ML-2 CODED PROJECT

Business Report

DSBA

Submitted By: Maheep Singh
Batch : PGP-DSBA (PGPDSBA.O.AUG24.A)

Table of Contents

List of Figures

List of Tables

Business Context & Data Dictionary

Context

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.

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:

1. Facilitate the process of visa approvals.
2. 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.

Data Description

The data contains the different attributes of the employee and the employer. The detailed data dictionary is given below: -

- 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

Rubric Question 1: Exploratory Data Analysis

Data Overview

- **Load dataset** & display top 5 rows (Transpose view due to high no. of columns): -

	0	1	2	3	4
case_id	EZYV01	EZYV02	EZYV03	EZYV04	EZYV05
continent	Asia	Asia	Asia	Asia	Africa
education_of_employee	High School	Master's	Bachelor's	Bachelor's	Master's
has_job_experience	N	Y	N	N	Y
requires_job_training	N	N	Y	N	N
no_of_employees	14513	2412	44444	98	1082
yr_of_estab	2007	2002	2008	1897	2005
region_of_employment	West	Northeast	West	West	South
prevailing_wage	592.203	83425.650	122996.860	83434.030	149907.390
unit_of_wage	Hour	Year	Year	Year	Year
full_time_position	Y	Y	Y	Y	Y
case_status	Denied	Certified	Denied	Denied	Certified
company_age	0-10	10-20	0-10	50+	10-20
hourly_wage	592.203	40.263	59.361	40.267	72.349
company_size	4000+	2000-3000	4000+	0-1000	1000-2000

Figure 1: Top 5 rows of the dataset

- There are **25,480 rows & 12 columns** in the dataset
- **Checking datatypes**: -

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25480 entries, 0 to 25479
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   case_id                25480 non-null  object 
 1   continent              25480 non-null  object 
 2   education_of_employee  25480 non-null  object 
 3   has_job_experience     25480 non-null  object 
 4   requires_job_training  25480 non-null  object 
 5   no_of_employees        25480 non-null  int64  
 6   yr_of_estab            25480 non-null  int64  
 7   region_of_employment   25480 non-null  object 
 8   prevailing_wage        25480 non-null  float64
 9   unit_of_wage           25480 non-null  object 
 10  full_time_position     25480 non-null  object 
 11  case_status            25480 non-null  object 
dtypes: float64(1), int64(2), object(9)
```

Figure 2: Datatypes in the Dataset

  - There are 12 columns - 9 object (category) type, 1 float64 (numeric) & 2 int64 (numeric) datatypes in the dataset.

- **Data Transformation/Feature Engineering**: -
  - **'yr_of_estab'** in itself serves no purpose. Let's derive company age (considering, its 2016 today, as mentioned in the problem) & categorize it in bins. A **new variable 'company_age'** is created for the same: -

```
company_age
10-20    8378
50+      5109
0-10     4856
30-50    3746
20-30    3391
Name: count, dtype: int64
```

Figure 3: Derive Company Age & Categorize in Bins

- **'prevailing_wage'** is calculated based on '**unit_of_wage'**, so it may give wrong inference if we use it as-is.
  - We need to standardize the values by converting all of them to same unit i.e. hourly wages.
  - In order to standardize, we create a **new variable 'hourly_wage'** & calculate hourly wages by assuming the following: -
    - ✓ A year includes 2,072 work-hours
    - ✓ A month includes 171 work-hours
    - ✓ A week includes 38 work-hours
  - Below is the sample of 5 Hourly wages post above treatment: -

	unit_of_wage	hourly_wage
7417	Year	2.918
2307	Year	24.516
10648	Hour	216.482
13771	Year	1.807
13563	Year	58.203

Figure 4: Sample of 5 Hourly-wages post Standardization

- **'no_of_employees'** in a company, as a continuous variable, may not have a direct relevance. It is better to categorize it in bins so as to capture all company sizes. We create a **new variable 'company_size'** for the same.
  - In order to ensure we don't capture any erroneous values (i.e. negative values, as employee count cannot be negative), we ensure all non-negative values are replaced by median.
  - Post negative-value treatment, we categorize it in bins: -

```
Negative Values in Employee Field = 33
Negative Values in Employee Field after Treatment = 0

company_size
0-1000       6208
1000-2000    5915
2000-3000    5011
4000+        4798
3000-4000    3548
```

Figure 5: 'no_of_employees' Negative-value Treatment & Categorization

  - Please note that we only had **33 erroneous values**, we may have also dropped them since they only account for only 0.13% of values. However, we chose to **replace by median** to avoid any loss of information.

- **Check & Treat Missing/Duplicate Values**: -
  - ✓ Upon checking, neither missing nor duplicate values were found. Hence, no treatment required.

```
Missing Values:-

case_id                  0
continent                0
education_of_employee    0
has_job_experience       0
requires_job_training    0
no_of_employees          0
yr_of_estab              0
region_of_employment     0
prevailing_wage          0
unit_of_wage             0
full_time_position       0
case_status              0
company_age              0
hourly_wage              0
company_size             0

  Duplicated Values:  0
```

Figure 6: Missing/Duplicate Value-check

- **Dropping redundant columns & converting all 'object' to 'category' datatype: -**
  - ✓ **case_id**: This is a unique identifier in the table, otherwise, it serves no relevance in terms of analysing data.
  - ✓ **no_of_employees**: New categorical variable 'company_size' has been created to capture this.
  - ✓ **yr_of_estab**: New categorical variable 'company_age' has been created to capture this.
  - ✓ **prevailing_wage**: New numerical variable 'hourly_wage has been created to capture this

```
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   continent              25480 non-null  category
 1   education_of_employee  25480 non-null  category
 2   has_job_experience     25480 non-null  category
 3   requires_job_training  25480 non-null  category
 4   region_of_employment   25480 non-null  category
 5   unit_of_wage           25480 non-null  category
 6   full_time_position     25480 non-null  category
 7   case_status            25480 non-null  category
 8   company_age            25480 non-null  category
 9   hourly_wage            25480 non-null  float64
 10  company_size           25480 non-null  category
dtypes: category(10), float64(1)
```

Figure 7: Final list of Columns (with Datatypes) post Data-Treatment & Feature Engineering

- Upon checking unique values of all categorical variables, **no error/mistypes were found** in the data. Hence, no treatment required.

```
Unique values in continent are :
continent
Asia             16861
Europe            3732
North America     3292
South America      852
Africa             551
Oceania            192
Name: count, dtype: int64
**************************************************
Unique values in education_of_employee are :
education_of_employee
Bachelor's     10234
Master's        9634
High School     3420
Doctorate       2192
Name: count, dtype: int64
**************************************************
Unique values in has_job_experience are :
has_job_experience
Y    14802
N    10678
Name: count, dtype: int64
**************************************************
Unique values in requires_job_training are :
requires_job_training
N    22525
Y     2955
Name: count, dtype: int64
**************************************************
Unique values in region_of_employment are :
region_of_employment
Northeast    7195
South        7017
West         6586
Midwest      4307
Island        375
Name: count, dtype: int64
**************************************************
Unique values in unit_of_wage are :
unit_of_wage
Year     22962
Hour      2157
Week       272
Month       89
Name: count, dtype: int64
**************************************************
Unique values in full_time_position are :
full_time_position
Y    22773
N     2707
Name: count, dtype: int64
**************************************************
Unique values in case_status are :
case_status
Certified    17018
Denied        8462
Name: count, dtype: int64
**************************************************
Unique values in company_age are :
company_age
10-20    8378
50+      5109
0-10     4856
30-50    3746
20-30    3391
Name: count, dtype: int64
**************************************************
Unique values in company_size are :
company_size
0-1000       6208
1000-2000    5915
2000-3000    5011
4000+        4798
3000-4000    3548
Name: count, dtype: int64
**************************************************
```

Figure 8: Error-value check (for Categorical values) in the Dataset

- **Statistical Summary of the dataset**: -

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
continent	25480	6	Asia	16861	NaN	NaN	NaN	NaN	NaN	NaN	NaN
education_of_employee	25480	4	Bachelor's	10234	NaN	NaN	NaN	NaN	NaN	NaN	NaN
has_job_experience	25480	2	Y	14802	NaN	NaN	NaN	NaN	NaN	NaN	NaN
requires_job_training	25480	2	N	22525	NaN	NaN	NaN	NaN	NaN	NaN	NaN
region_of_employment	25480	5	Northeast	7195	NaN	NaN	NaN	NaN	NaN	NaN	NaN
unit_of_wage	25480	4	Year	22962	NaN	NaN	NaN	NaN	NaN	NaN	NaN
full_time_position	25480	2	Y	22773	NaN	NaN	NaN	NaN	NaN	NaN	NaN
case_status	25480	2	Certified	17018	NaN	NaN	NaN	NaN	NaN	NaN	NaN
company_age	25480	5	10-20	8378	NaN	NaN	NaN	NaN	NaN	NaN	NaN
hourly_wage	25480.000	NaN	NaN	NaN	95.060	278.205	0.048	22.736	39.979	60.244	7004.399
company_size	25480	5	0-1000	6208	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figure 9: Statistical Summary of the Dataset

Type	Columns	Observations & Insights
Categorical	continent	✓ Multivariate variable with 6 unique values. ✓ 'Asia' is the most dominant value, implying majority of the applicants are from the Asean region.
Categorical	education_of_employee	✓ Multivariate variable with 4 unique values. ✓ "Bachelor's" is the most dominant value, implying majority of the applicants carry a Bachelor's degree.
Categorical	has_job_experience	✓ Bivariate variable with 2 unique values. ✓ 'Y' is the most dominant value, implying majority of the applicants have prior work-experience.
Categorical	requires_job_training	✓ Bivariate variable with 2 unique values. ✓ 'N' is the most dominant value, implying majority of the applicants don't require job training.
Categorical	region_of_employment	✓ Multivariate variable with 5 unique values. ✓ 'Northeast' is the most dominant value, implying majority of the applicants' preferred region of employment is Northeast.
Categorical	unit_of_wage	✓ Multivariate variable with 4 unique values. ✓ 'Year' is the most dominant value, implying majority of the hourly-wage values were initially recorded in the yearly unit (though we have standardized the same now).
Categorical	full_time_position	✓ Bivariate variable with 2 unique values. ✓ 'Y' is the most dominant value, implying majority of the applicants prefer applying for a full-time position.
Categorical	case_status	✓ Bivariate variable with 2 unique values. ✓ 'Certified' is the most dominant value, implying majority of the applicants were certified with Visa.
Categorical	company_age	✓ Multivariate variable with 5 unique values. ✓ '10-20' is the most dominant value, implying majority of the companies where applicants apply are 10-20 years old.
Numerical	hourly_wage	✓ Hourly wages range between ~$0.05 ~$7004, which is a very wide range as expected, depicting wage-divide in the job market. ✓ Mean is ~$95 & median is ~$40 ✓ Standard Deviation is ~$278
Categorical	company_size	✓ Multivariate variable with 5 unique values. ✓ '0-1000' is the most dominant value, implying majority of the companies where applicants apply have employees in the range 0-1000.

Table 1: Statistical Summary – Observations

Univariate & Bivariate Analysis

- **Perform Univariate Analysis** – Use Histograms & Boxplots to analyse each numerical variable, followed by Barplots for categorical variables: -
    1. Distribution of **hourly_wage**: -

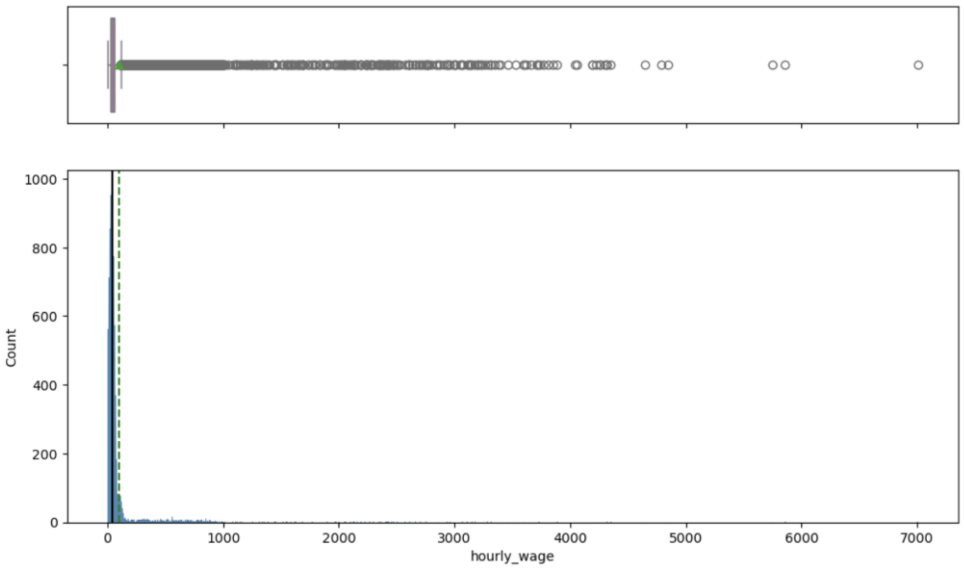


Figure 10: Univariate Analysis – hourly_wage

✓ **Observations & Insights** can be summarized below: -

- There is a large variation in the hourly wage of employees.
- Distribution seems to be highly right-skewed with different mean & median.
- Seems to be unimodal distribution having a single peak.
- Lot of outliers observed, but we would keep the data as-is to avoid any loss of information.

2. Distribution of **continent** & **region_of_employment**: -

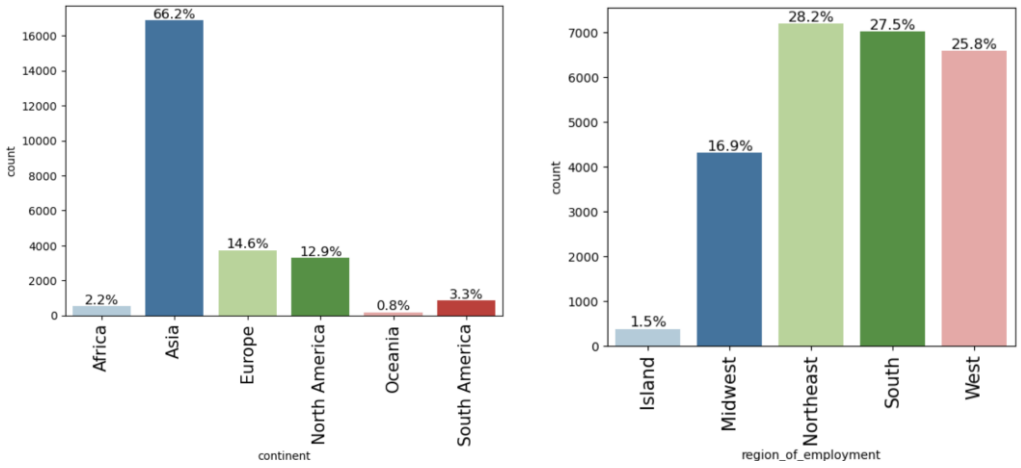


Figure 11: Univariate Analysis – continent & region_of_employment

✓ **Observations & Insights** can be summarized below: -

- Majority of the applicants come from Asean region (66%), followed by Europe (15%) & North America (13%).
- Most of the applicants prefer employment in the Northeast (28%), followed by South (27%) & West (25%) regions.

3. Distribution of **education_of_employee, has_job_experience** & **requires_job_training:** -

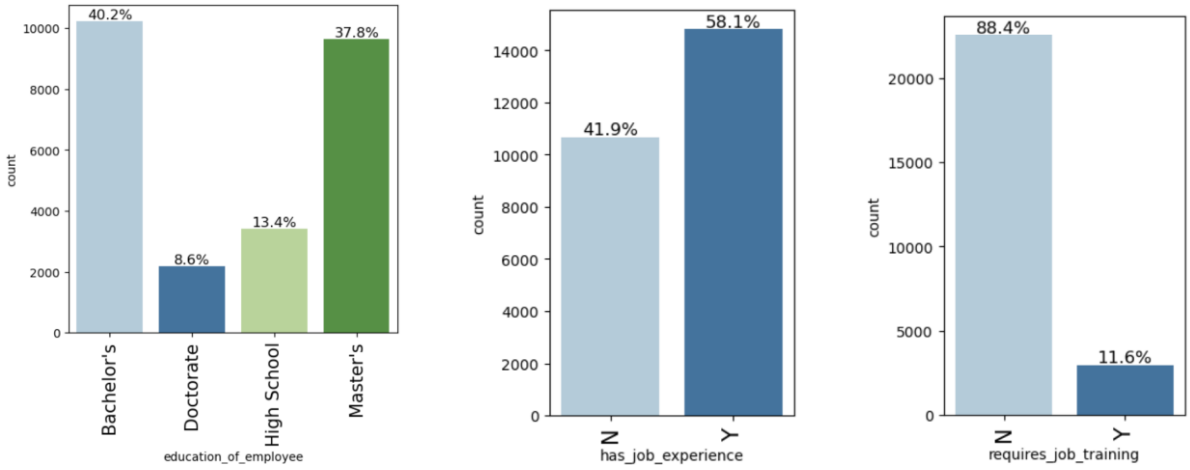


Figure 12: Univariate Analysis – education_of_employee, has_job_experience & requires_job_training

✓ **Observations & Insights** can be summarized below: -

- Majority of the applicants have either a Bachelor's Degree (40%) or a Master's Degree (38%), followed by High School (13%). Very few have Doctorate (9%).
- Majority of the applicants have prior work-experience (58%) as against those who are freshers (42%).
- Most of the applicants required job training (88%) as against who did not (12%).

4. Distribution of **full_time_position** & **unit_of_wage**: -

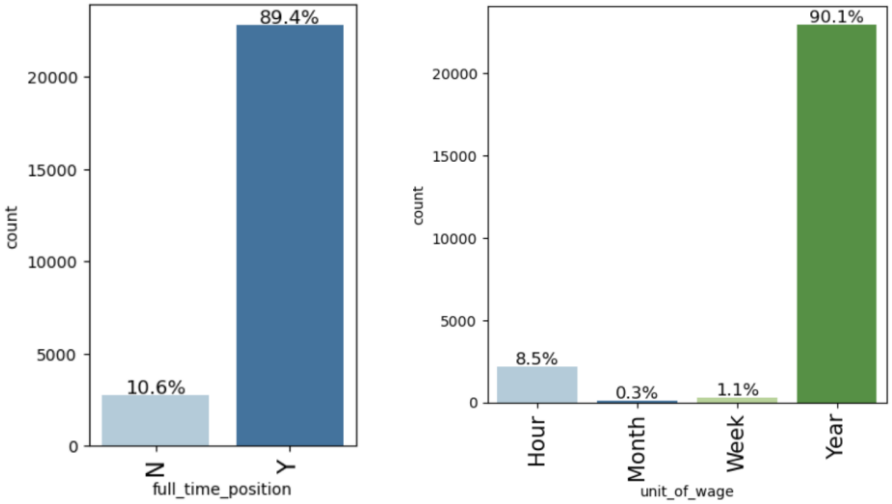


Figure 13: Univariate Analysis – full_time_position & unit_of_wage

✓ **Observations & Insights** can be summarized below: -

- Majority of the applicants applied for a full-time position (89%) as against temporary position (11%).
- Most of the applicants' unit of wage is yearly (90%) as against Hourly/Weekly/Monthly (10%).

5. Distribution of **company_age** & **company_size:** -

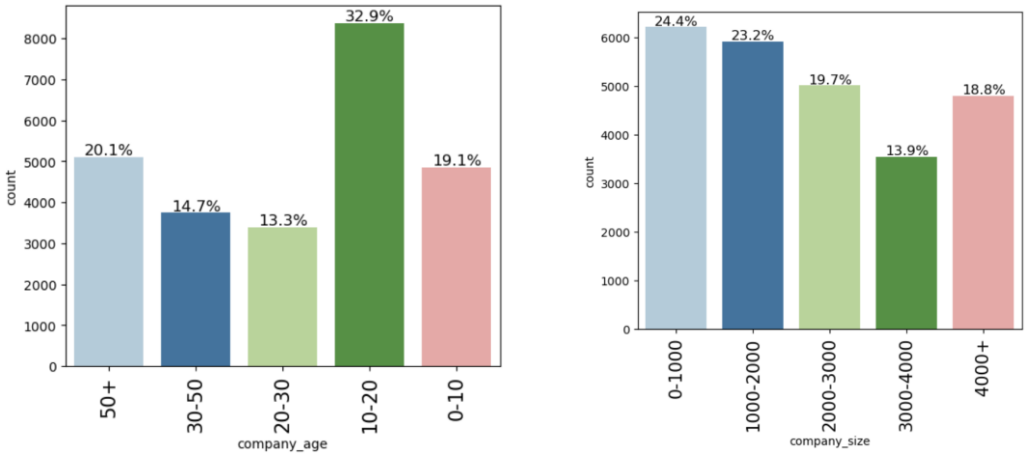


Figure 14: Univariate Analysis – company_age & company_size

✓ **Observations & Insights** can be summarized below: -

- Majority of the applicants prefer company that is 10-20 years old (33%), followed by 50+ years (21%) & 0-10 years (19%).
- Majority of the employers of the applicants have 0-1000 employees (24%), followed by 1000-2000 (23%) & 2000-3000 (20%).

6. Distribution of **case_status**: -

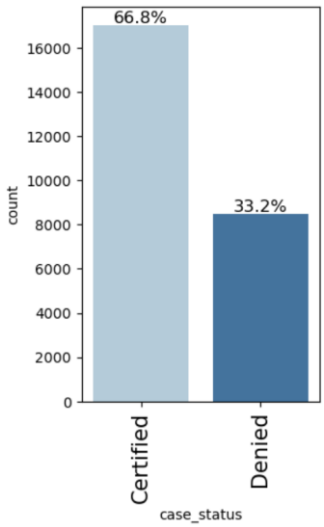


Figure 15: Univariate Analysis – case_status

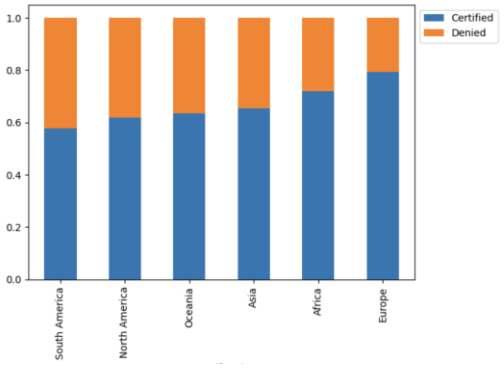
✓ **Observations & Insights** can be summarized below: -

- Majority of the applicants, as per the dataset, get Certified with Visa (67%), as against Denied with Visa (33%).

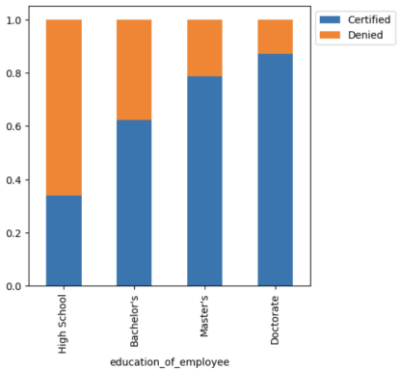
- **Perform Bivariate Analysis** – Use Stacked Barplots to carry out bivariate analysis between categorical variables, followed by Distribution plots between numerical & categorical variables (Please note, we don't need to use a heatmap as we only have 1 numerical variable. There is no other numerical variable to analyse the relationship trend): -

1. **Between Categorical Variables (with Target Variable 'case_status', as it is the Subject of Interest)**: -

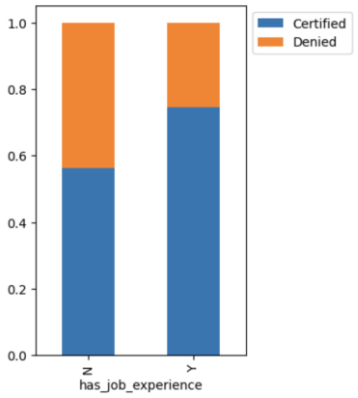
case_status	Certified	Denied	All
continent			
All	17018	8462	25480
Asia	11012	5849	16861
North America	2037	1255	3292
Europe	2957	775	3732
South America	493	359	852
Africa	397	154	551
Oceania	122	70	192



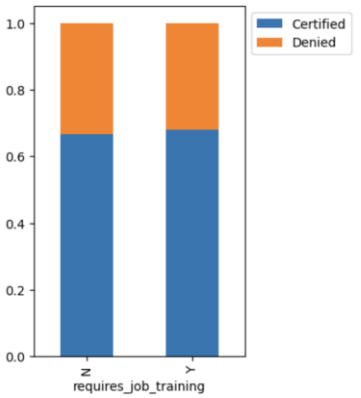
case_status	Certified	Denied	All
education_of_employee			
All	17018	8462	25480
Bachelor's	6367	3867	10234
High School	1164	2256	3420
Master's	7575	2059	9634
Doctorate	1912	280	2192



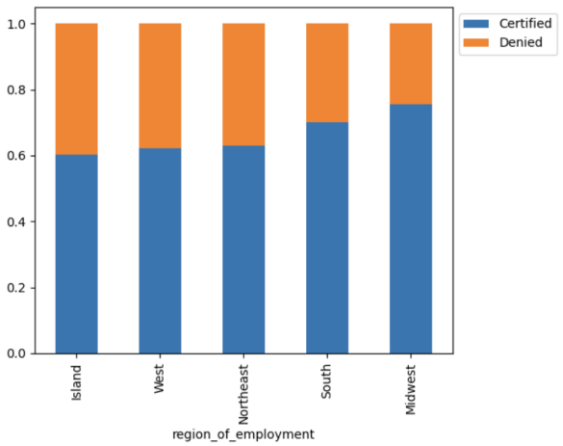
case_status	Certified	Denied	All
has_job_experience			
All	17018	8462	25480
N	5994	4684	10678
Y	11024	3778	14802



