing the packages
    #Data processing packages
    import numpy as np
    import pandas as pd

    #Visualization packages
    import matplotlib.pyplot as plt
    import seaborn as sns

    #Machine Learning packages
    from sklearn.metrics import accuracy_score
    from sklearn.preprocessing import LabelEncoder
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.metrics import confusion_matrix

    #Suppress warnings
    import warnings
    warnings.filterwarnings('ignore')
```

The necessary libraries for data manipulation, visualization, and machine learning are imported, including NumPy, Pandas, Matplotlib, Seaborn, and Scikit-learn.

2. Load the Dataset

Import data

```
[ ] #Import Employee Attrition data
    data=pd.read_csv('/content/WA_Fn-UseC_-HR-Employee-Attrition.csv')
    data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Relat:
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

5 rows × 35 columns

The employee attrition dataset is loaded into a Pandas DataFrame using `pd.read_csv()'.The `data.head()` command displays the first few rows of the dataset for initial inspection.

3. Check for Missing Values

Check and remediate if there are any null values

```
data.info()
```

```
<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
```

4. Remove Irrelevant Features

```
[ ] data['Over18'].value_counts()
```

```
Over18
Y    1470
Name: count, dtype: int64
```

COMMENT: From the above output ALL the employees are above 18, so this field does not add any value.

```
[ ] data.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	...	RelationshipSatisfaction	StandardHours
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	...	1470.000000	1470.0
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	2.063946	...	2.712245	80.0
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	1.106940	...	1.081209	0.0
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	1.000000	...	1.000000	80.0
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	1.000000	...	2.000000	80.0
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	...	3.000000	80.0
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	...	4.000000	80.0
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	...	4.000000	80.0

8 rows × 26 columns

COMMENT: Standard deviation(std) for the fields "EmployeeCount" and ."StandardHours" are ZERO. Hence these fields does not add value, hence they can be removed.

```
[ ] #These fields does not add value, hence removed
    data = data.drop(['EmployeeCount','Over18'], axis = 1)
```

The code checks for features that do not add value to the analysis, such as `'Over18'` (all employees are over 18) and `'EmployeeCount'` and `'StandardHours'` (zero standard deviation). These features are then dropped from the dataset using `data.drop()`.

5. Preprocess the Data

```
[ ] #A lambda function is a small anonymous function.
    #A lambda function can take any number of arguments, but can only have one expression.
    data['Attrition']=data['Attrition'].apply(lambda x : 1 if x=='Yes' else 0)
```

The target variable `'Attrition'` is converted to binary numerical values (1 for 'Yes', 0 for 'No'). Categorical features are then encoded using one-hot encoding with `pd.get_dummies()`.

6. Split the Dataset

```
[ ] #Separating Feature and Target matrices
    X = data.drop(['Attrition'], axis=1)
    y=data['Attrition']
```

Scaling the data values to standardize the range of independent variables

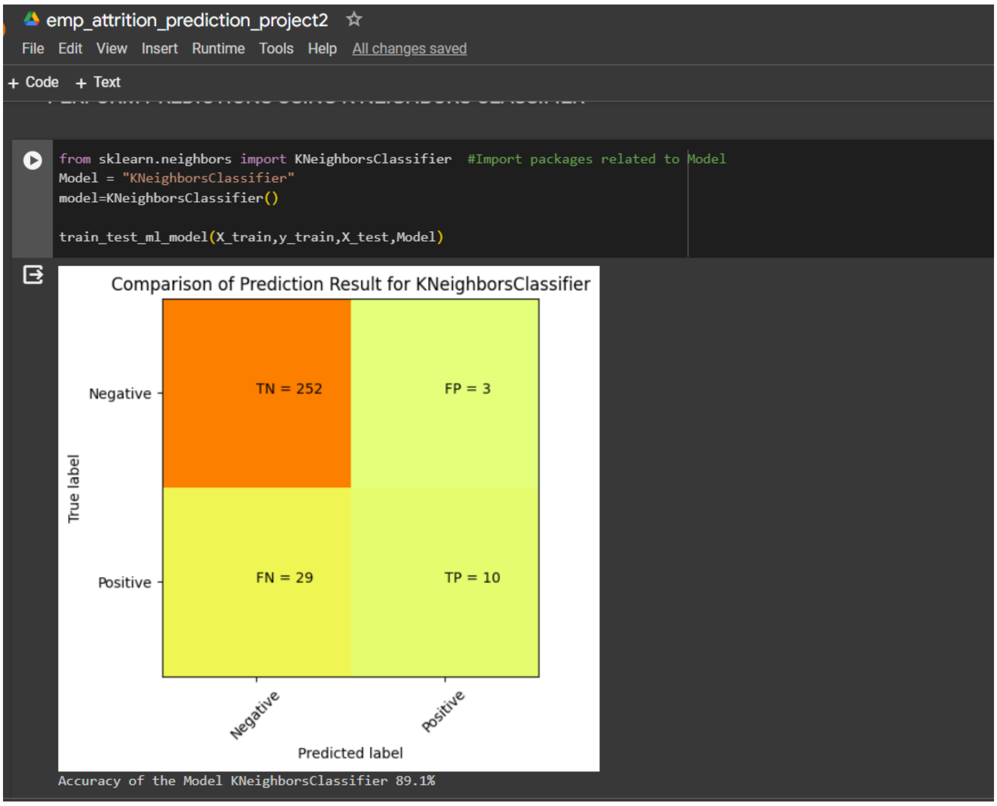
```
[ ] #Feature scaling is a method used to standardize the range of independent variables or features of data.
    #Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly
    from sklearn.preprocessing import StandardScaler
    scale = StandardScaler()
    X = scale.fit_transform(X)
```

Split the data into Training set and Testing set

```
# Split the data into Training set and Testing set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.2,random_state=42)
```

The features (`X`) and target variable (`y`) are separated. The features are then scaled using `StandardScaler` from Scikit-learn. Finally, the dataset is split into training and testing sets using `train_test_split()`.

7. Train and Evaluate the Model



The K-Nearest Neighbors Classifier model is imported from Scikit-learn. The `train_test_ml_model()` function (defined in the provided code) is called to train the model on the training data and evaluate its performance on the testing data using metrics like accuracy and confusion matrix.

8. Make Predictions on New Data

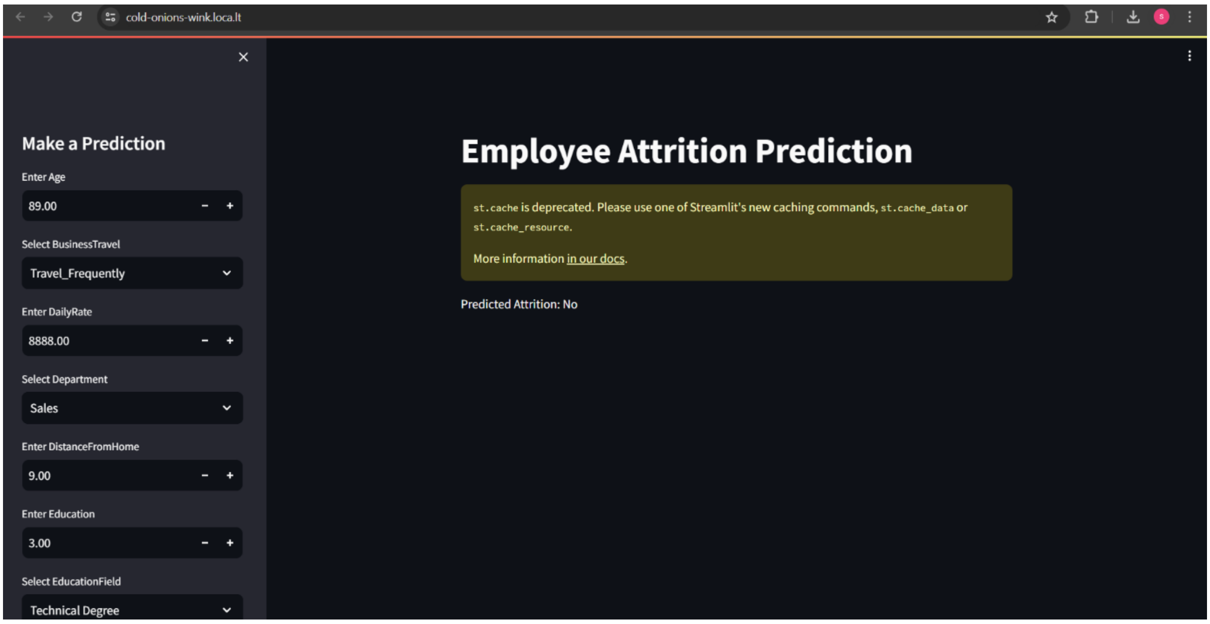
```
# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)

