Activity_ Course 7 Salifort Motors project lab

August 17, 2025

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?

[Double-click to enter your responses here.]

- The senior leadership team of Salifort Motors
- Build a model that predicts whether or not an employee will leave the company.
- From exploring the data intially there are 15000 rows and 10 columns which helps to discover the reasons behind their departure

2.2 Step 1. Imports

- Import packages
- Load dataset

2.2.1 Import packages

[1]:
```
# Import packages
# Import packages for data manipulation

import numpy as np
import pandas as pd


# Import packages for data visualization

import matplotlib.pyplot as plt
import seaborn as sns

# Import packages for data modeling

from sklearn.model_selection import GridSearchCV,␣
↪train_test_split,PredefinedSplit

from sklearn.metrics import accuracy_score, precision_score, recall_score,\
f1_score, confusion_matrix, ConfusionMatrixDisplay, RocCurveDisplay,␣
↪PrecisionRecallDisplay,classification_report,roc_auc_score

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier


from sklearn.tree import DecisionTreeClassifier,plot_tree


# This is the function that helps plot feature importance
from xgboost import plot_importance



# This module lets us save our models once we fit them.
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.

[2]:
```
# RUN THIS CELL TO IMPORT YOUR DATA.

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


# Display first few rows of the dataframe
### YOUR CODE HERE ###
df0.head(10)
```

[2]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	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
5	0.41	0.50	2	153	3	0	1	0	sales	low
6	0.10	0.77	6	247	4	0	1	0	sales	low
7	0.92	0.85	5	259	5	0	1	0	sales	low
8	0.89	1.00	5	224	5	0	1	0	sales	low
9	0.42	0.53	2	142	3	0	1	0	sales	low

```
4     low
5     low
6     low
7     low
8     low
9     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

[3]:
```
# Gather basic information about the data
### YOUR CODE HERE ###
df0.info()
#df0.value_counts('last_evaluation')
```

```
<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

[4]:
```
# Gather descriptive statistics about the data
### YOUR CODE HERE ###
df0.describe()
```

[4]:
```
       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]:
```
# Display all column names
### YOUR CODE HERE ###
df0.columns
```

[5]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
       'promotion_last_5years', 'Department', 'salary'],
      dtype='object')
```

[6]:
```
# Rename columns as needed
### YOUR CODE HERE ###
df0=df0.rename(columns={'Work_accident':'Work_accident',
                        'Department':'department',
                        'time_spend_company':'tenure',
                         'average_montly_hours':'average_monthly_hours'
                        })

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

[6]:
```
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.

[7]:
```
# Check for missing values
### YOUR CODE HERE ###
df0.isnull().sum()
```

[7]:
```
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.

[8]:
```
# Check for duplicates
### YOUR CODE HERE ###
df0.duplicated().sum()
```

[8]:
```
3008
```

[ ]:
```
```

[9]:
```
# Inspect some rows containing duplicates as needed
### YOUR CODE HERE ###
df0[df0.duplicated()].head()
```

[9]:
```
      satisfaction_level  last_evaluation  number_project  \
396                 0.46             0.57               2
866                 0.41             0.46               2
1317                0.37             0.51               2
1368                0.41             0.52               2
1461                0.42             0.53               2

      average_monthly_hours  tenure  Work_accident  left  \
396                     139       3              0     1
866                     128       3              0     1
1317                    127       3              0     1
1368                    132       3              0     1
1461                    142       3              0     1

      promotion_last_5years  department  salary
396                       0       sales     low
866                       0  accounting     low
1317                      0       sales  medium
1368                      0       RandD     low
1461                      0       sales     low
```

[10]:
```
# Drop duplicates and save resulting dataframe in a new variable as needed
### YOUR CODE HERE ###
df1=df0.drop_duplicates(keep='first')

# Display first few rows of new dataframe as needed
### YOUR CODE HERE ###
df1.head(10)
```

[10]:
```
   satisfaction_level  last_evaluation  number_project  average_monthly_hours  \
0                0.38             0.53               2                    157
1                0.80             0.86               5                    262
2                0.11             0.88               7                    272
3                0.72             0.87               5                    223
4                0.37             0.52               2                    159
5                0.41             0.50               2                    153
6                0.10             0.77               6                    247
7                0.92             0.85               5                    259
8                0.89             1.00               5                    224
```

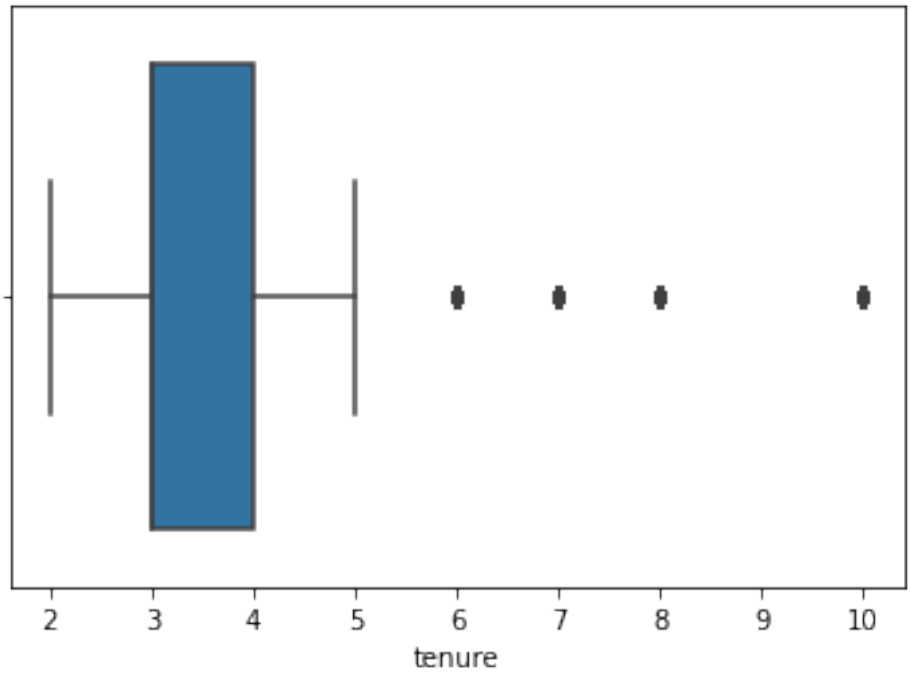
```
9                0.42             0.53                2                    142
```

	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
5	3	0	1	0	sales	low
6	4	0	1	0	sales	low
7	5	0	1	0	sales	low
8	5	0	1	0	sales	low
9	3	0	1	0	sales	low

2.3.6 Check outliers

Check for outliers in the data.

[11]:
```
# Create a boxplot to visualize distribution of `tenure` and detect any outliers
### YOUR CODE HERE ###
sns.boxplot(x=df1["tenure"])
plt.show()
```



[12]:
```
# Determine the number of rows containing outliers
### YOUR CODE HERE ###
Q1=df1['tenure'].quantile(.25)

Q2=df1['tenure'].quantile(.75)

iqr = Q2-Q1

upper_limit=Q2 + 1.5*iqr
lower_limit=Q1 - 1.5*iqr
print(upper_limit)
print(lower_limit)

outlier=df1[(df1['tenure']>upper_limit)|(df1['tenure']<lower_limit)]
len(outlier)
```

```
5.5
1.5
```

[12]: 824

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

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?

[Double-click to enter your responses here.]

3.1 Step 2. Data Exploration (Continue EDA)

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

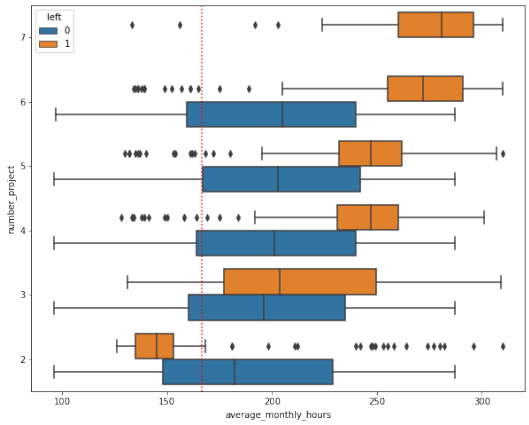
[13]:
```
# Get numbers of people who left vs. stayed
### YOUR CODE HERE ###
df1['left'].value_counts()
# Get percentages of people who left vs. stayed
### YOUR CODE HERE ###
df1['left'].value_counts(normalize=True)
```

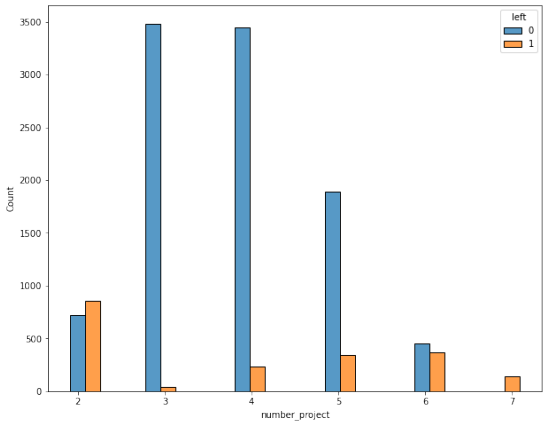
[13]:
```
0    0.833959
1    0.166041
Name: left, dtype: float64
```

3.1.1 Data visualizations

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

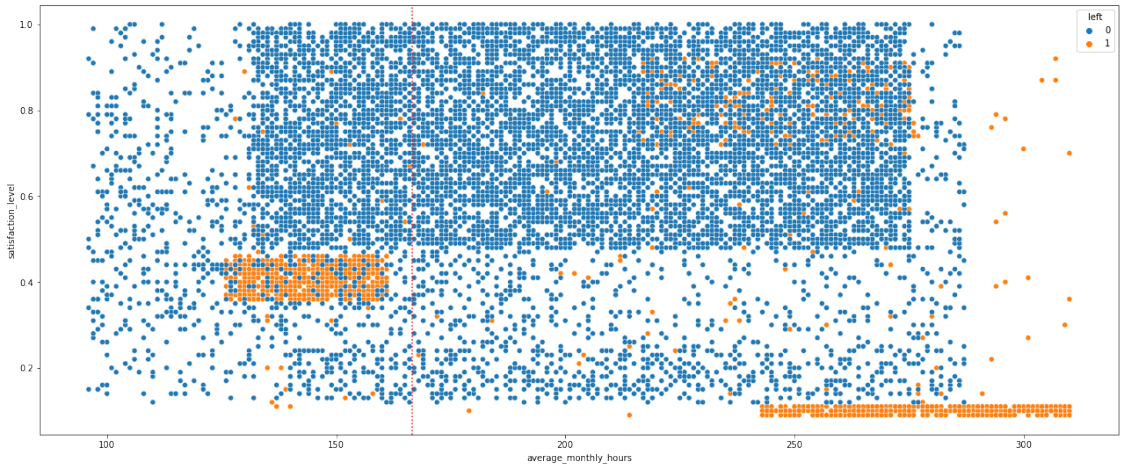
[14]:
```
# Create a plot as needed
### YOUR CODE HERE ###
fig, ax = plt.subplots(1,2, figsize = (22,8))

sns.boxplot(data=df1,x='average_monthly_hours',y='number_project',hue='left',␣
 ↪orient='h',ax=ax[0] ).axvline(x=166.67,color='r',ls=':')

ax[0].invert_yaxis()


#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])
plt.show()
```



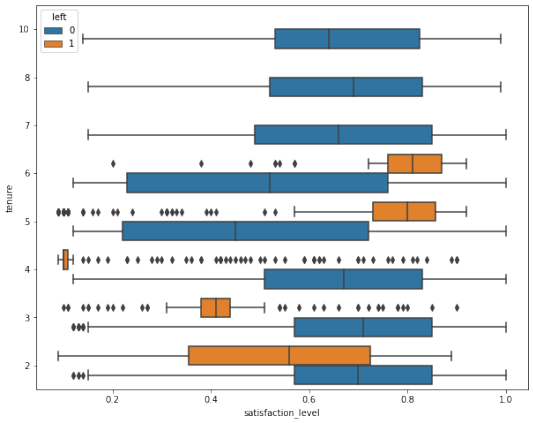


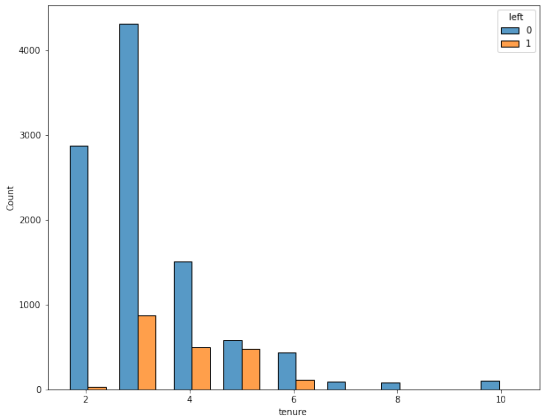
[15]:
```
# Create a plot as needed
### YOUR CODE HERE ###
plt.figure(figsize=(22,9))
sns.
 ↪scatterplot(data=df1,y='satisfaction_level',x='average_monthly_hours',hue='left').
 ↪axvline(x=166.67,color='r',ls=':')
```

[15]:
```
<matplotlib.lines.Line2D at 0x7c7113da2810>
```

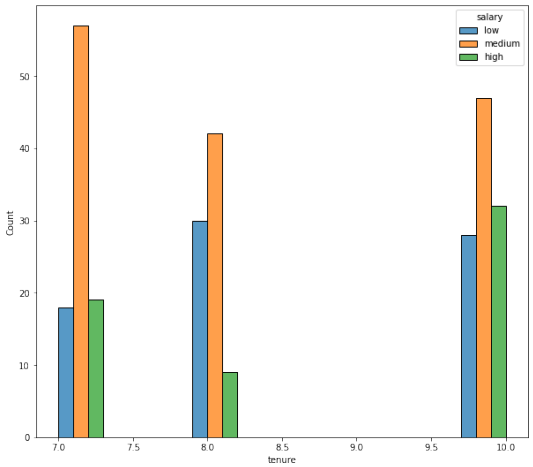


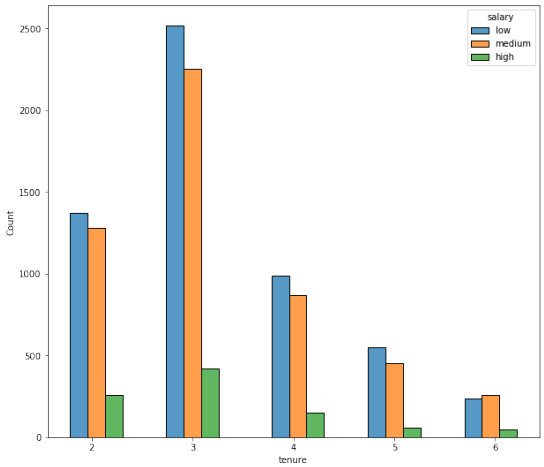
[16]:
```
# Create a plot as needed
### YOUR CODE HERE ###
fig, ax = plt.subplots(1,2, figsize = (22,8))

sns.boxplot(data=df1,x='satisfaction_level',y='tenure',hue='left',␣
 ↪orient='h',ax=ax[0] )

ax[0].invert_yaxis()


sns.histplot(data=df1, x='tenure', hue='left' ,multiple='dodge',shrink=8,␣
 ↪ax=ax[1])
plt.show()
```



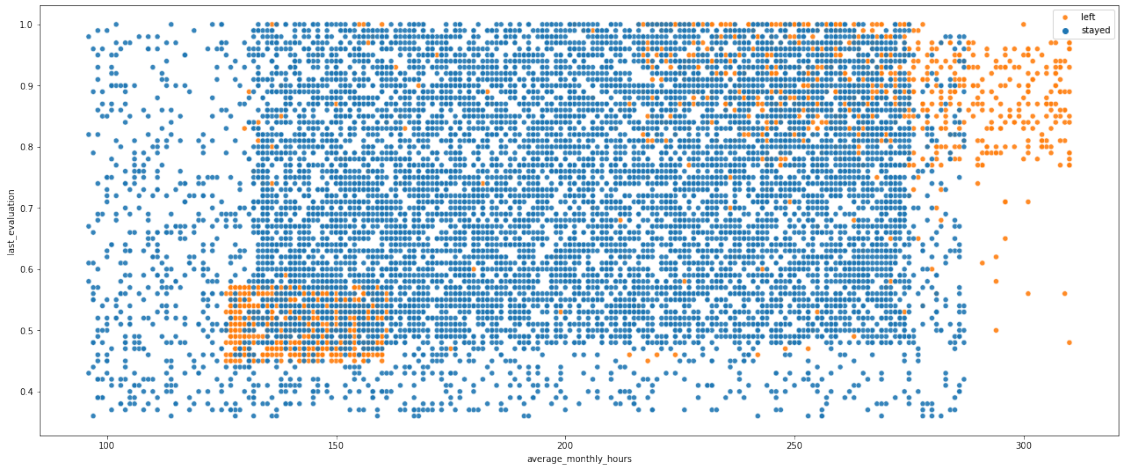


[17]:
```
# Create a plot as needed
### YOUR CODE HERE ###
fig, ax= plt.subplots(1,2,figsize=(22,9))

tenurs_long=df1[df1['tenure']>6]
tenurs_short=df1[df1['tenure']<7]

sns.histplot(data=tenurs_long,x='tenure',hue='salary',multiple='dodge' ,␣
↪hue_order=['low', 'medium', 'high'],ax=ax[0])

sns.
↪histplot(data=tenurs_short,x='tenure',hue='salary',multiple='dodge',shrink=6,␣
↪hue_order=['low', 'medium', 'high'],ax=ax[1])

plt.show()
```

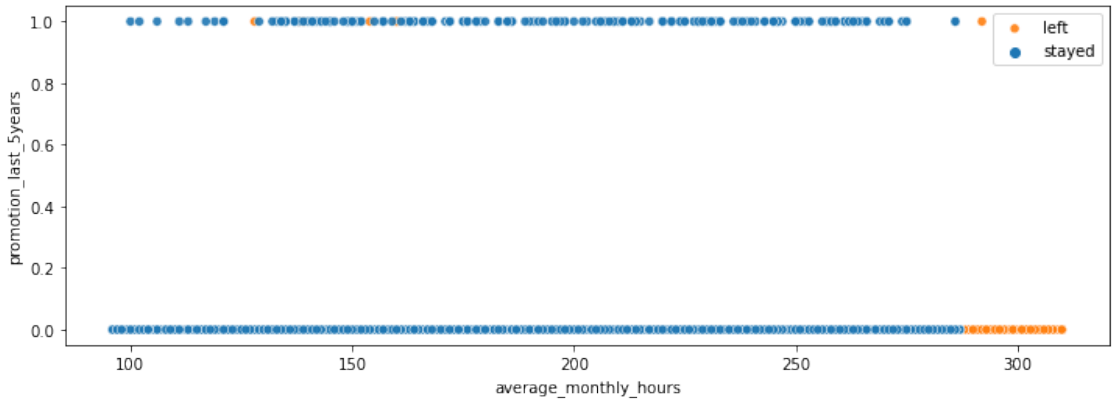




[18]:
```
# Create a plot as needed
### YOUR CODE HERE ###
plt.figure(figsize=(22,9))
sns.
 ↪scatterplot(data=df1,x='average_monthly_hours',y='last_evaluation',hue='left',alpha=0.
 ↪9)
plt.legend(labels=['left','stayed'])
plt.show()
```

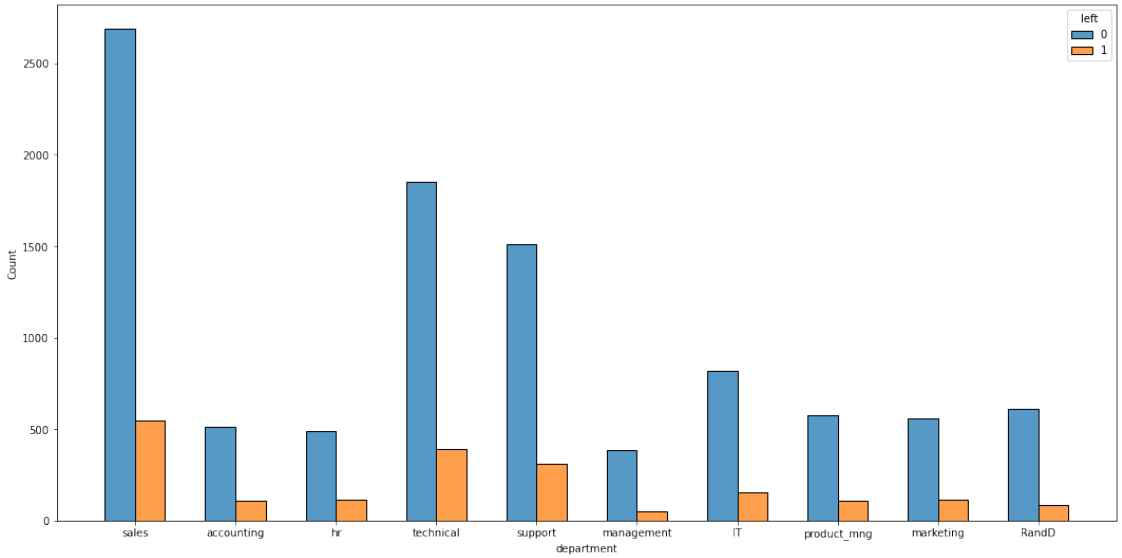


[19]:
```
# Create a plot as needed
### YOUR CODE HERE ###
plt.figure(figsize=(12,4))
sns.
 ↪scatterplot(data=df1,x='average_monthly_hours',y='promotion_last_5years',hue='left',alpha=0
 ↪9)
plt.legend(labels=['left','stayed'])
plt.show()
```



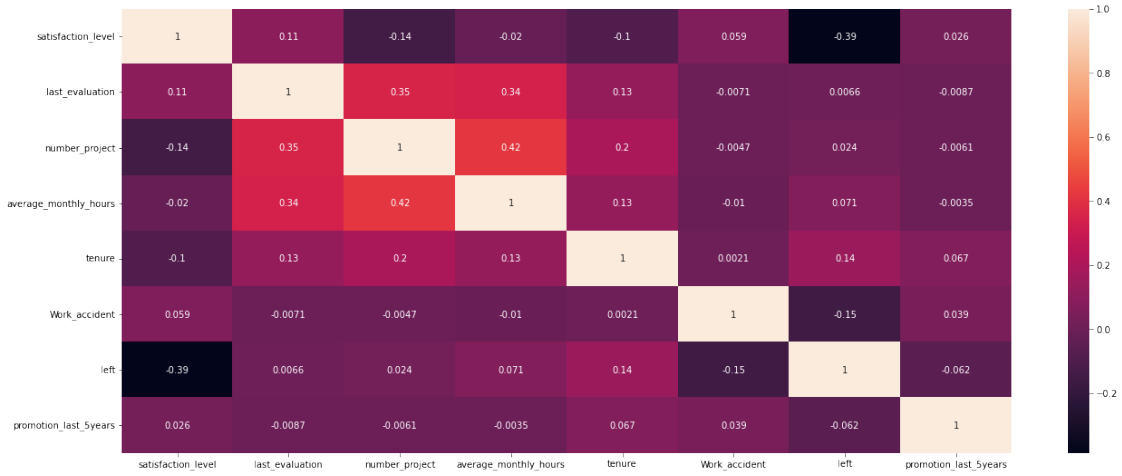
[20]:
```
# Create a plot as needed
### YOUR CODE HERE ###
plt.figure(figsize=(18,9))
sns.histplot(data=df1,x='department',hue='left',multiple='dodge' ,shrink=.6)
```

[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7c70fa984e10>



[21]:
```
# Create a plot as needed
### YOUR CODE HERE ###
plt.figure(figsize=(22,9))
sns.heatmap(data=df0.corr(),annot=True,vmax=1)
```

[21]: <matplotlib.axes._subplots.AxesSubplot at 0x7c711410c6d0>



3.1.2 Insights

[What insights can you gather from the plots you created to visualize the data? Double-click to enter your responses here.]

It appears that employees are leaving the company as a result of poor management. Leaving is tied to longer working hours, many projects, and generally lower satisfaction levels. It can be ungratifying to work long hours and not receive promotions or good evaluation scores. There's a sizeable group of employees at this company who are probably burned out. It also appears that if an employee has spent more than six years at the company, they tend not to leave.

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?

[Double-click to enter your responses here.]

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.

[Double-click to enter your responses here.]

The goal of the model is to predict whether the employees left or stayed which is categorical outcome which is either is 1 or 0

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

[Double-click to enter your responses here.]

The outcome variable is categorial hence we either use logestic regresion or tree calsifier or xgboost

4.1.3 Modeling

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

[22]:
```
df1.dtypes
```

[22]:
```
satisfaction_level       float64
last_evaluation          float64
number_project             int64
average_monthly_hours      int64
tenure                     int64
Work_accident              int64
left                       int64
promotion_last_5years      int64
department                object
salary                    object
dtype: object
```

[23]:
```
from sklearn.preprocessing import OneHotEncoder,OrdinalEncoder
```

[24]:
```
oe=OrdinalEncoder(categories=[['low','medium','high']])
df1[['salary']]=oe.fit_transform(df1[['salary']])
```

[25]:
```
df1=pd.get_dummies(df1,drop_first=False)
```

[26]:
```
df1
```

[26]:

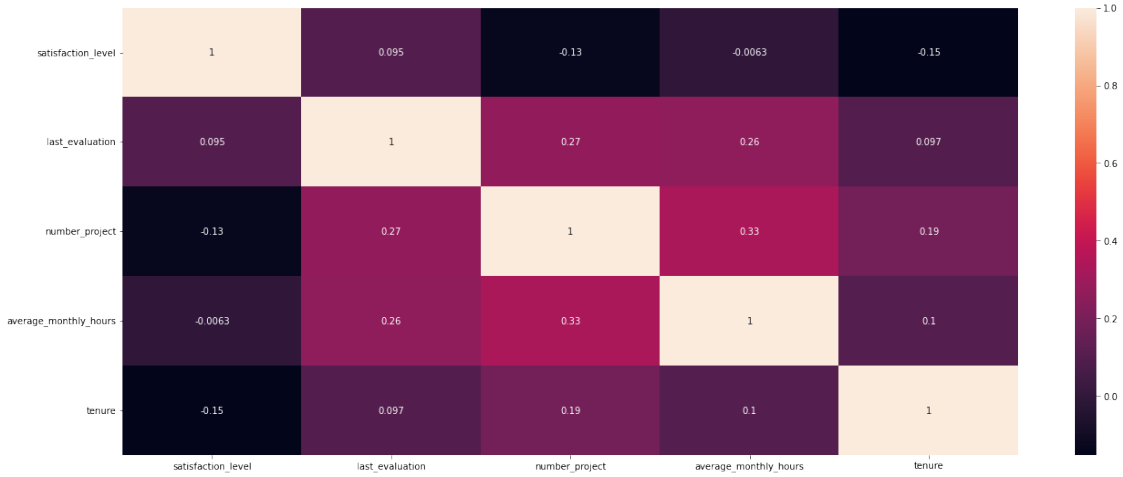
	satisfaction_level	last_evaluation	number_project	\
0	0.38	0.53	2	
1	0.80	0.86	5	
2	0.11	0.88	7	
3	0.72	0.87	5	
4	0.37	0.52	2	
…	…	…	…	

```
11995                0.90             0.55               3
11996                0.74             0.95               5
11997                0.85             0.54               3
11998                0.33             0.65               3
11999                0.50             0.73               4

       average_monthly_hours  tenure  Work_accident  left  \
0                        157       3              0     1
1                        262       6              0     1
2                        272       4              0     1
3                        223       5              0     1
4                        159       3              0     1
…                        …     …            …    …
11995                    259      10              1     0
11996                    266      10              0     0
11997                    185      10              0     0
11998                    172      10              0     0
11999                    180       3              0     0

       promotion_last_5years  salary  department_IT  department_RandD  \
0                          0     0.0              0                 0
1                          0     1.0              0                 0
2                          0     1.0              0                 0
3                          0     0.0              0                 0
4                          0     0.0              0                 0
…                          …     …            …               …
11995                      1     2.0              0                 0
11996                      1     2.0              0                 0
11997                      1     2.0              0                 0
11998                      1     2.0              0                 0
11999                      0     0.0              1                 0

       department_accounting  department_hr  department_management  \
0                          0              0                      0
1                          0              0                      0
2                          0              0                      0
3                          0              0                      0
4                          0              0                      0
…                          …            …                    …
11995                      0              0                      1
11996                      0              0                      1
11997                      0              0                      1
11998                      0              0                      0
11999                      0              0                      0

       department_marketing  department_product_mng  department_sales  \
0                         0                       0                 1
```

```
1                          0                          0                  1
2                          0                          0                  1
3                          0                          0                  1
4                          0                          0                  1
...                      ...                        ...                ...
11995                      0                          0                  0
11996                      0                          0                  0
11997                      0                          0                  0
11998                      1                          0                  0
11999                      0                          0                  0

       department_support  department_technical
0                       0                     0
1                       0                     0
2                       0                     0
3                       0                     0
4                       0                     0
...                   ...                   ...
11995                   0                     0
11996                   0                     0
11997                   0                     0
11998                   0                     0
11999                   0                     0

[11991 rows x 19 columns]
```

