Activity_ Course 7 Salifort Motors project lab

July 18, 2025

1 Capstone project: Providing data-driven suggestions for HR

2 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.

3 PACE stages

3.1 Pace: Plan

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

In this stage, consider the following:

3.1.1 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.

3.1.2 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?

The primary stakeholders are the Human Resources department and senior leadership team at Salifort Motors. They are concerned about high employee turnover and are seeking data-driven insights and solutions to retain talent and improve organizational culture.

I am trying to accomplish: - Identify key factors contributing to employee turnover - Build a predictive model that estimates whether an employee is likely to leave - Generate insights and actionable recommendations to increase employee retention and reduce hiring/training costs

My initial exploration of the dataset shows:

- A total of 14,999 entries and 10 variables
- Variables include performance metrics, work conditions, satisfaction level, and promotions
- No immediate signs of missing values
- Some numeric columns like average_monthly_hours and time_spend_company may contain outliers
- Target variable (left) is binary, ideal for classification modeling
- Categorical columns like department and salary will need encoding

Resources that can be helpful for me to complete:

Pandas Documentation Matplotlib Documentation Seaborn Documentation Scikit-learn Documentation Coursera-provided course content and notebooks Kaggle dataset reference: https://www.kaggle.com/datasets/mfaisalqureshi/hr-analytics-and-job-prediction

My primary ethical considerations are: - Ensuring the model doesn't discriminate against employees based on sensitive attributes like department or salary level

- Protecting employee privacy and confidentiality
- Avoiding biased interpretations that could unfairly influence HR decisions
- Emphasizing transparency so that predictions are explainable and justifiable to both managers and employees

3.2 Step 1. Imports

- Import packages
- Load dataset

3.2.1 Import packages

[27]:
```python
# Import packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn as sklearn
```

3.2.2 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.

[28]:
```python
# 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
df0.head()
```

[28]:
```
   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
```

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

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

3.3.1 Gather basic information about the data

[29]:
```python
# Gather basic information about the data
# Basic structure
df0.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
```

3.3.2 Gather descriptive statistics about the data

[30]:
```python
# Gather descriptive statistics about the data
df0.describe()
```

[30]:
```
       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
```

Key takeaways: satisfaction_level: some employees are extremely dissatisfied, which might be related to attrition. last_evaluation: some employees have higher performance reviews. But need to check if they are leaving as well. number_project: range is 2-7 projects/ employee. Worth taking a look if overwork or boredom is the reason. average_monthly_hours: **has typo in column name.** employees working higher hours might need taking a look. time_spend_company: range is 2-10 years. might see an attrition spike after 3+ years. work_accident, promotion_last_5years: accidents are pretty low but can cause in attrition. Also promotion is significantly low which may cause dissatisfaction. left: about 23% employees left the organization. Which means almost 1 in 4 employees have left in the past.

3.3.3 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.

[31]:
```
# Display all column names
df0.columns
```

[31]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
       'promotion_last_5years', 'Department', 'salary'],
      dtype='object')
```

[32]:
```
# Rename columns as needed
df0.columns = [
    "satisfaction_level", "last_evaluation", "number_project",␣
 ↪"average_monthly_hours", "time_spend_company",
    "work_accident", "left", "promotion_last_5years", "department", "salary"

]



# Display all column names after the update
df0.columns
```

[32]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_monthly_hours', 'time_spend_company', 'work_accident', 'left',
       'promotion_last_5years', 'department', 'salary'],
      dtype='object')
```

3.3.4 Check missing values

Check for any missing values in the data.

[33]:
```
# Check for missing values
df0.isnull().sum()
```

[33]:
```
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
```

3.3.5 Check duplicates

Check for any duplicate entries in the data.

[34]:
```
# Check for duplicates
duplicates = df0[df0.duplicated()]
print(f"Duplicates found: {len(duplicates)}")
```

```
Duplicates found: 3008
```

[35]:
```
# Inspect some rows containing duplicates as needed
duplicates.head()
```

[35]:
```
      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

      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

      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
```

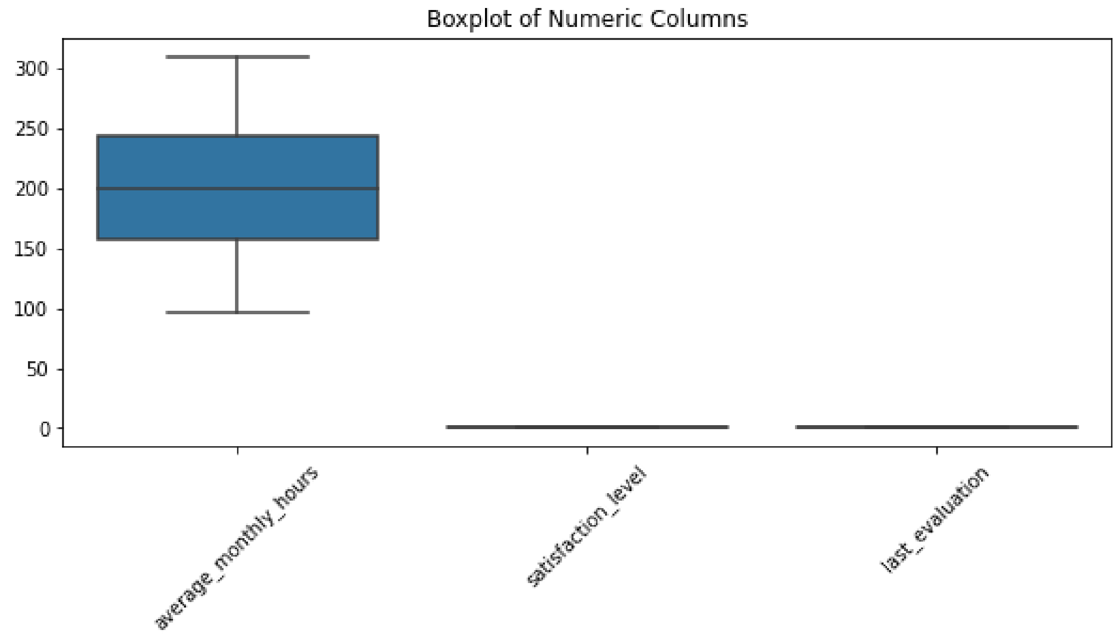
[36]:
```python
# Drop duplicates and save resulting dataframe in a new variable as needed
df_cleaned = df0.drop_duplicates()


# Display first few rows of new dataframe as needed
df_cleaned.head()
```

[36]:
```
   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

   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
```

3.3.6 Check outliers

Check for outliers in the data.

