HR Analytics: Job Change

Report by

Adina Dingankar

Pranay Bhakthula

Rehapriadarsini Manikandasamy

The George Washington University

Author Note

This report was prepared for DATS 6103, taught by Professor Amir Jafari

TABLE OF CONTENTS

INTRODUCTION

A company which is active in Big Data and Data Science wants to hire data scientists amongst the people who successfully pass some courses which are conducted by the company and signup for the training. Company wants to know which of these candidates are willing to work for the company after training or are looking for a new employment because it helps the firm to reduce the cost and time involved in delivering the quality of training or planning the courses and selecting the candidates. Information related to demographics, education, experience is obtained from the candidate's signup form and enrolment. The dataset is designed to understand the factors that lead a person to leave current job for HR researches too. By model(s) will use the current credentials, demographics, experience data to predict the probability of a candidate to look for a new job or who will continue to work for the company, as well as interpreting factors affecting the employee decision. The project is demonstrated by using three machine learning algorithms: Random Forest Classifier, Decision Tree and Support Vector Machine respectively and develop a GUI based application to display the end-to-end modelling.

DESCRIPTION OF THE DATASET

The dataset used is sourced from Kaggle which has educational and professional records of various candidates who have completed training in a company. The dataset has 19158 observations and 14 features, most features are categorical (Nominal, Ordinal, Binary) and some with high cardinality. The dataset is imbalanced and 8 amongst the 14 features has missing values.

Features are as described:

1. Enrolee_ id: Unique ID for candidate
2. city: City code
3. city_ development _index: Development index of the city (scaled)
4. gender: Gender of candidate
5. relevant_ experience: Relevant experience of candidate
6. enrolled _ university: Type of University course enrolled if any
7. education _ level: Education level of candidate
8. major _ discipline: Education major discipline of candidate
9. experience: Candidate total experience in years
10. company _ size: No of employees in current employer's company
11. company _ type: Type of current employer
12. last _ new _job: Difference in years between previous job and current job
13. training_ hours: training hours completed
14. target: 0 – Not looking for job change, 1 – Looking for a job change

DESCRIPTION OF DATA MINING ALGORITHMS

Decision Tree Classifier

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset.

Entropy: It is defined as a measure of impurity present in the data. The entropy is almost zero when the sample attains homogeneity but is one when it is equally divided. Entropy with the lowest value makes a model better in terms of prediction as it segregates the classes better. Entropy is calculated based on the following formula:

$$Entropy = -p_1 * log_2\, p_1 - \ \ldots - -p_n * log_2\, p_n$$

Here n is the number of classes. Entropy tends to be maximum in the middle with value up to 1 and minimum at the ends with value up to 0.

Gini: It is a measure of misclassification and is used when the data contain multi class labels. Gini is similar to entropy but it calculates much quicker than entropy. Algorithms like CART (Classification and Regression Tree) use Gini as an impurity parameter.

Gini is calculated as:

$$Gini = 1 - \sum_i p_i^2$$ where i is the no of classes.

Random Forest Classifier

The Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction

Moreover, a random forest technique has a capability to focus both on observations and variables of a training data for developing individual decision trees and take maximum voting for classification and the total average for regression problem respectively. It also uses a bagging technique that takes observations in a random manner and selects all columns which are incapable of representing significant variables at the root for all decision trees. In this manner, a random forest makes trees only which are dependent on each other by penalising accuracy. We have a thumb rule which can be implemented for selecting sub-samples from observations using random forest. If we consider 2/3 of observations for training data and p be the number of columns then

1. For classification, we take sqrt(p) number of columns
2. For regression, we take p/3 number of columns.

The above thumb rule can be tuned in case of increasing the accuracy of the model

Random Forest pseudocode:

1. Randomly select “k” features from total “m” features.
    a. Where k << m
2. Among the “k” features, calculate the node “d” using the best split point.
3. Split the node into daughter nodes using the best split.
4. Repeat 1 to 3 steps until “l” number of nodes has been reached.
5. Build forest by repeating steps 1 to 4 for “n” number times to create “n” number of trees.

Support Vector Machine

Support Vector Machine is one such algorithm. It is considered as the black box technique as there are unknown parameters that are not so easy to interpret and assume how it works. It depends on three working principles:

1. Maximum margin classifiers
2. Support vector classifiers
3. Support vector machines

1.Maximum margin classifier

They are often generalized with support vector machines but SVM has many more parameters compared to it. The maximum margin classifier considers a hyperplane with maximum separation width to classify the data. But infinite hyperplanes can be drawn in a set of data. It is indeed important to choose the ideal hyperplane for classification.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 < 0$$

2. Support Vector Classifiers

This type of classifier can be regarded as an extended version of the maximum margin classifier which also deals with the non-separable cases. When we deal with real-life data, we find, most of the observations are in overlapping classes. This is why we use support vector classifiers

3. Support Vector Machine

The support vector machine approach is considered during a non-linear decision and the data is not separable by a support vector classifier irrespective of the cost function.

Kernel Functions

Kernel functions can also be regarded as the tuning parameters in an SVM model. They are responsible for removing the computational requirement to achieve the higher dimensional vector space and deal with the non-linear separable data

1. Polynomial Kernel

A polynomial function is used with a degree 2 to separate the non-linear data by transforming them into higher dimensions. Take a look at the following equation:

$$K(x, x') = (1 + x * x')^k$$

2. Radial Basis Function kernel

This kernel function is also known as the Gaussian kernel function. It is capable of producing an infinite number of dimensions to separate the non-linear data. It depends on a hyperparameter 'γ'(gamma) which needs to be scaled while normalizing the data. The smaller the value of the hyperparameter, the smaller the bias and higher the variance it gives. While a higher value of hyperparameter gives a higher bias and lower variance solutions. It is explained with the help of the following equation:

$$K(x, x') = e(-\gamma \, ||x - x' \, ||^2 \,) ; \;\; \gamma \; = hyperparameter$$

EXPERIMENTAL SETUP

Data Cleaning and Pre-processing:

The HR _Analytics dataset has 14 attributes amongst which the following columns had missing values as stated:

gender :4508

enrolled _university :386

education _level :460

major _discipline: 2813

Experience: 65

Company _size :5938

Company _type: 6140

Last _ new _job :423

In order to remove the missing values fill.na () function has been used for the imputation.

```
data = data.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

Label Encoding:

Label encoding has been applied to the columns: city, gender, relevant _experience, enrolled _university, education _level, major _discipline, experience, company _size, company _type, last _ new _job. Label Encoder is applied to the features as our use case being the classification problem. Encoding was done to decide in a better way on how those labels must be operated and labels are converted into numeric form.

```
class_le1 = LabelEncoder()
features_list1 = self.current_features.loc[:, self.current_features.dtypes ==  'object'].columns
for i in features_list1:
    self.current_features[i] = class_le1.fit_transform(self.current_features[i])
```

EDA Analysis:

To visualize the dataset graphically histogram and scatter plots have been used.

Histogram has been used to understand the distribution of each variables in the dataset.

```
self.current_features.value_counts().plot(kind='bar', ax=self.ax1)
self.ax1.set_title ('Histogram of: ' + x_a)
self.ax1.set_xlabel(x_a)
self.ax1.set_ylabel('frequency')
```

Scatter plots has been used to understand the relation amongst variables.

