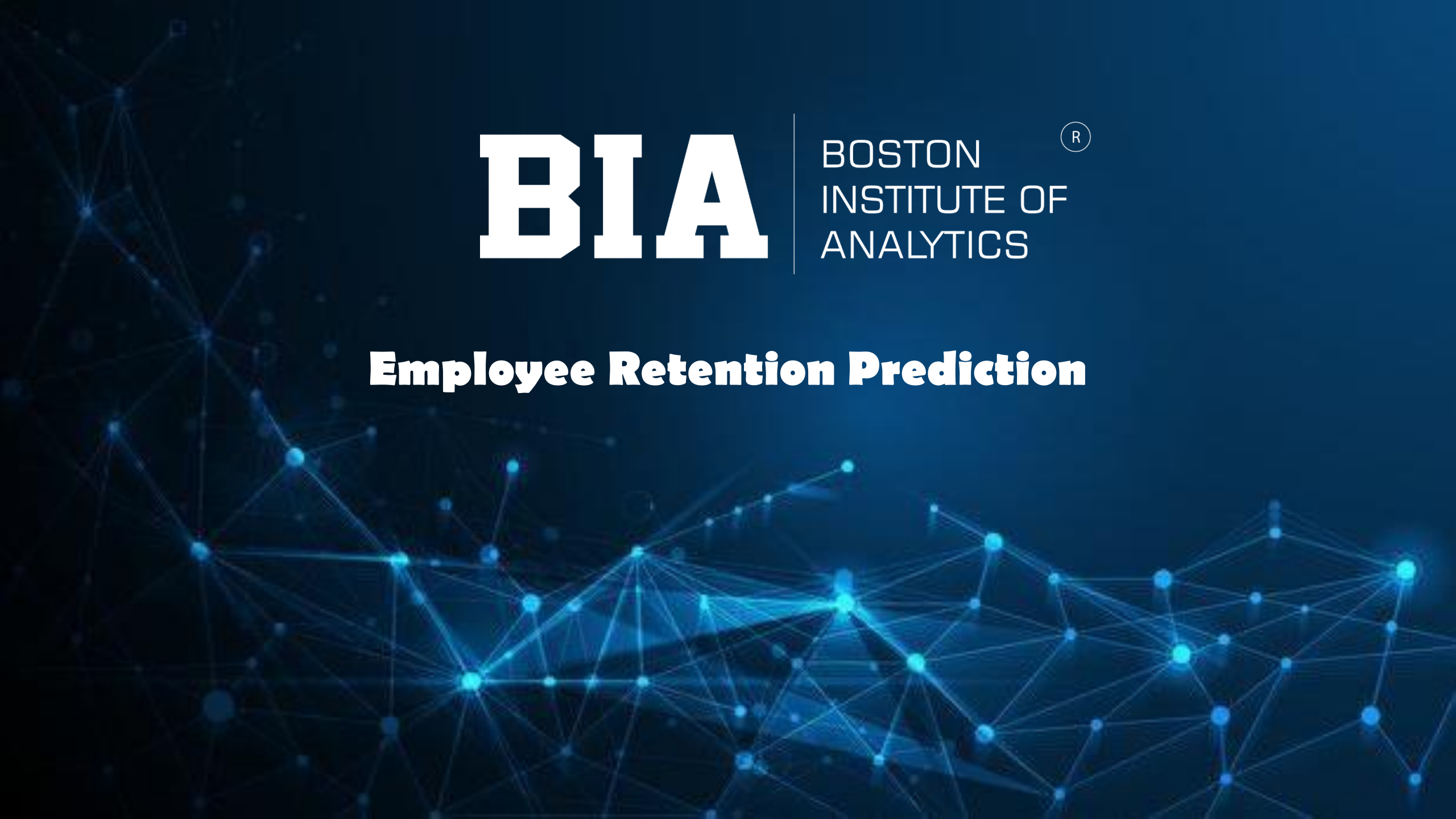
BIA | BOSTON INSTITUTE OF ANALYTICS ®

Employee Retention Prediction



Employee Retention Prediction Agenda

Objective :

Predict whether a data scientist is looking for a job change.

Help HR teams in optimizing retention strategies and recruitment planning.

Data Preprocessing :

Handling missing values (mode imputation for categorical features)

Label encoding for categorical variables

Feature scaling (if needed for some models)

Business Impact :

Helps HR proactively identify at-risk talent

Optimizes training, hiring, and employee engagement

Reduces attrition cost through early intervention

Introduction

Employee retention is a critical concern for organizations, particularly in the fast-paced and highly competitive field of data science. High turnover rates among skilled professionals can lead to disruptions in project continuity, increased hiring costs, and loss of intellectual capital.

This project aims to develop a **machine learning model** that predicts whether a data scientist is likely to seek a job change. By leveraging employee attributes such as education, experience, company characteristics, and job preferences, the model can help Human Resources (HR) and management teams proactively identify employees at risk of leaving.

The insights derived from this model will enable:

- **Proactive retention strategies**
- **Efficient workforce planning**
- **Optimized resource allocation for upskilling and engagement**

By integrating predictive analytics into HR decision-making, organizations can not only reduce attrition rates but also improve employee satisfaction and long-term productivity.

BIA BOSTON INSTITUTE OF ANALYTICS ®

THE WORK FLOW OF THE PROJECT IS GIVEN BELOW :

- Importing Important Libraries.
- Data Overview.
- EDA (Exploratory Data Analysis).
- Data Preprocessing.
- Modeling.
- Evaluation.

Project Goal

The goal of this project is to build a robust machine learning model that predicts whether a data scientist is likely to look for a job change.
By accurately identifying potential attrition, the model will support:

- **Improved employee retention strategies**
- **Proactive recruitment planning**
- **Efficient HR resource allocation**

Project Plan

Data Understanding

- Review the structure and quality of the training and test datasets.
- Identify key features related to employee behavior and intent.

Exploratory Data Analysis (EDA)

- Analyze distributions, correlations, and patterns in the data.
- Visualize relationships between features and the target

Feature Engineering

- Transform ordinal features (e.g., experience, last_new_job)
- Create new derived features if relevant
- Remove irrelevant or redundant columns

Model Development

Train classification model :

- Logistic Regression

Model Evaluation

Here's a complete **Model Evaluation code** for **classification problems** using Scikit-learn. This includes:

✅ Accuracy
✅ Confusion Matrix
✅ Classification Report
✅ ROC-AUC Score
✅ ROC Curve (optional)

Importing Important Libraries

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

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Data Overview

```
data.head()
```

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	NaN	
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15	50-99	Pv
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5	NaN	
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	<1	NaN	Pv
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	>20	50-99	Funded St

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   enrollee_id             19158 non-null  int64  
 1   city                    19158 non-null  object 
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object 
 4   relevent_experience     19158 non-null  object 
 5   enrolled_university     18772 non-null  object 
 6   education_level         18698 non-null  object 
 7   major_discipline        16345 non-null  object 
 8   experience              19093 non-null  object 
 9   company_size            13220 non-null  object 
 10  company_type            13018 non-null  object 
 11  last_new_job            18735 non-null  object 
 12  training_hours          19158 non-null  int64  
 13  target                  19158 non-null  float64
dtypes: float64(2), int64(2), object(10)
memory usage: 2.0+ MB
```

Data Summary

- Total rows: **19,158**
- Total columns: **14**
- Column types:
  - **10 categorical (object)**
  - **2 integers (int64)**
  - **2 floats (float64)**

BIA BOSTON INSTITUTE OF ANALYTICS ®

```
data.shape
```

```
(19158, 14)
```

```
data.isna().sum()  #isna().sum() use for detect missing values
```

```
enrollee_id                0
city                       0
city_development_index     0
gender                  4508
relevent_experience        0
enrolled_university      386
education_level          460
major_discipline        2813
experience                65
company_size            5938
company_type            6140
last_new_job             423
training_hours             0
target                     0
dtype: int64
```

Find null values

```
(data.isnull().sum()/data.shape[0])*100
```

```
enrollee_id              0.000000
city                     0.000000
city_development_index   0.000000
gender                  23.530640
relevent_experience      0.000000
enrolled_university      2.014824
education_level          2.401086
major_discipline        14.683161
experience               0.339284
company_size            30.994885
company_type            32.049274
last_new_job             2.207955
training_hours           0.000000
target                   0.000000
dtype: float64
```

BIA BOSTON INSTITUTE OF ANALYTICS ®

Drop null values

```
data.dropna(inplace=True)
```

```
data.isnull().sum()
```

```
enrollee_id             0
city                    0
city_development_index  0
gender                  0
relevent_experience     0
enrolled_university     0
education_level         0
major_discipline        0
experience              0
company_size            0
last_new_job            0
training_hours          0
target                  0
dtype: int64
```

Remove null values

Clean data → better predictions

