Employee Turnover Analytics

January 30, 2023

0.1 Import Libraries

[1]:
```
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
```

0.1.1 Load the Dataset

[2]:
```
df = pd.read_csv('Turnover.csv')
df.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	
3	0.72	0.87	5	223	5	0	1	0	sales	
4	0.37	0.52	2	159	3	0	1	0	sales	

```
2  medium
3     low
4     low
```

[3]:
```
df=df.rename(columns={'average_montly_hours':'average_weekly_hours','sales':
 ↪'department'})
df['average_weekly_hours']=df['average_weekly_hours']*12/52
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_weekly_hours   14999 non-null  float64
 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(3), int64(5), object(2)
memory usage: 1.1+ MB
```

[4]:
```
print (np.corrcoef(df['number_project'], df['average_weekly_hours']))
```

```
[[1.         0.41721063]
 [0.41721063 1.        ]]
```

[5]:
```
df.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_weekly_hours	time_spend_company	Work_accident	left	\
count	14999.000000	14999.000000	14999.000000	14999.000000	
mean	46.396232	3.498233	0.144610	0.238083	

```
std               11.525331            1.460136       0.351719       0.425924
min               22.153846            2.000000       0.000000       0.000000
25%               36.000000            3.000000       0.000000       0.000000
50%               46.153846            3.000000       0.000000       0.000000
75%               56.538462            4.000000       0.000000       0.000000
max               71.538462           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]:
```
df.describe(include=['O'])
```

[6]:
```
       department salary
count       14999  14999
unique         10      3
top         sales    low
freq         4140   7316
```

0.1.2 Feature Selection

[7]:
```
df[['Work_accident', 'left']].groupby(['Work_accident'], as_index=False).mean().
↪sort_values(by='left')
```

[7]:
```
   Work_accident      left
1              1  0.077916
0              0  0.265160
```

[8]:
```
df[['department', 'left']].groupby(['department'], as_index=False).mean().
↪sort_values(by='left', ascending=False)
```

[8]:
```
    department      left
3           hr  0.290934
2   accounting  0.265971
9    technical  0.256250
8      support  0.248991
7        sales  0.244928
5    marketing  0.236597
0           IT  0.222494
6  product_mng  0.219512
```