# Train the model
model = RandomForestClassifier()
model.fit(X_scaled, y)

# Make a new prediction
new_data = pd.DataFrame({
    'Age': [30],
    'BusinessTravel': ['Travel_Frequently'],
    'DailyRate': [300],
    'Department': ['Sales'],
    'DistanceFromHome': [10],
    'Education': [3],
    'EducationField': ['Marketing'],
    'EnvironmentSatisfaction': [2],
    'Gender': ['Male'],
    'HourlyRate': [50],
    'JobInvolvement': [3],
    'JobLevel': [2],
    'JobRole': ['Sales Executive'],
    'JobSatisfaction': [4],
    'MaritalStatus': ['Single'],
    'MonthlyIncome': [5000],
    'MonthlyRate': [15],
    'NumCompaniesWorked': [2],
    'OverTimePercent': [10],
    'PerformanceRating': [3],
    'RelationshipSatisfaction': [3],
    'SalaryHike': [15],
    'StandardHours': [80],
    'StockOptionLevel': [1],
    'TotalWorkingYears': [5],
    'TrainingTimesLastYear': [2],
    'WorkLifeBalance': [3],
    'YearsAtCompany': [3],
    'YearsInCurrentRole': [2],
    'YearsSinceLastPromotion': [1],
    'YearsWithCurrManager': [2]
})

# Preprocess the new data
new_data = pd.get_dummies(new_data)
new_data = new_data.reindex(columns=X_scaled.columns, fill_value=0)
new_data_scaled = scaler.transform(new_data)

# Make the prediction
prediction = model.predict(new_data_scaled)

# Print the prediction
print(f"Predicted Attrition: {'Yes' if prediction[0] == 1 else 'No'}")
```

```
Predicted Attrition: No
```

This section of the code demonstrates how to make predictions on new employee data. First, the features are scaled using `StandardScaler`. Then, a Random Forest Classifier model is trained on the scaled features and target variable. New employee data is prepared, preprocessed (encoded and scaled), and fed into the trained model for prediction using `model.predict()`. The predicted attrition status ('Yes' or 'No') is printed.

Throughout the process, various functions defined in the provided code, such as `train_test_ml_model()` and `cm_plot()`, are used to facilitate model training, evaluation, and visualization of the confusion matrix.

4. Output



5. Summary

The employee attrition prediction project aimed to develop a machine learning model capable of predicting employee attrition based on various factors such as age, job satisfaction, work-life balance, performance ratings, and more. By accurately identifying employees at risk of leaving the organization, companies can take proactive measures to address their concerns and implement effective retention strategies.

The project followed a systematic approach, starting with data preprocessing steps to handle missing values, remove irrelevant features, and convert categorical variables to numerical representations. The dataset was then split into training and testing sets to train and evaluate the machine learning models.

The KNN model was selected based on the evaluation metrics and domain knowledge. This model was then integrated into a user-friendly web application developed using the Streamlit library. The web application allowed users to input relevant employee information and obtain predictions on the likelihood of attrition for a given employee.

The development of the employee attrition prediction model and the accompanying web application demonstrated the potential of machine learning techniques in addressing real-world challenges faced by organizations. By leveraging historical employee data and identifying patterns or relationships between various features and attrition outcomes, the project provided organizations with a valuable tool to support data-driven decision-making and improve employee retention strategies.

Overall, the employee attrition prediction project showcased the power of machine learning in solving complex business problems and highlighted the importance of interdisciplinary collaboration between data scientists, human resource professionals, and domain experts. The insights gained from this project can be further expanded and refined to develop more robust and accurate models, ultimately contributing to the overall success and growth of organizations.