```
data.head()
```

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	last_new
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15	50-99	
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	>20	50-99	
7	402	city_46	0.762	Male	Has relevent experience	no_enrollment	Graduate	STEM	13	<10	
8	27107	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	7	50-99	
11	23853	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	5	5000-9999	

Exploratory Data Analysis (EDA)

```
data.describe()
```

	enrollee_id	city_development_index	training_hours	target
count	9417.000000	9417.000000	9417.000000	9417.000000
mean	16906.814697	0.844092	65.110120	0.164915
std	9914.920028	0.116086	60.268856	0.371123
min	2.000000	0.448000	1.000000	0.000000
25%	8253.000000	0.794000	23.000000	0.000000
50%	16993.000000	0.910000	47.000000	0.000000
75%	25839.000000	0.920000	88.000000	0.000000
max	33380.000000	0.949000	336.000000	1.000000

```
data.columns
```

```
Index(['enrollee_id', 'city', 'city_development_index', 'gender',
       'relevent_experience', 'enrolled_university', 'education_level',
       'major_discipline', 'experience', 'company_size', 'last_new_job',
       'training_hours', 'target'],
      dtype='object')
```

Only Numerical Data

```
corr_data = data[['enrollee_id','city_development_index','training_hours','target']]
corr_data.head()
```

	enrollee_id	city_development_index	training_hours	target
1	29725	0.776	47	0.0
4	666	0.767	8	0.0
7	402	0.762	18	1.0
8	27107	0.920	46	1.0
11	23853	0.920	108	0.0

Correlation only Numerical Data

BIA BOSTON INSTITUTE OF ANALYTICS ®

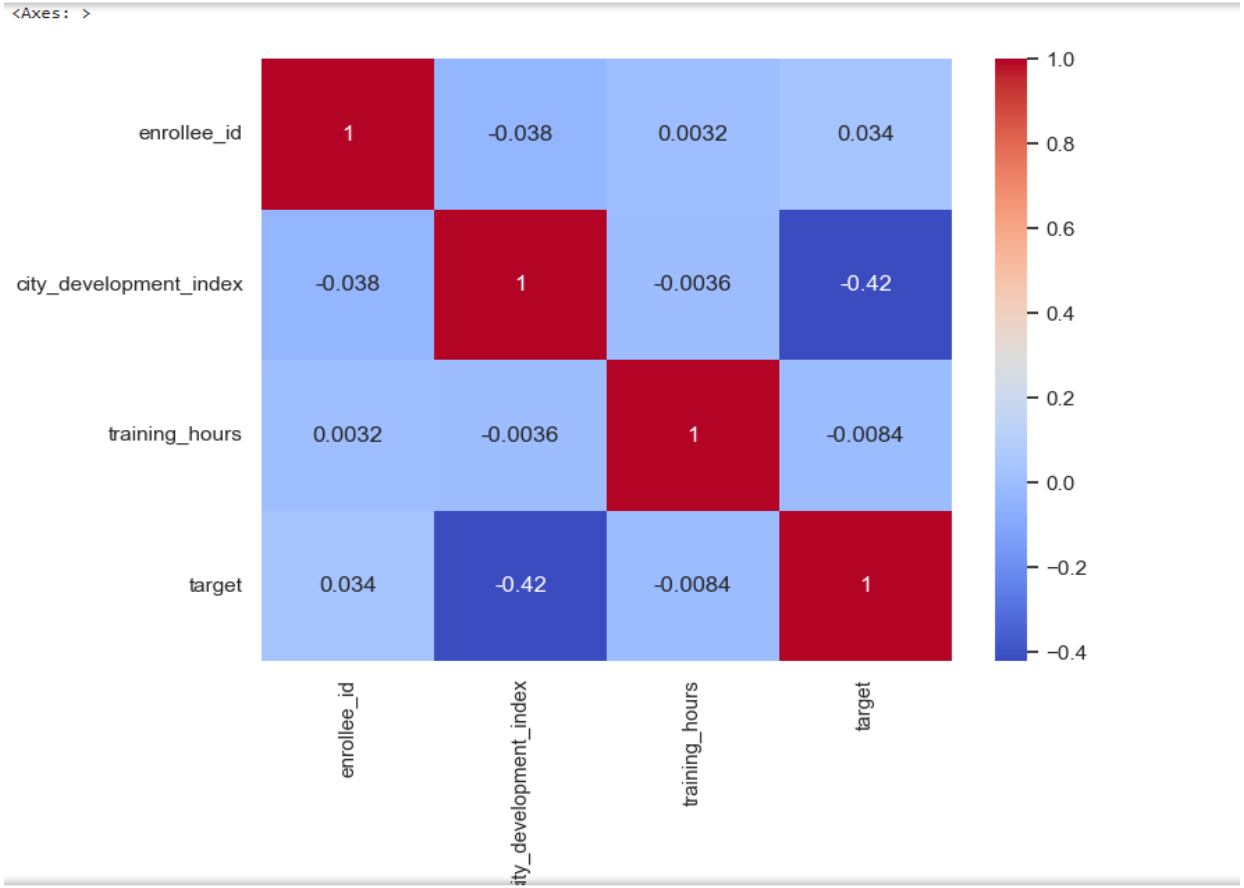
Seaborn Correlation Heatmap

```
corr_data.corr?
```

```
[1;31mSignature:[0m ...
```

Seaborn Correlation Heatmap

```
plt.figure(figsize=(8,6))
sns.heatmap(corr_data.corr(), annot=True, cmap='coolwarm')
```

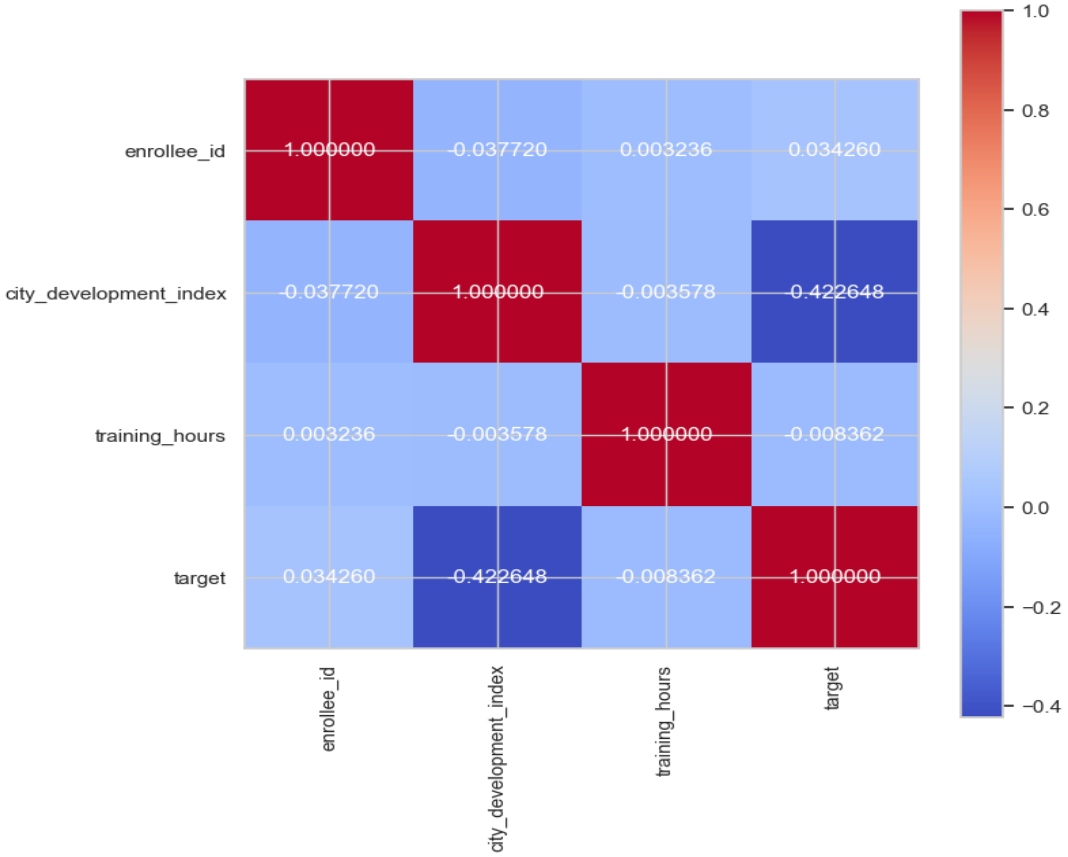


BIA BOSTON INSTITUTE OF ANALYTICS ®

Matplotlib Correlation Heatmap

Matplotlib Correlation Heatmap

```
plt.imshow(corr_data.corr(),cmap='coolwarm', interpolation='none')
plt.colorbar()
plt.xticks(range(len(corr_data.columns)), corr_data.columns, rotation=90)
plt.yticks(range(len(corr_data.columns)), corr_data.columns)
plt.gcf().set_size_inches(8,8)

labels = corr_data.corr().values
for y in range(labels.shape[0]):
    for x in range (labels.shape[1]):
        plt.text(x, y, '{:2f}'.format(labels[y, x]), ha='center', va='center', color='white')
```



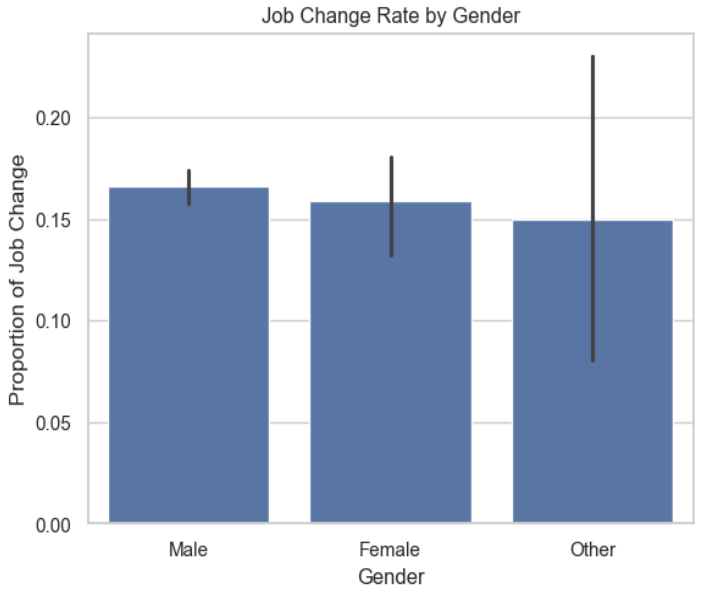
BIA BOSTON INSTITUTE OF ANALYTICS ®

Plots :

- **Bar Plots**

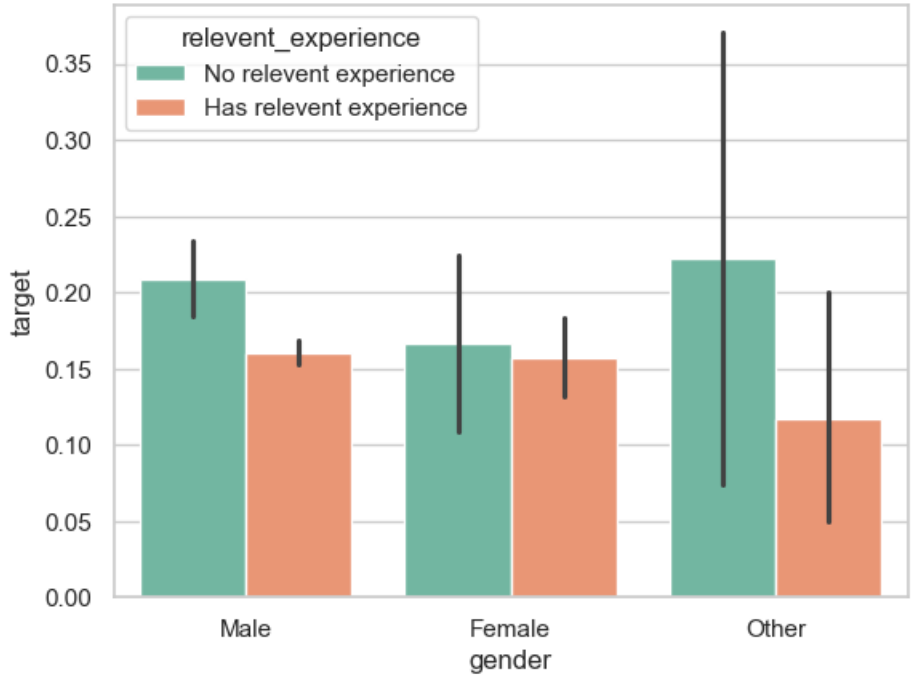
Barplot

```
]: sns.barplot(x='gender', y='target', data=data)
   plt.title("Job Change Rate by Gender")
   plt.xlabel("Gender")
   plt.ylabel("Proportion of Job Change")
   plt.show()
```



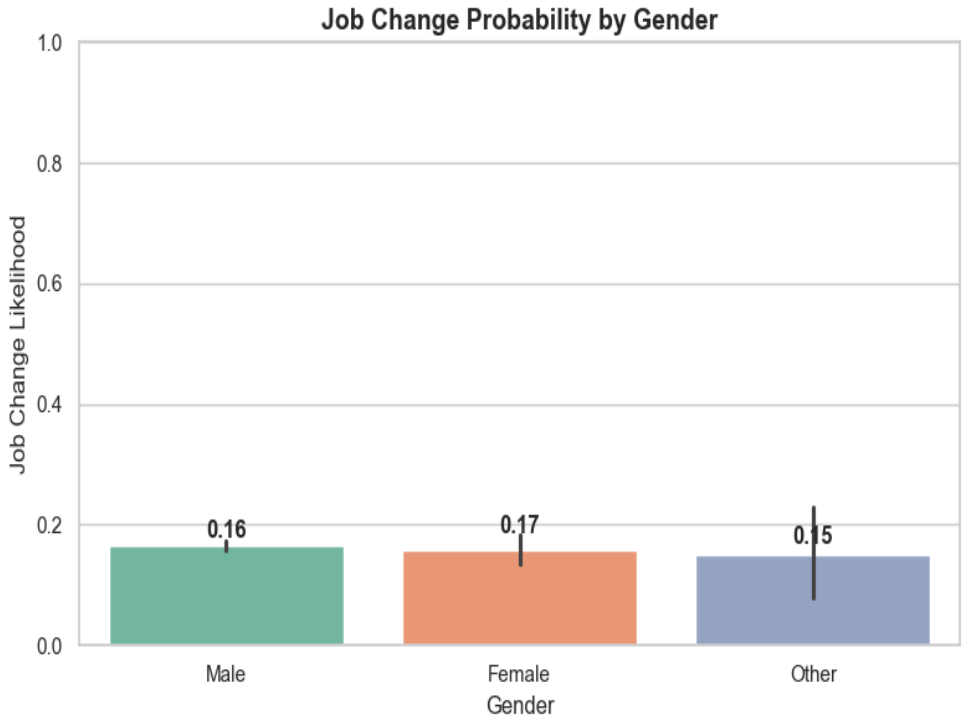

```
sns.barplot(x='gender', y='target', hue='relevent_experience', data=data, palette='Set2')
```

```
<Axes: xlabel='gender', ylabel='target'>
```



BIA BOSTON INSTITUTE OF ANALYTICS ®

```python
sns.set(style="whitegrid")  # other options: "darkgrid", "ticks", "white", etc.

# Create the barplot
plt.figure(figsize=(8, 5))  # adjust figure size
sns.barplot(x='gender', y='target', data=data, palette='Set2')  # try 'pastel', 'Set1', 'coolwarm'

# Add title and labels
plt.title("Job Change Probability by Gender", fontsize=14, fontweight='bold')
plt.xlabel("Gender", fontsize=12)
plt.ylabel("Job Change Likelihood", fontsize=12)

