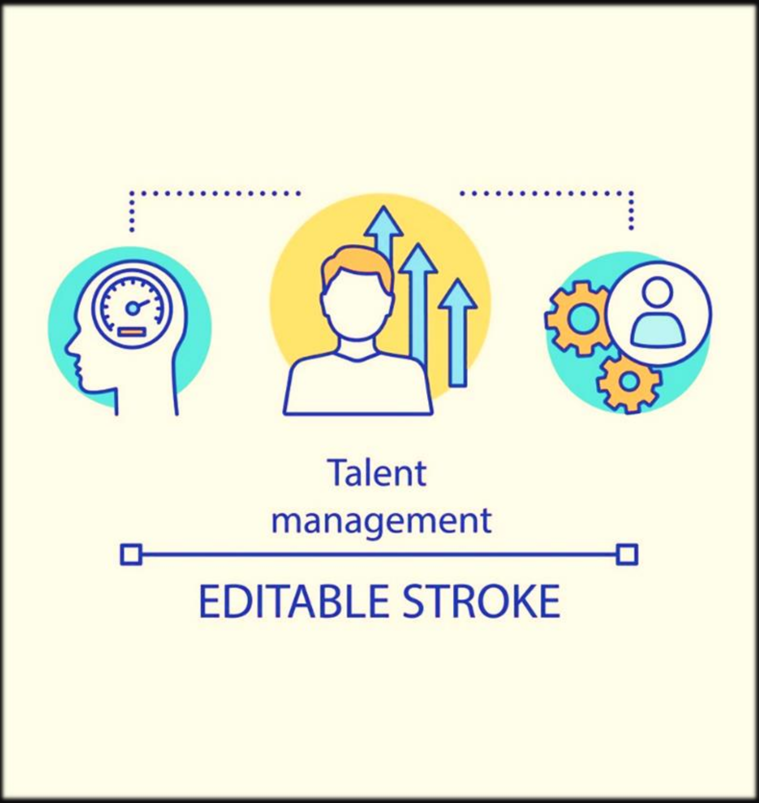
HR Data Analytics & a Comparative Algorithm Study

Sourav Pattanayak



Introduction

In this project, we delve into the realm of Human Resources (HR) data analysis, leveraging machine learning techniques to gain insights and make informed decisions. The HR department plays a pivotal role in any organization, tasked with managing personnel, assessing performance, and fostering employee growth. However, with vast amounts of data at their disposal, extracting meaningful insights can be a difficult task.

Our project aims to address this challenge by applying machine learning algorithms to HR data, specifically focusing on predicting employee performance ratings. By analyzing various features such as employee demographics, department, job role, and tenure, we seek to understand the factors influencing performance and identify patterns that contribute to success.

Additionally, we complement our machine learning approach with Python analysis to uncover deeper insights into the workforce. Through Python analysis, we explore trends, correlations, and other valuable information about employee demographics, department dynamics, promotion trends, and more.

Furthermore, I have conducted a comparative analysis of different machine learning algorithms, such as Logistic Regression and Random Forest Classifier, to evaluate their performance in predicting employee performance ratings. By comparing the accuracy and effectiveness of these algorithms, I aim to provide HR professionals with actionable insights for making better decisions and improving organizational outcomes.

By this, we not only aim to optimize HR processes and decision-making but also to empower organizations to cultivate a thriving and productive workforce.

HR Data Analysis and Comparative Algorithm Study

[3]:
```
from IPython.core.display import display, HTML
display(HTML("<style>.cm-s-ipython span.cm-comment { color: yellow; }</style>"))
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_15832\4045986198.py:1: DeprecationWarning: Importing
display from IPython.core.display is deprecated since IPython 7.14, please import from IPython display
  from IPython.core.display import display, HTML
```

```
<IPython.core.display.HTML object>
```

[1]:
```
#Importing the necessary libraries:
import pandas as pd import
numpy as np
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt import
seaborn as sns
```

1.0.1 TASK: Our task is to predict the Performance Rating of the employees (based on the 'features"), which forms our target variable.

[2]:
```
#Import the excel file
df=pd.read_excel('Hr_data.xls') df
```

[2]:
```
       EmpNumber  Age  Gender  EducationBackground  MaritalStatus   \
0      E1001000   32    Male            Marketing          Single
1      E1001006   47    Male            Marketing          Single
2      E1001007   40    Male        Life Sciences         Married
3      E1001009   41    Male      Human Resources        Divorced
4      E1001010   60    Male            Marketing          Single
...         ...  ...     ...                  ...             ...
1195    E100992   27  Female              Medical        Divorced
1196    E100993   37    Male        Life Sciences          Single
1197    E100994   50    Male              Medical         Married
1198    E100995   34  Female              Medical          Single
1199    E100998   24  Female        Life Sciences          Single
          EmpDepartment       EmpJobRole  BusinessTravelFrequency   \
0                 Sales  Sales Executive            Travel_Rarely
1                 Sales  Sales Executive            Travel_Rarely
2                 Sales  Sales Executive        Travel_Frequently
3       Human Resources          Manager            Travel_Rarely
4                 Sales  Sales Executive            Travel_Rarely
...                 ...              ...                      ...
1195              Sales  Sales Executive        Travel_Frequently
```

```
1196        Development     Senior Developer            Travel_Rarely
1197        Development     Senior Developer            Travel_Rarely
1198       Data Science       Data Scientist            Travel_Rarely
1199              Sales      Sales Executive            Travel_Rarely

      DistanceFromHome   EmpEducationLevel   ...  EmpRelationshipSatisfaction      \
0                   10                   3   ...                            4
1                   14                   4   ...                            4
2                    5                   4   ...                            3
3                   10                   4   ...                            2
4                   16                   4   ...                            4
...                ...                 ... ...                    ...
1195                 3                   1   ...                            2
1196                10                   2   ...                            1
1197                28                   1   ...                            3
1198                 9                   3   ...                            2
1199                 3                   2   ...                            1

      TotalWorkExperienceInYears        TrainingTimesLastYear   EmpWorkLifeBalance  \
0                             10                            2                    2
1                             20                            2                    3
2                             20                            2                    3
3                             23                            2                    2
4                             10                            1                    3
...                          ...                          ...                 ...
1195                           6                            3                    3
1196                           4                            2                    3
1197                          20                            3                    3
1198                           9                            3                    4
1199                           4                            3                    3

          ExperienceYearsAtThisCompany   ExperienceYearsInCurrentRole  \
0                                   10                              7
1                                    7                              7
2                                   18                             13
3                                   21                              6
4                                    2                              2
...                                ...                            ...
```

```
1195                          6                          5
1196                          1                          0
1197                         20                          8
1198                          8                          7
1199                          2                          2

      YearsSinceLastPromotion   YearsWithCurrManager   Attrition  \
0                           0                      8          No
1                           1                      7          No
2                           1                     12          No
3                          12                      6          No
4                           2                      2          No
...                       ...                    ...        ...
1195                        0                      4          No
1196                        0                      0          No
1197                        3                      8          No
1198                        7                      7          No
1199                        2                      0         Yes

         PerformanceRating
0                        3
1                        3
2                        4
3                        3
4                        3
...                    ...
1195                     4
1196                     3
1197                     3
1198                     3
1199                     2

[1200 rows x 28 columns]
```

Checking Null/NaN values

[4]:
```
#Is there any Null/NaN Value?

df.isnull().values.any()

# No Null values.
```

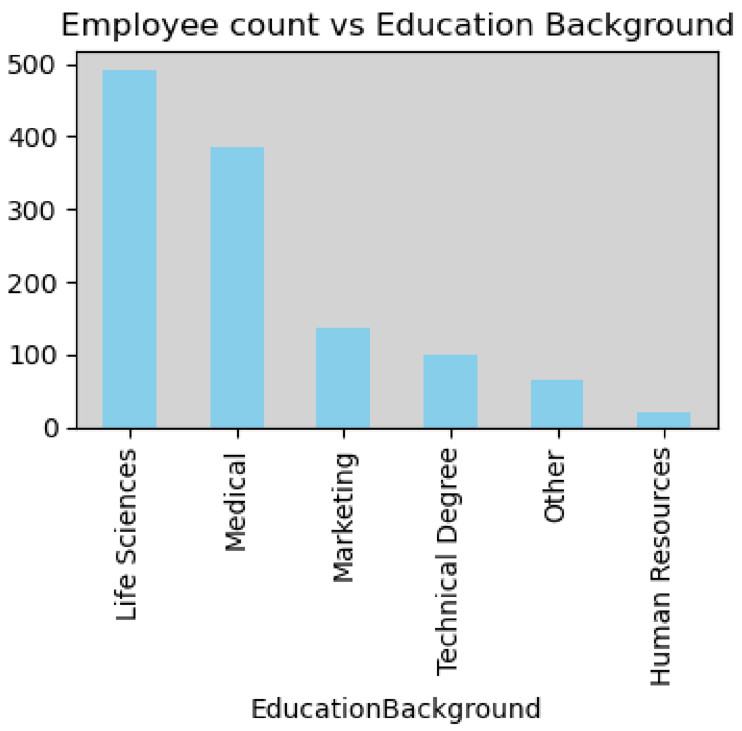
[4] :
```
False
```

1.0.2 –General Analysis:

I. Analysing employees with their Education background:

[5] :
```
education_counts = df['EducationBackground'].value_counts()
```

[6] :
```
plt.figure(figsize=(4, 4)) education_counts.plot(kind='bar',
color='skyblue') plt.title('Employee count vs Education Background')
plt.tight_layout() plt.gca().set_facecolor('lightgrey')
plt.show()
```



Conclusion

1. Life Sciences education background of the greatest number of employees, followed by Medical.
2. Human Resources has the lowest number of employees in it.

II. Analysing employees with their Departments:

[7] :
```
df['EmpDepartment'].value_counts()
```

[7] :
```
EmpDepartment
Sales                     373
Development               361
Research & Development    343
Human Resources            54
Finance                    49
```

```
Data Science                         20
Name: count, dtype: int64
```

[ ]:

```

```

Conclusion:

We can assume that the Sales department has the largest number of workforce, and Data Science the lowest.

III. Analysing department-wise performance rating of the employees:

[8] :

```
df.groupby('EmpDepartment').PerformanceRating.mean()
```

[8] :

```
EmpDepartment
 Data Science                     3.050000
 Development                      3.085873
 Finance                          2.775510
 Human Resources                  2.925926
 Research & Development           2.921283
 Sales                            2.860590
 Name: PerformanceRating,       dtype: float64
```

Conclusion:

1. Performance ratings for employees in the Data Science department is highest.

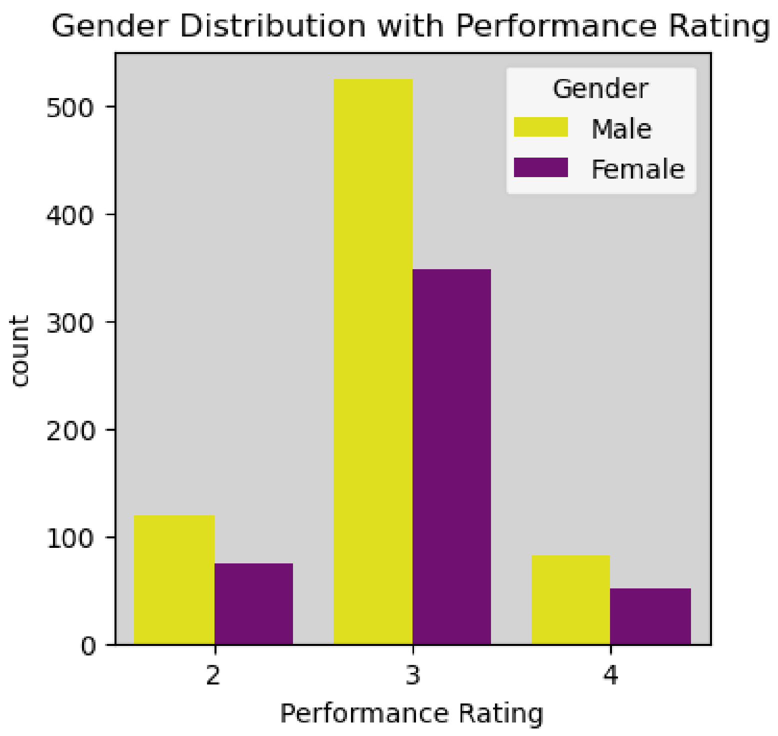
2.Performance ratings for employees in the Sales department is the lowest.

IV. Analysing Employee Gender Distribution with Performance Rating:

[9] :

```
plt.figure(figsize=(4,4))
custom_palette = {'Male': 'yellow', 'Female': 'purple'}
sns.countplot(data=df, x='PerformanceRating',␣

   ↪hue='Gender',palette=custom_palette)
plt.title('Gender Distribution with Performance Rating')
plt.xlabel('Performance Rating')
plt.legend(title='Gender')
plt.gca().set_facecolor('lightgrey') plt.show()
```



Conclusion:

1. Most of the employees have Performance Rating Greater than 3.

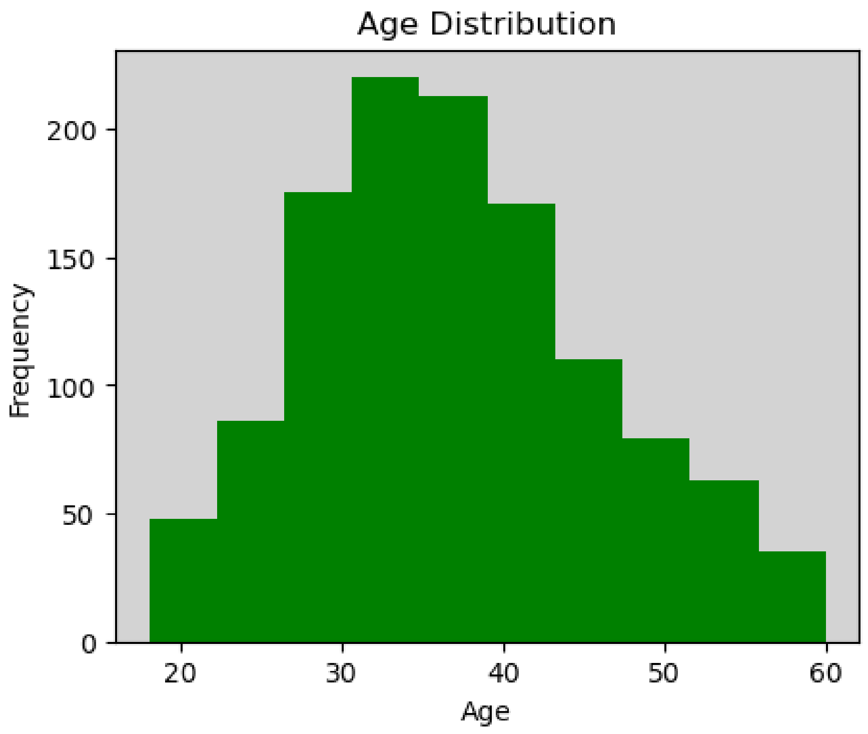
2. In all the rating ranges,cout of male employees is higher than that of female.

So,it can be concluded that men employees outperform women. It is also observed that the rating of 3 is the most common.

V. Employee Age Distribution:

[10] :

```
plt.figure(figsize=(5,4))
age_distribution=df['Age'].plot.hist(color='green') plt.title('Age
Distribution')
plt.xlabel('Age') plt.ylabel('Frequency')
plt.gca().set_facecolor('lightgrey') plt.show()
```



Conclusion:

1. we see that a good many number of the employees are in the age group of 30–40.

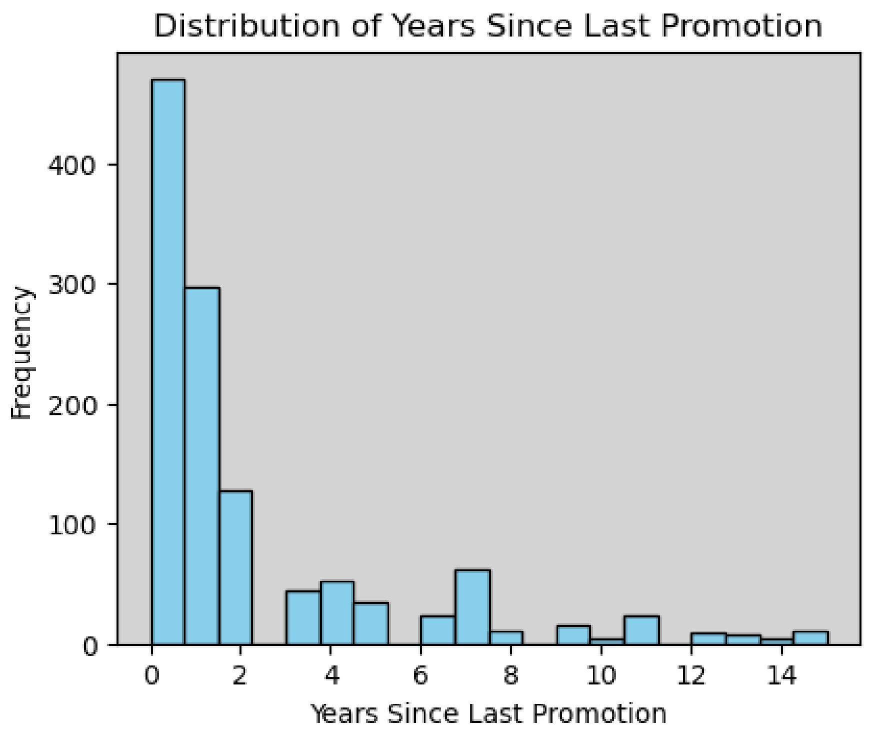
2. There are very few people in the age group of 55–60.

3. Majority of the working class is therefore in their late 30’s.

[11] :

```
# Assuming df is my DataFrame containing the data
plt.figure(figsize=(5,4))
plt.hist(df['YearsSinceLastPromotion'], bins=20, color='skyblue', ␣

  ↪edgecolor='black')
plt.title('Distribution of Years Since Last Promotion')
plt.xlabel('Years Since Last Promotion')
plt.ylabel('Frequency') plt.gca().set_facecolor('lightgrey')
plt.show()
```



Conclusion:

It is seen that a whole lot of employees were being promoted quite often i.e. in 0–1.5 years.

1.0.3 –A comparative Study of Algorithms:

Regression or Classification?

Now,If we check the Performance Rating,which is out target variable,as we can see,it has three values,2,3 and 4,which implies that it is a classification problem as this column is a categorical column.

METHOD-1-FINDING MODEL ACCURACY USING LOGISTIC REGRESSION

[12] :
```
df['PerformanceRating'].unique()
```

[12] :
```
array([3, 4, 2], dtype=int64)
```

Change the Categorical Data using One Hot Encoding method.

[13] :
```
#copying the dafatframe and naming it 'new_df'
new_df = df.copy()

# Assuming 'le' is your LabelEncoder instance
le = LabelEncoder()
```

```
# Assuming df is my DataFrame and here is the list of categorical columns i␣
↪have:
categorical_columns = ['EmpNumber','Gender', 'EducationBackground',␣
↪'MaritalStatus',  'EmpDepartment',␣
↪'EmpJobRole','BusinessTravelFrequency','Attrition','OverTime']
# Applying label encoding to each categorical column
for col in categorical_columns:
        new_df[col] = le.fit_transform(new_df[col])

# Now my categorical columns are encoded with numerical values
```

Training the model:

```
[14] :  #import the test_train split function.
        from sklearn.model_selection import train_test_split
```

```
[15] :  #define x and y: col=list(df)
        x=new_df[col[1:27]]
        y=new_df['PerformanceRating']
```

```
[16] :  #We want to keep 30 % of the data as test size,so,0.3
        x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
[17] :  # length of x_train data?
        len(x_train)
```

```
[17]: 840
```

```
[18] :  # length of x_test data?
        len(x_test)
```

```
[18]: 360
```

```
[19] :  # y_test data
        y_test
```

```
[19]:   175         2
        363         2
        374         4
        161         3
        952         2
                    ..
        1063        3
        221         2
```

```
488     3
317     3
405     2
Name: PerformanceRating, Length: 360, dtype: int64
```

[20] :
```
# revisiting the new_df
new_df
```

[20]:
```
      EmpNumber  Age  Gender  EducationBackground  MaritalStatus  \
0             0   32       1                    2              2
1             1   47       1                    2              2
2             2   40       1                    1              1
3             3   41       1                    0              0
4             4   60       1                    2              2
...         ...  ...     ...                  ...            ...
1195       1195   27       0                    3              0
1196       1196   37       1                    1              2
1197       1197   50       1                    3              1
1198       1198   34       0                    3              2
1199       1199   24       0                    1              2

      EmpDepartment  EmpJobRole  BusinessTravelFrequency  DistanceFromHome  \
0                 5          13                        2                10
1                 5          13                        2                14
2                 5          13                        1                 5
3                 3           8                        2                10
4                 5          13                        2                16
...             ...         ...                      ...               ...
1195              5          13                        1                 3
1196              1          15                        2                10
1197              1          15                        2                28
1198              0           1                        2                 9
1199              5          13                        2                 3

      EmpEducationLevel  ...  EmpRelationshipSatisfaction  \
0                     3  ...                            4
1                     4  ...                            4
2                     4  ...                            3
3                     4  ...                            2
4                     4  ...                            4
...                 ...  ...                          ...
1195                  1  ...                            2
1196                  2  ...                            1
1197                  1  ...                            3
1198                  3  ...                            2
1199                  2  ...                            1
```

```
      TotalWorkExperienceInYears        TrainingTimesLastYear   EmpWorkLifeBalance      \
0                             10                            2                    2
1                             20                            2                    3
2                             20                            2                    3
3                             23                            2                    2
4                             10                            1                    3
...                          ...                          ...                  ...
1195                           6                            3                    3
1196                           4                            2                    3
1197                          20                            3                    3
1198                           9                            3                    4
1199                           4                            3                    3

        ExperienceYearsAtThisCompany  ExperienceYearsInCurrentRole          \
0                                 10                             7
1                                  7                             7
2                                 18                            13
3                                 21                             6
4                                  2                             2
...                              ...                           ...
1195                               6                             5
1196                               1                             0
1197                              20                             8
1198                               8                             7
1199                               2                             2

        YearsSinceLastPromotion      YearsWithCurrManager      Attrition   \
0                             0                         8              0
1                             1                         7              0
2                             1                        12              0
3                            12                         6              0
4                             2                         2              0
...                         ...                       ...            ...
1195                          0                         4              0
1196                          0                         0              0
1197                          3                         8              0
1198                          7                         7              0
1199                          2                         0              1

      PerformanceRating
0                     3
1                     3
2                     4
3                     3
4                     3
...                 ...
1195                  4
```

```
1196         3
1197         3
1198         3
1199         2

[1200 rows x 28 columns]
```

[21] :
```
# importing logistic regression:
from sklearn.linear_model import LogisticRegression model=
LogisticRegression()
```

[22] :
```
#training the model
model.fit(x_train,y_train)
```

```
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in: https://scikit-
    learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options: https://scikit-
    learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[22] : LogisticRegression()

Testing/ Making Predictions:

[23] :
```
model.predict(x_test)
```

```
[23]: array([4, 2, 4, 3, 2, 2, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2,
             3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
             3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 2, 3, 3, 2, 2, 2, 3, 3, 3,
             3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 2,
             3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 4, 3, 3,
             4, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 4, 3, 3, 3, 2, 3, 2, 3, 3, 3, 3,
             3, 4, 2, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 2, 3, 2, 3, 3, 3,
             3, 3, 3, 4, 3, 3, 3, 3, 3, 3, 4, 2, 3, 3, 3, 3, 3, 4, 3, 3, 3, 3,
             3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 2, 3, 2, 4, 3, 3, 3, 3, 3, 3,
             3, 3, 2, 3, 3, 4, 3, 4, 3, 3, 3, 3, 2, 3, 3, 2, 3, 3, 3, 2, 3, 3,
             3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 2, 3, 2, 4, 3, 3, 3, 3, 3, 3,
             3, 3, 2, 3, 3, 3, 2, 3, 2, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
             2, 3, 3, 4, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 2, 3, 4, 3, 3, 3,
             3, 3, 3, 3, 3, 2, 3, 3, 4, 3, 2, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3,
             2, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 3, 3, 2, 3, 3, 3, 3, 3, 4, 3, 3,
             3, 3, 3, 3, 2, 3, 4, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3,
                3, 2, 3, 3, 4, 3, 3, 2], dtype=int64)
```

Check Accuracy of the Model:

```
[24] :  model.score(x_test,y_test)

        #the model is 78% accurate
```

```
[24]: 0.7861111111111111
```

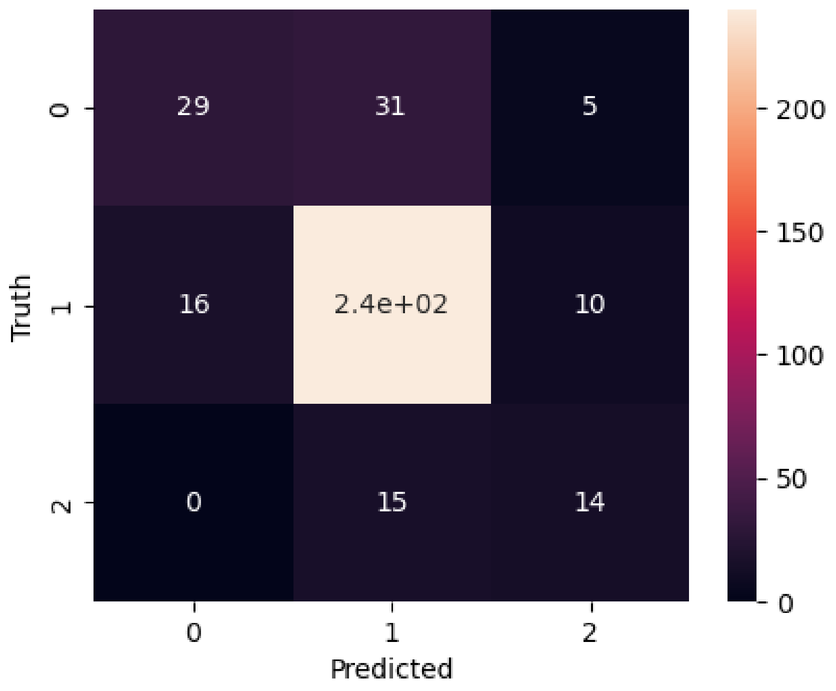
Logistic Regression- Confusion Matrix to Better Visualize the Accuracy and Inaccuracy of the Model:

```
[48]:   y_predicted_m1=model.predict(x_test)
        from     sklearn.metrics     import     confusion_matrix
        cm_1=confusion_matrix(y_test,y_predicted_m1) cm_1
```

```
 [48]: array([[ 29,  31,                   5],
                         [ 16, 240,  10],
               [  0,  15,  14]], dtype=int64)
```

```
[49]:   import seaborn as sn plt.figure(figsize =
        (5,4)) sn.heatmap(cm_1, annot=True)
        plt.xlabel('Predicted') plt.ylabel('Truth')
```

```
[49]: Text(33.22222222222222, 0.5, 'Truth')
```



METHOD-2-RANDOM FOREST CLASSIFIER

The Random Forest is a better approach over other algorithms because it uses the entire dataset optimally which reduces bias error. The algorithm can also provide maximum reduction in variance as it gives the average output from an ensemble of several decision trees; hence the name 'Random Forest'.

[27]:
```
# importing the RandomForestClassifier function
from sklearn.ensemble import RandomForestClassifier
```

[42]:
```
# training the model using random forest
clf=RandomForestClassifier(n_jobs=2,oob_score=True,n_estimators=500) clf.fit(x_train,y_train)
```

[42]: RandomForestClassifier(n_estimators=500, n_jobs=2, oob_score=True) [34]:

```
#Applying classifier to test data:
clf.predict(x_test)
```

```
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 3, 3, 3, 3, 3, 2, 3, 3, 4, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 2, 2, 2, 3, 3, 3,
3, 3, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 2,
3, 3, 3, 3, 2, 4, 2, 4, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 4, 3, 3,
4, 3, 3, 3, 3, 3, 3, 2, 4, 3, 3, 4, 3, 3, 3, 2, 3, 2, 3, 3, 3, 2,
2,  4,  2,  4,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  2,  3,  2,  3,  3,  3,
3,  3,  3,  2,  3,  3,  2,  3,  3,  3,  4,  2,  4,  3,  3,  3,  2,  3,  3,  3,  2,  2,
3,  3,  3,  3,  2,  3,  3,  3,  3,  3,  3,  3,  3,  3,  2,  3,  2,  3,  3,  3,  3,  3,
3,  4,  2,  3,  3,  3,  2,  4,  3,  3,  3,  2,  2,  3,  3,  2,  3,  3,  3,  2,  3,  3,
3,  3,  3,  2,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  2,  4,  3,  3,  3,  3,  3,  3,
3,  3,  3,  3,  3,  3,  2,  3,  2,  3,  3,  3,  3,  3,  3,  3,  3,  3,  2,  3,  3,  3,
3,  4,  2,  4,  3,  4,  3,  3,  3,  3,  3,  3,  3,  3,  4,  4,  3,  3,  4,  4,  3,  3,
3,  3,  3,  3,  3,  2,  3,  3,  4,  3,  2,  2,  3,  3,  3,  3,  2,  2,  2,  4,  3,  3,
3,  3,  3,  3,  3,  3,  2,  3,  3,  4,  3,  2,  3,  3,  3,  3,  3,  3,  3,  3,  3,  4,
2,  3,  3,  3,  2,  3,  2,  3,  2,  3,  3,  3,  3,  3,  2,  3,  2,  3,  3,  3,  3,  3,
3, 2, 2, 3, 2, 3, 3, 2], dtype=int64)
```

Check Accuracy of the Model:

[51]:
```
clf.score(x_test,y_test)

# the model is 93 % accurate
```

[51]:
```
0.9388888888888889
```

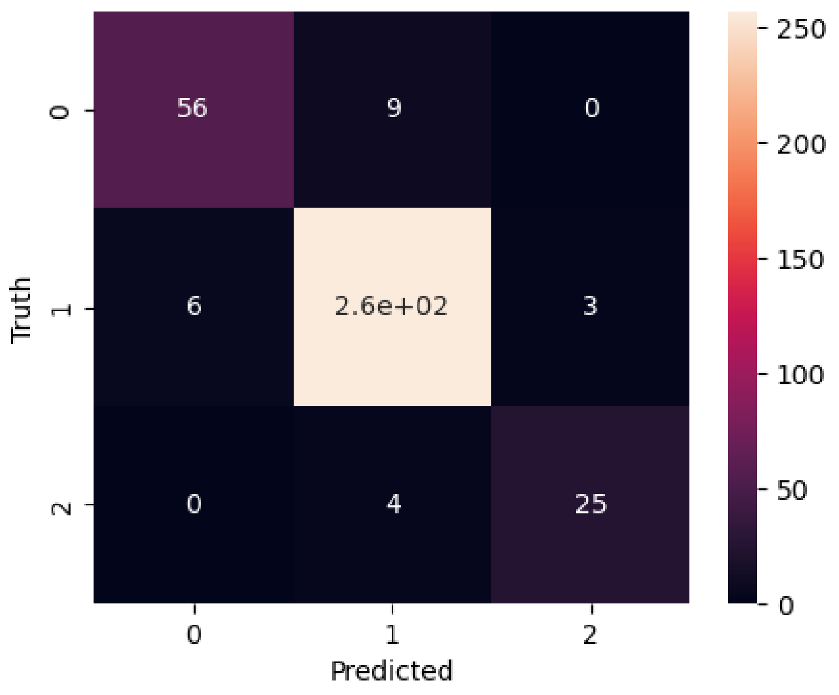
Random Forest- Confusion Matrix to Better Visualize the Accuracy and Inaccuracy of the Model:

[50]:
```
y_predicted_m2=clf.predict(x_test)
from sklearn.metrics import confusion_matrix
cm_2=confusion_matrix(y_test,y_predicted_m2) cm_2
```

[50]:
```
array([[ 56,   9,   0],
       [  6, 257,   3],
       [  0,   4,  25]], dtype=int64)
```

[52]:
```
import seaborn as sn plt.figure(figsize =
(5,4)) sn.heatmap(cm_2, annot=True)
plt.xlabel('Predicted') plt.ylabel('Truth')
```

[52]:
```
Text(33.22222222222222, 0.5, 'Truth')
```



The Acuracy of the model increases from 78% to 93% using Random Forest Classifier Method.

Recommendations to the organization:

After careful evaluation of the drawn observations, we can recommend the following to increase employee performance at an organization:

1. Ensure a more improved rate of salary raises for the employees

2. Create a more friendly, comfortable and inclusive office environment

3. Help upgrade the skill sets of the current employees, so as to make them suitable to take up more responsibilities and challenges, and in turn, prepare them for promotions.

Comparison between Algorithms

There are several reasons why the Random Forest method works more accurately than other Algorithms(Here Logistic Regression).

- ➢ Model Complexity:
  - **Logistic Regression:** Assumes a simple linear relationship between features and the target, suitable for binary classification tasks.
  - **Random Forest Classifier**: Utilizes multiple decision trees to capture complex feature interactions, enhancing its ability to understand intricate relationships.
- ➢ Handling Non-linearity:
  - Logistic Regression may struggle with highly non-linear relationships, whereas Random Forest Classifier excels in capturing non-linear patterns effectively.
- ➢ Overfitting:
  - Logistic Regression is less prone to overfitting, especially with smaller datasets or large feature sets.
  - Random Forest Classifier, despite its complexity, may generalize well to the data, evident from the significantly higher accuracy achieved.
- ➢ Feature Importance:
  - Random Forest Classifier provides feature importance scores, aiding in identifying influential features for predictions and enhancing interpretability.
- ➢ Data Size and Complexity:
  - The performance of algorithms can vary based on dataset characteristics. In this scenario, Random Forest Classifier might benefit from the larger dataset or the presence of non-linear relationships.

These insights shed light on the nuances between Logistic Regression and Random Forest Classifier, highlighting their strengths and weaknesses in predicting employee performance ratings.