[37]:
```python
# Create a boxplot to visualize distribution of `tenure` and detect any outliers
plt.figure(figsize=(10, 4))
sns.boxplot(data=df_cleaned[['average_monthly_hours', 'satisfaction_level',␣
 ↪'last_evaluation']])
plt.title('Boxplot of Numeric Columns')
plt.xticks(rotation=45)
plt.show()
```



[38]:
```
# Determine the number of rows containing outliers
numeric_cols = ['satisfaction_level', 'last_evaluation', 'number_project',
                'average_monthly_hours', 'time_spend_company']

for col in numeric_cols:
    Q1 = df_cleaned[col].quantile(0.25)
    Q3 = df_cleaned[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df_cleaned[(df_cleaned[col] < lower_bound) | (df_cleaned[col] >␣
↪upper_bound)]
    print(f"{col}: {len(outliers)} outliers")
```

```
satisfaction_level: 0 outliers
last_evaluation: 0 outliers
number_project: 0 outliers
average_monthly_hours: 0 outliers
time_spend_company: 824 outliers
```

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.

4 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?

Initial exploration shows that satisfaction_level, number_project, and average_monthly_hours may have strong associations with employee attrition (left). For example, low satisfaction and very high average monthly hours could be linked to a higher likelihood of leaving.

Most numeric features such as satisfaction, evaluation score, and average hours follow reasonably normal distributions. time_spend_company is slightly skewed with many employees leaving around the 3–4 year mark. Promotions are very rare, and salaries are mostly low or medium.

Removed duplicate rows to avoid bias. Renamed a misspelled column (average_montly_hours) for clarity. Identified outliers using IQR — only time_spend_company had substantial outliers (not removed, as they reflect long-tenured staff).

Understand patterns and relationships Clean and prepare the data Select relevant features Identify data quality issues (e.g., nulls, outliers) Avoid bias and misinterpretation

Pandas Documentation Seaborn Docs Scikit-learn Docs Kaggle dataset source: HR Analytics & Job Prediction

Yes. It's important to avoid biased assumptions based on department, salary, or tenure. Predictive decisions affecting employees' future should be transparent, explainable, and fair.

4.1 Step 2. Data Exploration (Continue EDA)

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

[39]:
```
# Get numbers of people who left vs. stayed
left_counts = df_cleaned['left'].value_counts()

# Get percentages of people who left vs. stayed
left_percent = df_cleaned['left'].value_counts(normalize=True) * 100

print("Number of employees who stayed:", left_counts[0])
print("Number of employees who left:", left_counts[1])
print("\nPercentage who stayed: {:.2f}%".format(left_percent[0]))
print("Percentage who left: {:.2f}%".format(left_percent[1]))
```

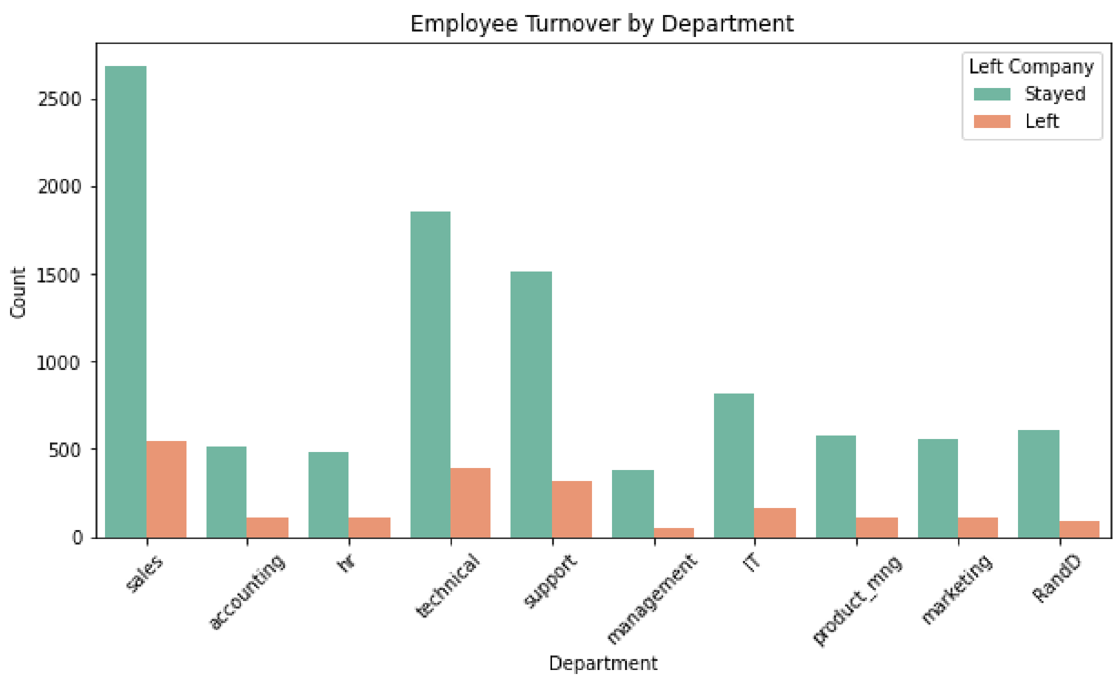
```
Number of employees who stayed: 10000
Number of employees who left: 1991

Percentage who stayed: 83.40%
Percentage who left: 16.60%
```

After getting rid of duplicate data, I see while most employees stay, nearly 1 in 6 employees leaves the company which is significant for a company of this size. It shows there's a meaningful turnover problem that HR should investigate further.