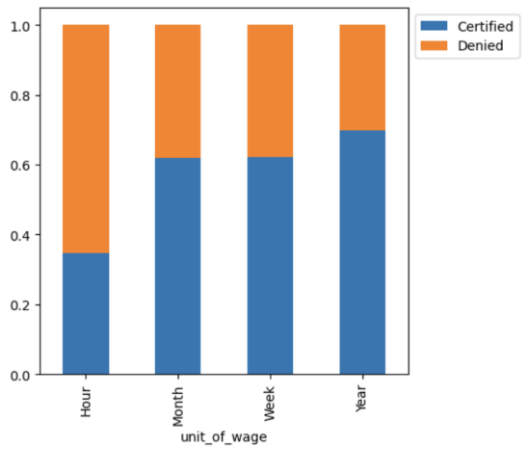
case_status	Certified	Denied	All
requires_job_training			
All	17018	8462	25480
N	15012	7513	22525
Y	2006	949	2955



case_status	Certified	Denied	All
region_of_employment			
All	17018	8462	25480
Northeast	4526	2669	7195
West	4100	2486	6586
South	4913	2104	7017
Midwest	3253	1054	4307
Island	226	149	375



case_status	Certified	Denied	All
unit_of_wage			
All	17018	8462	25480
Year	16047	6915	22962
Hour	747	1410	2157
Week	169	103	272
Month	55	34	89



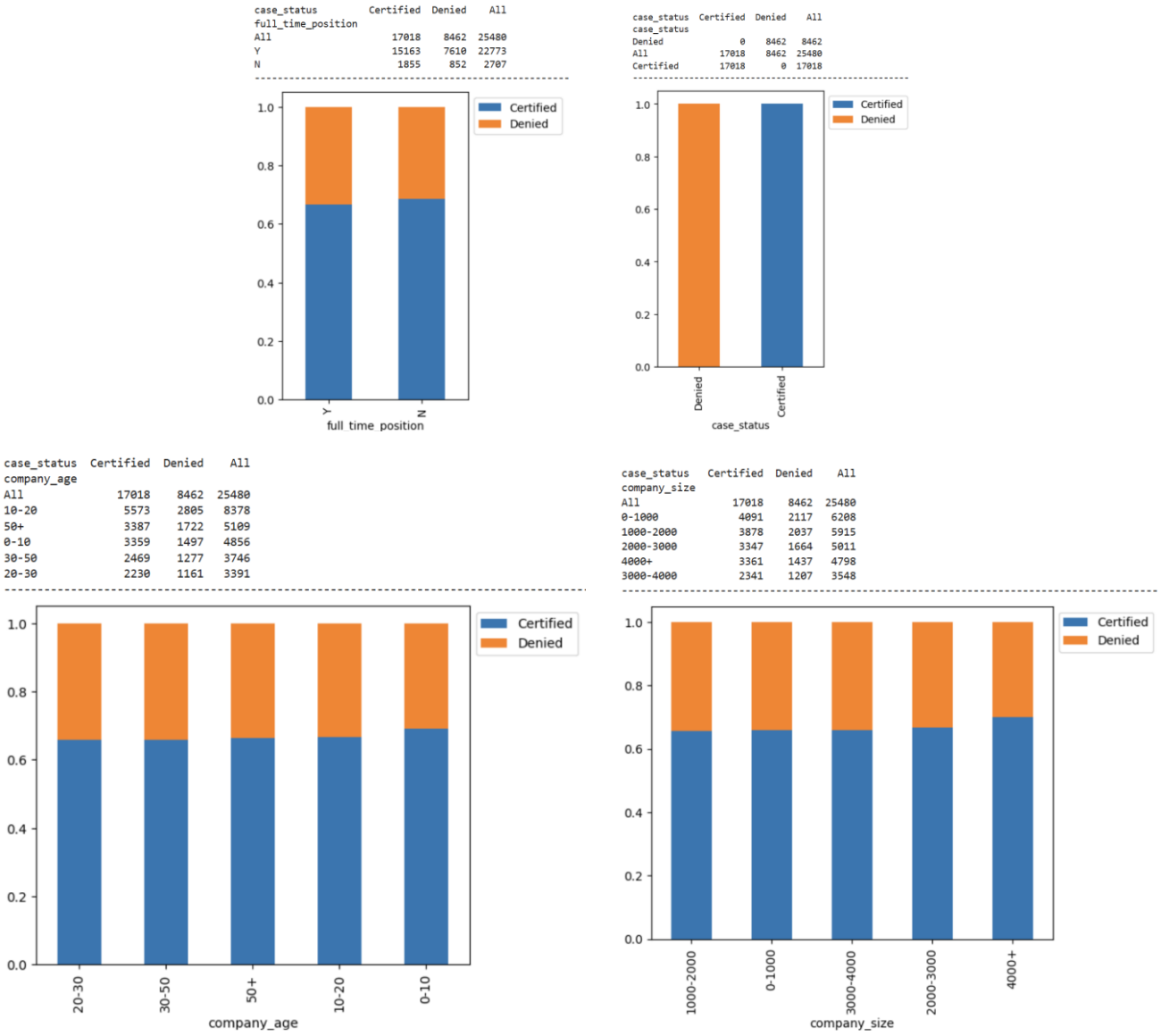


Figure 16: Bivariate Analysis (Categorical vs Categorical) - Relationship between Target Variable (case_status)

✓ **Observations & Insights** can be summarized below: -

- **Case Status vs Continent**: -
  - Europe and Africa have higher certification rates compared to other regions.
  - America (North & South) shows the highest denial rate, indicating stricter criteria or systemic issues.
- **Case Status vs Education**: -
  - Higher the education level (Doctorate & Master's) of the applicants, the more their chances of visa certification
  - Similarly, the likelihood of visa certification is the lowest with the education level as High School.
- **Case Status vs Job Experience**: -
  - Having prior job experience increases the likelihood of visa certification.
- **Case Status vs Job Training Requirement**: -
  - Visa certification likelihood is nearly unaffected by the job training requirement.
- **Case Status vs Region of Employment**: -
  - Visa applications filed by the employers in the Midwest region have the highest likelihood of visa certification.
- **Case Status vs Unit of Wage**: -
  - Applicants whose wage unit is year are more likely to get visa certified than other applicants.
  - Applicants who are paid by hour are the least likely to be certified for a visa. It can be said that hourly jobs are usually less important for the growth of the United States and they could be done by normal American workers.

- **Case Status vs Full-time Position**: -
  - Visa certification seems to be unaffected by whether a position is full-time or part-time.
- **Case Status vs Company-age**: -
  - Visa certification seems to be unaffected by the age of the employer.
- **Case Status vs Company-size**: -
  - Visa certification seems to be unaffected by the no. of employees in a company.

2. **Between Categorical & Numerical Variables** -

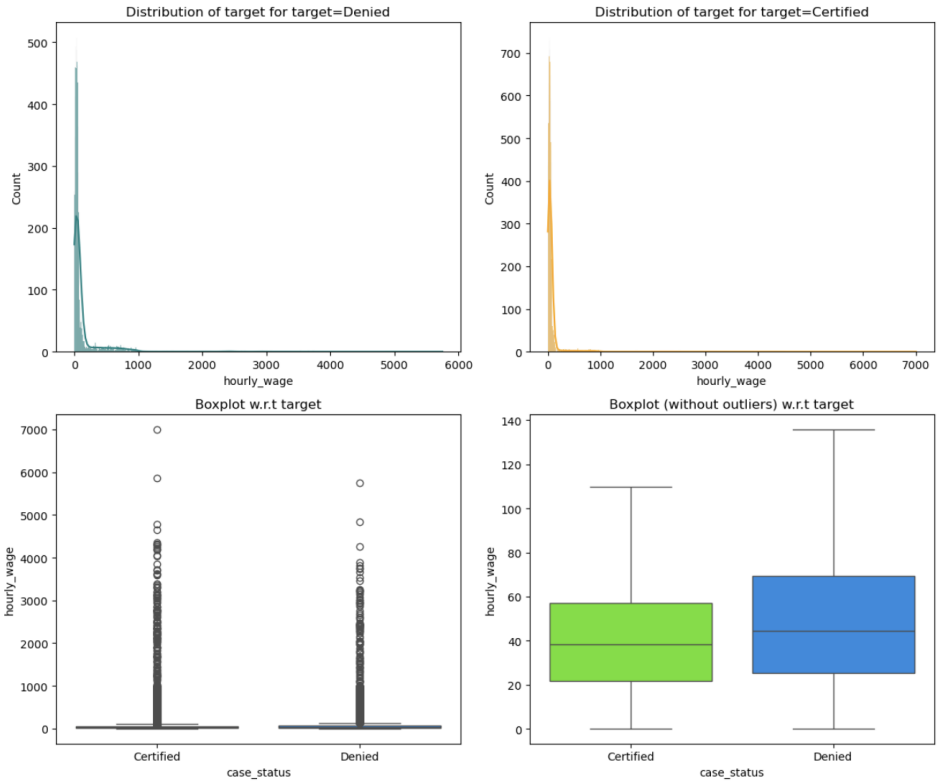


Figure 17: Bivariate Analysis (Categorical vs Numerical) – case_status vs hourly_wage

✓ **Observations & Insights** can be summarized below: -

- The hourly wage distribution for denied cases is highly skewed to the right, with most values concentrated at lower hourly wages. Very few denied applications have higher hourly wages.
- Similarly, the hourly wage distribution for certified cases is also highly right-skewed but seems to have a alightly broader spread compared to denied cases. Certified applications tend to have slightly higher hourly wages than denied ones overall.
- Both certified and denied applications have a significant number of outliers, indicating a few applications with extremely high hourly wages.
- Denied cases exhibit higher variability in hourly wages as compared to certified cases.
- While certified cases generally have lower median wages compared to denied cases, the narrower spread in certified wages suggests that consistency in meeting wage thresholds plays a role in approval

▪ **Please Note** – While we tried to establish first impressions & trends as to how various variables may/may-not impact whether the application is certified or denied, we shall go ahead & **Build Classifier Models through Machine Learning** to **establish Feature importances** in the subsequent sections.

Rubric Question 2: Data Preprocessing

Duplicate & Missing/Error Value-check

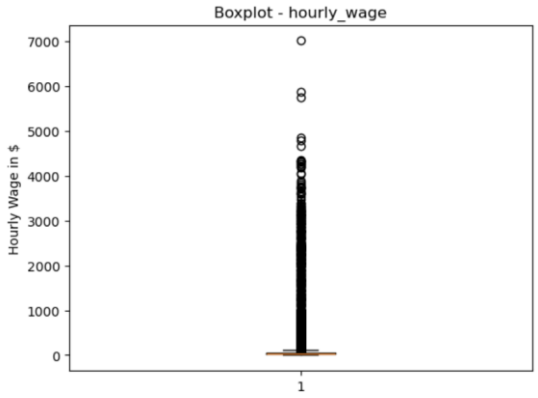
- Please refer Check & Treat Missing/Duplicate Values section.
- No treatment required as explained in the section above.

Feature Engineering

- Please refer Data Transformation/Feature Engineering section.
- 'company_age' derived from 'yr_of_estab' & converted to bins (categorical).
- 'company_size' derived from 'no_of_employees' & converted to bins (categorical).
- 33 error values (negative values in 'company_size') replaced with median.
- 'hourly_wage' derived from 'prevailing_wage', post standardization of wages based on unit of wage.
- Redundant columns dropped → please refer Dropping redundant columns section.

Outlier Treatment

- Below is a summary of the histograms & outlier information for the one numerical variable (hourly_wage): -




