Salifort Motors Data Analytics Lab

August 31, 2024

1 Capstone project: Providing data-driven suggestions for HR

1.1 Description and deliverables

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

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

2.2 Step 1. Imports

- Import packages
- Load dataset

2.2.1 Import packages

[5]:
```
# Import packages

# For data manipulation
import numpy as np
import pandas as pd

# For data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# For displaying all of the columns in dataframes
pd.set_option('display.max_columns', None)

# For data modeling
from xgboost import XGBClassifier
from xgboost import XGBRegressor
from xgboost import plot_importance

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# For metrics and helpful functions
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,\
f1_score, confusion_matrix, ConfusionMatrixDisplay, classification_report
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.tree import plot_tree

# For saving models
import pickle
```

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.

[6]:
```
# Load dataset into a dataframe
df0 = pd.read_csv("HR_capstone_dataset.csv")

# Display first few rows of the dataframe
df0.head(5)
```

[6]:
```
   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

[7]:
```
# Gather basic information about the data
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
```

2.3.2 Gather descriptive statistics about the data

[8]:
```
# Gather descriptive statistics about the data
df0.describe()
```

[8]:

	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.

[9]:
```
# Display all column names
df0.columns
```

[9]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
       'promotion_last_5years', 'Department', 'salary'],
      dtype='object')
```

[10]:
```
# Standardize columns so they are all in snake case, lower case, and making␣
↪names more concise if needed
df0 = df0.rename(columns={'Work_accident': 'work_accident',␣
↪'average_montly_hours': 'average_monthly_hours',
                          'time_spend_company': 'tenure', 'Department':␣
↪'department'})

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

[10]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_monthly_hours', '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.

[11]:
```
# This reveals 0 missing values
df0.isna().sum()
```

[11]:
```
satisfaction_level       0
last_evaluation          0
number_project           0
average_monthly_hours    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.

[12]:
```
```

```
# This reaveals 3,008 rows contain duplicates, we will need to inspect this␣
 ↪further
df0.duplicated().sum()
```

[12]:
```
3008
```

[13]:
```
# we will need to proove if these are truly duplicates or if they are employees␣
 ↪just reporting the same results
df0[df0.duplicated()].head()
```

[13]:

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	tenure	work_accident	left	promotion_last_5years	department	salary
396	0.46	0.57	2	139	3	0	1	0	sales	low
866	0.41	0.46	2	128	3	0	1	0	accounting	low
1317	0.37	0.51	2	127	3	0	1	0	sales	medium
1368	0.41	0.52	2	132	3	0	1	0	RandD	low
1461	0.42	0.53	2	142	3	0	1	0	sales	low

[14]:
```
# Drop duplicates and save resulting dataframe in a new variable as needed
df1 = df0.drop_duplicates(keep='first')

# Display first few rows of new dataframe as needed
df1.head()
```

[14]:

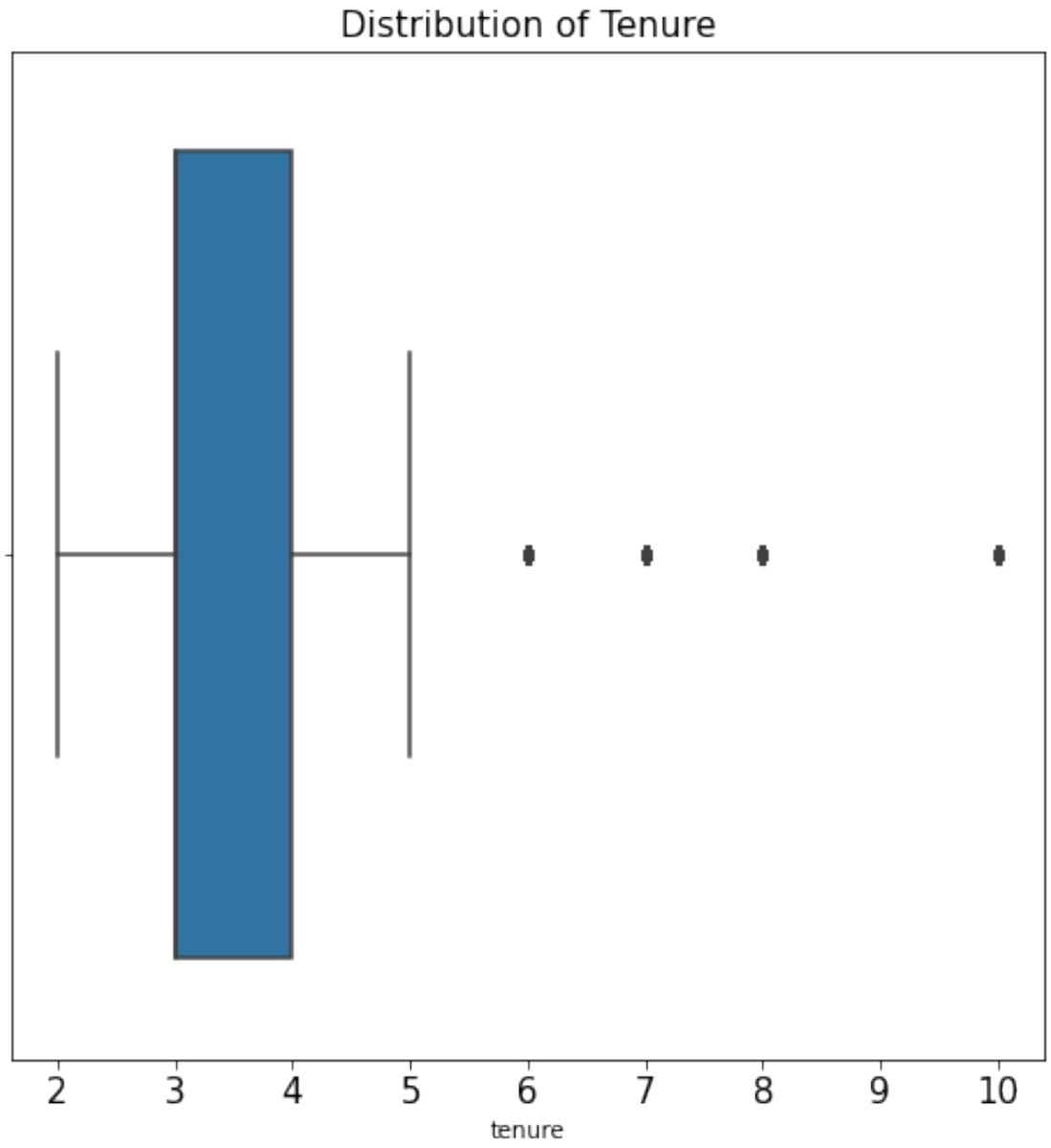
	satisfaction_level	last_evaluation	number_project	average_monthly_hours	tenure	work_accident	left	promotion_last_5years	department	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low
4	0.37	0.52	2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

```
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.

[15]:
```
# Create a boxplot to visualize distribution of `tenure` and detect any outliers
plt.figure(figsize=(8,8))
plt.title('Distribution of Tenure',fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
sns.boxplot(x=df1['tenure'])
plt.show()
```



[16]:
```
# Determine the number of rows containing outliers

