exploratory

June 7, 2023

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[2]:
```
data=pd.read_csv('HR_comma_sep.csv')
data.head()
```

[2]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	sales	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

[3]:
```
data.tail()
```

[3]:

	satisfaction_level	last_evaluation	number_project
14994	0.40	0.57	2
14995	0.37	0.48	2
14996	0.37	0.53	2
14997	0.11	0.96	6
14998	0.37	0.52	2

```
       average_montly_hours  time_spend_company  Work_accident  left  \
14994                   151                   3              0     1
14995                   160                   3              0     1
14996                   143                   3              0     1
14997                   280                   4              0     1
14998                   158                   3              0     1

       promotion_last_5years    sales salary
14994                      0  support    low
14995                      0  support    low
14996                      0  support    low
14997                      0  support    low
14998                      0  support    low
```

[4]:
```
data.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   sales                  14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

[5]:
```
data.describe()
```

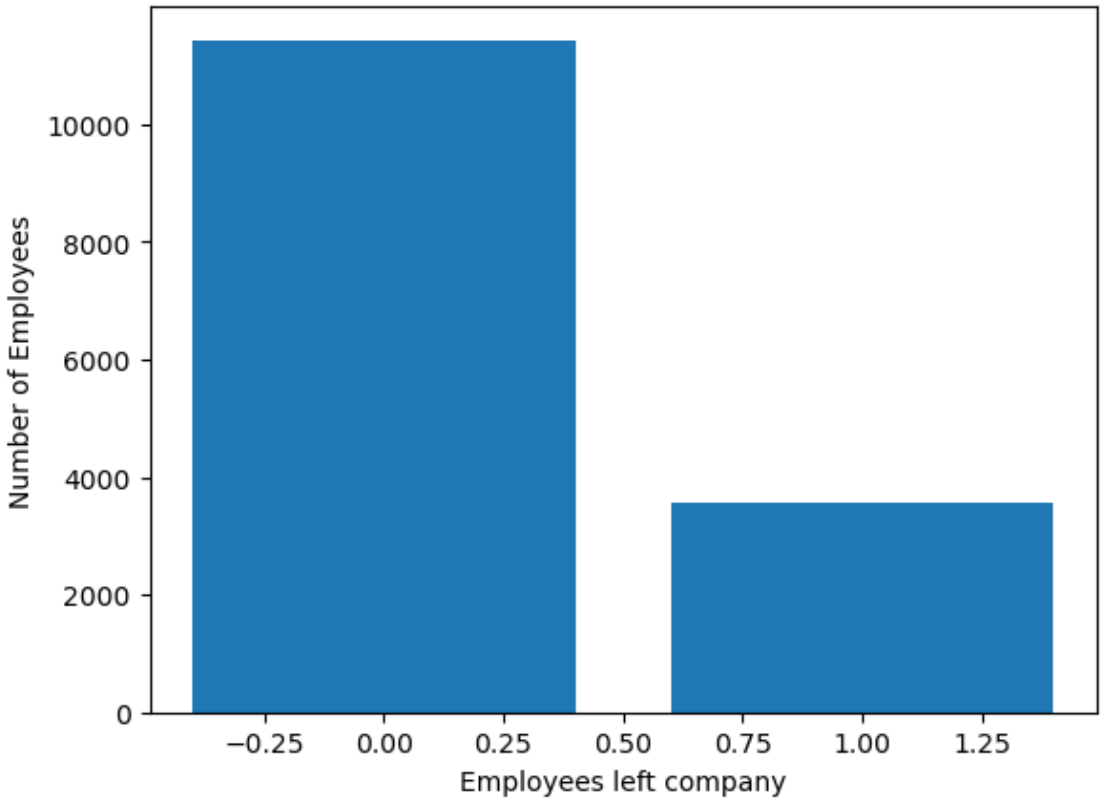
[5]:
```
       satisfaction_level  last_evaluation  number_project  \
count        14999.000000     14999.000000    14999.000000
mean             0.612834         0.716102        3.803054
std              0.248631         0.171169        1.232592
min              0.090000         0.360000        2.000000
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
```

[6]:
```
left=data.groupby('left')
left.mean()
```

[6]:
```
      satisfaction_level  last_evaluation  number_project  \
left
0               0.666810         0.715473        3.786664
1               0.440098         0.718113        3.855503

      average_montly_hours  time_spend_company  Work_accident  \
left
0               199.060203            3.380032       0.175009
1               207.419210            3.876505       0.047326

      promotion_last_5years
left
0                  0.026251
1                  0.005321
```

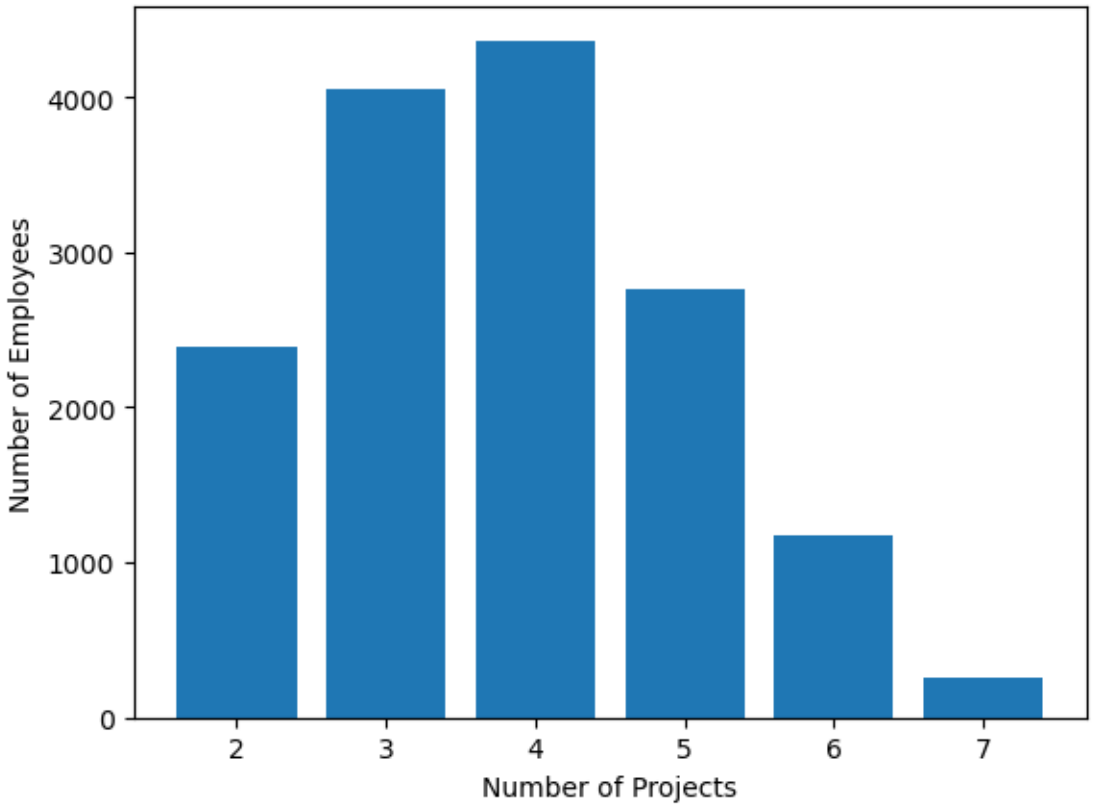
[7]:
```
left_count=data.groupby('left').count()
plt.bar(left_count.index.values,left_count['satisfaction_level'])
plt.xlabel('Employees left company')
plt.ylabel('Number of Employees')
plt.show()
```



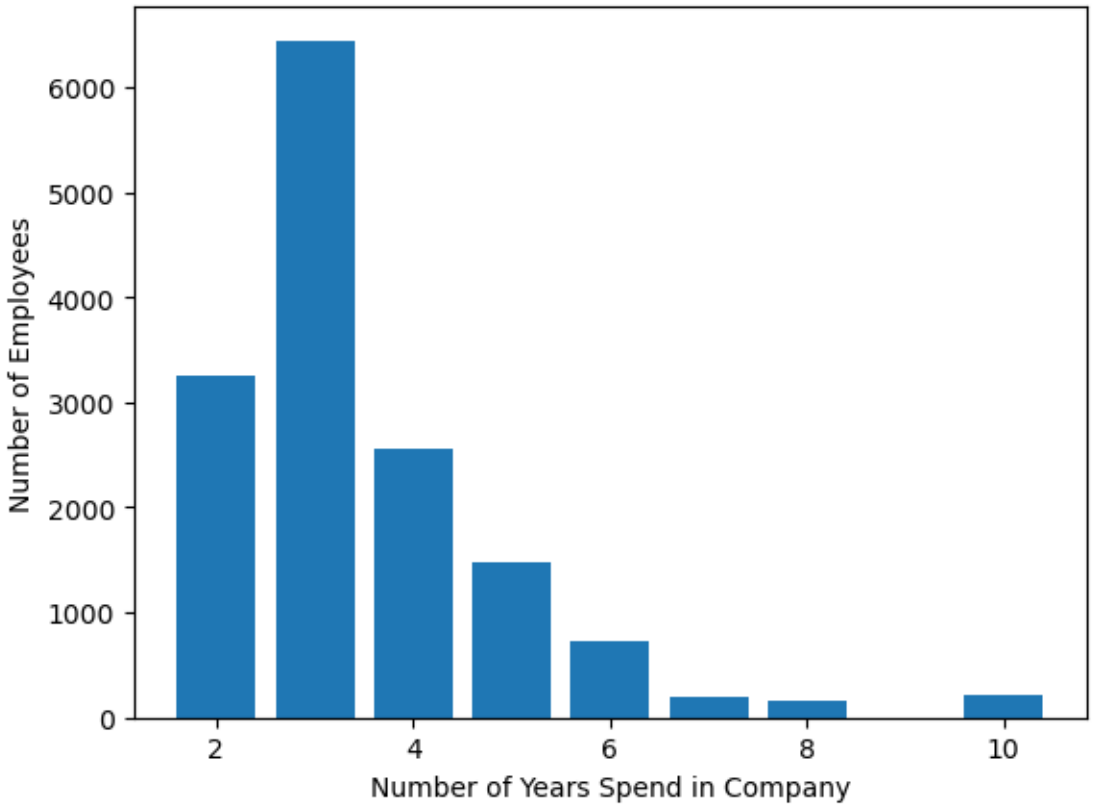
[11]:
```
data.Work_accident.value_counts()
```

[11]:
```
0    12830
1     2169
Name: Work_accident, dtype: int64
```

[12]:
```
num_projects=data.groupby('number_project').count()
plt.bar(num_projects.index.values, num_projects['satisfaction_level'])
plt.xlabel('Number of Projects')
plt.ylabel('Number of Employees')
plt.show()
```



[13]:
```
time_spent=data.groupby('time_spend_company').count()
plt.bar(time_spent.index.values, time_spent['satisfaction_level'])
plt.xlabel('Number of Years Spend in Company')
plt.ylabel('Number of Employees')
plt.show()
```

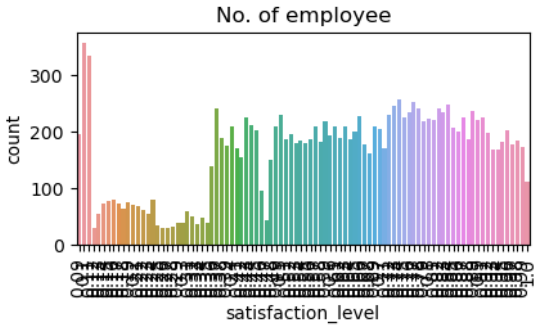


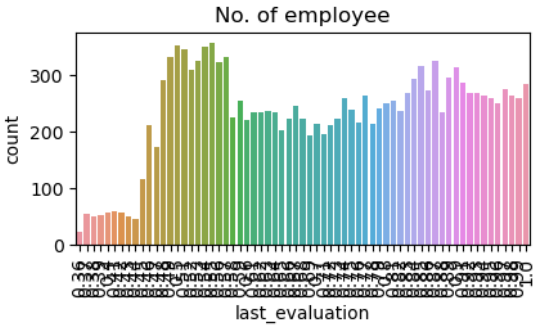
[14]:
```
x1=data
fig=plt.subplots(figsize=(10,15))
for i, j in enumerate(x1):
     plt.subplot(4, 2, i+1)
     plt.subplots_adjust(hspace = 1.0)
     sns.countplot(x=j,data = data)
     plt.xticks(rotation=90)
     plt.title("No. of employee")
```

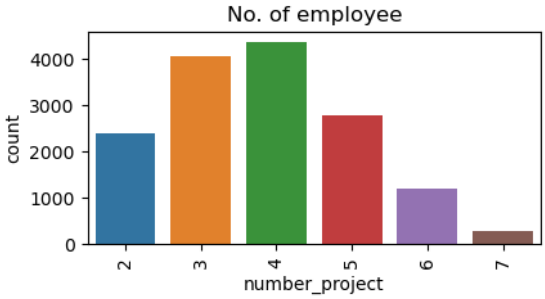
```
C:\Users\pksef\AppData\Local\Temp\ipykernel_13060\3705581917.py:4:
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
ax.remove() as needed.
  plt.subplot(4, 2, i+1)
```

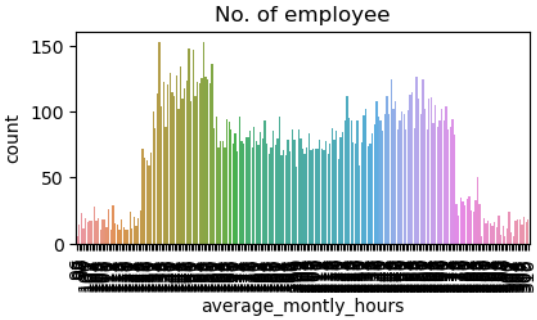
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13060\3705581917.py in <module>
      2 fig=plt.subplots(figsize=(10,15))
      3 for i, j in enumerate(x1):
----> 4      plt.subplot(4, 2, i+1)
```

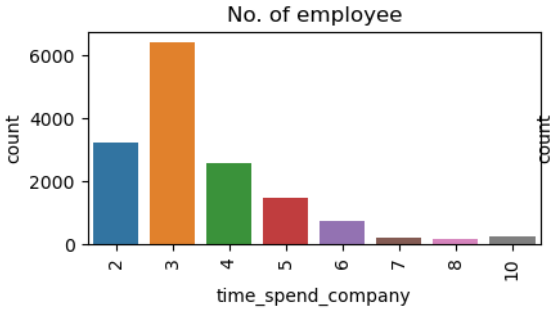
```
      5       plt.subplots_adjust(hspace = 1.0)
      6       sns.countplot(x=j,data = data)

~\anaconda3\lib\site-packages\matplotlib\pyplot.py in subplot(*args, **kwargs)
   1262
   1263     # First, search for an existing subplot with a matching spec.
-> 1264     key = SubplotSpec._from_subplot_args(fig, args)
   1265
   1266     for ax in fig.axes:

~\anaconda3\lib\site-packages\matplotlib\gridspec.py in␣
 ↪_from_subplot_args(figure, args)
    610         else:
    611             if not isinstance(num, Integral) or num < 1 or num >␣
 ↪rows*cols:
--> 612                 raise ValueError(
    613                     f"num must be 1 <= num <= {rows*cols}, not {num!r}"
    614             i = j = num

ValueError: num must be 1 <= num <= 8, not 9
```


No. of employee
count
satisfaction_level

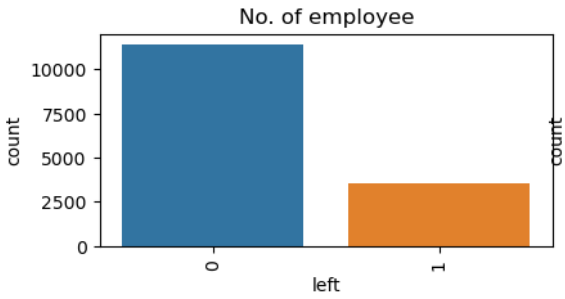

No. of employee
count
last_evaluation

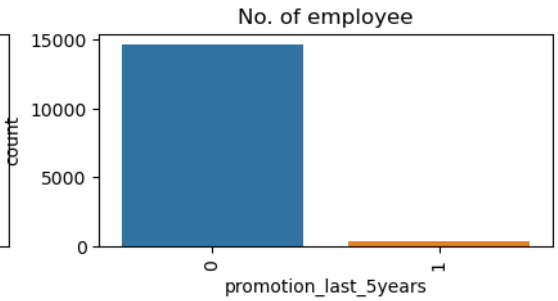

No. of employee
count
number_project


No. of employee
count
average_montly_hours


No. of employee
count
time_spend_company


No. of employee
count
Work_accident


No. of employee
count
left


No. of employee
count
promotion_last_5years

[15]:
```
#import module
from sklearn.cluster import KMeans

# Filter data
left_emp =  data[['satisfaction_level', 'last_evaluation']][data.left == 1]

# Create groups using K-means clustering.
kmeans = KMeans(n_clusters = 3, random_state = 0).fit(left_emp)

# 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 Clusters of employees who left')
plt.show()
```

[19]:
```
# Import LabelEncoder
from sklearn import preprocessing

# Creating labelEncoder
le = preprocessing.LabelEncoder()

# Converting string labels into numbers.
data['salary']=le.fit_transform(data['salary'])
data['sales']=le.fit_transform(data['sales'])
```

[20]:
```
# Spliting data into Feature and
X=data[['satisfaction_level', 'last_evaluation', 'number_project',␣
 ↪'average_montly_hours', 'time_spend_company', 'Work_accident',␣
 ↪'promotion_last_5years', 'sales', 'salary']]
y=data['left']
# 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=42)  # 70% training and 30% test
```

[21]:
```
#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)
```

[22]:
```
# 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: 0.9715555555555555
```

```
Precision: 0.958252427184466
Recall: 0.9207089552238806
```

[ ]: