**Name:** Manish Shashikant Jadhav

**UID:** 2023301005

**Batch:** TE Comps-B Batch B

**Experiment No. 7**

import pandas as pd import numpy as np

from matplotlib import pyplot as plt import seaborn as sns # read csv file

df=pd.read\_csv('/content/Employee.csv')

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Education** | **JoiningYear** | **City** | **PaymentTier** | **Age** | **Gender** | **EverBenched** | **ExperienceInCurrentDomain** | **LeaveOrNot** |
| **0** Bachelors | 2017 | Bangalore | 3 | 34 | Male | No | 0 | 0 |
| **1** Bachelors | 2013 | Pune | 1 | 28 | Female | No | 3 | 1 |
| **2** Bachelors | 2014 | New Delhi | 3 | 38 | Female | No | 2 | 0 |
| **3** Masters | 2016 | Bangalore | 3 | 27 | Male | No | 5 | 1 |
| **4** Masters | 2017 | Pune | 3 | 24 | Male | Yes | 2 | 1 |

|  |  |  |
| --- | --- | --- |
|  |  |  |

df.head()

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Education** | **JoiningYear** | **City** | **PaymentTier** | **Age** | **Gender** | **EverBenched** | **ExperienceInCurrentDomain** | **LeaveOrNot** |
| **4648** | Bachelors | 2013 | Bangalore | 3 | 26 | Female | No | 4 | 0 |
| **4649** | Masters | 2013 | Pune | 2 | 37 | Male | No | 2 | 1 |
| **4650** | Masters | 2018 | New Delhi | 3 | 27 | Male | No | 5 | 1 |
| **4651** | Bachelors | 2012 | Bangalore | 3 | 30 | Male | Yes | 2 | 0 |
| **4652** | Bachelors | 2015 | Bangalore | 3 | 33 | Male | Yes | 4 | 0 |

df.tail()

df.shape

(4653, 9)

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 4653 entries, 0 to 4652 Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

1. Education 4653 non-null object
2. JoiningYear 4653 non-null int64
3. City 4653 non-null object
4. PaymentTier 4653 non-null int64
5. Age 4653 non-null int64
6. Gender 4653 non-null object
7. EverBenched 4653 non-null object
8. ExperienceInCurrentDomain 4653 non-null int64 8 LeaveOrNot 4653 non-null int64 dtypes: int64(5), object(4) memory usage: 327.3+ KB

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **count mean**  **std min**  **25%** | **JoiningYear**  4653.000000  2015.062970  1.863377  2012.000000  2013.000000 | **PaymentTier**  4653.000000  2.698259  0.561435  1.000000  3.000000 | **Age**  4653.000000  29.393295  4.826087  22.000000  26.000000 | **ExperienceInCurrentDomain**  4653.000000  2.905652  1.558240  0.000000  2.000000 | **LeaveOrNot**  4653.000000  0.343864  0.475047  0.000000  0.000000 |
| **50%** | 2015.000000 | 3.000000 | 28.000000 | 3.000000 | 0.000000 |
| **75%** | 2017.000000 | 3.000000 | 32.000000 | 4.000000 | 1.000000 |
| **max** | 2018.000000 | 3.000000 | 41.000000 | 7.000000 | 1.000000 |

df.describe()

df.isnull().sum()

**0**

**Education** 0

**JoiningYear** 0

**City** 0

**PaymentTier** 0

**Age** 0

**Gender** 0

**EverBenched** 0

**ExperienceInCurrentDomain** 0

**LeaveOrNot** 0

|  |  |  |
| --- | --- | --- |
|  |  |  |

**dtype:** int64

from sklearn.preprocessing import LabelEncoder

label\_encoder = LabelEncoder() categorial\_columns = ['Gender', 'EverBenched', 'Education', 'City']

#Apply label encoding to each categorial column for column in categorial\_columns: df[column] = label\_encoder.fit\_transform(df[column])

x = df.drop('LeaveOrNot', axis=1) y = df['LeaveOrNot'] 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)

from sklearn.ensemble import RandomForestClassifier model = RandomForestClassifier(n\_estimators=100,random\_state=42)



y\_pred = model.predict(x\_test)

from sklearn.metrics import accuracy\_score

accuracy = accuracy\_score(y\_test, y\_pred) print(f"Accuracy: {accuracy \* 100:.2f}%")

Accuracy: 84.96%

from sklearn.metrics import classification\_report print(classification\_report(y\_test, y\_pred)) precision recall f1-score support

0 0.86 0.92 0.89 610 1 0.82 0.72 0.77 321

accuracy 0.85 931 macro avg 0.84 0.82 0.83 931 weighted avg 0.85 0.85 0.85 931

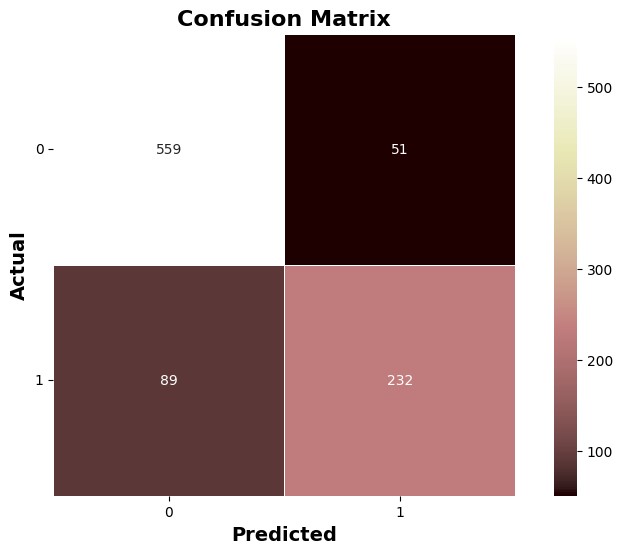
from sklearn.metrics import confusion\_matrix cm = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(10, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='pink', linewidths=0.4, square=True, cbar=True, xticklabels=["0", "1"], yticklabels=["0", "1"]

)

)

plt.xlabel('Predicted', fontsize=14, fontweight='bold') plt.ylabel('Actual', fontsize=14, fontweight='bold') plt.title('Confusion Matrix', fontsize=16, fontweight='bold') plt.yticks(rotation=360)



plt.show

(

)

# Example of a model that may be underfitting from sklearn.tree import DecisionTreeClassifier

#Assume X\_train and y\_train are your training features and labels clf=DecisionTreeClassifier(max\_depth=1)



accuracy = accuracy\_score(y\_test, y\_predict) print(f"Accuracy: {accuracy \* 100:.2f}%")

Accuracy: 74.22%

print(classification\_report(y\_test, y\_predict)) precision recall f1-score support

0 0.72 1.00 0.84 610 1 1.00 0.25 0.40 321

accuracy 0.74 931 macro avg 0.86 0.63 0.62 931 weighted avg 0.81 0.74 0.69 931

from sklearn.naive\_bayes import GaussianNB

# Build a Gaussian Classifier model = GaussianNB()

# Train the classifier on the training data



y\_pred1 = model.predict(x\_test)

accuracy = accuracy\_score(y\_test, y\_pred1) print(f"Accuracy: {accuracy \* 100:.2f}%")

Accuracy: 68.53%

#importing SVM

from sklearn.svm import SVC model\_svm = SVC(kernel = 'linear', random\_state = 0)



accuracy = accuracy\_score(y\_test, y\_pred) print(f"Accuracy: {accuracy \* 100:.2f}%")

Accuracy: 68.42%