HR Analytics: - Predicting Job change of a Data Scientist

Milestone: Project Report

Group 16

Nitesh Gupta

617-784-2099

gupta.nite@northeastern.edu

Percentage of effort contribution by student: - 100

Signature of Student 1: Nitesh Gupta

Submission Date: 04/23/2021

Problem Setting

A company which is active in Big Data and Data Science wants to hire data scientists among people who successfully pass some courses which conduct by the company. Many people signup for their training. Company wants to know which of these candidates are really wants to work for the company after training or looking for a new employment because it helps to reduce the cost and time as well as the quality of training or planning the courses and categorization of candidates. Information related to demographics, education, experience is in hands from candidate's signup and enrollment.
This dataset designed to understand the factors that lead a person to leave current job for HR research too. By model(s) that uses the current credentials, demographics, experience data you will predict the probability of a candidate to look for a new job or will work for the company, as well as interpreting affected factors on employee decision.

The whole data divided to train and test. Target isn't included in test, but the test target values data file is in hands for related tasks. A sample submission correspond to enrollee id of test set provided too with columns: enrollee _id , target

Note:

- The dataset is imbalanced.
- Most features are categorical (Nominal, Ordinal, Binary), some with high cardinality.
- Missing imputation can be a part of pipeline as well.

Problem Definition

- Predict the probability of a candidate will work for the company
- Interpret model(s) such a way that illustrate which features affect candidate decision

Data Sources

Dataset: - HR analytics: Job change of a Data scientist.

Data source: - **https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data-scientists**

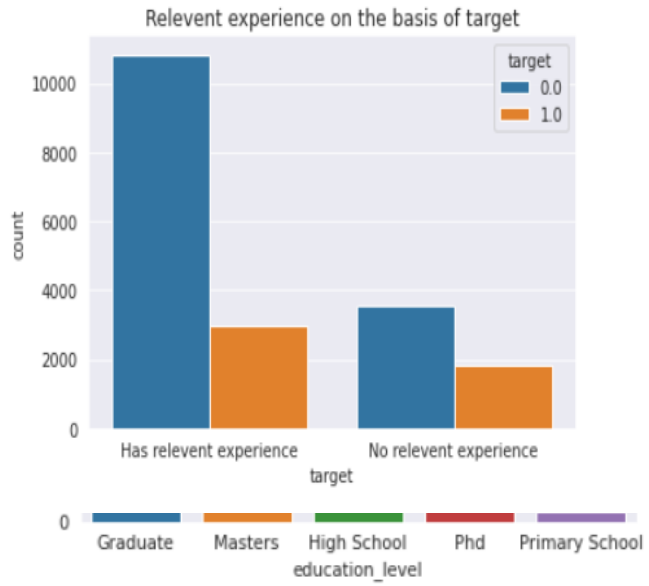
Data Description

The dataset contains 3 .csv files (Train, Test, Test values) with 14 variables and 25,964 rows of data. Sample of few variable names are given below:

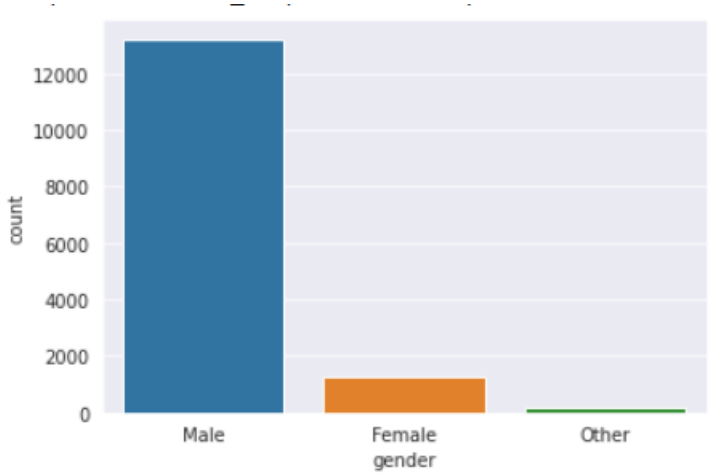
Features

- enrollee id: Unique ID for candidate
- city: City code
- city_ development _index: Development index of the city (scaled)
- gender: Gender of candidate
- relevant experience: Relevant experience of candidate
- enrolled university: Type of University course enrolled if any
- education level: Education level of candidate
- major discipline: Education major discipline of candidate
- experience: Candidate total experience in years
- company size: No of employees in current employer's company
- company type: Type of current employer
- last *new* job: Difference in years between previous job and current job
- training hours: training hours completed.
- target: 0 – Not looking for job change, 1 – Looking for a job change

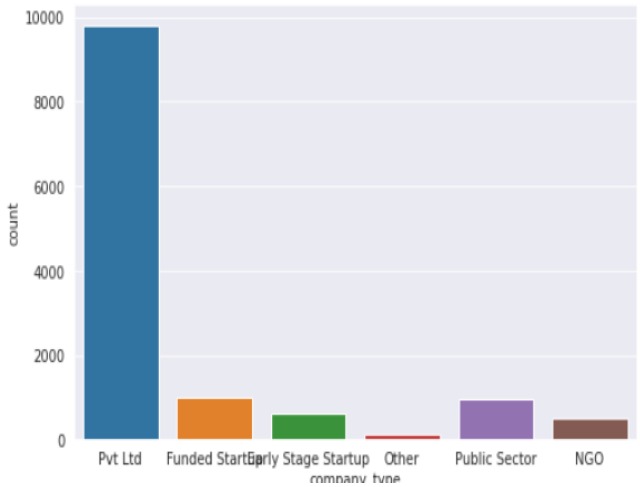
Data Exploration



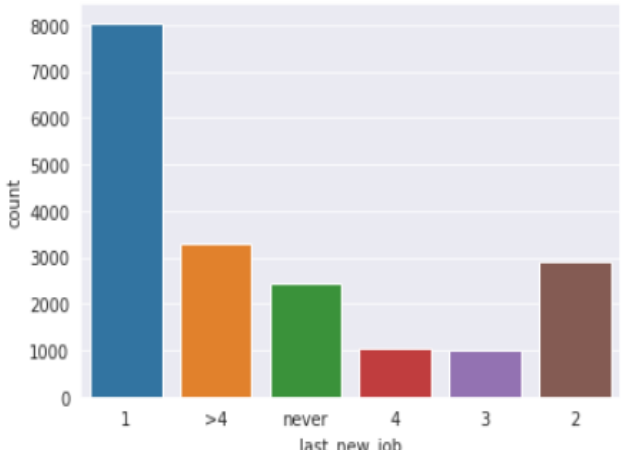
As we can see in the vertical bar plot, the most number of data scientist are graduated from college.



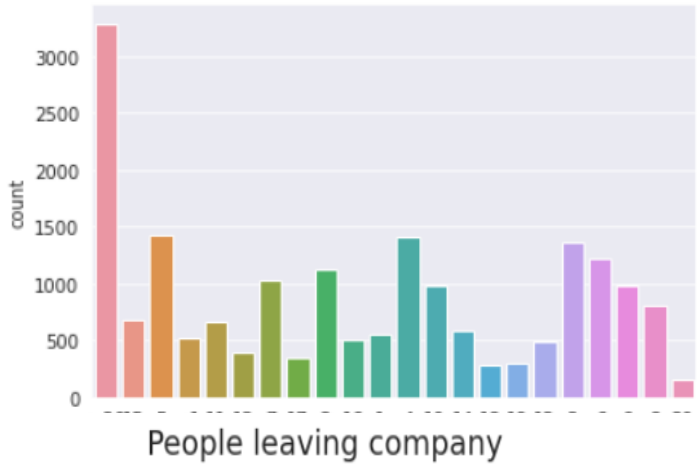
As we can see in the plot that the greatest number of data scientist are male.



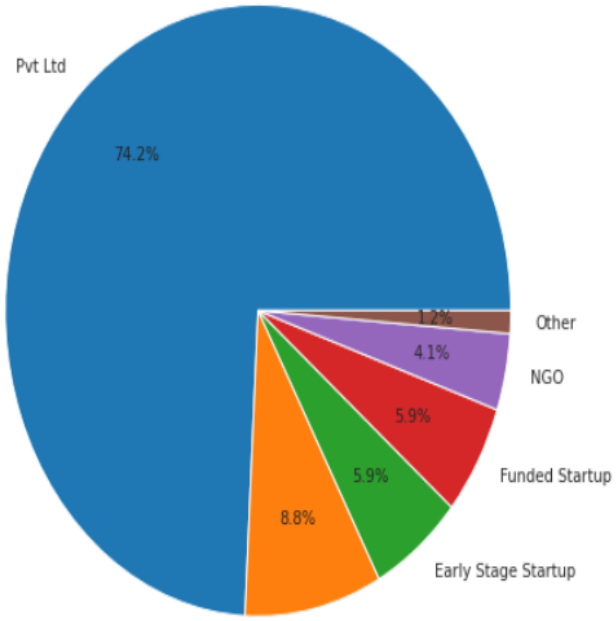
As we can see that the most of the data scientist had worked for a Pvt. LTD company



As we can see that the for the most of data scientist the difference between last job and new job is 1 year



As we can see most of the data scientist has experience greater than 20 years.



As we can see that the most of the data scientist had worked for a Pvt. LTD company

Data Mining Tasks

1. First, we are going to drop unnecessary columns, so we don't require enrolle_id and city column

	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	last_new_job	training_hours	target
0	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	NaN	NaN	1	36	1.0
1	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15	50-99	Pvt Ltd	>4	47	0.0
2	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5	NaN	NaN	never	83	0.0
3	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	<1	NaN	Pvt Ltd	never	52	1.0
4	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	>20	50-99	Funded Startup	4	8	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
19153	0.878	Male	No relevent experience	no_enrollment	Graduate	Humanities	14	NaN	NaN	1	42	1.0
19154	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	14	NaN	NaN	4	52	1.0
19155	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	50-99	Pvt Ltd	4	44	0.0
19156	0.802	Male	Has relevent experience	no_enrollment	High School	NaN	<1	500-999	Pvt Ltd	2	97	0.0
19157	0.855	NaN	No relevent experience	no_enrollment	Primary School	NaN	2	NaN	NaN	1	127	0.0

19158 rows × 12 columns

2) dealing with null values

```
city_development_index     0.000000
gender                    23.530640
relevent_experience        0.000000
enrolled_university        2.014824
education_level            2.401086
major_discipline          14.683161
experience                 0.339284
company_size              30.994885
company_type              32.049274
last_new_job               2.207955
training_hours             0.000000
target                     0.000000
dtype: float64
```

3) The columns in which we have 2% or less than 2% null values we can drop those null values

```
[88] # after dropping those null values
     df.shape

     (18014, 12)
```

4) Now we are going to fill null values with their mode as all the columns left have data type as 'object'

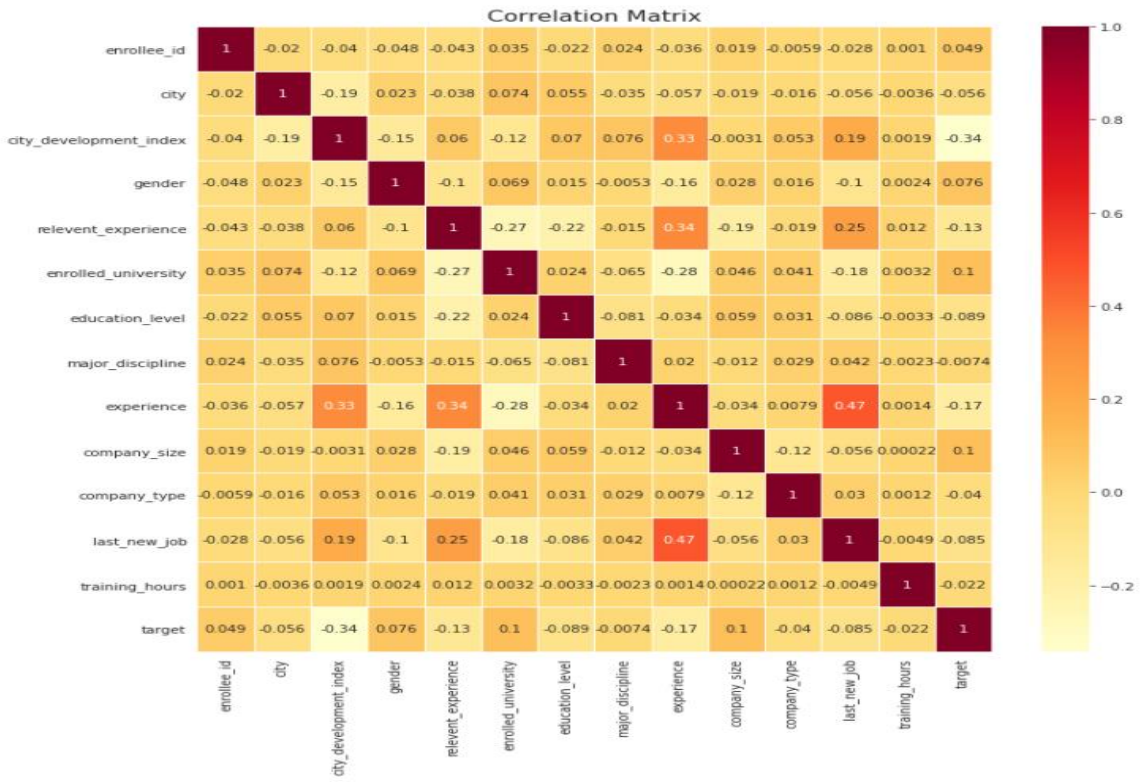
```
[88] # after dropping those null values
     df.shape

     (18014, 12)
```

5) We changed the data type of experience and last_new_job column

6) To handle categorical Data by adding dummy variable

	city_development_index	experience	last_new_job	training_hours	target	High_School	Masters	Phd	Primary_School	around_100	around_1000	around_10000	around_50	around_500	around_5000	more_than_10
0	0.920	21	1	36	1.0	0	0	0	0	1	0	0	0	0	0	
1	0.776	15	5	47	0.0	0	0	0	0	1	0	0	0	0	0	
2	0.624	5	0	83	0.0	0	0	0	0	1	0	0	0	0	0	
4	0.767	21	4	8	0.0	0	1	0	0	1	0	0	0	0	0	
5	0.764	11	1	24	1.0	0	0	0	0	1	0	0	0	0	0	

7) To check for the correlation between the variable by assuming cut-off value=0.5



8) To check for multicollinearity by using VIF function

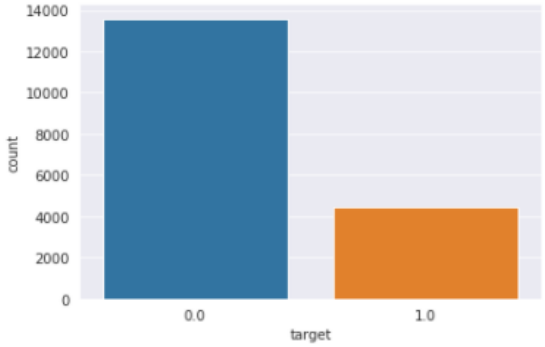
	experience	last_new_job	training_hours	High_School	Masters	Phd	Primary_School	around_100	around_1000	around_10000	around_50	around_500	around_5000	more_than_10000	Funded_Startup	NGO	Ot
0	21	1	36	0	0	0	0	1	0	0	0	0	0	0	0	0	
1	15	5	47	0	0	0	0	1	0	0	0	0	0	0	0	0	
2	5	0	83	0	0	0	0	1	0	0	0	0	0	0	0	0	
4	21	4	8	0	1	0	0	1	0	0	0	0	0	0	1	0	
5	11	1	24	0	0	0	0	1	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19153	14	1	42	0	0	0	0	1	0	0	0	0	0	0	0	0	
19154	14	4	52	0	0	0	0	1	0	0	0	0	0	0	0	0	
19155	21	4	44	0	0	0	0	1	0	0	0	0	0	0	0	0	
19156	0	2	97	1	0	0	0	0	1	0	0	0	0	0	0	0	
19157	2	1	127	0	0	0	1	1	0	0	0	0	0	0	0	0	

18014 rows × 25 columns

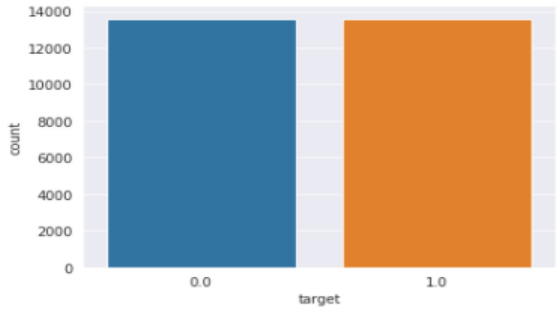
9) Let us check data is balanced or not

```
In [50]: sns.countplot(df['target'])
```

```
Out[50]: <AxesSubplot:xlabel='target', ylabel='count'>
```



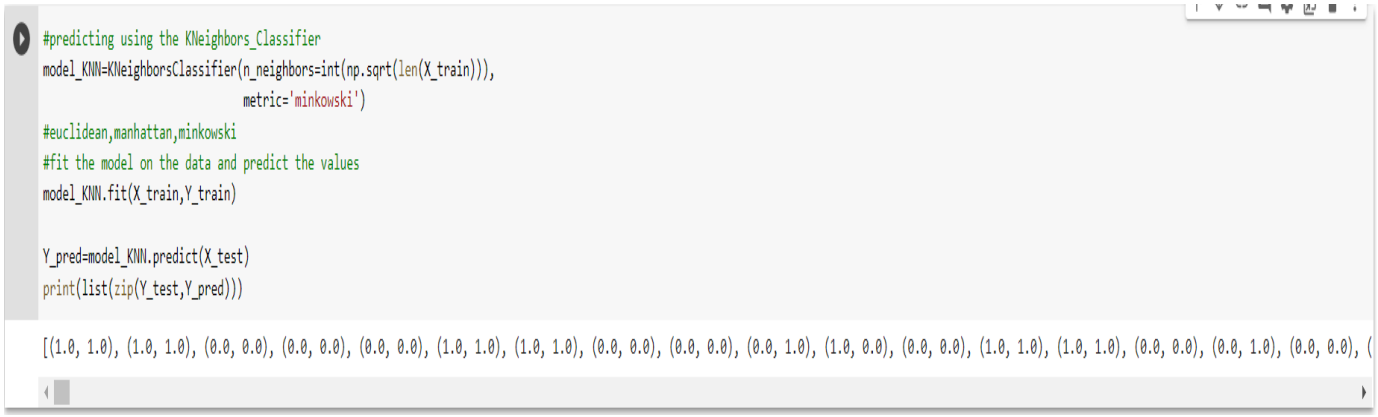
As we can see the data is not balanced. The target variable 1 is only 25% of the target variable 0. Hence, we are going to do oversample using SMOTE technique.



10) After that we used standard scaler function to standardize the data and then we split the data into 80% training set and 20% test set.

Data Mining Models/Methods

K-NN Classifier: - The first model we have used is the **K-NN** models because it is easy to implement, works well on small datasets, and the quick calculation time. It converts all the variable into binary values and calculate the distance between them.



```
#predicting using the KNeighbors_Classifier
model_KNN=KNeighborsClassifier(n_neighbors=int(np.sqrt(len(X_train))),
                               metric='minkowski')
#euclidean,manhattan,minkowski
#fit the model on the data and predict the values
model_KNN.fit(X_train,Y_train)

Y_pred=model_KNN.predict(X_test)
print(list(zip(Y_test,Y_pred)))
```

```
[(1.0, 1.0), (1.0, 1.0), (0.0, 0.0), (0.0, 0.0), (0.0, 0.0), (1.0, 1.0), (1.0, 1.0), (0.0, 0.0), (0.0, 0.0), (0.0, 1.0), (1.0, 0.0), (0.0, 0.0), (1.0, 1.0), (1.0, 1.0), (0.0, 0.0), (0.0, 1.0), (0.0, 0.0), (
```

So, this is the table of Accuracy of k-NN predictions in training set for various choices of k. At k=16 we found the highest accuracy.

```
For k value: 1 MSE value is: 0.1851783744023538
For k value: 2 MSE value is: 0.1613184994483266
For k value: 3 MSE value is: 0.1550202280250092
For k value: 4 MSE value is: 0.1562959727841118
For k value: 5 MSE value is: 0.15499080544317767
For k value: 6 MSE value is: 0.15475971558170898
For k value: 7 MSE value is: 0.15425839331687072
For k value: 8 MSE value is: 0.15539088359691064
For k value: 9 MSE value is: 0.15551287464981228
For k value: 10 MSE value is: 0.1557374034571534
For k value: 11 MSE value is: 0.1568743370040638
For k value: 12 MSE value is: 0.15711326672387724
For k value: 13 MSE value is: 0.1574619541207936
For k value: 14 MSE value is: 0.15762003587753598
For k value: 15 MSE value is: 0.15800743737485187
For k value: 16 MSE value is: 0.15807454486943728
For k value: 17 MSE value is: 0.15859115210024038
For k value: 18 MSE value is: 0.15875594240802038
For k value: 19 MSE value is: 0.15910658452522977
For k value: 20 MSE value is: 0.15931408606105185
For k value: 21 MSE value is: 0.1599669412941099
For k value: 22 MSE value is: 0.1602664445788589
For k value: 23 MSE value is: 0.16053487639665143
For k value: 24 MSE value is: 0.16076874157165627
For k value: 25 MSE value is: 0.16142964325119527
For k value: 26 MSE value is: 0.16154999553873575
For k value: 27 MSE value is: 0.16152830939721546
For k value: 28 MSE value is: 0.16166094977895537
For k value: 29 MSE value is: 0.1619210654403176
For k value: 30 MSE value is: 0.16215622573658617
For k value: 31 MSE value is: 0.16212443440559152
For k value: 32 MSE value is: 0.1620971634792203
```

So, this is output we got from the K-NN model. The maximum accuracy we got for this model is 74.5%.

```
[[2104  652]
 [ 731 1951]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.74      0.76      0.75      2756
         1.0       0.75      0.73      0.74      2682

    accuracy                           0.75      5438
   macro avg       0.75      0.75      0.75      5438
weighted avg       0.75      0.75      0.75      5438

Accuracy of the model:  0.7456785582934903
```

Logistic Regression: - We have used the logistic regression model as a second model because it is easy to Implement, interpret, and efficient to train and works well on separable data. The table containing the p-value and the coefficients of variable are shown below.

