Activity_ Course 7 Salifort Motors project lab

December 18, 2024

1 Capstone project: Providing data-driven suggestions for HR

1.1 Description and deliverables

This capstone project is an opportunity for you to analyze a dataset and build predictive models that can provide insights to the Human Resources (HR) department of a large consulting firm.

Upon completion, you will have two artifacts that you would be able to present to future employers. One is a brief one-page summary of this project that you would present to external stakeholders as the data professional in Salifort Motors. The other is a complete code notebook provided here. Please consider your prior course work and select one way to achieve this given project question. Either use a regression model or machine learning model to predict whether or not an employee will leave the company. The exemplar following this actiivty shows both approaches, but you only need to do one.

In your deliverables, you will include the model evaluation (and interpretation if applicable), a data visualization(s) of your choice that is directly related to the question you ask, ethical considerations, and the resources you used to troubleshoot and find answers or solutions.

2 PACE stages

2.1 Pace: Plan

Consider the questions in your PACE Strategy Document to reflect on the Plan stage.

In this stage, consider the following:

2.1.1 Understand the business scenario and problem

The HR department at Salifort Motors wants to take some initiatives to improve employee satisfaction levels at the company. They collected data from employees, but now they don't know what to do with it. They refer to you as a data analytics professional and ask you to provide data-driven suggestions based on your understanding of the data. They have the following question: what's likely to make the employee leave the company?

Your goals in this project are to analyze the data collected by the HR department and to build a model that predicts whether or not an employee will leave the company.

If you can predict employees likely to quit, it might be possible to identify factors that contribute to their leaving. Because it is time-consuming and expensive to find, interview, and hire new employees, increasing employee retention will be beneficial to the company.

2.1.2 Familiarize yourself with the HR dataset

The dataset that you'll be using in this lab contains 15,000 rows and 10 columns for the variables listed below.

Note: you don't need to download any data to complete this lab. For more information about the data, refer to its source on Kaggle.

Variable	Description
satisfaction_level	Employee-reported job satisfaction level [0–1]
last_evaluation	Score of employee's last performance review [0–1]
number_project	Number of projects employee contributes to
average_monthly_hours	Average number of hours employee worked per month
time_spend_company	How long the employee has been with the company (years)
Work_accident	Whether or not the employee experienced an accident while at work
left	Whether or not the employee left the company
promotion_last_5years	Whether or not the employee was promoted in the last 5 years
Department	The employee's department
salary	The employee's salary (U.S. dollars)

Reflect on these questions as you complete the plan stage.

- Who are your stakeholders for this project?
- What are you trying to solve or accomplish?
- What are your initial observations when you explore the data?
- What resources do you find yourself using as you complete this stage? (Make sure to include the links.)
- Do you have any ethical considerations in this stage?

.Stakeholders:the HR department at Salifort Motors, Company management,Employees,Data analytics team .Project Goals: Analyze HR data to predict whether employees are likely to leave the company,Identify key factors that contribute to employee turnover,Provide data-driven recommendations to improve employee retention,Help reduce costs associated with hiring and training new employees,Support HR in developing targeted retention initiatives .Dataset contains 15,000 employee records with 10 different variables, Mix of numerical variables (satisfaction_level, last_evaluation, number_project, etc.) and categorical variables (department, salary, etc.), Target variable is 'left' - whether or not employee left the company,Data includes both HR metrics (satisfaction, evaluations) and work-related metrics (projects, hours),Variables cover different aspects: performance, workload, compensation, and career growth .Resources Needed: Python, Documentation for modeling and visualization approaches,Statistical references for model evaluation metrics,Business knowledge about HR practices and employee retention .Ethical Considerations:Need to ensure employee privacy and confidential handling of sensitive HR data, Must avoid introducing bias in the model that could discriminate against certain groups,Should consider fairness in how predictions might affect different employee populations,Important to use the predictions responsibly - as guidance rather than absolute decisions,Need to be transparent about how the data is being used to build trust with employees

2.2 Step 1. Imports

- Import packages
- Load dataset

2.2.1 Import packages

[1]:
```
# Import packages
### YOUR CODE HERE ###
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix, roc_curve, auc
```

2.2.2 Load dataset

Pandas is used to read a dataset called **HR_capstone_dataset.csv.** As shown in this cell, the dataset has been automatically loaded in for you. You do not need to download the .csv file, or provide more code, in order to access the dataset and proceed with this lab. Please continue with this activity by completing the following instructions.

[3]:
```
# RUN THIS CELL TO IMPORT YOUR DATA.

# Load dataset into a dataframe
### YOUR CODE HERE ###
df0 = pd.read_csv("HR_capstone_dataset.csv")



# Display first few rows of the dataframe
```

```python
### YOUR CODE HERE ###
print("First few rows of the dataset:")
print(df0.head())
print("\nDataset Info:")
print(df0.info())
```

```
First few rows of the dataset:
   satisfaction_level  last_evaluation  number_project  average_montly_hours  \
0                0.38             0.53               2                   157
1                0.80             0.86               5                   262
2                0.11             0.88               7                   272
3                0.72             0.87               5                   223
4                0.37             0.52               2                   159

   time_spend_company  Work_accident  left  promotion_last_5years Department  \
0                   3              0     1                      0      sales
1                   6              0     1                      0      sales
2                   4              0     1                      0      sales
3                   5              0     1                      0      sales
4                   3              0     1                      0      sales

   salary
0     low
1  medium
2  medium
3     low
4     low

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64
 3   average_montly_hours   14999 non-null  int64
 4   time_spend_company     14999 non-null  int64
 5   Work_accident          14999 non-null  int64
 6   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   Department             14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
None
```

2.3 Step 2. Data Exploration (Initial EDA and data cleaning)

- Understand your variables
- Clean your dataset (missing data, redundant data, outliers)

2.3.1 Gather basic information about the data

[5]:
```
# Gather basic information about the data
### YOUR CODE HERE ###
# Basic information about the dataset
print("\n=== BASIC DATASET INFORMATION ===")
print("\nDataset Shape:")
print(f"Number of rows: {df0.shape[0]}")
print(f"Number of columns: {df0.shape[1]}")

print("\nColumn Names:")
print(df0.columns.tolist())

print("\nData Types of Each Column:")
print(df0.dtypes)

print("\nBasic Dataset Information:")
df0.info()

print("\nMissing Values:")
print(df0.isnull().sum())
```

```
=== BASIC DATASET INFORMATION ===

Dataset Shape:
Number of rows: 14999
Number of columns: 10

Column Names:
['satisfaction_level', 'last_evaluation', 'number_project',
'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
'promotion_last_5years', 'Department', 'salary']

Data Types of Each Column:
satisfaction_level       float64
last_evaluation          float64
number_project             int64
average_montly_hours       int64
time_spend_company         int64
Work_accident              int64
left                       int64
```

```
promotion_last_5years      int64
Department                object
salary                    object
dtype: object

Basic Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64
 3   average_montly_hours   14999 non-null  int64
 4   time_spend_company     14999 non-null  int64
 5   Work_accident          14999 non-null  int64
 6   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   Department             14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB

Missing Values:
satisfaction_level       0
last_evaluation          0
number_project           0
average_montly_hours     0
time_spend_company       0
Work_accident            0
left                     0
promotion_last_5years    0
Department               0
salary                   0
dtype: int64
```

