

## Perform the following steps:

1. Perform data quality check by checking for missing values if any.

```
.ipynb
File Settings Help
Python 3 (ipykernel)

[1]: import pandas as pd

[2]: df = pd.read_excel('1673872196_hr_comma_sep.xlsx')

[3]: df.head()

[4]:
satisfaction_level  last_evaluation  number_project  average_monthly_hours  time_spent_company  Work_accident  left  promotion_last_5years  sales  salary
0      0.38      0.53      2      157      157      0      0      1      0  sales  low
1      0.80      0.86      5      262      262      0      0      1      0  sales  medium
2      0.11      0.88      7      272      272      0      0      1      0  sales  medium
3      0.72      0.87      5      223      223      0      0      1      0  sales  low
4      0.37      0.52      2      159      159      0      0      1      0  sales  low

[5]: df.tail()

[6]:
satisfaction_level  last_evaluation  number_project  average_monthly_hours  time_spent_company  Work_accident  left  promotion_last_5years  sales  salary
14994      0.40      0.57      2      151      151      0      0      1      0  support  low
14995      0.37      0.48      2      160      160      0      0      1      0  support  low
14996      0.37      0.53      2      143      143      0      0      1      0  support  low
14997      0.11      0.96      6      280      280      0      0      1      0  support  low
14998      0.37      0.52      2      158      158      0      0      1      0  support  low

[7]: Missing_values = df.isna()

[8]:
for column in Missing_values.columns.values.tolist():
    print(column)
    print(Missing_values[column].value_counts())
    print("")

satisfaction_level
False      14999
Name: satisfaction_level, dtype: int64

In [ ]: 
```

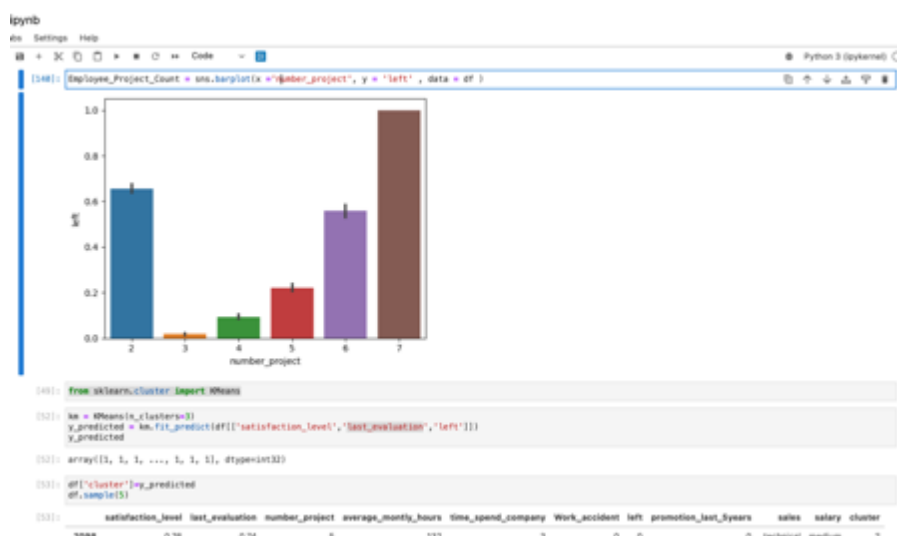
2. Understand what factors contributed most to employee turnover by EDA.
  - 2.1. Draw a heatmap of the Correlation Matrix between all numerical features/columns in the data.



- 2.2. Draw the distribution plot of
  - Employee Satisfaction (use column satisfaction\_level)
  - Employee Evaluation (use column last\_evaluation)
  - Employee Average Monthly Hours (use column average\_monthly\_hours)



2.3. Draw the bar plot of Employee Project Count of both employees who left and who stayed in the organization (use column number\_project and hue column left) and give your inferences from the plot.

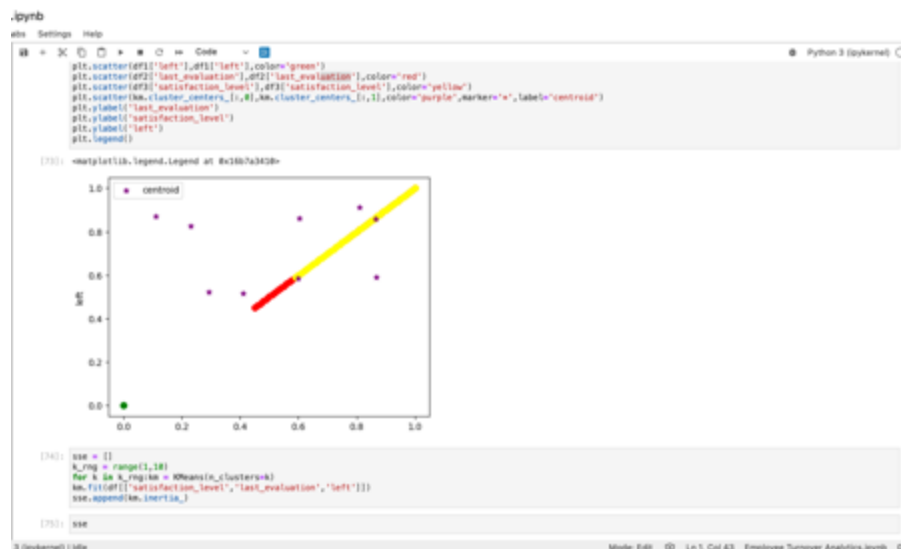
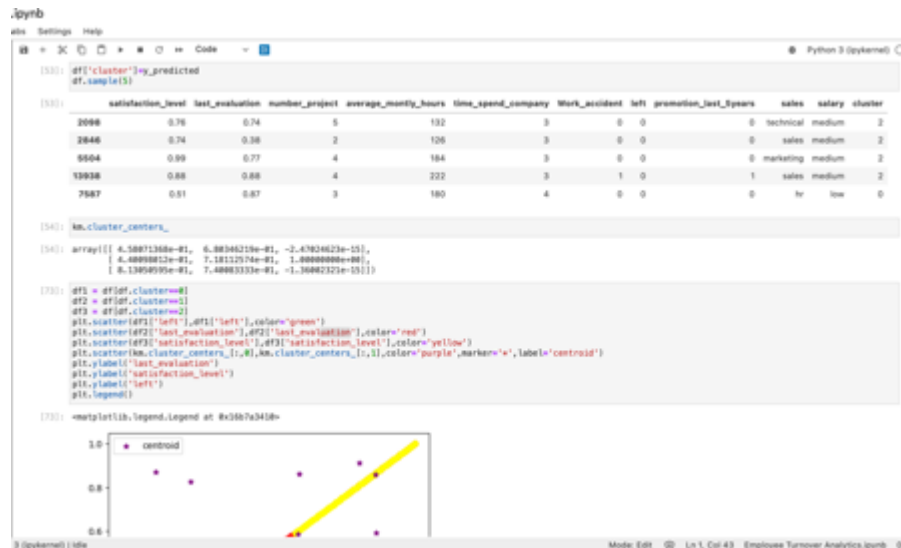


