Experiment 7

November 9, 2023

Name: Adwait Purao UID: 2021300101 Batch: B2

[ ]:

**import numpy as np import pandas as pd**

**import matplotlib.pyplot as plt**

%matplotlib inline

[ ]:

df = pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/Employee.csv")

[ ]:

df.head()

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

[ ]:

missing\_data = df.isna() missing\_counts = missing\_data.sum() print(missing\_counts)

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()

df['Education'] = label\_encoder.fit\_transform(df['Education']) df.head()

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [ | ]: |  | Education | JoiningYear | City | | PaymentTier Age Gender | | EverBenched | | \ |
|  |  | 0 | 0 | 2017 | Bangalore | | 3 34 Male | | No | |  |
|  |  | 1 | 0 | 2013 | Pune | | 1 28 Female | | No | |  |
|  |  | 2 | 0 | 2014 | New Delhi | | 3 38 Female | | No | |  |
|  |  | 3 | 1 | 2016 | Bangalore | | 3 27 Male | | No | |  |
|  |  | 4 | 1 | 2017 | Pune | | 3 24 Male | | Yes | |  |
| ExperienceInCurrentDomain LeaveOrNot | | | | | | | | | | | |
| 0 | | | 0 | | | | 0 | | | | |
| 1 | | | 3 | | | | 1 | | | | |
| 2 | | | 2 | | | | 0 | | | | |
| 3 | | | 5 | | | | 1 | | | | |
| 4 | | | 2 | | | | 1 | | | | |
| [ | ]: | df['City'] = label\_encoder.fit\_transform(df['City']) df.head() | | | | | | | | | |
| [ | ]: |  | Education | JoiningYear | City | PaymentTier Age Gender | | EverBenched | | \ | |
|  |  | 0 | 0 | 2017 | 0 | 3 34 Male | | No | |  | |
|  |  | 1 | 0 | 2013 | 2 | 1 28 Female | | No | |  | |
|  |  | 2 | 0 | 2014 | 1 | 3 38 Female | | No | |  | |
|  |  | 3 | 1 | 2016 | 0 | 3 27 Male | | No | |  | |
|  |  | 4 | 1 | 2017 | 2 | 3 24 Male | | Yes | |  | |

ExperienceInCurrentDomain LeaveOrNot

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0 |
| 1 | 3 | 1 |
| 2 | 2 | 0 |
| 3 | 5 | 1 |
| 4 | 2 | 1 |

[ ]:

df['Gender'] = label\_encoder.fit\_transform(df['Gender']) df.head()

[ ]: Education JoiningYear City PaymentTier Age Gender EverBenched \ 0 0 2017 0 3 34 1 No

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 2013 | 2 | 1 | 28 | 0 | No |
| 2 | 0 | 2014 | 1 | 3 | 38 | 0 | No |
| 3 | 1 | 2016 | 0 | 3 | 27 | 1 | No |
| 4 | 1 | 2017 | 2 | 3 | 24 | 1 | Yes |
| ExperienceInCurrentDomain LeaveOrNot | | | | | | | |

[ ]:

df['EverBenched'] = label\_encoder.fit\_transform(df['EverBenched']) df.head()

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0 |
| 1 | 3 | 1 |
| 2 | 2 | 0 |
| 3 | 5 | 1 |
| 4 | 2 | 1 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [ | ]: | Education | JoiningYear | City | PaymentTier | Age | Gender | EverBenched | \ |
|  |  | 0 0 | 2017 | 0 | 3 | 34 | 1 | 0 |  |
|  |  | 1 0 | 2013 | 2 | 1 | 28 | 0 | 0 |  |
|  |  | 2 0 | 2014 | 1 | 3 | 38 | 0 | 0 |  |
|  |  | 3 1 | 2016 | 0 | 3 | 27 | 1 | 0 |  |
|  |  | 4 1 | 2017 | 2 | 3 | 24 | 1 | 1 |  |

|  |  |
| --- | --- |
| ExperienceInCurrentDomain | LeaveOrNot |
| 0 0 | 0 |
| 1 3 | 1 |
| 2 2 | 0 |
| 3 5 | 1 |
| 4 2 | 1 |

[ ]:

**from sklearn.model\_selection import** train\_test\_split

[ ]:

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4)

[ ]:

**from sklearn.metrics import** classification\_report,confusion\_matrix

[ ]:

**from sklearn.naive\_bayes import** GaussianNB

[ ]:

nb\_classifier = GaussianNB() nb\_classifier.fit(X\_train, y\_train) y\_naive\_bayes = nb\_classifier.predict(X\_test)

[ ]:

print("The confusion matrix for Naive Bayes is : ") print("") print(confusion\_matrix(y\_test,y\_naive\_bayes))

[ ]:

The confusion matrix for Naive Bayes is :

[[978 238]

[374 272]]

print("The classification report for Naive Bayes is : ") print("") print(classification\_report(y\_test,y\_naive\_bayes))

[ ]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.72 | 0.80 | 0.76 | 1216 |
| 1 | 0.53 | 0.42 | 0.47 | 646 |
| accuracy |  |  | 0.67 | 1862 |
| macro avg | 0.63 | 0.61 | 0.62 | 1862 |
| weighted avg | 0.66 | 0.67 | 0.66 | 1862 |

[ ]:

The classification report for Naive Bayes is :

**from sklearn.tree import** DecisionTreeClassifier

[ ]:

dtree = DecisionTreeClassifier() dtree.fit(X\_train,y\_train) y\_decision\_tree = dtree.predict(X\_test)

print("The confusion matrix for Decision Tree is : ") print("") print(confusion\_matrix(y\_test,y\_decision\_tree))

[ ]:

The confusion matrix for Decision Tree is :

[[1060 156]

[ 203 443]]

print("The classification report for Decision Tree is : ") print("") print(classification\_report(y\_test,y\_decision\_tree))

The classification report for Decision Tree is :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.84 | 0.87 | 0.86 | 1216 |
| 1 | 0.74 | 0.69 | 0.71 | 646 |
| accuracy |  |  | 0.81 | 1862 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| macro avg | 0.79 | 0.78 | 0.78 | 1862 |
| weighted avg | 0.80 | 0.81 | 0.81 | 1862 |

[ ]:

**from sklearn.ensemble import** RandomForestClassifier

[ ]:

rfc = RandomForestClassifier(n\_estimators=1000) rfc.fit(X\_train,y\_train)

y\_random\_forest = rfc.predict(X\_test)

[ ]:

print("The confusion matrix for Random Forest is : ") print("") print(confusion\_matrix(y\_test,y\_random\_forest))

[ ]:

The confusion matrix for Random Forest is :

[[1103 113]

[ 209 437]]

print("The classification report for Decision Tree is : ") print("") print(classification\_report(y\_test,y\_random\_forest))

[ ]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.84 | 0.91 | 0.87 | 1216 |
| 1 | 0.79 | 0.68 | 0.73 | 646 |
| accuracy |  |  | 0.83 | 1862 |
| macro avg | 0.82 | 0.79 | 0.80 | 1862 |
| weighted avg | 0.82 | 0.83 | 0.82 | 1862 |

[ ]:

accuracy\_nb = accuracy\_score(y\_test, y\_naive\_bayes) accuracy\_dtc = accuracy\_score(y\_test, y\_decision\_tree) accuracy\_rfc = accuracy\_score(y\_test, y\_random\_forest)

print(f'Accuracy for Naive Bayes: **{**accuracy\_nb**:**.2f**}**') print(f'Accuracy for Decision Tree 1: **{**accuracy\_dtc**:**.2f**}**') print(f'Accuracy for Random Forest Classifier 2: **{**accuracy\_rfc**:**.2f**}**')

The classification report for Decision Tree is :

**from sklearn.metrics import** accuracy\_score

Accuracy for Naive Bayes: 0.67 Accuracy for Decision Tree 1: 0.81

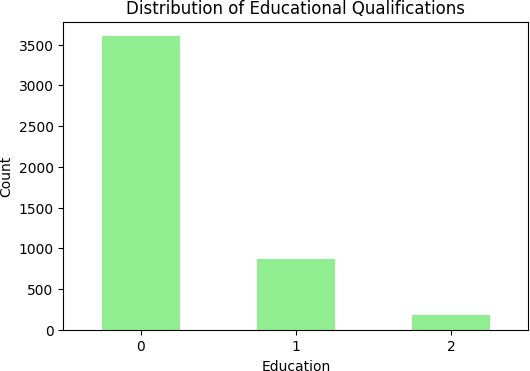
Accuracy for Random Forest Classifier 2: 0.83

# Q1. What is the distribution of educational qualifications among employees?

[ ]:

education\_counts = df['Education'].value\_counts() plt.figure(figsize=(6, 4)) education\_counts.plot(kind='bar', color='lightgreen') plt.title('Distribution of Educational Qualifications') plt.xlabel('Education')

plt.ylabel('Count') plt.xticks(rotation=0) plt.show()



# Q2. How does the length of service (Joining Year) vary across different cities?

[ ]:

