ML-2 Project on Easy Visa (ML-2 Coded)

Apr B Sunday Onkar 10:30 AM Batch

Arindam Saha

Table of Contents

List of Figures

List of Tables

1. Background

Business communities in the United States are facing high demand for human resources, but one of the constant challenges is identifying and attracting the right talent, which is perhaps the most important element in remaining competitive. Companies in the United States look for hard-working, talented, and qualified individuals both locally as well as abroad.

The Immigration and Nationality Act (INA) of the US permits foreign workers to come to the United States to work on either a temporary or permanent basis. The act also protects US workers against adverse impacts on their wages or working conditions by ensuring US employers' compliance with statutory requirements when they hire foreign workers to fill workforce shortages. The immigration programs are administered by the Office of Foreign Labor Certification (OFLC).

OFLC processes job certification applications for employers seeking to bring foreign workers into the United States and grants certifications in those cases where employers can demonstrate that there are not sufficient US workers available to perform the work at wages that meet or exceed the wage paid for the occupation in the area of intended employment.

2. Objective

In FY 2016, the OFLC processed 775,979 employer applications for 1,699,957 positions for temporary and permanent labor certifications. This was a nine percent increase in the overall number of processed applications from the previous year. The process of reviewing every case is becoming a tedious task as the number of applicants is increasing every year.

The increasing number of applicants every year calls for a Machine Learning based solution that can help in shortlisting the candidates having higher chances of VISA approval. OFLC has hired the firm EasyVisa for data-driven solutions. You as a data scientist at EasyVisa have to analyze the data provided and, with the help of a classification model:

Facilitate the process of visa approvals. Recommend a suitable profile for the applicants for whom the visa should be certified or denied based on the drivers that significantly influence the case status.

3. Data Dictionary

Variables	Description
case_id	ID of each visa application
continent	Information of continent the employee
education_of_employee	Information of education of the employee
has_job_experience	Does the employee has any job experience? Y= Yes; N = No
requires_job_training	Does the employee require any job training? Y = Yes; N = No
no_of_employees	Number of employees in the employer's company
yr_of_estab	Year in which the employer's company was established
region_of_employment	Information of foreign worker's intended region of employment in the US.
prevailing_wage	Average wage paid to similarly employed workers in a specific occupation in the area of intended employment. The purpose of the prevailing wage is to ensure that the foreign worker is not underpaid compared to other workers offering the same or similar service in the same area of employment.
unit_of_wage	Unit of prevailing wage. Values include Hourly, Weekly, Monthly, and Yearly.

full_time_position	Is the position of work full-time? Y = Full-Time Position; N = Part-Time Position
case_status	Flag indicating if the Visa was certified or denied

Table 1: Data Dictionary

4. Data Information

Note:-

- There are in total 25480 rows and 12 columns present.
- There are 3 numeric (float and int type) and 9 string (object type) columns in the data.
- The target variable is the case_status (certified / denied), which is categorical type.
- prevailing_wage, no_of_employees, yr_of_estab are of numerical type.
- There are no missing values in the dataset.
- Converting "objects" to "category" reduces the data space required to store the dataframe.
- There are no duplicate values in the dataset.
- There are no missing values.
- case_id is an ID variable and not useful for modelling. Hence, we have removed in greeding method.
- Mean no_of_employees is 5667.04 and 75 percentile come under 3504 and maximum is at 602069, it has outliers.
- Mean prevailing_wage is 74455.81 and 75 percentile comes in 107735.51 and maximum is at 319210.27, seems to have outliers.
- Maximum of the employee's education level is Bachelor's.
- Top region of employment is Northeast
- Top case status is Certified.
- Most of the eployees doesn't require job trainings i.e. they have some experience.
- Maximum full-time positions are in YES.
- In total 6 unique continents are present ('Asia', 'Africa', 'North America', 'Europe', 'South America', 'Oceania')
- 4 types of education levels ('High School', 'Master's', 'Bachelor's', 'Doctorate')
- Job Experience / job training requirement/ full time position is either 'N' or 'Y'
- Region of employments are of 5 types 'West', 'Northeast', 'South', 'Midwest', 'Island'
- Unit of wages are of 4 types 'Hour', 'Year', 'Week', 'Month'
- Unique values in continent are :

Asia 16861

Europe 3732

North America 3292

South America 852

Africa 551

Oceania 192

- Unique values in education_of_employee are :

Bachelor's 10234

Master's 9634

High School 3420

Doctorate 2192

- Unique values in has_job_experience are :

Y 14802

N 10678

- Unique values in requires_job_training are :

N 22525

Y 2955

- Unique values in region_of_employment are :

Northeast 7195

South 7017

West 6586

Midwest 4307

- Unique values in unit_of_wage are :

unit_of_wage

Year 22962

Hour 2157

Week 272

Month 89

- Unique values in full_time_position are :

full_time_position

Y 22773

N 2707

- Unique values in case_status are :

case_status

Certified 17018

Denied 8462

5. Exploratory Data Analysis

5.1. Univariate analysis

5.1.1. Observations on prevailing_wage

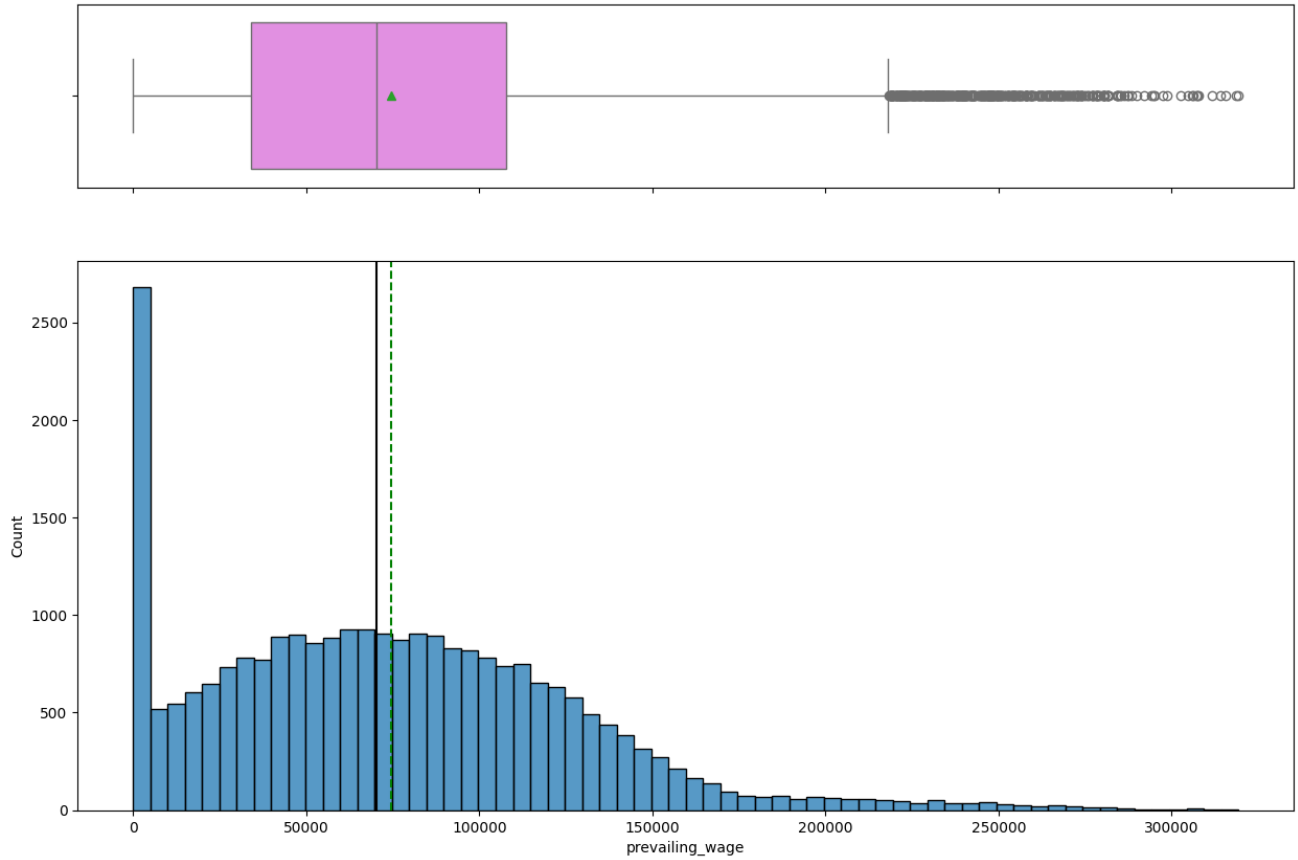

Figure 1: Univariate on prevailing wage

- The distribution is slightly right skewed.

5.1.2. Observation on no_of_employees

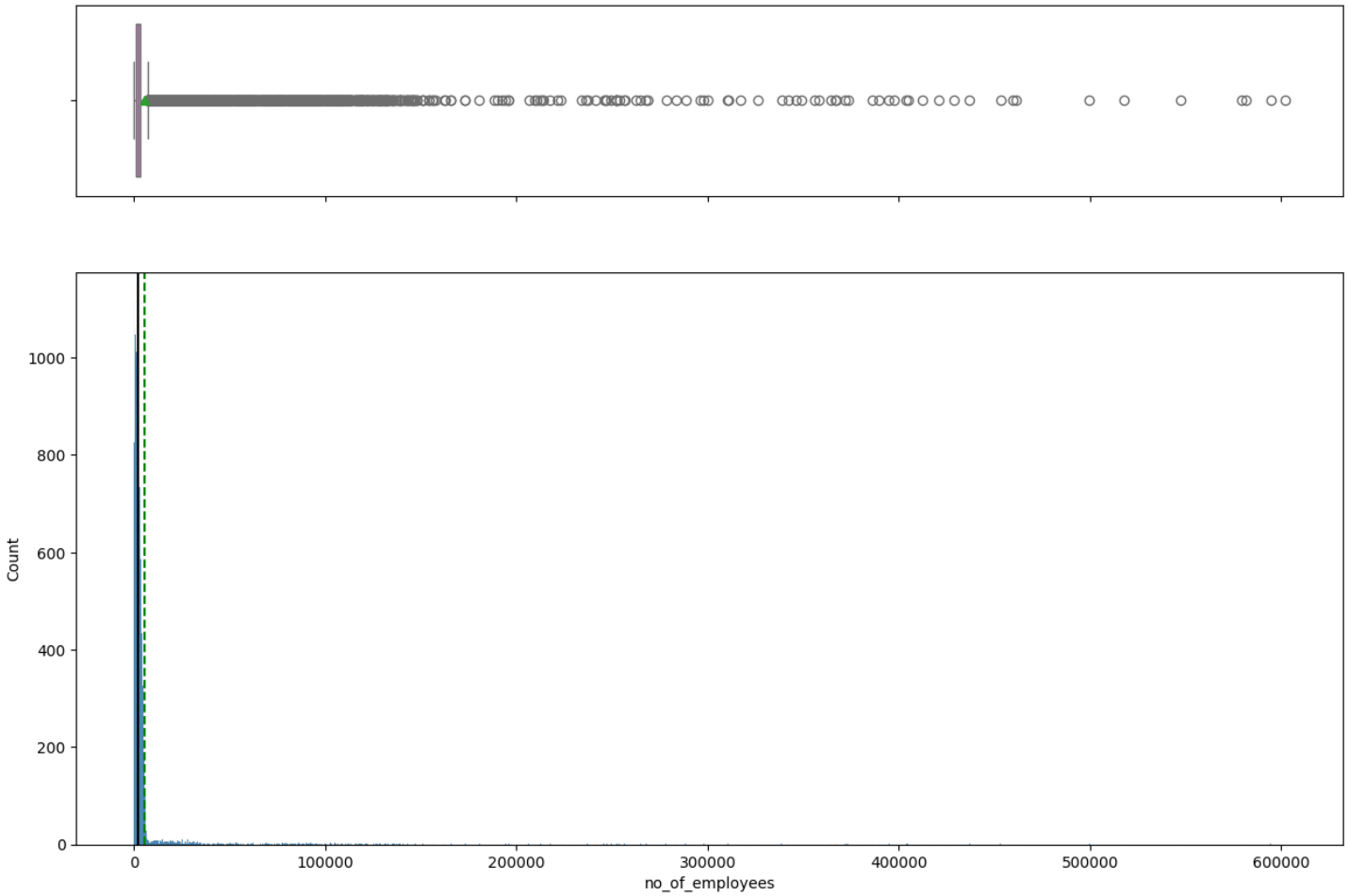

Figure 2: Univariate No of employees

- Most of the observation is 0 no of employees.

5.1.3. Observation on education_of_employee

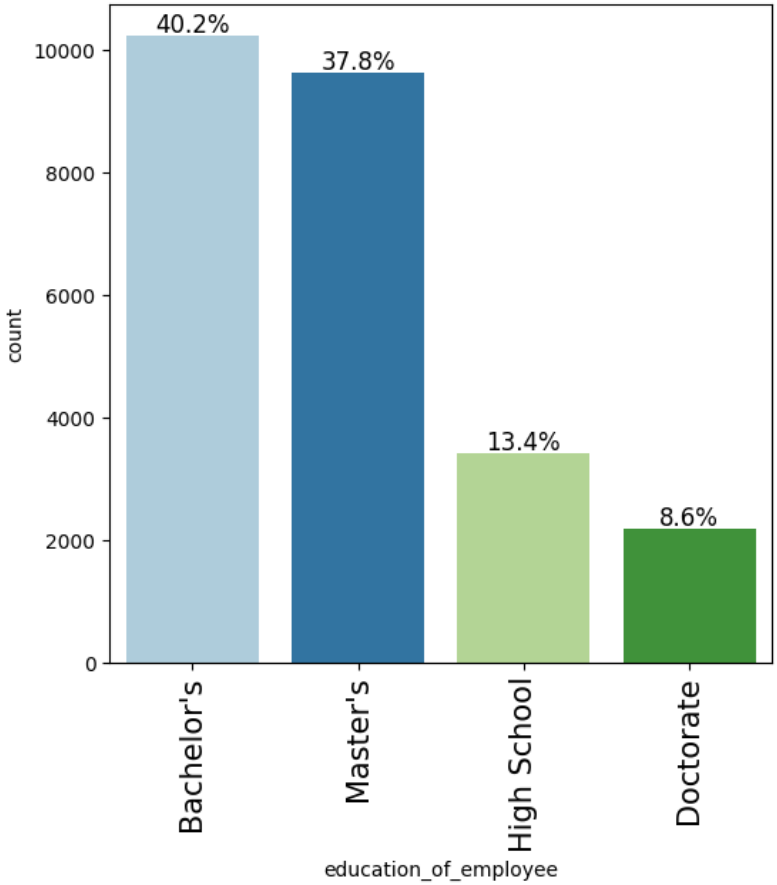

Figure 3: Univariate on education of employee

- Most of the employees are having either bachelor's or master's as their education status.

5.1.4. Observation on has_job_experience

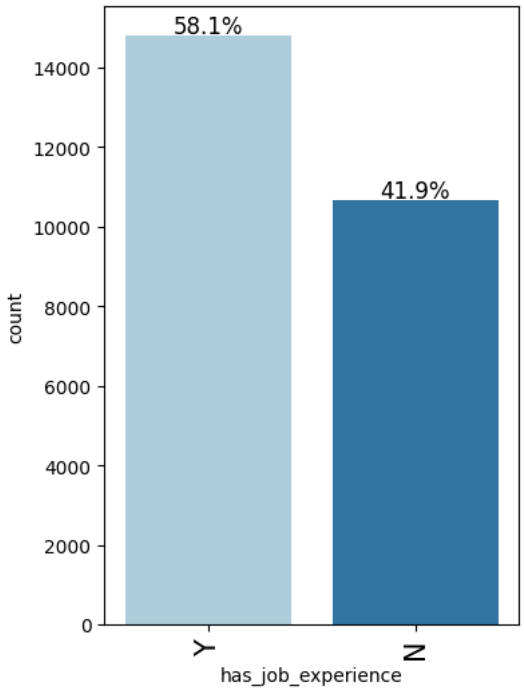

Figure 4: Univariate on job experience

- Around 58% of the employees have prior job experience rest does not have.