3. Perform clustering of Employees who left based on their satisfaction and evaluation.

3.1. Choose columns satisfaction\_level, last\_evaluation and left.

3.2. Do KMeans clustering of employees who left the company into 3 clusters.

3.3. Based on the satisfaction and evaluation factors, give your thoughts on the employee clusters.



4. Handle the left Class Imbalance using SMOTE technique.

4.1. Pre-Process the data by converting categorical columns to numerical columns by

- Separating categorical variables and numeric variables.
- Applying get\_dummies() to the categorical variables.
- Combining categorical variables and numeric variables.

```

.ipynb
File Settings Help
Python 3 (ipykernel)

[75]: sse

[75]: [179.20903681539693]

[79]: dummy_variable_1 = pd.get_dummies(df[['sales', 'salary']])
      dummy_variable_1.head()

[79]: _Rand0 sales_accounting sales_hr sales_management sales_marketing sales_product_mng sales_sales sales_support sales_technical salary_high salary_low salary_medium
0 0 0 0 0 0 0 0 1 0 0 0 0 1 0
1 0 0 0 0 0 0 1 0 0 0 0 0 0 1
2 0 0 0 0 0 0 1 0 0 0 0 0 1 0
3 0 0 0 0 0 0 1 0 0 0 0 1 0 0
4 0 0 0 0 0 0 1 0 0 0 0 1 0 0

[81]: dummy_variable_1.rename(columns={'sales':'SALES_B', 'salary':'SALARY_B'}, inplace=True)
      dummy_variable_1.head()

[81]: _Rand0 sales_accounting sales_hr sales_management sales_marketing sales_product_mng sales_sales sales_support sales_technical salary_high salary_low salary_medium
0 0 0 0 0 0 0 0 1 0 0 0 0 1 0
1 0 0 0 0 0 0 1 0 0 0 0 0 0 1
2 0 0 0 0 0 0 1 0 0 0 0 0 0 1
3 0 0 0 0 0 0 1 0 0 0 0 1 0 0
4 0 0 0 0 0 0 1 0 0 0 0 1 0 0

[82]: df = pd.concat([df, dummy_variable_1], axis=1)

[84]: df.drop(['sales', 'salary'],
          axis = 1, inplace=True)

[85]: df.head()

[85]: satisfaction_level last_evaluation number_project average_monthly_hours time_spent_company Work_accident left promotion_last_5years cluster sales_IT ... sales_hr sale
3 0.38 0.53 2 157 3 0 1 0 1 0 ... 0
1 0.80 0.86 5 262 6 0 1 0 1 0 ... 0
2 0.11 0.88 7 272 4 0 1 0 1 0 ... 0
3 0.72 0.87 5 223 5 0 1 0 1 0 ... 0
4 0.17 0.63 7 169 5 0 1 0 1 0 ... 0

```

4.2. Do the stratified split of the dataset to train and test in the ratio 80:20 with random\_state=123.

```

.ipynb
File Settings Help
Python 3 (ipykernel)

[96]: df.tail()

[96]: satisfaction_level last_evaluation number_project average_monthly_hours time_spent_company Work_accident left promotion_last_5years cluster sales_IT ... sales_hr
14994 0.40 0.57 2 151 3 0 1 0 1 0 ... 0
14995 0.37 0.48 2 160 3 0 1 0 1 0 ... 0
14996 0.37 0.53 2 143 3 0 1 0 1 0 ... 0
14997 0.11 0.96 6 280 4 0 1 0 1 0 ... 0
14998 0.37 0.52 2 158 3 0 1 0 1 0 ... 0
5 rows x 22 columns

[146]: from sklearn.model_selection import train_test_split
      X=df.drop(['left'],axis=1)
      y=df['left']
      X_train, X_test, y_train, y_test = train_test_split(X, y,
      test_size=0.2, random_state=123)

[133]: import numpy as np

[147]: from imblearn.over_sampling import SMOTE
      sm = SMOTE(random_state = 123)
      X_res, y_res = sm.fit_resample(X_train,y_train)

[148]: print(X_test.head())

satisfaction_level last_evaluation number_project \
6958 0.34 0.67 3
7534 0.72 0.52 3
2975 0.95 0.61 3
3983 0.78 0.79 3
8437 0.68 0.48 3

average_monthly_hours time_spent_company Work_accident \
6958 154 2 0
7534 143 4 1
2975 267 2 0
3983 263 2 0
8437 146 4 1

promotion_last_5years cluster sales_IT sales_Rand0 ... sales_hr \
6958 0 2 0 0 ... 0
7534 0 2 0 1 ... 0
2975 0 2 0 1 ... 0
3983 0 2 0 0 ... 0
8437 0 2 0 0 ... 0

