Pace: Plan Stage

- Understand your data in the problem context
- Consider how your data will best address the business need
- Contextualize & understand the data and the problem

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.

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)

Step 1. Imports

- Import packages
- Load dataset

Import packages

In [2]:
```
# Import required libraries
import numpy as np
import pandas as pd

# Set display options to show all columns
pd.set_option('display.max_columns', None)

# Import visualization libraries and set matplotlib to display inline
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

# Import machine learning models from xgboost, sklearn.linear_model, sklearn.tree, sklearn.ensemble
from xgboost import XGBClassifier, XGBRegressor, plot_importance
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier

# Import required libraries for model evaluation
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, ConfusionMatrixD
from sklearn.metrics import roc_auc_score, roc_curve

# Import pickle for model serialization and deserialization

import pickle
```

Load dataset

The dataset is located in the same folder as this notebook (i.e., in the current working directory on Coursera), and is called `HR_capstone_dataset.csv`. You can read in this data directly from Coursera without having to download it.

In [3]:
```
# Load dataset into a dataframe


df = pd.read_csv("HR_capstone_dataset.csv")

# Display first few rows of the dataframe

df.head()
```

Out[3]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
0	0.38	0.53	2	157	3	0	1	0
1	0.80	0.86	5	262	6	0	1	0
2	0.11	0.88	7	272	4	0	1	0
3	0.72	0.87	5	223	5	0	1	0
4	0.37	0.52	2	159	3	0	1	0

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

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

Gather basic information about the data

In [4]:
```
# Gather basic information about the data
df.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
```

Gather descriptive statistics about the data

In [5]:
```
# Gather descriptive statistics about the data
df.describe()
```

Out[5]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotic
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.238083	
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	

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.

In [6]:
```
# Display all column names
df.columns
```

Out[6]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
       'promotion_last_5years', 'Department', 'salary'],
      dtype='object')
```

In [7]:
```
# Rename columns as needed
df.columns = df.columns.str.lower()
df = df.rename(columns={'average_montly_hours': 'average_monthly_hours'})


# Display all column names after the update
### YOUR CODE HERE ###
df.columns
```

Out[7]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_monthly_hours', 'time_spend_company', 'work_accident', 'left',
       'promotion_last_5years', 'department', 'salary'],
      dtype='object')
```

Check missing values

Check for any missing values in the data.

In [8]:
```
# Check for missing values
df.isnull().sum()
```

Out[8]:
```
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
```

Check duplicates

Check for any duplicate entries in the data.

In [9]:
```
# Check for duplicates

df.duplicated().sum()
```

Out[9]:
```
3008
```

In [10]:
```
# Inspect some rows containing duplicates as needed
df[df.duplicated()].head()
```

Out[10]:

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	work_accident	left	promotion_last_5y
396	0.46	0.57	2	139	3	0	1	
866	0.41	0.46	2	128	3	0	1	
1317	0.37	0.51	2	127	3	0	1	
1368	0.41	0.52	2	132	3	0	1	
1461	0.42	0.53	2	142	3	0	1	

In [11]:
```
# Drop duplicates and save resulting dataframe in a new variable as needed
df1 = df.drop_duplicates(keep="first")
# Display first few rows of new dataframe as needed
df1.head()
```

Out[11]:

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	work_accident	left	promotion_last_5years
0	0.38	0.53	2	157	3	0	1	0
1	0.80	0.86	5	262	6	0	1	0
2	0.11	0.88	7	272	4	0	1	0
3	0.72	0.87	5	223	5	0	1	0
4	0.37	0.52	2	159	3	0	1	0

Check outliers

Check for outliers in the data.

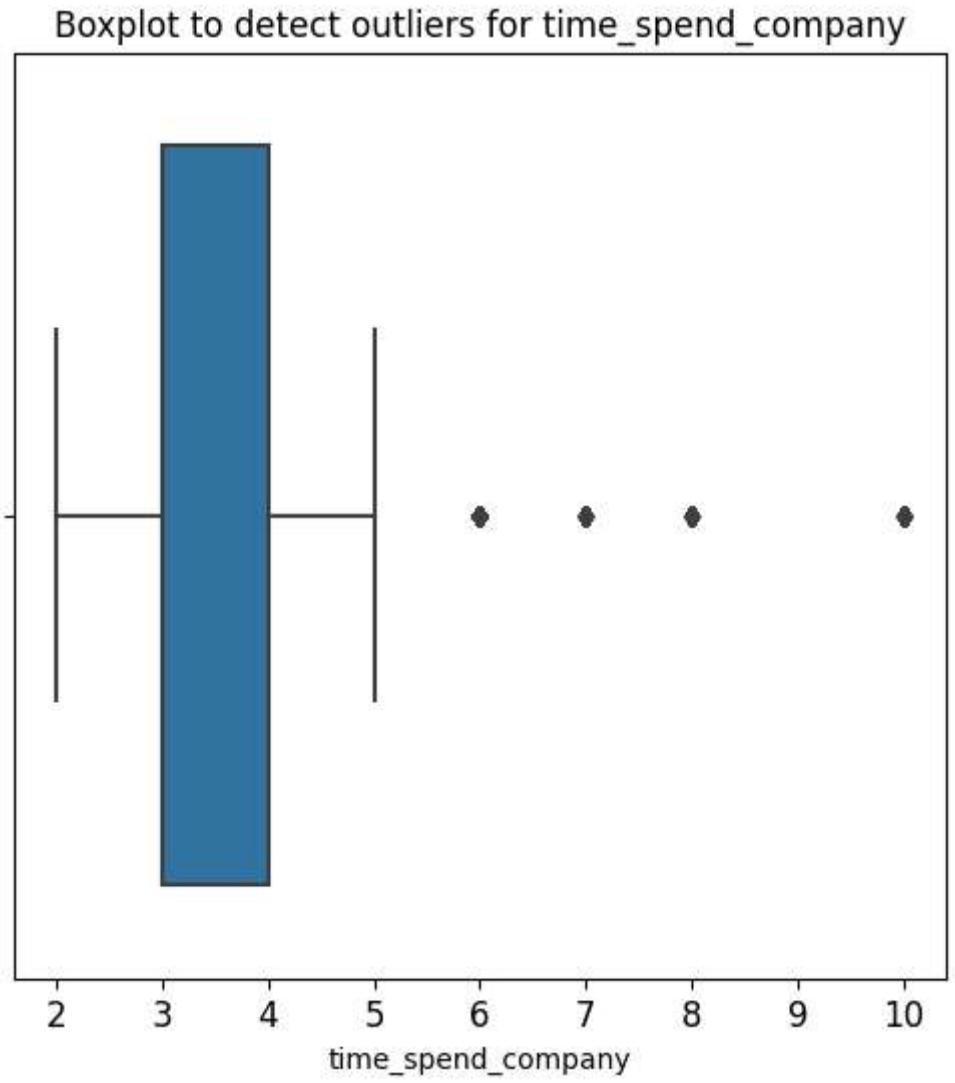
In [12]:
```
# Create a boxplot to visualize distribution of `time_spend_company` and detect any outliers
plt.figure(figsize=(6,6))

# Create a boxplot for the 'time_spend_company' column of the dataframe
sns.boxplot(x=df1['time_spend_company'])

# Set the title of the plot and the font size for the title and axes ticks
plt.title('Boxplot to detect outliers for time_spend_company', fontsize=12)
```

```
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

# Display the plot
plt.show()
```



In [13]:
```
# Determine the number of rows containing outliers

# Get the summary statistics for the 'time_spend_company' column
tsc_stats = df1['time_spend_company'].describe()
```

```
# Calculate the interquartile range (IQR)
Q1 = tsc_stats['25%']
Q3 = tsc_stats['75%']
IQR = Q3 - Q1


# Determine the upper and lower bounds for outliers
lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR


# Count the number of rows containing outliers
outlier_count = ((df1['time_spend_company'] < lower_bound) | (df1['time_spend_company'] > upper_bound)).sum()

print("number of rows that have outliers in time_spend_company:", outlier_count)
```

```
number of rows that have outliers in time_spend_company: 824
```

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.