```
Lower Wishker at -33.526853281853285 | Upper Whisker at 116.5061003861004
Lower Whisker Outlier Count = 0
Upper Whisker Outlier Count = 2385
Total Outlier Count= 2385
Outlier Percentage in hourly_wage= 9.36028257456829 %
```

Figure 18: Outlier Inspection

- There is **significant no. of outliers (9.4%)** in 'hourly_wage' column, however, this also shows the broader spectrum of wage-divide in the United States.
- We **choose not to treat the outliers to avoid any loss of information**. Such a wage disparity is not unlikely in a country like United States.

Label Encoding

- We shall convert categorical columns into numerical ones so that they can be fitted by machine learning models which only take numerical data.
- **Replace: -**
  - 'Certified' → 1
  - 'Denied'→ 0

```
case_status
1    17018
0     8462
Name: count, dtype: int64
```

Figure 19: Label Encoding of Target Variable

Data Preparation for Modelling

- As a data scientist, we want to analyse the data provided to find which factors have a high influence on visa status, build a model that can predict which application is going to be certified/denied in advance. We carry out following steps to further prepare the data for Modelling: -
  1. Split the data into Dependent (case_status) & Independent variable (remaining others) Data Frames.
  2. Split the data into **Train, Validation and Test datasets** to be able to evaluate the model. Build a model using the Train dataset, tune performance on Validation dataset & finally test the performance of the final model on the Test dataset (i.e. unseen data). This split is done to **avoid data leakage**.
  3. Before we proceed to build a model, we encode categorical features.

- First, we **split** the data into **Training Dataset & Temporary dataset,** using random_state value 1, in the **ratio of 80:20**. We use **stratified sampling technique** to avoid any bias in the sampling. Then we **further split** the Temporary dataset into **Validation dataset & Test dataset** using the same parameters. Below is the summary of the split datasets: -

```
Train dataset shape:  (15288, 10)
Validation dataset shape:  (5096, 10)
Test dataset shape:  (5096, 10)

Percentage of classes in Train dataset:  case_status
1   0.668
0   0.332
Name: proportion, dtype: float64

Percentage of classes in Validation dataset:  case_status
1   0.668
0   0.332
Name: proportion, dtype: float64

Percentage of classes in Test dataset:  case_status
1   0.668
0   0.332
Name: proportion, dtype: float64
```

Figure 20: Train-Validation-Test Data Split Summary

- **Encode the categorical variables through One-Hot Encoding technique**. Below is the summary of the datasets post encoding. We can see below; the columns have increased to 27: -

```
Train dataset shape post encoding:  (15288, 27)
Validation dataset shape post encoding:  (5096, 27)
Test dataset shape post encoding:  (5096, 27)
```

Figure 21: Dataset Summary post One-Hot Encoding

- Below is the snapshot of top 5 rows of Train Dataset with all 27 columns (in Transpose format for better visibility): -

	5008	12951	3214	18876	21939
hourly_wage	34.228	28.515	10.732	9.140	31.808
continent_Asia	1.000	0.000	1.000	0.000	1.000
continent_Europe	0.000	1.000	0.000	1.000	0.000
continent_North America	0.000	0.000	0.000	0.000	0.000
continent_Oceania	0.000	0.000	0.000	0.000	0.000
continent_South America	0.000	0.000	0.000	0.000	0.000
education_of_employee_Doctorate	0.000	0.000	0.000	0.000	0.000
education_of_employee_High School	0.000	0.000	0.000	0.000	0.000
education_of_employee_Master's	0.000	1.000	0.000	0.000	0.000
has_job_experience_Y	1.000	1.000	0.000	0.000	0.000
requires_job_training_Y	0.000	1.000	0.000	0.000	0.000
region_of_employment_Midwest	0.000	0.000	1.000	0.000	0.000
region_of_employment_Northeast	0.000	1.000	0.000	0.000	0.000
region_of_employment_South	1.000	0.000	0.000	0.000	1.000
region_of_employment_West	0.000	0.000	0.000	1.000	0.000
unit_of_wage_Month	0.000	0.000	0.000	0.000	0.000
unit_of_wage_Week	0.000	0.000	0.000	0.000	0.000
unit_of_wage_Year	1.000	1.000	1.000	1.000	1.000
full_time_position_Y	1.000	1.000	1.000	1.000	1.000
company_age_30-50	0.000	0.000	0.000	0.000	0.000
company_age_20-30	0.000	0.000	1.000	0.000	0.000
company_age_10-20	0.000	1.000	0.000	0.000	0.000
company_age_0-10	1.000	0.000	0.000	0.000	1.000
company_size_1000-2000	1.000	1.000	0.000	0.000	0.000
company_size_2000-3000	0.000	0.000	0.000	0.000	1.000
company_size_3000-4000	0.000	0.000	0.000	0.000	0.000
company_size_4000+	0.000	0.000	0.000	0.000	0.000

Figure 22: Train Dataset (Top 5 Rows) post One-Hot Encoding

Rubric Question 3: Model building – Original Data

Model Evaluation Criteria

- **Best Performance Metric for the Problem – F1 Score:** The F1 score is the best primary performance evaluation metric because it addresses the key business objective of **correctly classifying both certified and denied cases in an imbalanced data context**, ensuring a balance between precision and recall.
- Visa approval datasets typically exhibit **class imbalance**, where the majority of cases may be certified, and a minority may be denied. In such cases, traditional metrics like **Accuracy can be misleading** because a model that predicts most cases as 'certified' will show high accuracy but fail to identify denied cases correctly.
- The **business goal is to correctly classify both certified and denied cases**:
  - Misclassifying an applicant with high chances of visa denial as "certified" (False Positive) could lead to unnecessary delays or rejections, causing inefficiency in the visa processing pipeline. On the other hand, misclassifying a likely "certified" applicant as "denied" (False Negative) could deprive businesses of valuable talent.
  - A **False Positive would lead to the waste of the OFLC's time and staff resources**, while a **False Negative would prevent a qualified applicant who could fill essential jobs** in the United States from receiving work visa. Therefore, it appears that both errors could be equally important for the OFLC to be minimized.
  - We need to **minimize, both, False Positives & False Negatives simultaneously.**
  - Since, **high Recall scores minimizes False Negatives** & **high Precision scores minimizes False Positives**, we need to choose a metric that balances between both scores.
  - Since, **F1 score is the harmonic mean of Recall & Precision**, it is the **best metric** that balances both scores.

Model-building on Original Data

- Build 5 models with original data using different techniques.
- The techniques deployed are: -
  1. **Decision Tree** – Use DecisionTreeClassifier Function with random_state 1 & class_weight balanced
  2. **Bagging** – Use BaggingClassifier Function with estimator DecisionTreeClassifier & random_state 1
  3. **Random Forest** – Use RandomForestClassifier Function with random_state 1 & class_weight balanced
  4. **Adaboost** – Use AdaBoostClassifier Function with random_state 1
  5. **Gradient Boost Method** – Use GradientBoostingClassifierFunction with random_state 1
- Below is the performance summary on Training & Validation datasets for all 5 Models: -

```
Training Performance:                          Validation Performance:

Decision tree | Original | F1 score - 1.0      Decision tree | Original | F1 score - 0.741
Bagging | Original | F1 score - 0.987          Bagging | Original | F1 score - 0.774
Random forest | Original | F1 score - 1.0      Random forest | Original | F1 score - 0.775
Adaboost | Original | F1 score - 0.819         Adaboost | Original | F1 score - 0.816
Gradient Boost| Original | F1 score - 0.828    Gradient Boost| Original | F1 score - 0.826

Training and Validation Performance Difference:

Decision tree | Original | Training F1 Score: 1.000, Validation F1 Score: 0.741, Difference: 0.259
Bagging | Original | Training F1 Score: 0.987, Validation F1 Score: 0.774, Difference: 0.214
Random forest | Original | Training F1 Score: 1.000, Validation F1 Score: 0.775, Difference: 0.225
Adaboost | Original | Training F1 Score: 0.819, Validation F1 Score: 0.816, Difference: 0.003
Gradient Boost| Original | Training F1 Score: 0.828, Validation F1 Score: 0.826, Difference: 0.002
```

Figure 23: Model Performance Summary on Original Dataset

1. **Decision Tree: -**
   - ✓ Training F1 Score: 1.000 | Validation F1 Score: 0.741 | Difference: 0.259
   - ✓ The decision tree has a perfect F1 score on the training data but a significantly lower F1 score on the validation data, resulting in a large difference of **0.259**.
   - ✓ **Interpretation**: The decision tree is likely **overfitting** the training data, meaning it is memorizing the training data instead of generalizing well to unseen data.
2. **Bagging: -**
   - ✓ Training F1 Score: 0.987 | Validation F1 Score: 0.774| Difference: 0.214
   - ✓ Bagging performs better than the decision tree in terms of validation performance, but there is still a noticeable gap between training and validation F1 scores, with a difference of **0.214**.
   - ✓ **Interpretation**: Bagging reduces overfitting compared to a single decision tree but **still shows signs of overfitting**.

3. **Random Forest**
   - ✓ Training F1 Score: 1.000 | Validation F1 Score: 0.775 | Difference: 0.225
   - ✓ Random forest shows a perfect F1 score on the training data but a moderate F1 score of **0.775** on the validation data, with a difference of **0.225**.
   - ✓ **Interpretation**: Random forest is also **overfitting** the training data, but it performs similarly to bagging in validation.
4. **Adaboost**
   - ✓ Training F1 Score: 0.819 | Validation F1 Score: 0.816 | Difference: 0.003
   - ✓ AdaBoost shows a much smaller difference between training and validation F1 scores, with a difference of only **0.003**, indicating **good generalization**.
   - ✓ **Interpretation**: AdaBoost is **performing well with minimal overfitting**. It strikes a good balance between fitting the training data and generalizing to unseen data.
5. **Gradient Boost Method**
   - ✓ Training F1 Score: 0.828 | Validation F1 Score: 0.826 | Difference: 0.002
   - ✓ Gradient Boost has the smallest difference of **0.002** between training and validation F1 scores, indicating **excellent generalization**. Its validation **F1 score of 0.826 is the highest among all models**.
   - ✓ **Interpretation**: Gradient Boost **performs the best, with high validation performance and almost no overfitting**.

- **To Summarize**: -
  - The **Decision Tree**, **Bagging**, and **Random Forest** models suffer from **overfitting**, as indicated by the large differences between their training and validation F1 scores.
  - **AdaBoost** and **Gradient Boost generalize well**, with very small differences between their training and validation F1 scores.
  - **Gradient Boost** is the **best model** overall due to its highest validation F1 score and minimal overfitting.

Rubric Question 4: Model building – Oversampled Data

Oversampling Datasets

- Use **SMOTE** (Synthetic Minority Oversampling TEchnique), one of the Imblearn Techniques, to oversample the data.
  - It consists of synthesizing elements for the minority class, based on those that already exist.
  - Randomly picks a point from the minority class and computes the k-nearest neighbours for this point.
  - Synthetic points are added between the chosen point and its neighbours.

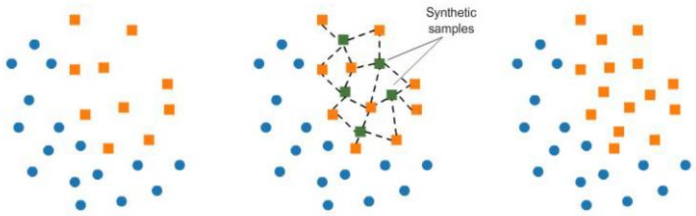


Figure 24: How SMOTE Works?

- Use **SMOTE Function** with **K-nearest neighbours 5** to oversample the Training dataset. Below is the summary of pre/post deployment of SMOTE to Training dataset: -

```
Before Oversampling, counts of label 'Yes': 10210
Before Oversampling, counts of label 'No': 5078

After Oversampling, counts of label 'Yes': 10210
After Oversampling, counts of label 'No': 10210

After Oversampling, the shape of train_X: (20420, 27)
After Oversampling, the shape of train_y: (20420,)
```

Figure 25: Train Dataset Summary post SMOTE deployment

- Clearly, the label counts for 'Yes' & 'No' have been balanced post SMOTE, thus, removing the bias from the dataset.

Model-building on Oversampled Data

- Build 5 models with oversampled data using different techniques.
- The techniques deployed are: -
  1. **Decision Tree** – Use DecisionTreeClassifier Function with random_state 1 & class_weight balanced
  2. **Bagging** – Use BaggingClassifier Function with estimator DecisionTreeClassifier & random_state 1
  3. **Random Forest** – Use RandomForestClassifier Function with random_state 1 & class_weight balanced
  4. **Adaboost** – Use AdaBoostClassifier Function with random_state 1
  5. **Gradient Boost Method** – Use GradientBoostingClassifierFunction with random_state 1
- Below is the performance summary on Training & Validation datasets for all 5 Models: -

```
Training Performance:                          Validation Performance:

Decision tree | Oversampled | F1 score - 1.0   Decision tree | Oversampled | F1 score - 0.742
Bagging | Oversampled | F1 score - 0.986       Bagging | Oversampled | F1 score - 0.763
Random forest | Oversampled | F1 score - 1.0   Random forest | Oversampled | F1 score - 0.772
Adaboost | Oversampled | F1 score - 0.777      Adaboost | Oversampled | F1 score - 0.788
Gradient Boost | Oversampled | F1 score - 0.81  Gradient Boost | Oversampled | F1 score - 0.81

Training and Validation Performance Difference:

Decision tree | Oversampled | Training F1 Score: 1.000, Validation F1 Score: 0.742, Difference: 0.258
Bagging | Oversampled | Training F1 Score: 0.986, Validation F1 Score: 0.763, Difference: 0.223
Random forest | Oversampled | Training F1 Score: 1.000, Validation F1 Score: 0.772, Difference: 0.228
Adaboost | Oversampled | Training F1 Score: 0.777, Validation F1 Score: 0.788, Difference: -0.011
Gradient Boost | Oversampled | Training F1 Score: 0.810, Validation F1 Score: 0.810, Difference: -0.001
```

Figure 26: Model Performance Summary on Oversampled Dataset

1. **Decision Tree: -**
   - ✓ Training F1 Score: 1.000 | Validation F1 Score: 0.742 | Difference: 0.258
   - ✓ Despite achieving a perfect F1 score on the training data, the decision tree shows a significant drop in the validation F1 score, resulting in a large difference of **0.258**.
   - ✓ **Interpretation**: The decision tree is **overfitting** even on the oversampled data, as indicated by the high difference between training and validation performance.
2. **Bagging: -**
   - ✓ Training F1 Score: 0.986 | Validation F1 Score: 0.763| Difference: 0.223
   - ✓ Bagging performs better than the decision tree in terms of validation F1 score, but it still shows a noticeable difference of **0.223** between training and validation performance.
   - ✓ **Interpretation**: Bagging reduces overfitting compared to a single decision tree but **still shows signs of overfitting**.
3. **Random Forest**
   - ✓ Training F1 Score: 1.000 | Validation F1 Score: 0.772 | Difference: 0.228
   - ✓ Random forest again shows a perfect F1 score on the training data and a moderate validation F1 score of **0.772**, with a difference of **0.228**.
   - ✓ **Interpretation**: Random forest, like bagging, is **overfitting** on the oversampled data, which is evident from the gap between training and validation F1 scores.
4. **Adaboost**
   - ✓ Training F1 Score: 0.777 | Validation F1 Score: 0.788 | Difference: -0.011
   - ✓ AdaBoost shows a slight negative difference of **-0.011**, meaning the validation F1 score is slightly higher than the training F1 score. This indicates that the model is **generalizing well** to unseen data without overfitting.
   - ✓ **Interpretation**: AdaBoost **performs well on oversampled data**, with **no significant overfitting** and good validation performance.
5. **Gradient Boost Method**
   - ✓ Training F1 Score: 0.810 | Validation F1 Score: 0.810 | Difference: -0.001
   - ✓ Gradient Boost achieves identical F1 scores on both training and validation data, with a minimal difference of **-0.001**. Its validation **F1 score of 0.810 is the highest** among all models.
   - ✓ **Interpretation**: Gradient Boost is the **best-performing model on oversampled data**, with **excellent generalization** and no signs of overfitting.

- **To Summarize**: -
  - The **Decision Tree**, **Bagging**, and **Random Forest** models suffer from **overfitting**, as indicated by the large differences between their training and validation F1 scores.
  - **AdaBoost** and **Gradient Boost generalize well**, with very small differences between their training and validation F1 scores.
  - **Gradient Boost** is the **best model** overall due to its highest validation F1 score and minimal overfitting.

Rubric Question 5: Model building – Undersampled Data

Undersampling Datasets

- Use **Random Undersampling Technique**, one of the Imblearn Techniques, to undersample the data.
  - This is used to increase the frequency of minority class by undersampling the majority class(es) by randomly picking samples with or without replacement.
  - Use **RandomUnderSampler Function** to undersample the Training dataset. Below is the summary of pre/post deployment of RandomUnderSampler to Training dataset: -

```
Before Under Sampling, counts of label 'Yes': 10210
Before Under Sampling, counts of label 'No': 5078

After Under Sampling, counts of label 'Yes': 5078
After Under Sampling, counts of label 'No': 5078