```

4.3. Upsample the train dataset using SMOTE technique from the imblearn module.

```

.ipynb
View Settings Help
[347]: from sklearn.over_sampling import SMOTE
sm = SMOTE(random_state = 123)
X_res, y_res = sm.fit_resample(X_train, y_train)

[329]: print(X_train.head())

satisfaction_level  last_evaluation  number_project  \
2053      0.31      0.54      6
2112      0.59      0.81      4
1794      0.58      0.51      2
13885     0.95      0.77      5
11258     0.58      0.77      3

average_monthly_hours  time_spent_company  Work_accident  \
2053      283      288      0
2112      288      288      0
1794      159      199      0
13885     295      285      0
11258     285      285      0

promotion_last_5years  cluster  sales_if  sales_RandD  ...  sales_hr  \
2053      0      0      0      0  ...      0
2112      0      0      0      0  ...      0
1794      0      1      0      0  ...      0
13885     0      2      0      0  ...      0
11258     0      0      1      0  ...      0

sales_management  sales_marketing  sales_product_mng  sales_sales  \
2053      0      0      0      0
2112      0      0      0      0
1794      0      0      0      0
13885     0      0      0      1
11258     0      0      0      1

sales_support  sales_technical  salary_high  salary_low  salary_medium
2053      0      1      0      1      0
2112      1      0      0      1      0
1794      0      1      0      1      0
13885     0      0      0      0      1
11258     0      0      0      0      1

[5 rows x 21 columns]

[338]: np.bincount(y_res)

```

5. Perform 5-Fold cross-validation model training and evaluate performance.
  - 5.1. Train a Logistic Regression model and apply a 5-Fold CV and plot the classification report.
  - 5.2. Train a Random Forest Classifier model and apply the 5-Fold CV and plot the classification report.
  - 5.3. Train a Gradient Boosting Classifier model and apply the 5-Fold CV and plot the classification report.

```

.ipynb
View Settings Help
[284]: from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()

[236]: Left_model_1 = lr.fit(X_train, y_train)

[237]: Left_model_1.score(X_train, y_train)

[237]: 0.8353196899674973

[238]: from sklearn.model_selection import cross_validate

[239]: cross_validate(Left_model_1, X_train, y_train, cv=5)

[239]: {'fit_time': array([0.46121597, 0.42563185, 0.53964995, 0.37851814, 0.43248297]),
'score_time': array([0.00812793, 0.00117922, 0.008885 , 0.00522876, 0.00904488]),
'test_score': array([0.83625 , 0.795 , 0.84125 , 0.8275 , 0.82489337])}

[240]: print(classification_report(y_train, Left_model_1.predict(X_train)))

precision    recall  f1-score   support

0           0.86     0.94     0.90     9137
1           0.73     0.49     0.59     2862

accuracy          0.79
macro avg         0.79     0.72     0.74     12000
weighted avg      0.83     0.64     0.82     12000

[241]: from sklearn.ensemble import GradientBoostingClassifier

[242]: Left_model_2 = GradientBoostingClassifier(random_state=123)
Left_model_2.fit(X_train, y_train)

[242]: GradientBoostingClassifier
GradientBoostingClassifier(random_state=123)

[243]: cross_validate(Left_model_2, X_train, y_train, cv=5)

[243]: {'fit_time': array([0.73262119, 0.68878912, 0.65535519, 0.82621789, 0.70465779]),
'score_time': array([0.00467873, 0.00484681, 0.00324678, 0.00623512, 0.0042181 ]),
'test_score': array([1., 1., 1., 1., 1.])}

[244]: print(classification_report(y_train, Left_model_2.predict(X_train)))

precision    recall  f1-score   support

0           1.00     1.00     1.00     9137

