HR Analytics: Promotions Prediction Model

This Analysis is a predictive model aimed at predicting staffs who are likely to receive a promotion based on the trained model.

- The model seeks to reduce staff attrition by making aware staffs who are due for promotion

Methodology

- Data Collection
- Data Preprocessing/ Wrangling
- Data exploration
- Data Modelling
- Data Evaluation

Importing Python Libraries

In [12]:

```
# Import all required python libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
%matplotlib inline
```

Importing the dataset

In [13]:

```
# Read csv file into python
df= pd.read_csv("C:/Users/User/Desktop/DataSets/HR_Analytics.csv")
```

In [14]:

```
# to view the top 10 records
df.head(10)
```

Out[14]:

	EmployeeNo	Division	Qualification category	Qualification	Gender Category	Gender	Channel_of_Recruitment	Trainings_Attended	Year_of_birth
0	YAK/S/00001	Commercial Sales and Marketing	1.0	MSc, MBA and PhD	0	Female	Direct Internal process	2	1986
1	YAK/S/00002	Customer Support and Field Operations	2.0	First Degree or HND	1	Male	Agency and others	2	1991
2	YAK/S/00003	Commercial Sales and Marketing	2.0	First Degree or HND	1	Male	Direct Internal process	2	1987
3	YAK/S/00004	Commercial Sales and Marketing	2.0	First Degree or HND	1	Male	Agency and others	3	1982
4	YAK/S/00006	Information and Strategy	2.0	First Degree or HND	1	Male	Direct Internal process	3	1990
5	YAK/S/00007	Customer Support and Field Operations	2.0	First Degree or HND	0	Female	Agency and others	2	1990
		Customer							

	EmployeeNo	Division	Qualification category	Qualification	Gender Category	Gender	Channel_of_Recruitment	Trainings_Attended	Year_of_birth
6	YAK/S/00008	Support and Field Operations	1.0	MSc, MBA and PhD	1	Male	Direct Internal process	2	1988
7	YAK/S/00009	Information and Strategy	2.0	First Degree or HND	1	Male	Agency and others	2	1993
8	YAK/S/00010	Commercial Sales and Marketing	1.0	MSc, MBA and PhD	1	Male	Direct Internal process	2	1989
9	YAK/S/00012	Commercial Sales and Marketing	2.0	First Degree or HND	0	Female	Direct Internal process	2	1986

10 rows × 29 columns

In [15]:

```
# view datatypes of the record
df.dtypes
```

Out[15]:

```
EmployeeNo                              object
Division                                object
Qualification category                 float64
Qualification                           object
Gender Category                          int64
Gender                                  object
Channel_of_Recruitment                  object
Trainings_Attended                       int64
Year_of_birth                            int64
Last_performance_score                 float64
Year_of_recruitment                      int64
Targets_met                              int64
Agent recruitment                        int64
Yearsof Service                          int64
Serve_Train                            float64
AgeNow                                   int64
Previous_Award                           int64
Training_score_average                   int64
State_Of_Origin                         object
SchoolID                                 int64
Foreign_schooled                        object
Marital_cat                              int64
Marital_Status                          object
Past_Disciplinary_Action_cat             int64
Past_Disciplinary_Action                object
Movement_cat                             int64
Previous_IntraDepartmental_Movement     object
No_of_previous_employers                object
Promoted_or_Not                          int64
dtype: object
```

In [16]:

```
# to check the total number of rows and columns of the dataset
df.shape
```

Out[16]:

```
(38312, 29)
```

Data Preprocessing/ Wrangling

Handling missing values

In [17]:

```
df.isnull().mean()*100
```

Out[17]:

```
EmployeeNo                               0.000000
Division                                 0.000000
Qualification category                   4.382439
Qualification                            4.382439
Gender Category                          0.000000
Gender                                   0.000000
Channel_of_Recruitment                   0.000000
Trainings_Attended                       0.000000
Year_of_birth                            0.000000
Last_performance_score                   0.000000
Year_of_recruitment                      0.000000
Targets_met                              0.000000
Agent recruitment                        0.000000
Yearsof Service                          0.000000
Serve_Train                              0.000000
AgeNow                                   0.000000
Previous_Award                           0.000000
Training_score_average                   0.000000
State_Of_Origin                          0.000000
SchoolID                                 0.000000
Foreign_schooled                         0.000000
Marital_cat                              0.000000
Marital_Status                           0.000000
Past_Disciplinary_Action_cat             0.000000
Past_Disciplinary_Action                 0.000000
Movement_cat                             0.000000
Previous_IntraDepartmental_Movement      0.000000
No_of_previous_employers                 0.000000
Promoted_or_Not                          0.000000
dtype: float64
```

In [18]:

```
# replacing missing values with the mode
df["Qualification category"].replace(np.nan, df["Qualification category"].value_counts().idxmax(),i
nplace= True)
df["Qualification"].replace(np.nan,df["Qualification"].value_counts().idxmax(), inplace= True)
```

In [19]:

```
# to check again for missing values
df.isnull().any().any()
```

Out[19]:

```
False
```

In [20]:

```
# to convert datatypes
df["Qualification category"]=df["Qualification category"].astype("int")
```

In [21]:

```
# to explore the dataset
df.describe()
```

Out[21]:

	Qualification category	Gender Category	Trainings_Attended	Year_of_birth	Last_performance_score	Year_of_recruitment	Targets_met	r
count	38312.000000	38312.000000	38312.000000	38312.000000	38312.000000	38312.000000	38312.000000	383
mean	1.742039	0.701608	2.253680	1986.209334	7.698959	2013.139695	0.352996	
std	0.471183	0.457558	0.609443	7.646047	3.744135	4.261451	0.477908	
min	1.000000	0.000000	2.000000	1950.000000	0.000000	1982.000000	0.000000	
25%	1.000000	0.000000	2.000000	1982.000000	5.000000	2012.000000	0.000000	
50%	2.000000	1.000000	2.000000	1988.000000	7.500000	2014.000000	0.000000	

	Qualification category	Gender Category	Trainings_Attended	Year_of_birth	Last_performance_score	Year_of_recruitment	Targets_met
75%	2.000000	1.000000	2.000000	1992.000000	10.000000	2016.000000	1.000000
max	3.000000	1.000000	11.000000	2001.000000	12.500000	2018.000000	1.000000

Binning some columns using their interquatile percentile, thus making them categorical

In [22]:

```
# Bin the following columns ["AgeNow","Last_performance_score","Training_score_average"]
bins = [18,27,31,37,69]
group_names = ["0","1","2","3"]
df["Age_binned"] = pd.cut(df["AgeNow"], bins= bins, labels = group_names, include_lowest= True)

bins = [1,3,5,7,37]
group_names = ["0","1","2","3"]
df["Yearsof Service_binned"] = pd.cut(df["Yearsof Service"], bins= bins, labels = group_names, incl
ude_lowest= True)

bins = [0,5,7.5,10,12.5]
group_names = ["0","1","2","3"]
df["Last_performance_score_binned"] = pd.cut(df["Last_performance_score"], bins= bins, labels = gro
up_names, include_lowest= True)

bins = [31,43,52,68,91]
group_names = ["0","1","2","3"]
df["Training_score_average_binned"] = pd.cut(df["Training_score_average"], bins= bins, labels = gro
up_names, include_lowest= True)
```

In [23]:

```
df.head()
```

Out[23]:

	EmployeeNo	Division	Qualification category	Qualification	Gender Category	Gender	Channel_of_Recruitment	Trainings_Attended	Year_of_birth
0	YAK/S/00001	Commercial Sales and Marketing	1	MSc, MBA and PhD	0	Female	Direct Internal process	2	1986
1	YAK/S/00002	Customer Support and Field Operations	2	First Degree or HND	1	Male	Agency and others	2	1991
2	YAK/S/00003	Commercial Sales and Marketing	2	First Degree or HND	1	Male	Direct Internal process	2	1987
3	YAK/S/00004	Commercial Sales and Marketing	2	First Degree or HND	1	Male	Agency and others	3	1982
4	YAK/S/00006	Information and Strategy	2	First Degree or HND	1	Male	Direct Internal process	3	1990

5 rows × 33 columns

Data Exploration

In [24]:

```
# to summarize Age binned by Promoted_or_not
df.groupby(["Age_binned", "Promoted_or_Not"]).count()["EmployeeNo"].to_frame()
```

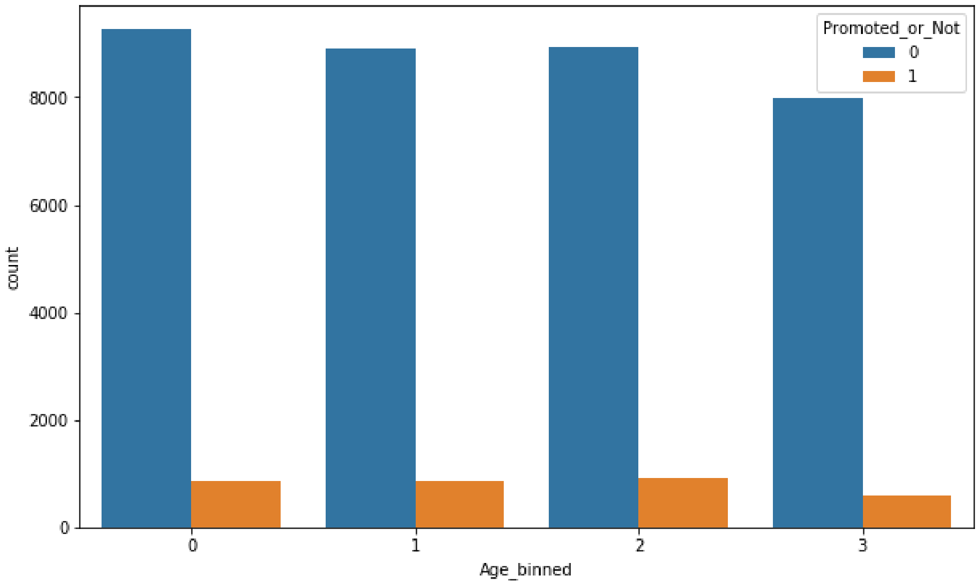
Out[24]:

		EmployeeNo
Age_binned	Promoted_or_Not	
0	0	9255

Age_binned	Promoted_or_Not	EmployeeNo
	1	858
1	0	8894
	1	865
2	0	8935
	1	912
3	0	7987
	1	606

In [26]:

```
# visualizing this data
plt.figure(figsize=(10,6))
sns.countplot("Age_binned", hue="Promoted_or_Not",data= df);
plt.figure(figsize= (14,8))
plt.show()
```




```
<Figure size 1008x576 with 0 Axes>
```

In [27]:

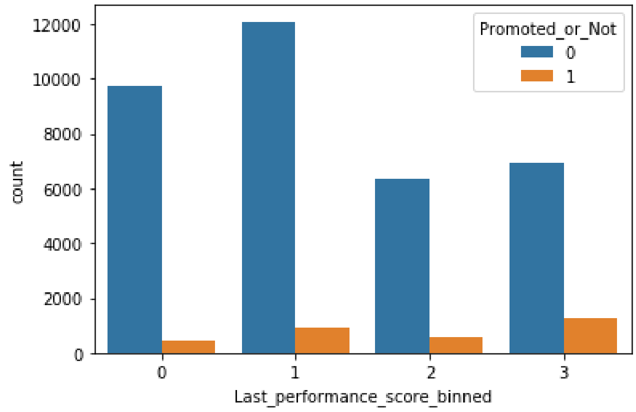
```
df.groupby(["Last_performance_score_binned", "Promoted_or_Not"]).count()["EmployeeNo"].to_frame()
```

Out[27]:

Last_performance_score_binned	Promoted_or_Not	EmployeeNo
0	0	9731
	1	443
1	0	12059
	1	934
2	0	6367
	1	565
3	0	6914
	1	1299

In [30]:

```
sns.countplot("Last_performance_score_binned", hue="Promoted_or_Not",data= df);
plt.figure(figsize= (14,8))
plt.show()
```




```
<Figure size 1008x576 with 0 Axes>
```