# Finding the interquartile range and the upper and lower limits
percentile1 = df1['tenure'].quantile(0.25)
percentile3 = df1['tenure'].quantile(0.75)
iqr = percentile3 - percentile1
upper_limit = percentile3 + 1.5 * iqr
lower_limit = percentile1 + 1.5 * iqr
print("IQR:", iqr)
print("Lower limit:", lower_limit)
```

```
print("Upper limit:", upper_limit)

# Finding the subset of data in Tenure containing outliers
outliers = df1[(df1['tenure'] > upper_limit) | (df1['tenure'] < lower_limit)]
print("Number of rows containing outliers in Tenure:", len(outliers))
```

```
IQR: 1.0
Lower limit: 4.5
Upper limit: 5.5
Number of rows containing outliers in Tenure: 10929
```

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?

3.1 Step 2. Data Exploration (Continue EDA)

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

[17]:
```
# Get numbers of people who left vs. stayed
print(df1['left'].value_counts())
print()

# Get percentages of people who left vs. stayed
print(df1['left'].value_counts(normalize=True))
```

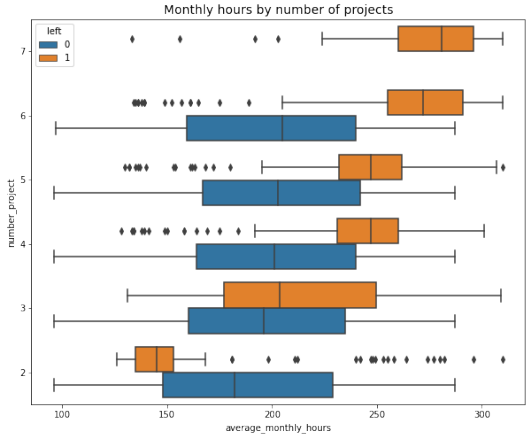
```
0    10000
1     1991
Name: left, dtype: int64

0    0.833959
```

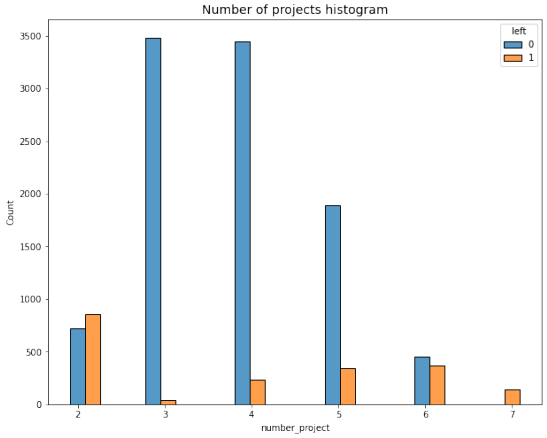
```
1    0.166041
Name: left, dtype: float64
```

3.1.1 Data visualizations

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

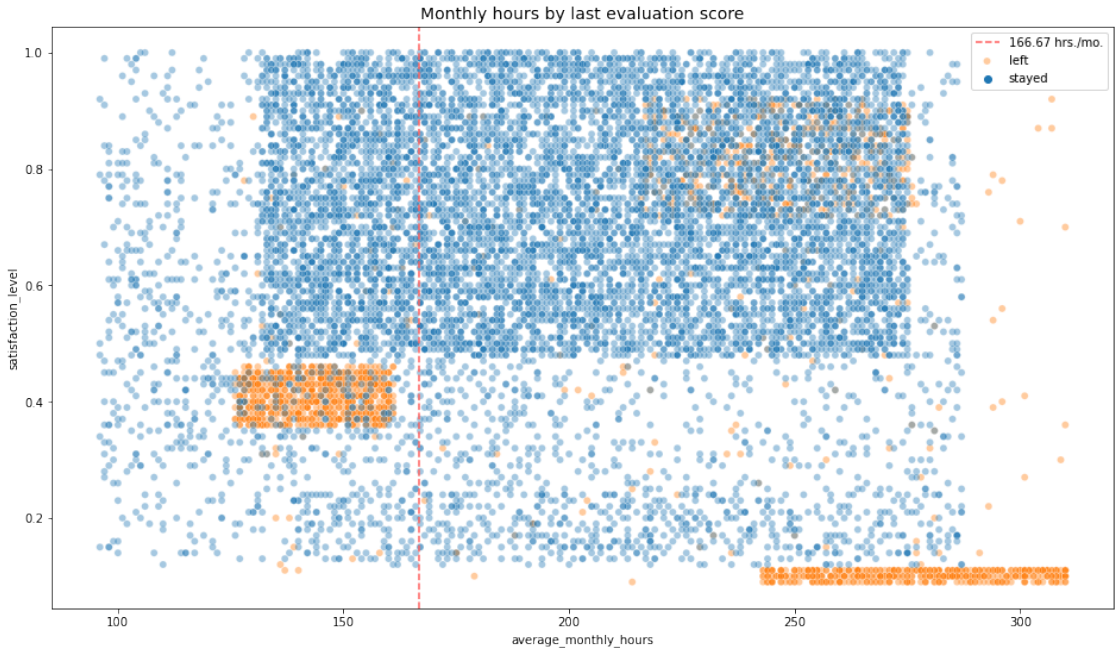
[19]:
```
# Using a stacked box comparing employees who left and those who stayed with how many hours they worked and their number of projects