```

6. Identify the best model and justify the evaluation metrics used.
  - 6.1. Find the ROC/AUC for each model and plot the ROC curve.

```

.ipynb
File Edit View Help
Python 3 (Spyder)

weighted avg 1.00 1.00 1.00 13999

[243]: from sklearn.ensemble import RandomForestClassifier

[244]: Left_model_3 = RandomForestClassifier(max_depth=2, random_state=0)
Left_model_3.fit(X_train, y_train)

[245]:
RandomForestClassifier
RandomForestClassifier(max_depth=2, random_state=0)

[247]: print(classification_report(y_train, Left_model_3.predict(X_train)))

precision recall f1-score support
0 0.89 1.00 0.94 9137
1 1.00 0.62 0.77 2862

accuracy 0.95
macro avg 0.95 0.80 0.86 13999
weighted avg 0.92 0.91 0.90 13999

[249]: from sklearn.metrics import roc_auc_score

[250]: roc_auc_score(y_train, Left_model_3.predict_proba(X_train)[:, 1])

[251]: 0.819058922862763

[252]: roc_auc_score(y_train, Left_model_2.predict_proba(X_train)[:, 1])

[253]: 1.0

[254]: roc_auc_score(y_train, Left_model_1.predict_proba(X_train)[:, 1])

[255]: 0.9627918458378412

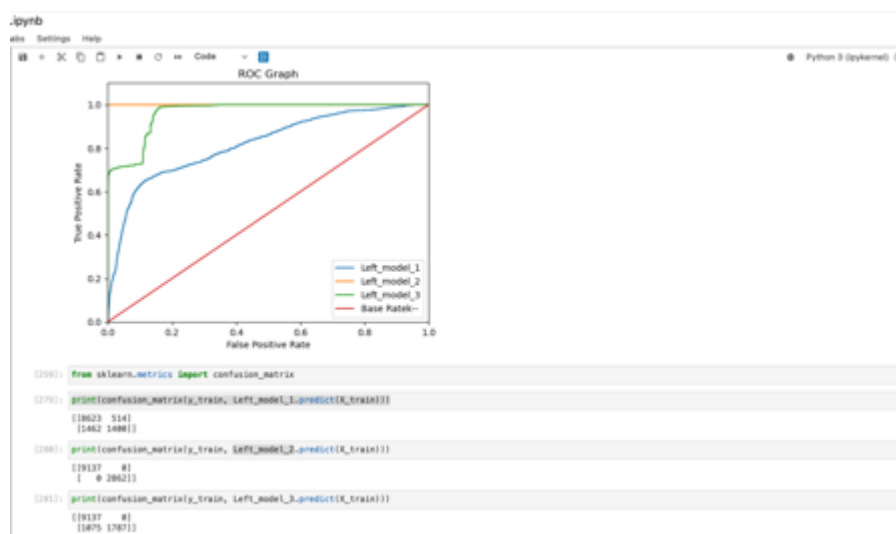
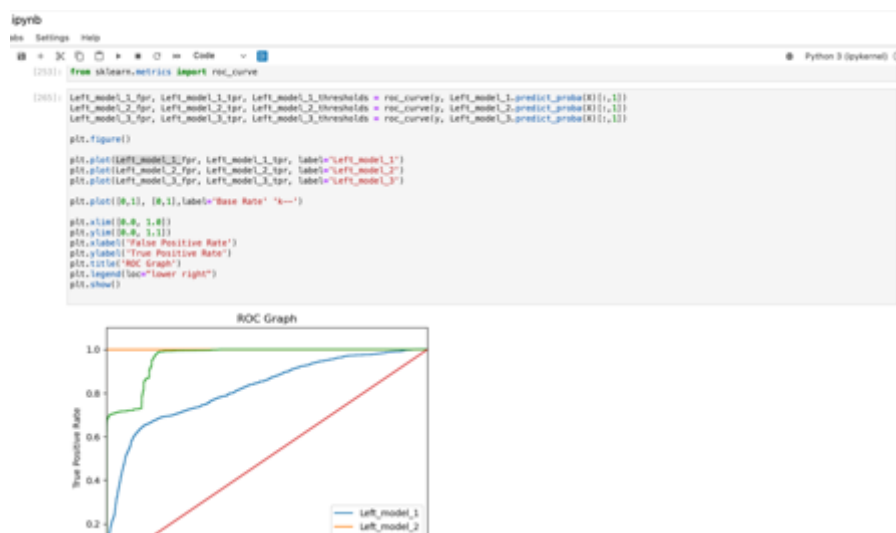
[256]: from sklearn.metrics import roc_curve

[257]: Left_model_1_fpr, Left_model_1_tpr, Left_model_1_thresholds = roc_curve(y_train, Left_model_1.predict_proba(X_train)[:, 1])
Left_model_2_fpr, Left_model_2_tpr, Left_model_2_thresholds = roc_curve(y_train, Left_model_2.predict_proba(X_train)[:, 1])
Left_model_3_fpr, Left_model_3_tpr, Left_model_3_thresholds = roc_curve(y_train, Left_model_3.predict_proba(X_train)[:, 1])