```
1         RandD  0.153748
4    management  0.144444
```

[9]:
```
df[['salary', 'left']].groupby(['salary'], as_index=False).mean().
↪sort_values(by='left', ascending=False)
```

[9]:
```
    salary      left
1      low  0.296884
2   medium  0.204313
0     high  0.066289
```

[10]:
```
df[['number_project', 'left']].groupby(['number_project'], as_index=False).
↪mean().sort_values(by='number_project')
```

[10]:
```
    number_project      left
0                2  0.656198
1                3  0.017756
2                4  0.093700
3                5  0.221659
4                6  0.557922
5                7  1.000000
```

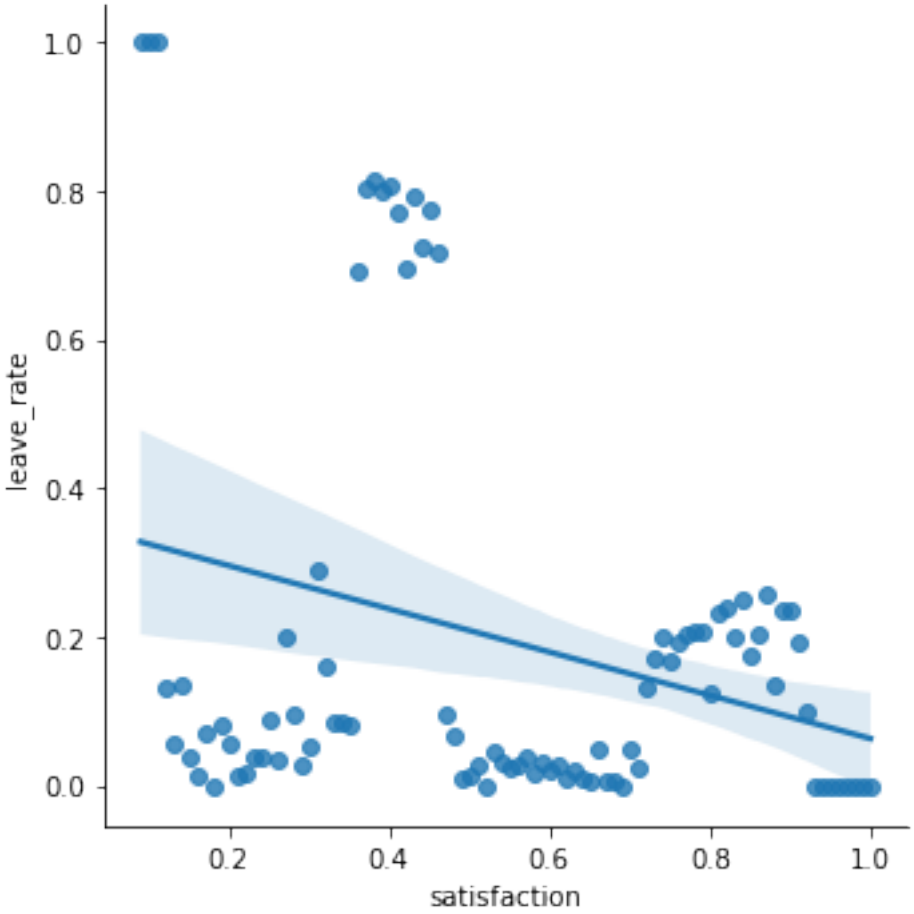
[11]:
```
df[['time_spend_company', 'left']].groupby(['time_spend_company'],␣
↪as_index=False).mean().sort_values(by='time_spend_company')
```

[11]:
```
    time_spend_company      left
0                    2  0.016338
1                    3  0.246159
2                    4  0.348064
3                    5  0.565513
4                    6  0.291086
5                    7  0.000000
6                    8  0.000000
7                   10  0.000000
```

[12]:
```
leave_sat=df.groupby('satisfaction_level').agg({'left': lambda x: len(x[x==1])})
leave_sat['total']=df.groupby('satisfaction_level').agg({'left': len})
leave_sat['leave_rate']=leave_sat['left']/leave_sat['total']
leave_sat['satisfaction']=df.groupby('satisfaction_level').
↪agg({'satisfaction_level': 'mean'})
g=sns.lmplot('satisfaction', 'leave_rate',data=leave_sat)
plt.show()
```

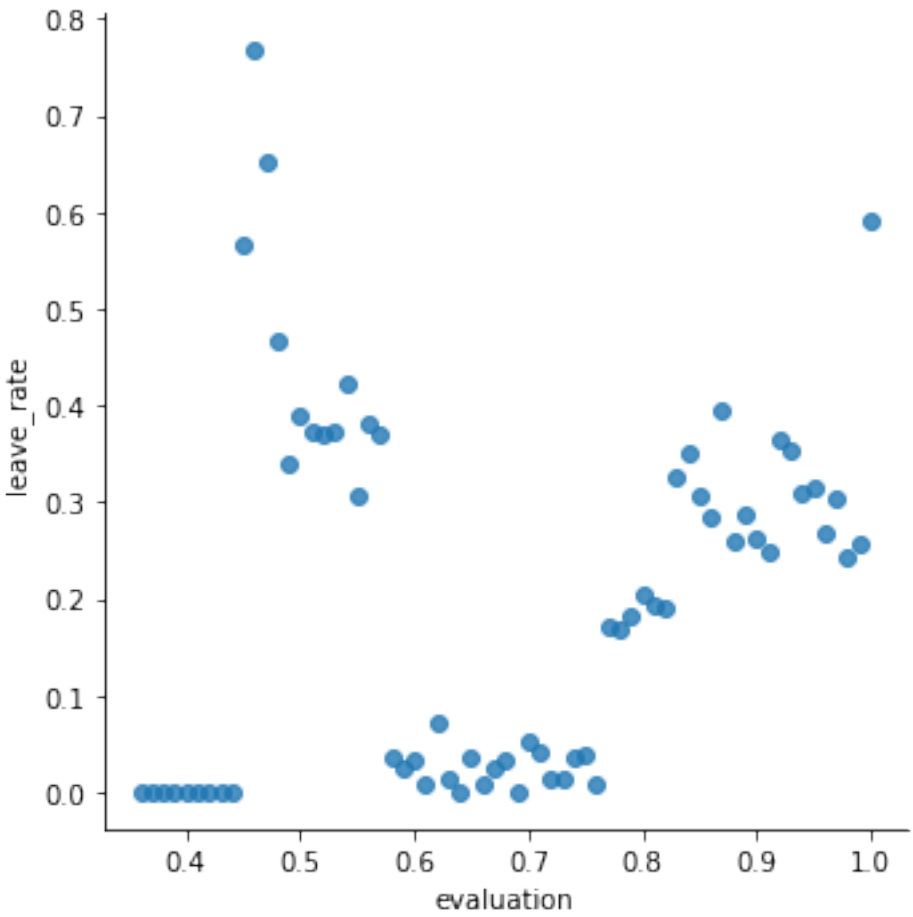
```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
```

```
  FutureWarning
```



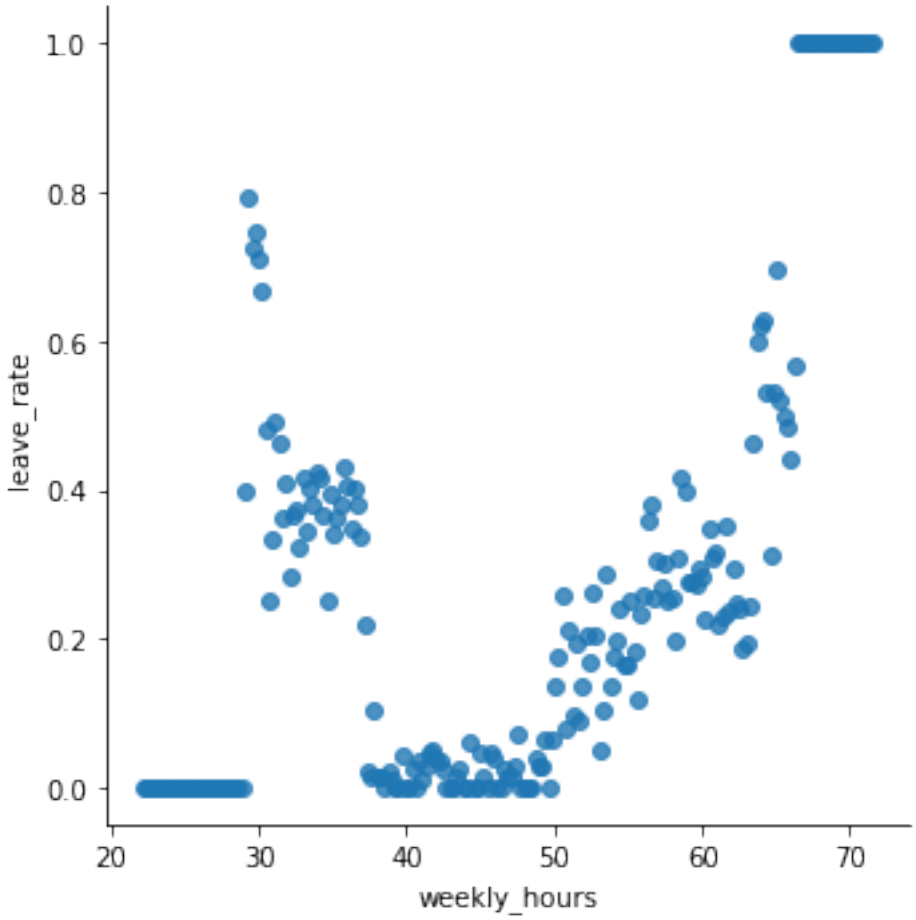
[13]:
```
leave_eval=df.groupby('last_evaluation').agg({'left': lambda x: len(x[x==1])})
leave_eval['total']=df.groupby('last_evaluation').agg({'left': len})
leave_eval['leave_rate']=leave_eval['left']/leave_eval['total']
leave_eval['evaluation']=df.groupby('last_evaluation').agg({'last_evaluation':␣
↪'mean'})
gr=sns.lmplot('evaluation', 'leave_rate',data=leave_eval,fit_reg=False)
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