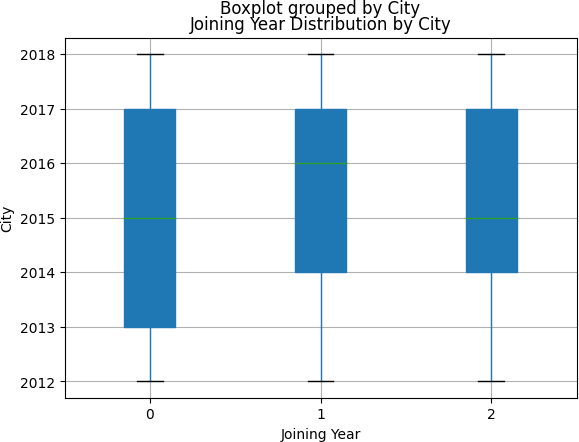
plt.figure(figsize=(10, 6))

df.boxplot(column='JoiningYear', by='City', patch\_artist=**True**) plt.title('Joining Year Distribution by City') plt.xlabel('Joining Year')

plt.ylabel('City')

plt.show()

<Figure size 1000x600 with 0 Axes>



# Q3. Is there a correlation between Payment Tier and Experience in Current Domain?

[ ]:

correlation = df['PaymentTier'].corr(df['ExperienceInCurrentDomain']) print(f"Pearson's Correlation Coefficient: **{**correlation**:**.2f**}**")

**if** correlation > 0:

interpretation = "There is a positive correlation."

**elif** correlation < 0:

interpretation = "There is a negative correlation."

**else**:

interpretation = "There is no linear correlation."

print(interpretation)

Pearson's Correlation Coefficient: 0.02 There is a positive correlation.

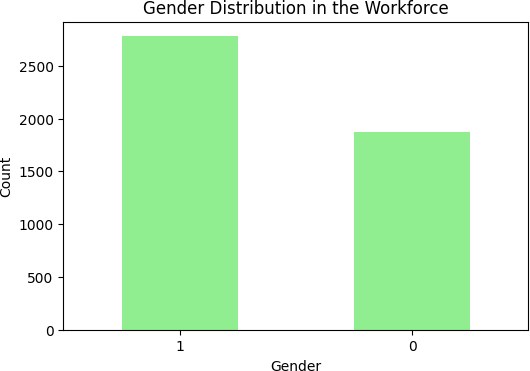
# Q4. What is the gender distribution within the workforce?

[ ]:

gender\_counts = df['Gender'].value\_counts() plt.figure(figsize=(6, 4)) gender\_counts.plot(kind='bar', color='lightgreen') plt.title('Gender Distribution in the Workforce') plt.xlabel('Gender')

plt.ylabel('Count') plt.xticks(rotation=0)

plt.show()

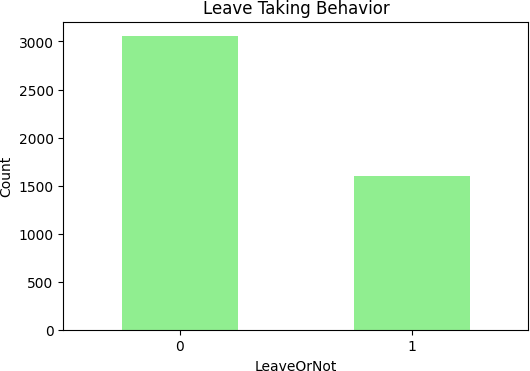


# Q5. Are there any patterns in leave-taking behavior among employees?

[ ]:

leave\_counts = df['LeaveOrNot'].value\_counts() plt.figure(figsize=(6, 4)) leave\_counts.plot(kind='bar', color='lightgreen') plt.title('Leave Taking Behavior') plt.xlabel('LeaveOrNot')

plt.ylabel('Count') plt.xticks(rotation=0) plt.show()



[ ]:

correlation\_leave = df.corr()['LeaveOrNot'].drop('LeaveOrNot') print(correlation\_leave)

|  |  |
| --- | --- |
| Education | 0.080497 |
| JoiningYear | 0.181705 |
| City | 0.201058 |
| PaymentTier | -0.197638 |
| Age | -0.051126 |
| Gender | -0.220701 |
| EverBenched | 0.078438 |
| ExperienceInCurrentDomain | -0.030504 |

Name: LeaveOrNot, dtype: float64

[ ]:

plt.scatter(df['ExperienceInCurrentDomain'], df['LeaveOrNot'], alpha=0.5) plt.title('Experience in Current Domain vs. Leave Taking') plt.xlabel('Experience in Current Domain')

plt.ylabel('LeaveOrNot') plt.show()

df.boxplot(column='PaymentTier', by='LeaveOrNot', patch\_artist=**True**) plt.title('PaymentTier by Leave Taking Behavior') plt.xlabel('PaymentTier')

plt.ylabel('LeaveOrNot')

plt.show()