```

6.2. Find the confusion matrix for each of the models.

6.3. From the confusion matrix, explain which metric needs to be used- Recall or Precision?



7. Suggest various retention strategies for targeted employees.
  - 7.1. Using the best model, predict the probability of employee turnover in the test data.

```

.pythb
File Settings Help
Python 3 (ipykernel)

[[9137 0]
 [1875 1787]]

(2942) Test_predict = Left_model_2.predict(X_test)
(2952) print(Test_predict)
[0 0 0 ... 1 0 0]

(2972) Test_predict.shape
(2972) (3000,)

(3012) print(classification_report(y_test, Left_model_2.predict(X_test)))
              precision    recall  f1-score   support

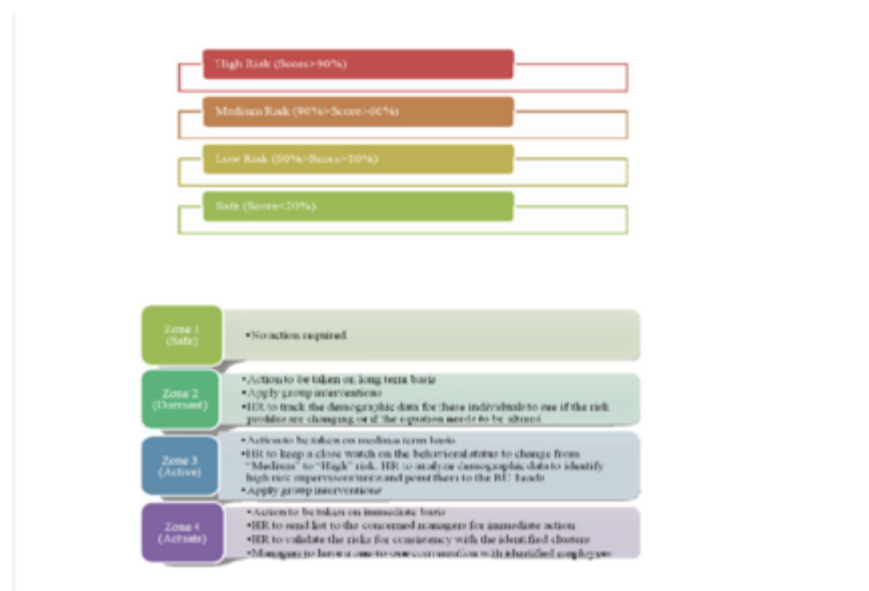
     0       1.00      1.00      1.00       2291
     1       1.00      1.00      1.00        709

 accuracy: 1.00
 macro avg: 1.00
 weighted avg: 1.00

(3022) Left_model_2.score(X_test, y_test)
(3022) 1.0
(3042) np.array(y_test)
(3042) array([0, 0, 0, ..., 1, 0, 0])
(3052) Test_predict = Left_model_2.predict(X_test)
np.mean(Test_predict == y_test)
(3052) 1.0
1 0

```

- 7.2. Based on the below probability score range, categorize the employees into four zones and suggest your thoughts on the retention strategies for each zone.
  - Safe Zone (Green) (Score < 20%)
  - Low Risk Zone (Yellow) (20% < Score < 60%)
  - Medium Risk Zone (Orange) (60% < Score < 90%)
  - High Risk Zone (Red) (Score > 90%).



## The rationale behind designing the problem statement on Talent Management:

*\*Note to Simplilearn:*