# Show percentages on top of bars
for i, val in enumerate(data.groupby('gender')['target'].mean()):
    plt.text(i, val + 0.02, f"{val:.2f}", ha='center', fontweight='bold')

plt.ylim(0, 1)  # set y-limit for clarity
plt.tight_layout()
plt.show()*
```



This visualization using **Seaborn** and **Matplotlib**. It presents a **bar plot** that compares the **probability of job change** across different **gender categories** ("Male", "Female", "Other").

What the Chart Shows:

- **Title**: *Job Change Probability by Gender*
- **X-axis**: Gender
- **Y-axis**: Job Change Likelihood (values range from 0 to 1)
- **Bars**: Represent the **average likelihood of changing jobs** for each gender category.
- **Text on Bars**: Displays the numerical percentage values for easier interpretation.
- **Values**:
  - **Male**: 0.16 (16%)
  - **Female**: 0.17 (17%)
  - **Other**: 0.15 (15%)

Why This is Important:

- **Workforce Insights**: Helps HR professionals or organizational analysts understand if there's a gender-based trend in employee turnover.
- **Bias Detection**: Identifies whether certain groups are more likely to leave, possibly due to workplace dissatisfaction, lack of growth opportunities, or bias.
- **Policy Making**: Guides companies to take targeted action (e.g., support for underrepresented groups or employee retention strategies).
- **Diversity & Inclusion Monitoring**: Ensures all gender identities are considered in organizational analysis, not just male/female.

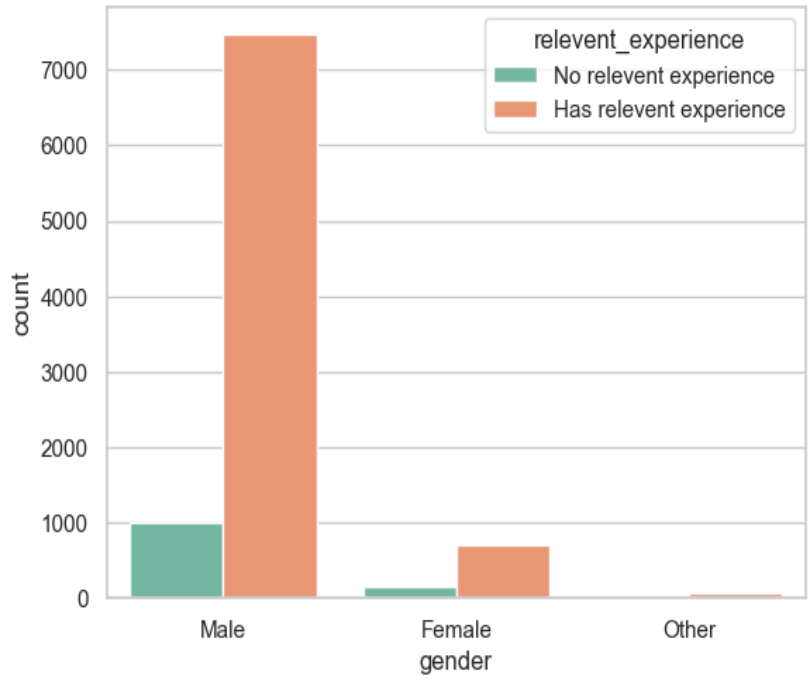
What You Can Gather from It:

- All gender groups have relatively similar job change likelihood (around 15-17%), suggesting no major imbalance in this specific dataset.
- The company or organization likely tracks gender and job change as part of its HR analytics.
- A small but visible difference might still be relevant depending on the sample size or business context.
- Would you like help interpreting this in a report or presentation format?

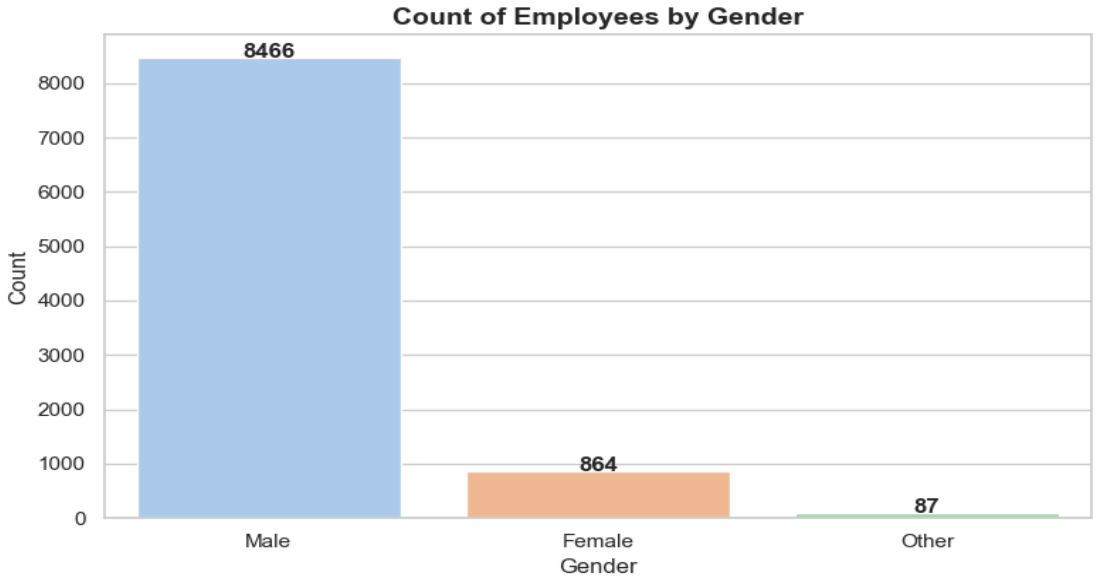
- **Count Plots**

```
sns.countplot(x='gender', hue='relevent_experience', data=data, palette='Set2')
```

```
<Axes: xlabel='gender', ylabel='count'>
```

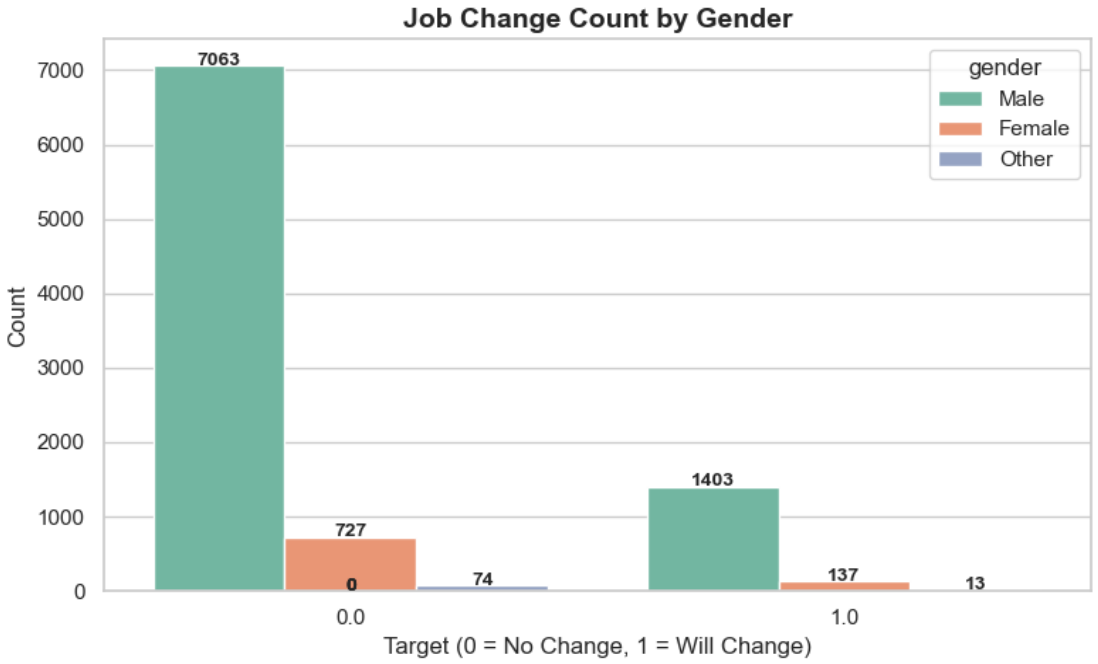



```
# Set Seaborn style
sns.set(style="whitegrid")

# Set figure size
plt.figure(figsize=(8, 5))

# Create the countplot
sns.countplot(x='gender', data=data, palette='pastel')

# Add labels and title
plt.title("Count of Employees by Gender", fontsize=14, fontweight='bold')
plt.xlabel("Gender", fontsize=12)
plt.ylabel("Count", fontsize=12)

# Show values on top of bars
gender_counts = data['gender'].value_counts()
for i, val in enumerate(gender_counts):
    plt.text(i, val + 5, str(val), ha='center', fontweight='bold')

plt.tight_layout()
plt.show()
```



BIA BOSTON INSTITUTE OF ANALYTICS ®

```python
# Set Seaborn style
sns.set(style="whitegrid")

# Set figure size
plt.figure(figsize=(8, 5))

# Create the countplot
sns.countplot(x='target', hue='gender', data=data, palette='Set2')

# Add labels and title
plt.title("Job Change Count by Gender", fontsize=14, fontweight='bold')
plt.xlabel("Target (0 = No Change, 1 = Will Change)", fontsize=12)
plt.ylabel("Count", fontsize=12)

