

Data Understanding Phase

Import libraries

```
In [1]: import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
sns.set(color_codes=True)  
%matplotlib inline
```

Import data using pandas

```
In [2]: df = pd.read_csv('HR_comma_sep.csv')
```

```
In [3]: df.head()
```

Out[3]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company
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	



Describe data

In [4]: `df.describe()`

Out[4]:

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.0
mean	0.612834	0.716102	3.803054	201.050337	3.4
std	0.248631	0.171169	1.232592	49.943099	1.4
min	0.090000	0.360000	2.000000	96.000000	2.0
25%	0.440000	0.560000	3.000000	156.000000	3.0
50%	0.640000	0.720000	4.000000	200.000000	3.0
75%	0.820000	0.870000	5.000000	245.000000	4.0
max	1.000000	1.000000	7.000000	310.000000	10.0

Explore Data

Check data for missing values (data quality)

In [5]: `missing_values = df.isnull()
sns.heatmap(data = missing_values, yticklabels=False, cbar=False, cmap='viridis')`

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x103a97128>



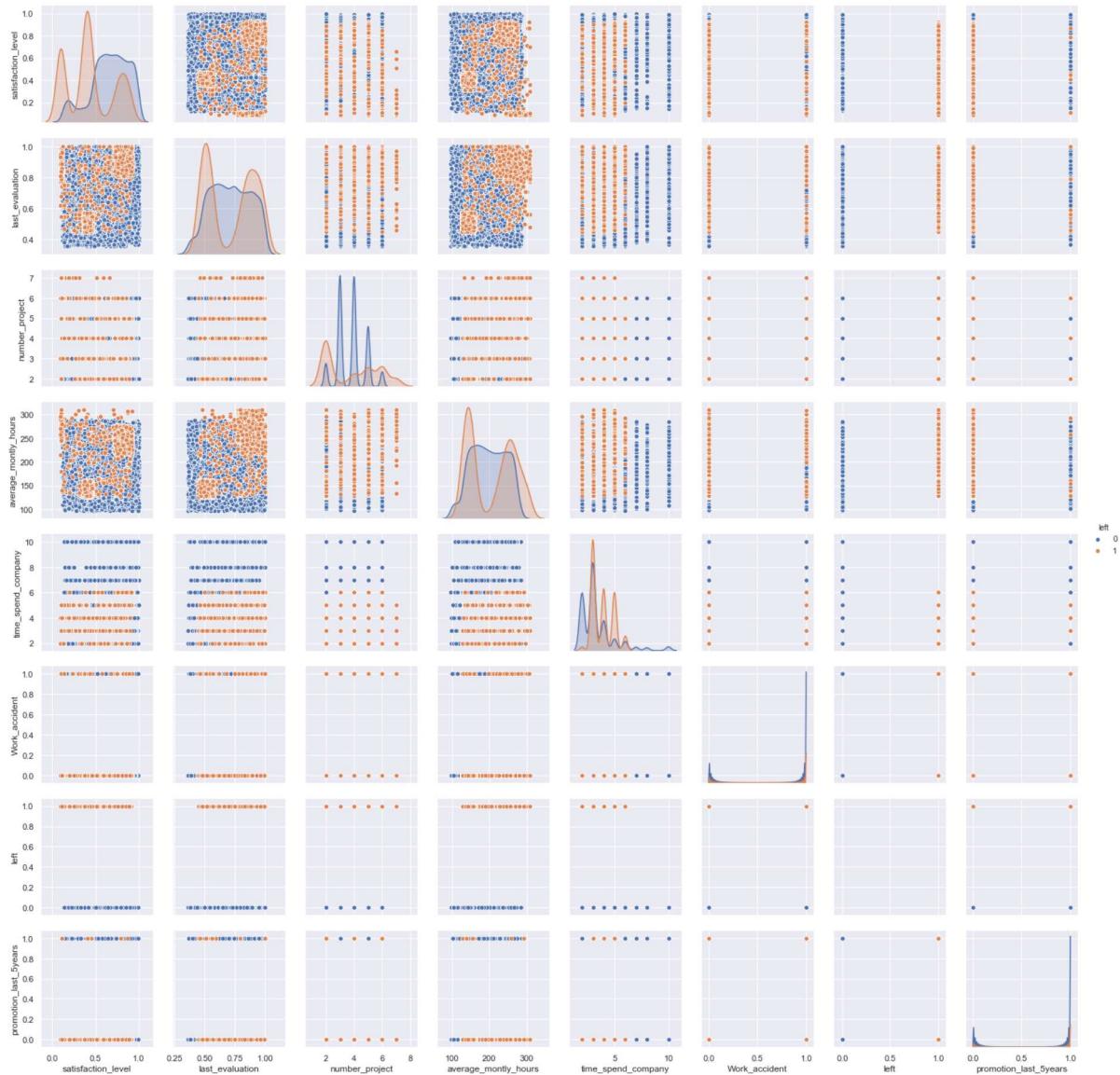
Since there is no missing data, we do not need to augment values