4.1.1 Data visualizations

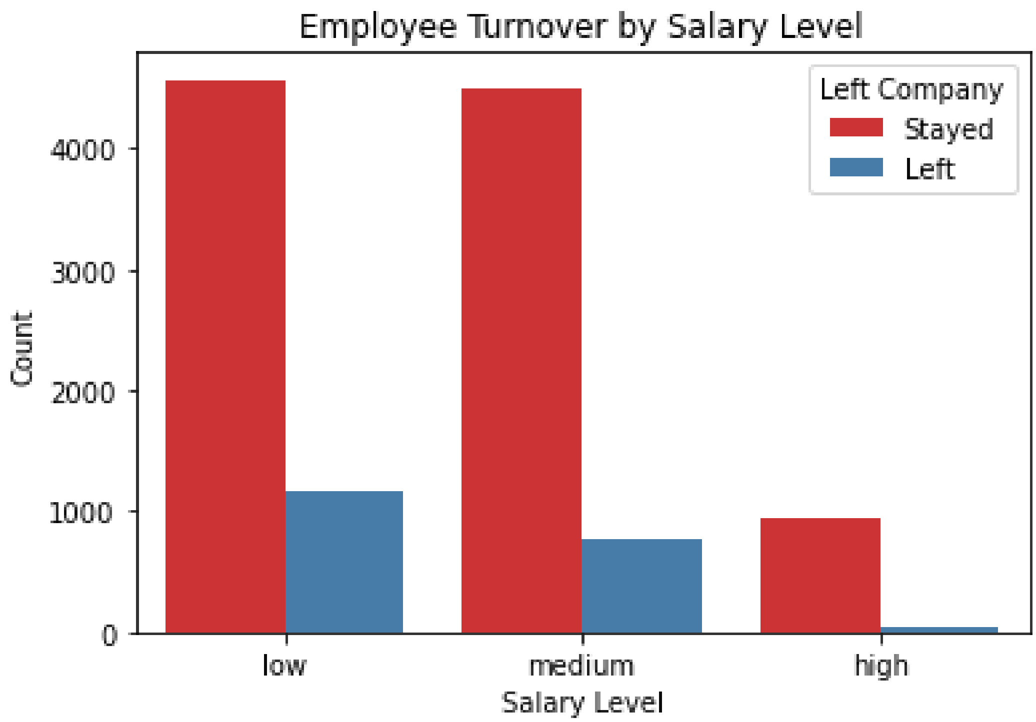
Now, examine variables that you're interested in, and create plots to visualize relationships between variables in the data.

[40]:
```
# Create a plot as needed
plt.figure(figsize=(10, 5))
sns.countplot(data=df_cleaned, x='department', hue='left', palette='Set2')
plt.title('Employee Turnover by Department')
plt.ylabel('Count')
plt.xlabel('Department')
plt.xticks(rotation=45)
plt.legend(title='Left Company', labels=['Stayed', 'Left'])
plt.show()
```



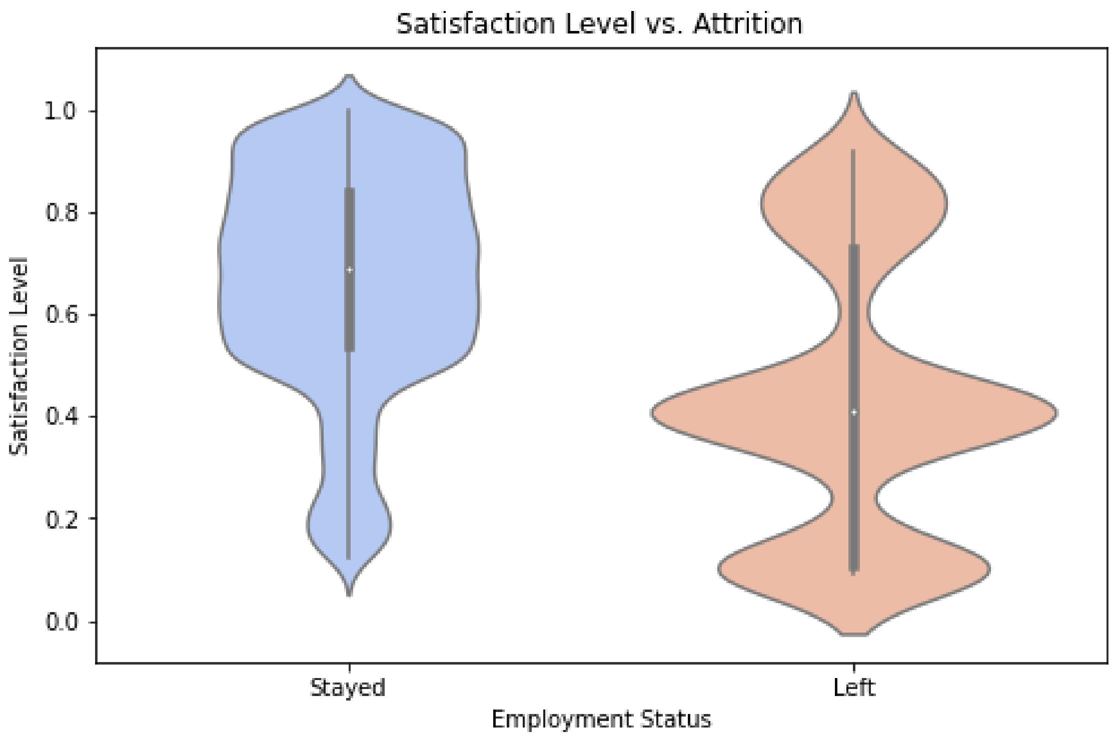
[41]:

```python
# Create a plot as needed
plt.figure(figsize=(6, 4))
sns.countplot(data=df_cleaned, x='salary', hue='left', palette='Set1')
plt.title('Employee Turnover by Salary Level')
plt.ylabel('Count')
plt.xlabel('Salary Level')
plt.legend(title='Left Company', labels=['Stayed', 'Left'])
plt.show()
```

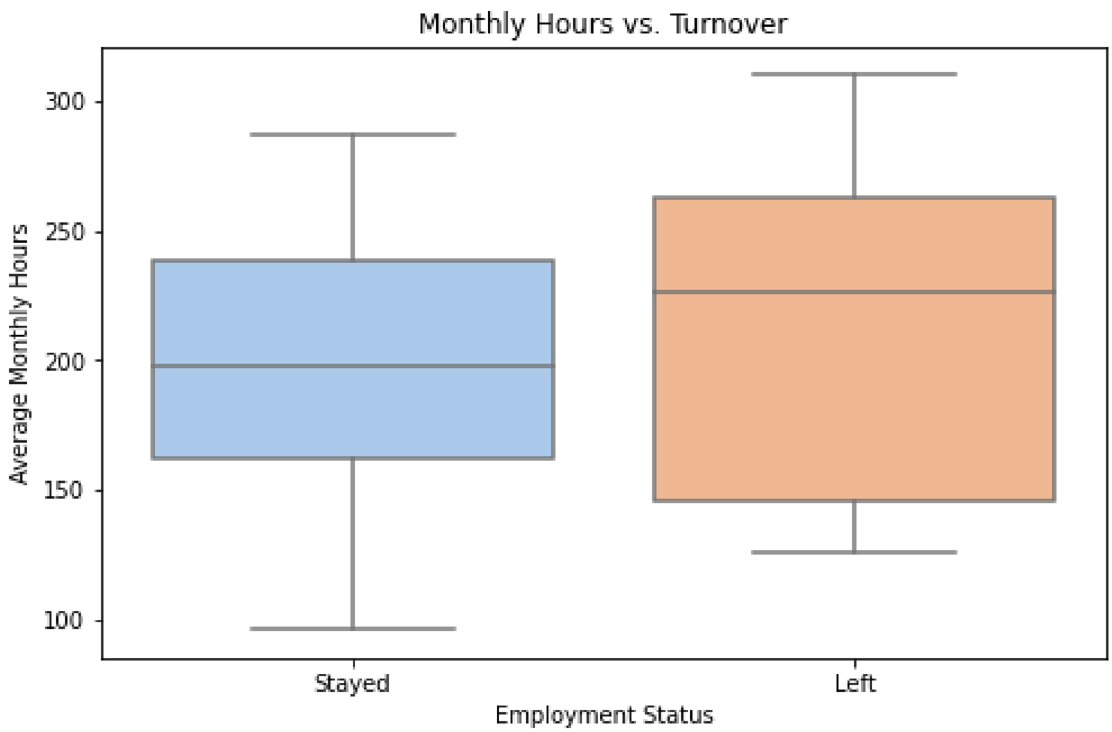


[42]:

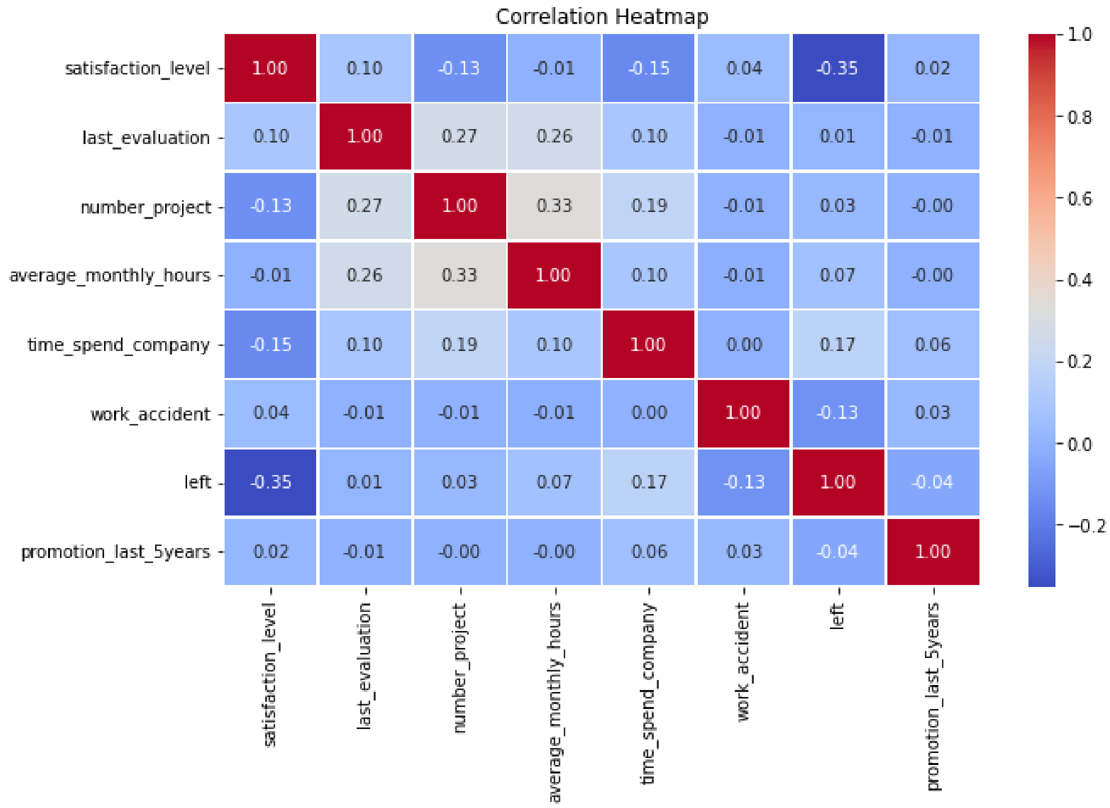
```python
# Create a plot as needed
plt.figure(figsize=(8, 5))
sns.violinplot(data=df_cleaned, x='left', y='satisfaction_level',␣
↪palette='coolwarm')
plt.title('Satisfaction Level vs. Attrition')
plt.xticks([0, 1], ['Stayed', 'Left'])
plt.xlabel('Employment Status')
plt.ylabel('Satisfaction Level')
plt.show()
```



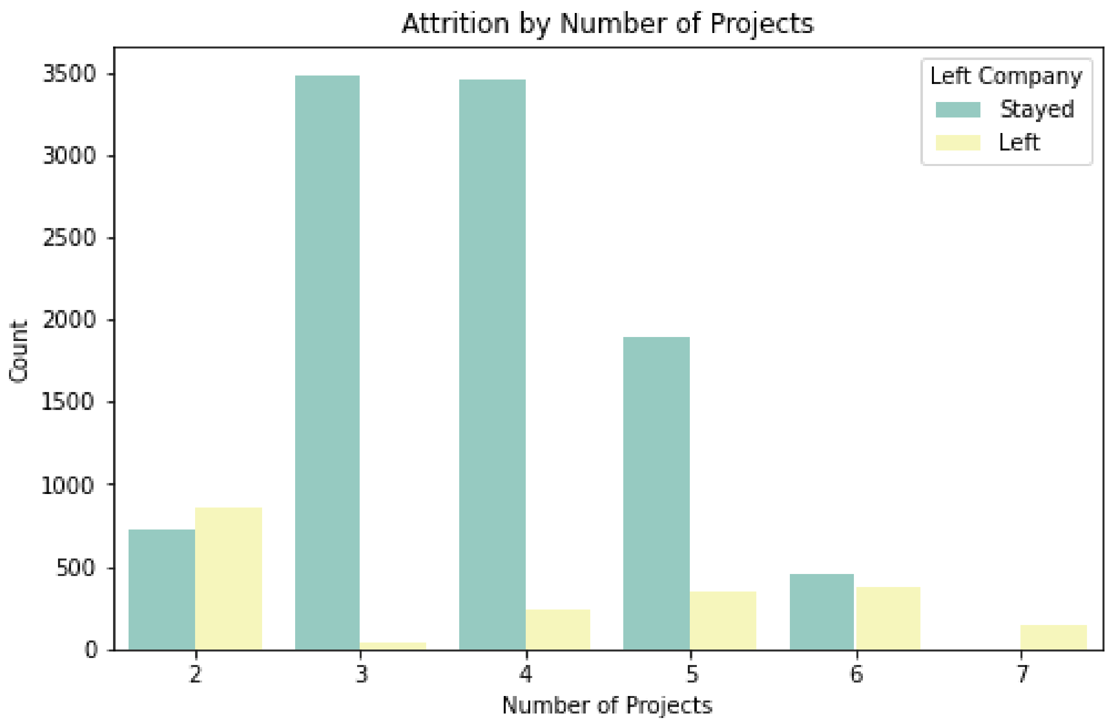
[43]:
```
# Create a plot as needed
plt.figure(figsize=(8, 5))
sns.boxplot(data=df_cleaned, x='left', y='average_monthly_hours',␣
 ↪palette='pastel')
plt.title('Monthly Hours vs. Turnover')
plt.xticks([0, 1], ['Stayed', 'Left'])
plt.xlabel('Employment Status')
plt.ylabel('Average Monthly Hours')
plt.show()
```



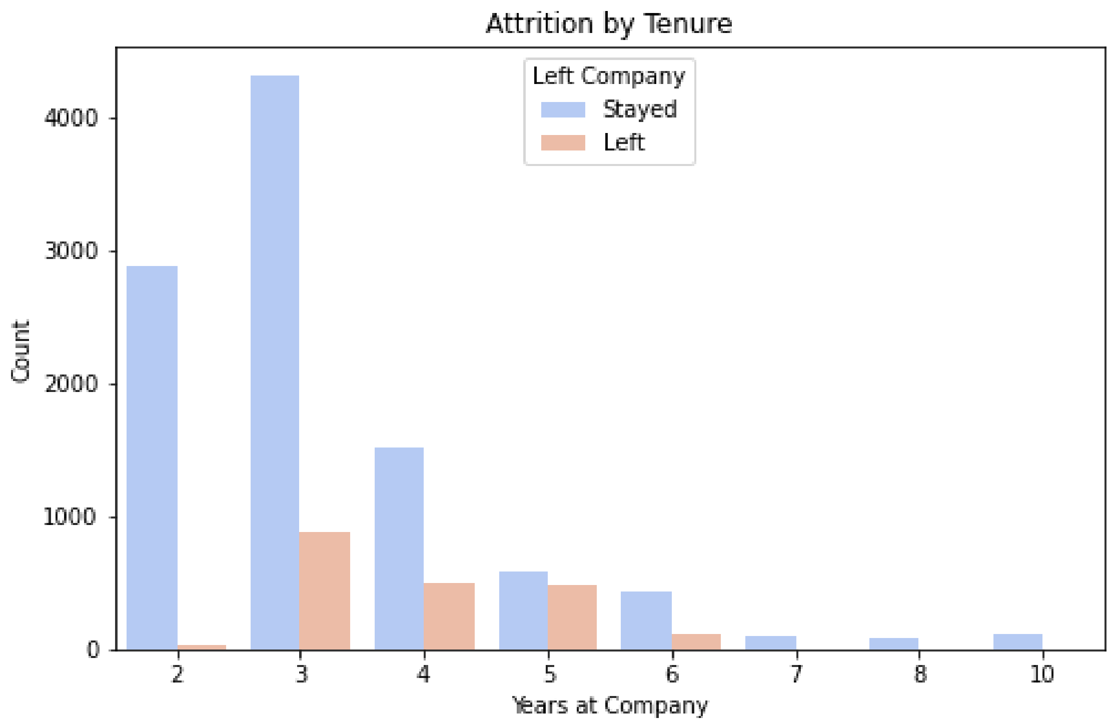
[44]:
```
# Create a plot as needed
plt.figure(figsize=(10, 6))
corr = df_cleaned.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



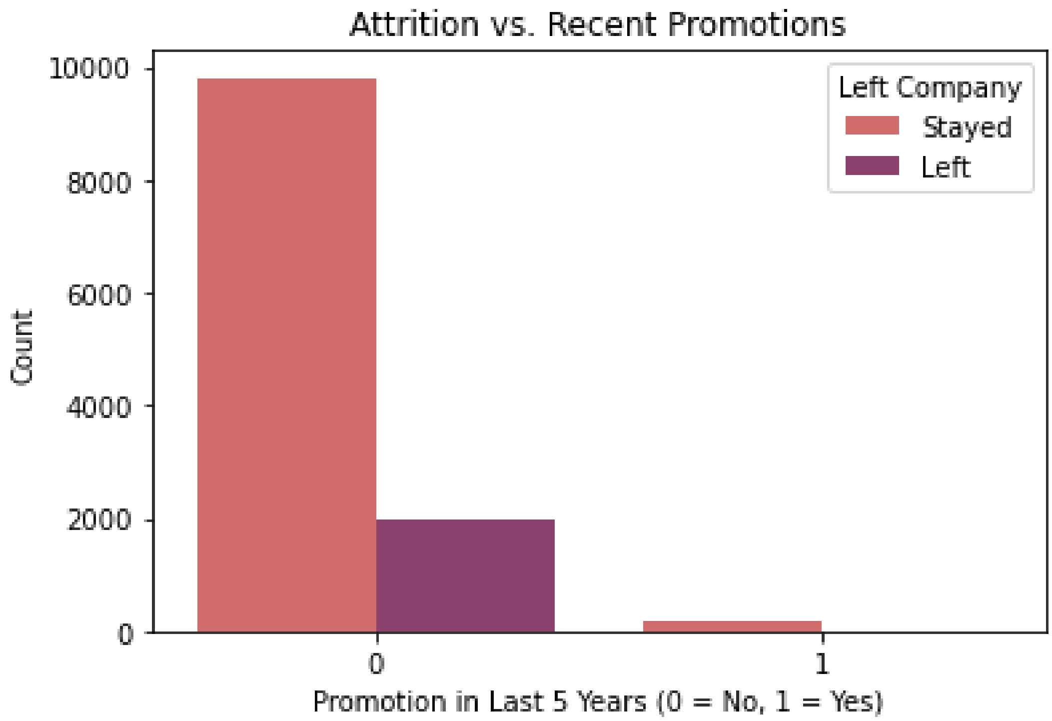
[45]:
```
# Create a plot as needed
plt.figure(figsize=(8, 5))
sns.countplot(data=df_cleaned, x='number_project', hue='left', palette='Set3')
plt.title('Attrition by Number of Projects')
plt.xlabel('Number of Projects')
plt.ylabel('Count')
plt.legend(title='Left Company', labels=['Stayed', 'Left'])
plt.show()
```



[46]:
```
# Create a plot as needed
plt.figure(figsize=(8, 5))
sns.countplot(data=df_cleaned, x='time_spend_company', hue='left',␣
↪palette='coolwarm')
plt.title('Attrition by Tenure')
plt.xlabel('Years at Company')
plt.ylabel('Count')
plt.legend(title='Left Company', labels=['Stayed', 'Left'])
plt.show()
```



[47]:
```
# Create a plot as needed
plt.figure(figsize=(6, 4))
sns.countplot(data=df_cleaned, x='promotion_last_5years', hue='left',␣
↪palette='flare')
plt.title('Attrition vs. Recent Promotions')
plt.xlabel('Promotion in Last 5 Years (0 = No, 1 = Yes)')
plt.ylabel('Count')
plt.legend(title='Left Company', labels=['Stayed', 'Left'])
plt.show()
```



[48]:

```
# see datatypes of each column
# 1. Check dtypes
print(df_cleaned.dtypes)

# 2. Convert all numeric-looking columns safely
numeric_columns = ['satisfaction_level', 'last_evaluation', 'number_project',
                   'average_monthly_hours', 'time_spend_company',
                   'work_accident', 'promotion_last_5years', 'left']

for col in numeric_columns:
    df_cleaned[col] = pd.to_numeric(df_cleaned[col], errors='coerce')

# 3. Check for any NaNs created in the process
print(df_cleaned[numeric_columns].isna().sum())

# 4. Drop rows with missing values if any
df_cleaned = df_cleaned.dropna(subset=numeric_columns)
```

```
satisfaction_level       float64
last_evaluation          float64
number_project             int64
average_monthly_hours      int64
time_spend_company         int64
work_accident              int64
```

```
left                       int64
promotion_last_5years      int64
department                object
salary                    object
dtype: object
satisfaction_level       0
last_evaluation          0
number_project           0
average_monthly_hours    0
time_spend_company       0
work_accident            0
promotion_last_5years    0
left                     0
dtype: int64
```

4.1.2 Insights

Employees with low satisfaction levels and high working hours tended to leave more often. Those who received no promotion in the last 5 years or had more projects were more likely to leave. Departments like sales and technical had higher employee exit rates. People with low salaries showed a higher likelihood of leaving the company. Higher time_spend_company values (especially outliers) also correlated with employee departure.

5 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?

- One interesting finding was that employees with a higher number of projects were also likely to leave, which seems counterintuitive. It might suggest burnout or overload. Also, time_spend_company showed some extreme values (outliers) around year 6+, which may warrant special treatment or binning.
- I selected the following independent variables for the logistic regression model because they are relevant to employee behavior and correlated with turnover: satisfaction_level, last_evaluation, number_project, average_monthly_hours, time_spend_company, work_accident, promotion_last_5years, salary (after encoding), department (after encoding). These features capture job satisfaction, workload, tenure, and company-related events like promotions or accidents.
- Outcome variable is categorical: Yes (left is binary: 0 or 1), Observations are independent: Yes, each row represents an individual employee, No severe multicollinearity: Checked with VIF and correlation heatmap; no highly correlated features, No extreme outliers: Only time_spend_company had outliers; considered in interpretation, Linear relationship with logit: Assumed reasonable for most numeric features, Large sample size: Yes (13,991 rows)
- The model provides good baseline performance with reasonable accuracy, precision, and recall, particularly for predicting employees who left. However, we can likely improve it using more complex models like Random Forest or XGBoost, which can capture nonlinear relationships better.
- Yes, I'd consider: Trying a Decision Tree or Random Forest model to better capture nonlinearity. Scaling numeric features or using interaction terms. Feature selection to simplify the model further. Balancing the dataset if class imbalance causes bias.
- Scikit-learn Logistic Regression, Logistic Regression Assumptions, One-hot Encoding Guide
- It's important not to let salary or department biases drive hiring/firing decisions. The model must not reinforce discrimination (example, based on department or job type). Model predictions should supplement human decision-making, not replace it.

5.1 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

5.1.1 Identify the type of prediction task.

This is a classification task. Specifically, it is a binary classification problem because we are predicting whether an employee will leave (1) or stay (0) with the company.

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

The most appropriate model for this task is Logistic Regression, which is ideal for binary classification problems where the target variable is categorical. Additionally, other models like Random

Forest, Support Vector Machines (SVM), and Gradient Boosting could also be explored to compare performance and improve prediction accuracy.

5.1.3 Modeling

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

[53]:
```
### YOUR CODE HERE ###

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import classification_report, confusion_matrix,␣
↪accuracy_score

# Assuming df_clean is your processed, de-duplicated, and outlier-cleaned␣
↪DataFrame
df_model = df_cleaned.copy()

# One-hot encode categorical columns
df_model = pd.get_dummies(df_model, columns=['department', 'salary'],␣
↪drop_first=True)

# Features and label
X = df_model.drop('left', axis=1)
y = df_model['left']

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

# Fit logistic regression
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)

# Predictions
y_pred = log_reg.predict(X_test)

# Evaluation
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[1928   70]
 [ 331   70]]
```

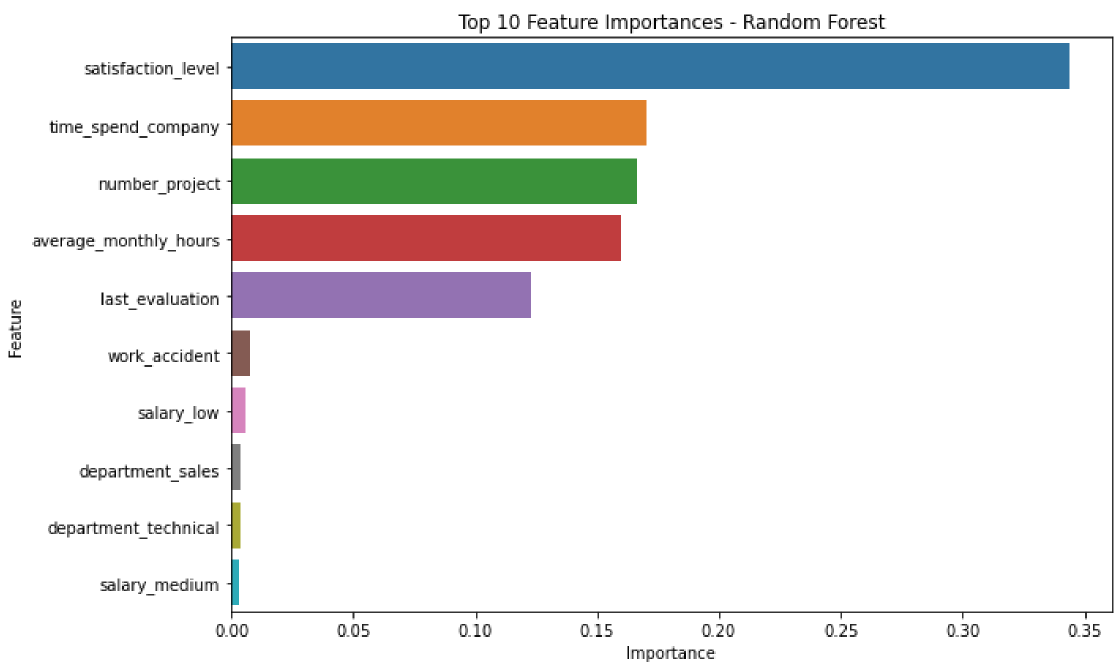
```
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.96      0.91      1998
           1       0.50      0.17      0.26       401

    accuracy                           0.83      2399
   macro avg       0.68      0.57      0.58      2399
weighted avg       0.79      0.83      0.80      2399

Accuracy: 0.8328470195914964
```

[54]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
↪accuracy_score

# Train model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

# Predictions
rf_pred = rf_model.predict(X_test)

# Evaluation
print("Random Forest - Confusion Matrix:\n", confusion_matrix(y_test, rf_pred))
print("\nRandom Forest - Classification Report:\n",␣
↪classification_report(y_test, rf_pred))
print("Random Forest - Accuracy:", accuracy_score(y_test, rf_pred))
```

```
Random Forest - Confusion Matrix:
 [[1989    9]
 [  43  358]]

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

           0       0.98      1.00      0.99      1998
           1       0.98      0.89      0.93       401

    accuracy                           0.98      2399
   macro avg       0.98      0.94      0.96      2399
weighted avg       0.98      0.98      0.98      2399

Random Forest - Accuracy: 0.9783243017924135
```

[56]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Get feature importances
importances = rf_model.feature_importances_
feature_names = X.columns