# Add value annotations on bars
# We loop through bars and put the height as text
for p in plt.gca().patches:
    height = int(p.get_height())
    plt.gca().annotate(f'{height}', (p.get_x() + p.get_width() / 2, height + 5),
                       ha='center', fontsize=10, fontweight='bold')

plt.tight_layout()
plt.show()
```



Count plot created with **Seaborn** and **Matplotlib** in Python. The plot visualizes the **number of people who changed jobs** (target = 1) or **did not change jobs** (target = 0), broken down by **gender**.

What the Chart Shows:

X-axis: Target

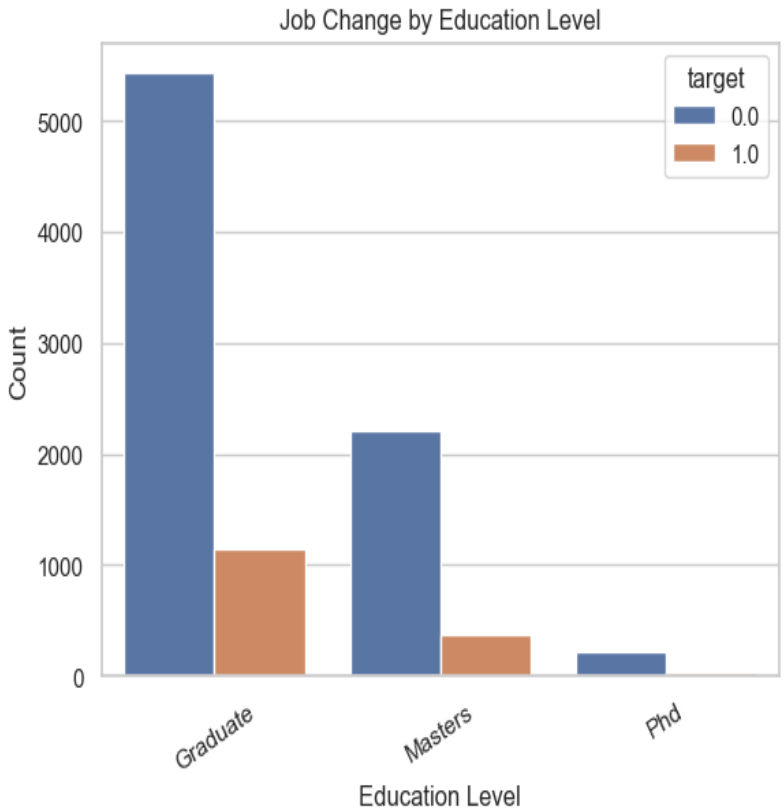
- 0 = No job change
- 1 = Will change job
- **Y-axis**: Count (number of individuals)
- **Hue**: Gender (Male, Female, Other)
- **Bar Values**: Each bar is labeled with its count value.

Count Breakdown:

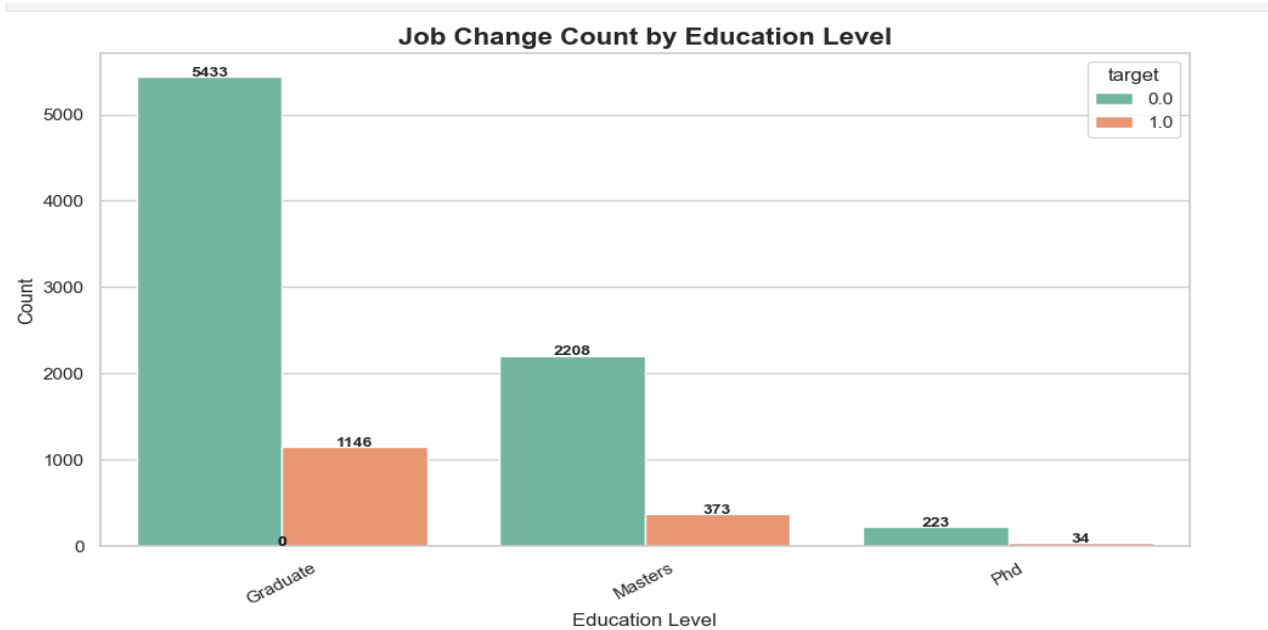
Target	Gender	Count
0 (No Change)	Male	7063
0 (No Change)	Female	727
0 (No Change)	Other	74
1 (Will Change)	Male	1403
1 (Will Change)	Female	137
1 (Will Change)	Other	13

BIA BOSTON INSTITUTE OF ANALYTICS

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='education_level', hue='target', data=data)

plt.title("Job Change by Education Level")
plt.xlabel("Education Level")
plt.ylabel("Count")
plt.xticks(rotation=30)  # Rotate x labels for clarity
plt.show()
```




```
# Set plot style
sns.set(style="whitegrid")

# Set figure size
plt.figure(figsize=(10, 6))

# Create countplot
sns.countplot(x='education_level', hue='target', data=data, palette='Set2')

# Add title and labels
plt.title("Job Change Count by Education Level", fontsize=16, fontweight='bold')
plt.xlabel("Education Level", fontsize=12)
plt.ylabel("Count", fontsize=12)
plt.xticks(rotation=30)  # Rotate x labels for better readability

# Annotate values on top of bars
for p in plt.gca().patches:
    height = int(p.get_height())
    plt.gca().annotate(f'{height}', (p.get_x() + p.get_width()/2, height + 2),
                       ha='center', fontsize=10, fontweight='bold')

plt.tight_layout()
plt.show()
```



BIA BOSTON INSTITUTE OF ANALYTICS

Why This is Important:

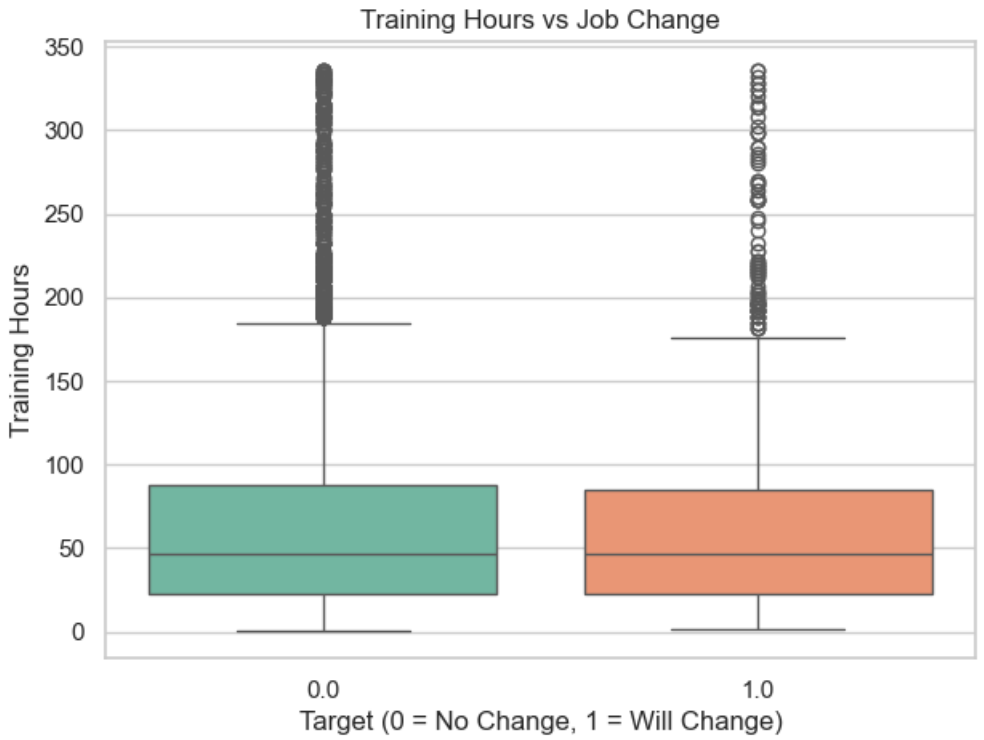
- **Raw Frequency Insight**: While the first chart showed the *probability*, this chart shows the *actual number of people* in each category, which is crucial for understanding the **scale** of job change behavior.
- **Context for Percentages**: Percentages alone can be misleading if group sizes are unequal. This chart shows that there are **far more males** in the dataset, which explains why their absolute job change numbers are also higher.
- **Gender Representation**: It highlights the **distribution of gender** in the dataset and shows that males dominate, with much smaller counts for female and other genders.
- **Balanced Comparison**: This helps prevent misinterpretation of patterns due to sample size imbalance—important in HR analytics, diversity analysis, and fair policy-making.

What You Can Gather from It:

- **Most individuals in the dataset are male**.
- **Males also make up the majority of those who will change jobs**, but this is expected due to their population size.
- **Females and 'Other' gender groups are underrepresented**, which could reflect real-world workforce trends or data collection biases.
- A full analysis requires looking at **both probability and count**, as done here with this chart and the one before it.
- Would you like a combined summary or recommendation based on both plots?

Box Plot

```
plt.figure(figsize=(7, 5))
sns.boxplot(x='target', y='training_hours', data=data, palette='Set2')

plt.title("Training Hours vs Job Change")
plt.xlabel("Target (0 = No Change, 1 = Will Change)")
plt.ylabel("Training Hours")
plt.show()
```



This boxplot shows the relationship between **training hours** and **job change behavior**:

- **X-axis**: Job change target (0 = No Change, 1 = Will Change)
- **Y-axis**: Training hours
- **Observation**:
  - Both groups (no change and will change) have **similar median training hours**.
  - There is a **wide spread** and **many outliers** in both categories.
  - Training hours alone **do not strongly distinguish** between employees who stay and those who leave.

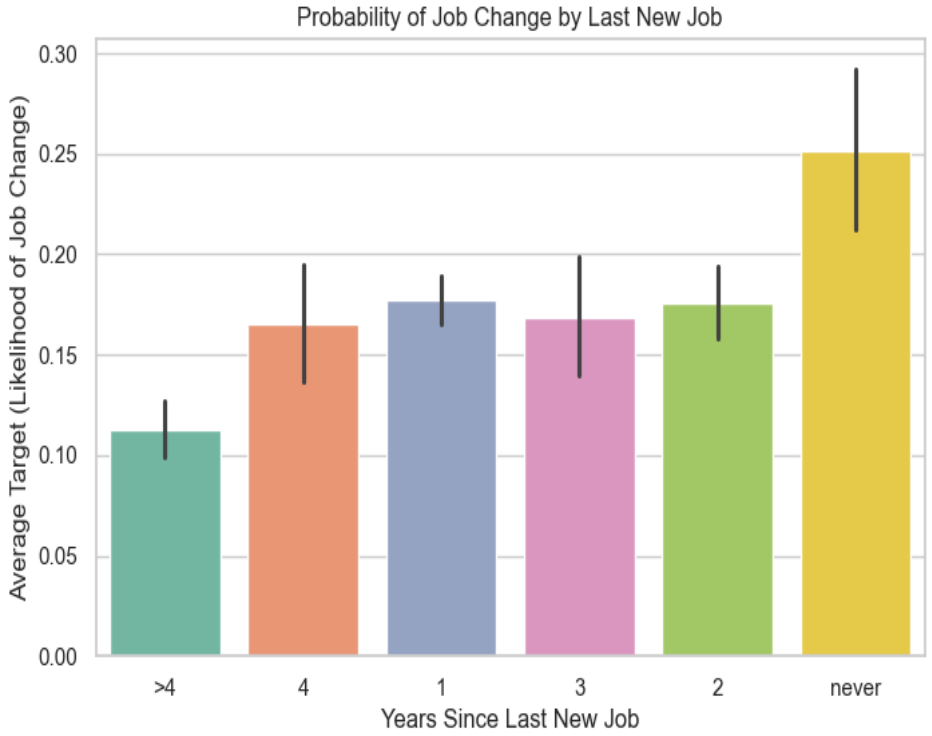
In Short:

Training hours are not a clear predictor of job change — people with both low and high training hours are found in both categories.

BIA BOSTON INSTITUTE OF ANALYTICS

Bar Plot

```
plt.figure(figsize=(8, 5))
sns.barplot(x='last_new_job', y='target', data=data, palette='Set2')

plt.title("Probability of Job Change by Last New Job")
plt.xlabel("Years Since Last New Job")
plt.ylabel("Average Target (Likelihood of Job Change)")
plt.show()
```



This chart shows how likelihood of job change varies by years since last new job:

In Short:

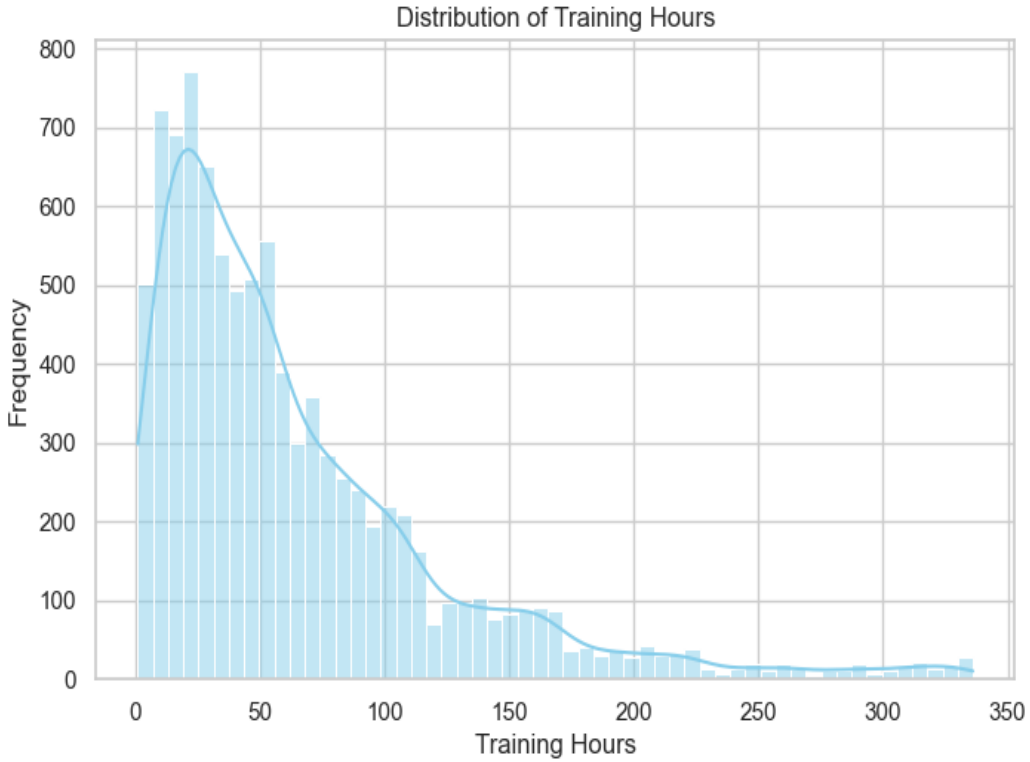
Employees who have never had a new job before are most likely to change jobs, while those whose last new job was over 4 years ago are least likely to change.

- **"Never" has the highest likelihood (~25%) of job change.**
- **Likelihood increases with fewer years since last new job.**
- **Indicates that less job experience or recent job entry correlates with a higher chance of leaving.**

BIA BOSTON INSTITUTE OF ANALYTICS

Bar Plot

```
# Example: Distribution of training_hours
plt.figure(figsize=(8, 5))
sns.histplot(data['training_hours'], kde=True, color='skyblue')

plt.title("Distribution of Training Hours")
plt.xlabel("Training Hours")
plt.ylabel("Frequency")
plt.show()
```



In Short:

Most employees have low training hours, with the number sharply dropping as training hours increase.

- The distribution is **right-skewed** (long tail to the right).
- Majority of training hours fall **below 100 hours**.
- Only a few employees have **very high training hours** (200+).
- This suggests training is concentrated among a smaller portion of employees.