[27]:
```
plt.figure(figsize=(22,9))
sns.
 ↪heatmap(data=df1[['satisfaction_level','last_evaluation','number_project','average_monthly_
 ↪corr(),annot=True,vmax=1)
```

[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7c70fce37790>
```



[28]:
```
mdf2 = df1[(df1['tenure']<=upper_limit)&(df1['tenure']>=lower_limit)]
mdf2
```

[28]:

```
       satisfaction_level  last_evaluation  number_project  \
0                    0.38             0.53               2
2                    0.11             0.88               7
3                    0.72             0.87               5
4                    0.37             0.52               2
5                    0.41             0.50               2
…                     …                …               …
11985                0.72             0.64               4
11986                0.48             0.50               5
11987                0.19             0.79               4
11992                0.62             0.85               3
11999                0.50             0.73               4

       average_monthly_hours  tenure  Work_accident  left  \
0                        157       3              0     1
2                        272       4              0     1
3                        223       5              0     1
4                        159       3              0     1
5                        153       3              0     1
…                        …     …              …   …
11985                    192       3              0     0
11986                    142       4              0     0
11987                    229       4              0     0
11992                    237       3              1     0
11999                    180       3              0     0

       promotion_last_5years  salary  department_IT  department_RandD  \
0                          0     0.0              0                 0
2                          0     1.0              0                 0
3                          0     0.0              0                 0
4                          0     0.0              0                 0
5                          0     0.0              0                 0
…                          …     …              …                 …
11985                      0     1.0              0                 0
11986                      0     1.0              1                 0
11987                      0     1.0              0                 0
11992                      0     1.0              1                 0
11999                      0     0.0              1                 0

       department_accounting  department_hr  department_management  \
0                          0              0                      0
2                          0              0                      0
3                          0              0                      0
```

```
4                          0              0                      0
5                          0              0                      0
…                        …              …                      …
11985                      0              0                      0
11986                      0              0                      0
11987                      0              0                      0
11992                      0              0                      0
11999                      0              0                      0

       department_marketing  department_product_mng  department_sales  \
0                         0                       0                 1
2                         0                       0                 1
3                         0                       0                 1
4                         0                       0                 1
5                         0                       0                 1
…                       …                       …               …
11985                     0                       0                 1
11986                     0                       0                 0
11987                     0                       1                 0
11992                     0                       0                 0
11999                     0                       0                 0

       department_support  department_technical
0                       0                     0
2                       0                     0
3                       0                     0
4                       0                     0
5                       0                     0
…                     …                     …
11985                   0                     0
11986                   0                     0
11987                   0                     0
11992                   0                     0
11999                   0                     0

[11167 rows x 19 columns]
```

[29]:
```
y=mdf2['left']
x=mdf2.drop('left',axis=1)
```

[30]:
```
x_train,x_test,y_train,y_test=  train_test_split(x, y, test_size=0.25,␣
↪stratify=y, random_state=42)
```

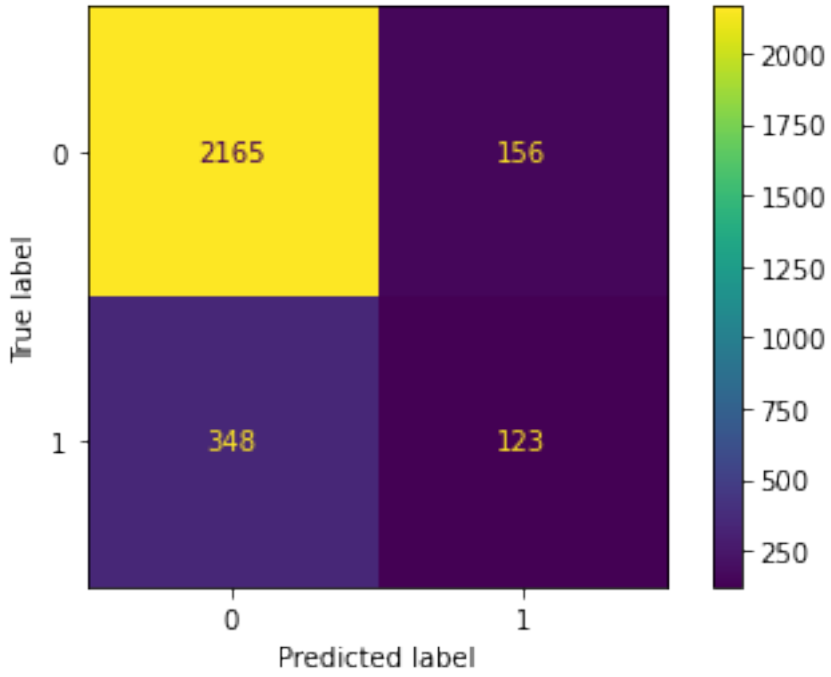
[31]:
```
from sklearn.linear_model import LogisticRegression
```

[32]:
```
log_clf= LogisticRegression(random_state=42,max_iter=500).fit(x_train,y_train)
```

[33]:
```
y_pred= log_clf.predict(x_test)
y_pred
```

[33]:
```
array([0, 0, 0, …, 0, 0, 0])
```

[34]:
```
cm=confusion_matrix(y_test,y_pred,labels=log_clf.classes_)
disp=ConfusionMatrixDisplay(cm,display_labels=log_clf.classes_)
disp.plot(values_format='')
plt.show()
```



[35]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.93      0.90      2321
           1       0.44      0.26      0.33       471

    accuracy                           0.82      2792
   macro avg       0.65      0.60      0.61      2792
weighted avg       0.79      0.82      0.80      2792
```

[36]:
```
X_train,X_test,Y_train,Y_test=  train_test_split(x, y, test_size=0.25,␣
↪stratify=y, random_state=0)
```

[37]:
```
tree= DecisionTreeClassifier(random_state=0)

cv_params={'max_depth':[4,6,8,None],
           'min_samples_leaf': [2, 5, 1],
           'min_samples_split':[2,4,6]}

scoring = {'accuracy', 'precision', 'recall', 'f1', 'roc_auc'}

tree1 = GridSearchCV(tree , cv_params, scoring=scoring ,cv=4, refit='roc_auc')
```

[38]:
```
tree1.fit(X_train,Y_train)
```

[38]:
```
GridSearchCV(cv=4, error_score=nan,
             estimator=DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort='deprecated',
                                              random_state=0, splitter='best'),
             iid='deprecated', n_jobs=None,
             param_grid={'max_depth': [4, 6, 8, None],
                         'min_samples_leaf': [2, 5, 1],
                         'min_samples_split': [2, 4, 6]},
             pre_dispatch='2*n_jobs', refit='roc_auc', return_train_score=False,
             scoring={'precision', 'f1', 'recall', 'accuracy', 'roc_auc'},
             verbose=0)
```

[39]:
```
tree1.best_params_
```

[39]:
```
{'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 2}
```

[40]:
```
tree1.best_score_
```

[40]:
```
0.9743823751317063
```

[41]:
```
cv=pd.DataFrame(tree1.cv_results_)
cv.head()
```

[41]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	\
0	0.007919	0.000468	0.006310	0.000284	
1	0.007504	0.000114	0.006048	0.000069	
2	0.007487	0.000096	0.005973	0.000127	

```
3       0.008767      0.002013         0.007749        0.001660
4       0.007459      0.000085         0.005933        0.000067

  param_max_depth param_min_samples_leaf param_min_samples_split  \
0               4                      2                       2
1               4                      2                       4
2               4                      2                       6
3               4                      5                       2
4               4                      5                       4

                                              params  split0_test_precision  \
0  {'max_depth': 4, 'min_samples_leaf': 2, 'min_s…              0.956268
1  {'max_depth': 4, 'min_samples_leaf': 2, 'min_s…              0.956268
2  {'max_depth': 4, 'min_samples_leaf': 2, 'min_s…              0.956268
3  {'max_depth': 4, 'min_samples_leaf': 5, 'min_s…              0.956012
4  {'max_depth': 4, 'min_samples_leaf': 5, 'min_s…              0.956012

   split1_test_precision  …  mean_test_accuracy  std_test_accuracy  \
0                0.94012  …            0.978508           0.004615
1                0.94012  …            0.978508           0.004615
2                0.94012  …            0.978508           0.004615
3                0.94012  …            0.977792           0.004275
4                0.94012  …            0.977792           0.004275

   rank_test_accuracy  split0_test_roc_auc  split1_test_roc_auc  \
0                  22             0.983591             0.966199
1                  22             0.983591             0.966199
2                  22             0.983591             0.966199
3                  28             0.983384             0.964095
4                  28             0.983384             0.964095

   split2_test_roc_auc  split3_test_roc_auc  mean_test_roc_auc  \
0             0.967327             0.980412           0.974382
1             0.967327             0.980412           0.974382
2             0.967327             0.980412           0.974382
3             0.964386             0.980458           0.973081
4             0.964386             0.980458           0.973081

   std_test_roc_auc  rank_test_roc_auc
0          0.007712                  1
1          0.007712                  1
2          0.007712                  1
3          0.008901                  4
4          0.008901                  4

[5 rows x 43 columns]
```

[42]:
```
def make_results (model_name,model_objects, metric):
    metric_dict = {'auc': 'mean_test_roc_auc',
                   'precision': 'mean_test_precision',
                   'recall': 'mean_test_recall',
                   'f1': 'mean_test_f1',
                   'accuracy': 'mean_test_accuracy'
                  }

    best_estimator_results=cv.iloc[cv[metric_dict[metric]].idxmax(),:]

    auc = best_estimator_results.mean_test_roc_auc
    f1 = best_estimator_results.mean_test_f1
    recall = best_estimator_results.mean_test_recall
    precision = best_estimator_results.mean_test_precision
    accuracy = best_estimator_results.mean_test_accuracy


    table = pd.DataFrame()
    table = pd.DataFrame({'model': [model_name],
                          'precision': [precision],
                          'recall': [recall],
                          'F1': [f1],
                          'accuracy': [accuracy],
                          'auc': [auc]
                         })
    return table
```

[43]:
```
tree1_cv_results = make_results('decision tree cv', tree1, 'auc')
tree1_cv_results
```

[43]:

	model	precision	recall	F1	accuracy	auc
0	decision tree cv	0.955522	0.91497	0.934765	0.978508	0.974382

[44]:
```
rfc_clf= RandomForestClassifier(random_state=0)

cv_params = {'max_depth': [3,5, None],
             'max_features': [1.0],
             'max_samples': [0.7, 1.0],
             'min_samples_leaf': [1,2,3],
             'min_samples_split': [2,3,4],
             'n_estimators': [300, 500],
             }

scoring = {'accuracy', 'precision', 'recall', 'f1', 'roc_auc'}
```

```
forest1 = GridSearchCV(rfc_clf , cv_params, scoring=scoring␣
↪,cv=4,refit='roc_auc')
```

[70]:
```
%%time
forest1.fit(X_train,Y_train)
```

```
CPU times: user 8min 43s, sys: 267 ms, total: 8min 44s
Wall time: 8min 44s
```

[70]:
```
GridSearchCV(cv=4, error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              max_samples=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators=100, n_jobs=None,…
                                              verbose=0, warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'max_depth': [3, 5, None], 'max_features': [1.0],
                         'max_samples': [0.7, 1.0],
                         'min_samples_leaf': [1, 2, 3],
                         'min_samples_split': [2, 3, 4],
                         'n_estimators': [300, 500]},
             pre_dispatch='2*n_jobs', refit='roc_auc', return_train_score=False,
             scoring={'precision', 'f1', 'recall', 'accuracy', 'roc_auc'},
             verbose=0)
```

[46]:
```
path = '/home/jovyan/work/'
```

[47]:
```
def write_pickle(path, model_object, save_as:str):
    '''
    In:
        path:         path of folder where you want to save the pickle
        model_object: a model you want to pickle
        save_as:      filename for how you want to save the model

    Out: A call to pickle the model in the folder indicated
    '''

    with open(path + save_as + '.pickle', 'wb') as to_write:
```

```
        pickle.dump(model_object, to_write)
```

[48]:
```
def read_pickle(path, saved_model_name:str):
    '''
    In:
        path:             path to folder where you want to read from
        saved_model_name: filename of pickled model you want to read in

    Out:
        model: the pickled model
    '''
    with open(path + saved_model_name + '.pickle', 'rb') as to_read:
        model = pickle.load(to_read)

    return model
```

[49]:
```
write_pickle(path, forest1, 'hr_rf1')
```

[50]:
```
rf1 = read_pickle(path, 'hr_rf1')
```

[51]:
```
forest1.best_score_
```

[51]:
```
0.9818158627884357
```

[52]:
```
tree1.best_score_
```

[52]:
```
0.9743823751317063
```

[53]:
```
forest1.best_params_
```

[53]:
```
{'max_depth': 5,
 'max_features': 1.0,
 'max_samples': 0.7,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 300}
```

[54]:
```
random_forest =make_results('Random Forest',forest1,'auc')
random_forest
```

[54]:

	model	precision	recall	F1	accuracy	auc
0	Random Forest	0.955522	0.91497	0.934765	0.978508	0.974382

[55]:
```
tree1_cv_results = make_results('decision tree cv', tree1, 'auc')
tree1_cv_results
```

[55]:
	model	precision	recall	F1	accuracy	auc
0	decision tree cv	0.955522	0.91497	0.934765	0.978508	0.974382

[56]:
```
def get_scores(model_name:str, model, X_test_data, y_test_data):
    '''
    Generate a table of test scores.

    In:
        model_name (string):  How you want your model to be named in the output table
        model:                A fit GridSearchCV object
        X_test_data:          numpy array of X_test data
        y_test_data:          numpy array of y_test data

    Out: pandas df of precision, recall, f1, accuracy, and AUC scores for your model
    '''

    preds = model.best_estimator_.predict(X_test_data)

    auc = roc_auc_score(y_test_data, preds)
    accuracy = accuracy_score(y_test_data, preds)
    precision = precision_score(y_test_data, preds)
    recall = recall_score(y_test_data, preds)
    f1 = f1_score(y_test_data, preds)

    table = pd.DataFrame({'model': [model_name],
                          'precision': [precision],
                          'recall': [recall],
                          'f1': [f1],
                          'accuracy': [accuracy],
                          'AUC': [auc]
                         })

    return table
```

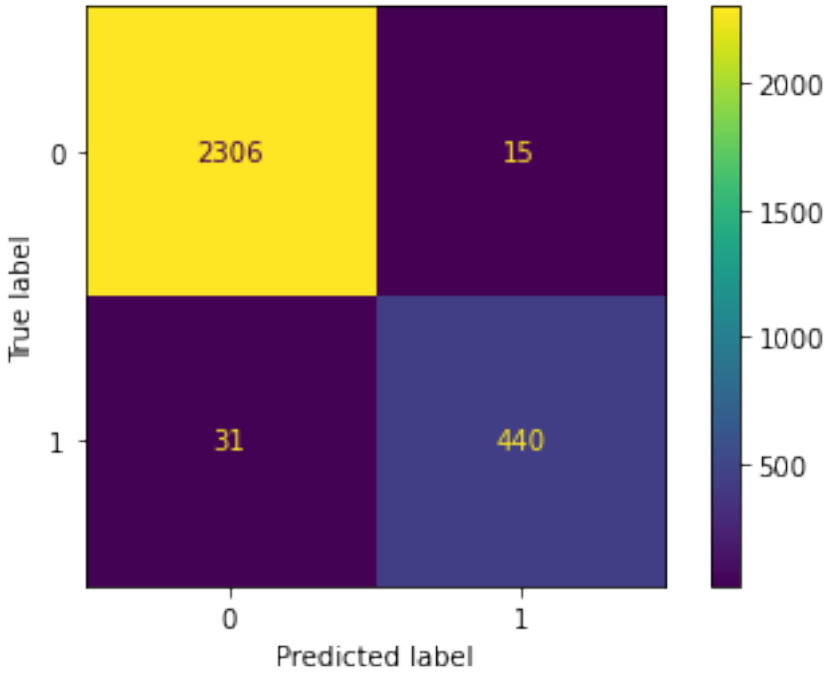
[57]:
```
scores=get_scores('Random Forest', forest1, X_test, Y_test)
scores
```

[57]:
	model	precision	recall	f1	accuracy	AUC
0	Random Forest	0.967033	0.934183	0.950324	0.983524	0.96386

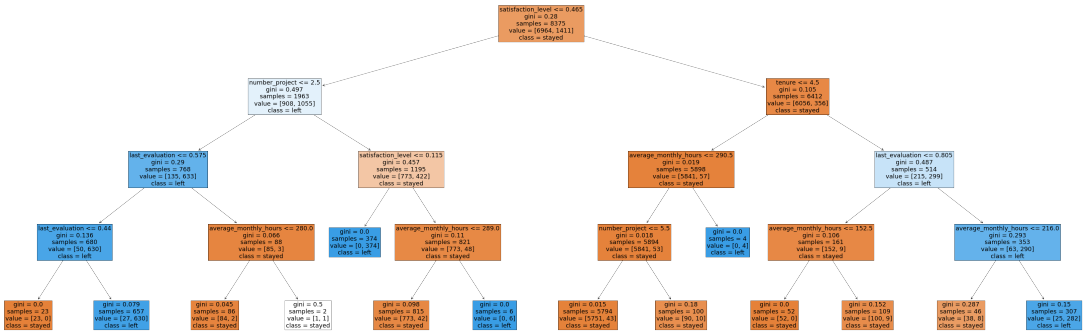
[58]:
```
predes = forest1.best_estimator_.predict(X_test)
```

[59]:
```
cm=confusion_matrix(Y_test,predes,labels=forest1.classes_)
disp=ConfusionMatrixDisplay(cm,display_labels=forest1.classes_)
disp.plot(values_format='')
```

```
plt.show()
```



[60]:
```
plt.figure(figsize=(60,20))

plot_tree(tree1.best_estimator_,max_depth=4,fontsize=18,feature_names=x.columns,
          class_names={0:'stayed',1:'left'},filled=True);
plt.show()
```



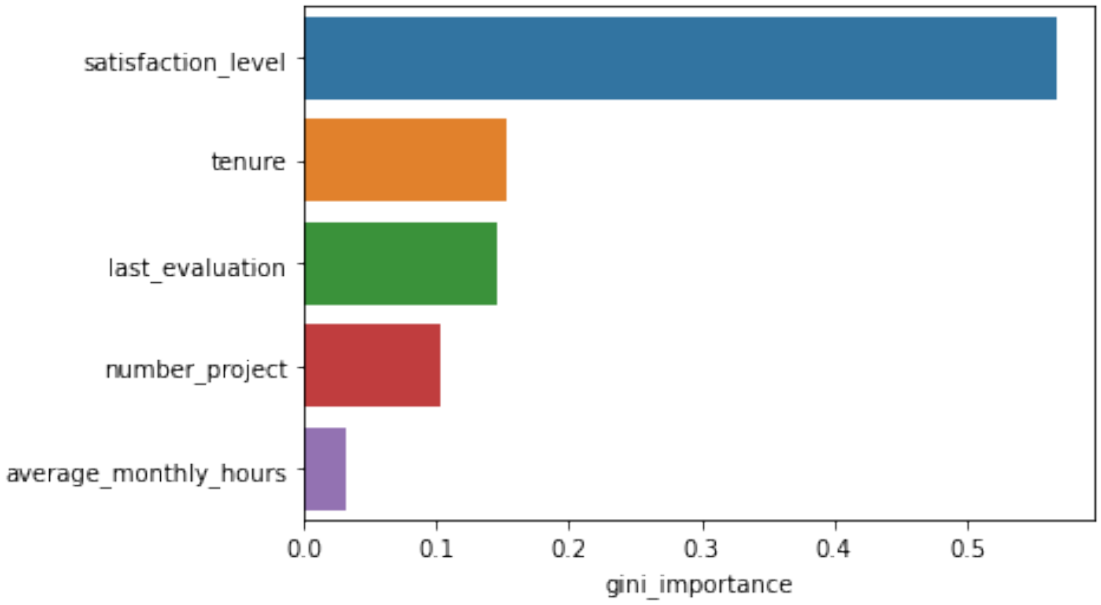
[61]:
```
tree_importance = pd.DataFrame(tree1.best_estimator_.feature_importances_,
                               index=x.columns,
                               columns=['gini_importance'])
```

```
tree_importance=tree_importance[tree_importance['gini_importance']!=0]

tree_importance=tree_importance.
 ↪sort_values(by='gini_importance',ascending=False)
tree_importance
```

[61]:

	gini_importance
satisfaction_level	0.567974
tenure	0.152343
last_evaluation	0.145035
number_project	0.102990
average_monthly_hours	0.031658

[62]:
```
sns.barplot(data=tree_importance,x='gini_importance',y=tree_importance.index)
plt.show()
```

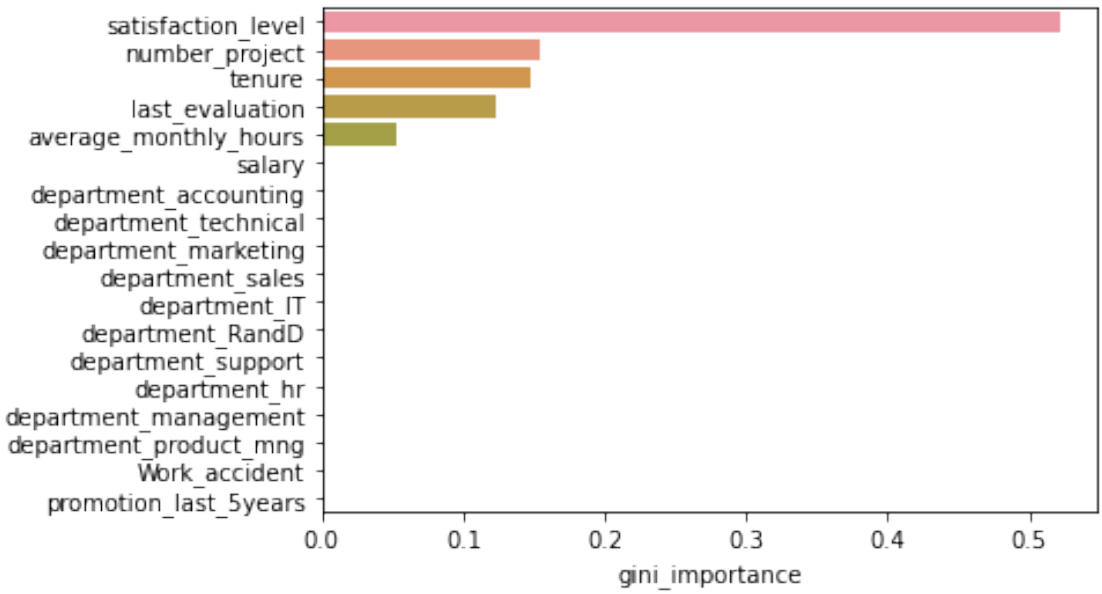


[63]:
```
feat_impt = pd.DataFrame(forest1.best_estimator_.feature_importances_,
                                 index=x.columns,
                                 columns=['gini_importance'])

feat_=tree_importance[tree_importance['gini_importance']!=0]

feat_impt=feat_impt.sort_values(by='gini_importance',ascending=False)
feat_impt
```

[63]:

	gini_importance
satisfaction_level	0.521505
number_project	0.153997
tenure	0.147826
last_evaluation	0.122281
average_monthly_hours	0.052708
salary	0.000306
department_accounting	0.000279
department_technical	0.000192
department_marketing	0.000183
department_sales	0.000171
department_IT	0.000168
department_RandD	0.000140
department_support	0.000109
department_hr	0.000051
department_management	0.000050
department_product_mng	0.000022
Work_accident	0.000010
promotion_last_5years	0.000002

[64]:
```
sns.barplot(data=feat_impt,x='gini_importance',y=feat_impt.index)
plt.show()
```



[65]:
```
xgb = XGBClassifier(objective='binary:logistic', random_state=0)
```

[66]:
```
cv_params = {'max_depth': [4, 6],
             'min_child_weight': [3, 5],
```

```
               'learning_rate': [0.1, 0.2, 0.3],
              'n_estimators': [5,10,15],
              'subsample': [0.7],
               'colsample_bytree': [0.7]
               }

scoring = {'accuracy', 'precision', 'recall', 'f1','roc_auc'}
```

[67]:
```
xgb_cv = GridSearchCV(xgb,
                      cv_params,
                      scoring = scoring,
                      cv = 4,
                      refit = 'roc_auc'
                     )
```

[68]:
```
%%time
xgb_cv = xgb_cv.fit(X_train, Y_train)
xgb_cv
```

```
CPU times: user 1h 4min 40s, sys: 0 ns, total: 1h 4min 40s
Wall time: 32min 22s
```

[68]:
```
GridSearchCV(cv=4, 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…
                                     predictor=None, random_state=0,
                                     reg_alpha=None, …),
             iid='deprecated', n_jobs=None,
             param_grid={'colsample_bytree': [0.7],
                         'learning_rate': [0.1, 0.2, 0.3], 'max_depth': [4, 6],
                         'min_child_weight': [3, 5],
                         'n_estimators': [5, 10, 15], 'subsample': [0.7]},
             pre_dispatch='2*n_jobs', refit='roc_auc', return_train_score=False,
             scoring={'precision', 'f1', 'recall', 'accuracy', 'roc_auc'},
             verbose=0)
```

[72]:
```
scores=get_scores('Random Forest', xgb_cv, X_test, Y_test)
scores
```

[72]:

	model	precision	recall	f1	accuracy	AUC
0	Random Forest	0.964602	0.92569	0.944745	0.981734	0.959398

[74]:
```
xgb_cv.best_score_
```

[74]:
```
0.9835412228841666
```

[ ]:
```

```