Jobathon Challenge Approach

Name: Shrey Arora

Email ID: shreyarora150@gmail.com

Mobile: +91 9878790868

Brief Approach

The challenge given is a classification problem where we had to predict whether an employee will leave the company in the coming quarters or not.

Two approaches came to my mind: -

1. **Approach#1 (Standard Classification)**
   Train a classification model on the given training data and using the model to predict the attrition for test data (by taking intersection of employee IDs in test data with existing training data).

2. **Approach#2 (Regression +Classification)**
   Train a classification model on the given training data but, Instead of taking intersection of employee IDs in test data with existing training data, we will estimate the values of Total Business Value and Quarterly rating for each employee for each reporting month from Jan 2018 to June 2018 by training a regression model and then use these predicted values to create a

new data set for testing and predicting the attrition on this data set instead.

Due to time constraints and desired accuracy not attained through approach 2, I had to stick to approach 1 instead.

Data Pre-processing/ Feature engineering

I carried out the following data preprocessing and feature engineering techniques: -

1. Converted 'Dateofjoining', 'MMM-YY' and 'LastWorkingDate' to datetime from object data type

```
47
48 train['Dateofjoining'] = pd.to_datetime(train['Dateofjoining'], errors='raise')
49 train['MMM-YY'] = pd.to_datetime(train['MMM-YY'], errors='raise')
50 train['LastWorkingDate'] = pd.to_datetime(train['LastWorkingDate'], errors='raise')
51
52
```

2. Extract the last reporting month for each employee.

   I wanted to reduce the unnecessary data to remove skewed from the training data because out of 19104 rows in training data only 1616 rows had last reporting month.

There was a trade off in only considering last reporting months data and reducing the training set from 19104 to 2381 rows but the scores for latter was better than the former.

```
66 def last_reporting_month(row):
67     return train[train['Emp_ID']==row['Emp_ID']]['MMM-YY'].max()
68
69 train['last_reporting_month'] = train.apply(lambda x: last_reporting_month(x),axis =1)
70 train1 = train[train['MMM-YY']==train['last_reporting_month']]
71
```

3. Adding new features

the Total Business Value and Quarterly Rating provides information only for that particular month or quarter, but since I had reduced the training set for only last reporting month, I added Average Business Value and Average Quarterly Rating to add information about past performances as well.

```
70 def total_business_value(row):
71     return train[train['Emp_ID']==row['Emp_ID']]['Total Business Value'].mean()
72
73 def avergae_quaterly_rating(row):
74      return train[train['Emp_ID']==row['Emp_ID']]['Quarterly Rating'].mean()
75
76 train['Net_Business_Value'] = train.apply(lambda x: total_business_value(x),axis =1)
77 train['avergae_quaterly_rating'] = train.apply(lambda x: avergae_quaterly_rating(x),axis =1)
78
```

I added a Target variable Y- categorical (1/0) based on the last working date column

```
77 def attrition(row):
78     if pd.isnull(row['LastWorkingDate']):
79         return 0
80     else :
81         return 1
82
83 train1['Y'] = train1.apply(lambda x: attrition(x),axis=1)
```

I added a new feature to calculate total months from joining date called 'Months_from_joining'

```
train1['Months_from_joining'] = (( train1['MMM-YY'] -  train1['Dateofjoining'])/np.timedelta64(1, 'M'))
train1['Months_from_joining'] = train1['Months_from_joining'].astype(int)
```

4. Encoding

I encoded the categorical variables like gender, city education level to numeric values

```
le = LabelEncoder()
var_mod = train1.select_dtypes(include='object').columns
for i in var_mod:
    train1[i] = le.fit_transform(train1[i])
```

5. Feature selection

I dropped the following variables because of redundancy and checking accuracies by hit and trial method

['Emp_ID', 'LastWorkingDate','Y','MMM-YY', 'last_reporting_month', 'Dateofjoining']

Model Selection

I trained the following models on the training data

- Decision Tree Regressor
- Random Forest Classifier
- Gradient Boosting Classifier
- XGB Classifier

I evaluated the F1 macro score on K-fold (k=5) cross validation and compared the mean scores.

```
## Model selection
scores = pd.DataFrame()
models = [DecisionTreeRegressor(random_state = 0),RandomForestClassifier(n_estimators = 100) ,
          GradientBoostingClassifier(),XGBClassifier()]
for m in models:
    regressor = m
    s = cross_val_score(regressor, X, y, cv=5, scoring='f1_macro')
    #regressor.fit(X_train, y_train)
    #  y_pred = regressor.predict(X_valid).astype(int)

    print(m)
    #y = f1_score(y_valid, y_pred, average='macro')*100
    print(s)
    scores = scores.append({"model":m,"avg score":s.mean()},ignore_index= True)
```

By then comparing Gradient Boost had the best score and was considered as final model

avg score	model
0.749216	Decision
0.823903	RandomF
0.829272	Gradient
0.807705	XGBClass