In [50]:

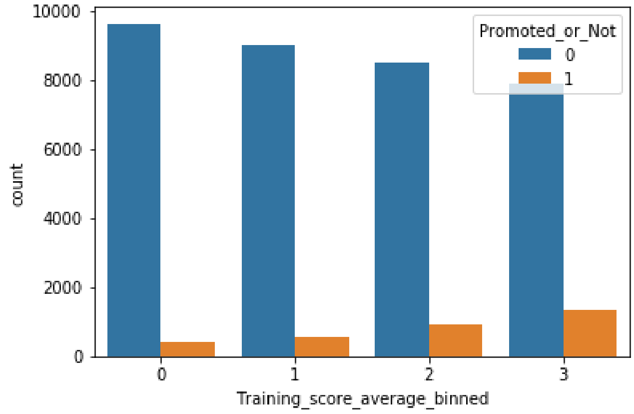
```
df.groupby(["Training_score_average_binned", "Promoted_or_Not"]).count()["EmployeeNo"].to_frame()
```

Out[50]:

		EmployeeNo
Training_score_average_binned	**Promoted_or_Not**	
0	**0**	**9627**
	1	**409**
1	**0**	**9033**
	1	**541**
2	**0**	**8506**
	1	**933**
3	**0**	**7905**
	1	**1358**

In [51]:

```
sns.countplot("Training_score_average_binned", hue="Promoted_or_Not",data= df);
plt.figure(figsize= (14,8))
plt.show()
```




```
<Figure size 1008x576 with 0 Axes>
```

In [52]:

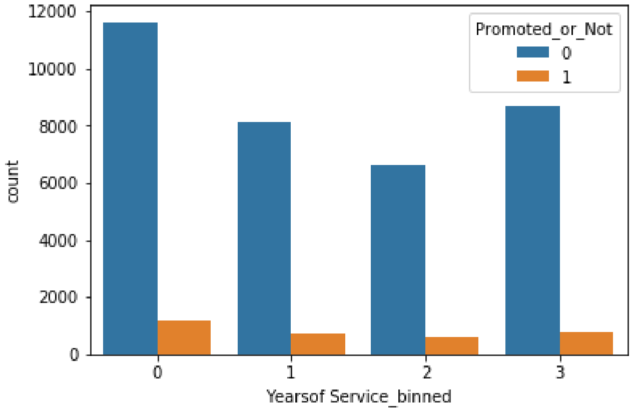
```
df.groupby(["Yearsof Service_binned", "Promoted_or_Not"]).count()["EmployeeNo"].to_frame()
```

Out[52]:

		EmployeeNo
Yearsof Service_binned	**Promoted_or_Not**	
0	0	11617
	1	1149
1	0	8151
	1	707
2	0	6630
	1	593
3	0	8673
	1	792

In [53]:

```
sns.countplot("Yearsof Service_binned", hue="Promoted_or_Not",data= df);
plt.figure(figsize= (14,8))
plt.show()
```




```
<Figure size 1008x576 with 0 Axes>
```

In [54]:

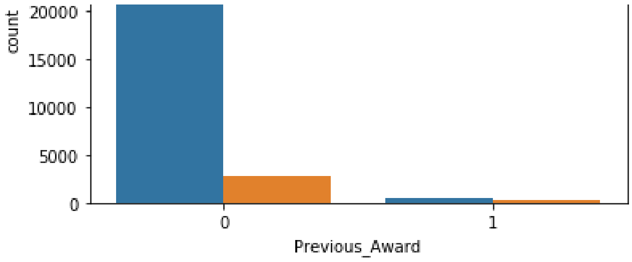
```
df.groupby(["Previous_Award", "Promoted_or_Not"]).count()["EmployeeNo"].to_frame()
```

Out[54]:

		EmployeeNo
Previous_Award	**Promoted_or_Not**	
0	0	34582
	1	2843
1	0	489
	1	398

In [55]:

```
sns.countplot("Previous_Award", hue="Promoted_or_Not",data= df);
plt.figure(figsize= (14,8))
plt.show()
```