BIA BOSTON INSTITUTE OF ANALYTICS

Model Training (Baseline + Advanced Models)

Logistic Regression Model Train and test

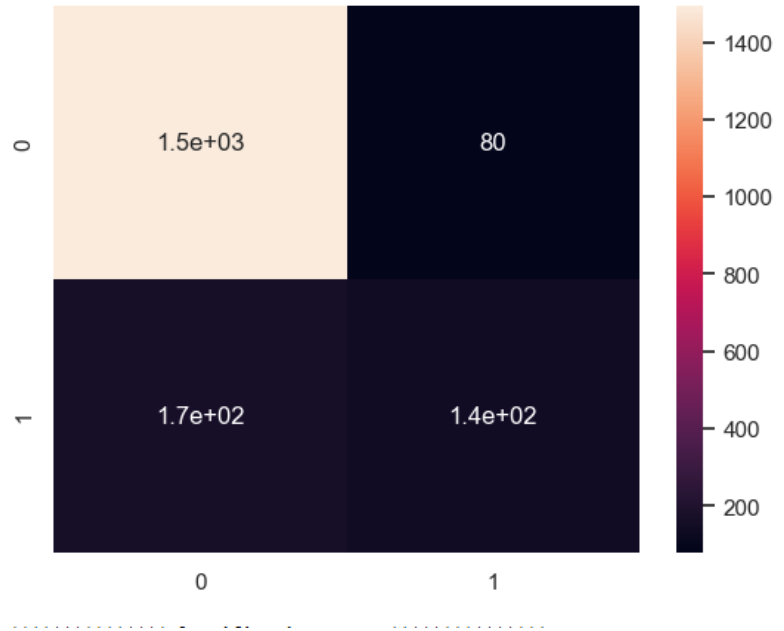
```
def training_model(models, X_train, X_test, y_train, y_test):
    for modelparams in models:
        print('***************', modelparams['Name'], '***************')

        model = modelparams['model']
        model.fit(X_train, y_train)

        train_predict = model.predict(X_train)
        print('***************Training Accuracy***************')
        print(accuracy_score(y_train, train_predict))

        test_predict = model.predict(X_test)
        print('***************Testing Accuracy***************')
        print(accuracy_score(y_test, test_predict))

        print('***************Confusion Matrix***************')
        cn_matrix = confusion_matrix(y_test, test_predict)
        sns.heatmap(cn_matrix, annot=True)
        plt.show()

        print('***************Classification Report***************')
        print(classification_report(y_test, test_predict))
        print('----------------------------------------------------')
        print('====================================================')
        print()
```

```
models = [
    {'Name': 'LogisticRegression', 'model': LogisticRegression()},
    {'Name': 'RandomForestClassifier', 'model': RandomForestClassifier()},
    {'Name': 'DecisionTreeClassifier', 'model': DecisionTreeClassifier()}]
```

```
training_model(models, X_train, X_test, y_train, y_test)
```

```
training_model(models, X_train, X_test, y_train, y_test)
```

```
*************** LogisticRegression ***************
***************Training Accuracy***************
0.855170582769149
***************Testing Accuracy***************
0.8683651804670913
***************Confusion Matrix***************
```

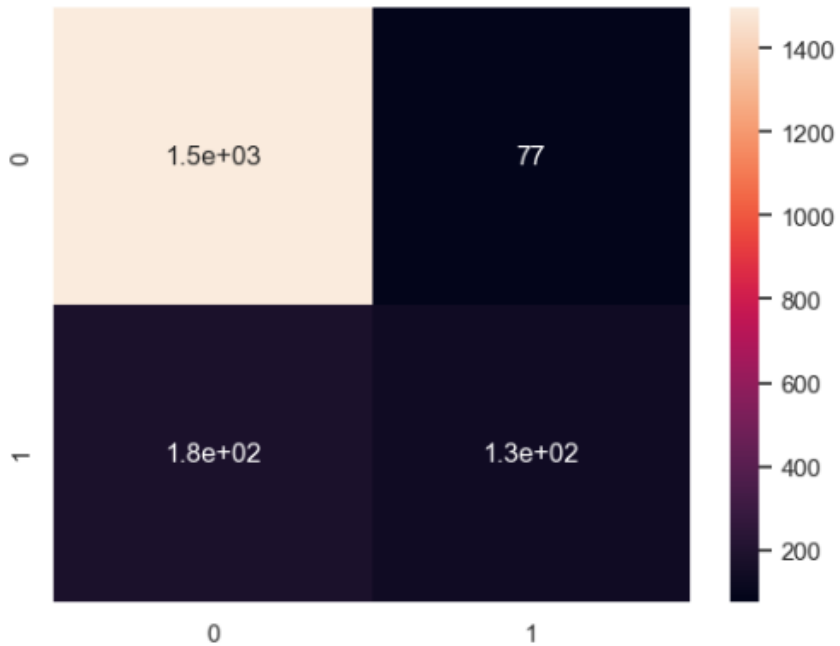


BIA BOSTON INSTITUTE OF ANALYTICS ®

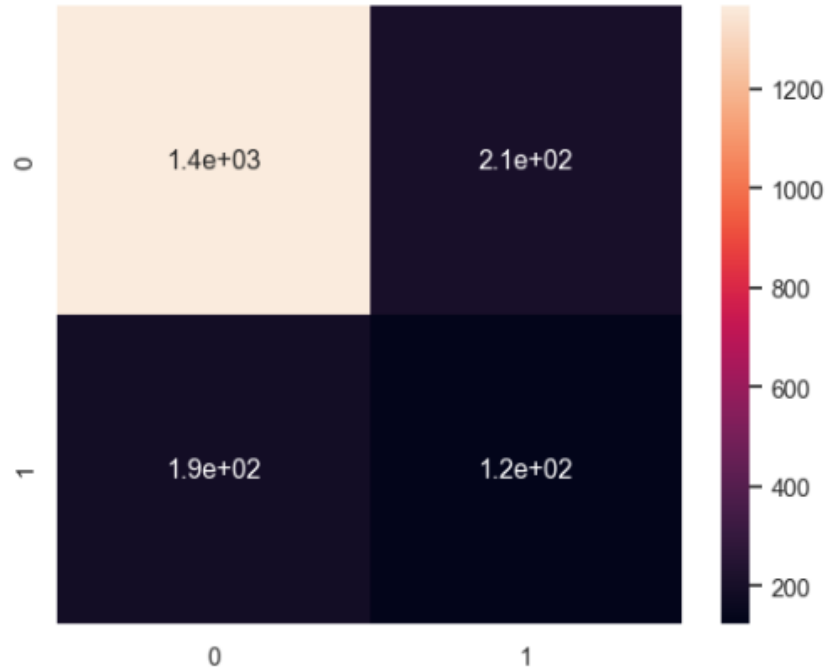
**********Classification Report**********

	precision	recall	f1-score	support
0.0	0.90	0.95	0.92	1573
1.0	0.64	0.46	0.54	311
accuracy			0.87	1884
macro avg	0.77	0.70	0.73	1884
weighted avg	0.86	0.87	0.86	1884

********** RandomForestClassifier **********
**********Training Accuracy**********
1.0
**********Testing Accuracy**********
0.8646496815286624
**********Confusion Matrix**********



********** DecisionTreeClassifier **********
**********Training Accuracy**********
1.0
**********Testing Accuracy**********
0.7919320594479831
**********Confusion Matrix**********



**********Classification Report**********

	precision	recall	f1-score	support
0.0	0.88	0.87	0.87	1573
1.0	0.38	0.40	0.39	311
accuracy			0.79	1884
macro avg	0.63	0.64	0.63	1884
weighted avg	0.80	0.79	0.79	1884

BIA BOSTON INSTITUTE OF ANALYTICS ®

Conclusion from the Logistic Regression Model Results:

✅ **Performance Metrics:**

- **Training Accuracy:** ~85.5%
- **Testing Accuracy:** ~86.8%

The model generalizes well — training and testing accuracy are close, suggesting **no overfitting**.

Confusion Matrix Insights:

	Predicted No Change (0)	Predicted Will Change (1)
Actual 0	**1500**	**80**
Actual 1	**170**	**140**

•**True Positives (140)**: Correctly predicted job changers
•**True Negatives (1500)**: Correctly predicted non-changers
•**False Positives (80)**: Predicted change, but no change
•**False Negatives (170)**: Missed actual job changers
Model is good at predicting who will not change jobs, but **misses a notable portion of actual changers (FN = 170)**.