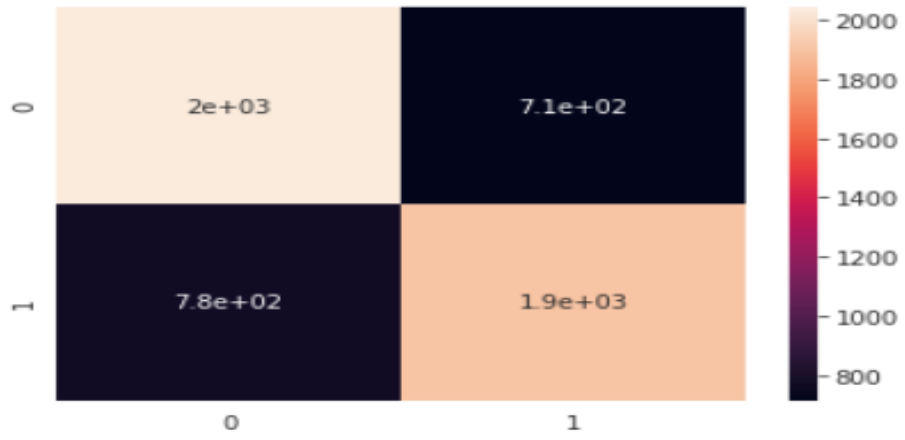
```
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                27186
Model:                          Logit   Df Residuals:                    27156
Method:                           MLE   Df Model:                           29
Date:                Fri, 23 Apr 2021   Pseudo R-squ.:                  0.1672
Time:                        17:08:17   Log-Likelihood:                -15694.
converged:                       True   LL-Null:                       -18844.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -5.1196      0.115    -44.447      0.000      -5.345      -4.894
x2            -0.0196      0.003     -7.374      0.000      -0.025      -0.014
x3             0.0464      0.010      4.703      0.000       0.027       0.066
x4            -0.0006      0.000     -2.381      0.017      -0.001   -9.73e-05
x5            -1.1997      0.054    -22.296      0.000      -1.305      -1.094
x6            -0.2790      0.036     -7.826      0.000      -0.349      -0.209
x7            -0.5751      0.116     -4.943      0.000      -0.803      -0.347
x8            -1.9777      0.141    -14.018      0.000      -2.254      -1.701
x9             1.3746      0.068     20.142      0.000       1.241       1.508
x10            0.1705      0.097      1.750      0.080      -0.020       0.361
x11            0.3882      0.108      3.585      0.000       0.176       0.600
x12            0.6680      0.081      8.208      0.000       0.508       0.827
x13            0.1900      0.077      2.459      0.014       0.039       0.341
x14            0.2027      0.089      2.285      0.022       0.029       0.377
```

This is the output we got from the logistic regression. The accuracy of the model is 72.62%.

```
[[2042  714]
 [ 776 1906]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.72      0.74      0.73      2756
         1.0       0.73      0.71      0.72      2682

    accuracy                           0.73      5438
   macro avg       0.73      0.73      0.73      5438
weighted avg       0.73      0.73      0.73      5438

Accuracy of the model:  0.7260022066936374
```

The confusion matrix of the logistic regression is given as:



Decision Tree: - We have used the decision tree as our 3rd model.

```
[192] #predicting using the Decision_Tree_Classifier
      model_DecisionTree=DecisionTreeClassifier(criterion="gini",random_state=10,splitter="best")
      #fit the model on the data and predict the values
      model_DecisionTree.fit(X_train,Y_train)
      Y_pred=model_DecisionTree.predict(X_test)
      print(Y_pred)
      #print(list(zip(Y_test,Y_pred)))

      [1. 1. 0. ... 1. 1. 0.]
```

This is the confusion matrix and the output we got for Decision tree and the accuracy we got it is 80%

```
[[2182  574]
 [ 513 2169]]
0.8001103346818683
              precision    recall  f1-score   support

         0.0       0.81      0.79      0.80      2756
         1.0       0.79      0.81      0.80      2682

    accuracy                           0.80      5438
   macro avg       0.80      0.80      0.80      5438
weighted avg       0.80      0.80      0.80      5438
```

Random Forest Classifier: - The 4th model which we have applying is the random forest classifier.

```
#predicting using the Random_Forest_Classifier
model_RandomForest=RandomForestClassifier(n_estimators=50, random_state=10)

#fit the model on the data and predict the values
model_RandomForest.fit(X_train,Y_train)

Y_pred=model_RandomForest.predict(X_test)
```

```
[196] #confusion matrix
      print(confusion_matrix(Y_test,Y_pred))
      print(accuracy_score(Y_test,Y_pred))
      print(classification_report(Y_test,Y_pred))
```

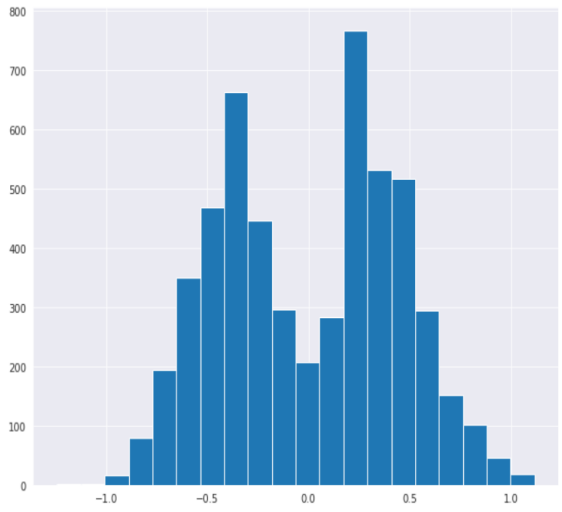
```
[[2411  345]
 [ 510 2172]]
0.8427730783376242
              precision    recall  f1-score   support

         0.0       0.83      0.87      0.85      2756
         1.0       0.86      0.81      0.84      2682

    accuracy                           0.84      5438
   macro avg       0.84      0.84      0.84      5438
weighted avg       0.84      0.84      0.84      5438
```

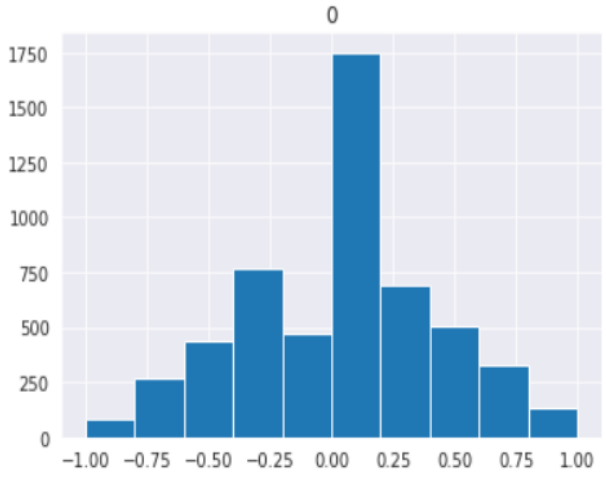
The output of the random forest classifier is 84%.

Performance Evaluation

This is the classification matrix for all four models.

	Precision	Recall	F-1	Accuracy
K-NN	0.75	0.73	0.74	0.7546
Logistic Regression	0.73	0.71	0.72	0.7260
Decision Tree	0.79	0.81	0.80	0.8000
Random Forest Classifier	0.86	0.81	0.84	0.8427





The left histogram plot is the residual plot for logistic regressor, and the right histogram plot is residual plot for K-NN model. The spread of errors in Model 2 is approximately same as that of Model 1. The predictors last new job, company size and company types of coefficients weights are very insignificant as compared to the other predictors. The resultant reduction in error can be attributed to the addition of other predictors in Model 2. The narrow spread of errors from these models indicates the reliability of predictions, hence from Model 1 and Model 2, the Model 2 will be preferable for future predictions.

Project Results

- As we can see that among all four model, the Random forest classifier has the highest accuracy of 84.27%.
- The predictors such as last new job, training hours, company size and company types of coefficients weights are very insignificant as compared to the other predictors.
- Since our aim was not to focus on accuracy but to classify the greatest number of candidates whether they are going to join the company or will look for another employer.
- As we can see the precision and recall of every model is above 0.7 which was our aim, and it shows that the model performs exceptionally well.
- Getting almost the same level of performance as on our training set, this confirms the models has good generalization capacity.
- The precision and recall of these models indicate the reliability of predictions, hence model will be preferable for future predictions.

Impact of the Project Outcomes

- This is critical for large companies with thousands of employees and multiple locations.

- Human resources professionals work to hire the best people for the right position in the shortest amount of time. To do this efficiently, recruiters often use HR analytics to not only find talent, but also to evaluate the recruiting process and quality of candidates.

- This project can help HR teams to monitor employee performance and engagement to build a more productive workforce by knowing which of these candidates are really wants to work for the company after training or looking for a new employment.

- It is going to help to reduce the cost and time as well as the quality of training or planning the courses and categorization of candidates.