HR Analytics: Job Change

Report by

Rehapriadarsini Manikandasamy

The George Washington University

Author Note

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

TABLE OF CONTENTS

INTRODUCTION

The project aims to predict the probability of the candidate who will be willing to work after training or will be looking for new job. The dataset sourced was developed by a Big Data and Data Science Company which needs to do the analytics to understand the candidate's interest of taking up job in the company after training hours. This analysis helps them to reduce the cost and time involved in delivering the quality of training or planning the course material for the selection process. The dataset used has information related to demographics, education, experience is obtained from the candidate's signup form and enrolment. The dataset is primarily designed to understand the employee's moto of taking up a job for HR researches. 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 developed 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 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 WORK

My contributions to the project are mainly to develop the support vector classifier model and setting up the main UI of the project. The support vector classifier modelling involves in presenting an application window to the user to implement the model for the dataset by manipulating the parameter and test size values. The main application is having been implemented using GUI elements provided by PyQt5 package. The GUI main window has the option to upload dataset, where the users can pass the path of their dataset. The EDA analysis part has drop down menu to select the plots of user's choice. The ML models option helps the user by providing options to shift modelling windows. Considering the overall contribution to the project, I have played part of designing one out of three models and contributed to the complete UI design of the project.

Information on Algorithm development:

Support Vector Classifier model of the project was inspired based on the following background study.

Support Vector Machine:

Support vector machine is a machine learning algorithm which is primarily used for classification, regression and outlier detection. The primary advantages of SVM are:

- Effective in high dimensional spaces
- Effective in cases where number of dimensions is greater than the number of samples
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile kernel functions available. Common kernels are provided, but it is also possible to specify custom kernels.

The support vector machine depends on three machine learning principles:

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

Considering the dataset chosen, I have selected to use support vector classifier for the model.

Support Vector Classifier:

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. The kernel function used in default is radial basis kernel function.

Radial Basis Kernel Function:

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

Support Vector Classifier:

Before training the model with cleaned dataset, the dataset needs to be encoded using Label Encoder. Encoding was done to decide in a better way on how those labels must be operated and labels are converted into numeric form.

```
from sklearn.preprocessing import LabelEncoder
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])
     X = self.current_features.values
    y = data.iloc[:, -1]
    class_le = LabelEncoder()
    # fit and transform the class
    y = class_le.fit_transform(y)
```

The code snippet shows the label encoding done before splitting the dataset for training and testing. The Label Encoder if fetched from Preprocessing module of sklearn package. The label encoder automatically converts the features selected for the model development into numerical form to ensure the model works smoothly.

The model development is done using the SVC module available in sklearn package to incorporate the support vector machine functions.

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

# split the dataset into train and test
 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=vtest_per, random_state=100)

# specify support vector classifier
    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)
```

The above defined code block show cases how the data is split into train and test set. In default the train and test split ratio is 70:30, because of the flexibility in the UI to take inputs from the user for the test size. The kernel in default used is 'rbf'. The model gets initially trained with train data and the trained model is then exposed to the test data to make the predictions.

The predictions accuracy and overall efficacy of the model is measured using the following performance measures: Confusion Matrix, Classification Report, Accuracy, ROC value.

```
# confusion matrix for RandomForest
        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)
```

The UI implementation of this algorithm involves in using the UI elements available in the PyQt5 package. The structure of support vector classifier's application window is explained below.

- The option in the main window to open the SVC 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

Main Application Window:

The GUI application window has four main selections for user. They are:

1. File – The File option in the menu bar has an exit drop down menu. It helps the user to quit the entire application.
2. Load dataset – The button "Upload data" pops out a window to user to set the path for the dataset.
3. EDA Analysis – This option has a drop-down menu to select the plots of user's choice. It has two menu items: "Scatter plot" and "Histogram". The user can't move to this part without loading the dataset. An error window will pop up to inform user to load data.
4. ML models – This option allows user to select a machine learning among the three models available in the drop-down menu. Like EDA option, this option too will not allow user to proceed further without specifying the path of the dataset.

The main application window is developed using the GUI elements available in the PyQt5 package. Below is the set of libraries used from pyqt5:

- PyQt5 – Package installed to develop the GUI components of the project
- QMainWindow – To create the main window
- QApplication – To load the application
- QWidget – Controls overall widgets
- QPushButton - Provides a button icon
- QAction – It triggers the action needs to be performed
- QComboBox – It has been used to create the drop-down menu
- QLabel – To encode the labels at necessary places
- QGridLayout – Sets the overall layout for the window
- QCheckBox – Sets up checkbox options for users to select features

RESULTS

Initial GUI Window: GUI window has four options to the user to load data, analyse data, build models and exit the application. An image file has been used to set the background of the window. The buttons available on main window helps users to navigate to respective functions.

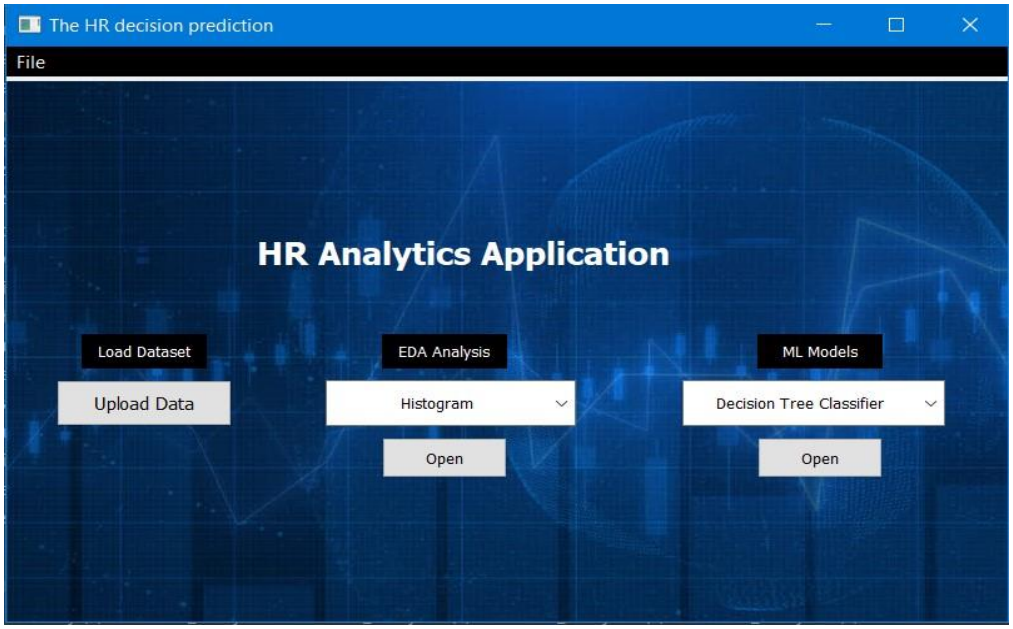

Fig.1: Main window

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.

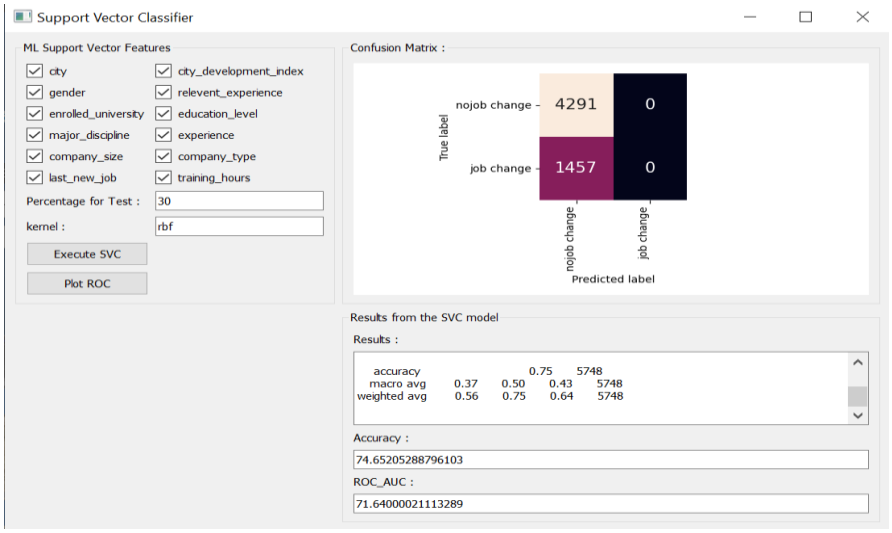

Fig.2: SVC

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

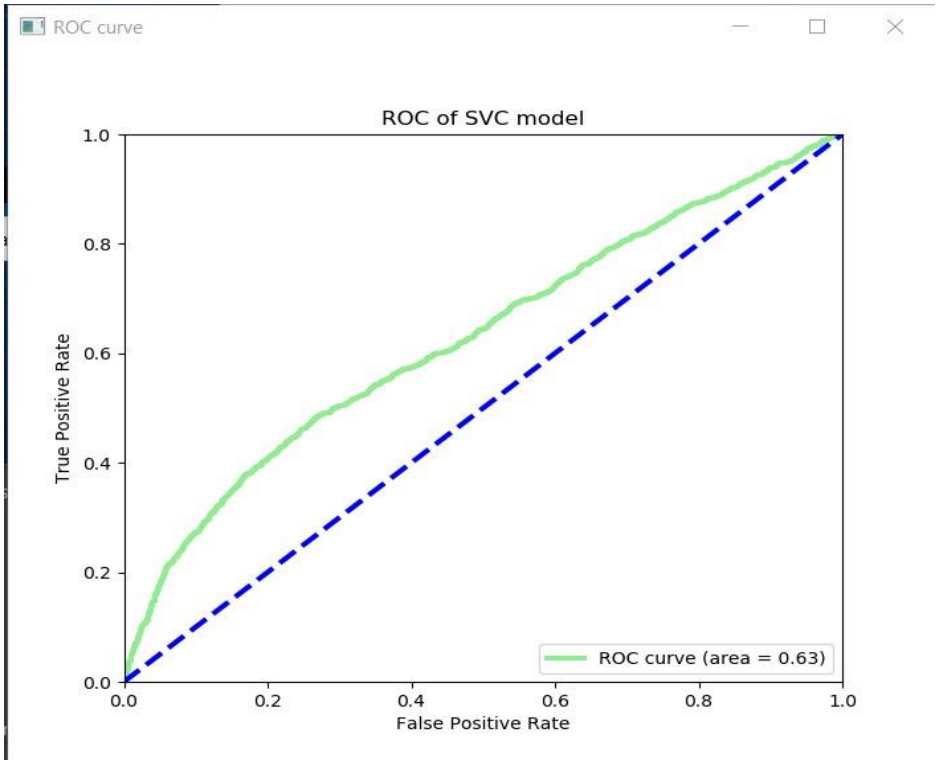

Fig.3: ROC_AUC Curve of Support vector classifier

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.

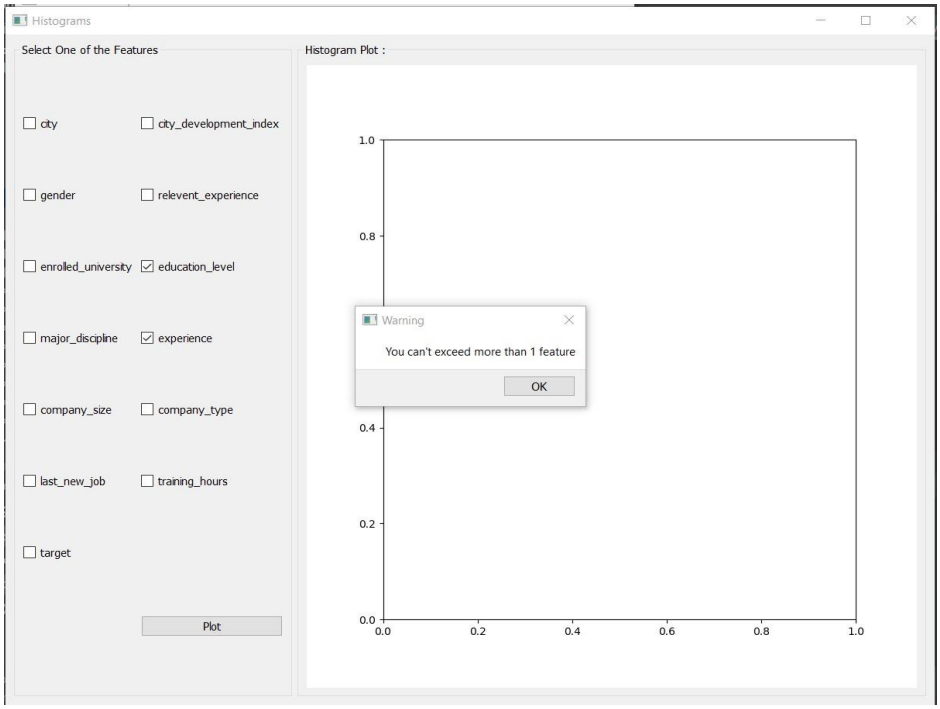

Fig.4.Error Window

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

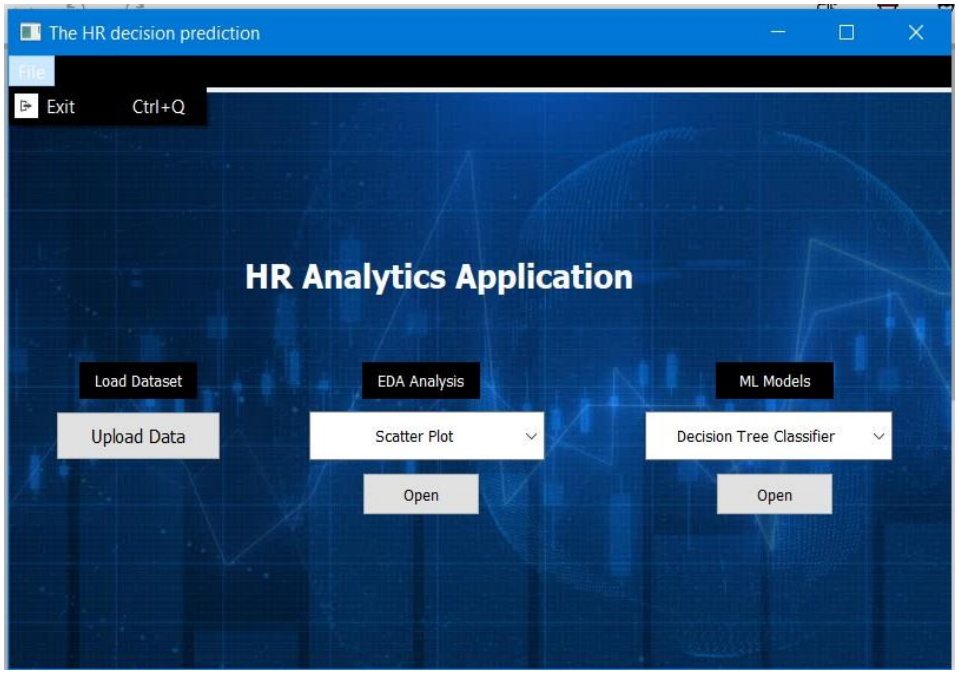

Fig.14.Closing Window

MODEL SUMMARY

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

Comparing the results of other models, the 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 model in future can be enhanced by tuning the parameters in a better way to make the accurate predictions. The model's GUI can be improved by adding some EDA analysis options for the user to analyse the data in higher end.

CODE PERCENTAGE

My code part contains total 608 lines of code. Out of which, I have referred to official documentations of SVC and PyQt5 to demonstrate the basic syntaxes of the code. Considering that as copied content my code will be having 300 lines of code and out of which 260 lines would have been modified according to the project needs. And I have added 308 lines to setup the UI as needed for the project along with implementing the SVC model part.

Code percentage = 300-260 / (300+308) *100 = 6.57%

REFERENCES

- https://scikit-learn.org/stable/modules/svm.html
- https://doc.bccnsoft.com/docs/PyQt5/
- https://pyqt5.files.wordpress.com/2017/06/pyqt5tutorial.pdf