# Takeaways: Two major groups who left the company were people who worked considerly less than their peers and those that worked much more.
# The largest number of employees left the company with only two projects and every employee left the company with seven projects.
# The least number of employees left the company with three and four projects

# Set figure and axes
fig, ax = plt.subplots(1, 2, figsize = (22,8))

# Creating stacked oxplot
sns.boxplot(data=df1, x='average_monthly_hours', y='number_project', hue='left', orient="h", ax=ax[0])
ax[0].invert_yaxis()
ax[0].set_title('Monthly hours by number of projects', fontsize='14')

# Creating histogram to represent the sample size
tenure_stay = df1[df1['left']==0]['number_project']
tenure_left = df1[df1['left']==1]['number_project']
sns.histplot(data=df1, x='number_project', hue='left', multiple='dodge', shrink=2, ax=ax[1])
ax[1].set_title('Number of projects histogram', fontsize='14')

plt.show()

# This confirms 145 employees with seven projects left the company
df1[df1['number_project']==7]['left'].value_counts()
```





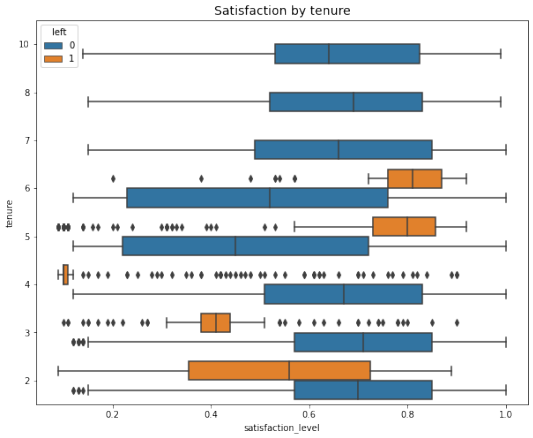
[19]:
```
1    145
Name: left, dtype: int64
```

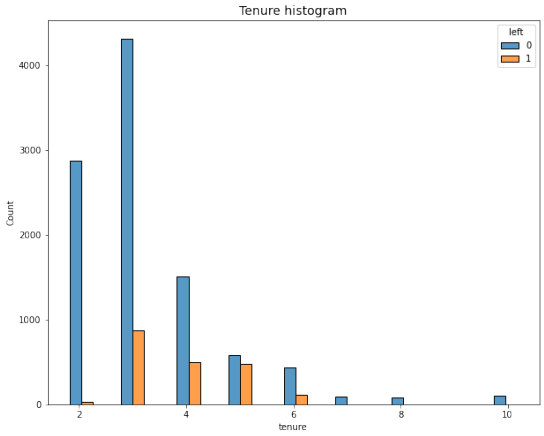
[20]:
```
# Using a scatter plot to compare average monthly hours and satisfaction level

# Takeaways: There is a large group of employees who worked 250-300 hours and␣
 ↪left.
# This amount of hours would mean employees are working 60-70 hours per week␣
 ↪which would explain the low satisfaction scores.
# However, there are also employees that work a large amount of hours per week␣
 ↪and have a high satisfaction score.
# Because of the strange shape of the distribution, we will need to do further␣
 ↪analysis on this relationship.

# Creating Scatter Plot
plt.figure(figsize=(16, 9))
sns.scatterplot(data=df1, x='average_monthly_hours', y='satisfaction_level',␣
 ↪hue='left', alpha=0.4)
plt.axvline(x=166.67, color='#ff6361', label='166.67 hrs./mo.', ls='--')
plt.legend(labels=['166.67 hrs./mo.', 'left', 'stayed'])
plt.title('Monthly hours by last evaluation score', fontsize='14');
```



[21]:
```
# Using a Stacked box plot and a histogram to explore the relationship between tenure and satisfaction level

# Takeaways: Employees who left and were dissatified had short tenures typically lasting 3 or 4 years.
# Employees who left and were satisfied had longer tenures typically lasting 5 or 6 years.
# The employees with tenures longer than 6 years were all employees who have stayed and there are not many with the company currently.

# Set figure and axes
fig, ax = plt.subplots(1, 2, figsize = (22,8))

# Creating boxplot
sns.boxplot(data=df1, x='satisfaction_level', y='tenure', hue='left', orient="h", ax=ax[0])
ax[0].invert_yaxis()
ax[0].set_title('Satisfaction by tenure', fontsize='14')

# Creating histogram
tenure_stay = df1[df1['left']==0]['tenure']
tenure_left = df1[df1['left']==1]['tenure']
sns.histplot(data=df1, x='tenure', hue='left', multiple='dodge', shrink=5, ax=ax[1])
ax[1].set_title('Tenure histogram', fontsize='14')
```

```
plt.show();

# This shows the satisfaction levels of those that have stayed and those that␣
 ↪have left
df1.groupby(['left'])['satisfaction_level'].agg([np.mean,np.median])
```



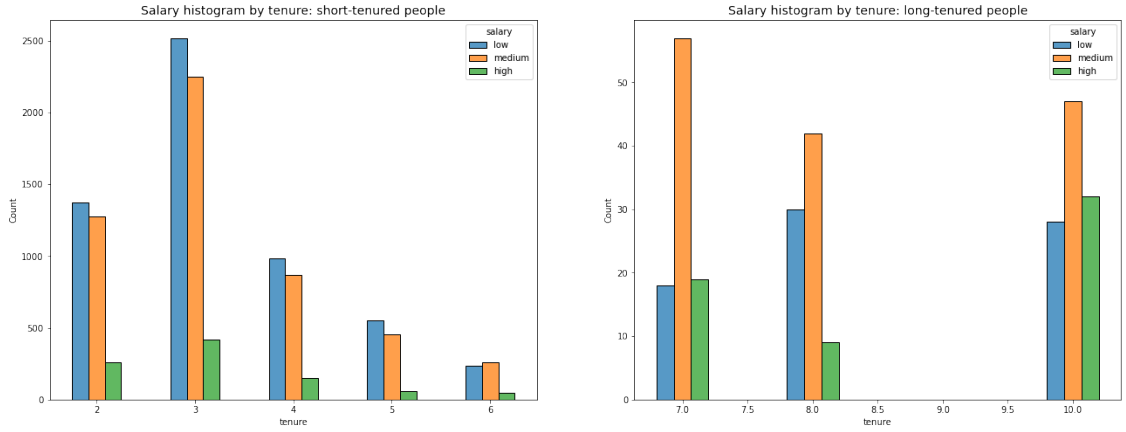


[21]:

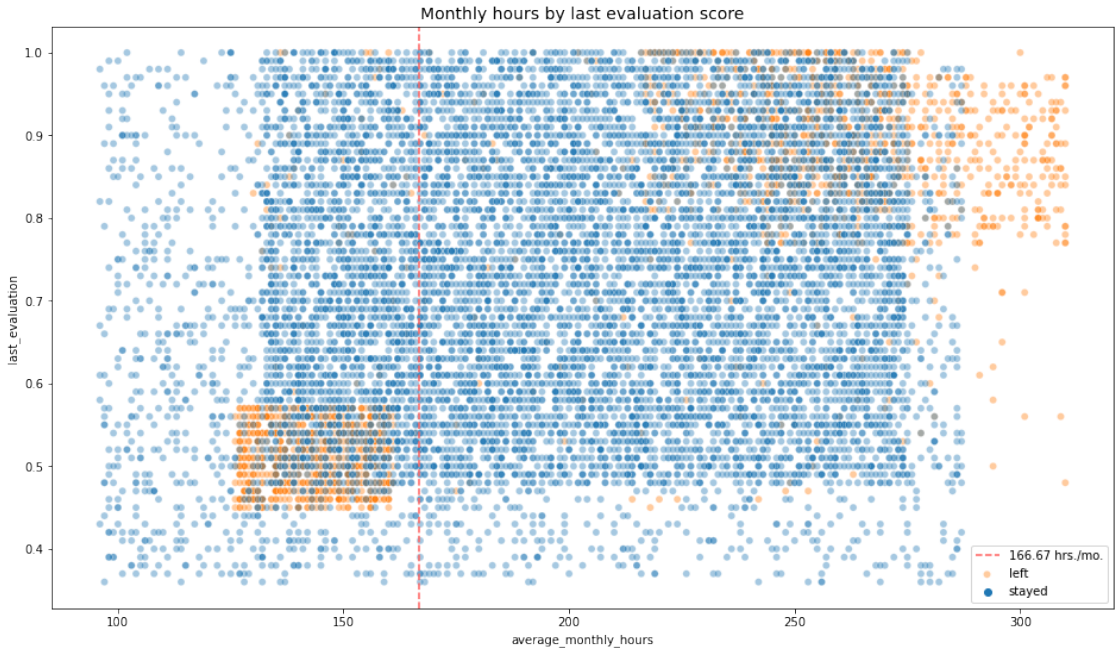
left	mean	median
0	0.667365	0.69
1	0.440271	0.41

[22]:
```
# Creating two histrograms to visualize the tenure of employees compared with␣
 ↪their salary

# Takeaways: This shows that while there was a larger count of lower salaried␣
 ↪people with shorter tenures,
# there was also a large count of medium salaried people with shorter tenures.
# Also, the longer tenured employes had a similar amount of low, medium, and␣
 ↪high salaried people.

# Set figure and axes
fig, ax = plt.subplots(1, 2, figsize = (22,8))

# Define short-tenured employees
tenure_short = df1[df1['tenure'] < 7]

# Define long-tenured employees
tenure_long = df1[df1['tenure'] > 6]

# Ploting short-tenured histogram
```

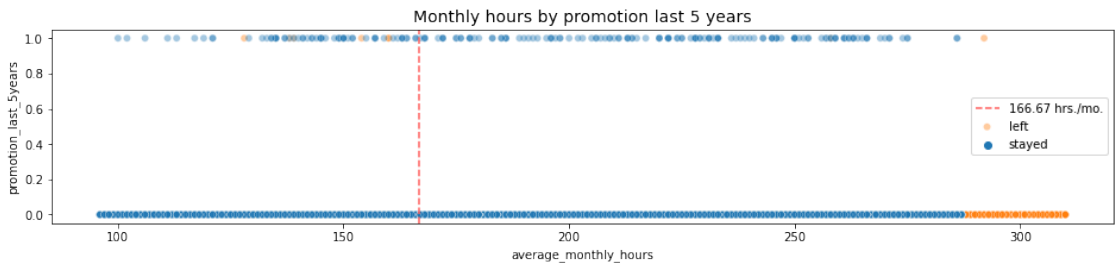
```
sns.histplot(data=tenure_short, x='tenure', hue='salary', discrete=1,
             hue_order=['low', 'medium', 'high'], multiple='dodge', shrink=.5,␣
 ↪ax=ax[0])
ax[0].set_title('Salary histogram by tenure: short-tenured people',␣
 ↪fontsize='14')

# Ploting long-tenured histogram
sns.histplot(data=tenure_long, x='tenure', hue='salary', discrete=1,
             hue_order=['low', 'medium', 'high'], multiple='dodge', shrink=.4,␣
 ↪ax=ax[1])
ax[1].set_title('Salary histogram by tenure: long-tenured people',␣
 ↪fontsize='14');
```



[23]:
```
# Creating a scatterplot to see the relationship between average monthly hours␣
 ↪and last evaluation

# Takeaways: There are more employees in the upper right quadrant than the␣
 ↪upper left.
# Therefore, there seems to be correlation between hours worked and evaluation␣
 ↪scores.
# However, there are also a large amount of employees in the lower right␣
 ↪quadrant.
# Therefore, working long hours does not guarentee a high evaluation score

# Creating scatterplot
plt.figure(figsize=(16, 9))
sns.scatterplot(data=df1, x='average_monthly_hours', y='last_evaluation',␣
 ↪hue='left', alpha=0.4)
plt.axvline(x=166.67, color='#ff6361', label='166.67 hrs./mo.', ls='--')
plt.legend(labels=['166.67 hrs./mo.', 'left', 'stayed'])
plt.title('Monthly hours by last evaluation score', fontsize='14');
```

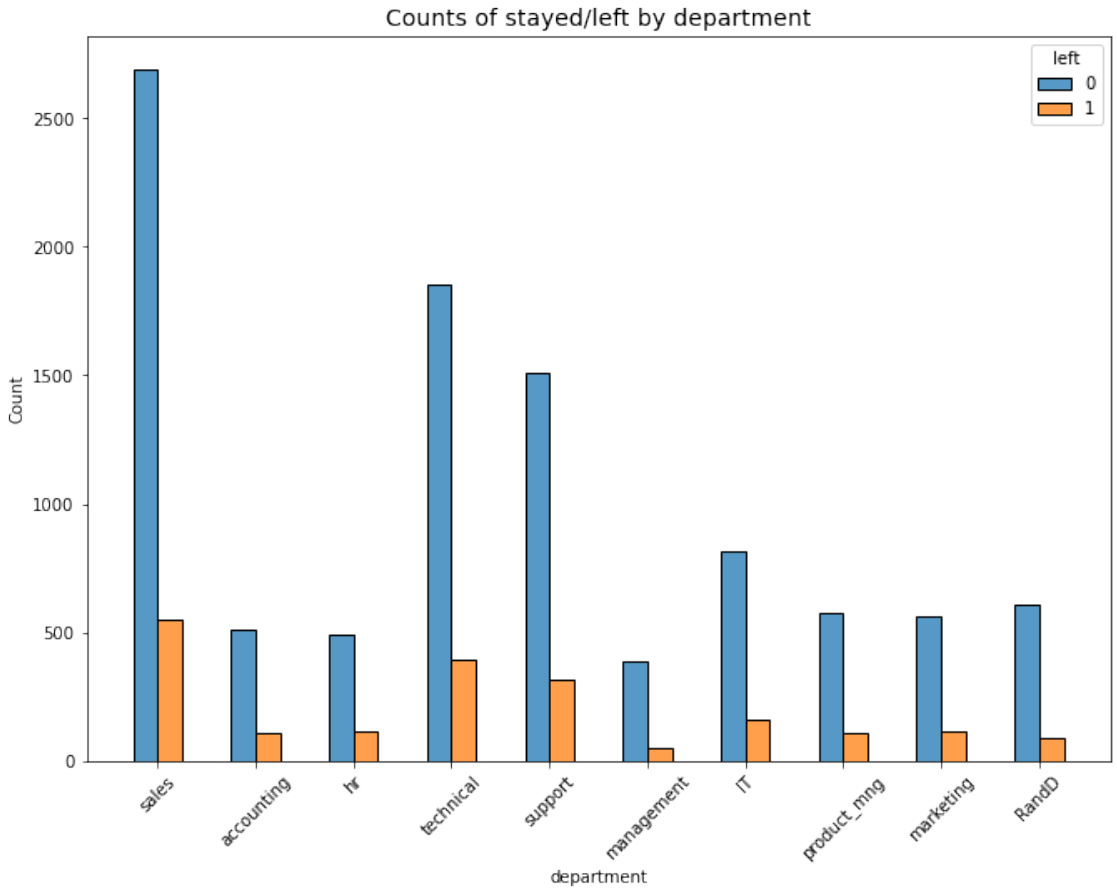


[26]:
```
# Creating a plot to see the likely hood that employees who worked a high␣
 ↪number of hours were promoted
# in the last 5 years

# Takeaways: Very few employees who were promoted in the last 5 years left␣
 ↪while all the employees who left were the ones working the largest number of␣
 ↪hours.
# Also, very few employees were worked the most hours were promoted.

# Creating plot
plt.figure(figsize=(16, 3))
sns.scatterplot(data=df1, x='average_monthly_hours', y='promotion_last_5years',␣
 ↪hue='left', alpha=0.4)
plt.axvline(x=166.67, color='#ff6361', ls='--')
plt.legend(labels=['166.67 hrs./mo.', 'left', 'stayed'])
plt.title('Monthly hours by promotion last 5 years', fontsize='14');
```

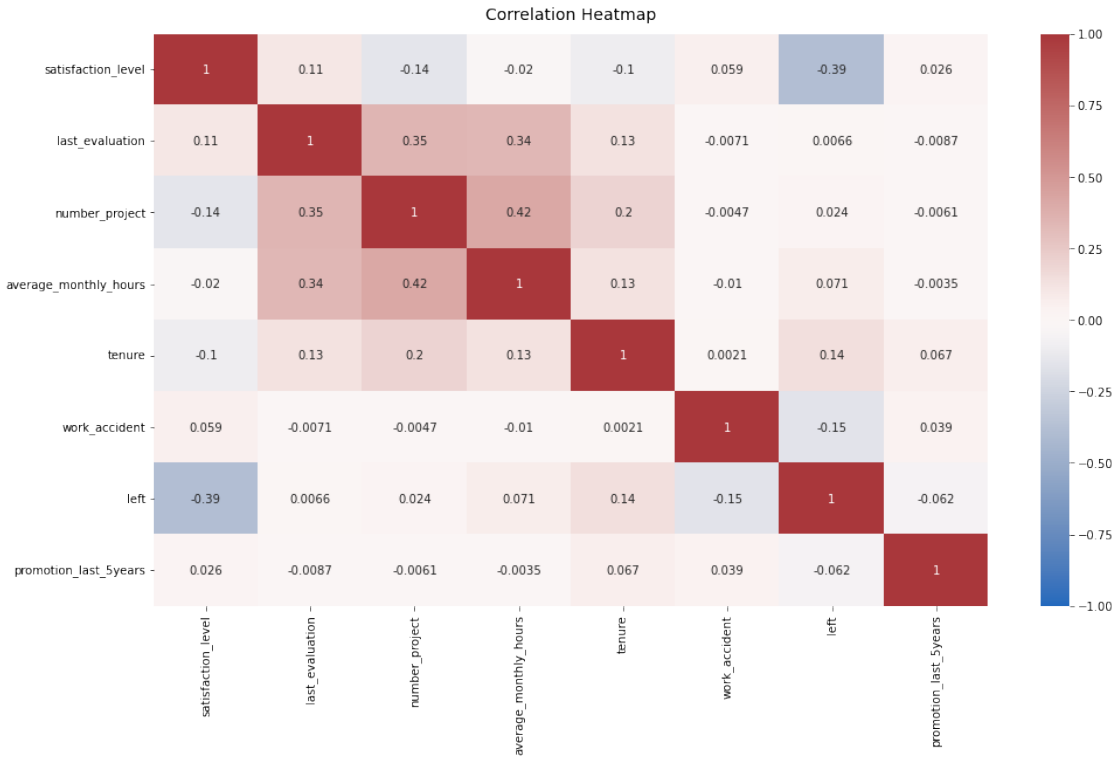


[29]:
```
# Creating a stacked histogram to see how the employees who left are␣
 ↪distributed across departments

