Activity_ Course 7 Salifort Motors project lab

December 21, 2023

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 of the project are the company's top management.
- The goals in this project are to analyse the data collected by the HR department and to build a model that predicts whether or not an employee will leave the company.
- 25% of the data contained missing values and was therefore removed. Most of the data features are numeric, except for only two which are department and salary, which are categorical.
- After checking for outlier data, the new dataframe contained 824 data outliers in the "tenure"

column. This should be no issue for decesion tree-based models, the XGB, however the Logistic Regression model is expected to be affected by them.
- Resources used throughout the project will be the company's dataset, besides python notebooks and libraries.
- Both FN and FP could prove problematic for the company. FN means the model mispredicts that an employee will not leave the company, leading the latter to not exerting necessary efforts and allocating necessary resources in order to devise the suitable policies to retain staff. On the other hand, FP means the model mispredicts that an employee will leave the company, leading the latter to dedicate unnecessary more resources for staff retention, increasing its cost accordingly. In such cases, F1 could be the most suitable score to measure the model's performance as it strikes the balance between precision and recall scores.

2.2 Step 1. Imports

- Import packages
- Load dataset

2.2.1 Import packages

[2]:
```
# Import packages
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, GridSearchCV
from sklearn.metrics import classification_report, accuracy_score,␣
↪precision_score, \
recall_score, f1_score, confusion_matrix, ConfusionMatrixDisplay
from xgboost import XGBClassifier
from xgboost import plot_importance
```

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 ###
hr = pd.read_csv("HR_capstone_dataset.csv")
```

```python
# Display first few rows of the dataframe
hr.head()
```

[3]:
```
   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
```

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

[60]:
```python
# Gather basic information about the data
hr.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
```

2.3.2 Gather descriptive statistics about the data

[61]:
```python
# Gather descriptive statistics about the data
hr.describe()
```

[61]:
```
       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
```

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.

[5]:
```
# Rename columns as needed
hr = hr.rename(columns={"Work_accident" : "work_accident",␣
↪"average_montly_hours" : "avg_monthly_hrs",
                    "time_spend_company" : "tenure", "Department" :␣
↪"department"})


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

[5]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'avg_monthly_hrs', 'tenure', 'work_accident', 'left',
       'promotion_last_5years', 'department', 'salary'],
      dtype='object')
```

2.3.4 Check missing values

Check for any missing values in the data.

[64]:
```
# Check for missing values
hr.isna().sum()
```

[64]:
```
satisfaction_level       0
last_evaluation          0
number_project           0
avg_monthly_hrs          0
tenure                   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.

[4]:
```
# Check for duplicates
hr.duplicated().sum()
```

[4]:
```
3008
```

[66]:
```
# Inspect some rows containing duplicates as needed
mask = hr.duplicated()
```

```
hr[mask]
```

[66]:
```
       satisfaction_level  last_evaluation  number_project  avg_monthly_hrs  \
396                  0.46             0.57               2              139
866                  0.41             0.46               2              128
1317                 0.37             0.51               2              127
1368                 0.41             0.52               2              132
1461                 0.42             0.53               2              142
…                     …                …               …               …
14994                0.40             0.57               2              151
14995                0.37             0.48               2              160
14996                0.37             0.53               2              143
14997                0.11             0.96               6              280
14998                0.37             0.52               2              158

       tenure  work_accident  left  promotion_last_5years  department  salary
396         3              0     1                      0       sales     low
866         3              0     1                      0  accounting     low
1317        3              0     1                      0       sales  medium
1368        3              0     1                      0       RandD     low
1461        3              0     1                      0       sales     low
…       …            …   …                    …     …     …
14994       3              0     1                      0     support     low
14995       3              0     1                      0     support     low
14996       3              0     1                      0     support     low
14997       4              0     1                      0     support     low
14998       3              0     1                      0     support     low

[3008 rows x 10 columns]
```

[6]:
```
# Drop duplicates and save resulting dataframe in a new variable as needed
hr2 = hr.drop_duplicates(keep='first', ignore_index=True)


# Display first few rows of new dataframe as needed
hr2.head()
```

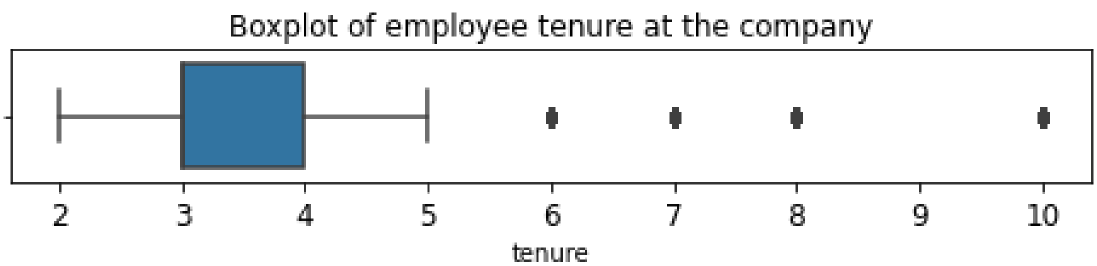
[6]:
```
   satisfaction_level  last_evaluation  number_project  avg_monthly_hrs  \
0                0.38             0.53               2              157
1                0.80             0.86               5              262
2                0.11             0.88               7              272
3                0.72             0.87               5              223
4                0.37             0.52               2              159

   tenure  work_accident  left  promotion_last_5years department  salary
0       3              0     1                      0      sales     low
1       6              0     1                      0      sales  medium
```

2	4	0	1	0	sales	medium
3	5	0	1	0	sales	low
4	3	0	1	0	sales	low

2.3.6 Check outliers

Check for outliers in the data.

[6]:
```
# Create a boxplot to visualize distribution of `tenure` and detect any outliers
plt.figure(figsize=(8,1))
plt.title('Boxplot of employee tenure at the company', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
sns.boxplot(x=hr2['tenure'])
plt.show()
```



[79]:
```
# Determine the number of rows containing outliers
Q1 = hr2['tenure'].quantile(0.25)
Q3 = hr2['tenure'].quantile(0.75)
IQR = Q3 - Q1
UL = Q3 + 1.5 * IQR
LL = Q1 - 1.5 * IQR
outlier_count = ((hr2['tenure'] > UL) | (hr2['tenure'] < LL)).sum()
print('Upper Limit:',UL)
print('Lower Limit:',LL)
print('Outlier Count:', outlier_count)
print('Rows with outliers:',np.where((hr2['tenure'] > UL) | (hr2['tenure'] <␣
↪LL)))
```

```
Upper Limit: 5.5
Lower Limit: 1.5
Outlier Count: 824
Rows with outliers: (array([    1,    17,    34,    47,    67,    83,    99,
122,   161,
         191,   199,   204,   229,   231,   251,   269,   275,   277,
         282,   315,   327,   351,   352,   397,   399,   410,   415,
```

429, 432, 481, 489, 494, 503, 514, 522, 533,
552, 571, 576, 599, 600, 624, 627, 658, 661,
666, 676, 717, 748, 799, 801, 832, 837, 846,
866, 898, 912, 926, 980, 1015, 1037, 1040, 1041,
1044, 1047, 1077, 1080, 1124, 1131, 1135, 1140, 1191,
1255, 1277, 1302, 1343, 1346, 1372, 1374, 1376, 1399,
1447, 1540, 1542, 1547, 1575, 1580, 1643, 1645, 1655,
1711, 1712, 1714, 1740, 1745, 1751, 1788, 1817, 1826,
1851, 1891, 1897, 1902, 1932, 1941, 1948, 1959, 1961,
1972, 2003, 2008, 2015, 2064, 2082, 2086, 2106, 2115,
2164, 2215, 2236, 2272, 2278, 2297, 2351, 2357, 2371,
2412, 2420, 2426, 2456, 2484, 2559, 2565, 2627, 2644,
2670, 2674, 2701, 2716, 2727, 2762, 2785, 2809, 2810,
2825, 2826, 2875, 2906, 2923, 2926, 2932, 2955, 2988,
3034, 3089, 3096, 3119, 3132, 3144, 3174, 3190, 3193,
3234, 3240, 3288, 3299, 3300, 3368, 3409, 3450, 3478,
3537, 3550, 3591, 3671, 3685, 3714, 3715, 3850, 3876,
3953, 3980, 4032, 4042, 4058, 4067, 4074, 4077, 4093,
4108, 4117, 4143, 4149, 4158, 4187, 4194, 4201, 4214,
4260, 4263, 4284, 4338, 4369, 4384, 4416, 4435, 4482,
4489, 4507, 4516, 4525, 4532, 4590, 4614, 4682, 4689,
4775, 4789, 4839, 4851, 4872, 4920, 4935, 4974, 5003,
5007, 5022, 5046, 5054, 5068, 5077, 5083, 5106, 5115,
5148, 5188, 5212, 5216, 5254, 5264, 5276, 5297, 5333,
5354, 5373, 5375, 5392, 5421, 5422, 5437, 5483, 5515,
5544, 5561, 5595, 5631, 5636, 5642, 5643, 5645, 5646,
5661, 5668, 5669, 5708, 5851, 5859, 5880, 5886, 5894,
5914, 5941, 5942, 5952, 5958, 5989, 6007, 6035, 6072,
6146, 6171, 6181, 6248, 6251, 6258, 6292, 6310, 6328,
6349, 6355, 6359, 6390, 6410, 6420, 6423, 6433, 6468,
6472, 6475, 6509, 6546, 6596, 6621, 6626, 6632, 6675,
6683, 6705, 6749, 6750, 6827, 6856, 6863, 6878, 6887,
6919, 6995, 7019, 7028, 7068, 7073, 7089, 7124, 7142,
7214, 7218, 7232, 7279, 7319, 7321, 7338, 7346, 7380,
7408, 7421, 7430, 7434, 7443, 7464, 7601, 7602, 7606,
7629, 7631, 7678, 7706, 7713, 7716, 7753, 7763, 7764,
7782, 7789, 7790, 7809, 7811, 7838, 7843, 7867, 7879,
7899, 7911, 7914, 7933, 7939, 7973, 7974, 7984, 8013,
8018, 8053, 8056, 8075, 8157, 8222, 8251, 8256, 8297,
8304, 8353, 8354, 8363, 8465, 8481, 8508, 8569, 8593,
8602, 8604, 8613, 8631, 8671, 8696, 8699, 8734, 8740,
8741, 8803, 8811, 8863, 8882, 8892, 8904, 8912, 8938,
8955, 8962, 8965, 9015, 9051, 9083, 9115, 9151, 9193,
9210, 9228, 9235, 9252, 9283, 9285, 9382, 9407, 9408,
9441, 9478, 9519, 9539, 9574, 9588, 9661, 9666, 9680,
9687, 9703, 9726, 9739, 9742, 9819, 9857, 9869, 9890,
9948, 9990, 9992, 10040, 10088, 10103, 10107, 10110, 10122,
10131, 10148, 10150, 10162, 10175, 10205, 10214, 10228, 10237,

```
10263, 10362, 10432, 10470, 10528, 10539, 10562, 10576, 10624,
10643, 10675, 10715, 10744, 10755, 10778, 10853, 10884, 10906,
10945, 10963, 10968, 10980, 10998, 10999, 11000, 11001, 11002,
11003, 11004, 11005, 11006, 11007, 11039, 11056, 11069, 11070,
11071, 11072, 11073, 11074, 11075, 11076, 11077, 11078, 11079,
11106, 11115, 11116, 11117, 11118, 11119, 11120, 11121, 11122,
11123, 11124, 11125, 11133, 11157, 11176, 11181, 11184, 11185,
11186, 11187, 11188, 11189, 11190, 11196, 11197, 11198, 11199,
11200, 11201, 11202, 11203, 11204, 11205, 11206, 11207, 11208,
11209, 11210, 11211, 11212, 11213, 11214, 11215, 11216, 11217,
11218, 11220, 11221, 11222, 11223, 11249, 11250, 11251, 11252,
11253, 11254, 11255, 11261, 11262, 11263, 11264, 11265, 11266,
11267, 11268, 11269, 11270, 11271, 11272, 11273, 11274, 11275,
11276, 11277, 11278, 11279, 11280, 11281, 11282, 11283, 11285,
11286, 11287, 11288, 11290, 11291, 11328, 11337, 11338, 11339,
11340, 11341, 11342, 11371, 11401, 11417, 11447, 11479, 11480,
11481, 11482, 11483, 11485, 11486, 11487, 11494, 11496, 11497,
11498, 11499, 11500, 11502, 11503, 11504, 11505, 11506, 11507,
11510, 11511, 11512, 11513, 11514, 11515, 11517, 11518, 11519,
11520, 11521, 11522, 11531, 11532, 11533, 11534, 11535, 11537,
11538, 11539, 11546, 11548, 11549, 11550, 11551, 11552, 11554,
11555, 11556, 11557, 11558, 11559, 11562, 11563, 11564, 11565,
11566, 11567, 11569, 11570, 11571, 11572, 11573, 11574, 11579,
11580, 11581, 11582, 11583, 11585, 11586, 11587, 11594, 11596,
11597, 11598, 11599, 11600, 11602, 11603, 11604, 11605, 11606,
11607, 11610, 11611, 11612, 11613, 11614, 11615, 11617, 11618,
11619, 11620, 11621, 11622, 11624, 11625, 11626, 11627, 11628,
11629, 11634, 11635, 11636, 11637, 11638, 11640, 11641, 11642,
11649, 11651, 11652, 11653, 11654, 11655, 11657, 11658, 11659,
11660, 11661, 11662, 11665, 11666, 11667, 11668, 11669, 11670,
11672, 11673, 11674, 11675, 11676, 11677, 11713, 11714, 11715,
11716, 11717, 11719, 11720, 11721, 11728, 11730, 11731, 11732,
11733, 11734, 11736, 11737, 11738, 11739, 11740, 11741, 11744,
11745, 11746, 11747, 11748, 11749, 11751, 11752, 11753, 11754,
11755, 11756, 11805, 11826, 11886, 11893, 11894, 11895, 11896,
11897, 11899, 11900, 11901, 11908, 11910, 11911, 11912, 11913,
11914, 11919, 11920, 11921, 11926, 11927, 11928, 11929, 11931,
11932, 11933, 11934, 11935, 11936, 11946, 11947, 11948, 11949,
11950, 11952, 11953, 11954, 11961, 11963, 11964, 11965, 11966,
11967, 11972, 11973, 11974, 11979, 11980, 11981, 11982, 11984,
11985, 11986, 11987, 11988, 11989]),)
```

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?

- There was no any linear relationship or multicollinearity observed in the data.
- Almost 17% of employess left the company, while 83% stayed. Those who left nearly represent fifth of the staff.
- EDA helps fully explore the data to have an elaborate idea about its features, for example whether it contains any missing values, data outliers, or duplicates, as well as the nature of relationship between features, i.e. the existence of any linear relationship or multicollinearity.
- The only ethical consideration thus far is to the importance of striking the balance between FNs and FPs, to minimise the company's losses in terms of staff turnover or financial resources as much as possible.

3.1 Step 2. Data Exploration (Continue EDA)

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

[7]:
```python
# Get numbers of people who left vs. stayed
print(hr2['left'].value_counts())
print()
# Get percentages of people who left vs. stayed
left_stayed_percent = round(hr2['left'].value_counts(normalize=True)*100,2)
print(left_stayed_percent)
```

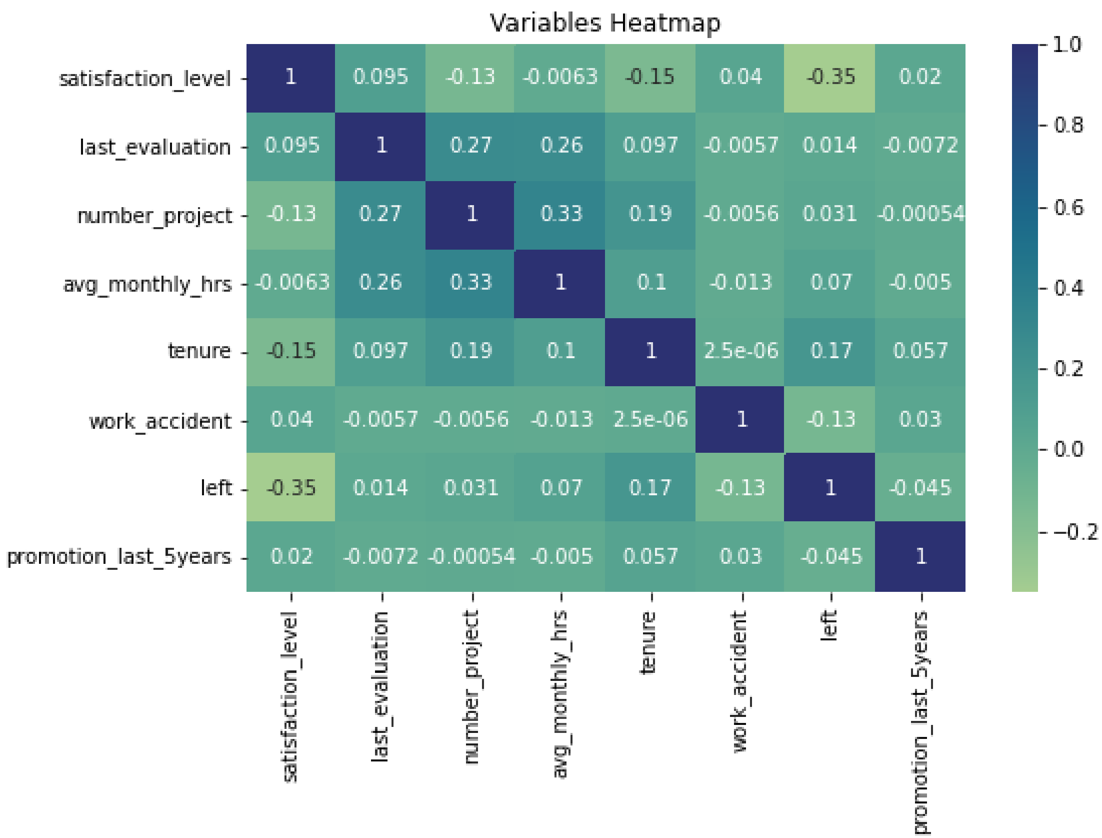
```
0    10000
1     1991
Name: left, dtype: int64

0    83.4
1    16.6
Name: left, dtype: float64
```

Almost 17% of employess left the company, while 83% stayed. Those who left nearly represent fifth of the staff.

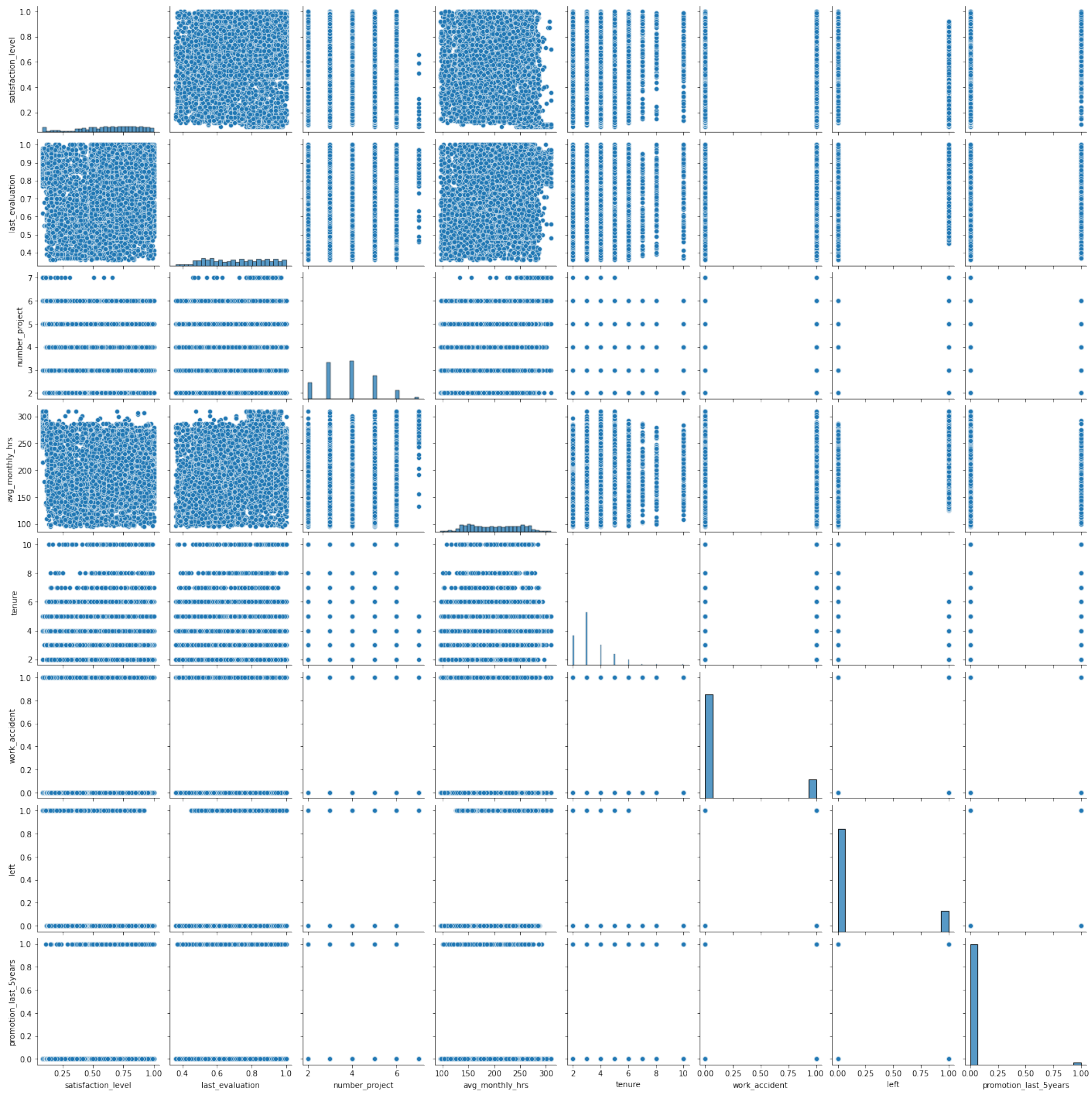
3.1.1 Data visualizations

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

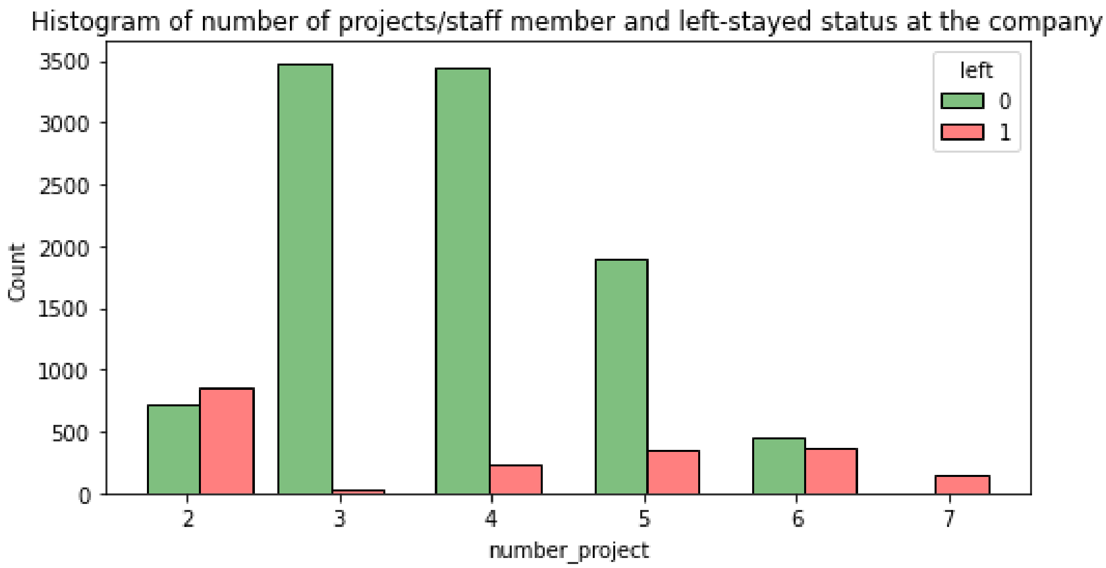
[73]:
```
# Create a plot as needed
plt.figure(figsize=(8,5))
sns.heatmap(hr2[['satisfaction_level', 'last_evaluation',␣
↪'number_project','avg_monthly_hrs', 'tenure',
                 'work_accident', 'left','promotion_last_5years', 'department',␣
↪'salary']].corr(),
    annot=True,
    cmap="crest")
plt.title("Variables Heatmap")
plt.show();
```



[74]:
```
# Create a plot as needed
sns.pairplot(hr2);
```

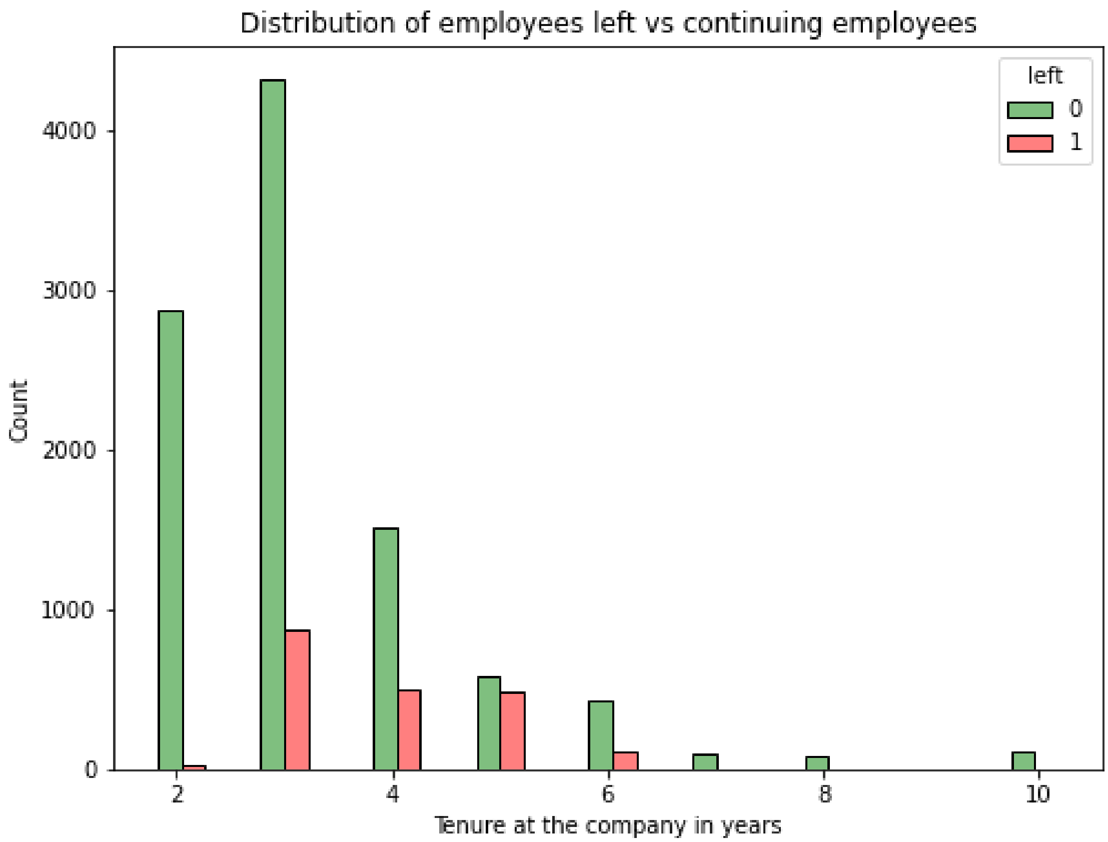


[61]:
```
# Create a plot as needed
plt.figure(figsize=(8,4))
sns.histplot(data=hr2,
             x='number_project',
             hue='left',
             palette={0:'green', 1:'red'},
             multiple='dodge',
             element='bars',
             shrink=4,
             alpha=.5)
plt.title('Histogram of number of projects/staff member and left-stayed status␣
↪at the company');
```



The above figure shows that the number of those who leave the company severely decreases when they engage three projects as compared to two projects only. 3 and 4 projects represent the highest numbers of projects engaged by those who stay, then thereafter the numbers of those who stay starts to decrease inversely with the number of projects engaged, until the latter reaches 7 projects, at which no one at the company stays. That may indicate that the maximum number of projects a staff member can engage at the company is 6.

[53]:
```
# Create a plot as needed
plt.figure(figsize=(8,6))
sns.histplot(data=hr2,
             stat="count",
             multiple="dodge",
             x="tenure",
             shrink=5,
             alpha=.5,
             hue="left",
             palette={0:'green', 1:'red'},
             element="bars", legend=True)

plt.xlabel("Tenure at the company in years")
plt.ylabel("Count")
plt.title("Distribution of employees left vs continuing employees")
plt.show();
```

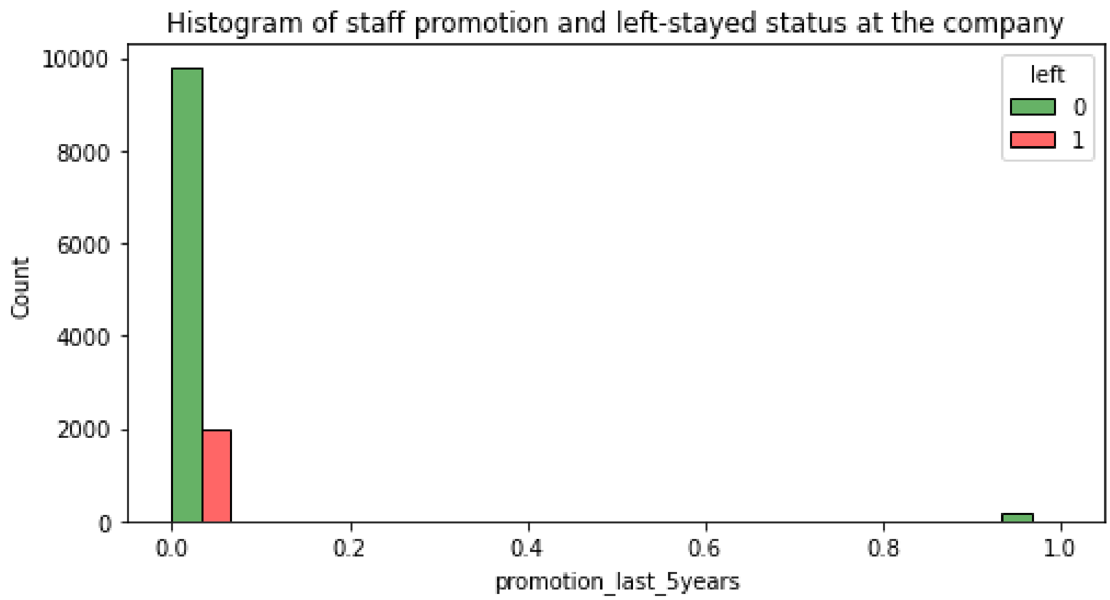


In a range of 2 to 10 years of staff tenure at the company, those who exceed 6 years stay up to ten years. Those who leave do so in between 2 to 6 years of tenure.

[45]:
```
hr2[hr2['tenure']==7]['left'].value_counts()
```

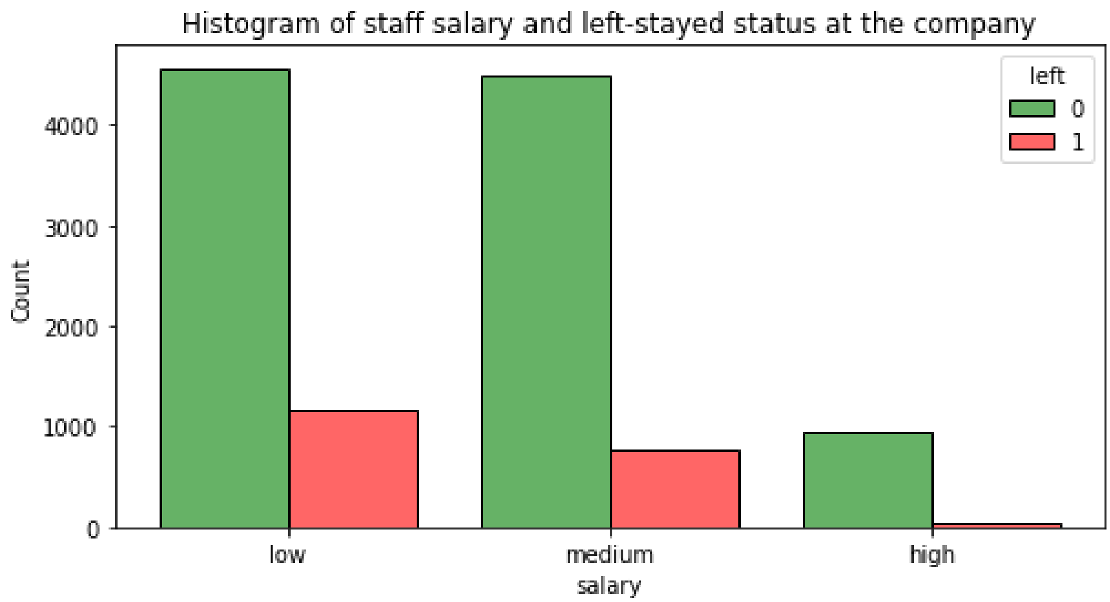
[45]:
```
False    94
Name: left, dtype: int64
```

[28]:
```
# Create a plot as needed
plt.figure(figsize=(8,4))
sns.histplot(data=hr2,
             x='promotion_last_5years',
             hue='left',
             palette={0:'green', 1:'red'},
             multiple='dodge',
             element='bars',
             shrink=1,
             alpha=.6)
plt.title('Histogram of staff promotion and left-stayed status at the company');
```

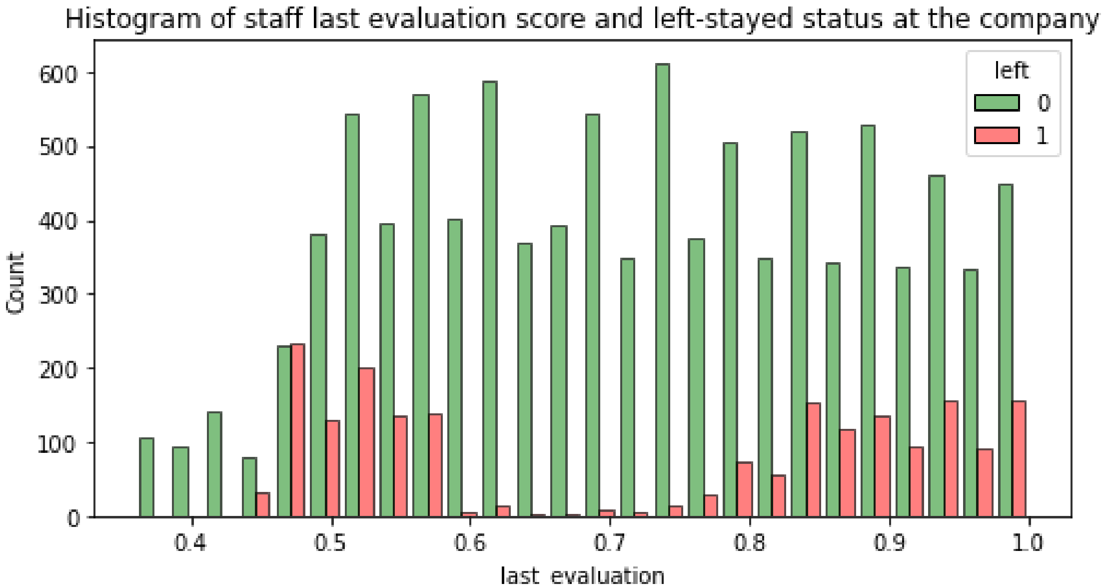


The likelihood of leaving the company increses for those who do not get promoted within the last five years.

[78]:
```python
# Create a plot as needed
plt.figure(figsize=(8,4))
sns.histplot(data=hr2,
             x='salary',
             hue='left',
             palette={0:'green', 1:'red'},
             multiple='dodge',
             element='bars',
             shrink=0.8,
             alpha=.6)
plt.title('Histogram of staff salary and left-stayed status at the company');
```




```
[54]: # Create a plot as needed
plt.figure(figsize=(8,4))
sns.histplot(data=hr2,
             x='last_evaluation',
             hue='left',
             palette={0:'green', 1:'red'},
             multiple='dodge',
             element='bars',
             shrink=0.8,
             alpha=.5)
plt.title('Histogram of staff last evaluation score and left-stayed status at␣
 ↪the company');
```

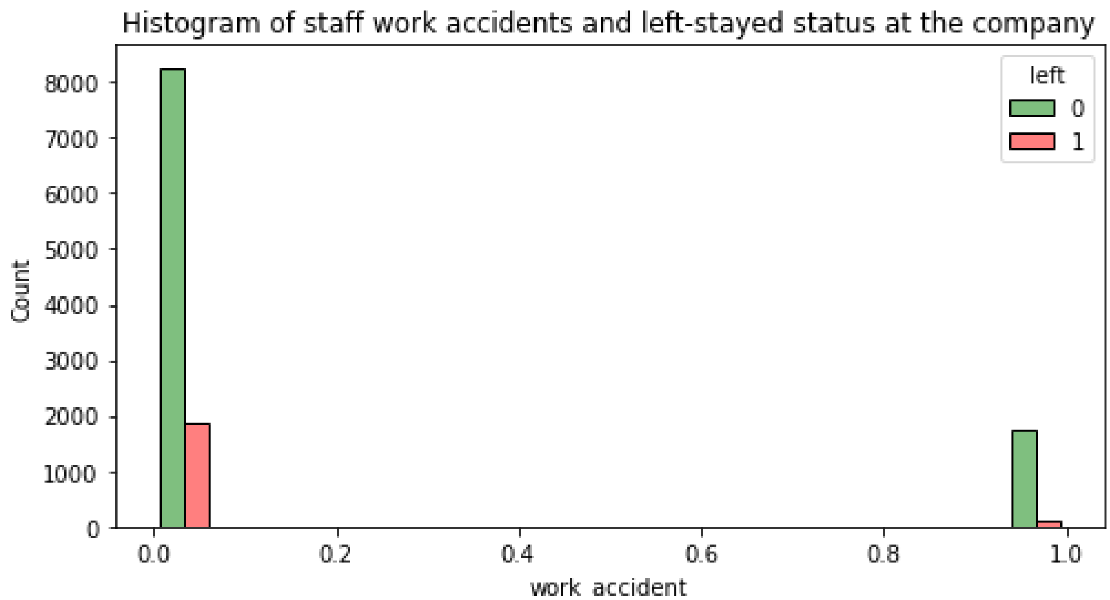


The numbers of those who stay largely exceeds the numbers of those who leave in accordnace and directly with the score they attain in their last performance evaluation. The numbers of those who left largely decreased within the range of 0.6-0.75 of last evaluation, after that the numbers have risen again. However, At any given score point, the number of those who stayed was much higher than that of those who left, reaching its highest point at score point of 0.74.

[59]:
```
hr2[hr2['last_evaluation']==0.74]['left'].value_counts()
```

[59]:
```
False    222
True       5
Name: left, dtype: int64
```

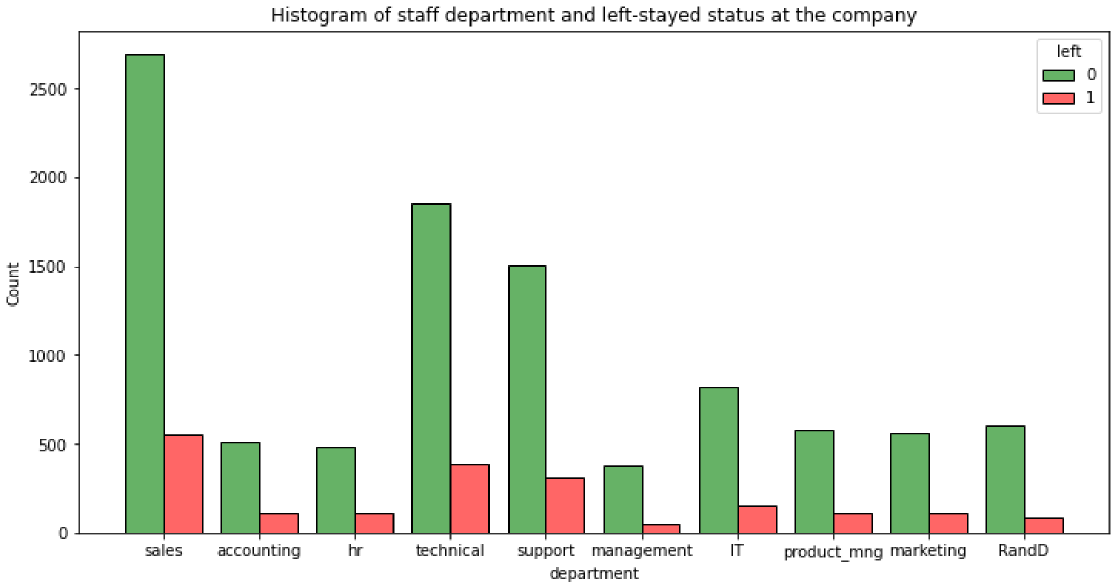
[55]:
```
# Create a plot as needed
plt.figure(figsize=(8,4))
sns.histplot(data=hr2,
             x='work_accident',
             hue='left',
             palette={0:'green', 1:'red'},
             multiple='dodge',
             element='bars',
             shrink=0.8,
             alpha=.5)
plt.title('Histogram of staff work accidents and left-stayed status at the␣
 ↪company');
```



Work accidents seem not to have any relationship with the staff member leaving or staying at the company. The proportion of those who left and those who stayed is almost the same, regardless of whether the employee had a workplace injury or not. However, the numbers of those who stayed remain higher than those who left.

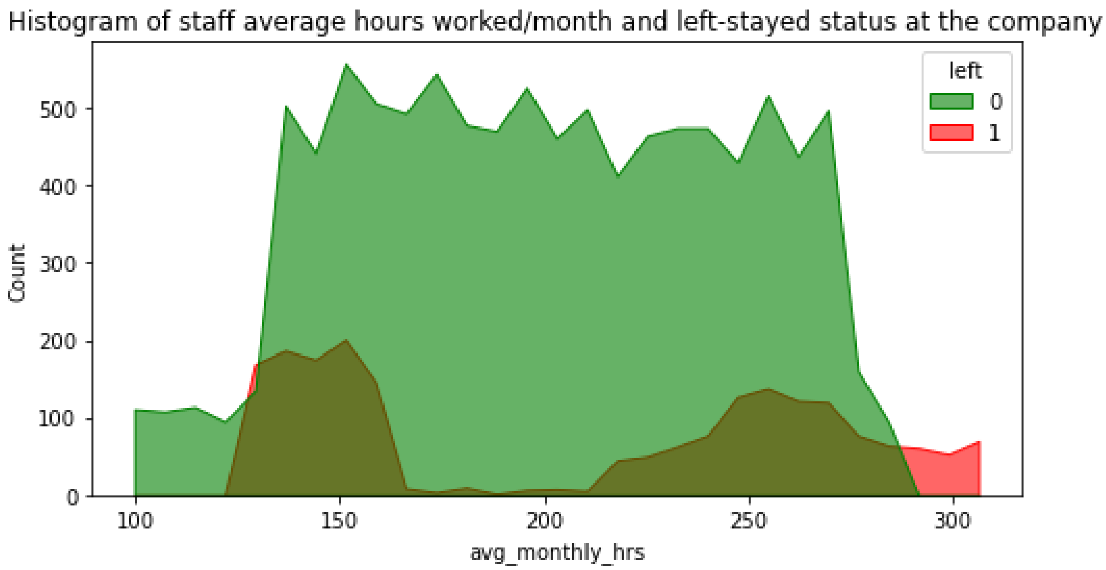
[9]:

```
# Create a plot as needed
plt.figure(figsize=(12,6))
sns.histplot(data=hr2,
             x='department',
             hue='left',
             palette={0:'green', 1:'red'},
             multiple='dodge',
             element='bars',
             shrink=0.8,
             alpha=.6)
plt.title('Histogram of staff department and left-stayed status at the␣
 ↪company');
```



[75]:

```
# Create a plot as needed
plt.figure(figsize=(8,4))
sns.histplot(data=hr2,
             x='avg_monthly_hrs',
             hue='left',
             palette={0:'green', 1:'red'},
             multiple='layer',
             element='poly',
             shrink=0.8,
             alpha=.6)
plt.title('Histogram of staff average hours worked/month and left-stayed status␣
↪at the company');
```

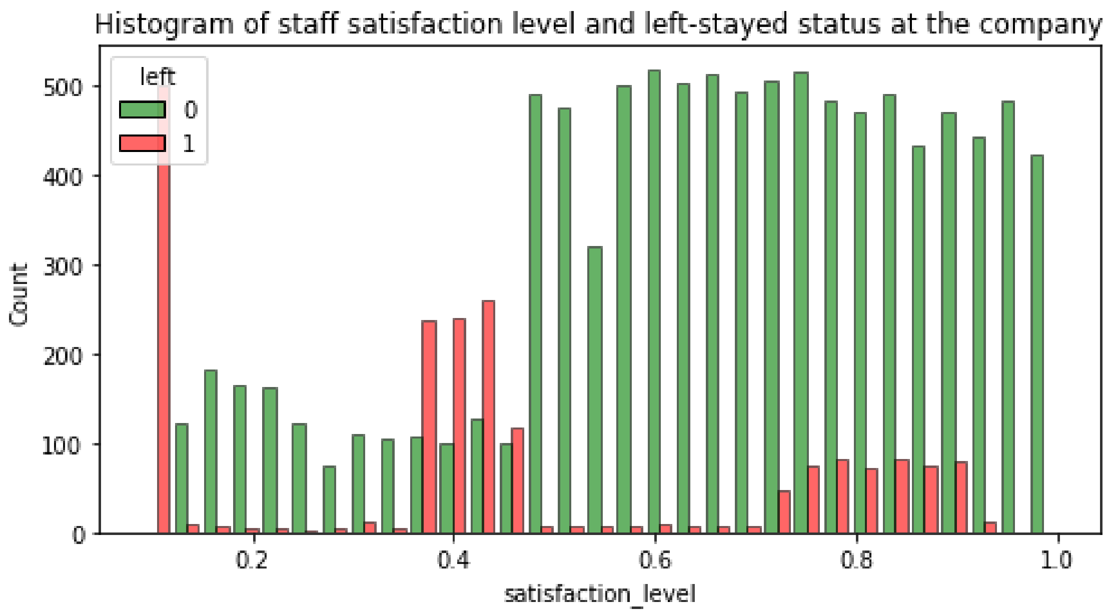


The minority of those who work around (+) or (-) 150 hours on average per month start leaving the company. The number of those who leave or stay fluctuates depending on the average number of hours worked, however those who stay largely exceed those who leave in terms of count, until the numbers of hours worked reach 288 hours and more then no one stays at the company.

[77]:
```
hr2[hr2['avg_monthly_hrs']==288]['left'].value_counts()
```

[77]:
```
True     6
Name: left, dtype: int64
```

[33]:
```
# Create a plot as needed
plt.figure(figsize=(8,4))
sns.histplot(data=hr2,
             x='satisfaction_level',
             hue='left',
             palette={0:'green', 1:'red'},
             multiple='dodge',
             element='bars',
             shrink=0.8,
             alpha=.6)
plt.title('Histogram of staff satisfaction level and left-stayed status at the␣
 ↪company');
```

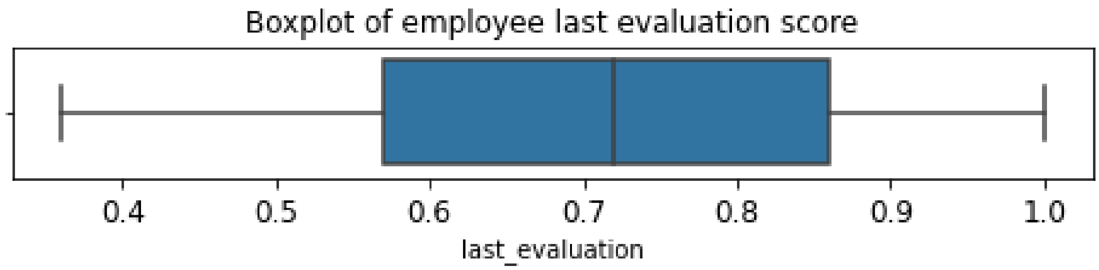


As could be expected, those who score very minimally in terms of satisfaction tend to leave the company in big numbers. The numbers of those who stay then largely exceed the numbers of those who leave when satisfaction level goes up.

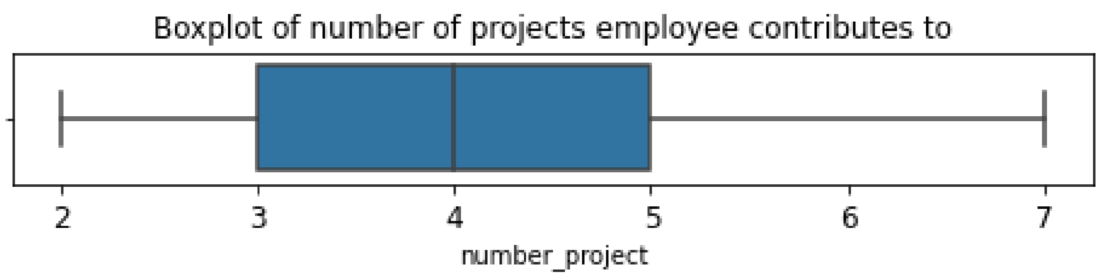
[81]:
```
# Create a plot as needed
plt.figure(figsize=(8,1))
plt.title('Boxplot of employee satisfaction with the company', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
sns.boxplot(x=hr2['satisfaction_level'])
plt.show();
```

[82]:
```
plt.figure(figsize=(8,1))
plt.title('Boxplot of employee last evaluation score', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
```

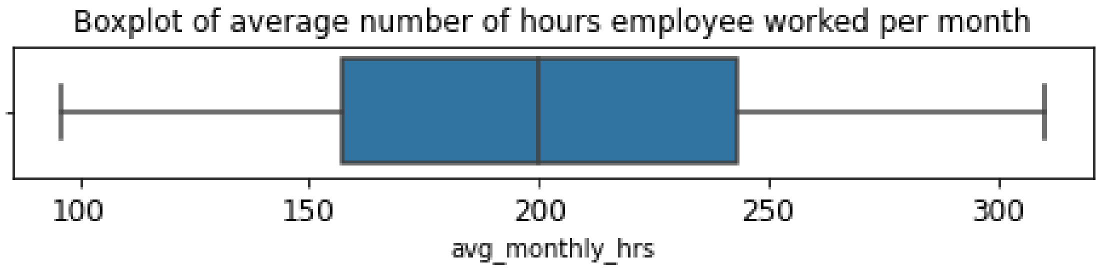
```
sns.boxplot(x=hr2['last_evaluation'])
plt.show();
```



[10]:
```
plt.figure(figsize=(8,1))
plt.title('Boxplot of number of projects employee contributes to', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
sns.boxplot(x=hr2['number_project'])
plt.show();
```



[84]:
```
plt.figure(figsize=(8,1))
plt.title('Boxplot of average number of hours employee worked per month',␣
↪fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
sns.boxplot(x=hr2['avg_monthly_hrs'])
plt.show();
```



3.1.2 Insights

[What insights can you gather from the plots you created to visualize the data?] - There's no any multicollinearity between the features in the data. - Only a minority of those who were not promoted within the last 5 years left the company; about 17%. - The majority of those who leave is from those who get paid the lowest, and vice versa, i.e. the highest the salary, the lowest the turnover rate. There's a direct relationship between salary class and staff turnover. - The highest percentage of those who left the company was among those who work for Sales, Technical, and Support departments. - There are no outliers in the data features, except in 'tenure' which has 824 outliers.

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?

- For both models, all of the independent features were chosen. Furthermore, outliers were dropped from 'tenure' in order for the logistic regression model to function properly, due to its known sensitivity towards outliers.
- For both models, the target variable was 'left'.
- The XGB model was found to be better than the Logistic Regression in terms of the four scores; f1, recall, precision, and accuracy. The Logistic Regression was weaker in performance with the inclusion and exclusion of 'tenure' from the model to test the effect of either on its performance.
- Ethical considerations remain the same throughout this process of model design, test, and evaluation.

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.

It's a classification task where the model will be classifying whether an employee will leave or stay at the company.

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

Both Logistic Regression and XGBoost could be suitable for the task.

4.1.3 Modeling

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

4.2 First: Logistic Regression Modelling

[80]:
```
X = hr2.copy()

# Drop outliers from 'tenure' to avoid affect on the model
X = X[(X['tenure'] >= LL) & (X['tenure'] <= UL)]



# Label encode 'salary'
X['salary'] = X['salary'].replace({'low': 1, 'medium': 2, 'high': 3})

# Dummy encode 'department'
X = pd.get_dummies(X, columns=['department'], drop_first=False)
X = X.reset_index(drop=True)
```

```
X.head()
```

[80]:
```
   satisfaction_level  last_evaluation  number_project  avg_monthly_hrs  \
0                0.38             0.53               2              157
1                0.11             0.88               7              272
2                0.72             0.87               5              223
3                0.37             0.52               2              159
4                0.41             0.50               2              153

   tenure  work_accident  left  promotion_last_5years  salary  stayed  …  \
0       3              0  True                      0       1   False  …
1       4              0  True                      0       2   False  …
2       5              0  True                      0       1   False  …
3       3              0  True                      0       1   False  …
4       3              0  True                      0       1   False  …

   department_IT  department_RandD  department_accounting  department_hr  \
0              0                 0                      0              0
1              0                 0                      0              0
2              0                 0                      0              0
3              0                 0                      0              0
4              0                 0                      0              0

   department_management  department_marketing  department_product_mng  \
0                      0                     0                       0
1                      0                     0                       0
2                      0                     0                       0
3                      0                     0                       0
4                      0                     0                       0

   department_sales  department_support  department_technical
0                 1                   0                     0
1                 1                   0                     0
2                 1                   0                     0
3                 1                   0                     0
4                 1                   0                     0

[5 rows x 21 columns]
```

[20]:
```
# Isolate trarget variable
y = X['left']
y.head()
```

[20]:
```
0    1
1    1
2    1
3    1
```

```
4    1
Name: left, dtype: int64
```

[21]:

```
# Drop 'left' from X
X = X.drop('left', axis=1)
X.head()
```

[21]:

```
   satisfaction_level  last_evaluation  number_project  avg_monthly_hrs  \
0                0.38             0.53               2              157
1                0.11             0.88               7              272
2                0.72             0.87               5              223
3                0.37             0.52               2              159
4                0.41             0.50               2              153

   tenure  work_accident  promotion_last_5years  salary  department_IT  \
0       3              0                      0       1              0
1       4              0                      0       2              0
2       5              0                      0       1              0
3       3              0                      0       1              0
4       3              0                      0       1              0

   department_RandD  department_accounting  department_hr  \
0                 0                      0              0
1                 0                      0              0
2                 0                      0              0
3                 0                      0              0
4                 0                      0              0

   department_management  department_marketing  department_product_mng  \
0                      0                     0                       0
1                      0                     0                       0
2                      0                     0                       0
3                      0                     0                       0
4                      0                     0                       0

   department_sales  department_support  department_technical
0                 1                   0                     0
1                 1                   0                     0
2                 1                   0                     0
3                 1                   0                     0
4                 1                   0                     0
```

[22]:

```
print(X.shape)
print(y.shape)
```

```
(11167, 18)
(11167,)
```

[23]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
↪stratify=y, random_state=0)
```

[25]:
```
# Get shape of each training, validation, and testing set
X_train.shape, X_test.shape, y_test.shape, y_train.shape
```

[25]:
```
((8375, 18), (2792, 18), (2792,), (8375,))
```

[26]:
```
# Construct a logistic regression model and fit it to the training set
from sklearn.linear_model import LogisticRegression
log_clf = LogisticRegression(random_state=0, max_iter=800).fit(X_train, y_train)
```

[27]:
```
# Use the logistic regression model to get predictions on the encoded testing␣
↪set
log_y_pred = log_clf.predict(X_test)
log_y_pred
```

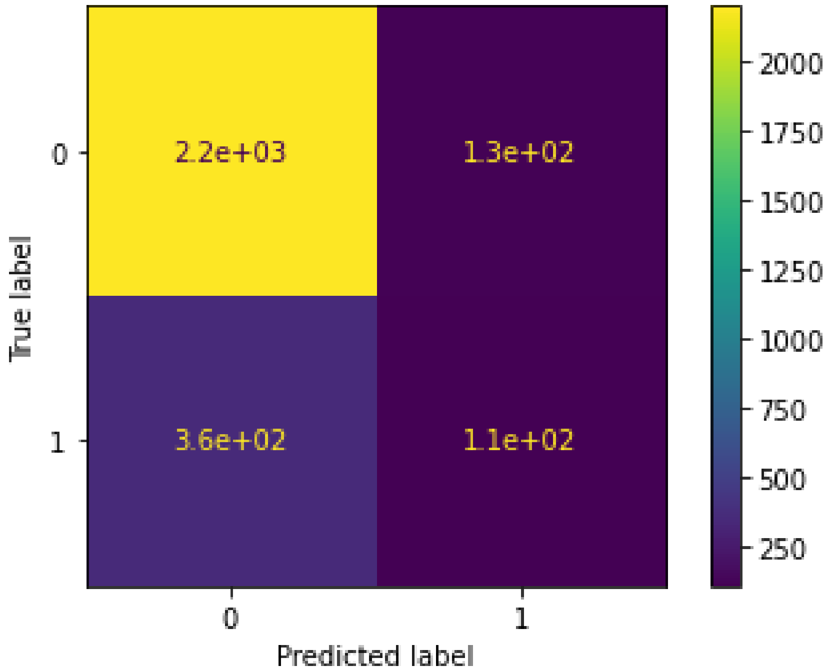
[27]:
```
array([0, 0, 0, …, 0, 0, 0])
```

[28]:
```
# Display the true labels of the testing set
y_test
```

[28]:
```
3568    0
5269    0
9764    0
2587    0
3689    0
       ..
6704    0
4923    0
5228    0
7601    0
569     1
Name: left, Length: 2792, dtype: int64
```

[29]:
```
# Create a confusion matrix to visualize the results of the classification model
# Compute values for confusion matrix
log_cm = confusion_matrix(y_test, log_y_pred, labels=log_clf.classes_)

# Create display of confusion matrix
log_disp = ConfusionMatrixDisplay(confusion_matrix=log_cm,␣
↪display_labels=log_clf.classes_)

# Plot confusion matrix
log_disp.plot()
```

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



[30]:
```
# Create classification report for logistic regression model
target_labels = ['0', '1']
print(classification_report(y_test, log_y_pred, target_names=target_labels))
```

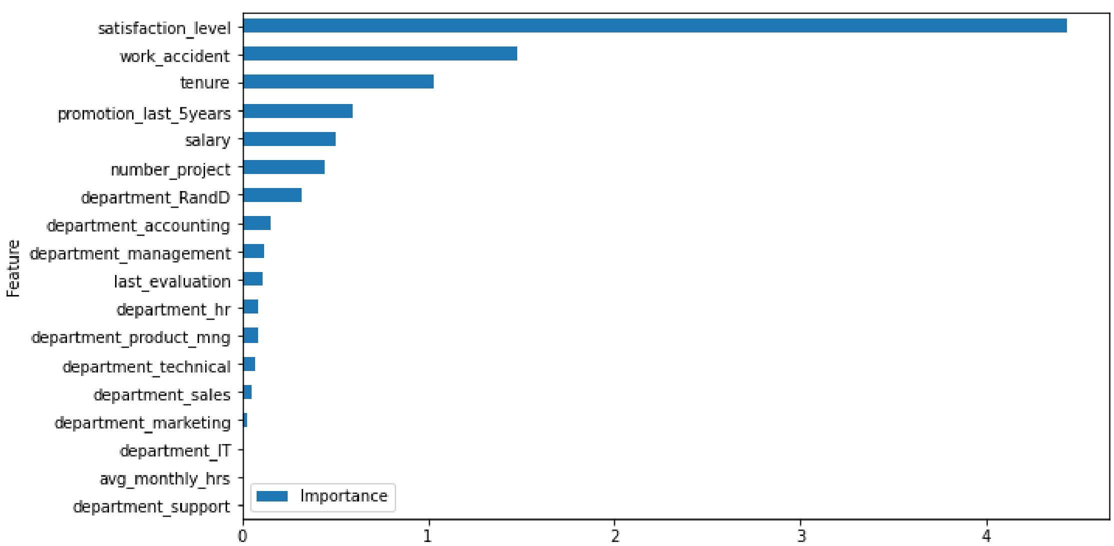
```
              precision    recall  f1-score   support

           0       0.86      0.95      0.90      2321
           1       0.47      0.24      0.32       471

    accuracy                           0.83      2792
   macro avg       0.67      0.59      0.61      2792
weighted avg       0.79      0.83      0.80      2792
```

[33]:
```
from sklearn import metrics
# Get scores for the logistic regression model.

### YOUR CODE HERE ###
log_accuracy = 0.83*100
print('accuracy score:', log_accuracy,'%')
```

```
log_precision = 0.79*100
print('precision score:', log_precision,'%')


log_recall = 0.83*100
print('recall score:', log_recall,'%')


log_f1 = 0.80*100
print('f1 score:', log_f1,'%')
```

```
accuracy score: 83.0 %
precision score: 79.0 %
recall score: 83.0 %
f1 score: 80.0 %
```

[61]:
```
# Plot the relative feature importance of the predictors in the Logistic␣
↪Regression model.

coefficients = log_clf.coef_[0]

feature_importance = pd.DataFrame({'Feature': X.columns, 'Importance': np.
↪abs(coefficients)})
feature_importance = feature_importance.sort_values('Importance',␣
↪ascending=True)
feature_importance.plot(x='Feature', y='Importance', kind='barh', figsize=(10,␣
↪6));
```



4.3 Second: XGBoost model

[36]:
```
y = hr2['left']
y.head()
```

[36]:
```
0    1
1    1
2    1
3    1
4    1
Name: left, dtype: int64
```

[37]:
```
X = hr2.copy()

# Drop unnecessary columns
X = X.drop(['left'], axis=1)


# Label encode 'salary'
X['salary'] = X['salary'].replace({'low': 1, 'medium': 2, 'high': 3})

# Dummy encode 'department'
X = pd.get_dummies(X, columns=['department'], drop_first=False)
X.head()
```

[37]:
```
   satisfaction_level  last_evaluation  number_project  avg_monthly_hrs  \
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

   tenure  work_accident  promotion_last_5years  salary  department_IT  \
0       3              0                      0       1              0
1       6              0                      0       2              0
2       4              0                      0       2              0
3       5              0                      0       1              0
4       3              0                      0       1              0

   department_RandD  department_accounting  department_hr  \
0                 0                      0              0
1                 0                      0              0
2                 0                      0              0
3                 0                      0              0
4                 0                      0              0

   department_management  department_marketing  department_product_mng  \
```

```
0                       0                      0                         0
1                       0                      0                         0
2                       0                      0                         0
3                       0                      0                         0
4                       0                      0                         0

   department_sales  department_support  department_technical
0                 1                   0                     0
1                 1                   0                     0
2                 1                   0                     0
3                 1                   0                     0
4                 1                   0                     0
```

[38]:
```
# Split the data into training and testing sets
X_tr, X_test, y_tr, y_test = train_test_split(X, y, test_size=0.2, stratify=y,␣
↪random_state=0)
```

[39]:
```
# Split the training data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_tr, y_tr, test_size=0.25,␣
↪random_state=0)
```

[40]:
```
# Get shape of each training, validation, and testing set
X_train.shape, X_val.shape, X_test.shape, y_train.shape, y_val.shape, y_test.
↪shape
```

[40]:
```
((7194, 18), (2398, 18), (2399, 18), (7194,), (2398,), (2399,))
```

[41]:
```
# Instantiate the XGBoost classifier
xgb = XGBClassifier(objective='binary:logistic', random_state=0)

# Create a dictionary of hyperparameters to tune
cv_params = {'max_depth': [4,8,12],
             'min_child_weight': [3, 5],
             'learning_rate': [0.01, 0.1],
             'n_estimators': [300, 500]
             }

# Define a dictionary of scoring metrics to capture
scoring = {'accuracy', 'precision', 'recall', 'f1'}

# Instantiate the GridSearchCV object
xgb_cv = GridSearchCV(xgb, cv_params, scoring=scoring, cv=5, refit='f1')
```

[42]:
```
%%time
xgb_cv.fit(X_train, y_train)
```

```
CPU times: user 8min 34s, sys: 4.01 s, total: 8min 38s
```

```
Wall time: 4min 19s
```

[42]:
```
GridSearchCV(cv=5, error_score=nan,
             estimator=XGBClassifier(base_score=None, booster=None,
                                     callbacks=None, colsample_bylevel=None,
                                     colsample_bynode=None,
                                     colsample_bytree=None,
                                     early_stopping_rounds=None,
                                     enable_categorical=False, eval_metric=None,
                                     gamma=None, gpu_id=None, grow_policy=None,
                                     importance_type=None,
                                     interaction_constraints=None,
                                     learning_rate=None, max…
                                     num_parallel_tree=None,
                                     objective='binary:logistic',
                                     predictor=None, random_state=0,
                                     reg_alpha=None, …),
             iid='deprecated', n_jobs=None,
             param_grid={'learning_rate': [0.01, 0.1], 'max_depth': [4, 8, 12],
                         'min_child_weight': [3, 5],
                         'n_estimators': [300, 500]},
             pre_dispatch='2*n_jobs', refit='f1', return_train_score=False,
             scoring={'precision', 'f1', 'recall', 'accuracy'}, verbose=0)
```

[43]:
```
xgb_cv.best_score_
```

[43]:
```
0.9442271713395967
```

[44]:
```
xgb_cv.best_params_
```

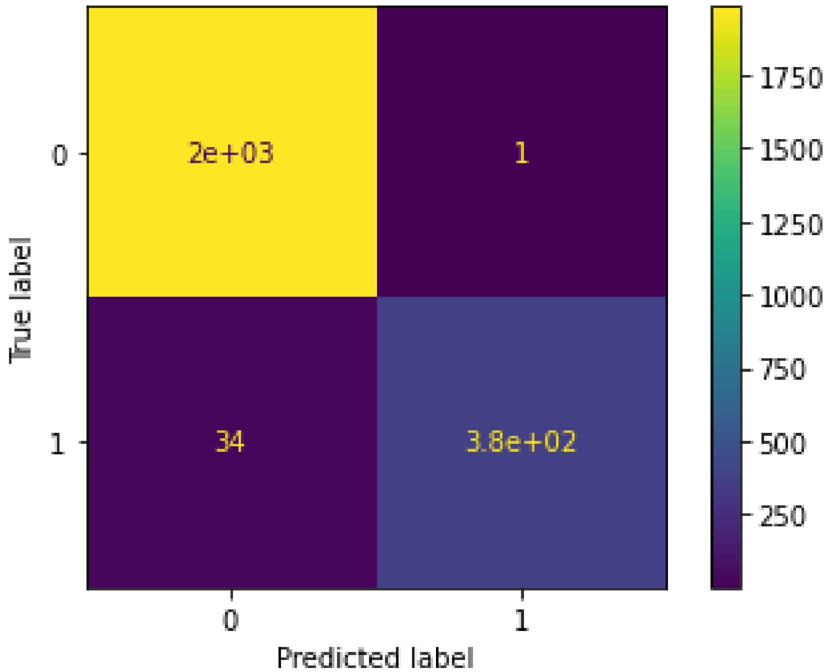
[44]:
```
{'learning_rate': 0.01,
 'max_depth': 8,
 'min_child_weight': 3,
 'n_estimators': 500}
```

[45]:
```
#Evaluate XGBoost model
xgb_y_pred = xgb_cv.best_estimator_.predict(X_val)
xgb_y_pred
```

[45]:
```
array([0, 0, 1, …, 0, 0, 0])
```

[46]:
```
# Compute values for confusion matrix
xgb_cm = confusion_matrix(y_val, xgb_y_pred)

# Create display of confusion matrix
xgb_disp = ConfusionMatrixDisplay(confusion_matrix=xgb_cm,␣
 ↪display_labels=xgb_cv.classes_)
```

```
# Plot confusion matrix
xgb_disp.plot()

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



[47]:
```
# Create a classification report
target_labels = ['0', '1']
print(classification_report(y_val, xgb_y_pred, target_names=target_labels))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1984
           1       1.00      0.92      0.96       414

    accuracy                           0.99      2398
   macro avg       0.99      0.96      0.97      2398
weighted avg       0.99      0.99      0.99      2398
```

[48]:
```
xgb_accuracy = 0.99*100
print('accuracy score:', xgb_accuracy,'%')
```

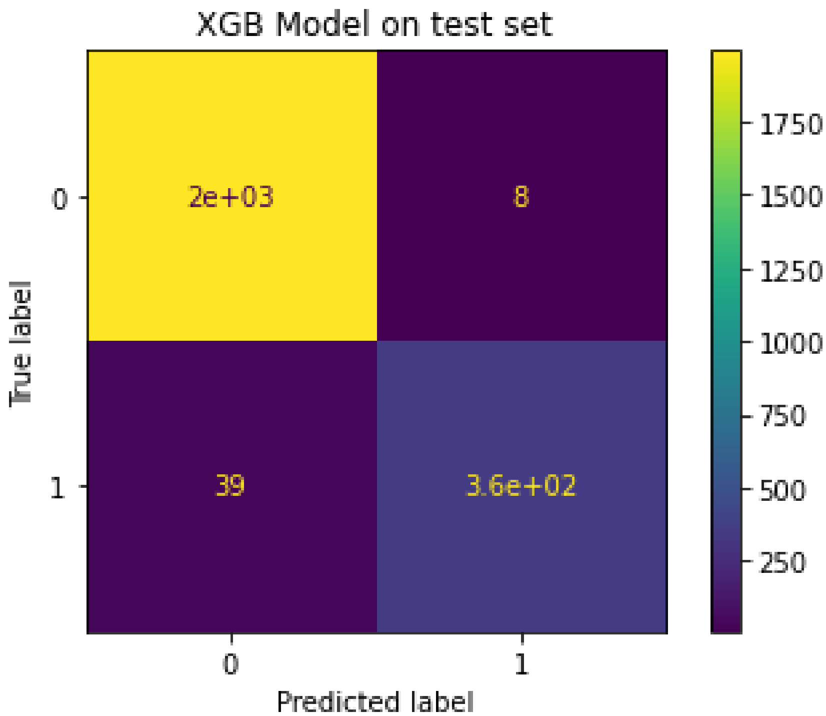
```
xgb_precision = 0.99*100
print('precision score:', xgb_precision,'%')


xgb_recall = 0.99*100
print('recall score:', xgb_recall,'%')


xgb_f1 = 0.99*100
print('f1 score:', xgb_f1,'%')
```

```
accuracy score: 99.0 %
precision score: 99.0 %
recall score: 99.0 %
f1 score: 99.0 %
```

4.4 Third: Use champion model to predict on test data

[49]:
```
# Use champion model to predict on test data
y_pred = xgb_cv.best_estimator_.predict(X_test)
```

[50]:
```
# Compute values for confusion matrix
cn_cm = confusion_matrix(y_test, y_pred)

# Create display of confusion matrix
cn_disp = ConfusionMatrixDisplay(confusion_matrix=cn_cm, display_labels=xgb_cv.
 ↪classes_)

# Plot confusion matrix
cn_disp.plot()

# Display plot
plt.title('XGB Model on test set');
plt.show()
```



[51]:
```
# Create a classification report
target_labels = ['0', '1']
print(classification_report(y_test, y_pred, target_names=target_labels))
```

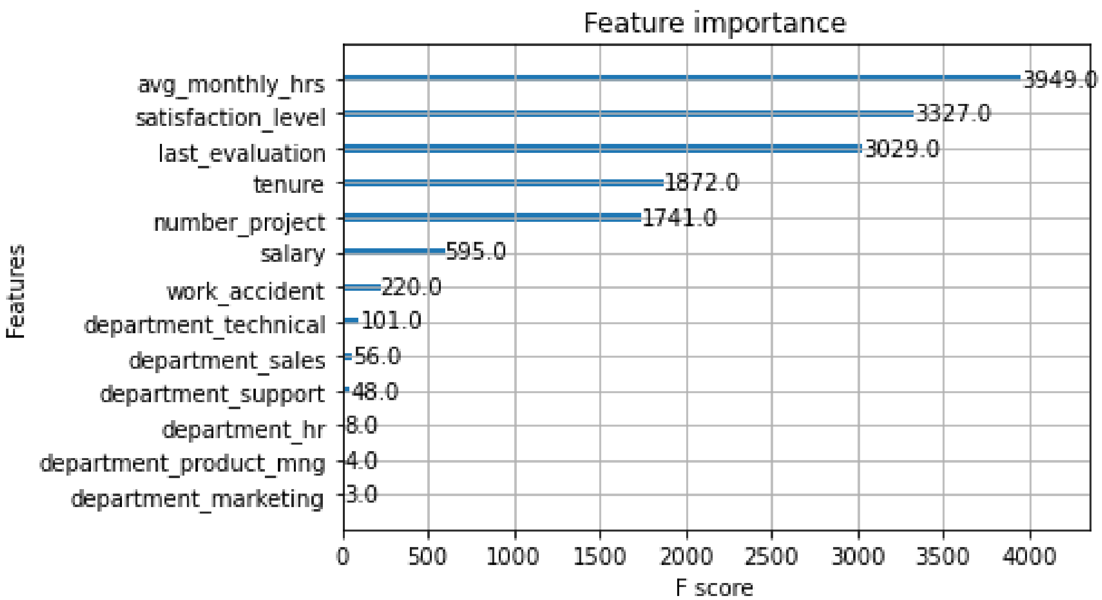
```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      2001
           1       0.98      0.90      0.94       398

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

[52]:
```
cn_accuracy = 0.98*100
print('accuracy score:', cn_accuracy,'%')


cn_precision = 0.98*100
print('precision score:', cn_precision,'%')


cn_recall = 0.98*100
print('recall score:', cn_recall,'%')
```

```
cn_f1 = 0.98*100
print('f1 score:', cn_f1,'%')
```

```
accuracy score: 98.0 %
precision score: 98.0 %
recall score: 98.0 %
f1 score: 98.0 %
```

[53]:
```
# Plot the relative feature importance of the predictor variables in the␣
↪champion model.

plot_importance(xgb_cv.best_estimator_);
```



[54]:
```
# Create a table of results to compare model performance.

table = pd.DataFrame()
table = table.append({'Model': "Logistic Regression",
                        'F1':  log_f1,
                        'Recall': log_recall,
                        'Precision': log_precision,
                        'Accuracy': log_accuracy
                      },
                        ignore_index=True
                    )
```

```
table = table.append({'Model': "XGBoost",
                      'F1':  xgb_f1,
                      'Recall': xgb_recall,
                      'Precision': xgb_precision,
                      'Accuracy': xgb_accuracy
                    },
                      ignore_index=True
                  )

table = table.append({'Model': "Champion (XGB)",
                      'F1':  cn_f1,
                      'Recall': cn_recall,
                      'Precision': cn_precision,
                      'Accuracy': cn_accuracy
                    },
                      ignore_index=True
                  )

table
```

[54]:

	Model	F1	Recall	Precision	Accuracy
0	Logistic Regression	80.0	83.0	79.0	83.0
1	XGBoost	99.0	99.0	99.0	99.0
2	Champion (XGB)	98.0	98.0	98.0	98.0

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?

5.1 Step 4. Results and Evaluation

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

5.1.1 Summary of model results

5.1.2 Conclusion, Recommendations, Next Steps

5.1.3 Conclusions and model results:

- Two models were designed, Logistic Regression and XGBoost. The first performed very well as measured by the four scores of precision, recall, F1, and accuracy. The lowest score it achieved in all of them was in Precision which was 79%. The second model performed outstandingly good where in each of the four scores it scored no less than 99%.
- Accordingly, the winning model was tested on the test data and it attained 98% in each of the four scores. Further emphasising the very good results achieved earlier by the same model (XGB) when it was validated.
- In terms of FP and FN classification, i.e. staff members who were misclassified by the models that will leave or stay at the company, again the XGB was found to be the best. Compared to the Logistic Regression which attained 130 FPs and 360 FNs, the XGB generated only 1 FP and 34 FNs, and when finally tested, it got only 8 FPs and 39 FNs.
- Most important predictors for the XGB model that were found to have the strongest ability to predict whether a staff member will leave or stay were average monthly hours, satisfaction level, and last evaluation. It is therefore recommended to further study these strongest 3 predictors and see how modifying or improving them could lead to a lower staff turnover and increase staff retention.

5.1.4 Recommendations

In order to increase staff retention and decrease turnover, the following procedures are recommended for consideration by the comapny's top management:

- Limit the number of projects engaged by an employee to 4 projects maximum.
- Revise promotion terms and scheme in order to guarantee timely promotion for those who earn it.
- Limit the number of hours worked by an employee per month to less than 288 hours.
- Review the current scale of wages and salaries at the company with the aim of increasing them.

- Introduce measures to make work at certain departments at the company, such as Sales, Technical, and Support more satisfactory and encouraging.

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.