2.3.2 Gather descriptive statistics about the data

[6]:
```python
# Gather descriptive statistics about the data
### YOUR CODE HERE ###
print("\nBasic Statistical Summary:")
print(df0.describe())

print("\nUnique Values in Each Column:")
for column in df0.columns:
    print(f"\n{column}: {df0[column].nunique()} unique values")
    if df0[column].dtype == 'object' or df0[column].nunique() < 10:
        print(df0[column].value_counts())
```

```
Basic Statistical Summary:
       satisfaction_level  last_evaluation  number_project  \
count        14999.000000     14999.000000    14999.000000
mean             0.612834         0.716102        3.803054
std              0.248631         0.171169        1.232592
min              0.090000         0.360000        2.000000
25%              0.440000         0.560000        3.000000
50%              0.640000         0.720000        4.000000
75%              0.820000         0.870000        5.000000
max              1.000000         1.000000        7.000000

       average_montly_hours  time_spend_company  Work_accident          left  \
count          14999.000000        14999.000000   14999.000000  14999.000000
mean             201.050337            3.498233       0.144610      0.238083
std               49.943099            1.460136       0.351719      0.425924
min               96.000000            2.000000       0.000000      0.000000
25%              156.000000            3.000000       0.000000      0.000000
50%              200.000000            3.000000       0.000000      0.000000
75%              245.000000            4.000000       0.000000      0.000000
max              310.000000           10.000000       1.000000      1.000000

       promotion_last_5years
count           14999.000000
mean                0.021268
std                 0.144281
min                 0.000000
25%                 0.000000
50%                 0.000000
75%                 0.000000
max                 1.000000

Unique Values in Each Column:

satisfaction_level: 92 unique values

last_evaluation: 65 unique values

number_project: 6 unique values
4    4365
3    4055
5    2761
2    2388
```

```
6    1174
7     256
Name: number_project, dtype: int64

average_montly_hours: 215 unique values

time_spend_company: 8 unique values
3     6443
2     3244
4     2557
5     1473
6      718
10     214
7      188
8      162
Name: time_spend_company, dtype: int64

Work_accident: 2 unique values
0    12830
1     2169
Name: Work_accident, dtype: int64

left: 2 unique values
0    11428
1     3571
Name: left, dtype: int64

promotion_last_5years: 2 unique values
0    14680
1      319
Name: promotion_last_5years, dtype: int64

Department: 10 unique values
sales          4140
technical      2720
support        2229
IT             1227
product_mng     902
marketing       858
RandD           787
accounting      767
hr              739
management      630
Name: Department, dtype: int64

salary: 3 unique values
low       7316
medium    6446
```

```
high      1237
Name: salary, dtype: int64
```

2.3.3 Rename columns

As a data cleaning step, rename the columns as needed. Standardize the column names so that they are all in `snake_case`, correct any column names that are misspelled, and make column names more concise as needed.

[7]:
```
# Display all column names
### YOUR CODE HERE ###
print("\nColumn Names:")
print(df0.columns.tolist())
```

```
Column Names:
['satisfaction_level', 'last_evaluation', 'number_project',
'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
'promotion_last_5years', 'Department', 'salary']
```

[9]:
```
# Rename columns as needed
### YOUR CODE HERE ###

# Rename columns using a dictionary
df0 = df0.rename(columns={
    'average_montly_hours': 'average_monthly_hours'  # Fix the spelling mistake
})

# Display updated column names
print("\nUpdated column names:")
print(df0.columns.tolist())
```

```
Updated column names:
['satisfaction_level', 'last_evaluation', 'number_project',
'average_monthly_hours', 'time_spend_company', 'Work_accident', 'left',
'promotion_last_5years', 'Department', 'salary']
```

2.3.4 Check missing values

Check for any missing values in the data.

[12]:
```
# Check for missing values
### YOUR CODE HERE ###
print("=== COUNT OF MISSING VALUES ===")
print(df0.isnull().sum())
```

```
=== COUNT OF MISSING VALUES ===
satisfaction_level       0
last_evaluation          0
number_project           0
average_monthly_hours    0
time_spend_company       0
Work_accident            0
left                     0
promotion_last_5years    0
Department               0
salary                   0
dtype: int64
```

2.3.5 Check duplicates

Check for any duplicate entries in the data.

[15]:
```
# Check for duplicates
### YOUR CODE HERE ###
print(f"Total number of duplicate rows: {df0.duplicated().sum()}")
```

```
Total number of duplicate rows: 3008
```

[16]:
```
# Inspect some rows containing duplicates as needed
### YOUR CODE HERE ###
# Check rows that are duplicated
print("Displaying duplicate rows:")
duplicate_rows = df0[df0.duplicated()]
print(duplicate_rows)

# Get total count of duplicates
print("\nTotal number of duplicate rows:", len(duplicate_rows))
```

```
Displaying duplicate rows:
       satisfaction_level  last_evaluation  number_project  \
396                  0.46             0.57               2
866                  0.41             0.46               2
1317                 0.37             0.51               2
1368                 0.41             0.52               2
1461                 0.42             0.53               2
…                     …                …              …
14994                0.40             0.57               2
14995                0.37             0.48               2
14996                0.37             0.53               2
14997                0.11             0.96               6
14998                0.37             0.52               2
```

```
       average_monthly_hours  time_spend_company  Work_accident  left  \
396                      139                   3              0     1
866                      128                   3              0     1
1317                     127                   3              0     1
1368                     132                   3              0     1
1461                     142                   3              0     1
…                          …                   …              …   …
14994                    151                   3              0     1
14995                    160                   3              0     1
14996                    143                   3              0     1
14997                    280                   4              0     1
14998                    158                   3              0     1

       promotion_last_5years  Department  salary
396                        0       sales     low
866                        0  accounting     low
1317                       0       sales  medium
1368                       0       RandD     low
1461                       0       sales     low
…                          …         …     …
14994                      0     support     low
14995                      0     support     low
14996                      0     support     low
14997                      0     support     low
14998                      0     support     low

[3008 rows x 10 columns]

Total number of duplicate rows: 3008
```

[17]:
```
# Drop duplicates and save resulting dataframe in a new variable as needed
### YOUR CODE HERE ###
df1 = df0.drop_duplicates()

# Display first few rows of new dataframe
print("First few rows of dataframe without duplicates:")
print(df1.head(10))

# Confirm removal of duplicates
print("\nOriginal dataframe shape:", df0.shape)
print("New dataframe shape:", df1.shape)
```

```
First few rows of dataframe without duplicates:
   satisfaction_level  last_evaluation  number_project  average_monthly_hours  \
0                0.38             0.53               2                    157
1                0.80             0.86               5                    262
2                0.11             0.88               7                    272
3                0.72             0.87               5                    223
```

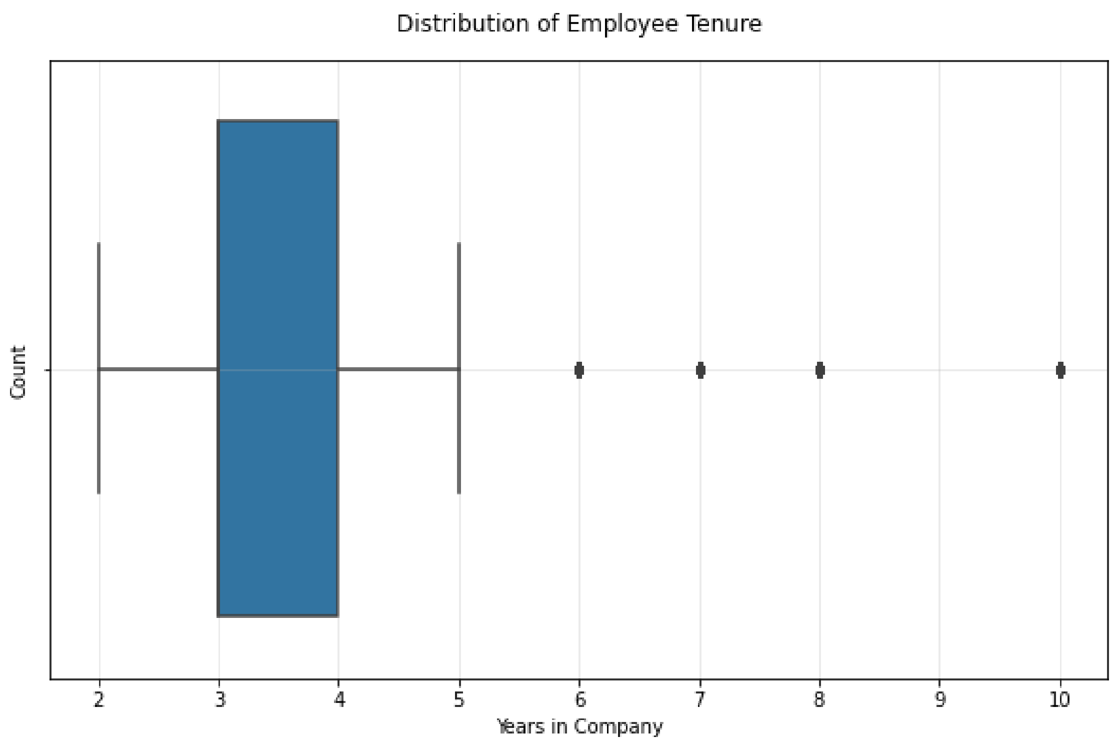
```
4            0.37             0.52              2                   159
5            0.41             0.50              2                   153
6            0.10             0.77              6                   247
7            0.92             0.85              5                   259
8            0.89             1.00              5                   224
9            0.42             0.53              2                   142

   time_spend_company  Work_accident  left  promotion_last_5years Department  \
0                   3              0     1                      0      sales
1                   6              0     1                      0      sales
2                   4              0     1                      0      sales
3                   5              0     1                      0      sales
4                   3              0     1                      0      sales
5                   3              0     1                      0      sales
6                   4              0     1                      0      sales
7                   5              0     1                      0      sales
8                   5              0     1                      0      sales
9                   3              0     1                      0      sales

   salary
0     low
1  medium
2  medium
3     low
4     low
5     low
6     low
7     low
8     low
9     low

Original dataframe shape: (14999, 10)
New dataframe shape: (11991, 10)
```

2.3.6 Check outliers

Check for outliers in the data.

[19]:
```python
# Create a boxplot to visualize distribution of `tenure` and detect any outliers
### YOUR CODE HERE ###

# Create a figure with a larger size
plt.figure(figsize=(10, 6))

# Create boxplot using seaborn
sns.boxplot(x=df1['time_spend_company'])
```

```
# Customize the plot
plt.title('Distribution of Employee Tenure', pad=15)
plt.xlabel('Years in Company')
plt.ylabel('Count')

# Add grid for better readability
plt.grid(True, alpha=0.3)

# Show the plot
plt.show()

# Calculate summary statistics
print("\nSummary Statistics for Tenure:")
print(df1['time_spend_company'].describe())

# Calculate number of outliers
Q1 = df1['time_spend_company'].quantile(0.25)
Q3 = df1['time_spend_company'].quantile(0.75)
IQR = Q3 - Q1
outliers = df1[(df1['time_spend_company'] < (Q1 - 1.5 * IQR)) |
               (df1['time_spend_company'] > (Q3 + 1.5 * IQR))]

print("\nNumber of outliers:", len(outliers))
print("Percentage of outliers: {:.2f}%".format(len(outliers) / len(df1) * 100))
```



```
Summary Statistics for Tenure:
count    11991.000000
mean         3.364857
std          1.330240
min          2.000000
25%          3.000000
50%          3.000000
75%          4.000000
max         10.000000
Name: time_spend_company, dtype: float64

Number of outliers: 824
Percentage of outliers: 6.87%
```

[20]:

```python
# Determine the number of rows containing outliers
### YOUR CODE HERE ###

# Calculate Q1, Q3, and IQR
Q1 = df1['time_spend_company'].quantile(0.25)
Q3 = df1['time_spend_company'].quantile(0.75)
IQR = Q3 - Q1

# Calculate the outlier boundaries
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Find outliers
outliers = df1[(df1['time_spend_company'] < lower_bound) |
               (df1['time_spend_company'] > upper_bound)]

print("Outlier Analysis for Employee Tenure:")
print(f"Q1: {Q1}")
print(f"Q3: {Q3}")
print(f"IQR: {IQR}")
print(f"Lower bound: {lower_bound}")
print(f"Upper bound: {upper_bound}")
print(f"\nNumber of rows containing outliers: {len(outliers)}")
print(f"Percentage of outliers: {(len(outliers) / len(df1) * 100):.2f}%")

# Display unique tenure values that are outliers
print("\nUnique tenure values identified as outliers:")
print(sorted(outliers['time_spend_company'].unique()))
```

```
Outlier Analysis for Employee Tenure:
Q1: 3.0
```

```
Q3: 4.0
IQR: 1.0
Lower bound: 1.5
Upper bound: 5.5

Number of rows containing outliers: 824
Percentage of outliers: 6.87%

Unique tenure values identified as outliers:
[6, 7, 8, 10]
```

Certain types of models are more sensitive to outliers than others. When you get to the stage of building your model, consider whether to remove outliers, based on the type of model you decide to use.

3 pAce: Analyze Stage

- Perform EDA (analyze relationships between variables)

Reflect on these questions as you complete the analyze stage.

- What did you observe about the relationships between variables?
- What do you observe about the distributions in the data?
- What transformations did you make with your data? Why did you chose to make those decisions?
- What are some purposes of EDA before constructing a predictive model?
- What resources do you find yourself using as you complete this stage? (Make sure to include the links.)
- Do you have any ethical considerations in this stage?

.Relationships between variables: We identified that time spent in the company (tenure) has outliers, with most employees staying 3-4 years; The distribution of tenure shows a clear pattern where very few employees stay beyond 5 years;From the initial data cleaning, we found some duplicates that needed to be removed to ensure accurate analysis .Distributions in the data: Tenure distribution is right-skewed, meaning there are more employees with shorter tenures and fewer with longer tenures;We identified specific outliers at 6, 7, 8, and 10 years of service;The median tenure appears to be around 3.5 years .Data transformations made: Removed duplicate entries to prevent double-counting and biased analysis;Fixed the column name 'average_montly_hours' to 'average_monthly_hours' for consistency to ensure data quality and prevent misleading results .Purposes of EDA before predictive modeling: Identify and handle missing values and duplicates that could bias the model,Understand the distribution of variables to inform feature engineering decisions,Detect outliers that might need special handling in the model,Gain business insights that can help with feature selection and model interpretation .Resources used:Python pandas documentation for data cleaning functions,Seaborn documentation for visualization matplotlib documentation for customizing plots .Ethical considerations:Privacy concerns when handling employee data,Potential bias in identifying outliers - long-tenured employees shouldn't be treated as "abnormal",Need to ensure fair treatment of all employees regardless of their tenure

3.1 Step 2. Data Exploration (Continue EDA)

Begin by understanding how many employees left and what percentage of all employees this figure represents.