Seaborn pairplot of all features

In [6]: `sns.pairplot(df, hue='left')`

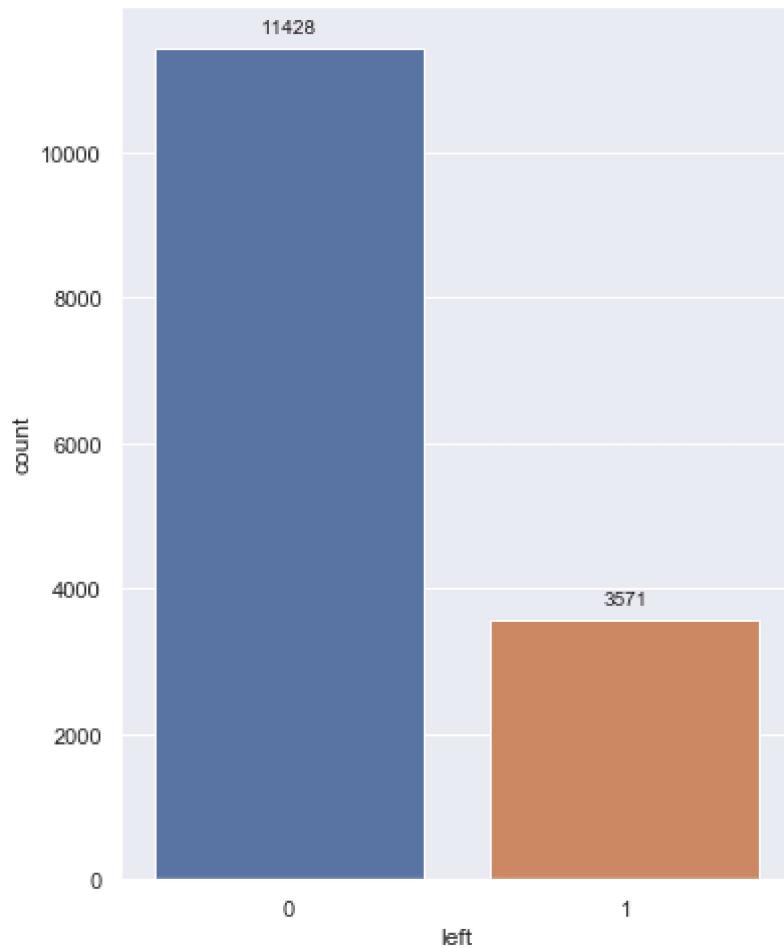
```
/Users/joshuasheriff/anaconda3/lib/python3.7/site-packages/statsmodels/nonpar
ametric/kde.py:487: RuntimeWarning: invalid value encountered in true_
divide
    binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/Users/joshuasheriff/anaconda3/lib/python3.7/site-packages/statsmodels/nonpar
ametric/kdetools.py:34: RuntimeWarning: invalid value encountered in double_s
calars
    FAC1 = 2*(np.pi*bw/RANGE)**2
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x1a176b8240>



Display bar chart for # of employees that left vs stayed

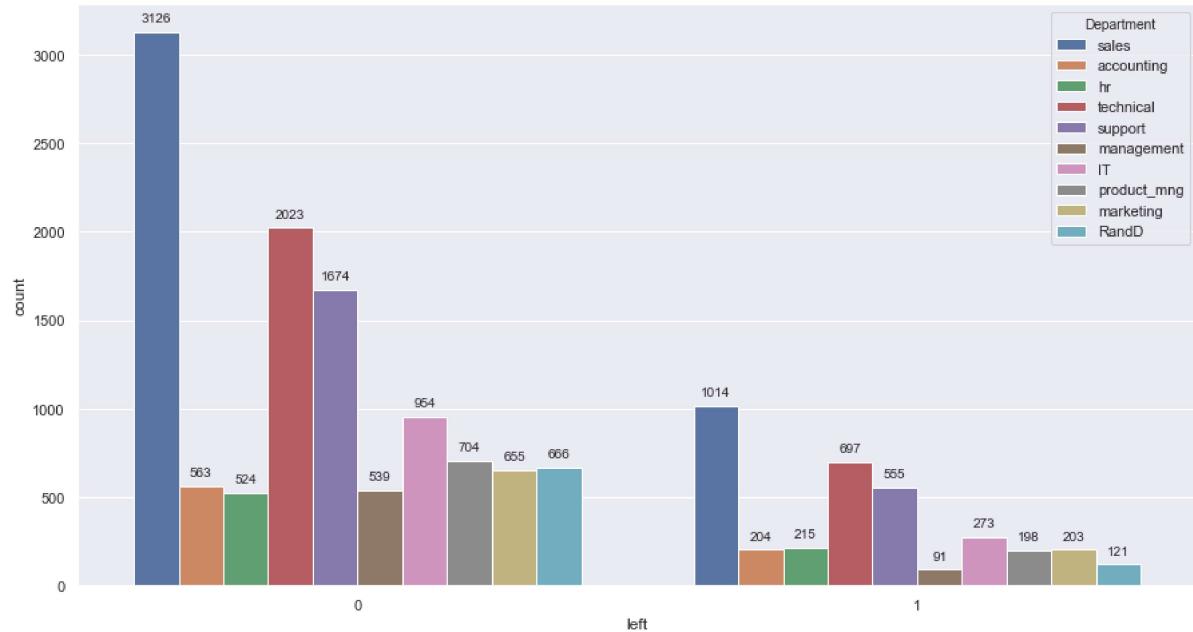
```
In [7]: plt.figure(figsize=(6, 8))
count_left = sns.countplot(x='left', data=df)
for p in count_left.patches:
    count_left.annotate(format(p.get_height()), (p.get_x() + p.get_width() / 2., p.get_height()), ha = 'center', va = 'center', xytext = (0, 10), textcoords = 'offset points')
```



Display bar chart for employee status by department

```
In [8]: #sns.countplot(x='Left', data=df, hue='Department')
```

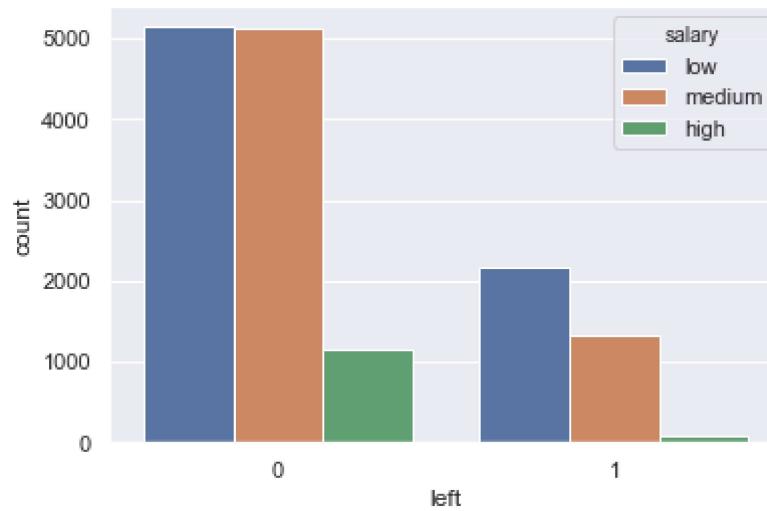
```
In [9]: plt.figure(figsize=(15, 8))
count_left_dept = sns.countplot(x='left', data=df, hue='Department')
for p in count_left_dept.patches:
    count_left_dept.annotate(format(p.get_height()), (p.get_x() + p.get_width()
() / 2., p.get_height()), ha = 'center', va = 'center', xytext = (0, 10), text
coords = 'offset points')
```



Display bar chart for employee status by salary category

```
In [10]: sns.countplot(x='left', data=df, hue='salary')
```

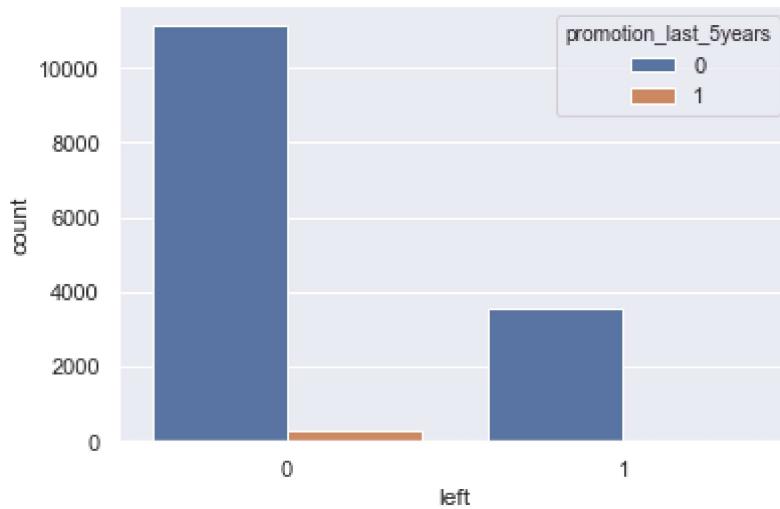
```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1a17f74048>
```



Display bar chart for employee status by whether they had a promotion in the last 5 years

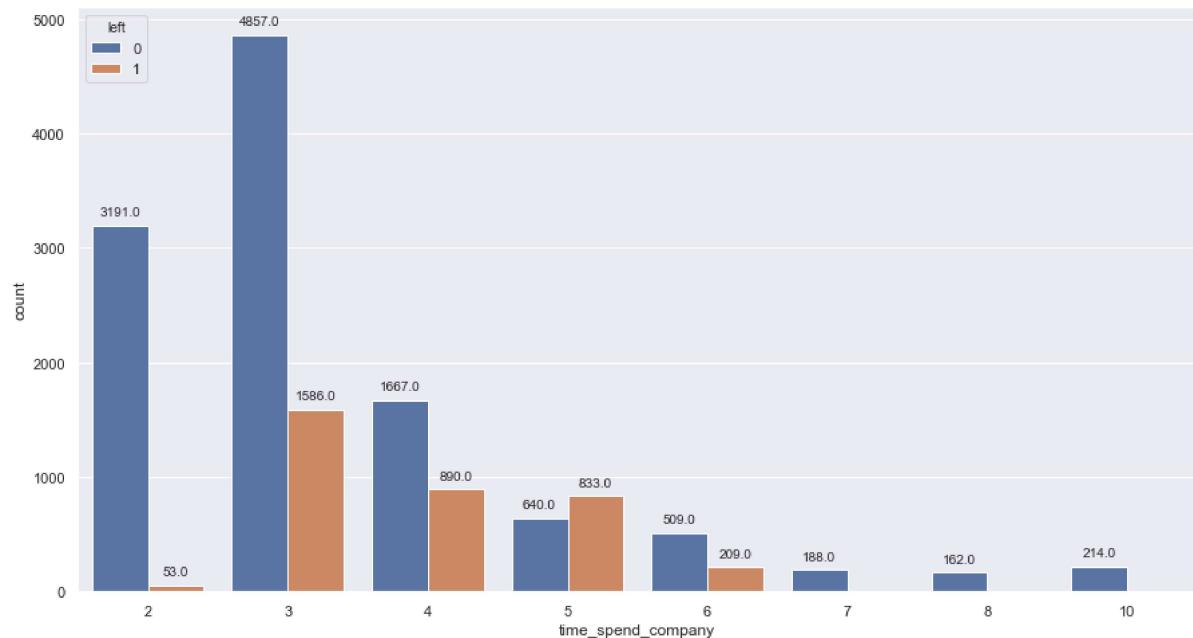
In [11]: `sns.countplot(x='left', data=df, hue='promotion_last_5years')`

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1c471080>



Display bar chart for employee status by the # of years they spent at the company

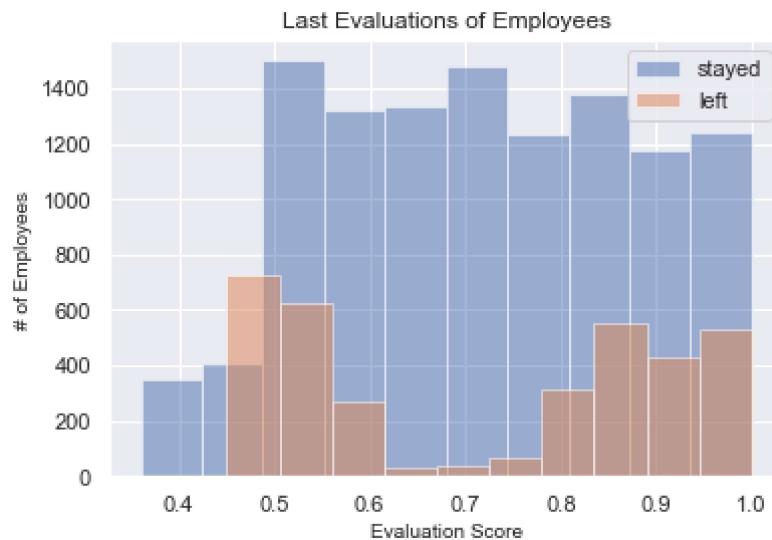
```
In [12]: plt.figure(figsize=(15, 8))
count_time_spent_company = sns.countplot(x='time_spend_company', data=df, hue='left')
for p in count_time_spent_company.patches:
    count_time_spent_company.annotate(format(p.get_height()), (p.get_x() + p.get_width() / 2., p.get_height()), ha = 'center', va = 'center', xytext = (0, 10), textcoords = 'offset points')
```



Display dual histogram of Last Evaluation and employee status

```
In [13]: left_last_eval = df[df.left == 1]
stayed_last_eval = df[df.left == 0]

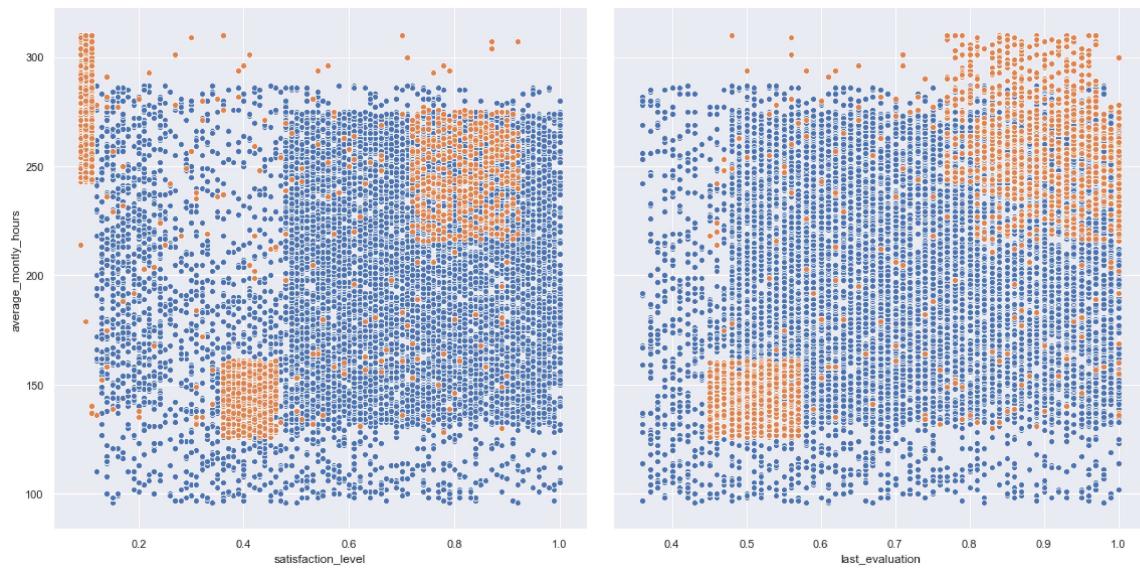
plt.hist(stayed_last_eval['last_evaluation'], label='stayed', bins=10, alpha=0.5)
plt.hist(left_last_eval['last_evaluation'], label='left', bins=10, alpha=0.5)
plt.legend(loc='best')
plt.xlabel('Evaluation Score', fontsize=10)
plt.ylabel('# of Employees', fontsize=10)
plt.title('Last Evaluations of Employees', fontsize=12)
plt.show()
```



Pairplot of Average Monthly Hours by Satisfaction Level and Last Evaluation Score

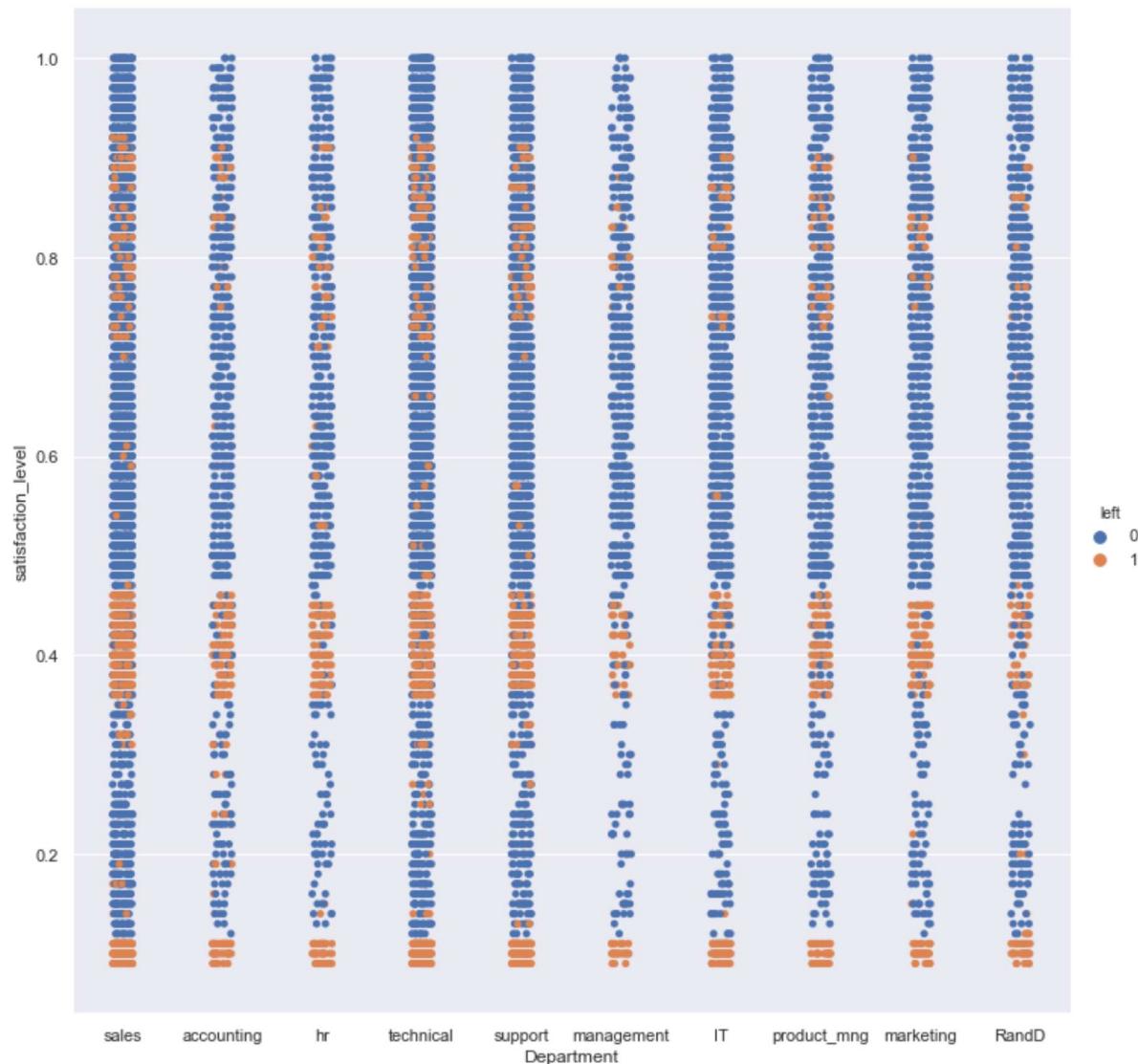
```
In [14]: sns.pairplot(df, x_vars=['satisfaction_level', 'last_evaluation'], y_vars=['average_montly_hours'], hue='left', height=8)
```

```
Out[14]: <seaborn.axisgrid.PairGrid at 0x1a1c5d5828>
```



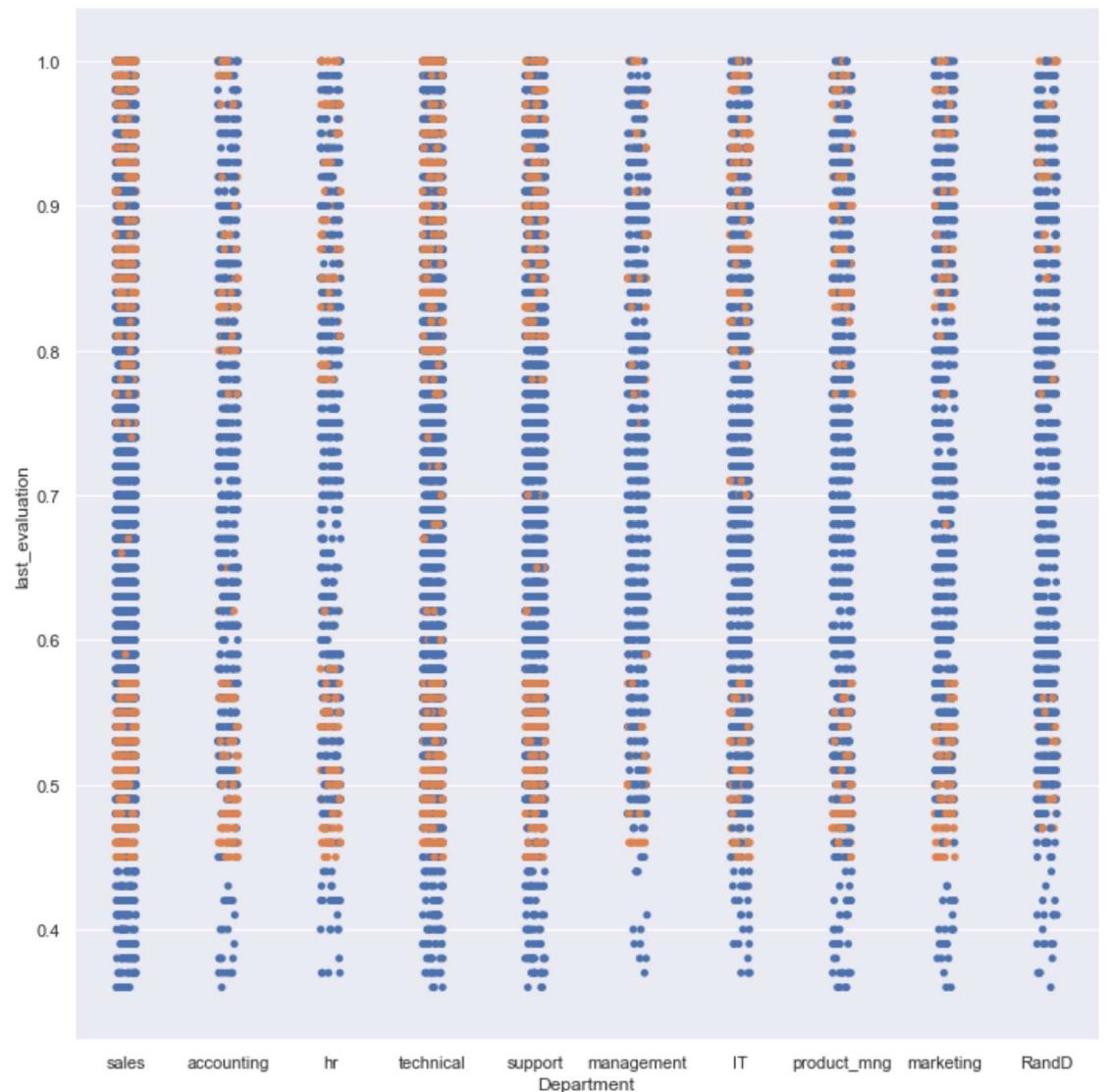
Employee Category Status plot of Satisfaction Level by Department

```
In [15]: sns.catplot(x="Department", y="satisfaction_level", hue="left", data=df, height = 10);
```



Employee Category Status plot of Last Evaluation by Department

```
In [16]: sns.catplot(x="Department", y="last_evaluation", hue="left", data=df, height = 10);
```



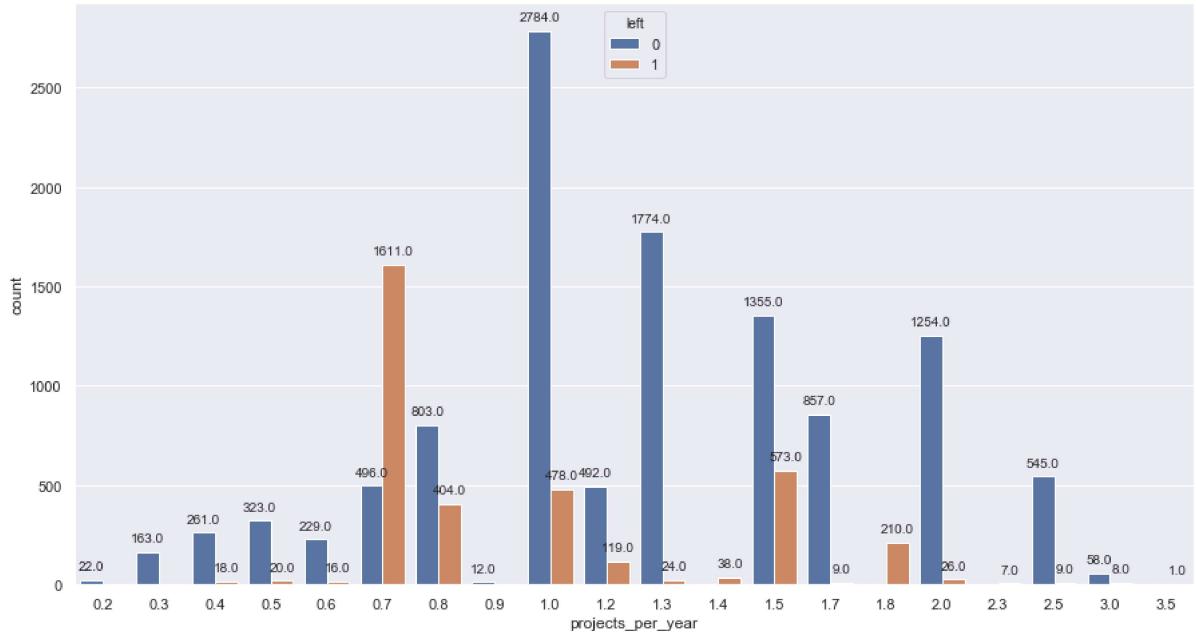
Data Preparation Phase

Construct new derived feature based on number of projects and time spent at company

```
In [17]: df['projects_per_year'] = df.apply(lambda row: round(row.number_project / row.time_spend_company,1), axis=1)
```

Plotting of new derived feature: # of projects per year by employee status

```
In [18]: plt.figure(figsize=(15, 8))
count_project_time_ratio = sns.countplot(x='projects_per_year', data=df, hue='left')
for p in count_project_time_ratio.patches:
    count_project_time_ratio.annotate(format(p.get_height()), (p.get_x() + p.get_width() / 2., p.get_height()), ha = 'center', va = 'center', xytext = (0, 10), textcoords = 'offset points')
```



Modelling Phase

```
In [19]: #Factorize department and salary
```

```
In [20]: #df["DepartmentID"] = pd.factorize(df.Department)[0]
#df["SalaryID"] = pd.factorize(df.salary)[0]
```

Transpose Department column with prefix 'Dept'

```
In [21]: dept = pd.get_dummies(df['Department'], prefix='Dept')
```

Transpose Salary column with prefix 'Salary'

```
In [22]: salary = pd.get_dummies(df['salary'], prefix='Salary')
```

Combine Department and Salary dataframes into new dataframe for training

```
In [23]: df2 = pd.concat([df, dept, salary], axis=1)
```

Drop insignificant features from the new combined dataframe

```
In [24]: df2.drop(['Department', 'salary'], axis=1, inplace=True)
```

```
In [25]: df2.dtypes
```

```
Out[25]: satisfaction_level      float64
last_evaluation                 float64
number_project                  int64
average_montly_hours           int64
time_spend_company              int64
Work_accident                   int64
left                           int64
promotion_last_5years          int64
projects_per_year               float64
Dept_IT                         uint8
Dept_RandD                      uint8
Dept_accounting                 uint8
Dept_hr                          uint8
Dept_management                 uint8
Dept_marketing                  uint8
Dept_product_mng                uint8
Dept_sales                      uint8
Dept_support                     uint8
Dept_technical                  uint8
Salary_high                      uint8
Salary_low                       uint8
Salary_medium                    uint8
dtype: object
```

Training the SVC Model (all features)

```
In [26]: # Split data into 'X' features and 'y' target Label sets
x=df2[['satisfaction_level', 'last_evaluation', 'number_project', 'average_montly_hours', 'time_spend_company',
        'Work_accident', 'promotion_last_5years', 'projects_per_year', 'Dept_IT',
        'Dept_RandD', 'Dept_accounting', 'Dept_hr', 'Dept_marketing', 'Dept_product_mng',
        'Dept_sales',
        'Dept_support', 'Dept_technical', 'Salary_high', 'Salary_low', 'Salary_medium']]
y=df2['left']
```

```
In [27]: # Import module to split dataset
from sklearn.model_selection import train_test_split
# Split data set into training and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=101)
```

```
In [28]: # Import svm model from sklearn
from sklearn import svm
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

# Apply training data to svc model
model_svc = svm.SVC(gamma='auto')
model_svc.fit(x_train,y_train)
```

```
Out[28]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
      decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
      max_iter=-1, probability=False, random_state=None, shrinking=True,
      tol=0.001, verbose=False)
```

Print Model Classification Report and Accuracy Score (all features)

```
In [29]: y_pred = model_svc.predict(x_test)
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.98	0.96	0.97	3431
1	0.88	0.93	0.90	1069
accuracy			0.95	4500
macro avg	0.93	0.95	0.94	4500
weighted avg	0.95	0.95	0.95	4500

```
In [30]: print(model_svc.score(x_test,y_test))
```

0.9528888888888889

Training the SVC Model (4 features)

```
In [31]: # Split data into 'X' features and 'y' target label sets
x2=df2[['satisfaction_level', 'last_evaluation', 'average_montly_hours', 'Dept_IT', 'Dept_RandD', 'Dept_accounting', 'Dept_hr', 'Dept_marketing', 'Dept_product_mng', 'Dept_sales',
         'Dept_support', 'Dept_technical']]
y2=df2['left']
```

```
In [32]: # Import module to split dataset
from sklearn.model_selection import train_test_split
# Split data set into training and test sets
x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, test_size=0.3,
random_state=101)
```

```
In [33]: # Import svm model from sklearn
from sklearn import svm

# Apply training data to svc model
model_svc2 = svm.SVC(gamma='auto')
model_svc2.fit(x2_train,y2_train)
```

```
Out[33]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
              decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
              max_iter=-1, probability=False, random_state=None, shrinking=True,
              tol=0.001, verbose=False)
```

Print Model Classification Report and Accuracy Score (4 features)

```
In [34]: y2_pred = model_svc2.predict(x2_test)
print(classification_report(y2_test, y2_pred))
```

	precision	recall	f1-score	support
0	0.88	0.95	0.91	3431
1	0.78	0.60	0.68	1069
accuracy			0.86	4500
macro avg	0.83	0.77	0.80	4500
weighted avg	0.86	0.86	0.86	4500

```
In [35]: print(model_svc2.score(x2_test,y2_test))
```

```
0.8646666666666667
```

```
In [ ]:
```