<Figure size 1008x576 with 0 Axes>

In [56]:

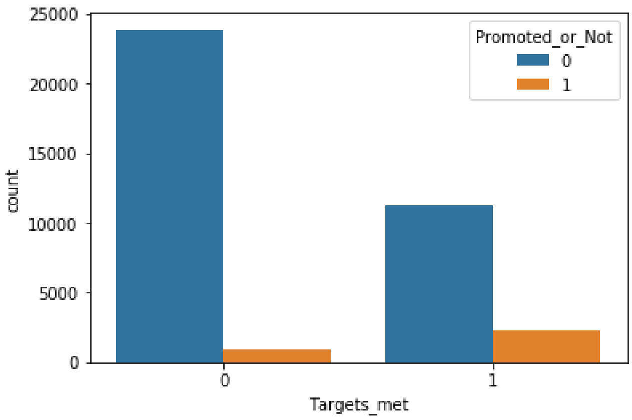
```
df.groupby(["Targets_met", "Promoted_or_Not"]).count()["EmployeeNo"].to_frame()
```

Out[56]:

		EmployeeNo
Targets_met	**Promoted_or_Not**	
0	**0**	23835
	1	953
1	**0**	11236
	1	2288

In [57]:

```
sns.countplot("Targets_met", hue="Promoted_or_Not",data= df);
plt.figure(figsize= (14,8))
plt.show()
```



<Figure size 1008x576 with 0 Axes>

Determining the coefficient of correlation and the p_value between an independent variable and target variable

In [33]:

```
pearson_coef, p_value= stats.pearsonr(df["AgeNow"], df["Promoted_or_Not"])
pearson_coef1, p_value2= stats.pearsonr(df["Gender Category"], df["Promoted_or_Not"])
```

In [34]:

```
pearson_coef, p_value
```

Out[34]:

```
(-0.010436563383179494, 0.041073761805808275)
```

In [61]:

```
df.corr()
```

Out[61]:

	Qualification category	Gender Category	Trainings_Attended	Year_of_birth	Last_performance_score	Year_of_recruitment
Qualification category	1.000000	0.020944	0.032098	0.389515	-0.124301	0.265524
Gender Category	0.020944	1.000000	0.084906	0.012095	-0.023586	0.017644
Trainings_Attended	0.032098	0.084906	1.000000	0.078710	-0.062042	0.056215
Year_of_birth	0.389515	0.012095	0.078710	1.000000	-0.175572	0.654666
Last_performance_score	-0.124301	-0.023586	-0.062042	-0.175572	1.000000	-0.190333
Year_of_recruitment	0.265524	0.017644	0.056215	0.654666	-0.190333	1.000000
Targets_met	-0.004676	-0.038382	-0.044789	0.025337	0.276350	0.076910
Agent recruitment	-0.316899	-0.002967	-0.062164	-0.833329	0.091178	-0.127697
Yearsof Service	-0.265524	-0.017644	-0.056215	-0.654666	0.190333	-1.000000
Serve_Train	-0.257366	-0.032692	-0.241498	-0.643042	0.190762	-0.969173
AgeNow	-0.389515	-0.012095	-0.078710	-1.000000	0.175572	-0.654666
Previous_Award	0.003246	0.001773	-0.007409	0.013627	0.026587	0.041995
Training_score_average	-0.025928	-0.024311	0.041065	0.048390	0.057836	0.037477
SchoolID	-0.005842	0.016073	-0.005108	-0.001877	-0.001923	-0.000253
Marital_cat	0.001333	-0.002753	-0.004499	-0.002487	0.000863	0.000897
Past_Disciplinary_Action_cat	-0.004463	0.012799	-0.002260	-0.000251	-0.003065	0.003217
Movement_cat	-0.001056	-0.002715	-0.005871	0.011412	-0.005478	0.004988
Promoted_or_Not	-0.024674	-0.010437	-0.024345	0.017991	0.119690	0.012287

In [35]:

```
# Saving as CSV
df.to_csv("C:/Users/User/Desktop/DataSets/Promotions_Prediction.csv")
```

Data Modelling

Splitting the data into Train and Test data

In [36]:

```
df.columns
```

Out[36]:

```
Index(['EmployeeNo', 'Division', 'Qualification category', 'Qualification',
       'Gender Category', 'Gender', 'Channel_of_Recruitment',
       'Trainings_Attended', 'Year_of_birth', 'Last_performance_score',
       'Year_of_recruitment', 'Targets_met', 'Agent recruitment',
```

```
       'Yearsof Service', 'Serve_Train', 'AgeNow', 'Previous_Award',
       'Training_score_average', 'State_Of_Origin', 'SchoolID',
       'Foreign_schooled', 'Marital_cat', 'Marital_Status',
       'Past_Disciplinary_Action_cat', 'Past_Disciplinary_Action',
       'Movement_cat', 'Previous_IntraDepartmental_Movement',
       'No_of_previous_employers', 'Promoted_or_Not', 'Age_binned',
       'Yearsof Service_binned', 'Last_performance_score_binned',
       'Training_score_average_binned'],
      dtype='object')
```

In [41]:

```
# Selecting columns
select_columns= df.loc[:, df.dtypes != np.object]
```

In [55]:

```
select_columns
```

Out[55]:

	Qualification category	Gender Category	Trainings_Attended	Targets_met	Previous_Award	SchoolID	Marital_cat	Past_Disciplinary_Action_cat
0	1	0	2	1	0	0	1	0
1	2	1	2	0	0	1	1	0
2	2	1	2	0	0	1	1	0
3	2	1	3	0	0	1	2	0
4	2	1	3	0	0	1	1	0
...	...	...	...	...	...	...	...	...
38307	2	0	2	0	0	1	1	0
38308	1	0	2	0	0	1	1	0
38309	2	1	2	1	0	0	1	0
38310	2	1	2	0	0	1	1	0
38311	2	1	2	0	0	1	2	0

38312 rows × 14 columns

Splitting the data

In [62]:

```
X_data= select_columns.drop(['Promoted_or_Not'],axis=1)
y_data= select_columns.iloc[:,9]
```

In [63]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test= train_test_split(X_data,y_data,test_size=0.20,random_state=0)
```

In [64]:

```
# count the number of rows and columns of train and test data
X_train.shape, X_test.shape
```

Out[64]:

```
((30649, 13), (7663, 13))
```

Training the model using Logistic Regression

In [65]:

In [65]:

```
from sklearn.linear_model import LogisticRegression
```

In [66]:

```
Lr= LogisticRegression(random_state= 0)
```

In [68]:

```
classifier= Lr.fit(X_train, y_train)
```

Testing the model using the test data

In [69]:

```
y_pred= Lr.predict(X_test)
```

In [70]:

```
y_pred
```

Out[70]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

Evaluating the model

In [71]:

```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_test, y_pred)
```

In [79]:

```
print("Score :{:.2f}%".format(score))
```

```
Score :0.92%
```

The score of the model is approximately 92%

In [73]:

```
from sklearn.model_selection import cross_val_score
cvs= cross_val_score(Lr, X_train,y_train,cv=20, scoring='accuracy')
cvs.mean(), cvs.std()
```

Out[73]:

```
(0.9148097384094737, 0.0020632109197529047)
```

In [74]:

```
print("Accuracy: {:.2f}%".format(cvs.mean()*100))
print("Standard deviation: {:.2f}%".format(cvs.std()*100))
```

```
Accuracy: 91.48%
Standard deviation: 0.21%
```

Observation:

- **Mean accuracy score is 91.48% with a standard deviation of +- 0.21**

Meaning our data lies within 91.45 + 0.21 or 91.48 - 0.21

Evaluating the model using confusion matrix

In [75]:

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
```

In [76]:

```
cm
```

Out[76]:

```
array([[6966,   41],
       [ 609,   47]], dtype=int64)
```

This shows that we have:

- **6966 True Positives (Correct Positive Predictions)**
- **41 False Positives (Incorrect Positive Predictions)**
- **609 False Negatives (Incorrect Negative Predictions)**
- **47 True Negatives (Correct Negative Predictions)**

In [ ]:

```

```