# Takeaway: There does not seem to be any one department that differs␣
 ↪significantly in its proportion of employees who left to those who stayed

# Display counts for each department
print(df1["department"].value_counts())

# Creating stacked histogram
plt.figure(figsize=(11,8))
sns.histplot(data=df1, x='department', hue='left', discrete=1,
             hue_order=[0, 1], multiple='dodge', shrink=.5)
plt.xticks(rotation='45')
plt.title('Counts of stayed/left by department', fontsize=14);
```

```
sales          3239
technical      2244
support        1821
IT              976
RandD           694
product_mng     686
marketing       673
accounting      621
hr              601
management      436
Name: department, dtype: int64
```



[30]:
```
# Creating a heatmap to see other connections between variables in the data

# Takeaways: Some of the strongest positive relationships are number of␣
 ↪projects and last evaluation, average monthly hours and last evaluation, and␣
 ↪number of projects and average
# monthly hours.
# Some of the strongest negative relationships are satisfaction level and␣
 ↪leaving, work accident and leaving, and number of projects and satisfaction␣
 ↪level.

# Ploting a correlation heatmap
plt.figure(figsize=(16, 9))
heatmap = sns.heatmap(df0.corr(), vmin=-1, vmax=1, annot=True, cmap=sns.
 ↪color_palette("vlag", as_cmap=True))
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':14}, pad=12);
```



3.1.2 Insights

3.1.3 It seems alot of the people leaving are due to poor management. People are working to many hours and are being assigned to many projects at a time resulting in low satisfaction scores.

3.1.4 People working large numbers of hours for many years are also unlikely to recieve promotions further increasing their feelings of burn out.

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?

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.

4.1.2 Our goal is to predict whether an employee will leave the company, which is a categorical outcome variable. Therefore, since an employee can either stay or leave, the task involves binary classification.

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

4.1.4 Since the variable we want to predict is categorical, we can either create a Logical Regression model, or a Tree-based Machine Learning model.

4.1.5 Modeling

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

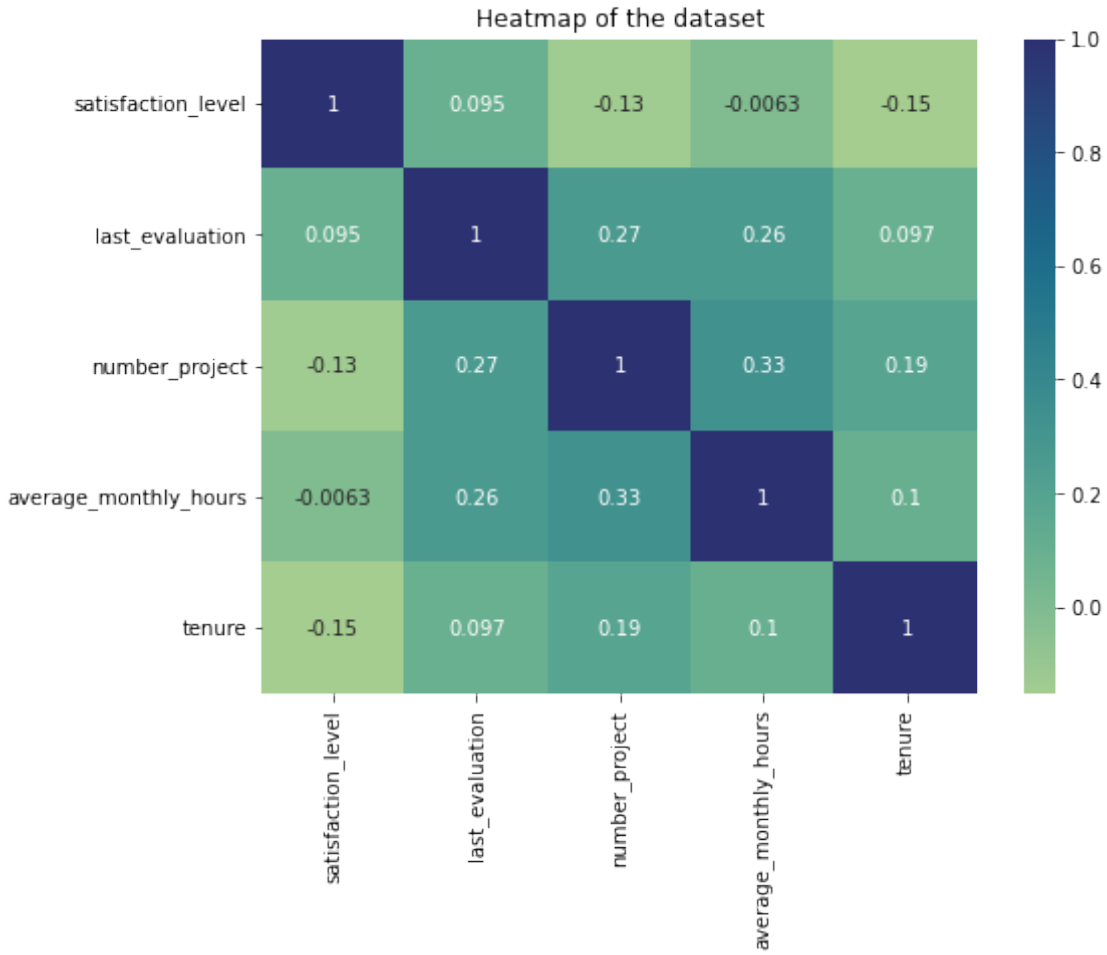
[42]:
```
# Creating a Logistic Regression Model to predict if an employee will leave the␣
 ↪company

# Copy the dataframe
df_enc = df1.copy()

# Encode the `salary` column as an ordinal numeric category
df_enc['salary'] = (
    df_enc['salary'].astype('category')
    .cat.set_categories(['low', 'medium', 'high'])
    .cat.codes
)

# Dummy encode the `department` column
df_enc = pd.get_dummies(df_enc, drop_first=False)
```

```
# Display the new dataframe
df_enc.head()
```

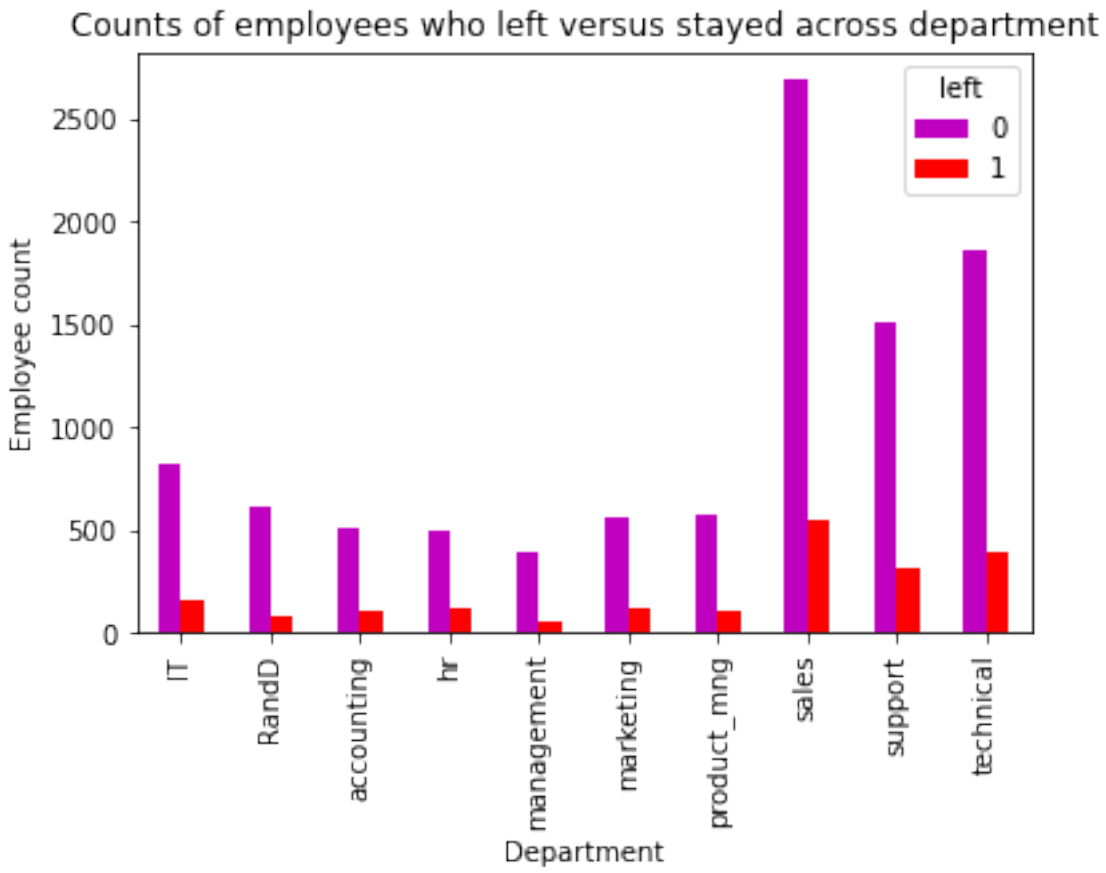
[42]:

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	tenure	work_accident	left	promotion_last_5years	salary	department_IT	department_RandD	department_accounting	department_hr	department_management	department_marketing	department_product_mng	department_sales	department_support	department_technical
0	0.38	0.53	2	157	3	0	1	0	0	0	0	0	0	0	0	0	1	0	0
1	0.80	0.86	5	262	6	0	1	0	1	0	0	0	0	0	0	0	1	0	0
2	0.11	0.88	7	272	4	0	1	0	1	0	0	0	0	0	0	0	1	0	0
3	0.72	0.87	5	223	5	0	1	0	0	0	0	0	0	0	0	0	1	0	0
4	0.37	0.52	2	159	3	0	1	0	0	0	0	0	0	0	0	0	1	0	0

[41]:
```
# Creating a heatmap to show the correlations between variables

plt.figure(figsize=(8, 6))
sns.heatmap(df_enc[['satisfaction_level', 'last_evaluation', 'number_project',␣
↪'average_monthly_hours', 'tenure']]
            .corr(), annot=True, cmap="crest")
plt.title('Heatmap of the dataset')
plt.show()
```



[43]:
```
# Creating a stacked bart plot to visualize number of employees across␣
 ↪department, comparing those who left with those who didn't

pd.crosstab(df1['department'], df1['left']).plot(kind ='bar',color='mr')
plt.title('Counts of employees who left versus stayed across department')
plt.ylabel('Employee count')
plt.xlabel('Department')
plt.show()
```



[44]:
```
# Since Logistic Regression is sensitive to outliters, we will remove them in␣
 ↪the tenure column

# Select rows without outliers in `tenure` and save resulting dataframe in a␣
 ↪new variable
df_logreg = df_enc[(df_enc['tenure'] >= lower_limit) & (df_enc['tenure'] <=␣
 ↪upper_limit)]

# Display first few rows of new dataframe
df_logreg.head()
```

[44]:

	satisfaction_level	last_evaluation	number_project	\
3	0.72	0.87	5	
7	0.92	0.85	5	
8	0.89	1.00	5	
12	0.84	0.92	4	
19	0.76	0.89	5	

	average_monthly_hours	tenure	work_accident	left	promotion_last_5years	\

```
3                    223       5              0     1                    0
7                    259       5              0     1                    0
8                    224       5              0     1                    0
12                   234       5              0     1                    0
19                   262       5              0     1                    0

    salary  department_IT  department_RandD  department_accounting  \
3        0              0                 0                      0
7        0              0                 0                      0
8        0              0                 0                      0
12       0              0                 0                      0
19       0              0                 0                      0

    department_hr  department_management  department_marketing  \
3               0                      0                     0
7               0                      0                     0
8               0                      0                     0
12              0                      0                     0
19              0                      0                     0

    department_product_mng  department_sales  department_support  \
3                        0                 1                   0
7                        0                 1                   0
8                        0                 1                   0
12                       0                 1                   0
19                       0                 1                   0

    department_technical
3                      0
7                      0
8                      0
12                     0
19                     0
```

[45]:
```
# Isolate the outcome variable
y = df_logreg['left']

# Display first few rows of the outcome variable
y.head()
```

[45]:
```
3     1
7     1
8     1
12    1
19    1
Name: left, dtype: int64
```

[48]:
```
# Select the features you want to use in your model
X = df_logreg.drop('left', axis=1)

# Display the first few rows of the selected features
X.head()
```

[48]:

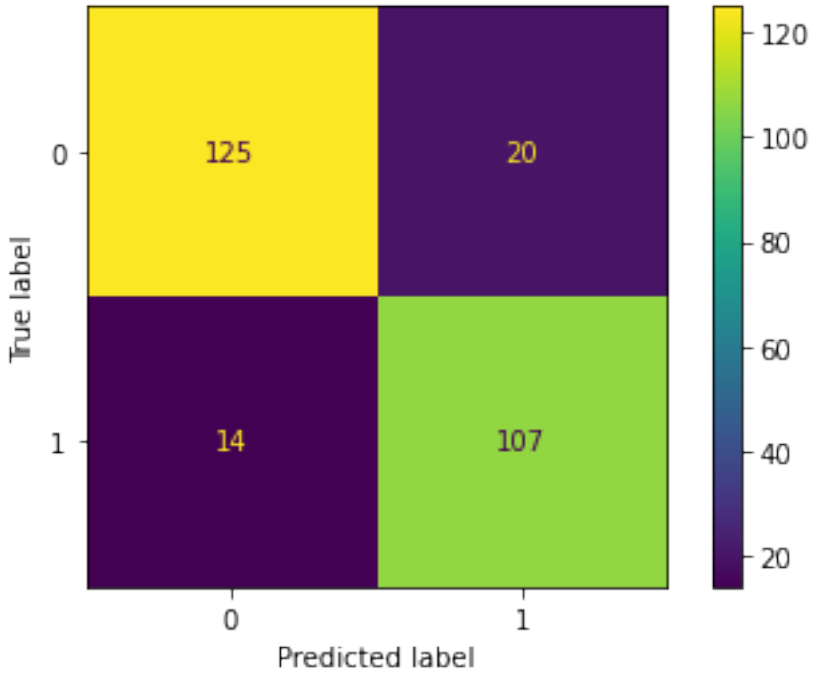
	satisfaction_level	last_evaluation	number_project	average_monthly_hours	tenure	work_accident	promotion_last_5years	salary	department_IT	department_RandD	department_accounting	department_hr	department_management	department_marketing	department_product_mng	department_sales	department_support	department_technical
3	0.72	0.87	5	223	5	0	0	0	0	0	0	0	0	0	0	1	0	0
7	0.92	0.85	5	259	5	0	0	0	0	0	0	0	0	0	0	1	0	0
8	0.89	1.00	5	224	5	0	0	0	0	0	0	0	0	0	0	1	0	0
12	0.84	0.92	4	234	5	0	0	0	0	0	0	0	0	0	0	1	0	0
19	0.76	0.89	5	262	5	0	0	0	0	0	0	0	0	0	0	1	0	0

[55]:
```
# Split the data into training set and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
↪stratify=y, random_state=42)
```

[71]:
```
# Construct a logistic regression model and fit it to the training dataset
log_clf = LogisticRegression(random_state=42, max_iter=500, solver='liblinear').
↪fit(X_train, y_train)
```

[72]:
```
# Use the logistic regression model to get predictions on the test set
y_pred = log_clf.predict(X_test)
```

[73]:
```
# Creating a confusion matrix to show the results of the logistic regression␣
↪model

# Results: The upper left shows the people the model predicted would not leave␣
↪and did not leave.
# The botton right shows the people the model predicted would leave and did␣
↪leave.
# The upper right shows false positives and the bottom left shows false␣
↪negatives.

# Compute values for confusion matrix
log_cm = confusion_matrix(y_test, 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(values_format='')

# Display plot
plt.show()
```



[74]:
```
# This reveals a 55-45 class split.
# While this is slightly imbalanced, it is not so imbalanced that we would to␣
 ↪resample the data

df_logreg['left'].value_counts(normalize=True)
```

[74]:
```
0    0.546139
1    0.453861
Name: left, dtype: float64
```

[75]:
```
# Creating classification report

# The report shows a precision, recall, f1-score, and accuracy all of 87%

target_names = ['Predicted would not leave', 'Predicted would leave']
print(classification_report(y_test, y_pred, target_names=target_names))
```

```
                           precision    recall  f1-score   support

Predicted would not leave       0.90      0.86      0.88       145
    Predicted would leave       0.84      0.88      0.86       121

                 accuracy                           0.87       266
                macro avg       0.87      0.87      0.87       266
             weighted avg       0.87      0.87      0.87       266
```

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?

6 Summary of model results

6.1 The classication report for the Logistic Regression model shows a precision, recall, f1-score, and accuracy all of 87%

7 Conclusions and Recommendations for the future

7.1 The Data shows that many employees in the company feel overworked.

7.2 To better retain employees, the following recommendations could be presented to shareholders:

7.2.1 Place a cap on the number of projects an employee can work on

7.2.2 Consider promoting employees that have been with the company for several years

7.2.3 Give employees more rewards for working extra hours, or do not require them to do so

7.2.4 Inform all employees about the companies overtime pay polies and the expectations around workload and timeoff

7.2.5 Hold company-wide and within-team discussions to understand the company work culture, across the board and in specific contexts

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.