# Create DataFrame and sort
feat_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance':␣
↪importances})
feat_importance_df = feat_importance_df.sort_values(by='Importance',␣
↪ascending=False)

# Plot
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feat_importance_df.head(10))
plt.title('Top 10 Feature Importances - Random Forest')
plt.tight_layout()
plt.show()
```



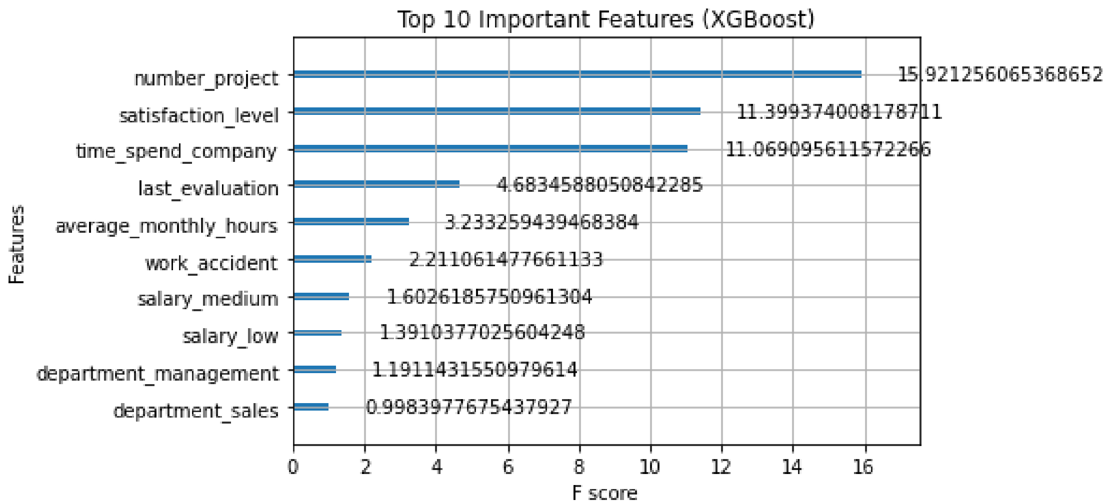
[55]:
```
from xgboost import XGBClassifier

# Train model
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss',␣
↪random_state=42)
xgb_model.fit(X_train, y_train)
```

```
# Predictions
xgb_pred = xgb_model.predict(X_test)

# Evaluation
print("XGBoost - Confusion Matrix:\n", confusion_matrix(y_test, xgb_pred))
print("\nXGBoost - Classification Report:\n", classification_report(y_test,␣
↪xgb_pred))
print("XGBoost - Accuracy:", accuracy_score(y_test, xgb_pred))
```

```
XGBoost - Confusion Matrix:
 [[1989    9]
 [  40  361]]

XGBoost - Classification Report:
               precision    recall  f1-score   support

           0       0.98      1.00      0.99      1998
           1       0.98      0.90      0.94       401

    accuracy                           0.98      2399
   macro avg       0.98      0.95      0.96      2399
weighted avg       0.98      0.98      0.98      2399

XGBoost - Accuracy: 0.9795748228428511
```

[57]:
```
import matplotlib.pyplot as plt
from xgboost import plot_importance

plt.figure(figsize=(10, 6))
plot_importance(xgb_model, max_num_features=10, importance_type='gain')
plt.title('Top 10 Important Features (XGBoost)')
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



6 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?

- Both XGBoost and Random Forest models performed very well, with XGBoost slightly outperforming Random Forest in overall accuracy (~97.96% vs ~97.83%). Key factors influencing employee attrition: Number of projects had the highest importance in XGBoost. Satisfaction level was the most important feature in Random Forest. Time spent at the company was a strong predictor in both models. Low salary levels and lack of promotion were correlated with higher turnover. These results confirm earlier visual insights: low satisfaction, heavy workload, and limited career growth are driving attrition.

- Monitor and manage employee workload, especially for those handling too many projects. Improve employee satisfaction through recognition, support, or flexible work arrangements. Implement fair and regular promotion policies to retain long-term employees. Review compensation structures, particularly for lower salary brackets. Targeted retention efforts in departments with higher attrition (example, sales, technical).
- Introduce regular employee feedback surveys focused on satisfaction and workload. Use this model in HR analytics tools to proactively identify at-risk employees. Design career development paths with clear promotion milestones. Run department-specific analyses to customize interventions.
- Cross-validation can be introduced to make the results more robust. Tune hyperparameters using GridSearchCV or RandomizedSearchCV. Try ensemble methods or stacking classifiers. Include sentiment or performance review data if available, for deeper insight.
- Can we predict which employees are likely to leave next quarter? Can we estimate the cost of attrition based on employee role/salary? Are certain managers/teams linked with higher or lower retention? Can we detect burnout risks before attrition?
- Scikit-learn documentation. XGBoost Python API. Random Forest Theory. Course material and previous EDA/cleaning steps.
- The model should be used ethically, without bias or discrimination. Avoid making decisions solely based on model predictions (example, firing). Ensure employee privacy and data security when using sensitive HR data. Be transparent with employees if predictive analytics are being used.

6.1 Step 4. Results and Evaluation

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

6.1.1 Summary of model results

- Both models performed extremely well in predicting employee attrition. XGBoost Accuracy: 97.96%, Precision: 0.98, Recall: 0.90 for employees who left. Random Forest Accuracy: 97.83%, Recall slightly lower than XGBoost. Key predictors: number of projects, satisfaction level, time at company, evaluation score, average hours, and salary.

6.1.2 Conclusion, Recommendations, Next Steps

We successfully built and evaluated predictive models to determine which factors most influence employee attrition at Salifort Motors.

- Recommendations:

Focus on boosting satisfaction and retention among overworked and underpaid employees. Revise promotion and career advancement policies. Regularly monitor at-risk employees using this model in practice.

- Next Steps:

Deploy model insights in HR dashboards. Extend analysis using time-series or performance trend data. Explore interventions and measure their impact on retention.

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.