

EMPLOYEE ATTRITION ANALYSIS REPORT

-by Nithish Gangineni

1.INTRODUCTION:

Employee Attrition means employees who are exiting or resigning from a company or organization voluntarily. The dataset used in this project is HR Employee Attrition dataset from Kaggle. It contains 1470 rows and 35 columns. Key attributes include Age, Gender, Job Role, Monthly Income, Distance from home, Overtime, Job Satisfaction and Attrition as Target variable.

Employee attrition refers to the gradual loss of employees over time. High attrition rates lead to increased recruitment costs, loss of organizational knowledge and reduced productivity.

So to address this we are using DataScience and Machine Learning techniques to predict and understand the factors that contribute to employee attrition.

2.OBJECTIVE:

The main objective of this project is to develop a predictive model that helps HR department anticipate employee turnover and take proactive measures to retain valuable talent. This process involves several important steps such as:

- Data Exploration
- Data Cleaning
- Data Preprocessing
- Label Encoding
- Train-Test splits
- Decision tree classifier

3.TOOLS & TECHNOLOGIES USED:

- ✓ Programming Language: Python
- ✓ Libraries: Pandas, Numpy, Matplotlib, Seaborn, Scikit-Learn, Graphviz
- ✓ IDE: Google Colab

PROJECT

1.IMPORTING REQUIRED LIBRARIES:

We need to import all the required libraries for data manipulation, visualization, and processing.

- **Pandas:**
A library for data manipulation and analysis, offering data structures like DataFrames
- **Numpy:**
The fundamental package for numerical computing in python, providing support for arrays and matrices
- **Matplotlib:**
A library for creating static , animated, and interactive visualization 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 that provides simple and efficient tools for data mining and data analysis
- **Graphviz:**

It open-source graph visualization software that represents structural information as abstract graph and network diagrams.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2.UPLOADING & READING DATASET:

For this project I am using HR Employee Attrition dataset from Kaggle. In order to load a dataset we use `read.csv()`. Here we need to mention the path of the dataset inside the parenthesis, so that the notebook reads the dataset and loads it.

```
data=pd.read_csv("C:\\Users\\Nithish\\Documents\\Employee_attrition.csv")
```

`df.head()`: Used to see first five rows of the dataset. Where df represents the dataframe and head represents the default first five rows of the dataset, you can mention the required number of rows in the parenthesis.

```
print(df.head())
```

	Age	Attrition	BusinessTravel	DailyRate	Department	
0	41	Yes	Travel_Rarely	1102		Sales
1	49	No	Travel_Frequently	279		Research & Development
2	37	Yes	Travel_Rarely	1373		Research & Development
3	33	No	Travel_Frequently	1392		Research & Development
4	27	No	Travel_Rarely	591		Research & Development

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

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

`df.shape()`: Used to check the dimensions of the dataset i.e rows and columns.

```
print(df.shape)#basic health of the data/health check
```

```
(1470, 35)
```

`df.dtypes()`: Used to find datatype of the factors in the dataset

```
print(df.dtypes)
```

Age	int64
Attrition	object
BusinessTravel	object
DailyRate	int64
Department	object
DistanceFromHome	int64
Education	int64
EducationField	object
EmployeeCount	int64
EmployeeNumber	int64
EnvironmentSatisfaction	int64
Gender	object
HourlyRate	int64
JobInvolvement	int64
JobLevel	int64
JobRole	object
JobSatisfaction	int64
MaritalStatus	object
MonthlyIncome	int64
MonthlyRate	int64
NumCompaniesWorked	int64
Over18	object

3.EXPLORATORY DATA ANALYSIS(EDA):

EDA is a process like searching for clues and patterns in the data to understand what's going on. In this EDA, we use two plots namely boxplot and count plot.

- BOX PLOT: It is a standardized way of displaying the distribution of data based on a five number summary(minimum, first quartile[Q1], median, third quartile[Q3], and maximum)

```
sns.boxplot(data=df,x="DistanceFromHome")
```

To visualize how far employees live from their work place and identify potential outliers. Where outliers are large commuting distances might contribute to higher attrition rates.

- COUNT PLOT: Here count plot is used to understand how many employees have stayed and how many left. This gives an overall balance of classes.

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

4.DATA CLEANING:

Some columns in the dataset have null values. So we will drop those irrelevant columns.

```
df.drop(['EmployeeCount','EmployeeNumber','StandardHours'],axis='columns',inplace=True)
```

This line of code removes four specific columns from the DataFrame df: 'EmployeeCount', 'EmployeeNumber', 'Over18', and 'StandardHours'.

axis="columns": This specifies that the operation should be performed on columns .

inplace=True: This modifies the DataFrame df directly, rather than returning a new DataFrame object.

```
categorical_col=[]
for column in df.columns:
    if df[column].dtype== object and len(df[column].unique())<=50:
        categorical_col.append(column)
```

This block of code iterates through all remaining columns in the DataFrame to identify potential categorical variables.

categorical_col = []: Initializes an empty list to store the names of identified categorical columns.

for column in df.columns:: Starts a loop that goes through each column name in the DataFrame.

if df[column].dtype == 'object' Checks if the data type of the column is 'object', which typically means it contains strings or mixed data types.

len(df[column].unique()) <= 50:: Further filters the columns. It checks if the number of *unique* values in the column is less than or equal to 50. data.

categorical_col.append(column): If a column meets both conditions, its name is added to the categorical_col list.

Remove “Attrition “ from the categorical_col because we need analyze the attrition of the Employee.so,we don’t want to encode the attrition column.

5.Label Encoding:

```
df['Attrition']=df.Attrition.astype("category").cat.codes #label encoding
```

This line transforms the 'Attrition' column from categorical labels into numerical codes. This process is known as categorical encoding or label encoding.

Label Encoding can be done by using LabelEncoder() which is present in sklearn library.We use for loop to traverse each and every column present in categorical_col .Then, we use fit_transform() method to analyzes the unique categories in that specific column and replaces the original text categories with the assigned numerical values for every column in the dataframe.

6. Train-Test splitting:

In this step , we use train_test_split() method from sklearn.model_selection. X represents the features, predictors, or independent variables. This is the input data that the model uses to learn patterns and make a prediction. Y represents the target variable, label, or dependent variable. This is the output data that the 'Attrition' status, or whether an employee stays or leaves their job is found. This method divides the dataset in 80:20 or 70:30 based on the test_size.The ratio determines that train: test data for the model.Here,we are using 70% of the data to train(X_train,Y_train)and remaining 30% is used for test(X_test,Y_test). The use of random_state=42 ensures this specific split is exactly the same every time the code runs, making the experiment repeatable and reliable.

```
from sklearn.model_selection import train_test_split
x=df.drop('Attrition',axis=1)
y=df.Attrition
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
```

7.Decision Tree classifier:

We use accuracy_score, confusion_matrix, classification_report, and DecisionTreeClassifier from sklearn.mertics.The function print_score is used to evaluate the performance of a classifier either on training or on test data and then uses this function to assess a DecisionTreeClassifier model.First, the print_score function takes a trained classifier, data, and a boolean flag called train. If True, it predicts on the training data, X_train, and prints out the training results: an accuracy score. If False, it does the same set of evaluations but on the test data, X_test, to show how well the model generalizes to unseen data.Then initialize a DecisionTreeClassifier, fits it to the training data, and then calls the print_score function, passing train=True to print its score on the training data, then calling print_score again with train=False to print its score on the testing data. This is a common way to check that training scores are not significantly higher than test scores, which would suggest overfitting.

```
from sklearn.tree import DecisionTreeClassifier

tree_clf=DecisionTreeClassifier(random_state=42)
tree_clf.fit(x_train,y_train)

print_score(tree_clf,x_train,x_test,y_train,y_test,train=True)
print_score(tree_clf,x_train,x_test,y_train,y_test,train=False)
```

RESULTS AND OBSERVATIONS

- The DecisionTreeClassifier successfully predicts the employee attrition
- Model performance is evaluated using accuracy, confusion matrix, and F1-score
- Key factors like distance from home ,monthly income,jobrole ,and job satisfaction are likely strong predictors
- The Decision Tree model performed well on the training data but may overfit slightly on test data. This can be improved using pruning or ensemble methods like Random Forest.

```

Train Result:
=====
Accuracy Score:100.00%
-----
Classification_Report:
      0      1  accuracy  macro avg  weighted avg
precision    1.0    1.0      1.0      1.0      1.0
recall       1.0    1.0      1.0      1.0      1.0
f1-score     1.0    1.0      1.0      1.0      1.0
support     853.0  176.0      1.0    1029.0    1029.0
-----
Confusion Matrix:
[[853   0]
 [  0 176]]

```

CONCLUSION:

This project helps identify the key factors that influence why employees leave a company. By applying data analysis and machine learning, we can predict employee attrition and help HR teams take steps to retain valuable staff. The insights gained from this analysis support better decision-making, reduce turnover, and improve overall employee satisfaction.