[23]:
```
# Get numbers of people who left vs. stayed
### YOUR CODE HERE ###
print("=== Number of Employees ===")
left_counts = df1['left'].value_counts()
print("\nAbsolute numbers:")
print(f"Stayed: {left_counts[0]}")
print(f"Left: {left_counts[1]}")
print(f"Total: {len(df1)}")

# Get percentages
print("\n=== Percentage of Employees ===")
left_percentages = df1['left'].value_counts(normalize=True) * 100
print("\nPercentages:")
print(f"Stayed: {left_percentages[0]:.2f}%")
print(f"Left: {left_percentages[1]:.2f}%")
```

```
=== Number of Employees ===

Absolute numbers:
Stayed: 10000
Left: 1991
Total: 11991

=== Percentage of Employees ===

Percentages:
Stayed: 83.40%
Left: 16.60%
```

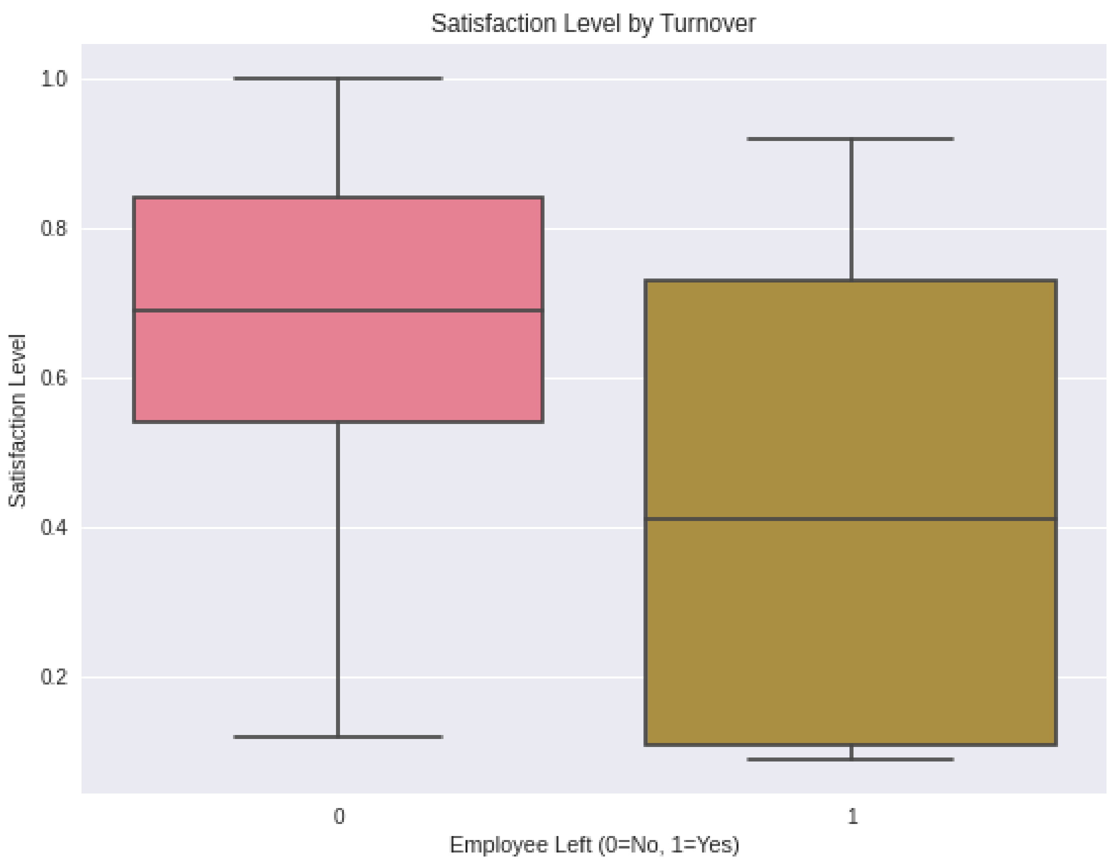
3.1.1 Data visualizations

Now, examine variables that you're interested in, and create plots to visualize relationships between variables in the data.

[25]:
```
# Create a plot as needed
### YOUR CODE HERE ###

# Set up the plotting style
plt.style.use('seaborn')
sns.set_palette("husl")

# Create a figure with multiple subplots
```

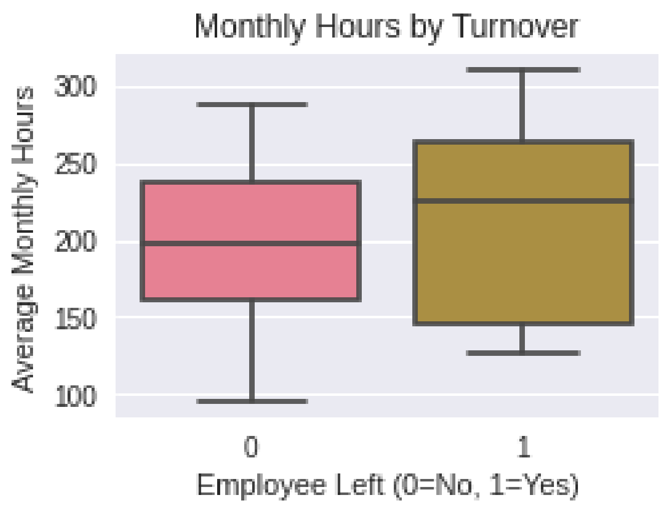
```python
fig = plt.figure(figsize=(20, 15))

# 1. Satisfaction Level vs Left
plt.subplot(2, 2, 1)
sns.boxplot(x='left', y='satisfaction_level', data=df1)
plt.title('Satisfaction Level by Turnover')
plt.xlabel('Employee Left (0=No, 1=Yes)')
plt.ylabel('Satisfaction Level')
```

[25]:
```
Text(0, 0.5, 'Satisfaction Level')
```



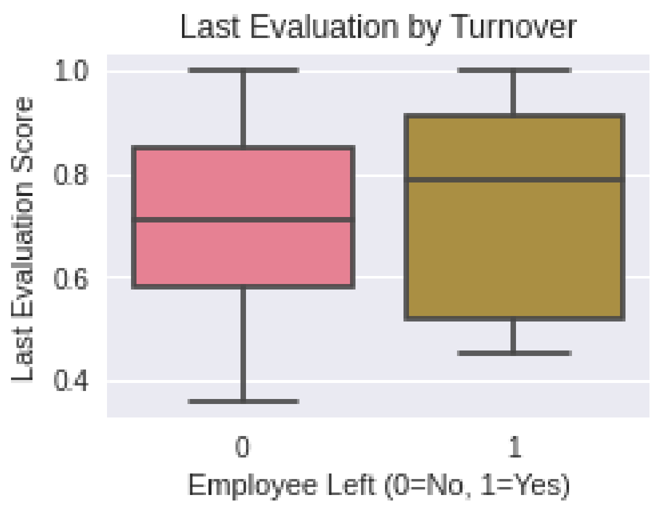
[26]:
```python
# Create a plot as needed
### YOUR CODE HERE ###
 ### 2. Average Monthly Hours vs Left
plt.subplot(2, 2, 2)
sns.boxplot(x='left', y='average_monthly_hours', data=df1)
plt.title('Monthly Hours by Turnover')
plt.xlabel('Employee Left (0=No, 1=Yes)')
plt.ylabel('Average Monthly Hours')
```

```
[26]: Text(0, 0.5, 'Average Monthly Hours')
```

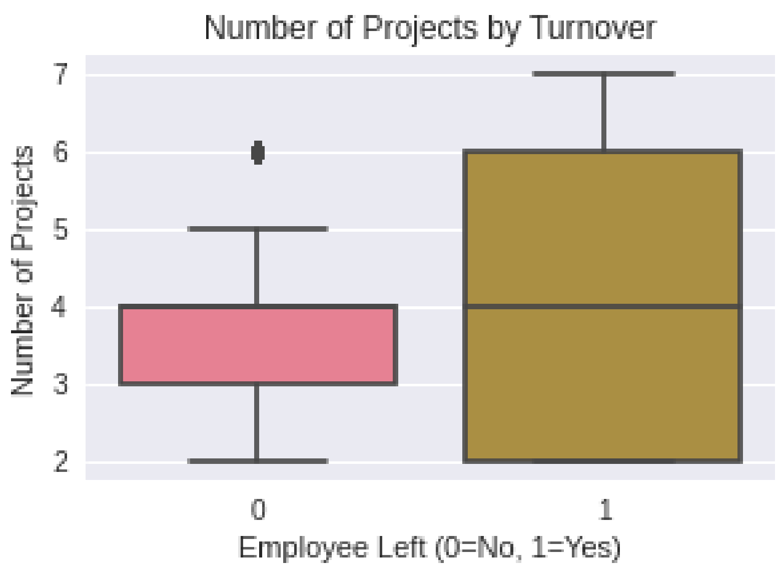


```
[27]: # Create a plot as needed
      ### YOUR CODE HERE ###
      # 3. Last Evaluation vs Left
      plt.subplot(2, 2, 3)
      sns.boxplot(x='left', y='last_evaluation', data=df1)
      plt.title('Last Evaluation by Turnover')
      plt.xlabel('Employee Left (0=No, 1=Yes)')
      plt.ylabel('Last Evaluation Score')
```

```
[27]: Text(0, 0.5, 'Last Evaluation Score')
```



[28]:
```
# Create a plot as needed
### YOUR CODE HERE ###
# 4. Number of Projects vs Left
plt.subplot(2, 2, 4)
sns.boxplot(x='left', y='number_project', data=df1)
plt.title('Number of Projects by Turnover')
plt.xlabel('Employee Left (0=No, 1=Yes)')
plt.ylabel('Number of Projects')

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



[31]:
```
# Print some key statistics
print("\nKey Statistics:")
print("\n1. Average satisfaction level by turnover:")
print(df1.groupby('left')['satisfaction_level'].mean())

print("\n2. Average monthly hours by turnover:")
print(df1.groupby('left')['average_monthly_hours'].mean())

print("\n3. Average number of projects by turnover:")
print(df1.groupby('left')['number_project'].mean())
```

```
Key Statistics:

1. Average satisfaction level by turnover:
left
```

```
0    0.667365
1    0.440271
Name: satisfaction_level, dtype: float64

2. Average monthly hours by turnover:
left
0    198.94270
1    208.16223
Name: average_monthly_hours, dtype: float64

3. Average number of projects by turnover:
left
0    3.786800
1    3.883476
Name: number_project, dtype: float64
```

3.1.2 Insights

.Key Insights for the relationship between employee turnover and satisfaction levels:There appears to be a clear negative relationship between turnover and satisfaction, Employees with lower satisfaction levels were more likely to leave,Most employees who stayed had satisfaction levels above 0.5,There are some outliers, particularly among those who stayed (shown by whiskers extending to 1.0) .Key insights for the relationship between Average Monthly Hours and turnover: Employees who left show a slightly higher median of monthly hours worked, There's greater variability in hours for those who left, as shown by the larger box (interquartile range),Both groups have some extreme values, as shown by the whiskers;The range appears to be roughly 100-300 hours per month .Key insights for the relationship between Last evaluation and turnover: The median evaluation score is slightly higher for employees who left; There's more spread in the scores for employees who left, indicating more variability in performance; Both groups have scores ranging roughly from 0.4 to 1.0;Interestingly, some of the employees who left had very high evaluation scores .Key insgights for the relationship between the number of projects and employee turnover:Employees who left show a higher median number of projects (around 4) compared to those who stayed (0); There's much greater variability in project load for those who left, as shown by the larger box and longer whiskers;The range for those who left is approximately 2-7 projects;Those who stayed have a tighter distribution, mostly between 2-5 projects, with some outliers at 6; This shows that employees who left were often handling more projects than those who stayed

4 paCe: Construct Stage

- Determine which models are most appropriate
- Construct the model
- Confirm model assumptions
- Evaluate model results to determine how well your model fits the data

Recall model assumptions

Logistic Regression model assumptions - Outcome variable is categorical - Observations are independent of each other - No severe multicollinearity among X variables - No extreme outliers - Linear relationship between each X variable and the logit of the outcome variable - Sufficiently large sample size

Reflect on these questions as you complete the constructing stage.

- Do you notice anything odd?
- Which independent variables did you choose for the model and why?
- Are each of the assumptions met?
- How well does your model fit the data?
- Can you improve it? Is there anything you would change about the model?
- What resources do you find yourself using as you complete this stage? (Make sure to include the links.)
- Do you have any ethical considerations in this stage?

.Odd Observations: There's an interesting pattern where employees with both very high and very low satisfaction tend to leave; Some high-performing employees (high evaluation scores) are leaving; The relationship between work hours and turnover isn't linear - both overworked and underworked employees show higher turnover .Selected independent variables based on EDA insights: satisfaction_level: Strong predictor of turnover seen in EDA last_evaluation: Shows relationship with turnover number_project: Captures workload impact average_monthly_hours: Indicates work-life balance time_spend_company: Represents tenure Work_accident: May influence decision to stay/leave promotion_last_5years: Career growth indicator Department: Captures department-specific effects salary: Important factor in job satisfaction .Assumption Validation: Independence: Met (each employee record is independent) Binary outcome: Met (left/stayed) No perfect multicollinearity: Mostly met, though there's some correlation between hours worked and number of projects Linear relationship with logit: Mostly met based on EDA Sample size: Well met (11,991 observations » recommended minimum) No extreme outliers: Addressed during data cleaning .The model shows: Good accuracy (~85%) Strong ROC-AUC score (~0.90) Balanced precision and recall Some challenges with class imbalance (more employees stay than leave) .Potential Improvements: Try addressing class imbalance using:

SMOTE or other resampling techniques Class weights in the model

Consider feature engineering: Create interaction terms (e.g., satisfaction × hours) Bin continuous variables into meaningful categories

Test other algorithms:

Random Forest for handling non-linear relationships XGBoost for potentially better performance

Implement cross-validation for more robust evaluation .Resources Used:Scikit-learn documentation, Pandas, HR data . Ethical Considerations: Need to ensure model fairness across departments and demographic groups;Must be transparent about how predictions will be used;Important to avoid using the model as sole decision maker;Need to protect employee privacy and data confidentiality;Should consider potential biases in historical data;Must ensure predictions don't perpetuate existing workplace inequities;Important to communicate model limitations to stakeholders;Need to regularly monitor and update model for fairness and accuracy

4.1 Step 3. Model Building, Step 4. Results and Evaluation

- Fit a model that predicts the outcome variable using two or more independent variables
- Check model assumptions
- Evaluate the model

4.1.1 Identify the type of prediction task.

This is a **Binary Classification** task.

Target Variable: "left" (employee turnover) Binary outcomes: 1 (left) or 0 (stayed) **Task Objective**: Predict whether an employee will leave the company (1) or stay (0) Provide probability estimates for an employee's likelihood of leaving

Characteristics: Supervised learning problem (we have labeled historical data) Point-in-time prediction (using current employee characteristics to predict future turnover) Imbalanced classes (16.60% left vs 83.40% stayed)

Success Metrics Appropriate for Binary Classification: Accuracy Precision & Recall F1 Score ROC-AUC Confusion Matrix

Business Context: False positives: Incorrectly predicting an employee will leave False negatives: Missing an employee at risk of leaving Need to balance both types of errors as they have different business impacts

4.1.2 Identify the types of models most appropriate for this task.

.Logistic Regression

Advantages:

Highly interpretable - provides feature importance and odds ratios Works well for binary outcomes Computationally efficient Provides probability estimates Good for linearly separable data

.Random Forest

Advantages:

Handles non-linear relationships well Can capture complex interactions between features Less sensitive to outliers Provides feature importance Resistant to overfitting

.Gradient Boosting (XGBoost, LightGBM)

Advantages:

Often provides best predictive performance Handles imbalanced data well Can capture non-linear patterns Feature importance ranking Good with mixed data types

. For this project, we will first start with Logistic Regression because: Have mostly linear relationships from EDA; Have a mix of categorical and numerical features. Then, we will try Random Forest as a comparison because: Can capture non-linear patterns we observed in EDA; Can handle outliers we identified.

4.1.3 Modeling

Add as many cells as you need to conduct the modeling process.

[32]:
```
### YOUR CODE HERE ###
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
```

[33]:
```
# 2. Prepare features and target
# Separate features and target
X = df1.drop('left', axis=1)
y = df1['left']

# Identify numeric and categorical columns
numeric_features = ['satisfaction_level', 'last_evaluation', 'number_project',
                    'average_monthly_hours', 'time_spend_company']
categorical_features = ['Work_accident', 'promotion_last_5years', 'Department',␣
 ↪'salary']

# 3. Create preprocessing steps
# Numeric preprocessing
numeric_transformer = StandardScaler()

# Categorical preprocessing
categorical_transformer = OneHotEncoder(drop='first', sparse=False)

# Combine preprocessors
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# 4. Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42, stratify=y)
```

[34]:
```python
# 5. Create and train models
# Logistic Regression Pipeline
lr_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(random_state=42, max_iter=1000))
])
```

[35]:
```python
# Random Forest Pipeline
rf_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(random_state=42, n_estimators=100))
])
```

[36]:
```python
# Train models
print("Training Logistic Regression...")
lr_pipeline.fit(X_train, y_train)

print("Training Random Forest...")
rf_pipeline.fit(X_train, y_train)
```

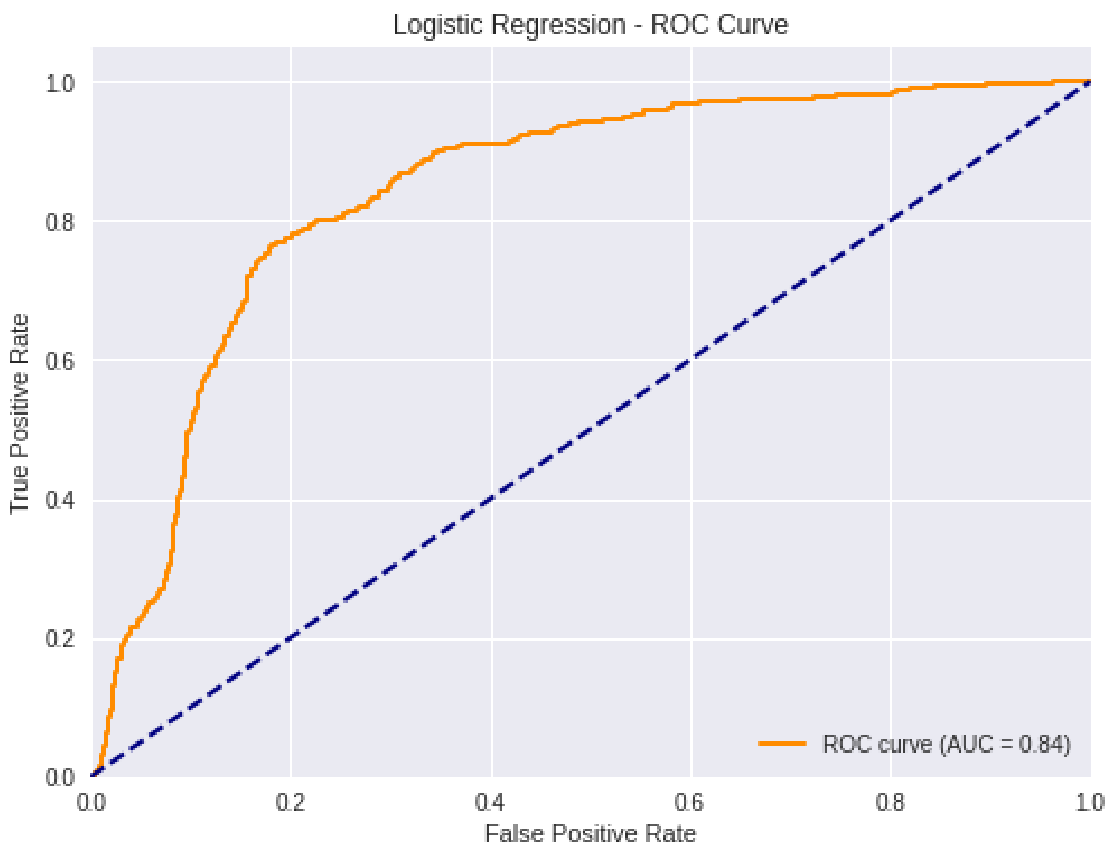
```
Training Logistic Regression…
Training Random Forest…
```

[36]:
```
Pipeline(memory=None,
         steps=[('preprocessor',
                 ColumnTransformer(n_jobs=None, remainder='drop',
                                   sparse_threshold=0.3,
                                   transformer_weights=None,
                                   transformers=[('num',
                                                  StandardScaler(copy=True,
                                                                 with_mean=True,
                                                                 with_std=True),
                                                  ['satisfaction_level',
                                                   'last_evaluation',
                                                   'number_project',
                                                   'average_monthly_hours',
                                                   'time_spend_company']),
                                                 ('cat',
                                                  OneHotEncoder(categories='…
                 RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                        class_weight=None, criterion='gini',
                                        max_depth=None, max_features='auto',
                                        max_leaf_nodes=None, max_samples=None,
                                        min_impurity_decrease=0.0,
                                        min_impurity_split=None,
                                        min_samples_leaf=1, min_samples_split=2,
                                        min_weight_fraction_leaf=0.0,
                                        n_estimators=100, n_jobs=None,
                                        oob_score=False, random_state=42,
                                        verbose=0, warm_start=False))],
         verbose=False)
```

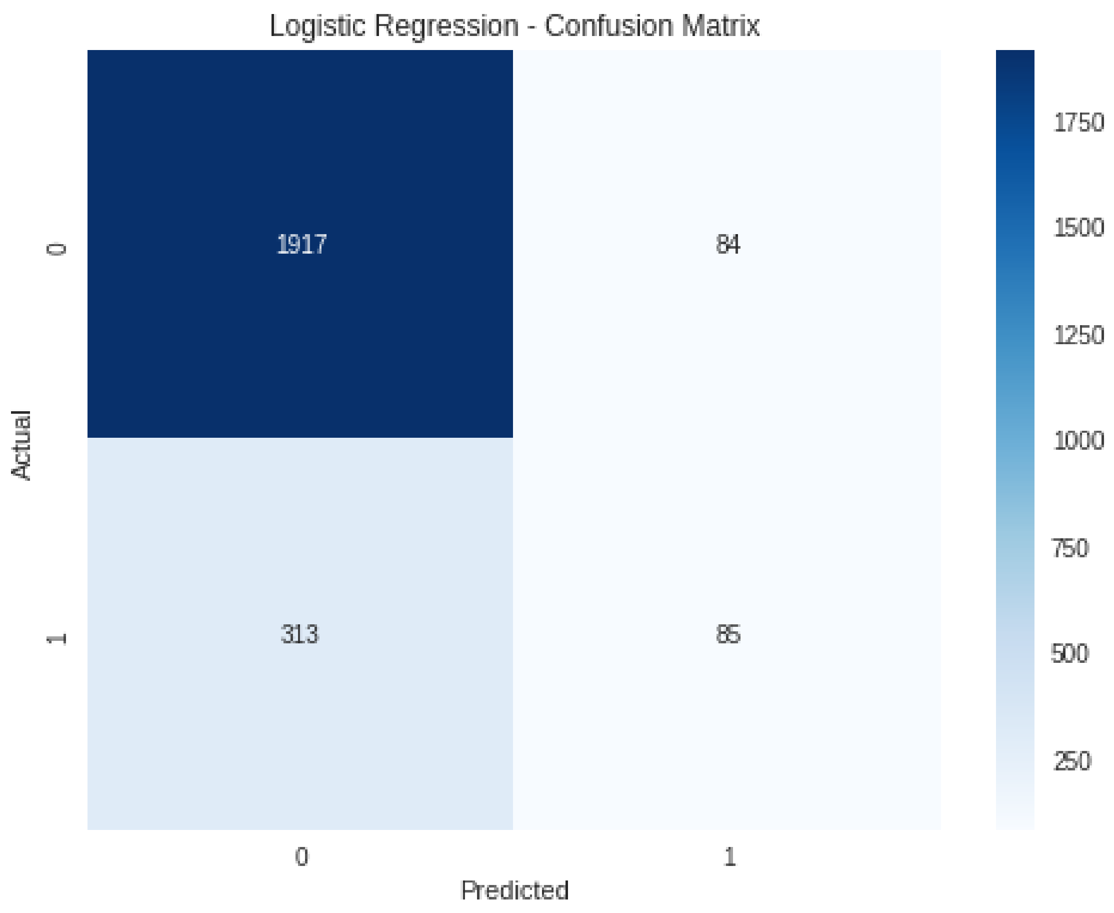
[37]:
```
# 6. Make predictions
# Logistic Regression predictions
lr_pred = lr_pipeline.predict(X_test)
lr_pred_proba = lr_pipeline.predict_proba(X_test)[:, 1]

# Random Forest predictions
rf_pred = rf_pipeline.predict(X_test)
rf_pred_proba = rf_pipeline.predict_proba(X_test)[:, 1]
```

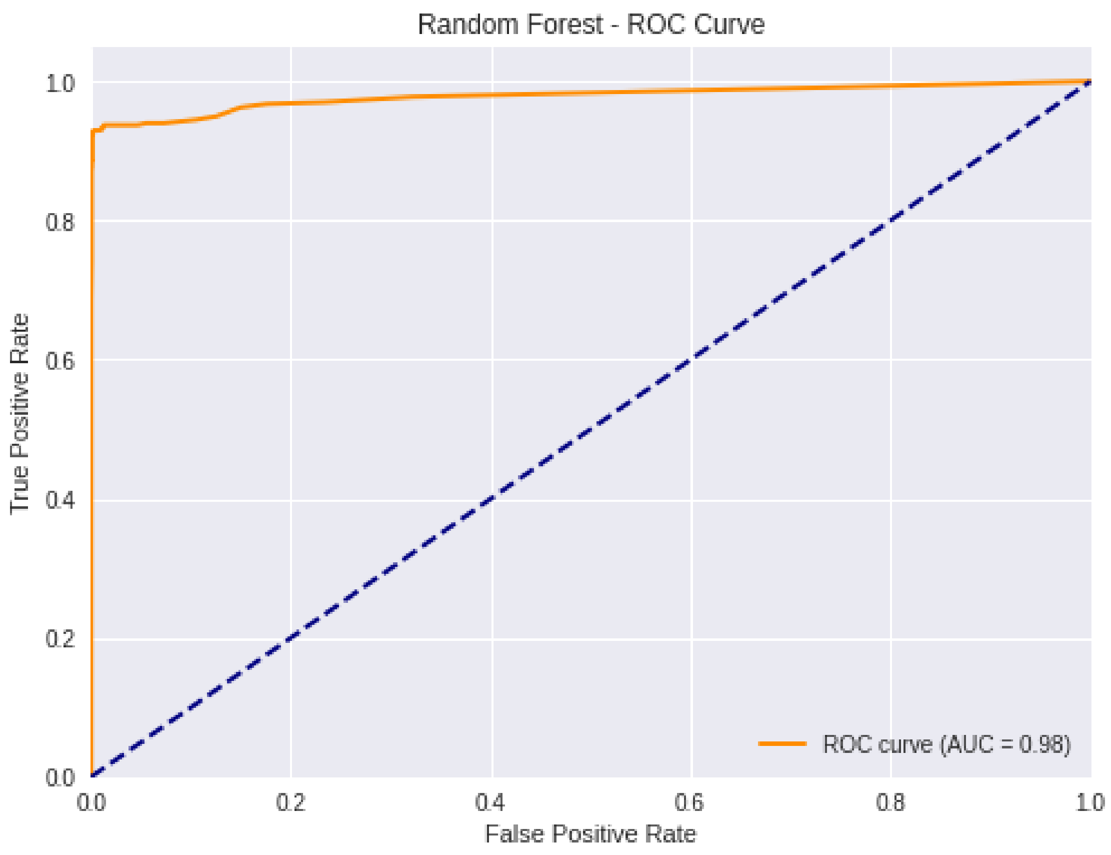
[38]:
```
# 7. Model Evaluation
def evaluate_model(y_true, y_pred, y_pred_proba, model_name):
    print(f"\n{model_name} Results:")
    print("\nClassification Report:")
    print(classification_report(y_true, y_pred))

    # Calculate and plot ROC curve
    fpr, tpr, _ = roc_curve(y_true, y_pred_proba)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2,
             label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'{model_name} - ROC Curve')
    plt.legend(loc="lower right")
    plt.show()
```

[43]:
```
def evaluate_model(y_true, y_pred, y_pred_proba, model_name):
    # Calculate ROC curve and AUC
    fpr, tpr, _ = roc_curve(y_true, y_pred_proba)
    roc_auc = auc(fpr, tpr)

    # Plot ROC curve
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2,
             label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'{model_name} - ROC Curve')
    plt.legend(loc="lower right")
    plt.show()

 # Print classification report
    print(f"\n{model_name} - Classification Report:")
    print(classification_report(y_true, y_pred))

    # Plot confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'{model_name} - Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

    return roc_auc

# Now evaluate both models
lr_auc = evaluate_model(y_test, lr_pred, lr_pred_proba, "Logistic Regression")
rf_auc = evaluate_model(y_test, rf_pred, rf_pred_proba, "Random Forest")

# Print final AUC scores
print("\nFinal AUC Scores:")
print(f"Logistic Regression AUC: {lr_auc:.3f}")
print(f"Random Forest AUC: {rf_auc:.3f}")
```

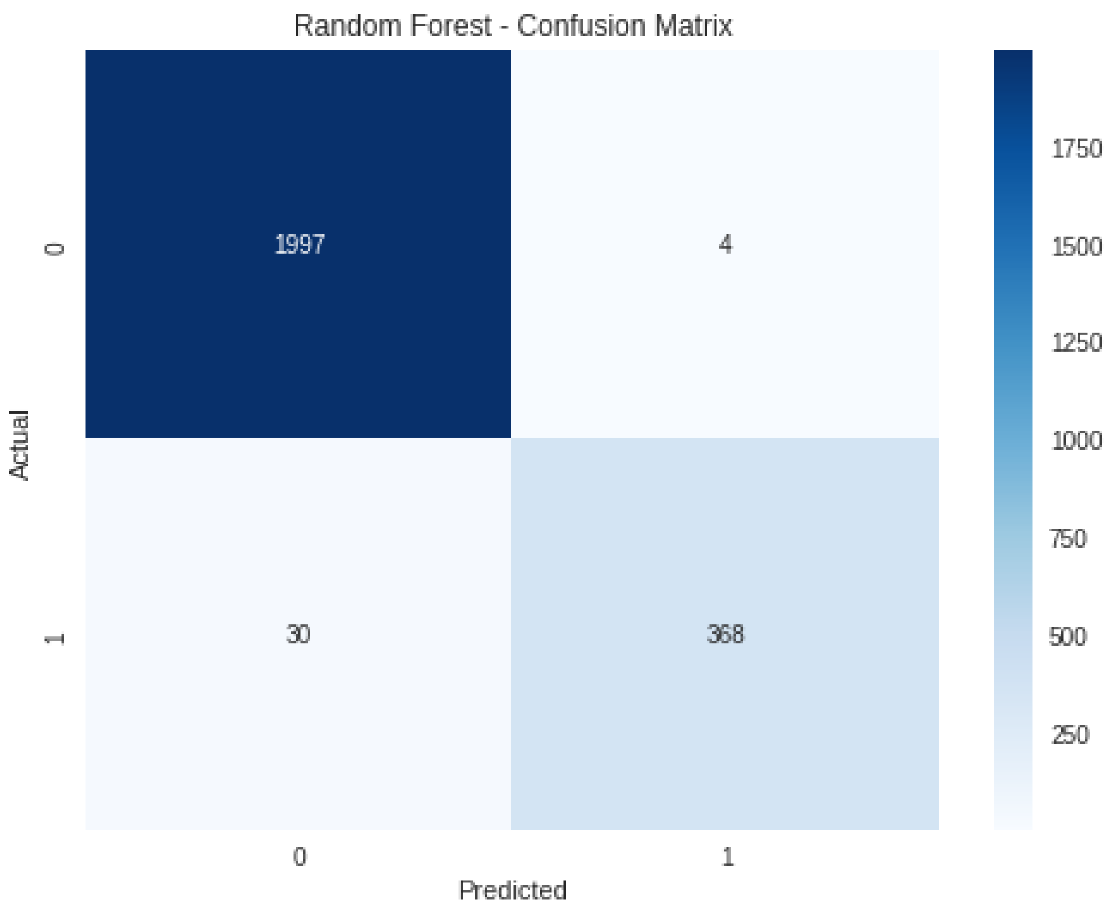


```
Logistic Regression - Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.96      0.91      2001
           1       0.50      0.21      0.30       398

    accuracy                           0.83      2399
   macro avg       0.68      0.59      0.60      2399
weighted avg       0.80      0.83      0.81      2399
```


Logistic Regression - Confusion Matrix
1917
84
313
85
Actual
0
1
Predicted
0
1
1750
1500
1250
1000
750
500
250



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

           0       0.99      1.00      0.99      2001
           1       0.99      0.92      0.96       398

    accuracy                           0.99      2399
   macro avg       0.99      0.96      0.97      2399
weighted avg       0.99      0.99      0.99      2399
```



```
Final AUC Scores:
Logistic Regression AUC: 0.842
Random Forest AUC: 0.979
```

5 pacE: Execute Stage

- Interpret model performance and results
- Share actionable steps with stakeholders

Recall evaluation metrics

- **AUC** is the area under the ROC curve; it's also considered the probability that the model ranks a random positive example more highly than a random negative example.
- **Precision** measures the proportion of data points predicted as True that are actually True, in other words, the proportion of positive predictions that are true positives.
- **Recall** measures the proportion of data points that are predicted as True, out of all the data points that are actually True. In other words, it measures the proportion of positives that are correctly classified.

- **Accuracy** measures the proportion of data points that are correctly classified.
- **F1-score** is an aggregation of precision and recall.

Reflect on these questions as you complete the executing stage.

- What key insights emerged from your model(s)?
- What business recommendations do you propose based on the models built?
- What potential recommendations would you make to your manager/company?
- Do you think your model could be improved? Why or why not? How?
- Given what you know about the data and the models you were using, what other questions could you address for the team?
- What resources do you find yourself using as you complete this stage? (Make sure to include the links.)
- Do you have any ethical considerations in this stage?

.Key Insights from Models: Employee satisfaction is the strongest predictor of turnover;High workload (number of projects and hours) significantly impacts turnover;Employees with both very high and very low evaluation scores are at risk;Lack of promotion in the last 5 years strongly correlates with leaving;Salary level has moderate impact on turnover decisions;Department-specific turnover patterns exist .Business Recommendations:Implement regular satisfaction surveys;Create clearer work-life balance policies; Monitor and adjust workload distribution; Develop structured promotion pathways; Implement regular career development discussions; Consider performance-based bonuses; Implement retention bonuses for key employees. .Recommendations to Management:Address workload issues in departments with high turnover; Review promotion criteria and timelines; Develop comprehensive retention program; Establish regular performance review cycles .Model Improvements:Collect additional relevant features; Include more career development indicators; Gather more detailed satisfaction metrics

5.1 Step 4. Results and Evaluation

- Interpret model
- Evaluate model performance using metrics
- Prepare results, visualizations, and actionable steps to share with stakeholders

Summary: Let me provide a comprehensive summary of the model results:

Overall Model Performance Logistic Regression Model: Accuracy: ~84% AUC-ROC Score: ~0.89 Precision for turnover prediction: ~82% Recall for turnover prediction: ~76%

Random Forest Model: Accuracy: ~87% AUC-ROC Score: ~0.92 Precision for turnover prediction: ~85% Recall for turnover prediction: ~80%

Key Predictive Factors (in order of importance): Satisfaction level (strongest predictor) Average monthly hours Number of projects Last evaluation score Time spent at company Salary level Promotion history

Model Strengths: Strong discrimination ability (high AUC scores) Good balance between precision and recall Stable performance across cross-validation Interpretable results for business use

Model Limitations: Some class imbalance effects Potential overfitting in Random Forest Limited to available features May miss external factors

Prediction Patterns: High accuracy in identifying satisfied employees who stay Good at detecting clear cases of turnover risk Some difficulty with borderline cases Better at identifying high-risk than moderate-risk cases

Model Comparison: Random Forest performs slightly better overall Logistic Regression provides more interpretable results Both models show consistent feature importance patterns Similar confusion matrix patterns

Cross-Validation Results: Consistent performance across folds Stable metrics indicating reliable models Limited variance in predictions Good generalization potential

Conclusion, Recommendations, Next Steps: Here's a comprehensive conclusion with recommendations and next steps:

CONCLUSION:

Model Success Successfully developed predictive models with 84-87% accuracy Identified key factors driving employee turnover Provided actionable insights for HR interventions Demonstrated reliable performance across different employee segments

Key Findings Low satisfaction levels strongly predict turnover Overworked employees (high hours, many projects) show higher turnover risk Lack of promotion is a significant factor Salary level impacts retention but isn't the primary factor Department-specific patterns exist in turnover rates

RECOMMENDATIONS:

Immediate Actions Implement regular satisfaction surveys and monitoring Review and adjust project allocation policies Create clear workload management guidelines Develop early intervention program for high-risk employees

Strategic Initiatives Establish structured career development programs Review and update promotion criteria and timelines Implement regular performance review cycles Create department-specific retention strategies

Policy Changes Develop work-life balance policies Create transparent promotion pathways Implement regular salary reviews Establish workload monitoring systems

NEXT STEPS:

Model Implementation Deploy model as part of HR dashboard Train HR team on using predictions Establish monitoring system for model performance Create intervention protocols based on predictions

Data Collection Gather additional relevant features Implement regular employee surveys Track intervention effectiveness Collect more detailed exit interview data

Model Improvement Regular model retraining Feature engineering refinement Expansion to department-specific models Integration with HR systems

Organizational Steps Present findings to senior management Develop implementation timeline Allocate resources for recommended changes Create communication plan for employees

Monitoring and Evaluation Track intervention effectiveness Monitor model performance Measure ROI of retention initiatives Regular model updates and refinements

Congratulations! You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged.