Human Resources Analytics:
Job Change of Data Scientists

Supervised Machine Learning: Classification
Week 6

Course Final Project

Name : Lau Eng Hui

Role : Data Scientists

Table of Contents

List of Figures

Chapter 1 : Introduction

1.1 Objective

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 are in hands from candidates 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, 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 is predicted.

1.2 Benefit

The benefits of this analysis to the business or stakeholders of the data will be the ability to identify any factors that may be contributing to high rates of job change among data scientists. This information can be used to make informed decisions about how to retain valuable data scientists and improve retention rates. Additionally, understanding the factors that influence job change among data scientists can help to inform HR policies and practices that may be more effective in attracting and retaining top talent in this field.

Chapter 2 : Data Description

2.1 Features

a. enrollee_id : Unique ID for candidate
b. city: City code
c. city_ development _index : Development index of the city (scaled)
d. gender: Gender of candidate
e. relevent_experience: Relevant experience of candidate
f. enrolled_university: Type of University course enrolled if any
g. education_level: Education level of candidate
h. major_discipline: Education major discipline of candidate
i. experience: Candidate total experience in years
j. company_size: Number of employees in current employer's company
k. company_type : Type of current employer
l. lastnewjob: Difference in years between previous job and current job
m. training_hours: training hours completed
n. target: 0 – Not looking for job change, 1 – Looking for a job change

2.2 Size

The dataset contains 19158 rows and 14 columns.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   enrollee_id             19158 non-null  int64
 1   city                    19158 non-null  object
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object
 4   relevent_experience     19158 non-null  object
 5   enrolled_university     18772 non-null  object
 6   education_level         18698 non-null  object
 7   major_discipline        16345 non-null  object
 8   experience              19093 non-null  object
 9   company_size            13220 non-null  object
 10  company_type            13018 non-null  object
 11  last_new_job            18735 non-null  object
 12  training_hours          19158 non-null  int64
 13  target                  19158 non-null  float64
dtypes: float64(2), int64(2), object(10)
memory usage: 2.0+ MB
```

Figure 2.1: Dataset Infomation

2.3 Inspiration

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

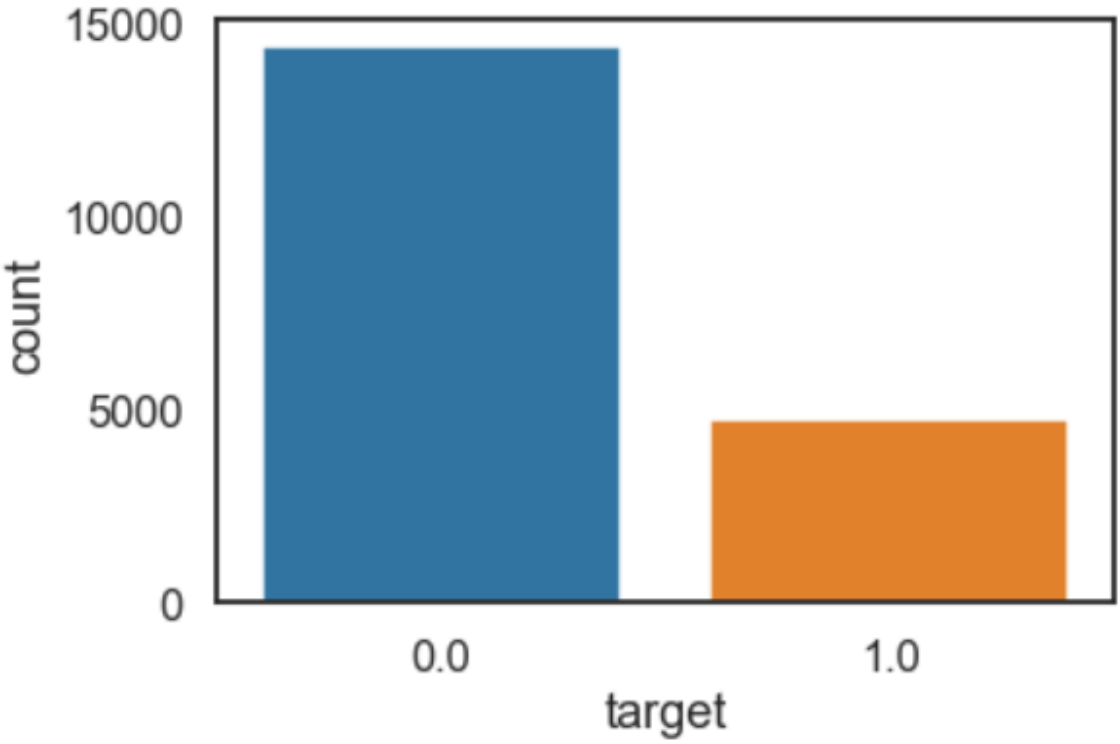

Figure 2.2: Imbalanced Dataset

Chapter 3 : Exploratory Data Analysis and Data Cleaning

3.1 Exploratory Data Analysis

Before modelling a machine learning, EDA should be done to understand the data and summarize their main characteristics. The relationship between each dependant variables and independent variable (target) is determined by data visualization using bar chart and histogram from seaborn library.

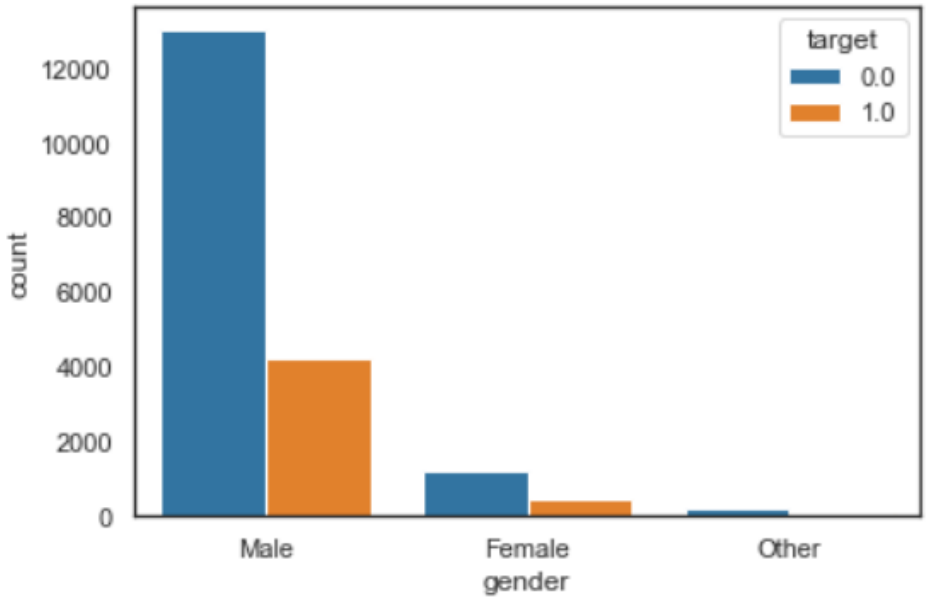

Figure 3.1: Bar chart to show the relationship between gender of candidates and target

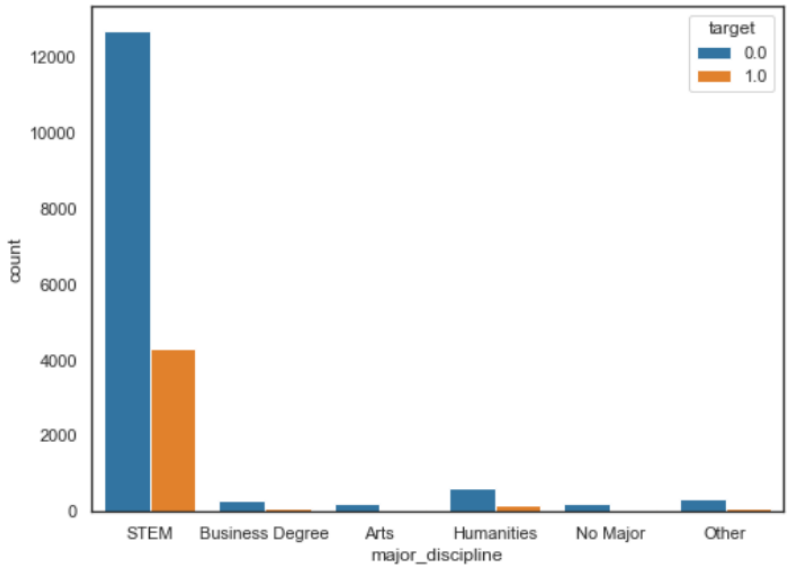

Figure 3.2: Distribution of Major Discipline

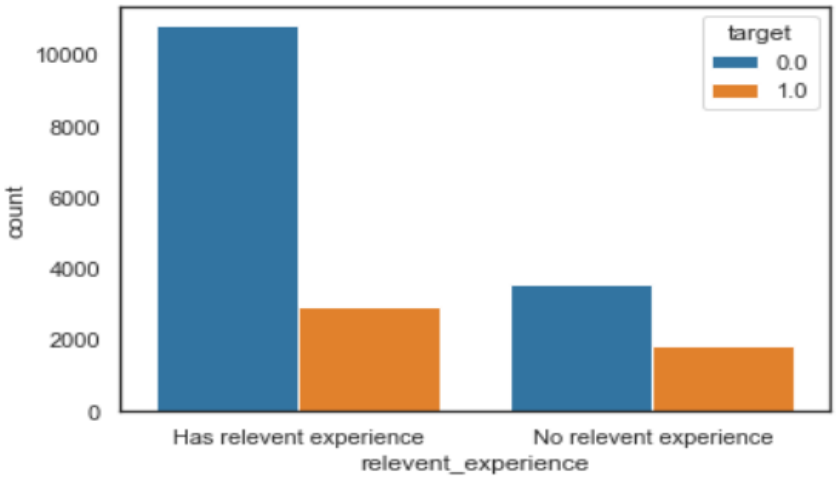

Figure 3.3: Distribution for Relevent Experience or Not

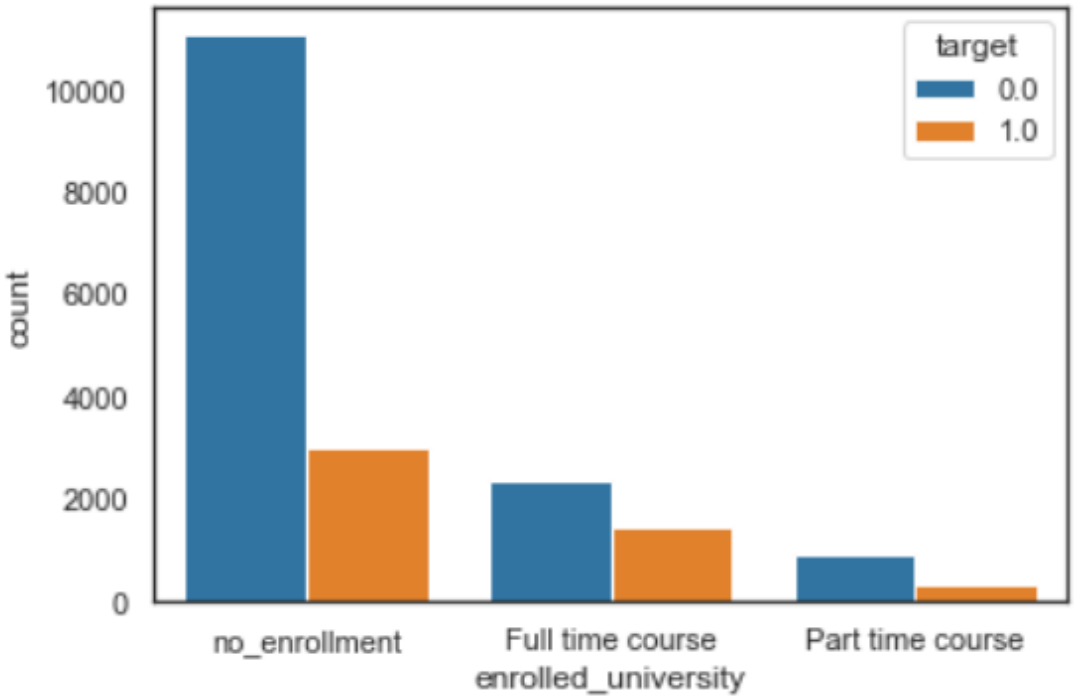

Figure 3.4: Distribution of Enrolled University

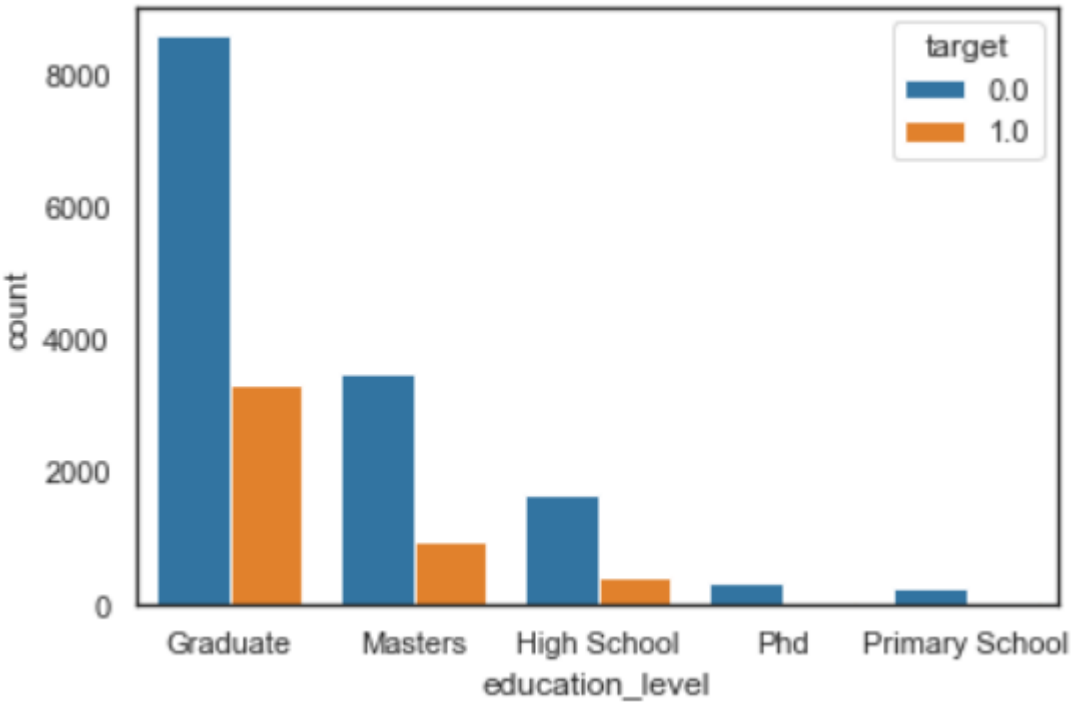

Figure 3.5: Distribution of Education Level

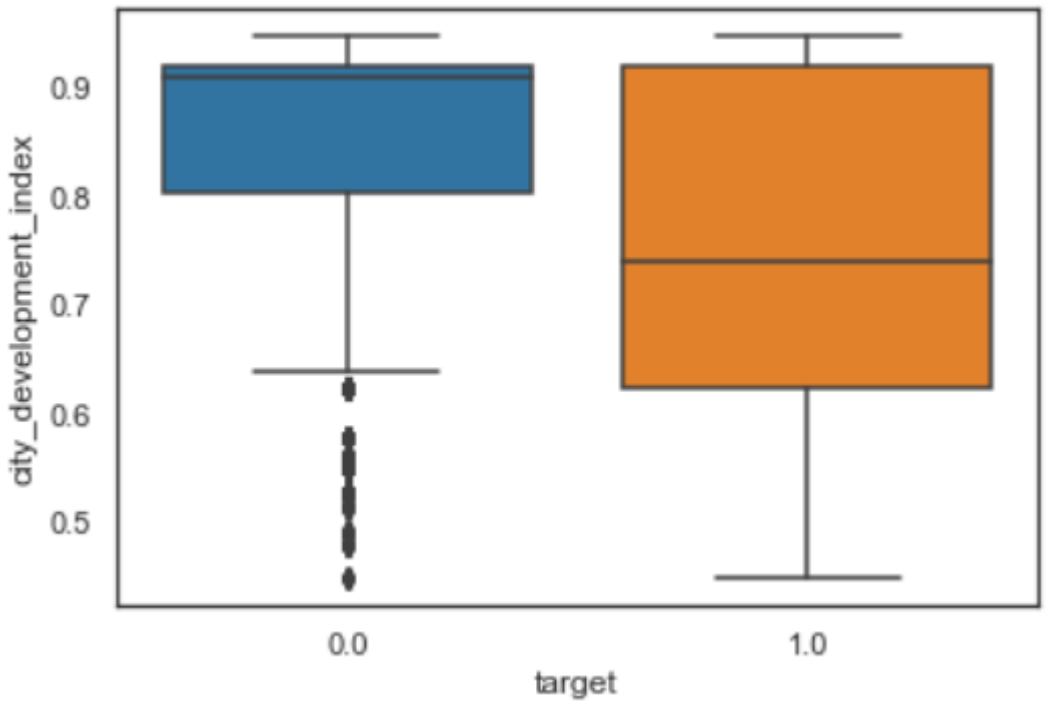

Figure 3.6: Boxplot for City Development Index

3.2 Data Cleaning and Pre-processing

Several actions are taken to clean and prepare the data for analysis. These actions include:

a. Removing any unnecessary or irrelevant data from the dataset. In this dataset, the 'enrolle_id' and 'city' features are not important or useful in data analysis. Thus, both of these features are removed.
b. Checking for and handling any missing or null values in the data. All Not a Number (NaN) are replaced by preceding values or next values.
c. Ensuring that all data is in the correct format and type for analysis.
d. Normalizing or standardizing any numerical data to ensure that it is on the same scale. The numerical values are standardized using the StandardScaler algorithms from sklearn.preprocessing library.
e. Encoding any categorical data as numerical data to prepare it for analysis. The categorical values are converted into a format that may be fed into machine learning algorithms by One-hot Encoding. Label Encoding is not used in this study as it will make the data seem that there is a ranking between values.
f. Checking for and handling any outliers in the data that may skew the results of the analysis.

3.3 Data Splitting

Before entering model training progress, the dataset is split into training and testing sets with ratio of 80/20. The training set is used to train the model while the testing set is used to evaluate the performance of the model. The dataset is split into independent variables, X and dependent variable, y. The dataset is also split into same percentages of classes in each dependent variable so that both of the training and testing sets are balanced.

- The size of training set for X is (15326, 63) while for y is (15326, 1)
- The size of testing set for X is (3832, 63) while for y is (3832, 1)

Chapter 4 : Model Training

In this analysis, at least three different classifier models will be trained and tested in order to identify the model that provides the best balance of explainability and predictability. These models may include:

a. Logistic Regression
b. Random Forest Classifier
c. XGBoosting Classifier
d. K-Nearest Neighbors Classifier

4.1 Logistic Regression

1. Model Training (Logistic Regression)

```
# Train model using Logistic Regression
lr = LogisticRegression()
lr_params = {'penalty': ['l2','l1','elasticnet'],
             'C': [0.1*n for n in range(10)],
             'solver': ['lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga']}
lr_cv = GridSearchCV(estimator=lr, param_grid=lr_params, cv=4)
lr_cv.fit(X_train, y_train)
```

```
GridSearchCV(cv=4, estimator=LogisticRegression(),
             param_grid={'C': [0.0, 0.1, 0.2, 0.30000000000000004, 0.4, 0.5,
                               0.6000000000000001, 0.7000000000000001, 0.8,
                               0.9],
                         'penalty': ['l2', 'l1', 'elasticnet'],
                         'solver': ['lbfgs', 'liblinear', 'newton-cg',
                                    'newton-cholesky', 'sag', 'saga']})
```

```
# Evaluate the metrics of the Logistic Regression Model
lr_yhat = lr_cv.predict(X_test)
lr_metrics = evaluate_metric(y_test, lr_yhat)
lr_metrics
```

```
{'accuracy': 0.763, 'precision': 0.729, 'recall': 0.763, 'f1_score': 0.725}
```

```
# Confusion matrix of Logistic Regression Model
plot_confusion_matrix(y_test, lr_yhat)
```

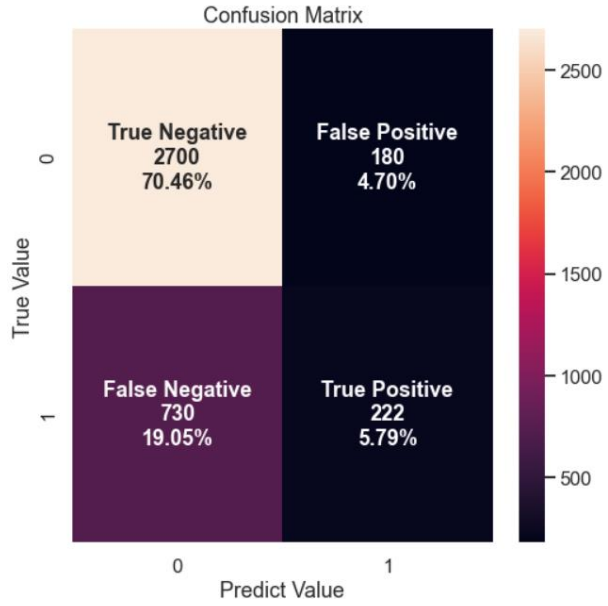

Figure 4.1: Logistic Regression Model

4.2 Random Forest Classifier

2. Model Training (Random Forest Classifier)

```
# Train model using Random Forest Classifier
rf = RandomForestClassifier()
rf_params= {'criterion': ['gini','entropy','log_loss'],
           'max_depth': [1+n*2 for n in range(5)],
           'n_estimators': [20*n for n in range(1,10)]}
rf_cv = GridSearchCV(rf, rf_params, cv=4)
rf_cv.fit(X_train, y_train)
```

```
GridSearchCV(cv=4, estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy', 'log_loss'],
                         'max_depth': [1, 3, 5, 7, 9],
                         'n_estimators': [20, 40, 60, 80, 100, 120, 140, 160,
                                          180]})
```

```
# Hyperparamters of Random Forest Classifier
rf_cv.best_params_
```

```
{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 60}
```

```
# Metrics evaluation of Random Forest Classifier
rf_yhat = rf_cv.predict(X_test)
rf_metrics = evaluate_metric(y_test, rf_yhat)
rf_metrics
```

```
{'accuracy': 0.763, 'precision': 0.728, 'recall': 0.763, 'f1_score': 0.705}
```

```
# Confusion matrix
plot_confusion_matrix(y_test, rf_yhat)
```

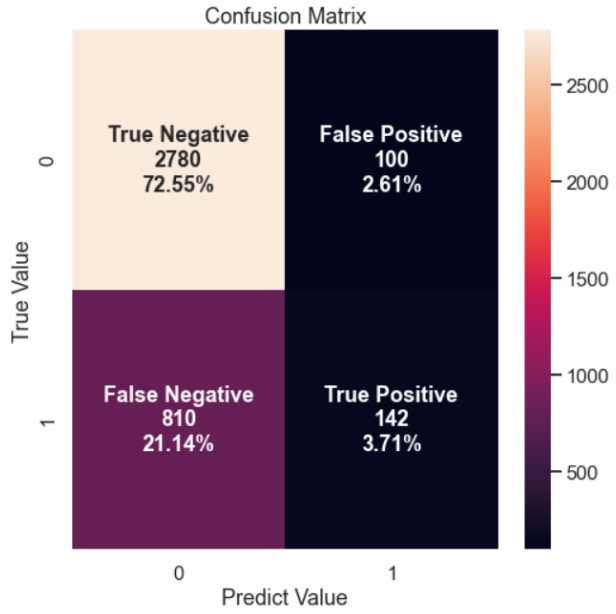

Figure 4.2: Random Forest Classifier Model

4.3 XGBoosting Classifier

3. Model Training (XGBoosting Classifier)

```
xgb = XGBClassifier()
xgb_params = {'learning_rate': [0.1*n+1 for n in range(10)],
              'n_estimators': [20*n for n in range(10)]}
xgb_cv = GridSearchCV(xgb, xgb_params, cv=4)
xgb_cv.fit(X_train, y_train)
```

```
xgb_cv.best_params_
```

```
{'learning_rate': 1.0, 'n_estimators': 20}
```

```
xgb_yhat = xgb_cv.predict(X_test)
xgb_metrics = evaluate_metric(y_test, xgb_yhat)
xgb_metrics
```

```
{'accuracy': 0.758, 'precision': 0.737, 'recall': 0.758, 'f1_score': 0.743}
```

```
plot_confusion_matrix(y_test, xgb_yhat)
```

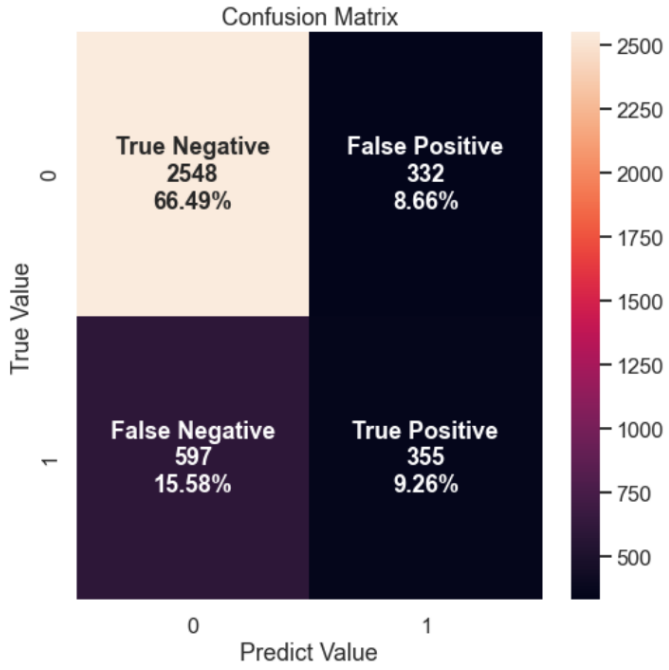

Figure 4.3: XGBoosting Classifier Model

4.4 K-Nearest Neighbors Classifier

4. Model Training (KNearestNeighbors Classifier)

```
knn = KNeighborsClassifier()
knn_params = {'n_neighbors': [n for n in range(1,15)],
              'weights': ['uniform', 'distance']}
knn_cv = GridSearchCV(knn, knn_params, cv=4)
knn_cv.fit(X_train, y_train)
```

```
GridSearchCV(cv=4, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                         13, 14],
                         'weights': ['uniform', 'distance']})
```

```
knn_cv.best_params_
```

```
{'n_neighbors': 14, 'weights': 'uniform'}
```

```
knn_yhat = knn_cv.predict(X_test)
knn_metrics = evaluate_metric(y_test, knn_yhat)
knn_metrics
```

```
{'accuracy': 0.752, 'precision': 0.698, 'recall': 0.752, 'f1_score': 0.67}
```

```
plot_confusion_matrix(y_test, knn_yhat)
```

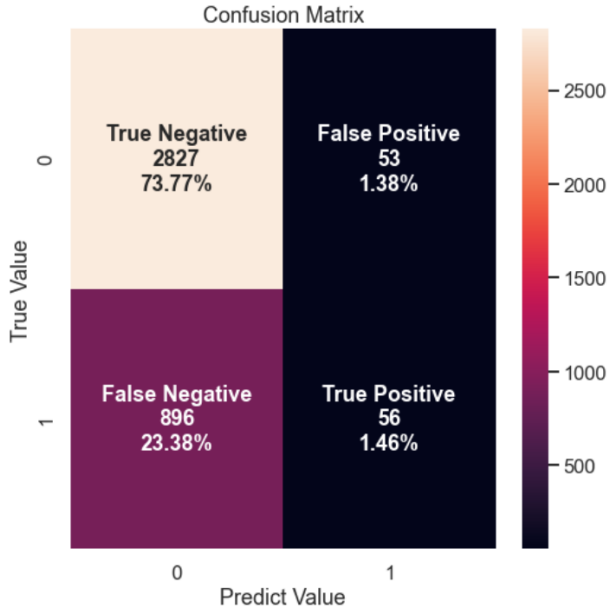

Figure 4.4: K-Nearest Neighbors Classifier Model

4.5 Best Model

The performance of each model is evaluated by some important metrics in classification model. These metrics are included Accuracy score, Precision score, Recall score and F1 score.

Model	Accuracy	Precision	Recall	F1_score
XGBoosting Classifier	0.758	0.737	0.758	0.743
Logistic Regression	0.763	0.729	0.763	0.725
Random Forest Classifier	0.763	0.728	0.763	0.705
K-Nearest Neighbors Classifier	0.752	0.698	0.752	0.670

Figure 4.5: Performance of Each Model

Based on the figure above, in terms of Accuracy score, Logistic Regression and Random Forest Classifier got the highest score. However, the dataset is imbalanced as shown in Figure 2.2. Thus, F1 score is better metrics than Accuracy score when there are imbalanced classes. Hence, in terms of F1 score, XGBoosting got the highest score, and it is considered as the best machine learning model among others.

Chapter 5 : Results and Discussion

There are few factors that influence the job employment of the Big Data and Data Science company. The factors included:

a. **City development index of candidates**

   From Figure 3.5, the boxplot show that the candidates who wish to work with the company are mostly from lower city development index. These findings can prove that candidates from lower city development index are more likely to work with such company compared to candidates from higher city development. This may due to candidates from lower city development index are less exposed to other companies.

b. Relevant Experience

   From Figure 3.2, the histogram shows that there is a higher percentage of candidates with no relevant experience are more likely to work with Big Data and Data Science company compared to candidates who has relevant experience. Due to imbalanced data, there are higher candidates with relevant experience compared to candidates with no relevant experience. However, the percentages of candidates with no relevant experience are more willing to work in the company is higher. This may due to the candidates with no relevant experience are less exposed to data science field, and less confidence in their potential in such field.

c. XGBoosting

   XGBoosting is the best classifier model to classify whether the candidates are willing to work with the company after training. In this imbalanced data, F1 score is better metric than Accuracy score. Thus, XGBoosting with the highest score among the four classifier evaluated in this study, becoming the best classifier model.

Chapter 6 : Suggestion

In the model, there are imbalanced data fitted into training model. Hence, the minority target should be oversampled to make sure the dataset is imbalanced. There are several potential next steps that could be taken to continue analysing this data and further improve the model's performance and explanatory power. Some suggestions for next steps might include:

a. Adding additional data features that may be relevant to understanding job change among data scientists. This could include data on factors such as job satisfaction, career advancement opportunities, or workplace culture.
b. Revisiting the model after adding these additional data features to see if they have a significant impact on the model's performance and explanatory power.
c. Exploring different modelling techniques or algorithms in order to identify any other models that may provide better performance or explainability.
d. Applying different pre-processing techniques or feature engineering approaches to the data in order to extract more information or to improve the model's performance.
e. Validating the model's performance on an additional, independent dataset in order to ensure that the model is generalizable to new, real-world data.

Overall, there are many potential next steps that could be taken in order to continue analyzing this data and to further improve the model's performance and explanatory power. The specific next steps that are taken will depend on the specific goals and needs of the business or stakeholders.