**Python Field Project**

# importing numpy, pandas, matplotlib, seaborn

import numpy as np  
import pandas as pd  
  
import matplotlib.pyplot as plt  
import seaborn as sns  
# reading the data  
  
train = pd.read\_csv('https://raw.githubusercontent.com/sameerCoder/DATA\_ANALYST\_DATASETS/main/HrAnalytics/HrAnalytics\_train.csv')  
test = pd.read\_csv('https://raw.githubusercontent.com/sameerCoder/DATA\_ANALYST\_DATASETS/main/HrAnalytics/HrAnalytics\_test.csv')

# getting their shapes

print("Shape of train :", train.shape)  
print("Shape of test :", test.shape)

# Getting head of train and test

train.head()  
  
test.head()

# describing the training set  
  
train.describe(include = 'all')

# Getting info of train

train.info()

# checking if there is any NULL value in the dataset  
  
train.isnull().any()

#Getting sum of null quantities.

test.isnull().sum()

\*\*UNi-variate Data Visualization\*\*

# looking at the most popular departments  
  
from wordcloud import WordCloud  
from wordcloud import STOPWORDS  
  
stopword = set(STOPWORDS)  
  
wordcloud = WordCloud(background\_color = 'white', stopwords = stopword).generate(str(train['department']))  
  
plt.rcParams['figure.figsize'] = (12, 8)  
print(wordcloud)  
plt.imshow(wordcloud)  
plt.title('Most Popular Departments', fontsize = 30)  
plt.axis('off')  
plt.show()

# checkig the no. of Employees Promoted  
  
train['is\_promoted'].value\_counts()  
  
# finding the %age of people promoted  
  
promoted = (4668/54808)\*100  
print("Percentage of Promoted Employees is {:.2f}%".format(promoted))

#plotting a scatter plot   
  
plt.hist(train['is\_promoted'])  
plt.title('plot to show the gap in Promoted and Non-Promoted Employees', fontsize = 30)  
plt.xlabel('0 -No Promotion and 1- Promotion', fontsize = 20)  
plt.ylabel('count')  
plt.show()

# checking the distribution of the avg\_training score of the Employees

import seaborn as sns  
plt.rcParams['figure.figsize'] = (15, 7)  
sns.histplot(train['avg\_training\_score'], color = 'blue')  
plt.title('Distribution of Training Score among the Employees', fontsize = 30)  
plt.xlabel('Average Training Score', fontsize = 20)  
plt.ylabel('count')  
plt.show()

# Counting the number of persons who have won the award

train['awards\_won?'].value\_counts()

# plotting a donut chart for visualizing each of the recruitment channel's share  
  
size = [53538, 1270]  
colors = ['magenta', 'brown']  
labels = "Awards Won", "NO Awards Won"  
  
my\_circle = plt.Circle((0, 0), 0.7, color = 'white')  
  
plt.rcParams['figure.figsize'] = (9, 9)  
plt.pie(size, colors = colors, labels = labels, shadow = True, autopct = '%.2f%%')  
plt.title('Showing a Percentage of employees who won awards', fontsize = 30)  
p = plt.gcf()  
p.gca().add\_artist(my\_circle)  
plt.legend()  
plt.show()

train['KPIs\_met >80%'].value\_counts()

# plotting a pie chart  
  
  
size = [35517, 19291]  
labels = "Not Met KPI > 80%", "Met KPI > 80%"  
colors = ['violet', 'grey']  
explode = [0, 0.1]  
  
plt.rcParams['figure.figsize'] = (8, 8)  
plt.pie(size, labels = labels, colors = colors, explode = explode, shadow = True, autopct = "%.2f%%")  
plt.title('A Pie Chart Representing Gap in Employees in terms of KPI', fontsize = 30)  
plt.axis('off')  
plt.legend()

plt.show()

# checking the distribution of length of service  
  
sns.distplot(train['length\_of\_service'], color = 'green')  
plt.title('Distribution of length of service among the Employees', fontsize = 30)  
plt.xlabel('Length of Service in years', fontsize = 15)  
plt.ylabel('count')  
plt.show()

train['previous\_year\_rating'].value\_counts().sort\_values().plot.bar(color = 'violet', figsize = (15, 7))  
plt.title('Distribution of Previous year rating of the Employees', fontsize = 30)  
plt.xlabel('Ratings', fontsize = 15)  
plt.ylabel('count')  
plt.show()

# checking the distribution of age of Employees in the company  
  
sns.distplot(train['age'], color = 'red')  
plt.title('Distribution of Age of Employees', fontsize = 30)  
plt.xlabel('Age', fontsize = 15)  
plt.ylabel('count')  
plt.show()

# checking the different no. of training done by the employees  
  
plt.rcParams['figure.figsize'] = (17, 7)  
sns.violinplot(train['no\_of\_trainings'], color = 'purple')  
plt.title('No. of trainings done by the Employees', fontsize = 30)  
plt.xlabel('No. of Trainings', fontsize = 15)  
plt.ylabel('Frequency')  
plt.show()

# checking/ the different types of recruitment channels for the company  
  
train['recruitment\_channel'].value\_counts()  
  
# plotting a donut chart for visualizing each of the recruitment channel's share  
  
size = [30446, 23220, 1142]  
colors = ['yellow', 'red', 'lightgreen']  
labels = "Others", "Sourcing", "Reffered"  
  
my\_circle = plt.Circle((0, 0), 0.7, color = 'white')  
  
plt.rcParams['figure.figsize'] = (9, 9)  
plt.pie(size, colors = colors, labels = labels, shadow = True, autopct = '%.2f%%')  
plt.title('Showing share of different Recruitment Channels', fontsize = 30)  
p = plt.gcf()  
p.gca().add\_artist(my\_circle)  
plt.legend()  
plt.show()

# checing the most popular education degree among the employees  
  
from wordcloud import WordCloud  
from wordcloud import STOPWORDS  
  
stopword = set(STOPWORDS)  
  
wordcloud = WordCloud(background\_color = 'white', stopwords = stopword, max\_words = 5).generate(str(train['education']))  
  
plt.rcParams['figure.figsize'] = (12, 8)  
print(wordcloud)  
plt.imshow(wordcloud)  
plt.title('Most Popular Degrees among the Employees', fontsize = 30)  
plt.axis('off')  
plt.show()

# checking the gender gap  
  
train['gender'].value\_counts()  
  
  
# plotting a pie chart  
  
size = [38496, 16312]  
labels = "Male", "Female"  
colors = ['yellow', 'orange']  
explode = [0, 0.1]  
  
plt.rcParams['figure.figsize'] = (8, 8)  
plt.pie(size, labels = labels, colors = colors, explode = explode, shadow = True, autopct = "%.2f%%")  
plt.title('A Pie Chart Representing GenderGap', fontsize = 30)  
plt.axis('off')  
plt.legend()  
plt.show()

# checking the different regions of the company  
  
plt.rcParams['figure.figsize'] = (20, 10)  
sns.countplot(train['region'], color = 'pink')  
plt.title('Different Regions in the company', fontsize = 30)  
plt.xticks(rotation = 60)  
plt.xlabel('Region Code', fontsize = 15)  
plt.ylabel('count', fontsize = 15)  
plt.show()

\*\*Bi-varaiate Data Visualization\*\*

# scatter plot between average training score and is\_promoted  
  
data = pd.crosstab(train['avg\_training\_score'], train['is\_promoted'])  
data.div(data.sum(1).astype(float), axis = 0).plot(kind = 'bar', stacked = True, figsize = (20, 9), color = ['darkred', 'lightgreen'])  
  
plt.title('Looking at the Dependency of Training Score in promotion', fontsize = 30)  
plt.xlabel('Average Training Scores', fontsize = 15)  
plt.legend()  
plt.show()

\*\*As, the Training Scores Increases, the chances of Promotion Increases Highly\*\*

# checking dependency of different regions in promotion  
  
data = pd.crosstab(train['region'], train['is\_promoted'])  
data.div(data.sum(1).astype('float'), axis = 0).plot(kind = 'bar', stacked = True, figsize = (20, 8), color = ['lightblue', 'purple'])  
  
plt.title('Dependency of Regions in determining Promotion of Employees', fontsize = 30)  
plt.xlabel('Different Regions of the Company', fontsize = 20)  
plt.legend()  
plt.show()

\*\*The above graph shows that there is no biasedness over regions in terms of Promotion as all the regions share promotions almost equally.\*\*

# dependency of awards won on promotion  
  
data = pd.crosstab(train['awards\_won?'], train['is\_promoted'])  
data.div(data.sum(1).astype('float'), axis = 0).plot(kind = 'bar', stacked = True, figsize = (10, 8), color = ['magenta', 'purple'])  
  
plt.title('Dependency of Awards in determining Promotion', fontsize = 30)  
plt.xlabel('Awards Won or Not', fontsize = 20)  
plt.legend()  
plt.show()

\*\*There is a very good chance of getting promoted if the employee has won an award\*\*

#dependency of KPIs with Promotion  
  
data = pd.crosstab(train['KPIs\_met >80%'], train['is\_promoted'])  
data.div(data.sum(1).astype('float'), axis = 0).plot(kind = 'bar', stacked = True, figsize = (10, 8), color = ['pink', 'darkred'])  
  
plt.title('Dependency of KPIs in determining Promotion', fontsize = 30)  
plt.xlabel('KPIs Met or Not', fontsize = 20)  
plt.legend()  
plt.show()

\*\*Again Having a good KPI score increases the chances of getting promoted in the company.\*\*

\*\*Again Having a good KPI score increases the chances of getting promoted in the company.\*\*  
  
# checking dependency on previous years' ratings  
  
data = pd.crosstab(train['previous\_year\_rating'], train['is\_promoted'])  
data.div(data.sum(1).astype('float'), axis = 0).plot(kind = 'bar', stacked = True, figsize = (15, 8), color = ['violet', 'pink'])  
  
plt.title('Dependency of Previous year Ratings in determining Promotion', fontsize = 30)  
plt.xlabel('Different Ratings', fontsize = 20)  
plt.legend()  
plt.show()

# checking dependency of age factor in promotion of employees  
  
data = pd.crosstab(train['age'], train['is\_promoted'])  
data.div(data.sum(1).astype('float'), axis = 0).plot(kind = 'bar', stacked = True, figsize = (20, 8), color = ['lightblue', 'green'])  
  
plt.title('Dependency of Age in determining Promotion of Employees', fontsize = 30)  
plt.xlabel('Age of Employees', fontsize = 20)  
plt.legend()  
plt.show()

\*\*This is Very Impressive that the company promotes employees of all the ages equally even the freshers have equal share of promotion and also the senior citizen employees are getting the equal share of Promotion in the Company\*\*

# checking which department got most number of promotions  
  
data = pd.crosstab(train['department'], train['is\_promoted'])  
data.div(data.sum(1).astype('float'), axis = 0).plot(kind = 'bar', stacked = True, figsize = (20, 8), color = ['orange', 'lightgreen'])  
  
plt.title('Dependency of Departments in determining Promotion of Employees', fontsize = 30)  
plt.xlabel('Different Departments of the Company', fontsize = 20)  
plt.legend()  
plt.show()

\*\*Again, Each of the departments have equal no. of promotions showing an equal developement in each of the departments of the company.\*\*

# checking dependency of gender over promotion  
  
data = pd.crosstab(train['gender'], train['is\_promoted'])  
data.div(data.sum(1).astype('float'), axis = 0).plot(kind = 'bar', stacked = True, figsize = (7, 5), color = ['pink', 'yellow'])  
  
plt.title('Dependency of Genders in determining Promotion of Employees', fontsize = 30)  
plt.xlabel('Gender', fontsize = 20)  
plt.legend()  
plt.show()

\*\*The above plot shows that there is no partiality between males and females in terms of promotion\*\*  
  
\*\*Data Pre-processing\*\*  
  
# filling missing values  
  
train['education'].fillna(train['education'].mode()[0], inplace = True)  
train['previous\_year\_rating'].fillna(1, inplace = True)  
  
# again checking if there is any Null value left in the data  
train.isnull().sum().sum()  
  
# filling missing values  
  
test['education'].fillna(test['education'].mode()[0], inplace = True)  
test['previous\_year\_rating'].fillna(1, inplace = True)  
  
# again checking if there is any Null value left in the data  
test.isnull().sum().sum()  
  
# removing the employee\_id column  
  
train = train.drop(['employee\_id'], axis = 1)  
  
train.columns

saving the employee\_id  
  
emp\_id = test['employee\_id']  
  
# removing the employee\_id column  
  
test = test.drop(['employee\_id'], axis = 1)  
  
test.columns  
  
# defining the test set  
  
x\_test = test  
  
x\_test.columns  
  
# one hot encoding for the test set  
  
x\_test = pd.get\_dummies(x\_test)  
  
x\_test.columns  
  
# splitting the train set into dependent and independent sets  
  
x = train.iloc[:, :-1]  
y = train.iloc[:, -1]  
  
print("Shape of x:", x.shape)  
print("Shape of y:", y.shape)  
  
# one hot encoding for the train set  
  
x = pd.get\_dummies(x)  
  
x.columns  
  
\*\*Oversampling of the Model\*\*