

```
In [36]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
In [18]: df = pd.read_csv('https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Employee+Churn+in+Python/HR_comma_sep.csv')
df
```

Out[18]:

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Departments	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
...	...	...	...	...	...	...	...	...	...	...
14994	0.40	0.57	2	151	3	0	1	0	support	low
14995	0.37	0.48	2	160	3	0	1	0	support	low
14996	0.37	0.53	2	143	3	0	1	0	support	low
14997	0.11	0.96	6	280	4	0	1	0	support	low
14998	0.37	0.52	2	158	3	0	1	0	support	low

14999 rows x 10 columns

```
In [19]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  --
0   satisfaction_level      14999 non-null  float64
1   last_evaluation        14999 non-null  float64
2   number_project         14999 non-null  int64
3   average_monthly_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   Departments            14999 non-null  object
9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

```
In [21]: #grouping the data for comparison
left = df.groupby('left')
left.mean()
```

Out[21]:

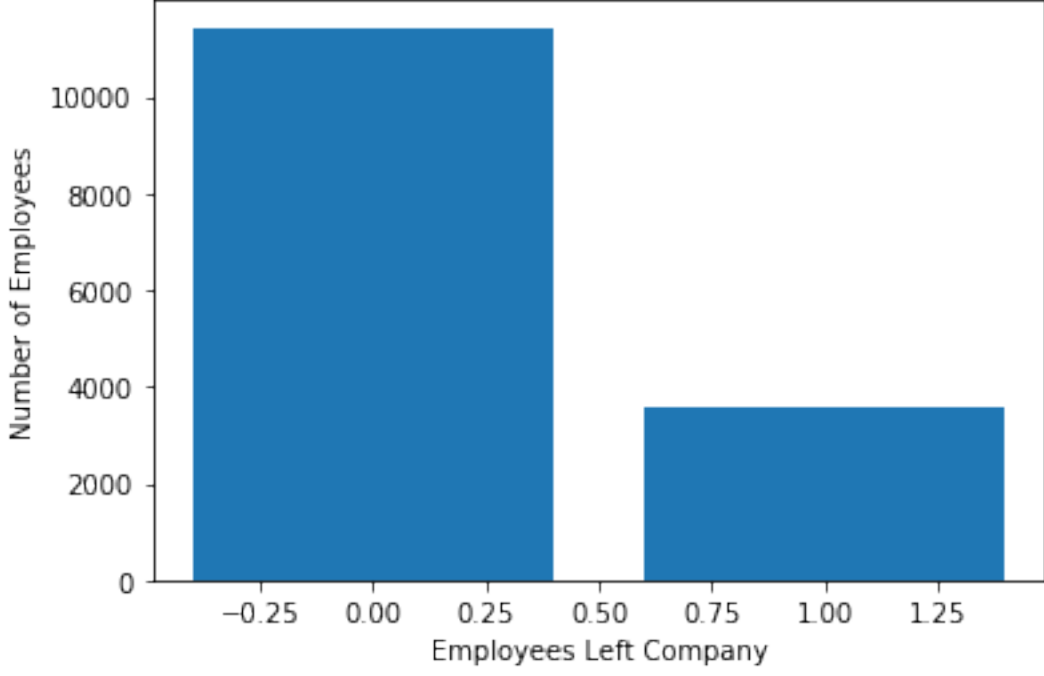
	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	Work_accident	promotion_last_5years
left							
0	0.666810	0.715473	3.786664	199.060203	3.380032	0.175009	0.026251
1	0.440098	0.718113	3.855503	207.419210	3.876505	0.047326	0.005321

```
In [22]: df.describe()
```

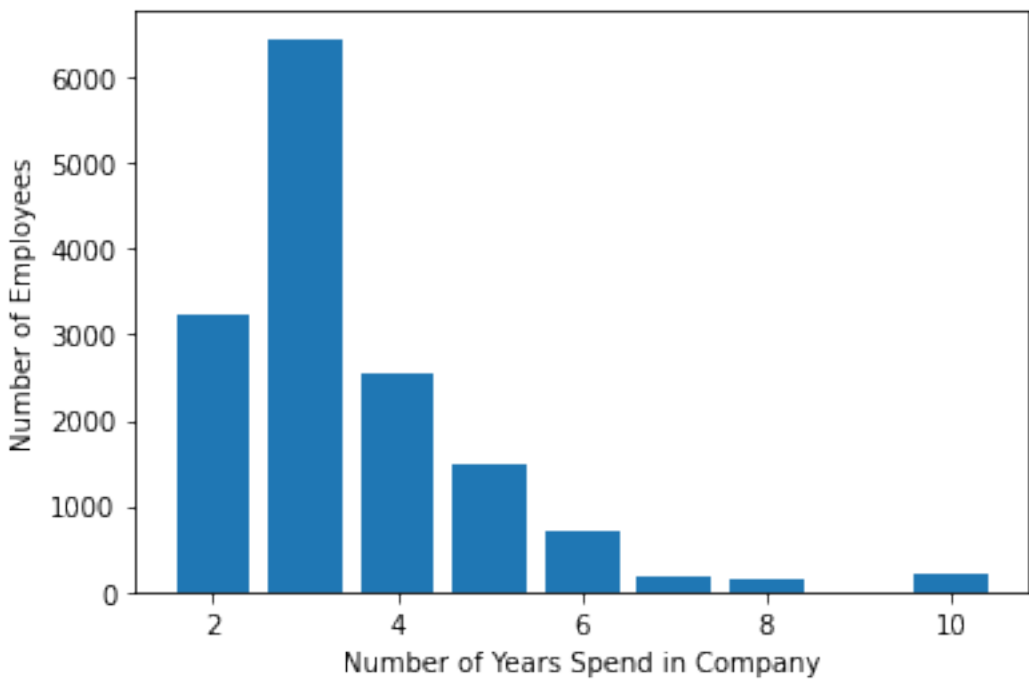
Out[22]:

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	Work_accident	left	promotion_last_5years
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.238083	0.021268
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	0.144281
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	0.000000
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	0.000000
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	0.000000
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	1.000000

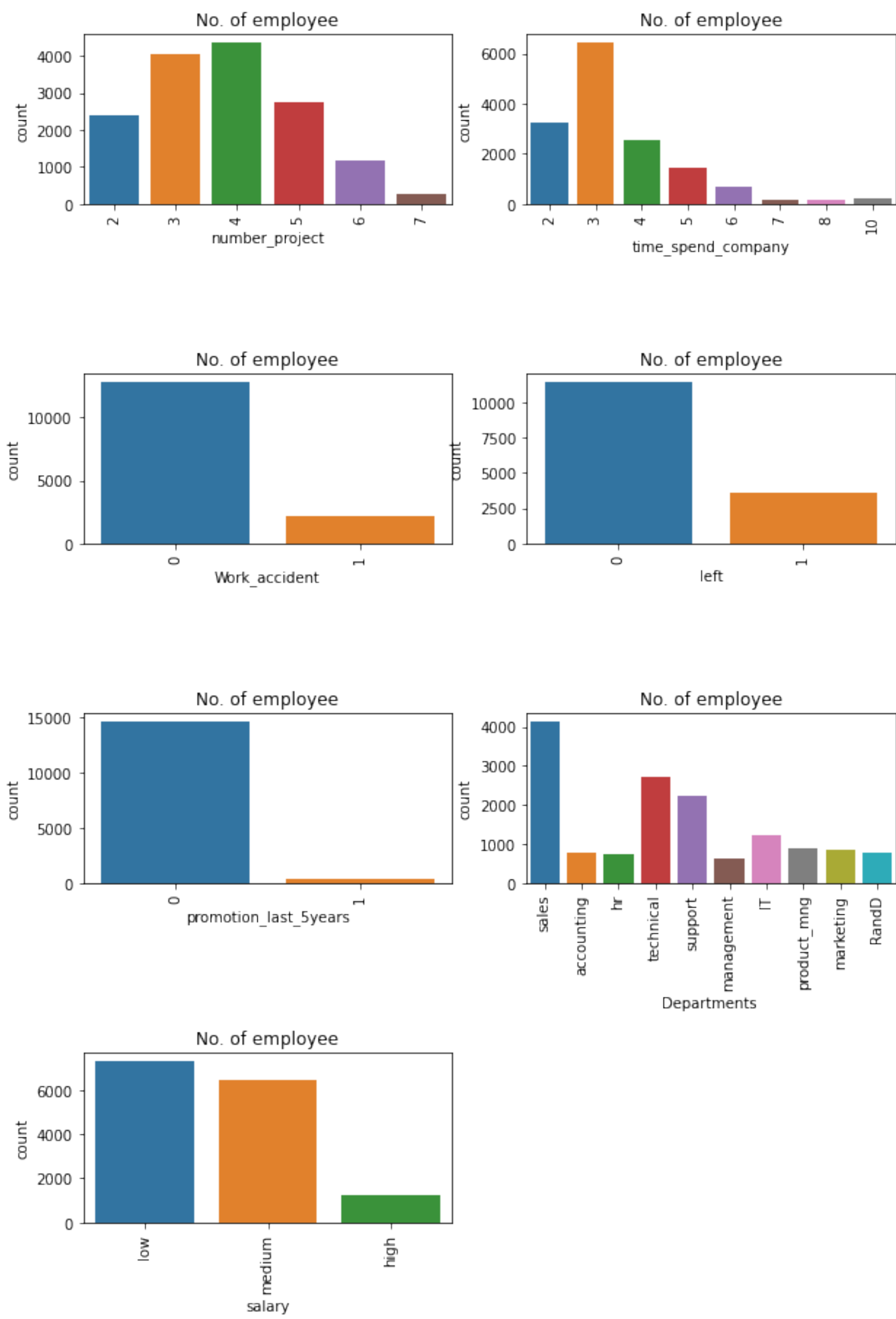
```
In [23]: #Employees who left the company
left_count=df.groupby('left').count()
plt.bar(left_count.index.values,left_count['satisfaction_level'])
plt.xlabel('Employees Left Company')
plt.ylabel('Number of Employees')
plt.show()
```



```
In [24]: time_spend=df.groupby('time_spend_company').count()
plt.bar(time_spend.index.values, time_spend['satisfaction_level'])
plt.xlabel('Number of Years Spend in Company')
plt.ylabel('Number of Employees')
plt.show()
```



```
In [50]: features=['number_project','time_spend_company','Work_accident','left', 'promotion_last_5years','Departments ','salary']
fig=plt.subplots(figsize=(10,15))
for i, j in enumerate(features):
    plt.subplot(4, 2, i+1)
    plt.subplots_adjust(hspace = 1.0)
    sns.countplot(x=j,data = df)
    plt.xticks(rotation=90)
    plt.title("No. of employee")
```



```
In [54]: #import module
from sklearn.cluster import KMeans
# Filter data
left_emp = df[['satisfaction_level','last_evaluation']][df.left == 1]
# Create groups using K-means clustering.
kmeans = KMeans(n_clusters = 3, random_state = 0).fit(left_emp)
```

```
In [58]: # Add new column "label" annd assign cluster labels.
left_emp['label'] = kmeans.labels_
# Draw scatter plot
plt.scatter(left_emp['satisfaction_level'], left_emp['last_evaluation'], c=left_emp['label'],cmap='Accent')
plt.xlabel('Satisfaction Level')
plt.ylabel('Last Evaluation')
plt.title('3 employee Clusters who left')
plt.show()
```



```
In [59]: # Import LabelEncoder
from sklearn import preprocessing
#creating LabelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
df['salary']=le.fit_transform(df['salary'])
df['Departments ']=le.fit_transform(df['Departments '])
```

```
In [60]: #Splitting data into Feature and
x=df[['satisfaction_level', 'last_evaluation', 'number_project',
      'average_monthly_hours', 'time_spend_company', 'Work_accident',
      'promotion_last_5years', 'Departments ', 'salary']]
y=df['left']
```

```
In [61]: # Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=44)
```

```
In [62]: #Import Gradient Boosting Classifier model
from sklearn.ensemble import GradientBoostingClassifier

#Create Gradient Boosting Classifier
gb = GradientBoostingClassifier()

#Train the model using the training sets
gb.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = gb.predict(X_test)
```

```
In [63]: #Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
# Model Precision
print("Precision:",metrics.precision_score(y_test, y_pred))
# Model Recall
print("Recall:",metrics.recall_score(y_test, y_pred))

Accuracy: 0.9775555555555555
Precision: 0.9722497522299306
Recall: 0.9307400379506642
```

## CONCLUSION

Classification rate of 97%, considered as good accuracy. Precision:when a model makes a prediction, how often it is correct. By applying Gradient Boosting model predicted an employee is going to leave, that employee actually left 97% of the time. Recall: If there is an employee who left present in the test set then Gradient Boosting model can identify it 93% of the time.

```
In [ ]:
```