BIA BOSTON INSTITUTE OF ANALYTICS ®

Prediction on Test Data

[462]:
```
train_data=pd.read_csv('aug_train.csv')
train_data.head()
```

[462]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	NaN	
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15	50-99	P
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5	NaN	
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	<1	NaN	P
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	>20	50-99	Funded St

[439]:
```
test_data=pd.read_csv('aug_test.csv')
test_data.head()
```

[439]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company
0	32403	city_41	0.827	Male	Has relevent experience	Full time course	Graduate	STEM	9	<10	
1	9858	city_103	0.920	Female	Has relevent experience	no_enrollment	Graduate	STEM	5	NaN	P
2	31806	city_21	0.624	Male	No relevent experience	no_enrollment	High School	NaN	<1	NaN	P
3	27385	city_13	0.827	Male	Has relevent experience	no_enrollment	Masters	STEM	11	10/49	P
4	27724	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	10000+	P

BIA BOSTON INSTITUTE OF ANALYTICS ®

```
[0]: # store the lengths
train_data_len = len(train_data)
test_data_len = len(test_data)
```

```
[1]: #concatenate both dataframes
new_data = pd.concat([train_data,test_data], axis=0)
new_data.reset_index(drop=True, inplace=True)
new_data.head()
```

[1]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	NaN	
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15	50-99	P
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5	NaN	
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	<1	NaN	P
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	>20	50-99	Funded St

```
[2]: new_data.tail()
```

[2]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	com
21282	1289	city_103	0.920	Male	No relevent experience	no_enrollment	Graduate	Humanities	16	NaN	Pu
21283	195	city_136	0.897	Male	Has relevent experience	no_enrollment	Masters	STEM	18	NaN	
21284	31762	city_100	0.887	Male	No relevent experience	no_enrollment	Primary School	NaN	3	NaN	
21285	7873	city_102	0.804	Male	Has relevent experience	Full time course	High School	NaN	7	100-500	Pu
21286	12215	city_102	0.804	Male	Has relevent experience	no_enrollment	Masters	STEM	15	10000+	

Concatenate Both Data frames

Concatenating Data Frames is used to **combine data** either by rows (axis=0) or columns (axis=1).
We use ignore_index=True to **reset the index** and avoid duplication or misalignment.
It helps in **merging datasets** like training + testing data or features + labels for easier processing.

BIA BOSTON INSTITUTE OF ANALYTICS ®

Test Data Sets

48]:
```
# One-hot encode test dataset
test_data_encoded = pd.get_dummies(test_data, drop_first=True)
```

49]:
```
# Align test dataset with training features
# Fill missing columns with 0 to match training set
X_test_final = X.columns  # From your training set

# Align columns
_, test_data_aligned = X.align(test_data_encoded, join='left', axis=1, fill_value=0)
```

[ ]:
```
# Predict job change (0 = No, 1 = Yes)
test_predictions = model.predict(test_data_scaled)

# If you want probabilities:
test_probabilities = model.predict_proba(test_data_scaled)[:, 1]
```

[ ]:
```
# Add predictions to the original test dataframe
test_data['predicted_target'] = test_predictions
test_data['change_probability'] = test_probabilities
```

60]:
```
test_data.head()
```

60]:

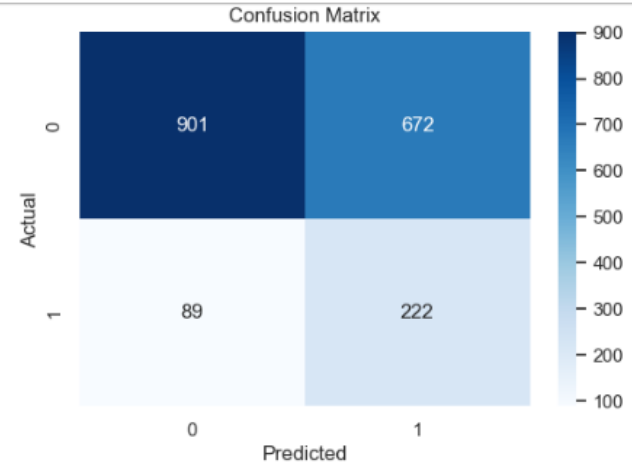
	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company
0	32403	city_41	0.827	Male	Has relevent experience	Full time course	Graduate	STEM	9	<10	
1	9858	city_103	0.920	Female	Has relevent experience	no_enrollment	Graduate	STEM	5	NaN	P
2	31806	city_21	0.624	Male	No relevent experience	no_enrollment	High School	NaN	<1	NaN	P
3	27385	city_13	0.827	Male	Has relevent experience	no_enrollment	Masters	STEM	11	10/49	P
4	27724	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	10000+	P

BIA BOSTON INSTITUTE OF ANALYTICS ®

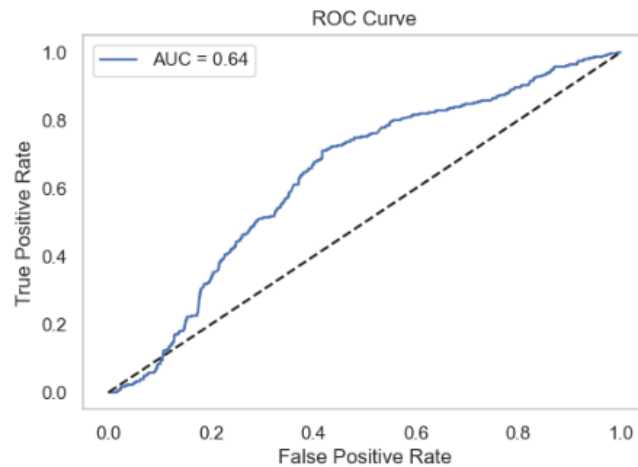
[461]:

```
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    classification_report,
    roc_auc_score,
    roc_curve
)
import matplotlib.pyplot as plt
import seaborn as sns

# • Predict on the test set
y_pred = model.predict(X_test_scaled)

# • Accuracy Score
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy: {acc:.4f}")

# • Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# • Classification Report (Precision, Recall, F1)
print("Classification Report:")
print(classification_report(y_test, y_pred))

# • ROC-AUC Score (only if binary classification)
if len(set(y_test)) == 2:
    y_proba = model.predict_proba(X_test_scaled)[:, 1]  # probability for class 1
    roc_auc = roc_auc_score(y_test, y_proba)
    print(f"ROC-AUC Score: {roc_auc:.4f}")

    # ROC Curve
    fpr, tpr, thresholds = roc_curve(y_test, y_proba)
    plt.figure(figsize=(6, 4))
    plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.2f}")
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("ROC Curve")
    plt.legend()
    plt.grid()
    plt.show()
```

```
Accuracy: 0.5961
```




```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.91      0.57      0.70      1573
         1.0       0.25      0.71      0.37       311

    accuracy                           0.60      1884
   macro avg       0.58      0.64      0.54      1884
weighted avg       0.80      0.60      0.65      1884

ROC-AUC Score: 0.6381
```



BIA BOSTON INSTITUTE OF ANALYTICS ®

Why This Is Important:

classification model using **multiple evaluation metrics—confusion matrix**, **classification report**, and **ROC curve**. These help in understanding not just overall accuracy, but **how well the model performs across both classes**, especially in imbalanced datasets.

Insights Gathered:

Confusion Matrix:

- **901** true negatives (0 predicted correctly)
- **222** true positives (1 predicted correctly)
- **672** false positives (predicted 1 but was 0)
- **89** false negatives (predicted 0 but was 1)

- **Accuracy**: 60%
- **ROC-AUC Score**: 0.638 (moderate performance)

- 📈 **ROC Curve:**
- AUC = 0.64 ⟶ Model has **some ability** to distinguish between classes, but not strong.

Classification Report:

- **Class 0 (no job change)**:
  - Precision: 0.91
  - Recall: 0.57
  - F1-score: 0.70

Class 1 (job change):

- Precision: 0.25
- Recall: 0.71
- F1-score: 0.37

📌 **Conclusion:**

•The model is **good at identifying people who won't change jobs** (class 0) but **struggles with predicting those who will** (class 1).

•**ROC-AUC and F1 scores show room for improvement**, suggesting the model needs **tuning** or **feature enhancement**.

•Important to track these metrics to avoid biased models and improve decision-making.

BIA BOSTON INSTITUTE OF ANALYTICS ®

Train data and Test data Split

```
]: 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)
```

```
]: data_encoded = pd.get_dummies(data, drop_first=True)
```

```
]: X = data_encoded.drop('target', axis=1)
y = data_encoded['target']
```

```
]: 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, stratify=y
)
```

```
]: from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
]: from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
]: print(data['target'].value_counts())
```

```
target
0.0    7864
1.0    1553
Name: count, dtype: int64
```

🔍 **Short Summary:**

- ✅ **One-hot encoding**: Converts categories to numbers.
- 🎯 **Feature/target split**: Separates inputs (X) and output (y).
- 🔀 **Train-test split**: 80% training, 20% testing (stratified to preserve class ratio).
- ⚖️ **Scaling**: Standardizes data for better model performance.
- 📊 **Class balance check**: Reveals data is imbalanced (more 0s than 1s).

👉 These steps are **essential for clean, fair, and effective machine learning modeling**.

BIA BOSTON INSTITUTE OF ANALYTICS ®


Thank You!