```
self.ax2.scatter(self.current_x, self.current_y)
self.ax2.set_title ('Scatter plot: ' + self.y_a + ' vs ' + self.x_a)
self.ax2.set_xlabel(self.x_a)
self.ax2.set_ylabel(self.y_a)
```

The selection of more than one variable is restricted by using the following command. This is being done to make sure that there will be no more than one variable plotted for the histogram and not more than two variable selection for the scatter plots

```
def Message_x(self):
    QMessageBox.about(self, "Warning", " You selected more than 1 variable")
```

Decision Tree Classifier:

The packages that have been used for the modelling is Scikit Learn and we have imported decision tree classifier library to perform the functions of the algorithm

```
from sklearn.tree import DecisionTreeClassifier
```

The data is encoded by using Label encoder function provided above to prepare the data for model training. The processed data is then split into train and test split at the ratio of 70:30 by default which is subject to change as per the user's requirements. The model gets trained and tested on the basis of criterions gini and entropy along with the max-depth provided by the user.

GINI model:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=vtest_per, random_state=100)

self.clf_df_gini = DecisionTreeClassifier(criterion="gini", random_state=100,
max_depth=vmax_depth)

self.clf_df_gini.fit(X_train, y_train)

y_pred_gini = self.clf_df_gini.predict(X_test)
y_pred_score_gini = self.clf_df_gini.predict_proba(X_test)
```

ENTROPY model:

```
self.clf_df_entropy = DecisionTreeClassifier(criterion="entropy", random_state=100,
max_depth=vmax_depth)
perform training
self.clf_df_entropy.fit(X_train, y_train)

y_pred_entropy = self.clf_df_entropy.predict(X_test)
y_pred_score_entropy = self.clf_df_entropy.predict_proba(X_test)
```

Once the model is initially trained and tested the dashboard of the decision tree classifier gets updated with the performance measures like confusion matrix, classification report, Accuracy score and ROC value.The below code snippet shows the above defined performance measures for entropy model and the same can be implemented for gini model as well.

```
# confusion matrix for Decision Tree
    conf_matrix_entropy = confusion_matrix(y_test, y_pred_entropy)

# clasification report

    self.class_rep_entropy = classification_report(y_test, y_pred_entropy)
    self.txtResults2.appendPlainText(self.class_rep_entropy)

# accuracy score

    self.accuracy_score_entropy = accuracy_score(y_test, y_pred_entropy) * 100
    self.txtAccuracy2.setText(str(self.accuracy_score_entropy))

# ROC-AUC
    self.rocauc_score_entropy = roc_auc_score(y_test, y_pred_score_entropy[:, 1]) * 100
    self.txtRoc_auc2.setText(str(self.rocauc_score_entropy))

    self.fpr_entropy, self.tpr_entropy, _ = roc_curve(y_test, y_pred_score_entropy[:, 1])
    self.auc_entropy = roc_auc_score(y_test, y_pred_score_entropy[:, 1])
```

The important features of the model can be viewed using Imp_Features option on the dashboard:

```
#important features

importances_entropy = self.clf_df_entropy.feature_importances_

# convert the importances into one-dimensional 1darray with corresponding df column names as axis
labels
f_importances_entropy = pd.Series(importances_entropy, self.current_features.columns)

# sort the array in descending order of the importances
 f_importances_entropy.sort_values(ascending=True, inplace=True)

 self.X_Features_entropy = f_importances_entropy.index
 self.y_Importance_entropy = list(f_importances_entropy)
```

Random Forest Classifier:

The packages that have been used for the modelling is Scikit Learn and we have imported random forest classifier library to perform the functions of the algorithm

```
from sklearn.ensemble import RandomForestClassifier
```

The data is encoded by using Label encoder function provided above to prepare the data for model training. The processed data is then split into train and test split at the ratio of 70:30 by default which is subject to change as per the user's requirements. The model gets trained and tested on the basis of criterions gini and entropy along with the number of estimators provided by the user.

GINI Model:

```
self.clf_rf_gini = RandomForestClassifier(n_estimators=n_esti, criterion='gini', random_state=100)

# perform training
self.clf_rf_gini.fit(X_train, y_train)

# predicton on test using all features
y_pred_gini = self.clf_rf_gini.predict(X_test)
s  y_pred_score_gini = self.clf_rf_gini.predict_proba(X_test)
```

ENTROPY Model:

```
# specify random forest classifier
self.clf_rf_entropy = RandomForestClassifier(n_estimators=n_esti, criterion='entropy',
random_state=100)

# perform training
self.clf_rf_entropy.fit(X_train, y_train)

# predicton on test using all features
y_pred_entropy = self.clf_rf_entropy.predict(X_test)
y_pred_score_entropy = self.clf_rf_entropy.predict_proba(X_test)
```

Once the model is initially trained and tested the dashboard of the random forest classifier gets updated with the performance measures like confusion matrix, classification report, Accuracy score and ROC value. The below code snippet shows the above defined performance measures for entropy model and the same can be implemented for gini model as well.

```
# confusion matrix for RandomForest
conf_matrix_entropy = confusion_matrix(y_test, y_pred_entropy)
```

```
# clasification report

        self.class_rep_entropy = classification_report(y_test, y_pred_entropy)
        self.txtResults2.appendPlainText(self.class_rep_entropy)

# accuracy score

        self.accuracy_score_entropy = accuracy_score(y_test, y_pred_entropy) * 100
        self.txtAccuracy2.setText(str(self.accuracy_score_entropy))

# ROC-AUC
        self.rocauc_score_entropy = roc_auc_score(y_test, y_pred_score_entropy[:, 1]) * 100
        self.txtRoc_auc2.setText(str(self.rocauc_score_entropy))

        self.fpr_entropy, self.tpr_entropy, _ = roc_curve(y_test, y_pred_score_entropy[:, 1])
        self.auc_entropy = roc_auc_score(y_test, y_pred_score_entropy[:, 1])
```

The important features of the model can be viewed using Imp_Features option on the dashboard:

```
#important features

 importances_entropy = self.clf_rf_entropy.feature_importances_

# convert the importances into one-dimensional 1darray with corresponding df column names as axis
labels
f_importances_entropy = pd.Series(importances_entropy, self.current_features.columns)

# sort the array in descending order of the importances
 f_importances_entropy.sort_values(ascending=True, inplace=True)
 self.X_Features_entropy = f_importances_entropy.index
 self.y_Importance_entropy = list(f_importances_entropy)
```

Support Vector Machine

The packages that have been used for the modelling is Scikit Learn and we have imported Support Vector Classifier library to perform the functions of the algorithm

```
from sklearn.svm import SVC
```

The data is encoded by using Label encoder function provided above to prepare the data for model training. The processed data is then split into train and test split at the ratio of 70:30 by default which is subject to change as per the user's requirements. The model gets trained and tested on the basis of kernel" rbf "along with the feature's selection provided by the user

```
self.clf_svc = SVC(kernel=kernel1)

# perform training
self.clf_svc.fit(X_train, y_train)

# predicton on test using all features
y_pred = self.clf_svc.predict(X_test)
y_pred_score = self.clf_svc.decision_function(X_test)
```

Once the model is initially trained and tested the dashboard of the support vector classifier gets updated with the performance measures like confusion matrix, classification report, Accuracy score and ROC value. The below code snippet shows the above defined performance measures value for rbf kernel

```
# confusion matrix for SVC
  conf_matrix = confusion_matrix(y_test, y_pred)

# clasification report

self.class_rep = classification_report(y_test, y_pred)
self. txtResults1.appendPlainText(self.class_rep)

# accuracy score

self.accuracy_score = accuracy_score(y_test, y_pred) * 100
self.txtAccuracy1.setText(str(self.accuracy_score))

# ROC-AUC
self.rocauc_score = roc_auc_score(y_test, y_pred_score) * 100
self.txtRoc_auc1.setText(str(self.rocauc_score))

self.fpr, self.tpr, _ = roc_curve(y_test, y_pred_score)
self.auc = roc_auc_score(y_test, y_pred_score)
```

