Employee Attrition Analysis and Prediction

This project aims to provide insights into the factors influencing employee attrition and predict which employees are likely to leave the company.

Problem Statement:

Acme Corporation, a leading tech company, is facing a significant challenge with employee turnover. The HR department is concerned about the increasing rate of attrition, as it negatively impacts team dynamics, project continuity, and overall company morale. To address this issue, Acme Corporation wants to leverage data analytics and machine learning to understand the factors influencing employee turnover and predict which employees are likely to leave in the near future.

Dataset:

Acme Corporation has provided historical data on employee demographics, job satisfaction, work environment, performance metrics, and turnover status. This dataset spans the last five years and includes information on employees who have left the company and those who are still currently employed.

The dataset typically includes several features that provide insights into employee characteristics, job satisfaction, and performance. While the exact features may vary, here's a general list of common features you might find in such a dataset:

Employee ID: A unique identifier for each employee.

Age: The age of the employee.

Attrition: A binary variable indicating whether the employee has left the company (1) or is still employed (0).

Business Travel: The frequency and nature of business-related travel (e.g., "Travel Rarely," "Travel_Frequently," "Non-Travel").

Department: The department to which the employee belongs (e.g., "Sales," "Research & Development," "Human Resources").

Distance From Home: The distance of the employee's residence from the workplace.

Education: The employee's level of education (e.g., "1: 'Below College'," "2: 'College'," "3: 'Bachelor'," "4: 'Master'," "5: 'Doctor').

Education Field: The field in which the employee's education lies (e.g., "Life Sciences," "Medical," "Marketing").

Environment Satisfaction: The level of satisfaction with the work environment on a scale.

Gender: The gender of the employee.

Job Involvement: The degree to which the employee is involved in their job.

Job Level: The level or rank of the employee's position.

Job Role: The specific role or title of the employee's job.

Job Satisfaction: The level of satisfaction with the job on a scale.

Marital Status: The marital status of the employee.

Monthly Income: The monthly salary of the employee.

Num Companies Worked: The number of companies the employee has worked for.

Over Time: Whether the employee works overtime or not.

Performance Rating: The performance rating of the employee.

Relationship Satisfaction: The level of satisfaction with relationships at the workplace.

Stock Option Level: The level of stock options provided to the employee.

Total Working Years: The total number of years the employee has been working.

Training Times Last Year: The number of training sessions the employee attended last year.

Work-Life Balance: The balance between work and personal life.

Years At Company: The number of years the employee has been with the current company.

Years In Current Role: The number of years the employee has been in their current role.

Years Since Last Promotion: The number of years since the last time the employee was promoted.

Years With Current Manager: The number of years the employee has been working under the current manager.

Please note that this is a general list, and the actual dataset might include additional features or variations. It's essential to explore the dataset thoroughly to understand the specifics of each feature and its relevance to the analysis.

Data Exploration:

Data Loading and Displaying:

The code starts by importing necessary libraries such as pandas, matplotlib, and seaborn.
It reads a CSV file containing HR employee data into a DataFrame named df.
The first few rows of the DataFrame are displayed using df.head().
Summary Statistics and Value Counts:

Summary statistics for numerical columns are displayed using df.describe().
Value counts for categorical variables such as BusinessTravel, Department, EducationField, Gender, JobRole, MaritalStatus, Over18, and OverTime are displayed using df['column_name'].value_counts().
Data Visualization:

Histograms and line plots are created using seaborn and matplotlib to visualize distributions and relationships within the data.
Histograms for 'Age', 'MonthlyIncome', and 'TotalWorkingYears' are displayed.
A line plot showing the change in monthly income across different age groups is presented.
Additional visualizations include histograms and a scatter plot showing the distribution of total working years and the relationship between age and monthly income, respectively.
The code also adds annotations such as mean and median lines to some of the visualizations for better interpretation.
Insights:

From the summary statistics, we can observe the central tendency and dispersion of numerical features like age, daily rate, total working years, etc.
The value counts provide insights into the distribution of categorical variables such as business travel frequency, department, education field, etc.

Visualizations help in understanding the distributions and relationships within the data. For example, the scatter plot shows that attrition tends to occur more frequently among younger employees with lower monthly incomes.
Overall, these analyses and visualizations provide a comprehensive understanding of the HR employee dataset, allowing for further exploration and insights into factors contributing to attrition and other HR-related metrics.

Data Loading and Displaying:

The code starts by importing necessary libraries: pandas, matplotlib.pyplot, and seaborn.
Then, it loads a dataset from a CSV file located at the given path using pd.read_csv().
The first few rows of the dataset are displayed using df.head().
Summary Statistics:

Summary statistics of numerical columns are printed using df.describe(). This includes count, mean, standard deviation, minimum, maximum, and quartiles for each numerical column.

Value Counts for Categorical Variables:

The frequency of unique values in certain categorical columns is printed using value_counts() method for each column of interest. Columns include 'MonthlyRate', 'DailyRate', 'Attrition', 'BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus', 'Over18', and 'OverTime'.
Visualization:

Various visualizations are created using seaborn and matplotlib.pyplot libraries.
Histograms for 'Age', 'MonthlyIncome', and 'TotalWorkingYears' are plotted using sns.histplot().
A line plot depicting the change in 'MonthlyIncome' across 'Age' is created using sns.lineplot().
Additional histograms for 'TotalWorkingYears' are plotted with different visual styles such as color, grid, and annotations for mean and median.

Additional Visualization:

A scatter plot showing 'Age' vs. 'MonthlyIncome' with points colored by 'Attrition' is created using sns.scatterplot(). This allows visualizing the relationship between age, monthly income, and attrition status.
Overall, the code performs data exploration tasks including loading the dataset, displaying summary statistics, examining the distribution of numerical variables, and visualizing relationships between variables using various plots.

Data Cleaning.

Handling Missing Values:

First, the dataset is checked for missing values using the .isnull().sum() method, which returns the count of missing values in each column.
Then, missing values are dropped using the .dropna() method with inplace=True to modify the DataFrame in place.
Removing Duplicate Rows:

Duplicate rows are removed using the .drop_duplicates() method with inplace=True to modify the DataFrame in place.
Data Type Conversion (Optional):

Optionally, you may want to convert data types if needed. However, in the provided code, there's no explicit conversion of data types.
Final Check and Saving the Cleaned Dataset:

After cleaning, the final check for data types is performed using the .dtypes attribute.
The cleaned dataset is saved to a CSV file using the .to_csv() method with index=False to exclude row indices from the saved file.
These steps ensure that the dataset is cleaned from missing values and duplicates, making it ready for further analysis or modeling. Additionally, saving the cleaned dataset allows for easy access and reuse of the cleaned data.
Handling Missing Values: The dataset did not contain any missing values, so no imputation or removal of missing values was required.

Data Type Conversion: The data types were already appropriate for most columns. However, if there were any columns with incorrect data types (e.g., numerical columns mistakenly stored as strings), they would need to be converted to the correct data type.

Removing Duplicate Rows: Duplicate rows were removed from the dataset to ensure each observation is unique.

Saving the Cleaned Dataset: Finally, the cleaned dataset was saved to a new CSV file named "cleaned_dataset.csv" for further analysis.

Insights:

The cleaned dataset contains information about employee attributes such as age, attrition status, business travel frequency, daily rate, department, distance from home, education level, education field, gender, job role, marital status, monthly income, and various other factors related to the employee's job and workplace environment.

With the dataset cleaned and ready for analysis, further exploration can be conducted to gain insights into factors influencing employee attrition, job satisfaction, performance, and other relevant metrics. This analysis could help organizations identify patterns and take proactive measures to improve employee retention and satisfaction

Data Encoding

In the provided code snippets, various data encoding techniques have been applied to prepare the dataset for machine learning tasks. Here's a summary of the encoding techniques used:

One-Hot Encoding: This technique is applied to categorical variables to convert them into binary vectors. Each category is transformed into a binary feature, where a value of 1 indicates the presence of that category and 0 indicates absence. It has been applied to columns such as 'BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus', and 'OverTime'.

Label Encoding: Label encoding is used to convert categorical variables into numerical labels. Each unique category is assigned a unique integer. It has been applied to the 'Attrition' column, where 'Yes' is encoded as 1 and 'No' is encoded as 0.

Normalization: Numerical features are often normalized to ensure that all features have the same scale. Min-Max scaling, a type of normalization, has been applied to numerical columns such as 'Age', 'DailyRate', 'DistanceFromHome', 'HourlyRate', 'MonthlyIncome', 'PercentSalaryHike', 'TotalWorkingYears', and others. This scaling technique rescales the features to a fixed range between 0 and 1.

Train-Test Split: Finally, the dataset has been split into training and testing sets using the train_test_split function from the sklearn.model_selection module. This step is essential for evaluating the performance of machine learning models on unseen data.

Overall, these encoding techniques ensure that the dataset is in a suitable format for training machine learning models, with categorical variables appropriately encoded and numerical features scaled to a uniform range.

After encoding categorical variables and scaling numerical features, the dataset is now ready for further analysis and modeling. Here are the key steps that were performed:

One-Hot Encoding Categorical Variables: Categorical variables such as 'BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus', and 'OverTime' were one-hot encoded to convert them into numerical format suitable for machine learning models.

Label Encoding Target Variable: The target variable 'Attrition' was label encoded using sklearn's LabelEncoder to convert the 'Yes' and 'No' values into numerical format (0 and 1).

Scaling Numerical Features: Numerical features such as 'Age', 'DailyRate', 'DistanceFromHome', 'HourlyRate', 'MonthlyIncome', 'PercentSalaryHike', 'TotalWorkingYears', 'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion', and 'YearsWithCurrManager' were scaled using MinMaxScaler to ensure all features have the same scale, which is important for many machine learning algorithms.

Splitting the Dataset: The dataset was split into training and testing sets using sklearn's train_test_split function, with 80% of the data used for training and 20% for testing.

With the dataset prepared and split into training and testing sets, the next steps would involve building machine learning models to predict employee attrition and evaluate their performance using the testing set. This could include various classification algorithms such as logistic regression, random forest, or gradient boosting classifiers. Additionally, model evaluation metrics such as accuracy, precision, recall, and F1-score could be used to assess the model's performance.

Data Labelling

code provided seems to demonstrate how to label data based on certain conditions. Here's an explanation of each part of the code:

Reading the Dataset:

The code starts by importing the necessary libraries: pandas for data manipulation and matplotlib. pyplot and seaborn for visualization.
It then reads the dataset from a CSV file located at 'F:\Technopolis\WA_Fn-UseC_-HR-Employee-Attrition.csv' using the pd. read_csv() function and assigns it to a DataFrame called df.

Defining Labelling Functions:

Several labelling functions are defined to assign labels based on specific conditions:
label function: Labels each row based on the 'Age' column. If the age is greater than 30, it assigns 'labella'; otherwise, it assigns 'label'.
label attrition: Labels each row based on the 'Attrition' column. If the value is 'Yes', it assigns 'High Attrition'; otherwise, it assigns 'Low Attrition'.
label income: Labels each row based on the 'MonthlyIncome' column. If the monthly income is greater than 5000, it assigns 'High Income'; otherwise, it assigns 'Low Income'.

Applying Labelling Functions:

The labelling functions are applied to the DataFrame using the apply () function along the specified axis (usually axis=1, which applies the function row-wise).
Displaying the Labelled DataFrame:

The code prints the first few rows of the DataFrame to verify the labelling.
Further Explanation:

The code demonstrates different approaches to labelling data based on various conditions such as age, attrition status, and income level.
These labelled data can be useful for exploratory data analysis, visualization, or predictive modeling tasks.

The code provided performs data labelling based on certain conditions and creates additional columns for the labelled data. Here's a summary of the insights from this code:

Data Loading: The code first loads a dataset from a CSV file using pandas.

Labelling Based on Age and Department:

It defines a function label function that assigns labels ('labella', 'label', etc.) based on conditions related to the 'Age' and 'Department' columns.
This function is applied to each row of the DataFrame to create a new column called 'label'.
Labelling Attrition and Income:

Two functions (label attrition and label income) are defined to label the 'Attrition' and 'MonthlyIncome' columns, respectively, based on specific conditions.
These functions are applied to each row of the DataFrame to create new columns called 'Attrition Label' and 'Income Label'.
Displaying the Results:

The code prints the first few rows of the DataFrame with the newly added labels for verification.
Insights:

With the labelled data, further analysis can be performed, such as understanding the distribution of employees based on age, department, attrition rate, and income level.
These labels can be utilized in various analyses, such as identifying patterns related to attrition, income, and their relationship with other factors in the dataset.
Overall, the provided code facilitates data understanding and analysis by adding meaningful labels to the dataset based on specified condition.