[14]:
```
leave_hours=df.groupby('average_weekly_hours').agg({'left': lambda x:␣
 ↪len(x[x==1])})
leave_hours['total']=df.groupby('average_weekly_hours').agg({'left': len})
leave_hours['leave_rate']=leave_hours['left']/leave_hours['total']
leave_hours['weekly_hours']=df.groupby('average_weekly_hours').
 ↪agg({'average_weekly_hours': 'mean'})
grid=sns.lmplot('weekly_hours', 'leave_rate',data=leave_hours,fit_reg=False)
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



[15]:
```
df[['department', 'average_weekly_hours']].groupby(['department'],␣
↪as_index=False).mean().sort_values(by='average_weekly_hours',␣
↪ascending=False)
```

[15]:

	department	average_weekly_hours
9	technical	46.730175
0	IT	46.665225
4	management	46.442125
2	accounting	46.422224
7	sales	46.364158
1	RandD	46.338579
8	support	46.328813
6	product_mng	46.145915
5	marketing	46.012103
3	hr	45.850317

[16]:
```
(df.promotion_last_5years==1).sum()
df=df.drop(['promotion_last_5years'],axis=1)
```

0.1.3 Data Wrangling

[17]:
```
df=df.drop(['Work_accident','department','average_weekly_hours'],axis=1)
df.columns
```

[17]:
```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'time_spend_company', 'left', 'salary'],
      dtype='object')
```

[18]:
```
bins=[0,2,5,10]
names=[1,0,1]
df['abnormal_proj']=pd.cut(df['number_project'],bins,names,labels=False)
#banding years at the firm
bins2=[0,1,2,3,4,5,6,100]
names2=['1','2','3','4','5','6','7']
df['years_at_company']=pd.
 ↪cut(df['time_spend_company'],bins2,names2,labels=False)
#banding last_evaluation
bins3=[0,.6,.8,1]
names3=[1,0,1]
df['abnormal_eval']=pd.cut(df['last_evaluation'],bins3,names3,labels=False)
df.head()
```

[18]:

	satisfaction_level	last_evaluation	number_project	time_spend_company	left	salary	abnormal_proj	years_at_company	abnormal_eval
0	0.38	0.53	2	3	1	low	0	2	0
1	0.80	0.86	5	6	1	medium	1	5	2
2	0.11	0.88	7	4	1	medium	2	3	2
3	0.72	0.87	5	5	1	low	1	4	2
4	0.37	0.52	2	3	1	low	0	2	0

[19]:
```
df=df.drop(['number_project','time_spend_company','last_evaluation'],axis=1)
df.head()
```

[19]:

	satisfaction_level	left	salary	abnormal_proj	years_at_company	abnormal_eval
0	0.38	1	low	0	2	0
1	0.80	1	medium	1	5	2
2	0.11	1	medium	2	3	2
3	0.72	1	low	1	4	2
4	0.37	1	low	0	2	0

```
   abnormal_eval
0              0
1              2
2              2
3              2
4              0
```

[20]:
```
df['salary']=df['salary'].map({'low':0,'medium':1,'high':2}).astype(int)
pd.to_numeric(df['abnormal_proj'], errors='coerce')
pd.to_numeric(df['years_at_company'], errors='coerce')
pd.to_numeric(df['abnormal_eval'], errors='coerce')
df.head()
```

[20]:
```
   satisfaction_level  left  salary  abnormal_proj  years_at_company  \
0                0.38     1       0              0                 2
1                0.80     1       1              1                 5
2                0.11     1       1              2                 3
3                0.72     1       0              1                 4
4                0.37     1       0              0                 2

   abnormal_eval
0              0
1              2
2              2
3              2
4              0
```

0.1.4 Split the dataset into train and test data

[21]:
```
## Modeling
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(df,df['left'],test_size=.2)
X_train=X_train.drop('left',axis=1)
X_test=X_test.drop('left',axis=1)
print (X_train.shape, Y_train.shape)
print (X_test.shape, Y_test.shape)
```

```
(11999, 5) (11999,)
(3000, 5) (3000,)
```

[22]:
```
## Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
Y_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
acc_log
```

[22]:
```
79.13
```

[23]:
```
coeff_df = pd.DataFrame(X_train.columns)
coeff_df.columns = ['Feature']
coeff_df["Coefficient"] = pd.Series(logreg.coef_[0])

coeff_df.sort_values(by='Coefficient', ascending=False)
```

[23]:
```
              Feature  Coefficient
3    years_at_company     0.420547
4       abnormal_eval     0.395897
1              salary    -0.744835
2       abnormal_proj    -1.377905
0  satisfaction_level    -4.521359
```

[24]:
```
#KNN
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
acc_knn
```

[24]:
```
97.92
```

[25]:
```
svc = SVC()
svc.fit(X_train, Y_train)
Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_train, Y_train) * 100, 2)
acc_svc
```

[25]:
```
95.36
```

Naive Bayes Algorithm

[26]:
```
gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
acc_gaussian
```

[26]:
```
83.83
```

Decision Tree

[27]:
```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_test)
```

```
acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)
acc_decision_tree
```

[27]:
```
98.66
```

[28]:
```
import os
os.environ["PATH"] += os.pathsep + 'D:/Program Files (x86)/Graphviz2.38/bin/'
```

[29]:
```
from sklearn import tree
import graphviz
import os
os.environ["PATH"] += os.pathsep + 'D:/Program Files (x86)/Graphviz2.38/bin/'
dot_data=tree.export_graphviz(decision_tree,out_file=None, max_depth=3)
graph=graphviz.Source
graph
```

[29]:
```
graphviz.files.Source
```

[30]:
```
models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Naive Bayes', 'Decision Tree'],
    'Score': [acc_svc, acc_knn, acc_log,
              acc_gaussian,acc_decision_tree]})
models.sort_values(by='Score', ascending=False)
```

[30]:

	Model	Score
4	Decision Tree	98.66
1	KNN	97.92
0	Support Vector Machines	95.36
3	Naive Bayes	83.83
2	Logistic Regression	79.13

[ ]:
```

```