Structure of the application:

1. **File**
   - **Exit:** It quits the entire application
2. **Load Dataset**
   - **Upload Data:** It takes up data from the user and displays the features. The application will throw an error when user proceed to next steps without uploading the dataset
3. **EDA analysis**
   - **Histogram:** This option presents distribution of each features in the processed dataset. The user can choose features of their own choice to view the histogram of respective features. A warning window will pop up when user selects more than one feature.
   - **Scatter Plot:** This option displays a box plot displaying the relation amongst two features in the dataset. The user can select two features of their own choice for the plot and application will throw an error when more than two variables are selected
4. **ML Models**
   - **Decision Tree Classifier:**
     - This option populates a dashboard with all the results generated from the decision tree algorithm
     - The features to be included in the algorithm are already chosen. However, the user can alter the number of features need to be included in the model. The parameters like test size and maximum depth of the tree can also be changed.
     - Once the features and parameters are selected, user can click Build Model button to populate the dashboard with data
     - The algorithm can be executed any number of times. The dashboard gets cleared every time and produce new output for each selection
     - The Plot ROC options builds two roc_auc curves for gini and entropy model. The Imp_features option displays the important features of the model. View Tree option download and open the pdfs of decision trees of gini and entropy models
   - **Random Forest Classifier:**
     - This option populates a dashboard with all the results generated from the random forest algorithm
     - The features to be included in the algorithm are chosen. However, the user can manipulate the number of features and parameters (test size and number of estimators) they want by altering the checkboxes in the screen
     - Once the features and parameters are selected, user can click Execute RF button to populate the dashboard with data
     - The Plot ROC options builds two roc_auc curves for gini and entropy model. The Imp_features option displays the important features of the model
     - The algorithm can be executed any number of times. The dashboard gets cleared every time and produce new output for each selection
   - **Support Vector Classifier:**
     - This option populates a dashboard with all the results generated from the support vector classifier algorithm

- The features to be included in the algorithm are already chosen. However, the user can alter the number of features need to be included in the model. The parameters like test size and kernel can also be changed.
- Once the features and parameters are selected, user can click Execute SVC button to populate the dashboard with data
- The Plot ROC options builds a roc_auc curve for the model.
- The algorithm can be executed any number of times. The dashboard gets cleared every time and produce new output for each selection

RESULTS

Initial UI Window: This is how the HR Analytics Application looks, it has three buttons one for the Upload data (loading the dataset), EDA Analysis & ML Models.

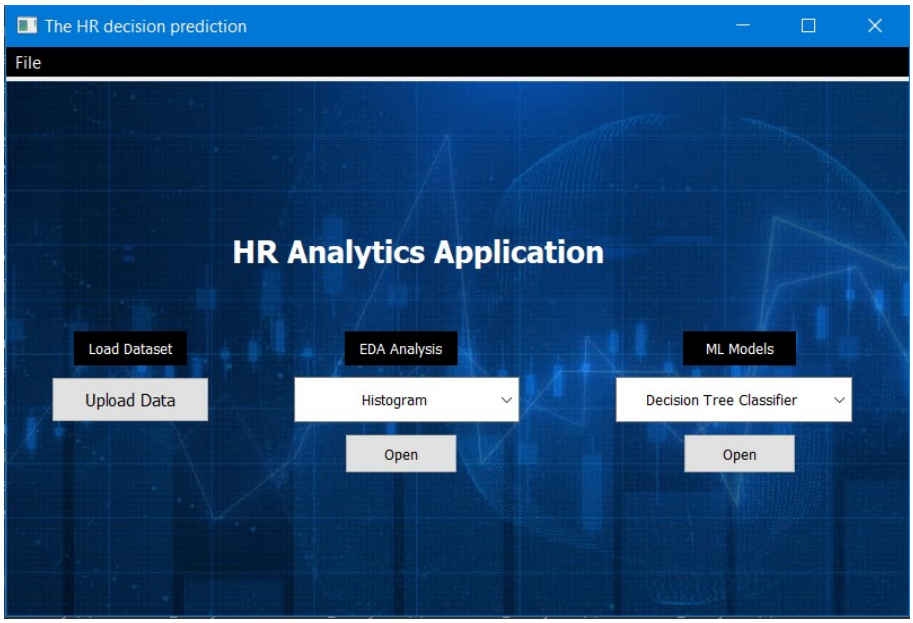

Fig.1.Initial UI Window

Upload Dataset: This feature is being used to upload the dataset from the source directory and after uploading it displays the dataset attributes.

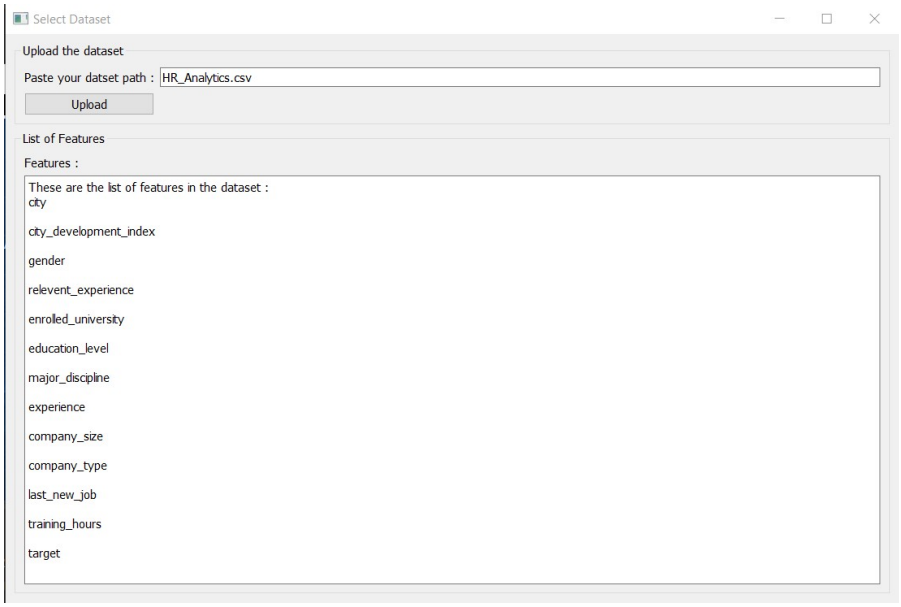

Fig.2.Upload Dataset

Histogram: The image describes the pictorial representation of histogram and the various features provided on the left grid and after pushing the plot button the histogram gets displayed on the canvas.

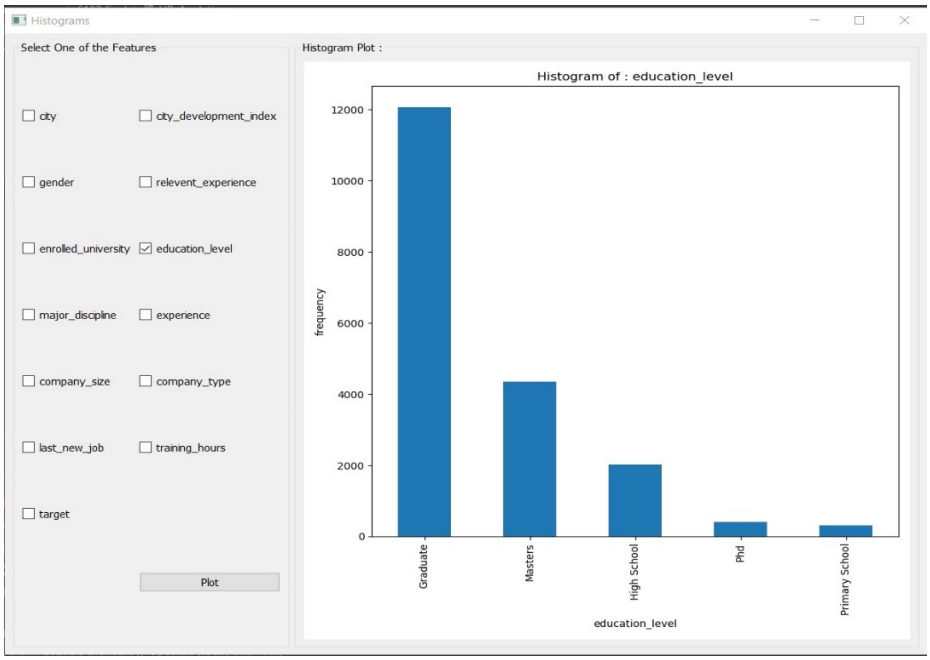

Fig.3.Histogram

Histogram for Target: There are observations with more no of 0's than 1's in the dataset and less precision on predicting the 1's.

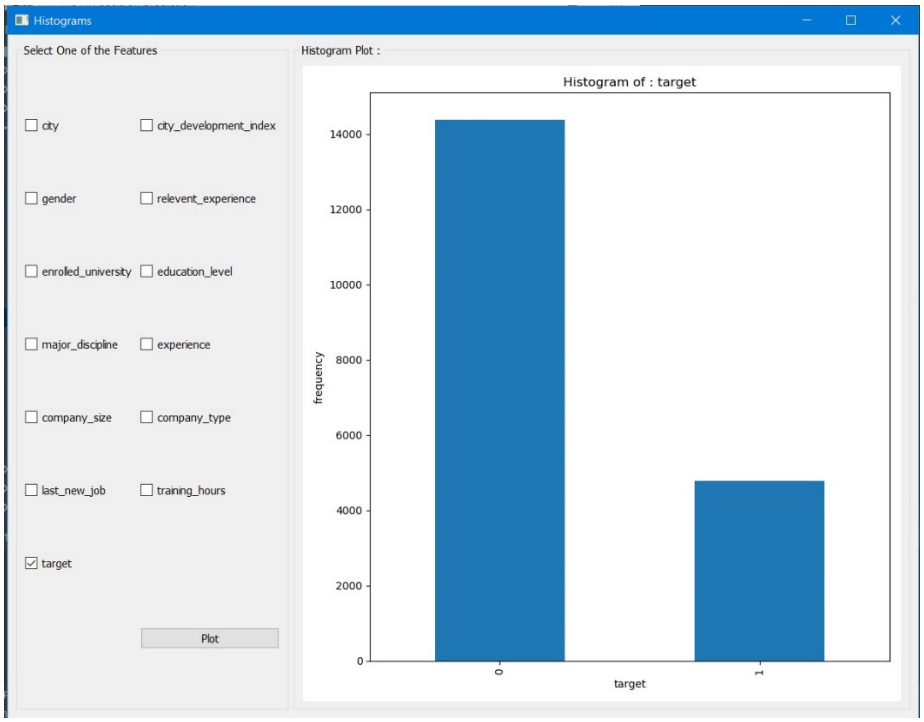

Fig.4.Histogram for Target

Scatterplot: This image describes the scatter plot graphical representation of the various features selected; left grid represents the variables to be selected for y axis and upper grid represents the variable to be selected for the x-axis.

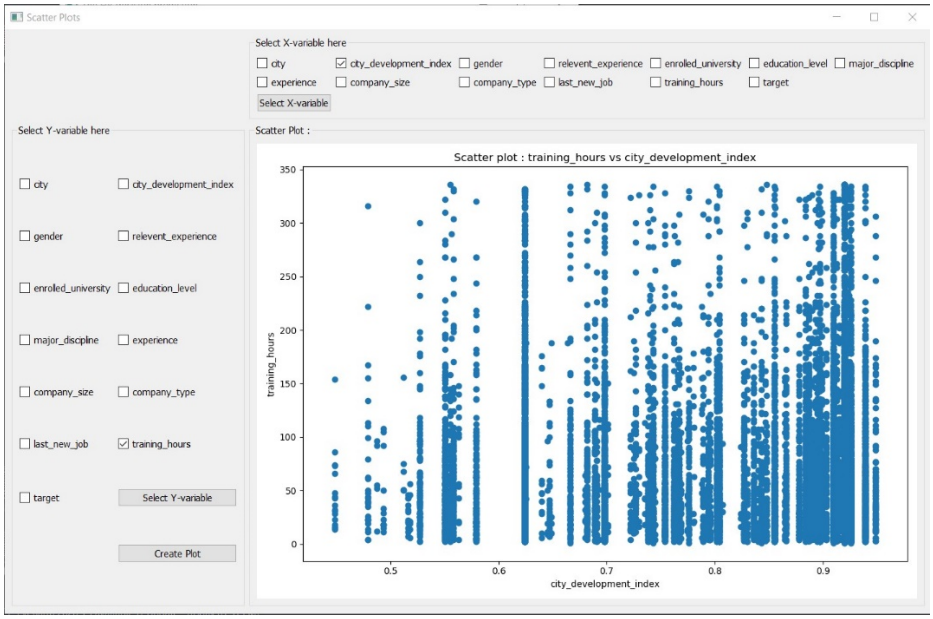

Fig.5.ScatterPlot

Decision Tree Classifier: The image displays the decision tree dashboard and the user can manually change the percentage for test and also the maximum depth, The features can be selected as per the user's choice and execute the decision tree.The plot roc button displays the ROC graph and view tree button provides two graphs one for entropy and gini in pdf format.

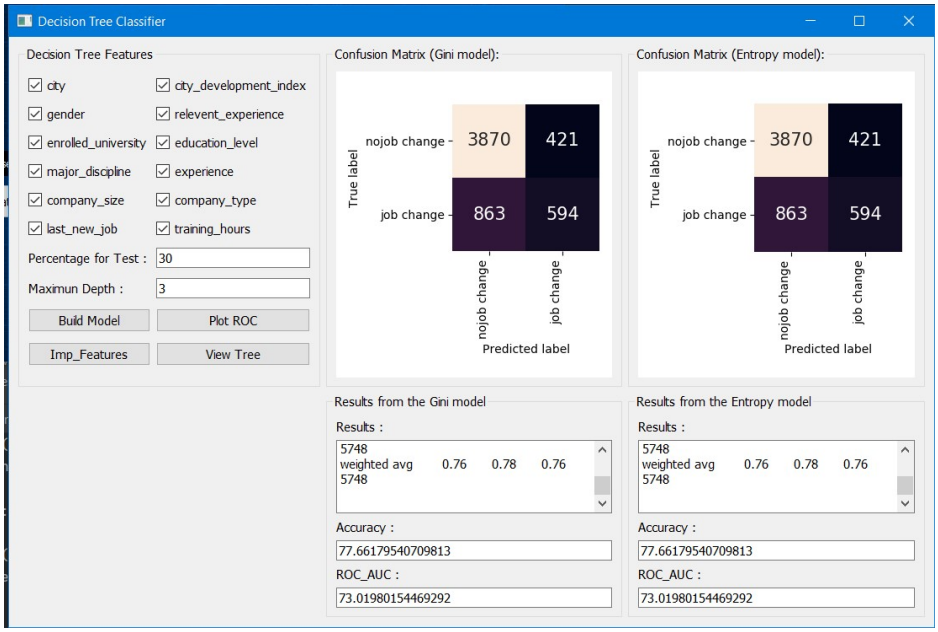

Fig.6.Decision Tree Classifier

Feature Importance Graph (Decision Tree): The feature importance graph in decision tree displays the most important features which needs to be considered while building a model. The below figure shows that only 5 features have significant importance to the model.

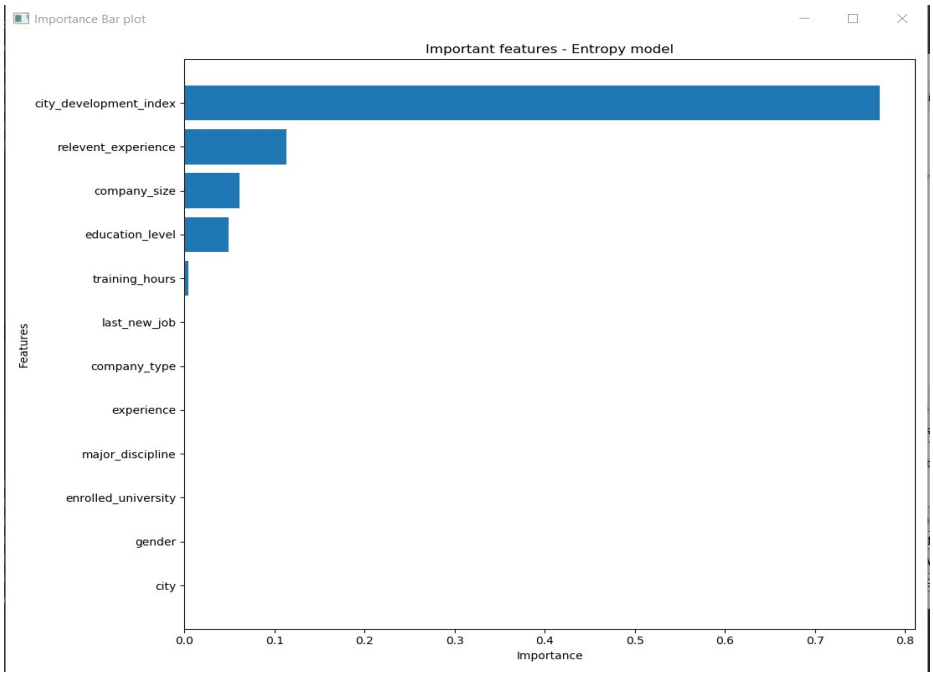

Fig.7.Feature Importance Graph (Decision Tree)

ROC_AUC Curve of gini and entropy model (Decision Tree): ROC is a probability curve and AUC represent the degree or measure of separability. The green colored represents the roc curve and blue represents the auc. The roc_auc value of decision tree models is 0.73.

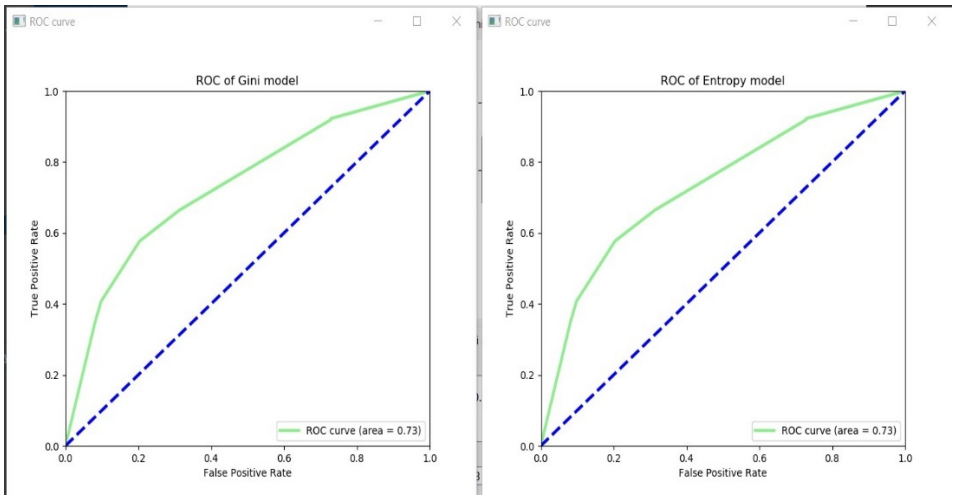

Fig.8. ROC_AUC Curve of gini and entropy model (Decision Tree)

Random Forest Classifier: The image displays the random forest dashboard and the user can change the no of estimators, select and unselect the features, set the percentage for the test size and execute button will provide the accuracy and confusion matrix on the canvas as displayed and plot roc button displays the ROC graph.

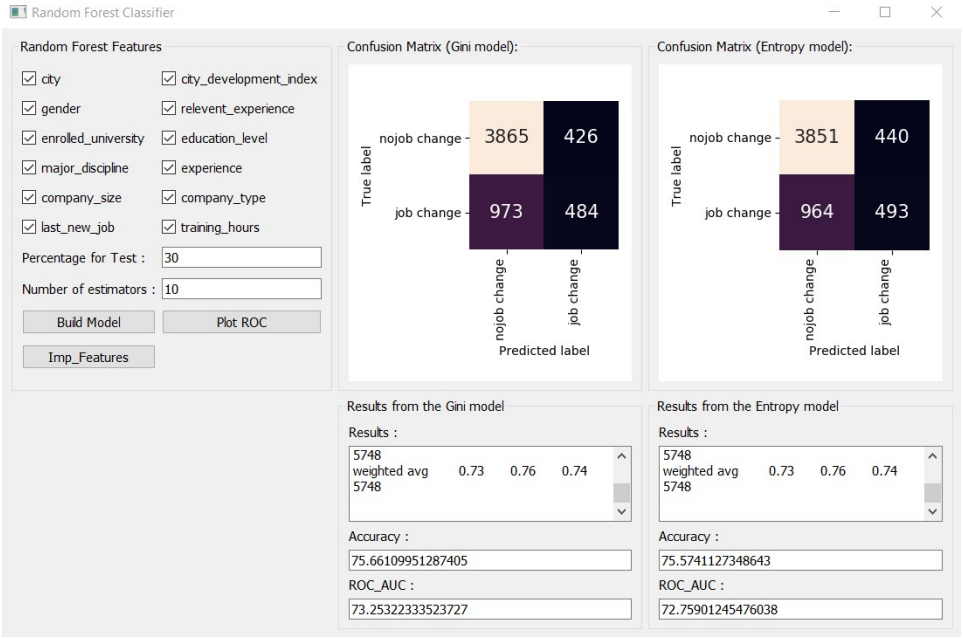

Fig.9.Random Forest Classifier

Feature Importance Graph (Random Forest): The plot of important features for random forest shows the most significant features which needs to be considered while developing a model.

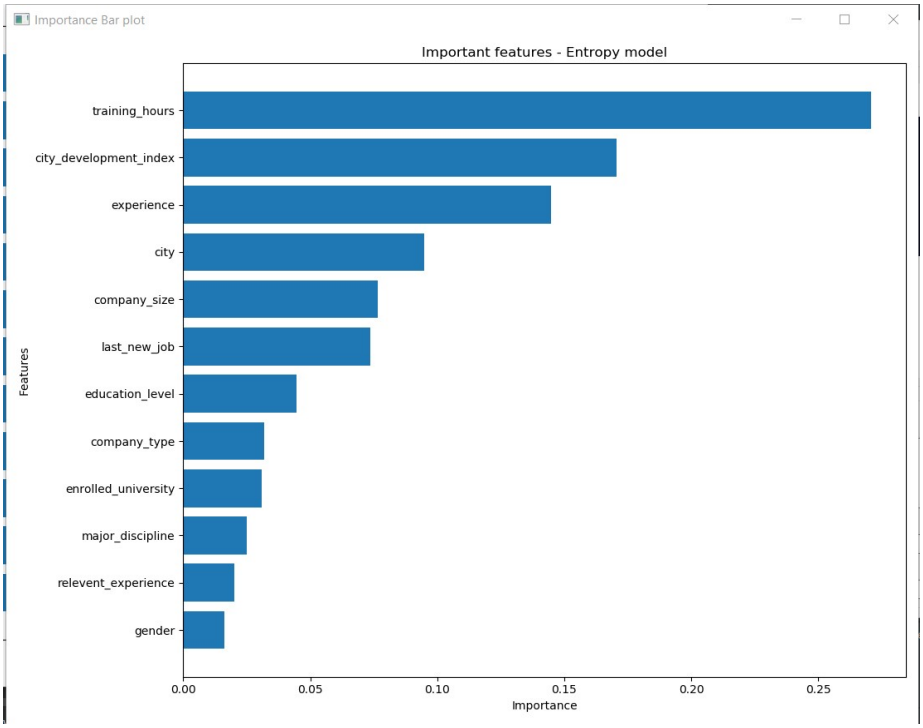

Fig.10.Feature Importance Graph (Random Forest)

ROC_AUC Curve of gini and entropy model (Random Forest): This curve defines how much the model is capable to distinguish different classes in the model. The green colored curve represents the roc and blue represents the auc. The roc_auc value of random forest models is 0.73.

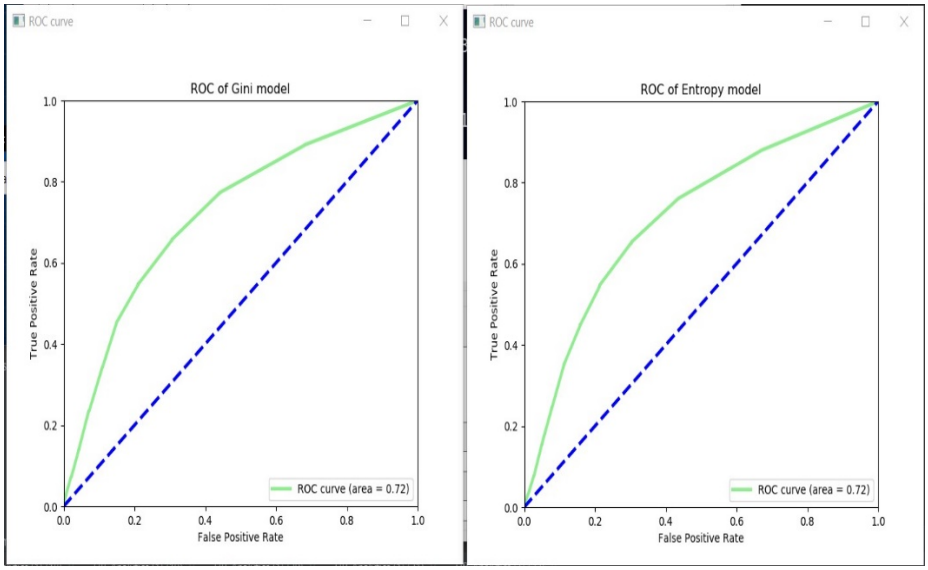

Fig.11.ROC_AUC Curve of gini and entropy model (Random Forest)

Support Vector Classifier : This image describes the SVM dashboard, the kernel is being auto set to rbf to provide better results for the modelling and execute svc button will display the accuracy, roc_auc rate and confusion matrix on the canvas, whereas the plot roc push button displays the roc curve graph.

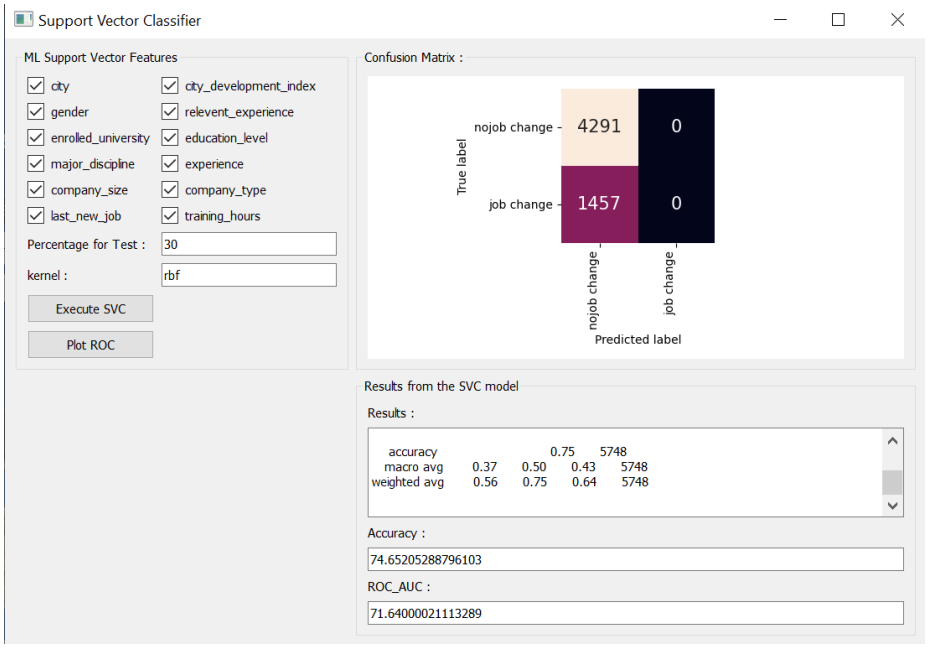

Fig.12.Support Vector Classifier

ROC_AUC Curve of Support vector classifier: The roc_auc value of support vector classifier 0.63. It shows that the model does not work comparatively with respect to other models

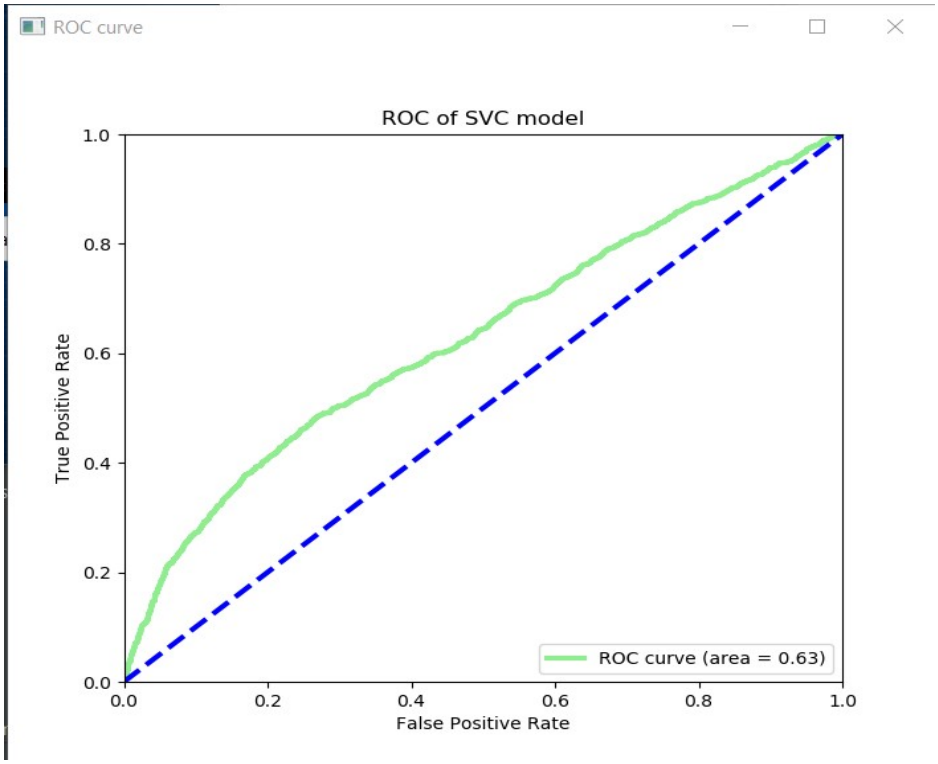

Fig.13.ROC_AUC Curve of Support vector classifier

Error Window: The validation has been given on graphs to make sure that the feature selection should not exceed more than the required variables respectively.

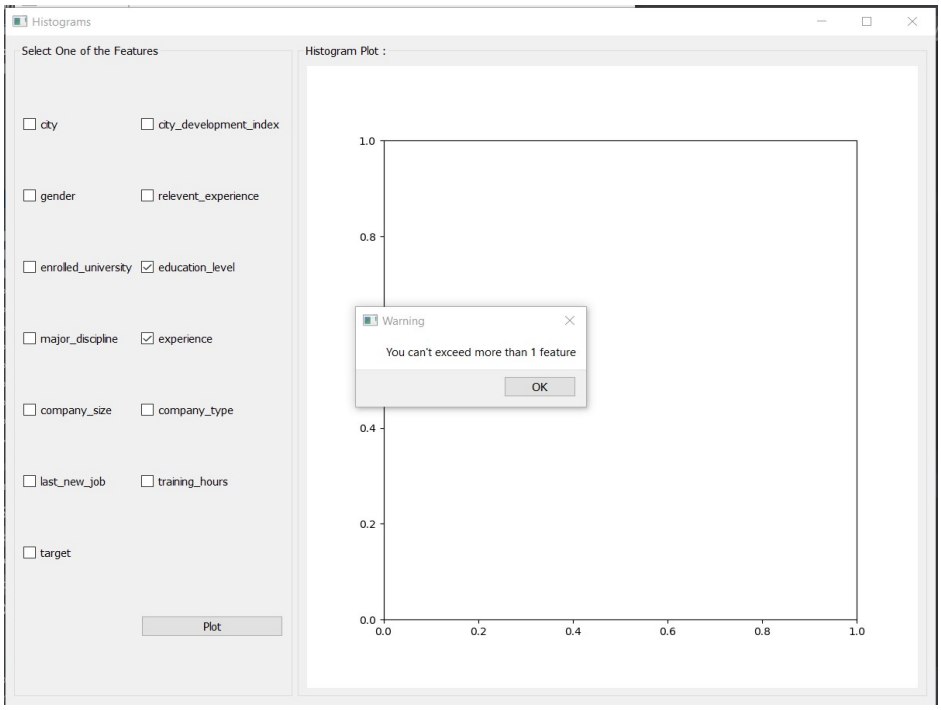

Fig.14.Error Window

Closing Window: A menu bar with a closing file option has being given on the dashboard of the application.

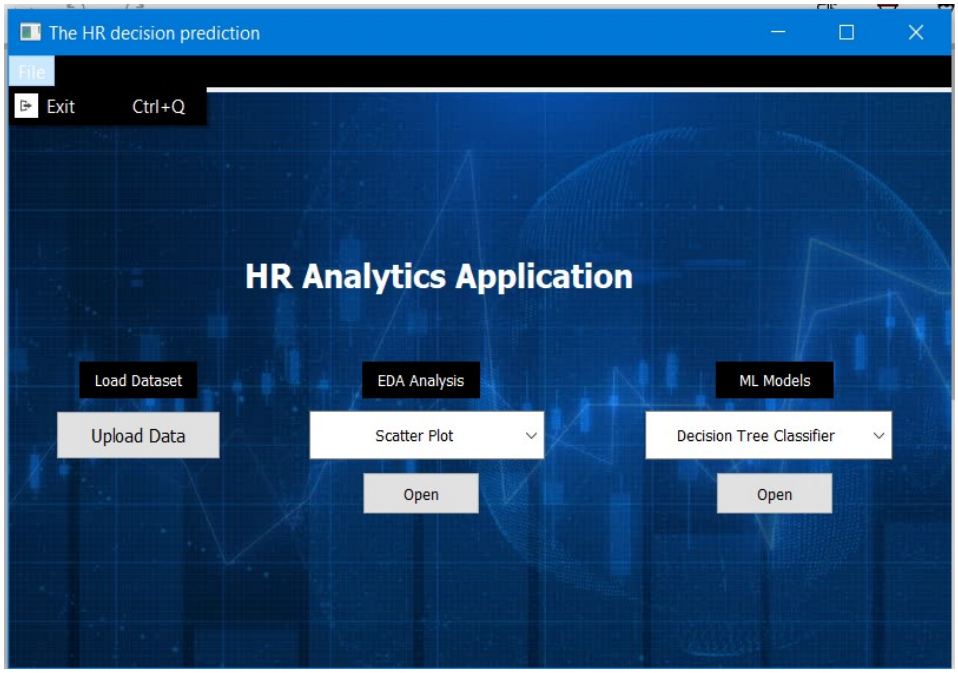

Fig.15.Closing Window

SUMMARY

Decision Tree Classifier:

For gini model:

1. Accuracy of the gini model, Accuracy = 77.66%.
2. From the classification report of gini model:
    1. F1 score for 0's = 0.86 and f1 score for 1's = 0.48
    2. Precision for 0's = 0.82 and precision for 1's = 0.59
    3. Recall for 0's = 0.90 and recall for 1's = 0.41
3. Area under the curve for the gini model, AUC = 0.73, A descent model should have AUC value above 0.8. So, we can say that this model is not the best model.
4. The features with highest importance are – city_development_index, relevant_expericence, company_size, education_level and training_hours. So, these features are most correlated to the target variable in this model, other features have no importance at all in this model.

Accuracy is 77.66% which is acceptable but we cannot say it is a great model.

We can observe that f1-scores, precision and recall rate is high for 0's in this model. The reason is because there are a greater number of observations which have 0's in target variable than those which have 1's as we have observed the histogram of target variable in EDA analysis.

For entropy model:

1. Accuracy of the gini model, Accuracy = 77.66%.
2. From the classification report of gini model:
    1. F1 score for 0's = 0.86 and f1 score for 1's = 0.48
    2. Precision for 0's = 0.82 and precision for 1's = 0.59
    3. Recall for 0's = 0.90 and recall for 1's = 0.41
3. Area under the curve for the gini model, AUC = 0.73, A descent model should have AUC value above 0.8. So, we can say that this model is not the best model.
4. The features with highest importance are – city_development_index, relevant_expericence, company_size, education_level and training_hours. So, these features are most correlated to the target variable in this model, other features have no importance at all in this model.

Accuracy is 77.66% which is acceptable but we cannot say it is a great model.

We can observe that f1-scores, precision and recall rate is high for 0's in this model. The reason is because there are a greater number of observations which have 0's in target variable than those which have 1's as we have observed the histogram of target variable in EDA analysis.

We can also observe that the results of both the gini and entropy DT models are same.

Since there are only 5 features which have high importance in this model and the rest have none, it is better to drop all the remaining variables and build new decision tree models

So, we build two new decision tree models with X-variables as city_development_index, relevant_expericence, company_size, education_level and training_hours and drop the remaining variables.

For both gini and entropy models:

1. Accuracy of the gini model, Accuracy = 77.66%.
2. From the classification report of gini model:
    1. F1 score for 0's = 0.86 and f1 score for 1's = 0.48
    2. Precision for 0's = 0.82 and precision for 1's = 0.59
    3. Recall for 0's = 0.90 and recall for 1's = 0.41
3. Area under the curve for the gini model, AUC = 0.73, A descent model should have AUC value above 0.8. So, we can say that this model is not the best model.

We can observe that the results have not changes even after we dropped the features of less importance.

We conclude that we can use the second model which is more efficient since it has same accuracy, f1-score, precision, recall rate and AUC as the first model and less dimensions than the first model.

Random Forest Classifier

For gini model:

1. Accuracy of the gini model, Accuracy = 75.66%.
2. From the classification report of gini model:
    1. F1 score for 0's = 0.85 and f1 score for 1's = 0.41
    2. Precision for 0's = 0.80 and precision for 1's = 0.53
    3. Recall for 0's = 0.90 and recall for 1's = 0.33
3. Area under the curve for the gini model, AUC = 0.73, A descent model should have AUC value above 0.8. So, we can say that this model is not the best model.
4. From the importance of features plot we can observe that all the variables have some importance in the models used. The top 6 variables with highest importance are – training_hours, city_development_index, experience, city, company_size and last_new_job. So, these features are most correlated to the target variable in this model.

Accuracy is 75.66% which is acceptable but we cannot say it is a great model.

We can observe that f1-scores, precision and recall rate is high for 0's in this model. The reason is because there are a greater number of observations which have 0's in target variable than those which have 1's as we have observed the histogram of target variable in EDA analysis.

For entropy model:

1. Accuracy of the entropy model, Accuracy = 75.57%.
2. From the classification report of entropy model:
    1. F1 score for 0's = 0.85 and f1 score for 1's = 0.41
    2. Precision for 0's = 0.80 and precision for 1's = 0.53
    3. Recall for 0's = 0.90 and recall for 1's = 0.34
3. Area under the curve for the entropy model, AUC = 0.72, A descent model should have AUC value above 0.8. So, we can say that this model is not the best model.
4. From the importance of features plot we can observe that all the variables have some importance in the models used. The top 6 variables with highest importance are – training_hours, city_development_index, experience, city, company_size and last_new_job. So, these features are most correlated to the target variable in this model.

Accuracy is 75.57% which is acceptable but we cannot say it is a great model.

We can observe that f1-scores, precision and recall rate is high for 0's in this model. The reason is because there are a greater number of observations which have 0's in target variable than those which have 1's as we have observed the histogram of target variable in EDA analysis.

We can also observe that the results of both the gini and entropy DT models are same.

We can also observe that results of both the models of Random Forest classifier is very close to results of both the first models of Decision Tree classifier. But still the decision tree model has better accuracy.

From the importance of features plot we can observe that all the variables have some importance in the models used. So, we can start dropping least important dimensions one by one and build new Random Forest models until there is major change in the accuracy.

After dropping major_discipline, relevant_experience, gender variables we stop from dropping more variables as the accuracy has changed drastically.

So, we build two new Random Forest models with all X-variables except major_discipline, relevant_experience, gender.

For gini model:

1. Accuracy of the gini model, Accuracy = 75.13%.
2. From the classification report of gini model:

    1. F1 score for 0's = 0.84 and f1 score for 1's = 0.41
    2. Precision for 0's = 0.80 and precision for 1's = 0.51
    3. Recall for 0's = 0.89 and recall for 1's = 0.34
3. Area under the curve for the gini model, AUC = 0.72, A descent model should have AUC value above 0.8. So, we can say that this model is not the best model.

Accuracy is 75.13% which is acceptable but we cannot say it is a great model.

For entropy model:

1. Accuracy of the entropy model, Accuracy = 75.15%.
2. From the classification report of entropy model:
    1. F1 score for 0's = 0.84 and f1 score for 1's = 0.42
    2. Precision for 0's = 0.80 and precision for 1's = 0.51
    3. Recall for 0's = 0.89 and recall for 1's = 0.35
3. Area under the curve for the entropy model, AUC = 0.715, A descent model should have AUC value above 0.8. So, we can say that this model is not the best model.

Accuracy is 75.15% which is acceptable but we cannot say it is a great model.

We can observe that the results do not have drastic changes even after we dropped the three features of least importance.

We conclude that we can use the second model which is more efficient since it has nearly similar accuracy, f1-score, precision, recall rate and AUC as the first model and less dimensions than the first model.

Support Vector Classifier

For SVC model:

1. Accuracy of the model, Accuracy = 75.36%.
2. From the classification report of model:
    1. F1 score for 0's = 0.85 and f1 score for 1's = 0.25
    2. Precision for 0's = 0.77 and precision for 1's = 0.55
    3. Recall for 0's = 0.95 and recall for 1's = 0.16
3. Area under the curve for the model, AUC = 0.63, A descent model should have AUC value above 0.8. So, we can say that this model is not the best model.

Accuracy is 75.36% which is acceptable but we cannot say it is a great model.

From the confusion matrix we can observe that majority of the predictions are 0's, so also contains lot of False negatives.

CONCLUSION

Decision Tree model is the best model of the three different classifiers because it has the highest accuracy, f1-score and precision. It has the best AUC of the three classifiers as well. Also, we have built the model with only 5 features which have highest importance of features, so it is more efficient of the three classifiers in the sense of having more accuracy with only 5 dimensions used.

Random Forest model is the second-best model of the three classifiers as it has slightly less accuracy and other results than decision tree model. We can improve the accuracy and other results by increasing the number of estimators in the model, but that would also require higher computational power.

Support Vector model is the least favourite of the three classifiers though the accuracy, f1 score, precision is high the AUC value is very low, at 0.63. This means that the model is assigning the test results to 0's more leading to high false negative value.

The dataset has a greater number of 0's than 1's in the dataset, they are not equally proportional so there is always a chance of more false negative values which leads to recall rate and precision of 1's to very low. So, we cannot be sure of the result when we predict 1's but we will be confident of our prediction of 0's as it has high precision and recall rate.

In conclusion we can say that of the three different classifiers, we can use Decision Tree model to predict whether an employee is going to change to new job or not.

REFERENCES

- https://doc.bccnsoft.com/docs/PyQt5/
- https://pyqt5.files.wordpress.com/2017/06/pyqt5tutorial.pdf
- https://numpy.org/doc/
- https://matplotlib.org/stable/users/index.html
- https://matplotlibguide.readthedocs.io/en/latest/
- https://sklearn.org/user_guide.html
- https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/
- https://towardsdatascience.com/explain-your-machine-learning-with-feature-importance-774cd72abe
- https://www.mygreatlearning.com/blog/introduction-to-support-vector-machine/
- https://www.mygreatlearning.com/blog/random-forest-algorithm/
- https://www.mygreatlearning.com/blog/decision-tree-algorithm/
- https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

APPENDIX

Technical Considerations:

1. Installation of Python 3.5 & above, Pycharm and Anaconda are necessary
2. Packages like numpy, pandas, matplotlib, sklearn, PyQt5 needs to be installed in the computer in order to execute the application
3. The application also uses graphviz-2.38, the user needs to install this to download the decision tree pdf's. The directory path included in the application is:
   'C:\\Program Files (x86)\\graphviz-2.38\\release\\bin'
4. The path to the Graphviz directory needs to be changed based on the path in the user's computer before the execution of HR_Analytics.py

GitHub Repo Link:

All the documents related to this project are included in the following repo link:

https://github.com/adingankar/FINAL_PROJECT_GROUP7

The repo has following folders and file:

1. README.md – Defines the structure of the repo
2. Code – This folder contains the code, images used in the code, PDF of tree models, Dataset
3. Final _Group _Presentation – This folder has the PDF version of group project presentation
4. Final_Group_Project_Report – This folder includes the complete report of the project in PDF format.
5. Individual_Project – This folder contains the folders of report and codes of the group members.

Project Demo:

The demo of the project can be accessed using the following link:

https://drive.google.com/file/d/1Y_u4un0_inFmVXfVGb4GQ2kmbNNgqJ7d/view?usp=sharing