5.1.5. Observation on requires_job_training

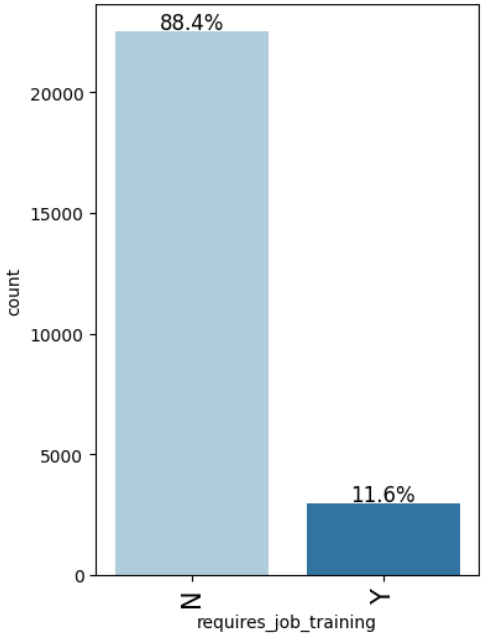

Figure 5: Univariate on job training

- 88% of the employees need job training.

5.1.6. Observation on full_time_position

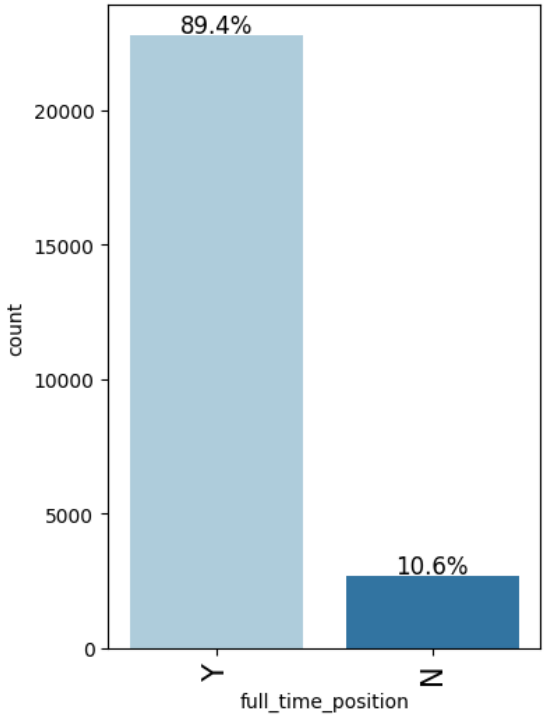

Figure 6: Univariate on Full time Position

- Around 89.4% of the employees has full time position.

5.1.7. Observation on region_of_employment

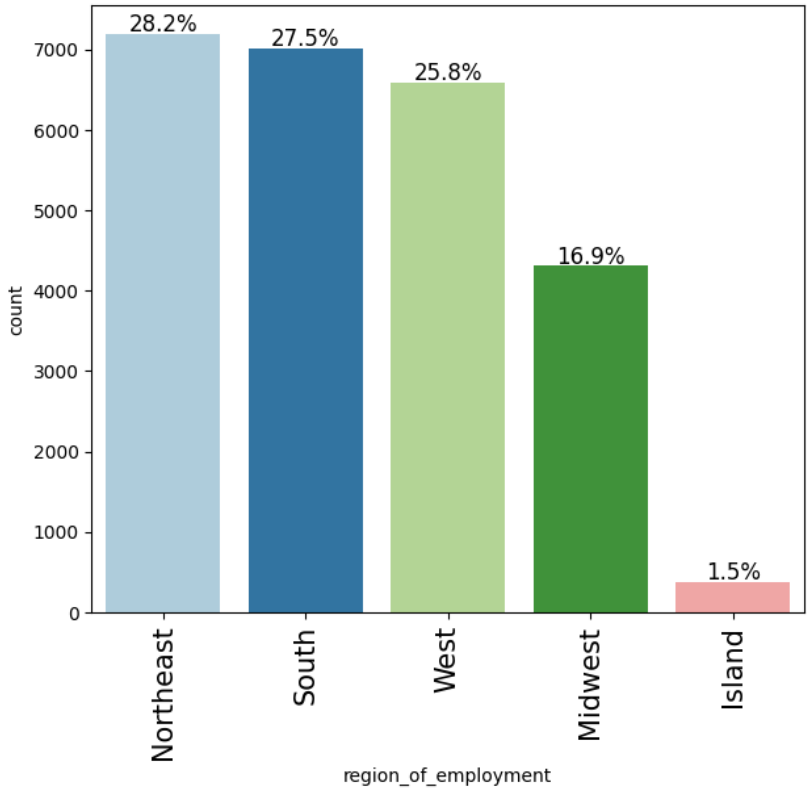

Figure 7: Univariate on region of employment

- Most of the employments were occurred in northeast or south or west region.

5.2. Bivariate Analysis

5.2.1. Pari Plot and Heat Map

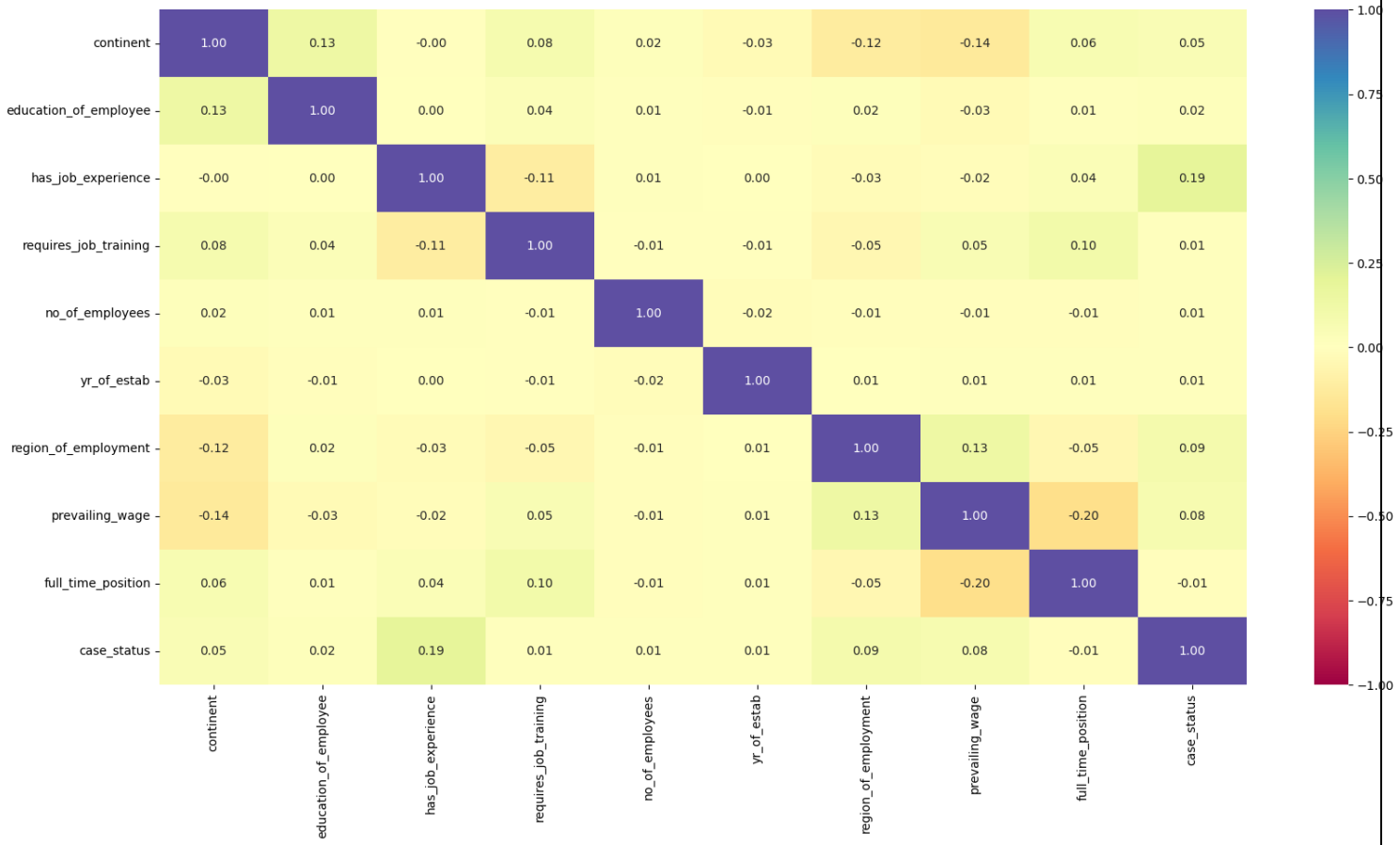

Figure 8: Heat Map

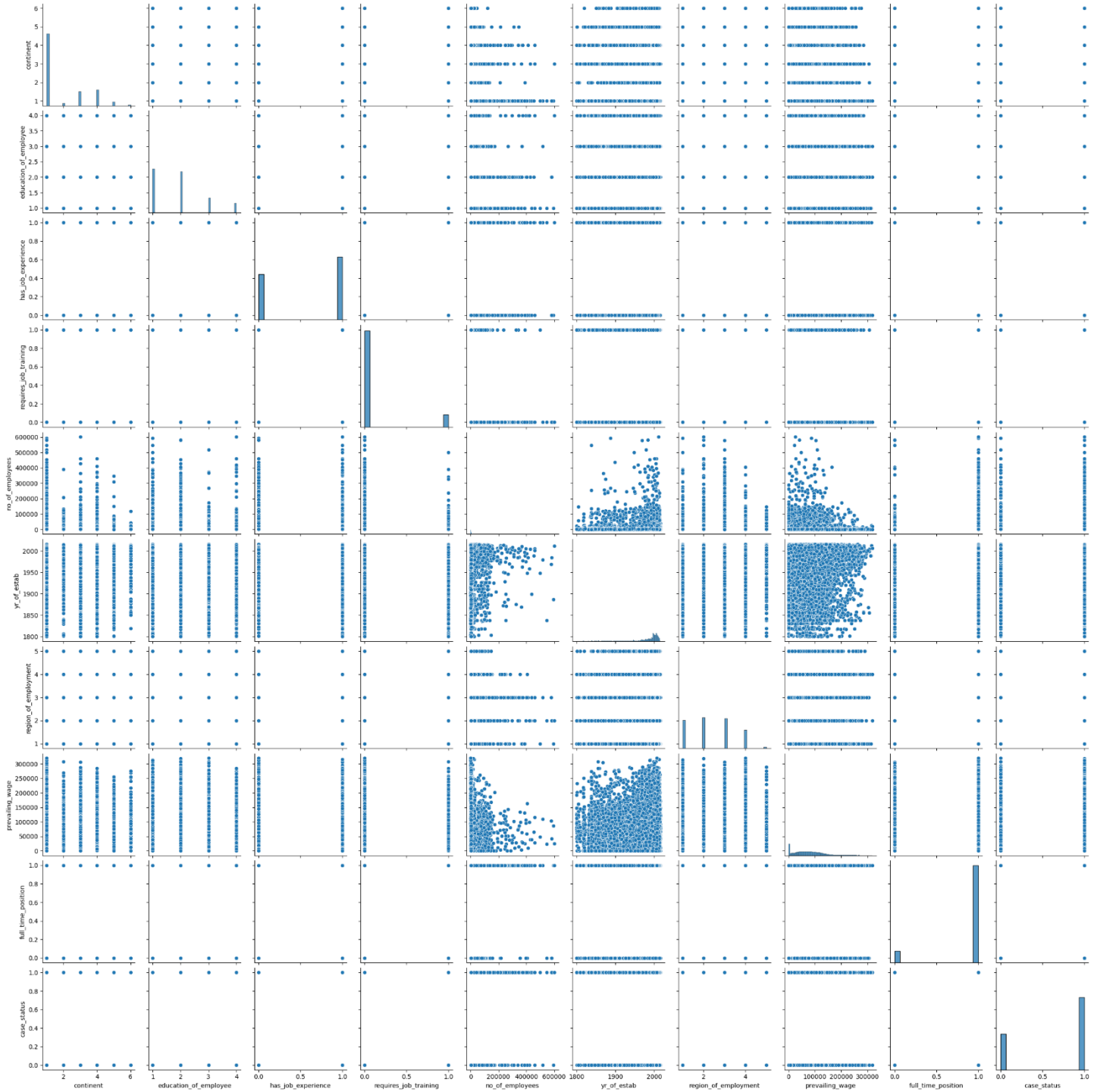

Figure 9: Pair Plot

Observations

- Job experience has some positive correlations with case status.
- Region, continental, and full-time position have some correlations with the prevailing wage.

5.2.2. Observation on case status w.r.t. job experience

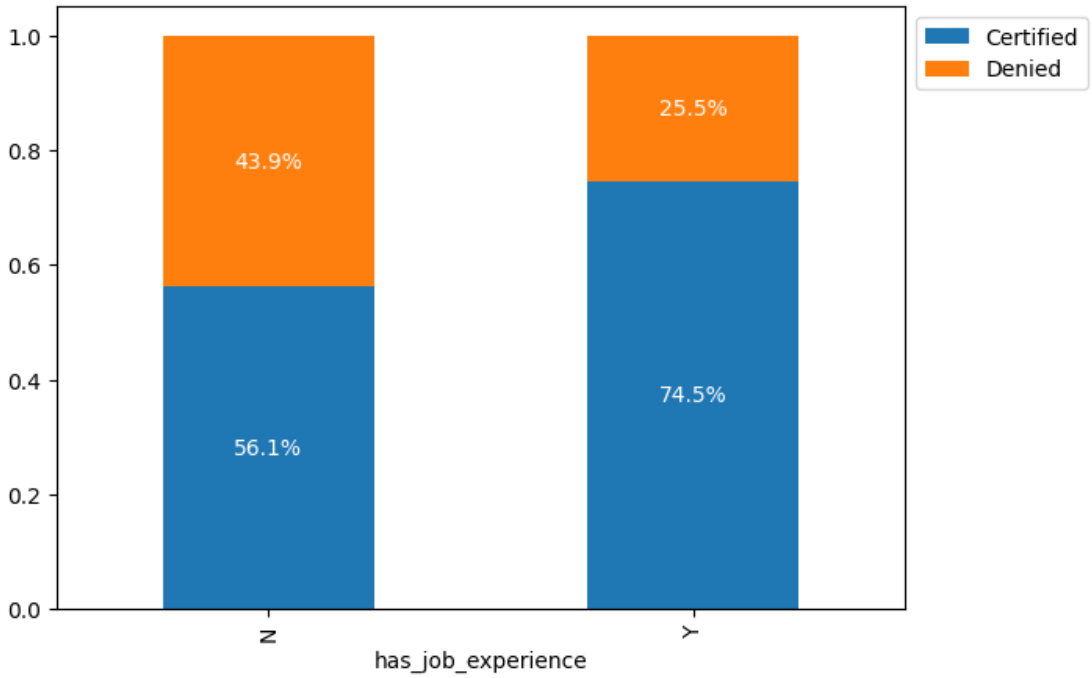

Figure 10: Bivariate analysis case status w.r.t. job experience

- It is clear that an employee is having job experience, has more probability to get certified.

5.2.3. Observations on full time position w.r.t. prevailing wage

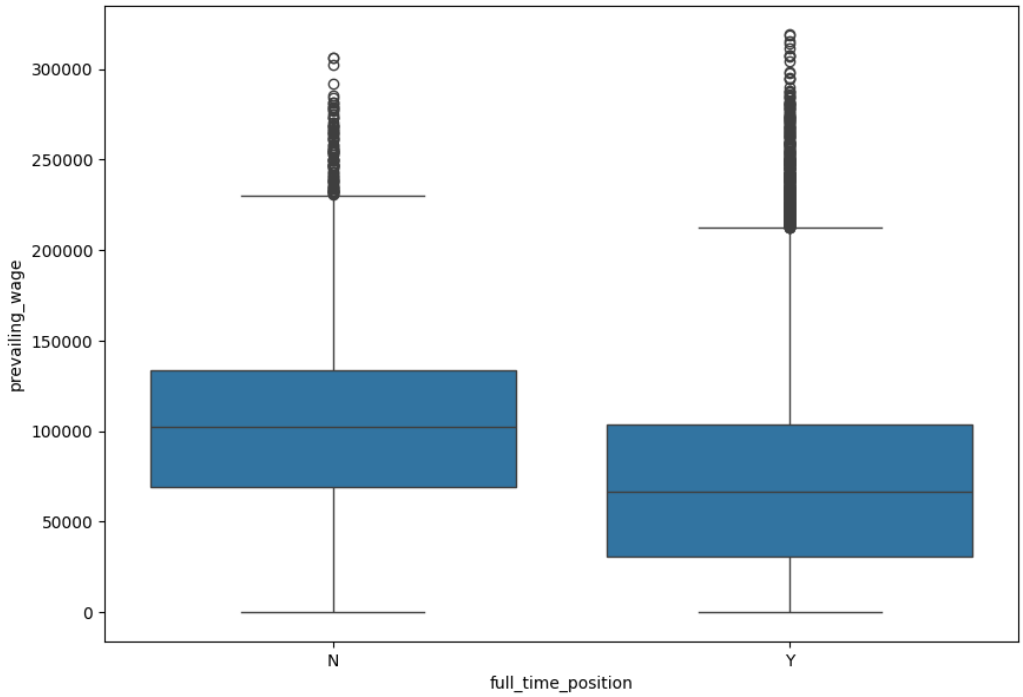

Figure 11: Boxplot full time position vs prevailing wage

Observations

- Outliers can be observed towards right side.

- Average wages are more for those employees who are in part time position.

5.2.4. Observations on continent w.r.t. prevailing wage

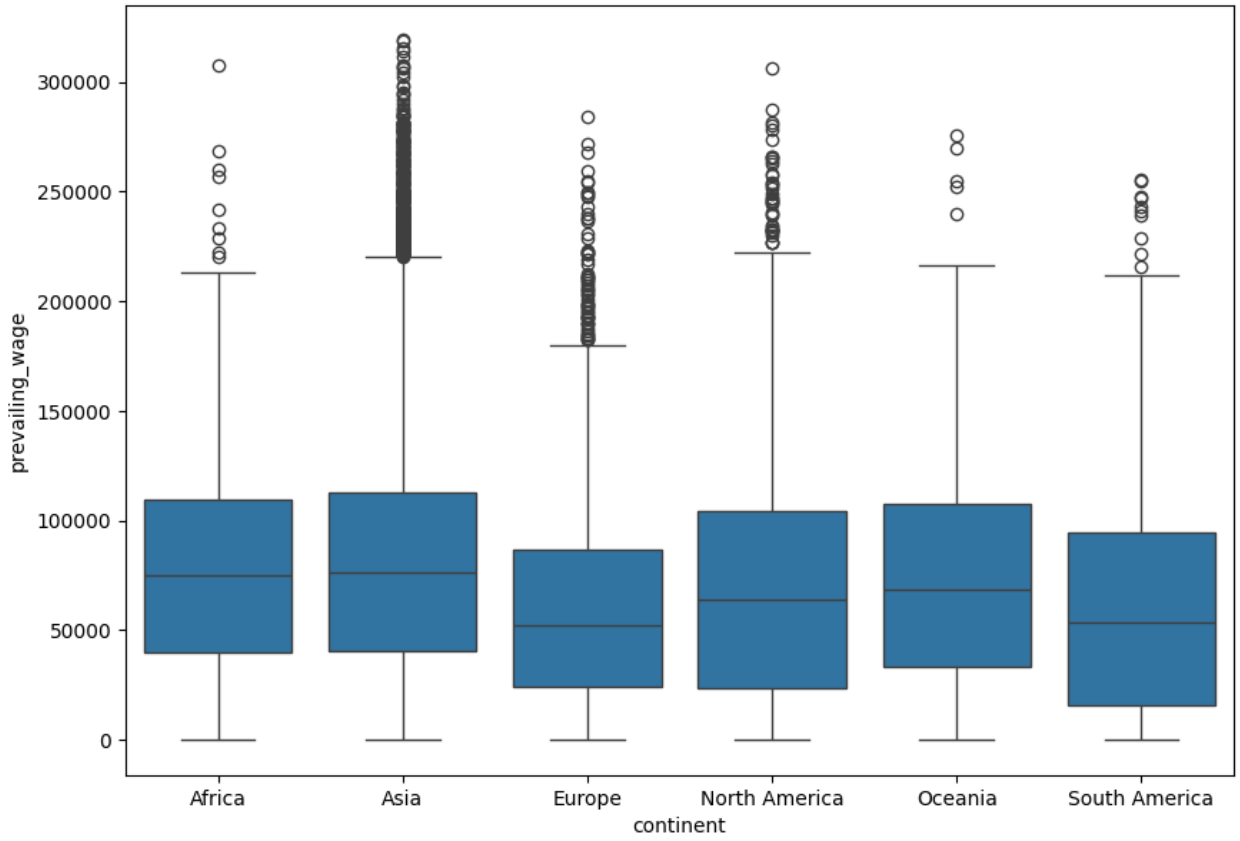

Figure 12: Bivariate analysis on prevailing wage w.r.t. continent

Observations

- Outliers can be observed for all the continents.
- Europe and South America has lower average wages.

6. Data Preprocessing

6.1. Outlier Detection and Treatment

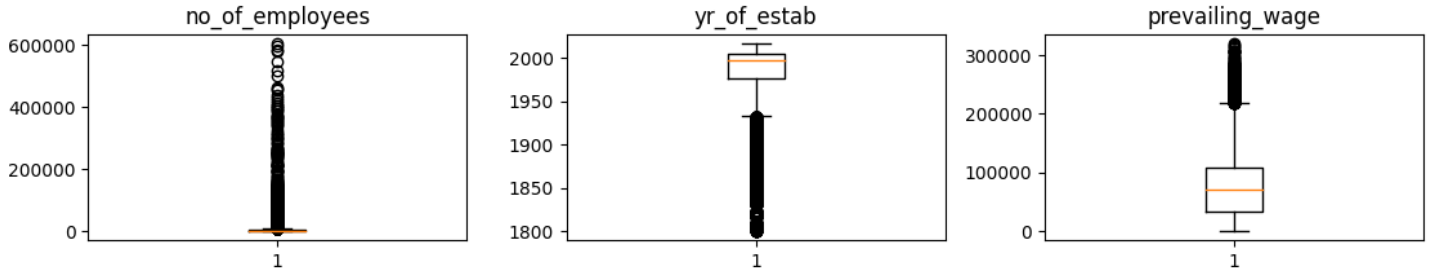

Figure 13: Outliers detection

- There are much a few outliers in the data.
- However, we will not treat them as they are proper values.

6.2. Data Preparation for model building

- When classification problems exhibit a significant imbalance in the distribution of the target classes, it is good to use stratified sampling to ensure that relative class frequencies are approximately preserved in train and test sets.
- This is done using the stratify parameter in the train_test_split function.
- We have divided our data in to train and test 70:30 respectively.
- Our train set contains 17836 and test contains 7644 no of observations.

7. Model Building

Model evaluation criterion

Model can make wrong predictions as:

- Predicting an employee will get certified and the employee doesn't get certified
- Predicting an employee will not get certified and the employee get certified

Which case is more important?

- Predicting that employee will not get certified but he/she certified i.e. not providing valuable to the employee.

How to reduce this loss i.e need to reduce False Negatives?

- Company wants Recall to be maximized, greater the Recall higher the chances of minimizing false negatives. Hence, the focus should be on increasing Recall or minimizing the false negatives or in other words identifying the true positives (i.e. Class 1) so that the company can provide US visa certification to the deserving employes.

Our model will be based on classification as our target is classified in two categories.

7.1. Decision tree model

- We will build our model using the DecisionTreeClassifier function. Using default 'gini' criteria to split.
- If the frequency of class A is 10% and the frequency of class B is 90%, then class B will become the dominant class and the decision tree will become biased toward the dominant classes.
- In this case, we can pass a dictionary {0:0.17,1:0.83} to the model to specify the weight of each class and the decision tree will give more weightage to class 1.
- class_weight is a hyperparameter for the decision tree classifier.

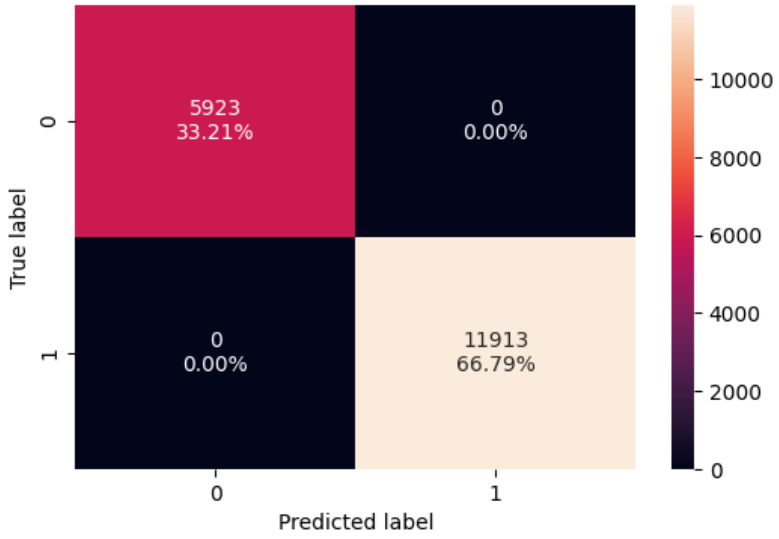

Figure 14: Decision tree training set confusion matrix

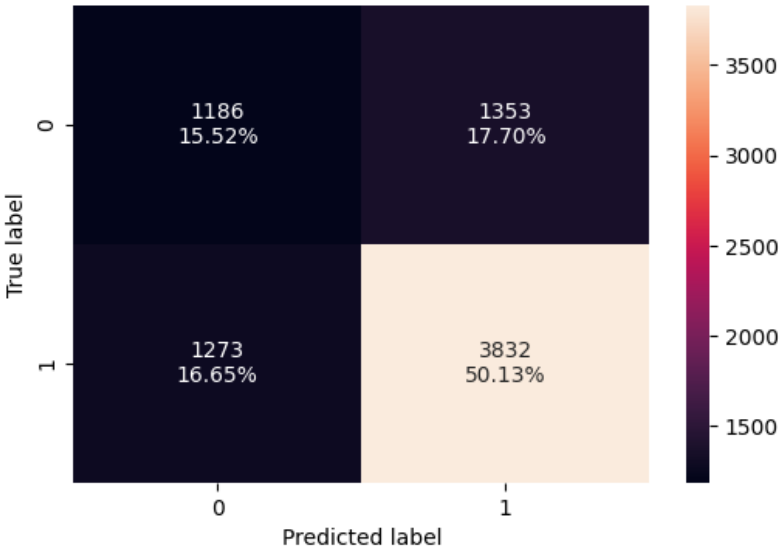

Figure 15: Decision tree testing set confusion matrix

Training performance			
Accuracy	Recall	Precision	f1
1	1	1	1
Testing performance			
0.656463	0.750637	0.739055	0.744801

Table 2: Decision tree model performance

- Decision tree is working well on the training data but is not able to generalize well on the test data concerning the recall.

7.2. Bagging Classifier Model

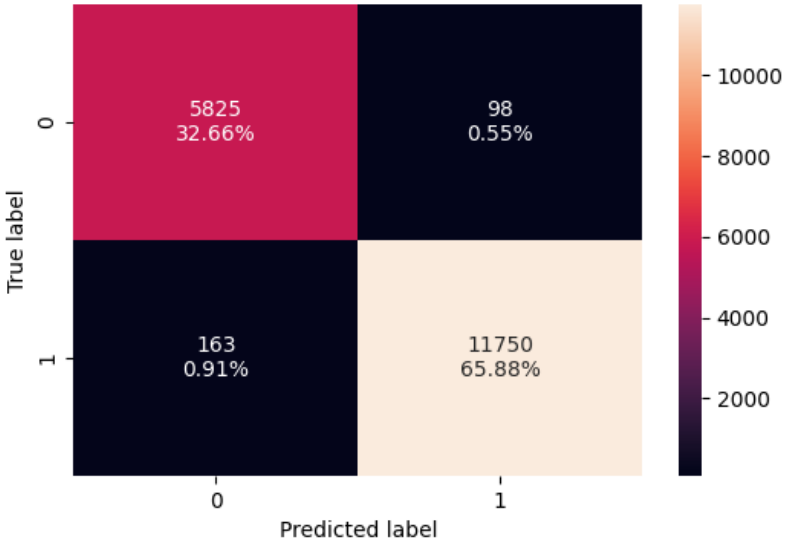

Figure 16: Bagging Classifier Training Set Confusion matrix

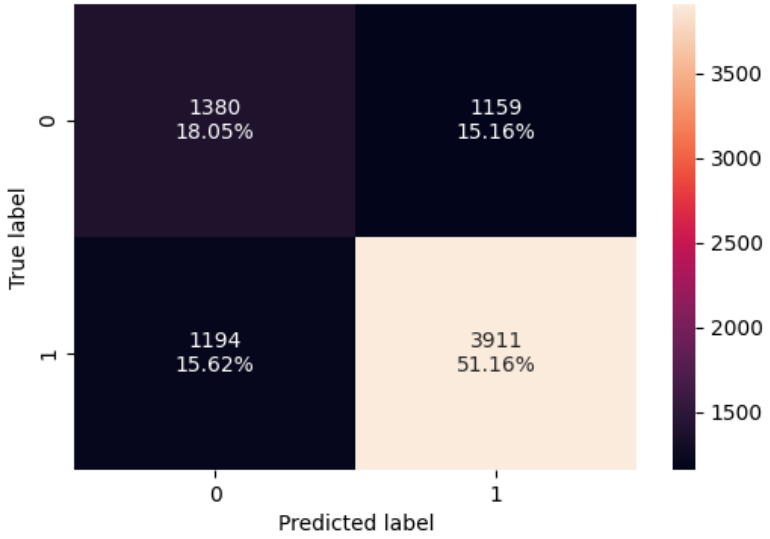

Figure 17: Bagging Classifier Testing Set Confusion matrix

Training performance			
Accuracy	Recall	Precision	f1
0.985367	0.986317	0.991729	0.989016
Testing performance			
0.692177	0.766112	0.7714	0.768747

Table 3:Bagging Classifier Model Performance

- Bagging classifier is overfitting on the training set and is performing poorly on the test set in terms of recall.

Bagging Classifier with weighted decision tree

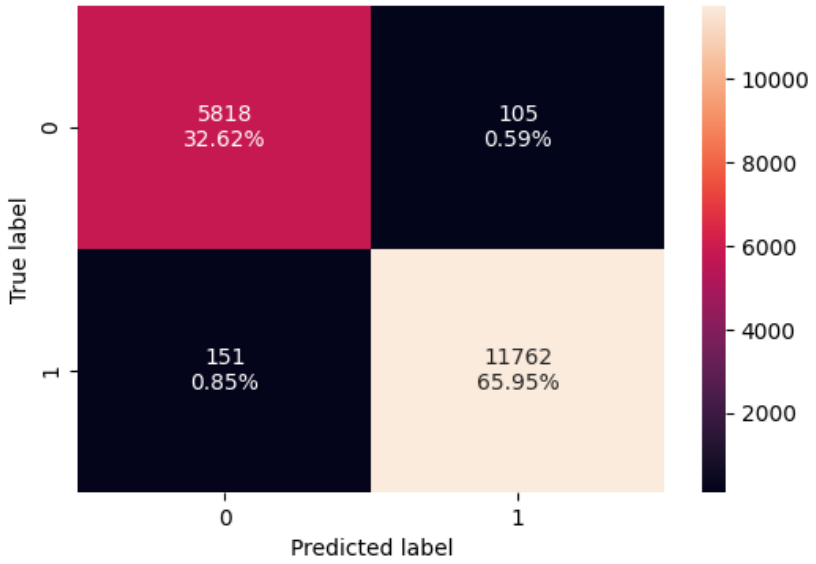

Figure 18: Weighted Bagging Classifier Training Set Confusion Matrix

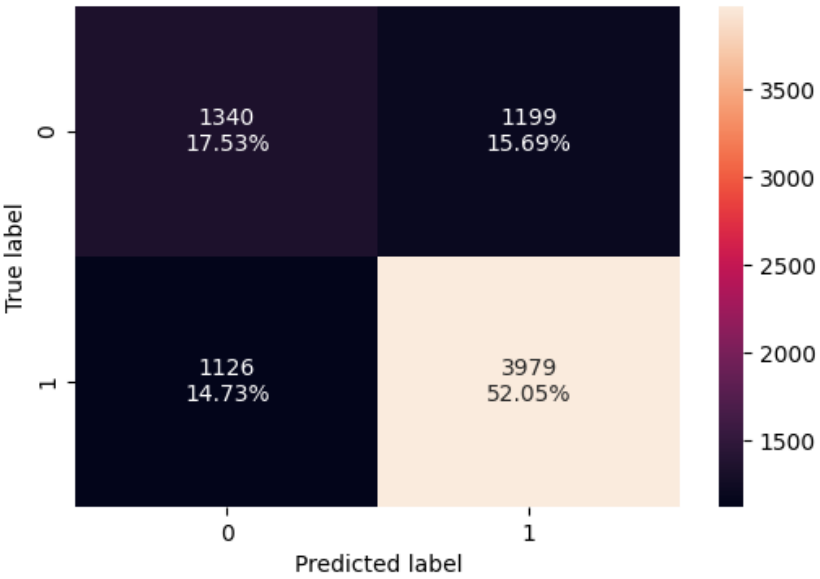

Figure 19: Weighted Bagging Classifier Testing Set Confusion Matrix

Training performance			
Accuracy	Recall	Precision	f1
0.985647	0.987325	0.991152	0.989235
Testing performance			
0.69584	0.779432	0.768443	0.773899

Figure 20: Weighted Bagging Classifier Model Performance

- Bagging classifier with a weighted decision tree is giving very good accuracy and prediction but is not able to generalize well on test data in terms of recall.

7.3. Random Forest Classifier

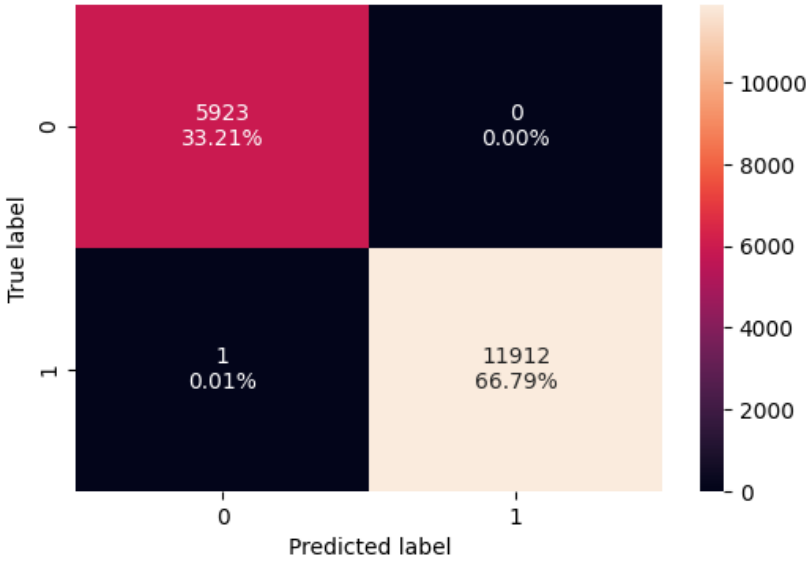

Figure 21: Random Forest Classifier Training Set Confusion Matrix

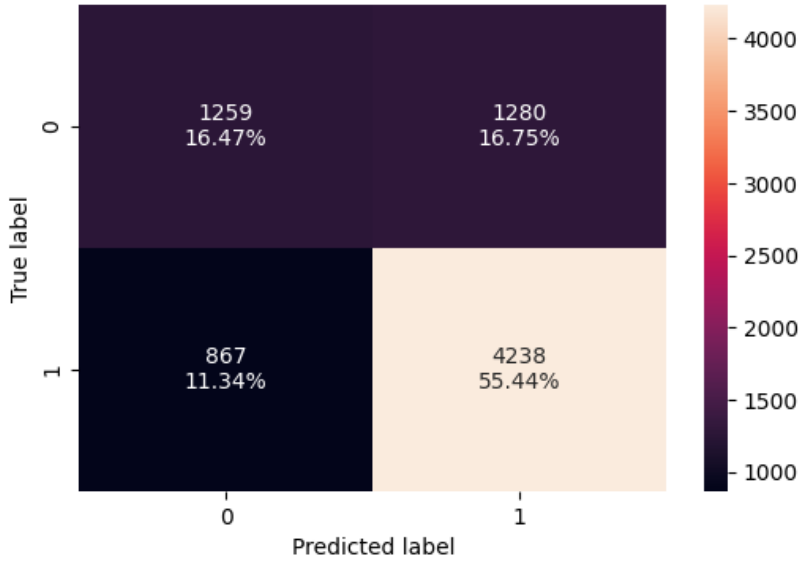

Figure 22: Random Forest Classifier Testing Set Confusion Matrix

Training performance			
Accuracy	Recall	Precision	f1
0.999944	0.999916	1	0.999958
Testing performance			
0.719126	0.830167	0.768032	0.797891

Table 4: Random Forest Classifier Performance

- Random Forest has performed well in terms of accuracy and precision, but it is not able to generalize well on the test data in terms of recall.

Random forest with class weights

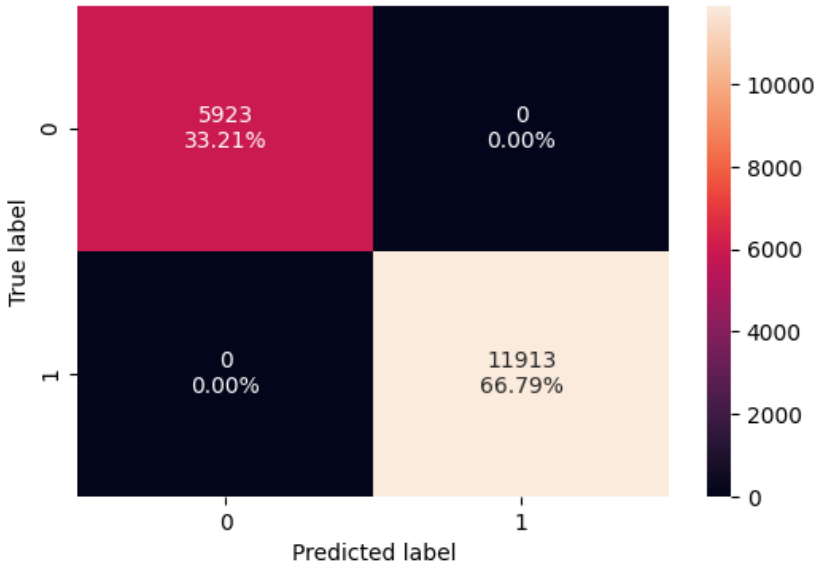

Figure 23: Weighted Random Forest Classification Training Set Confusion Matrix

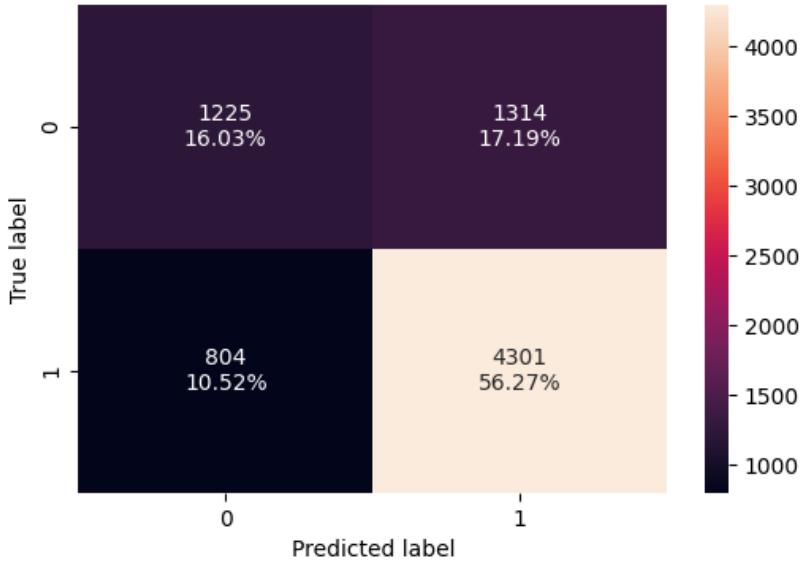

Figure 24: Weighted Random Forest Classification Testing Set Confusion Matrix

Training performance			
Accuracy	Recall	Precision	f1
1	1	1	1
Testing performance			
0.72292	0.842507	0.765984	0.802425

Figure 25: Weighted Random Forest Classification Performance

- There is not much improvement in metrics of weighted random forest as compared to the unweighted random forest. Model is over fitting.

7.4. Boosting Technique

7.4.1. AdaBoost Classification

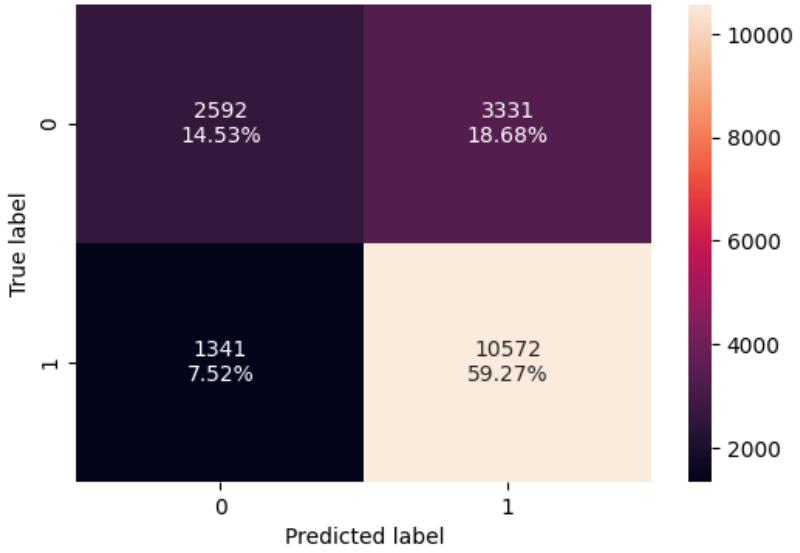

Figure 26: AdaBoosting training set Confusion Matrix

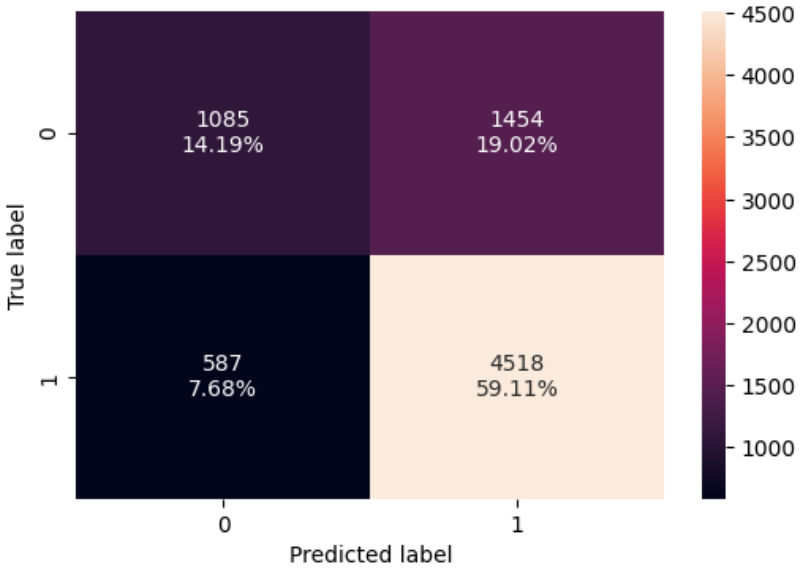

Figure 27: AdaBoosting testing set Confusion Matrix

Training performance			
Accuracy	Recall	Precision	f1
0.738058	0.887434	0.760411	0.819027
Testing performance			
0.732993	0.885015	0.75653	0.815744

Table 5: AdaBoosting Performance

- AdaBoost is generalizing well but it is giving poor performance, in terms of accuracy and precision.

Random forest with class weights

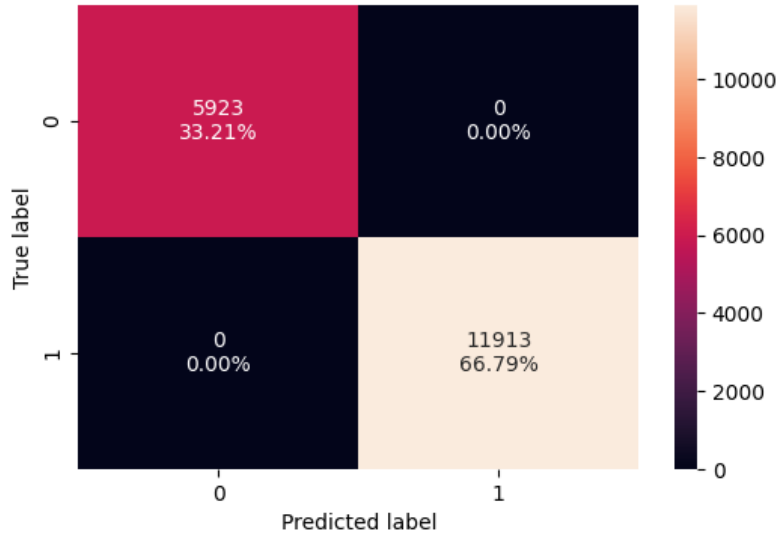

Figure 28: AdaBoosting with Class Weight Training Set Confusion Metrix

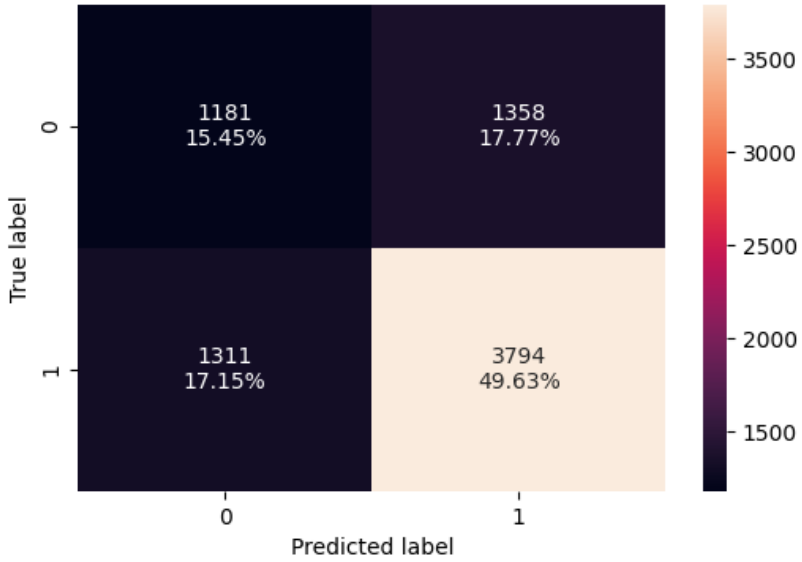

Figure 29: AdaBoosting with Class Weight Testing Set Confusion Metrix

Training performance			
Accuracy	Recall	Precision	f1
1	1	1	1
Testing performance			
0.650837	0.743193	0.736413	0.739787

Table 6: AdaBoosting with Class Weight Performance

- With class weight it is giving poor performance. F1 score has dropped. Also, model is overfitting.

7.4.2. Gradient boosting

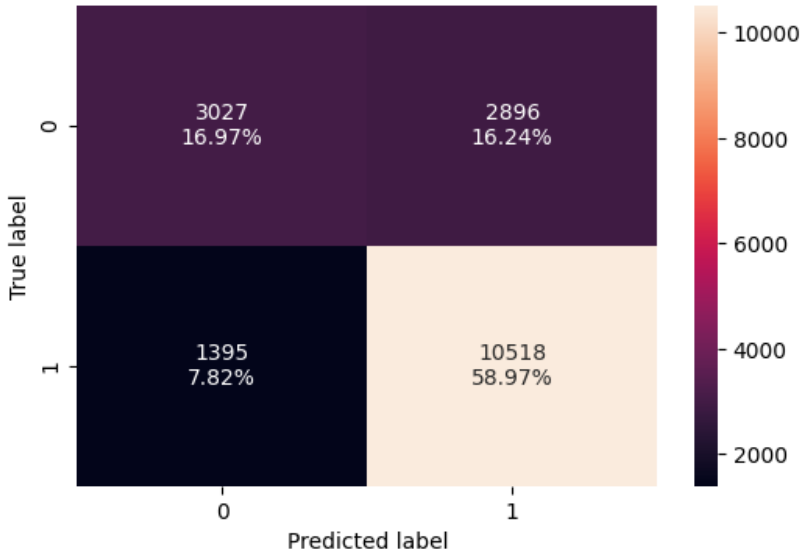

Figure 30: Gradient Boosting Training Set Confusion Metrix

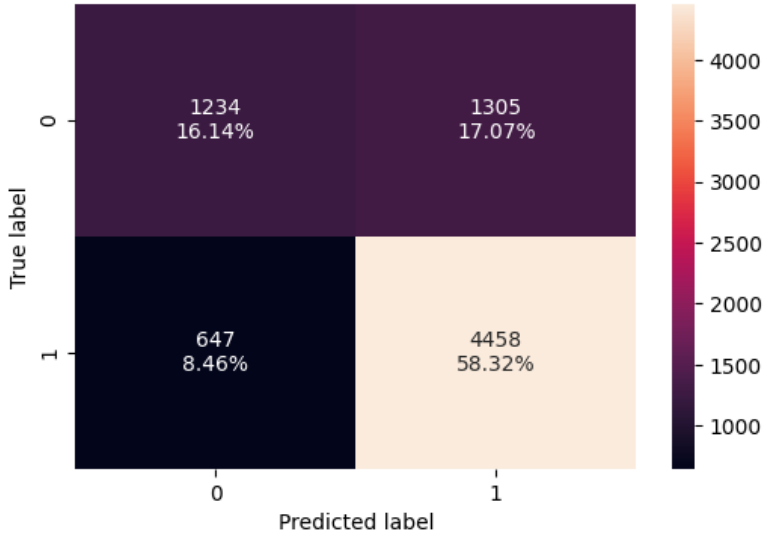

Figure 31: Gradient Boosting Testing Set Confusion Metrix

Training performance			
Accuracy	Recall	Precision	f1
0.985647	0.987325	0.991152	0.989235
Testing performance			
0.69584	0.779432	0.768443	0.773899

Table 7: Gradient Boosting Performance

- Gradient Boost is generalizing well but it is giving poor performance, in terms of accuracy and precision.

7.4.3. XgBoosting Classifier

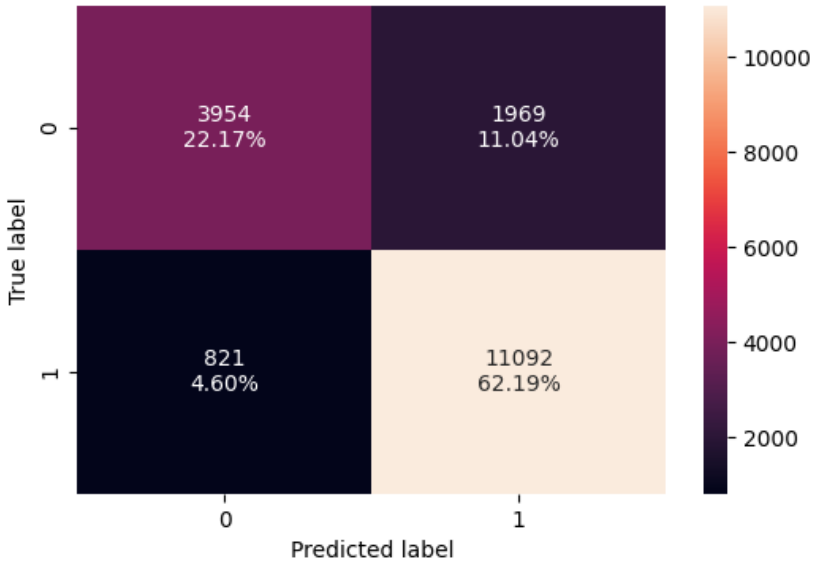

Figure 32: XgBoosting Classifier Training Set Confusion Matrix

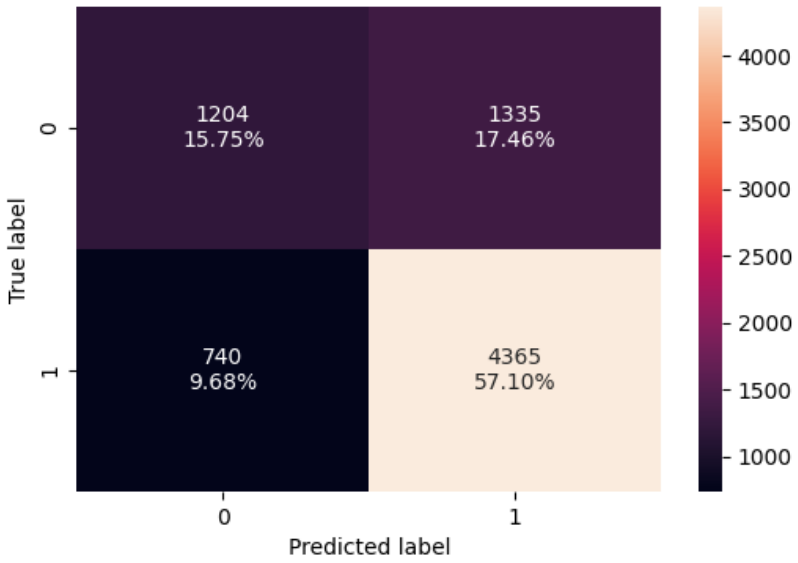

Figure 33: XgBoosting Testing set Confusion Matrix

Training performance			
Accuracy	Recall	Precision	f1
0.843575	0.931084	0.849246	0.888284
Testing performance			
0.728545	0.855044	0.765789	0.807959

Table 8: XgBoosting Model Performance

- AdaBoost classifier has better test accuracy among these 3 models.
- GB classifier has least test accuracy and test recall.
- The model has good f1 score and able to generalize will. However, the precision and acuracy level is low.

8. Model Tuning

Using GridSearch for Hyperparameter tuning model

- Hyperparameter tuning is also tricky in the sense that there is no direct way to calculate how a change in the hyperparameter value will reduce the loss of your model, so we usually resort to experimentation. i.e we'll use Grid search.
- Grid search is a tuning technique that attempts to compute the optimum values of hyperparameters.
- It is an exhaustive search that is performed on the specific parameter values of a model.
- The parameters of the estimator/model used to apply these methods are optimized by cross-validated grid-search over a parameter grid.

8.1. Tuning Decision Tree

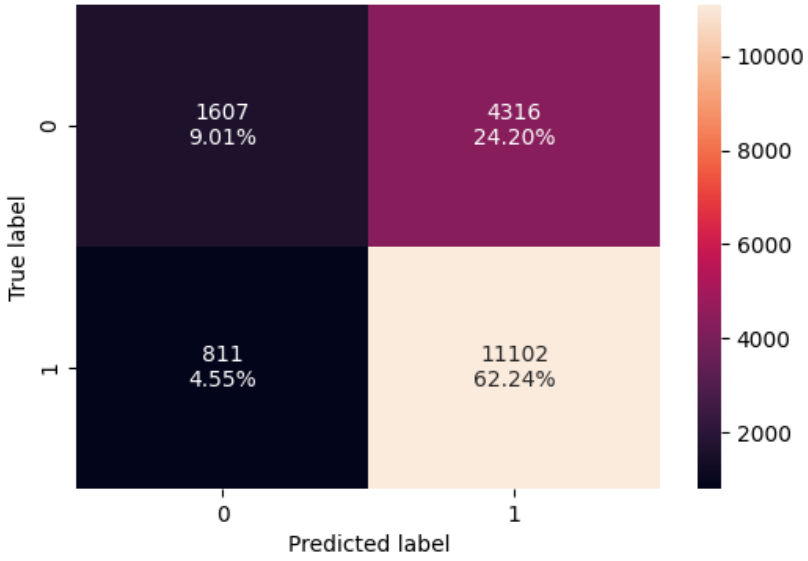

Figure 34: Tuned Decision Tree Training Set Confusion Matrix

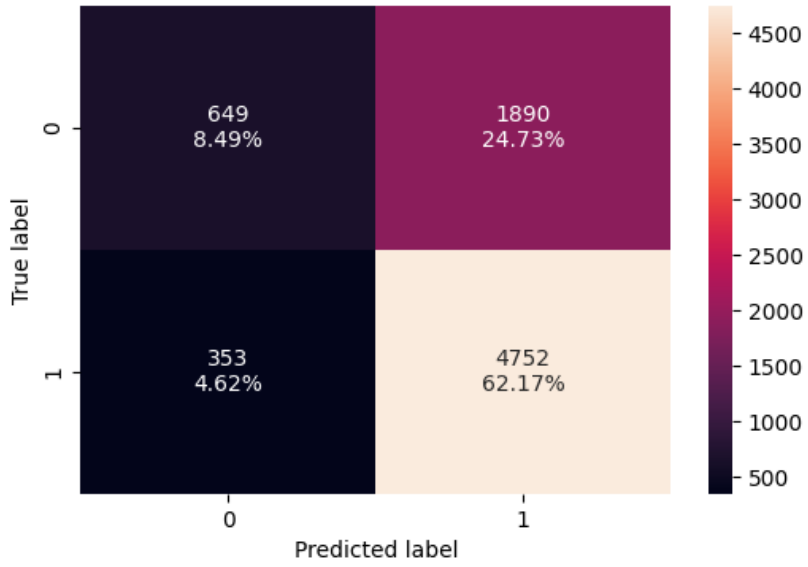

Figure 35: Tuned Decision Tree Testing Set Confusion Matrix

Training performance			
Accuracy	Recall	Precision	f1
0.712548	0.931923	0.720067	0.812411
Testing performance			
0.706567	0.930852	0.715447	0.809058

Table 9: Tuned Decision Tree model performance

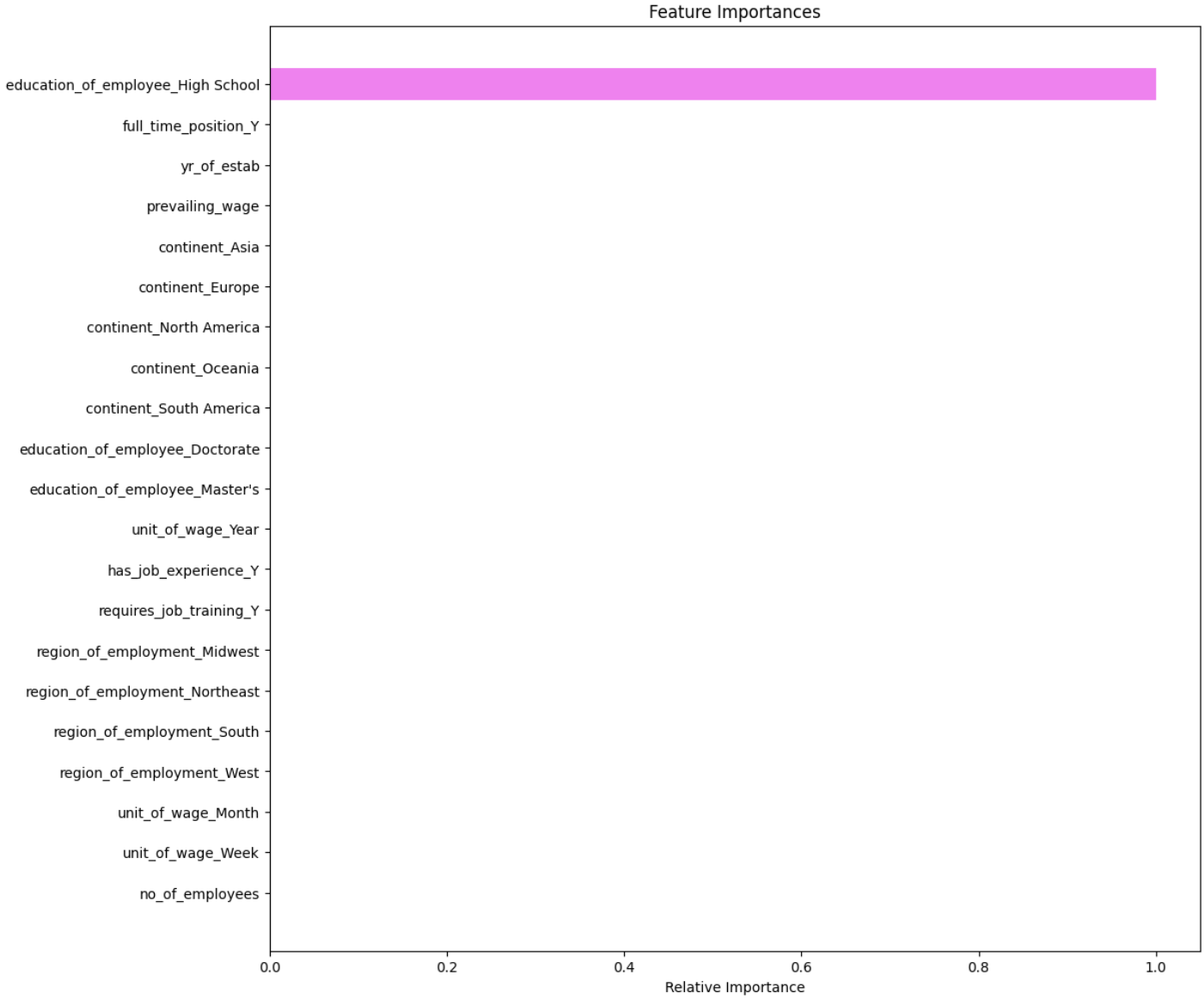

Figure 36: Important features of Tuned Decision Tree Model

- Overfitting in decision tree has reduced and the recall score also significantly good. Overall model is able to generalize well. However, accuracy and precision are still low.
- Model is only considering education of the employee. Hence, the performance is not good.

8.2. Tuning Bagging Classifier

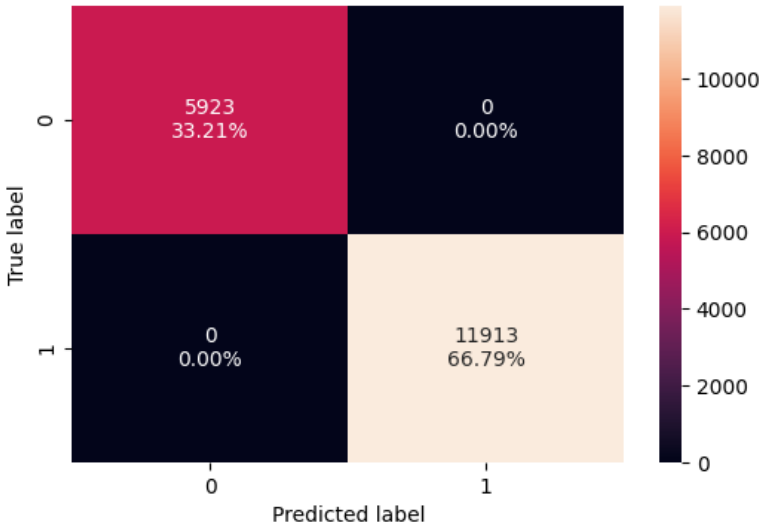

Figure 37: Tuned Bagging Classification Training Set Confusion Matrix

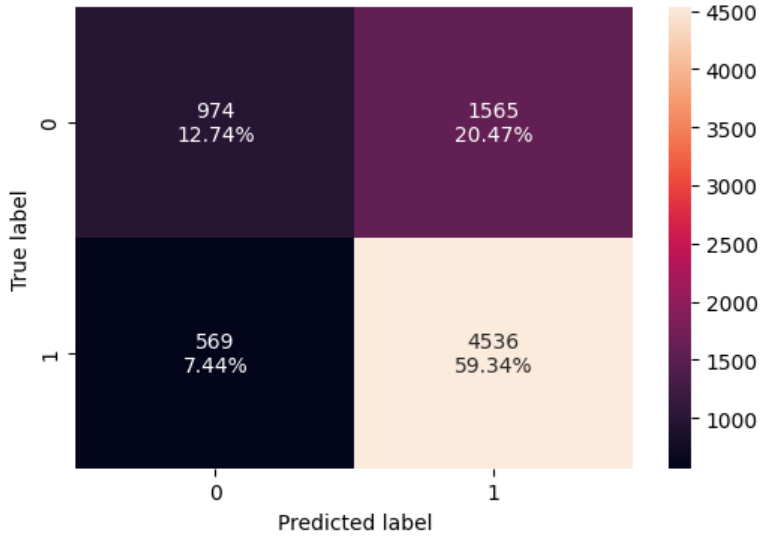

Figure 38: Tuned Bagging Classification Testing Set Confusion Matrix

Training performance			
Accuracy	Recall	Precision	f1
1	1	1	1
Testing performance			
0.720827	0.888541	0.743485	0.809566

Table 10: Tuned Bagging Classification performance

- Recall and accuracy has improved but the precision of the model has dropped and also, it's an over fit model. Overall, the model is making many mistakes.

8.3. Tuning Random Forest

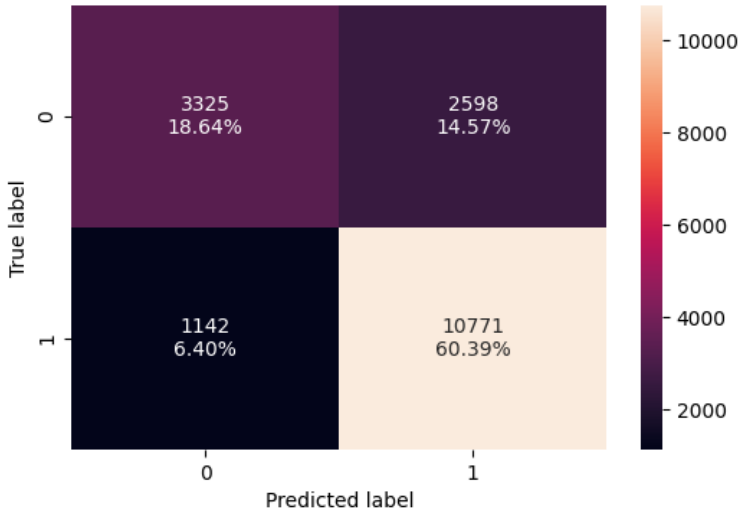

Figure 39:Tuned Random Forest Classification Training Set Confusion Matrix

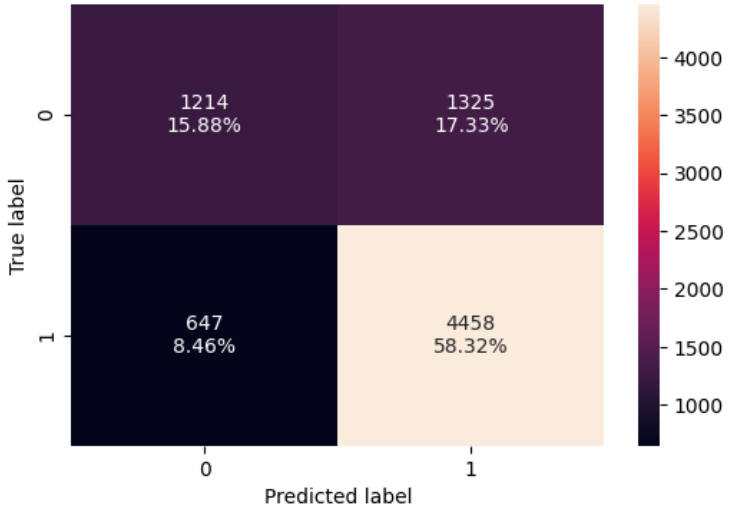

Figure 40: Tuned Random Forest Classification Testing Set Confusion Matrix

Training performance			
Accuracy	Recall	Precision	f1
0.790312	0.904138	0.80567	0.852069
Testing performance			
0.74202	0.873262	0.77088	0.818883

Table 11: Tuned Random Forest Performance

- Random forest after tuning has given better performance as un-tuned random forest. It has good f1 and recall score. However, precision and accuracy are still low.

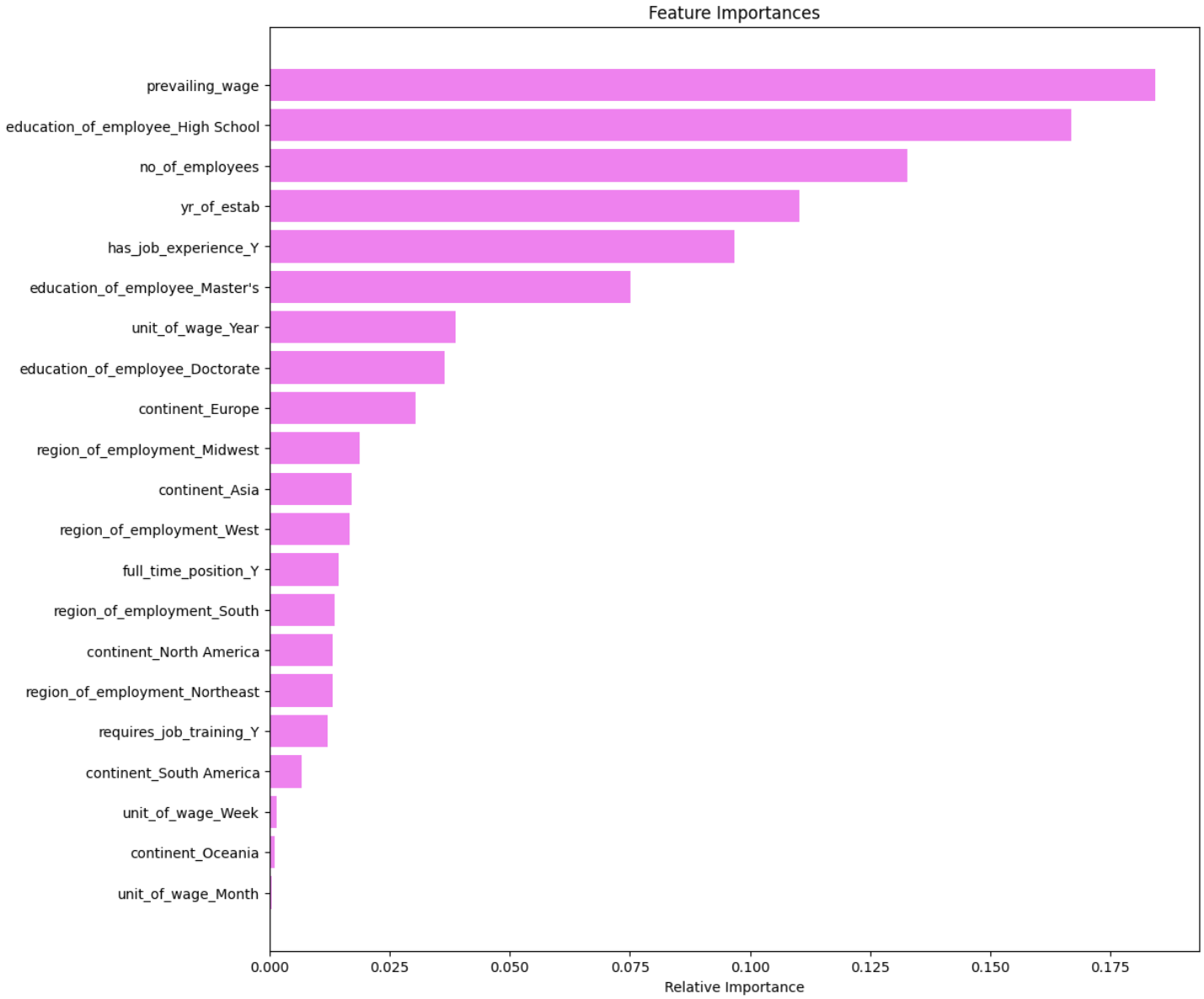

Figure 41: Important features of Tuned Random Forest Model

- Model has considered the Prevailing wage and high school education as most important features followed by no of employees, year of establishment and job experience.

8.4. Tuning Boosting

- We are going to build 3 ensemble models here - AdaBoost Classifier, Gradient Boosting Classifier and XGBoost Classifier.
- First, let's build these models with default parameters and then use hyperparameter tuning to optimize the model performance.
- We will calculate all three metrics - Accuracy, Precision and Recall but the metric of interest here is recall.
- Recall - It gives the ratio of True positives to Actual positives, so high Recall implies low false negatives, i.e. low chances of predicting a defaulter as non-defaulter.

8.4.1. Tuning Adaboosting

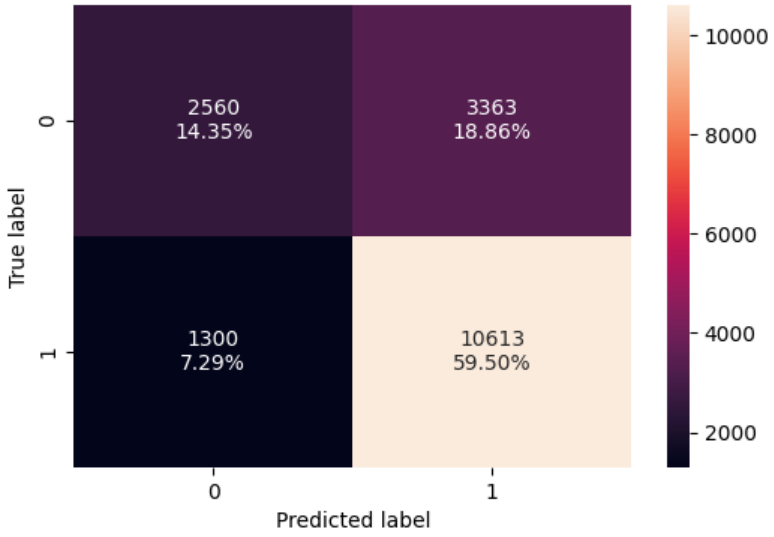

Figure 42: Tuned AdaBoosting training set Confusion Metrix

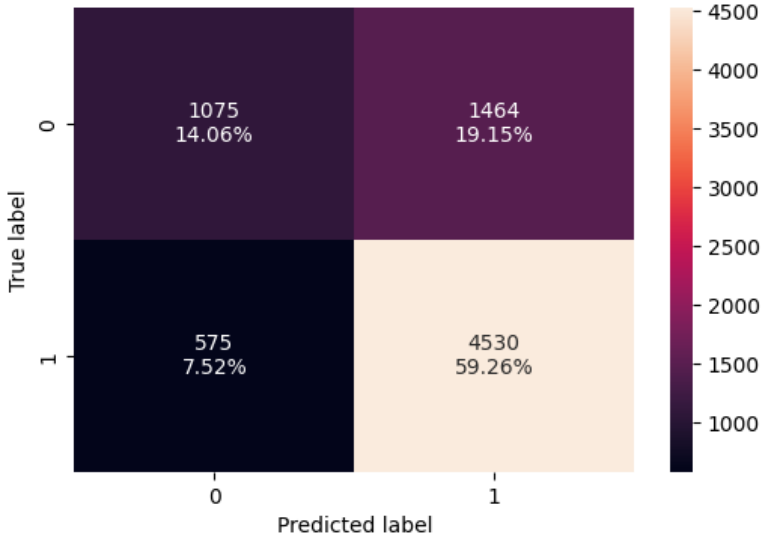

Figure 43: Tuned AdaBoosting testing set Confusion Metrix

Training performance			
Accuracy	Recall	Precision	f1
0.738562	0.890876	0.759373	0.819885
Testing performance			
0.733255	0.887365	0.755756	0.81629

Table 12: Tuned AdaBoosting Performance

- Model has generalized very well with high recall and f1 score.

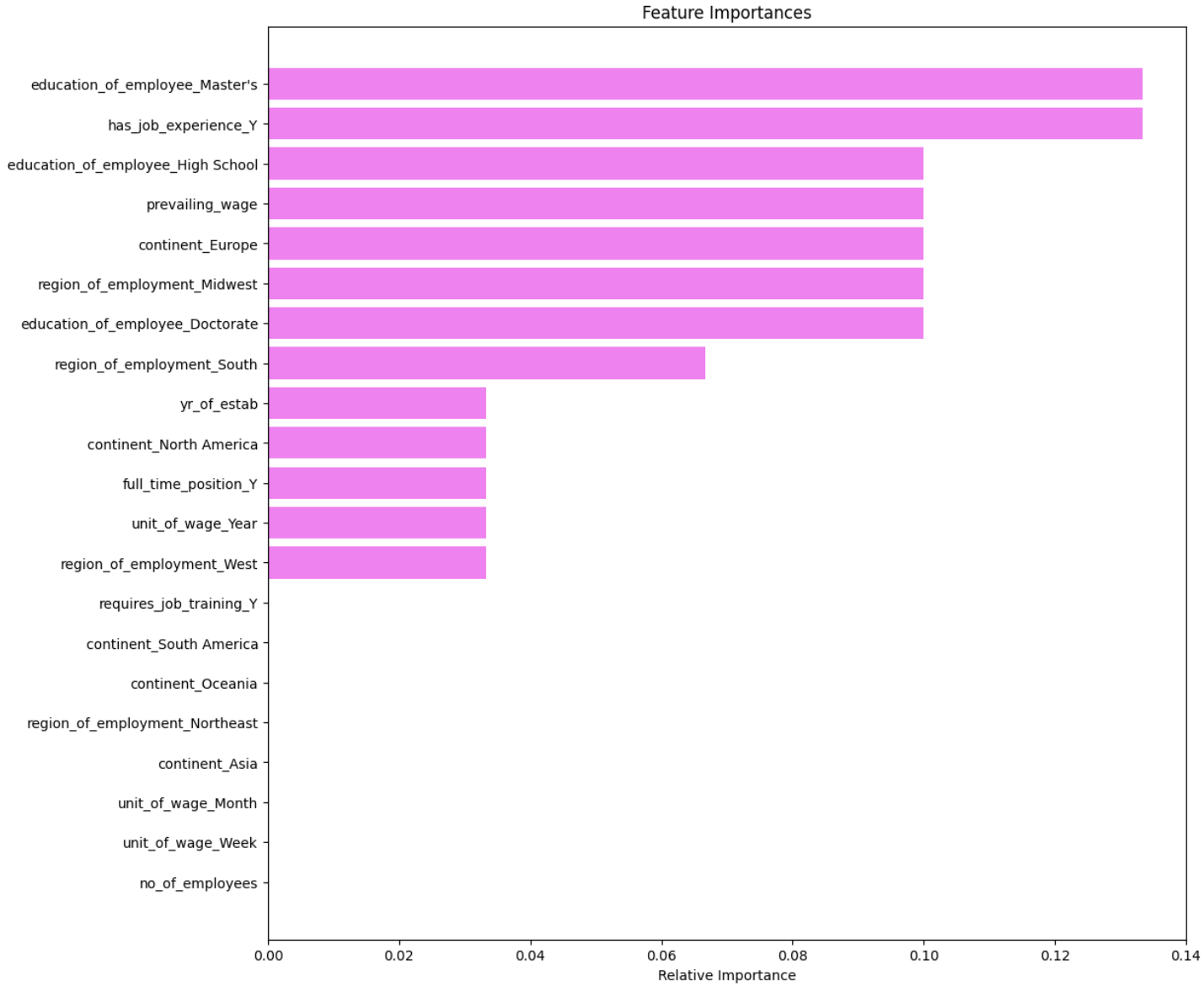

Figure 44: Tuned AdaBoosting Important Features

- Model has given more weightage to the Job experience and master's education, followed by education level at high school, prevailing wages, continent and education at doctorate level.

8.4.2. Tuning Gradient Boosting

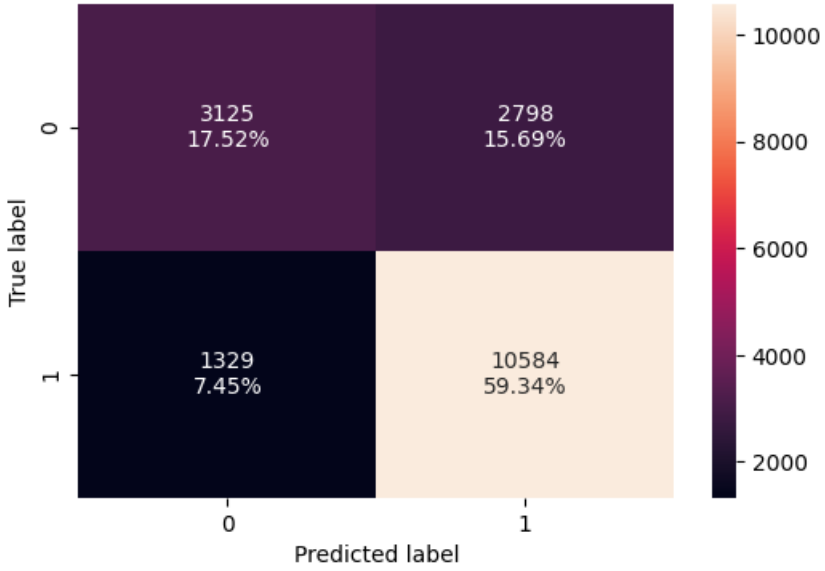

Figure 45: Tuned Gradient Boosting Training Set Confusion Metrix

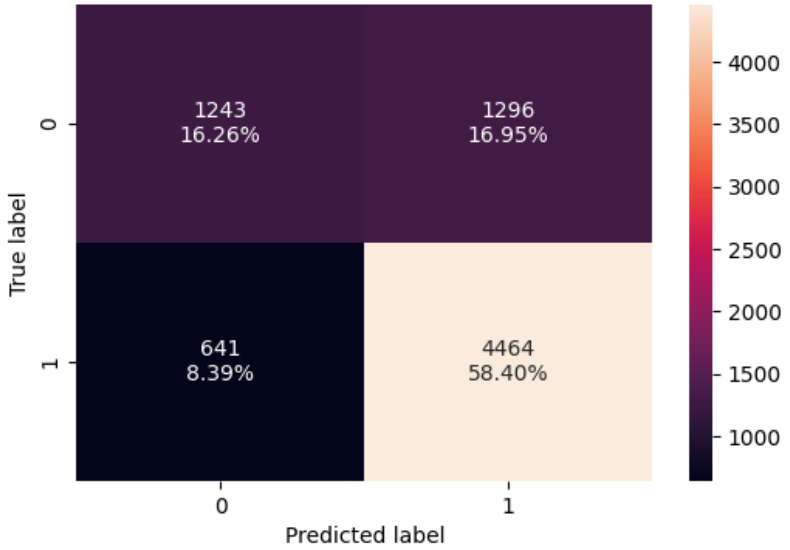

Figure 46: Tuned Gradient Boosting Testing Set Confusion Metrix

Training performance			
Accuracy	Recall	Precision	f1
0.768614	0.888441	0.790913	0.836845
Testing performance			
0.746599	0.874437	0.775	0.821721

Table 13: Tunned Gradient Boosting Performance

- It has a good F1 and recall score. Model is generalizing well.

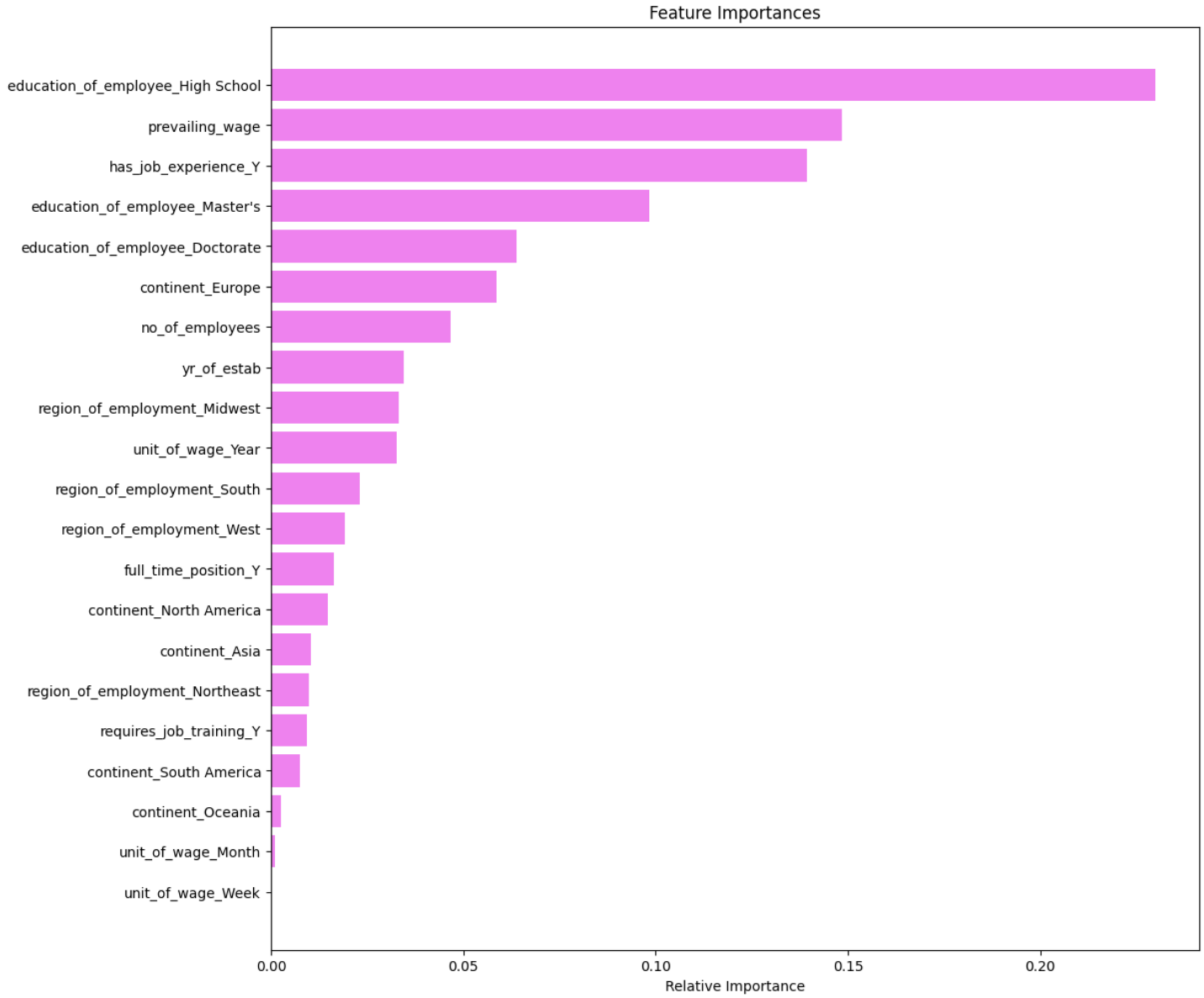

Figure 47: Tuned Gradient Boosting Important Features

- Model has given more weightage to the high school education, followed by prevailing wage and job experience.

8.4.3. Tuning XGBoosting

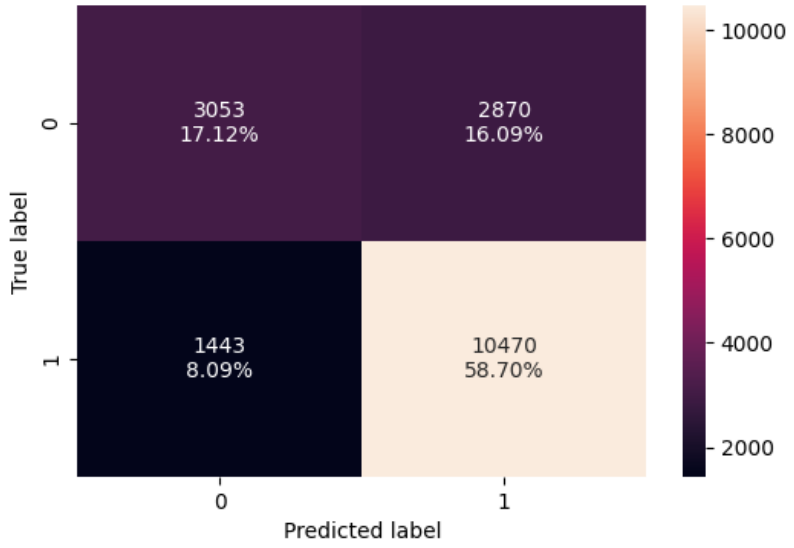

Figure 48: Tunned XGBoosting Training Set Confusion Metrix

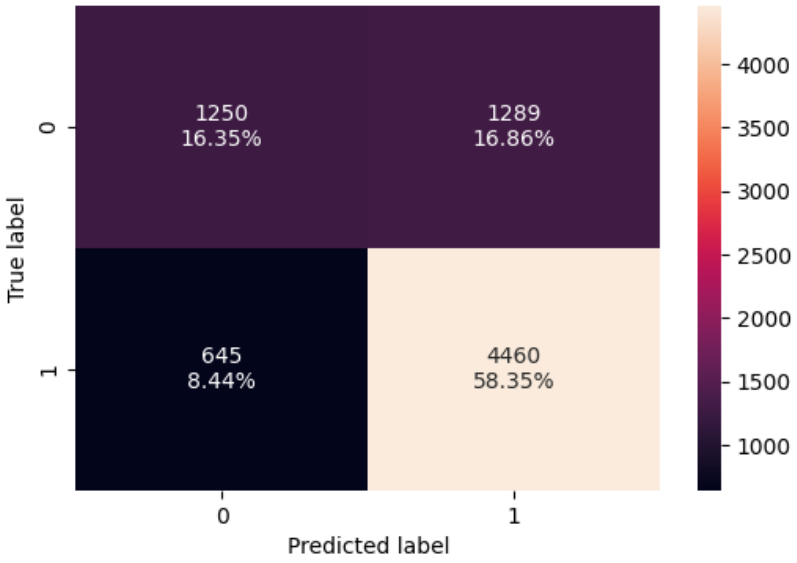

Figure 49: Tunned XGBoosting Testing Set Confusion Metrix

Training performance			
Accuracy	Recall	Precision	f1
0.758186	0.878872	0.784858	0.829208
Testing performance			
0.746991	0.873653	0.775787	0.821817

Table 14: Tunned XGBoosting Model Performance

- The model has performed very well. It has good f1 and recall score.

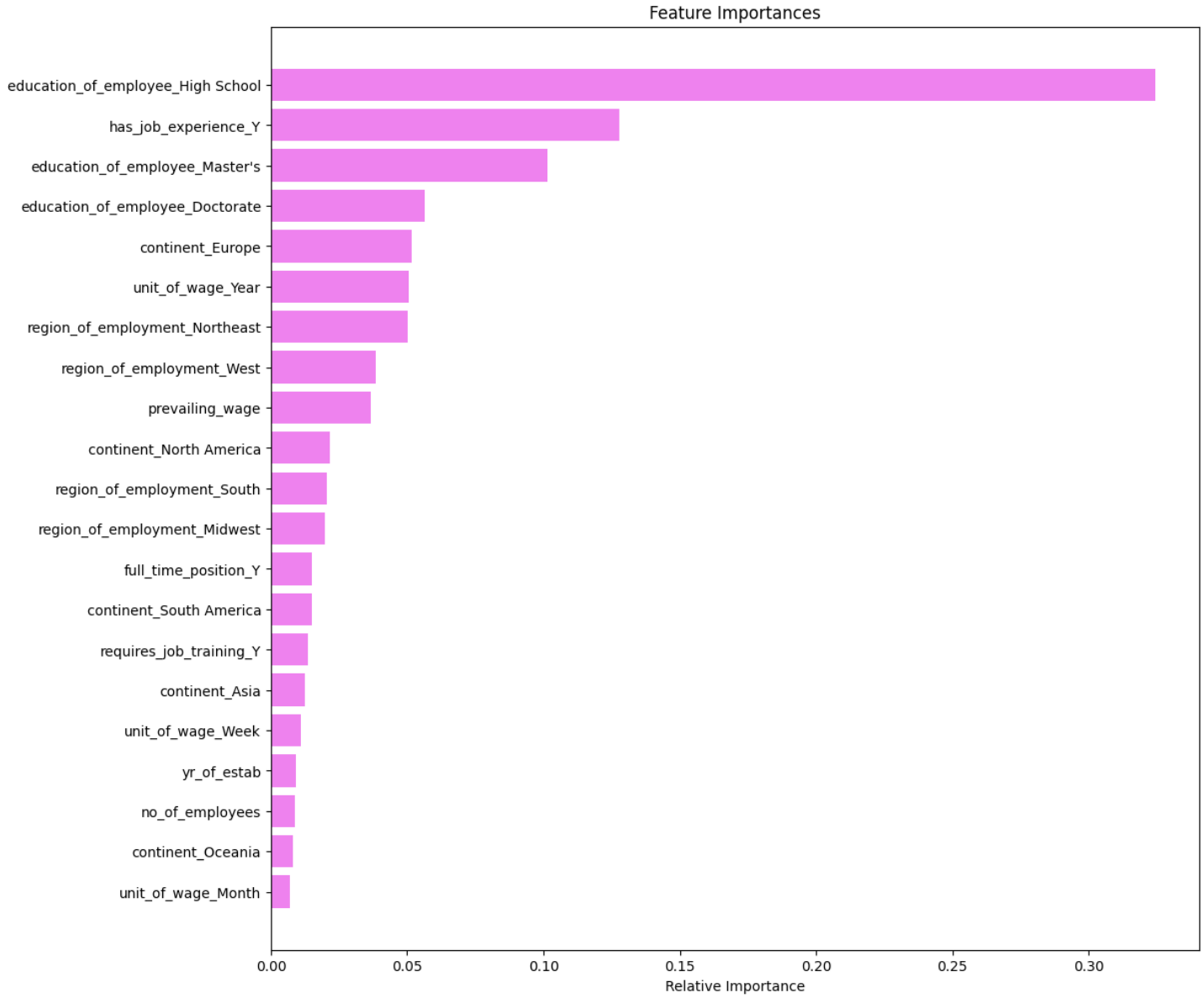

Figure 50: Tuned XGBoosting Model Important Features

- Model has given more importance to the high school level education followed by the job experience and master's education.

8.4.4. Stacking Model

Now, let's build a stacking model with the tuned models - decision tree, random forest, and gradient boosting, then use XGBoost to get the final prediction.

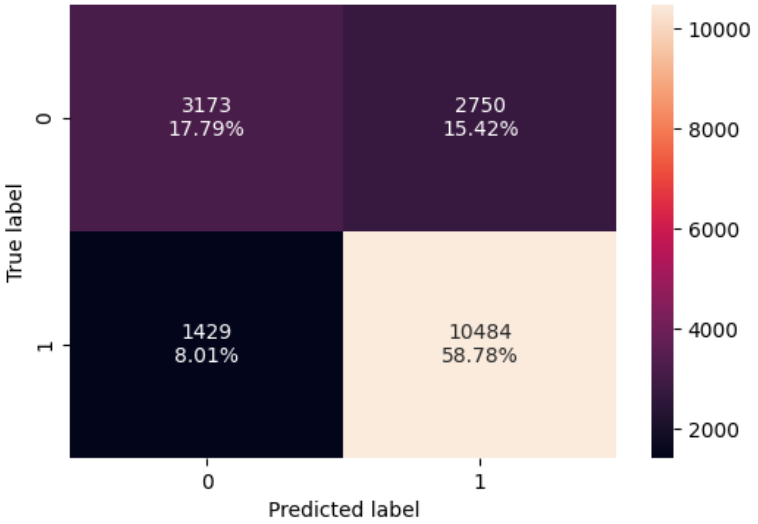

Figure 51: Stacking Training Set Confusion Metrix

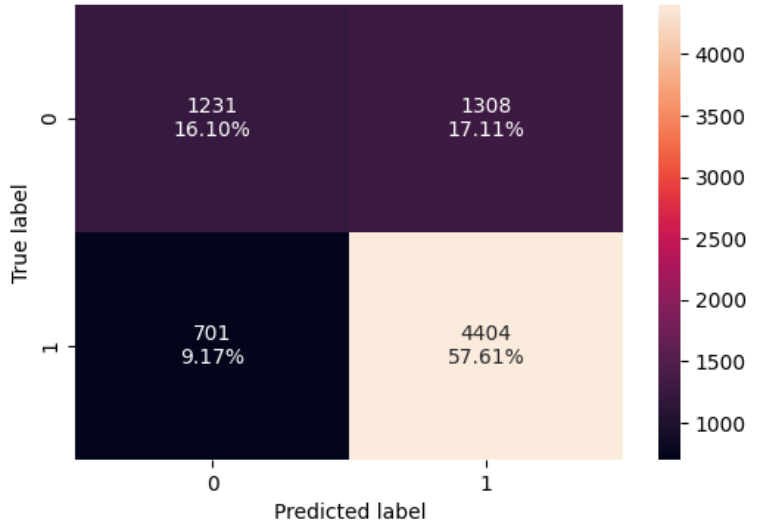

Figure 52: Stacking Testing Set Confusion Metrix

Training performance			
Accuracy	Recall	Precision	f1
0.765699	0.880047	0.792202	0.833817
Testing performance			
0.737179	0.862684	0.771008	0.814274

Table 15: Stacking Performance

- Overall, the performance has decreased than tuned XGBoosting.

9. Model Comparison

Training performance																
	Decision Tree	Bagging Classifier	Weighted Bagging Classifier	Random Forest Classifier	Weighted Random Forest Classifier	AdaBoosting Classifier	Weighted AdaBoosting Classifier	Gradient Boosting Classifier	XG Boosting Classifier	Decision Tree Estimator	Bagging Estimator	Random Forest Estimator	AdaBoosting Estimator	Gradient Boosting Estimator	XG Boosting Estimator	XGBoosting Stacking Estimator
Accuracy	1	0.985367	0.985647	0.999944	1	0.738058	1	0.759419	0.843575	0.712548	1	0.790312	0.738562	0.768614	0.758186	0.765699
Recall	1	0.986317	0.987325	0.999916	1	0.887434	1	0.882901	0.931084	0.931923	1	0.904138	0.890876	0.888441	0.878872	0.880047
Precision	1	0.991729	0.991152	1	1	0.760411	1	0.784106	0.849246	0.720067	1	0.80567	0.759373	0.790913	0.784858	0.792202
F1	1	0.989016	0.989235	0.999958	1	0.819027	1	0.830576	0.888284	0.812411	1	0.852069	0.819885	0.836845	0.829208	0.833817
Testing performance																
	Decision Tree	Bagging Classifier	Weighted Bagging Classifier	Random Forest Classifier	Weighted Random Forest Classifier	AdaBoosting Classifier	Weighted AdaBoosting Classifier	Gradient Boosting Classifier	XG Boosting Classifier	Decision Tree Estimator	Bagging Estimator	Random Forest Estimator	AdaBoosting Estimator	Gradient Boosting Estimator	XG Boosting Estimator	XGBoosting Stacking Estimator
Accuracy	0.656463	0.692177	0.69584	0.719126	0.72292	0.732993	0.650837	0.744636	0.728545	0.706567	0.720827	0.74202	0.733255	0.746599	0.746991	0.737179
Recall	0.750637	0.766112	0.779432	0.830167	0.842507	0.885015	0.743193	0.873262	0.855044	0.930852	0.888541	0.873262	0.887365	0.874437	0.873653	0.862684
Precision	0.739055	0.7714	0.768443	0.768032	0.765984	0.75653	0.736413	0.773555	0.765789	0.715447	0.743485	0.77088	0.755756	0.775	0.775787	0.771008
F1	0.744801	0.768747	0.773899	0.797891	0.802425	0.815744	0.739787	0.82039	0.807959	0.809058	0.809566	0.818883	0.81629	0.821721	0.821817	0.814274

Table 16: Models Performance Comparison

- Gradient boosting classifier/ estimator and XG Boosting Estimator has good f1 score.
- Tuned XGBoosting has better f1 score and overall performing well.
- As the final results depend on the parameters used/checked using GridSearchCV, there may be yet better parameters which may result in a better performance.

10. Model Tuning

10.1. Data Preprocessing

10.1.1. Feature Engineering

- We have assigned the numerical values to each categorical column.
- Job training needed / experience / full time position 'Y' and 'N' has been assigned 1 and 0 respectively.
- Wages unit has been assigned 0,1,2, and 3 respectively for Hour, Week, Month, Year.
- Region of employment has been assigned 0,1,2,3, and 4 respectively for 'West', "Northeast', 'South', 'Midwest', and 'Island'.
- Education of the employee has been assigned 0,1,2, and 3 respectively for High School, Batchelor's, Master's, and Doctorate.

10.1.2. Data Preparation for Modeling

- We have divided our data into three sets training (15288 rows), validation (5096 rows), and test (5096 rows).
- This partition will help in preventing data leakage in the model.
- We don’t have missing values and duplicate records.
- We have done the inverse mapping of our training, testing and validation data.
- We have dropped the case id column as it doesn’t serve any purpose.

10.2. Initial Model Building

10.2.1. Model Building - Original Data

Recall score:

Training and Validation Performance Difference:

Bagging: Training Score: 0.9888, Validation Score: 0.7832, Difference: 0.2056

Random forest: Training Score: 1.0000, Validation Score: 0.8408, Difference: 0.1592

GBM: Training Score: 0.8796, Validation Score: 0.8743, Difference: 0.0054

Adaboost: Training Score: 0.8878, Validation Score: 0.8822, Difference: 0.0056

dtree: Training Score: 1.0000, Validation Score: 0.7450, Difference: 0.2550

- **Lowest difference observed in gradient boosting and ada boosting method.**

10.2.2. Model Building - Oversampled Data

Recall score:

Training and Validation Performance Difference:

Bagging: Training Score: 0.9828, Validation Score: 0.7512, Difference: 0.2316

Random forest: Training Score: 1.0000, Validation Score: 0.8132, Difference: 0.1868

GBM: Training Score: 0.8514, Validation Score: 0.8449, Difference: 0.0065

Adaboost: Training Score: 0.8446, Validation Score: 0.8425, Difference: 0.0020

dtree: Training Score: 1.0000, Validation Score: 0.7180, Difference: 0.2820

- **Lowest difference observed in gradient boosting and ada boosting method.**

10.2.3. Model Building – Under sampled Data

Training and Validation Performance Difference:

Bagging: Training Score: 0.9687, Validation Score: 0.6178, Difference: 0.3509

Random forest: Training Score: 1.0000, Validation Score: 0.6742, Difference: 0.3258

GBM: Training Score: 0.7483, Validation Score: 0.7321, Difference: 0.0162

Adaboost: Training Score: 0.7176, Validation Score: 0.7074, Difference: 0.0102

dtree: Training Score: 1.0000, Validation Score: 0.6313, Difference: 0.3687

- **Lowest difference observed in gradient boosting and ada boosting method.**

We are going to train our the GBM and Adaboost models on an undersampled dataset, as well as the GBM model trained on an oversampled dataset, followed by tune them to understand how it performs.

10.3. Hyperparameter Tuning

10.3.1. Tuning AdaBoost Classifier model with Under sampled data

Training performance			
Accuracy	Recall	Precision	f1
0.703525	0.7436	0.688423	0.714948
Validation performance			
0.71978	0.740599	0.822244	0.779289

Table 17: Tuned AdaBoost Model Performance on Under Sample Data

10.3.2. Tuning Gradient Boosting model with Under sampled Data

Training performance			
Accuracy	Recall	Precision	f1
0.705396	0.757385	0.686051	0.719955
Validation performance			
0.718014	0.743831	0.817565	0.778957

Table 18: Tuned Gradient Boost Model Performance on Under Sample Data

10.3.3. Tuning Gradient Boosting model with Oversampled data

Training performance			
Accuracy	Recall	Precision	f1
0.696817	0.753379	0.676815	0.713048
Validation performance			
0.718014	0.743831	0.817565	0.778957

Table 19: Tuned Gradient Boost Model Performance on Over Sample Data

10.4. Model Comparison and Final Model Selection

	Gradient boosting trained with Under sampled data	Gradient boosting trained with Oversampled data	AdaBoost trained with Under sampled data
Training performance comparison:			
Accuracy	0.705396	0.696817	0.703525
Recall	0.757385	0.753379	0.743600
Precision	0.686051	0.676815	0.688423
F1	0.719955	0.713048	0.714948
Validation performance comparison:			
Accuracy	0.718014	0.718014	0.719780
Recall	0.743831	0.743831	0.740599
Precision	0.817565	0.817565	0.822244
F1	0.778957	0.778957	0.779289

Table 20: Overall comparison of all the tuned models

- All the tuned models are performing well on validation data set. However, Adaboosting shows best f1 and recall score.
- Now we can actually compare how our model is performing on test data.

Testing AdaBoosing performance			
Accuracy	Recall	Precision	F1
0.714089	0.75235	0.806614	0.778538

Table 21: Tuned AdaBoosing performance on Testing Data

- The Adaboost model trained on undersampled data has given ~75% recall on the test set.
- This performance is in line with what we achieved with this model on the train and validation sets.
- So, this is a generalized model.

10.5. Feature Importance

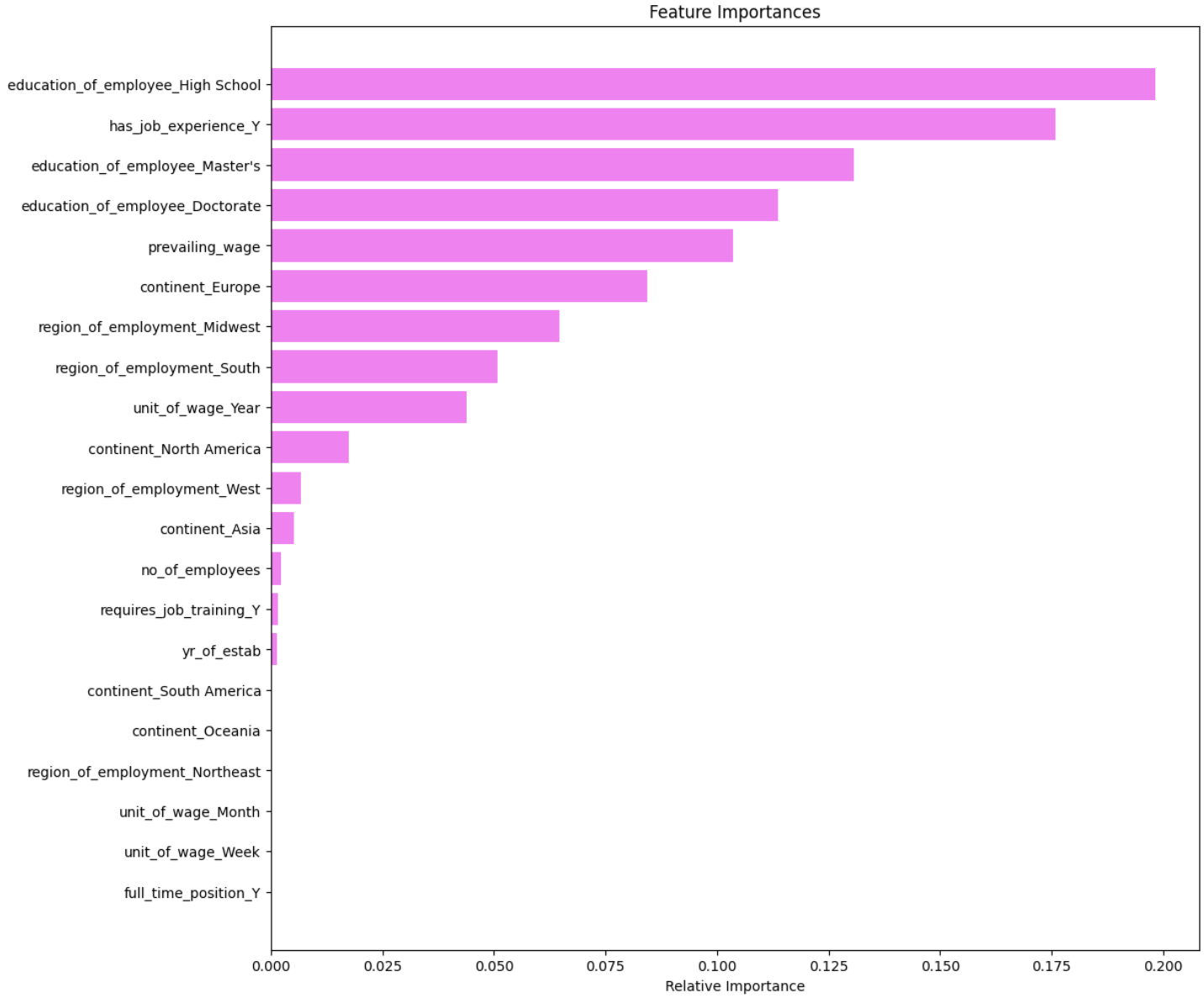

Figure 53: Tuned AdaBoosing important features

- The most important features are education of the employee and job experience.

11. Actionable Insights & Recommendations

1. Develop an initial filtering system that prioritizes candidates with a high school education in the early stages of the review process. This will help reduce processing time for these cases, leading to a more efficient workflow.
2. Include job experience as a weighted feature in the machine learning model to prioritize applicants with substantial work history. Employers should be encouraged to emphasize the relevant experience of foreign workers to increase their chances of certification.
3. Create additional ranking tiers within the model based on educational qualifications, giving higher preference to candidates with advanced degrees. This could serve as a secondary filter for applicants after job experience and high school education.

4. Implement wage thresholds in the model to ensure that candidates with salaries meeting or exceeding these levels are ranked higher. Companies should be made aware of the importance of offering competitive wages to foreign workers, as this significantly improves their visa approval chances.

5. Consider regional data as a feature in the machine learning model. Emphasize outreach and talent acquisition efforts in European countries and US regions where visa certification is more likely to succeed.

6. Avoid overfitting the model by down weighting less impactful variables. Focus resources and automation efforts on the key drivers (education, experience, wages, and region) to ensure a streamlined and accurate certification process.