After Under Sampling, the shape of train_X: (10156, 27)
After Under Sampling, the shape of train_y: (10156,)
```

Figure 27: Train Dataset Summary post RandomUnderSampler deployment

- Clearly, the label counts for 'Yes' & 'No' have been balanced post RandomUnderSampler, thus, removing the bias from the dataset.

Model-building on Undersampled Data

- Build 5 models with oversampled data using different techniques.
- The techniques deployed are: -
  1. **Decision Tree** – Use DecisionTreeClassifier Function with random_state 1 & class_weight balanced
  2. **Bagging** – Use BaggingClassifier Function with estimator DecisionTreeClassifier & random_state 1
  3. **Random Forest** – Use RandomForestClassifier Function with random_state 1 & class_weight balanced
  4. **Adaboost** – Use AdaBoostClassifier Function with random_state 1
  5. **Gradient Boost Method** – Use GradientBoostingClassifierFunction with random_state 1
- Below is the performance summary on Training & Validation datasets for all 5 Models: -

```
Training Performance:                                  Validation Performance:

Decision tree | Undersampled | F1 score - 1.0          Decision tree | Undersampled | F1 score - 0.685
Bagging | Undersampled | F1 score - 0.976              Bagging | Undersampled | F1 score - 0.693
Random forest | Undersampled | F1 score - 1.0          Random forest | Undersampled | F1 score - 0.711
Adaboost | Undersampled | F1 score - 0.704            Adaboost | Undersampled | F1 score - 0.762
Gradient Boost | Undersampled | F1 score - 0.726      Gradient Boost | Undersampled | F1 score - 0.777

Training and Validation Performance Difference:

Decision tree | Undersampled | Training F1 Score: 1.000, Validation F1 Score: 0.685, Difference: 0.315
Bagging | Undersampled | Training F1 Score: 0.976, Validation F1 Score: 0.693, Difference: 0.283
Random forest | Undersampled | Training F1 Score: 1.000, Validation F1 Score: 0.711, Difference: 0.289
Adaboost | Undersampled | Training F1 Score: 0.704, Validation F1 Score: 0.762, Difference: -0.058
Gradient Boost | Undersampled | Training F1 Score: 0.726, Validation F1 Score: 0.777, Difference: -0.051
```

Figure 28: Model Performance Summary on Undersampled Dataset

1. **Decision Tree: -**
   - ✓ Training F1 Score: 1.000 | Validation F1 Score: 0.685 | Difference: 0.315
   - ✓ The decision tree achieves a perfect F1 score on the training data but has a much lower F1 score on the validation data, with a large difference of **0.315**.
   - ✓ **Interpretation**: The decision tree is **overfitting** even on undersampled data, meaning it is not generalizing well to new data.
2. **Bagging: -**
   - ✓ Training F1 Score: 0.976 | Validation F1 Score: 0.693| Difference: 0.283
   - ✓ Bagging performs slightly better than the decision tree, but it still shows a significant difference of **0.283** between training and validation F1 scores.

- ✓ **Interpretation**: Bagging reduces overfitting compared to the decision tree but still suffers from **some degree of overfitting**.

3. **Random Forest**
   - ✓ Training F1 Score: 1.000 | Validation F1 Score: 0.711 | Difference: 0.289
   - ✓ Random forest also achieves a perfect F1 score on the training data but performs moderately on the validation data, with a difference of **0.289**.
   - ✓ **Interpretation**: Like bagging, random forest suffers from **overfitting** and does not generalize well, despite being trained on undersampled data.
4. **Adaboost**
   - ✓ Training F1 Score: 0.704 | Validation F1 Score: 0.762 | Difference: -0.058
   - ✓ AdaBoost shows a slight negative difference of **-0.058**, meaning the validation F1 score is higher than the training F1 score. This indicates that the model is **generalizing well** and not overfitting.
   - ✓ **Interpretation**: AdaBoost performs well on undersampled data, with **minimal overfitting** and **good validation performance.**
5. **Gradient Boost Method**
   - ✓ Training F1 Score: 0.726 | Validation F1 Score: 0.777 | Difference: -0.051
   - ✓ Gradient Boost also shows a slight negative difference of **-0.051**, indicating that the validation F1 score is slightly higher than the training F1 score. With a validation F1 score of **0.777**, it has the highest validation performance among all models
   - ✓ **Interpretation**: Gradient Boost performs the best on undersampled data, showing excellent generalization and minimal overfitting.

- **To Summarize**: -
  - The **Decision Tree**, **Bagging**, and **Random Forest** models suffer from **overfitting**, as indicated by the large differences between their training and validation F1 scores.
  - **Gradient Boost achieves the highest validation F1 score (0.777) with minimal difference (-0.051).**
  - **AdaBoost is a close second, with a validation F1 score of 0.762 and a slight negative difference (-0.058).**
  - All models show **lower F1 scores compared to their performance on oversampled data.** This is expected, as undersampling reduces the amount of data available for training, which can affect model performance.

Rubric Question 6: Model Performance Improvement using Hyperparameter Tuning

Choose Top 3 Models for Tuning

- Below is the Performance Summary of all the models: -

	Model	Training F1-Score	Validation F1-Score	Difference
0	Decision tree | Original	1.000	0.741	0.259
0	Bagging | Original	0.987	0.774	0.214
0	Random forest | Original	1.000	0.775	0.225
0	Adaboost | Original	0.819	0.816	0.003
0	Gradient Boost| Original	0.828	0.826	0.002
0	Decision tree | Oversampled	1.000	0.742	0.258
0	Bagging | Oversampled	0.986	0.763	0.223
0	Random forest | Oversampled	1.000	0.772	0.228
0	Adaboost | Oversampled	0.777	0.788	-0.011
0	Gradient Boost | Oversampled	0.810	0.810	-0.001
0	Decision tree | Undersampled	1.000	0.685	0.315
0	Bagging | Undersampled	0.976	0.693	0.283
0	Random forest | Undersampled	1.000	0.711	0.289
0	Adaboost | Undersampled	0.704	0.762	-0.058
0	Gradient Boost | Undersampled	0.726	0.777	-0.051

Figure 29: Model Performance Summary Table

```
Decision Tree | F1 Score Difference Average =  0.277
Bagging | F1 Score Difference Average =  0.24
Random Forest | F1 Score Difference Average =  0.247
Adaboost | F1 Score Difference Average =  0.024
Gradient Boost | F1 Score Difference Average =  0.018
```

Figure 30: F1 Score Difference Average for all 5 Models

- **Criteria for Model Selection**: -
  - ✓ **Low Overfitting** i.e. Lowest difference between Training & Validation F1-scores
    - I. Looking at the table above, clearly **Adaboost & Gradient Boost** Models show the least difference in F1-scores. Therefore, they do well in generalizing data (good performance on unseen data).
    - II. Between Decision Tree, Bagging & Random Forest models, that all suffer from overfitting, **Bagging** suffers the least from overfitting in comparison to other two. Let's evaluate if this model can be hyper-tuned to show better performance in subsequent sections.
    - III. Top 3 models on the basis of Overfitting: **Adaboost, Gradient Boost & Bagging**
  - ✓ **High F1-scores on Validation dataset**
    - I. All models show **lower F1 scores compared to their performance on oversampled data, so we choose not to select models on undersampled data**

II. From the top 3 models (that generalize well!), lets **compare the Validation F1-scores between Original & Oversampled** datasets.

- **Gradient Boost**

Dataset	Training F1 Score	Validation F1 Score	Difference	Remarks
Original	0.828	0.826	0.002	High validation F1 score, minimal overfitting.
Oversampled	0.810	0.810	-0.001	Slightly lower F1 score than original, excellent generalization.
Undersampled	0.726	0.777	-0.051	Lower F1 scores, some underfitting indicated.

Table 2: Gradient Boost Model Comparison for Selection

✓ The **Gradient Boost model trained with original dataset** is the best choice for further tuning because it has:
- The highest validation F1 score (0.826).
- Minimal overfitting, with a very small difference of 0.002 between training and validation performance.

- **Adaboost**

Dataset	Training F1 Score	Validation F1 Score	Difference	Remarks
Original	0.819	0.816	0.003	**Highest validation F1 score**, minimal overfitting.
Oversampled	0.777	0.788	-0.011	Slightly lower validation F1 score, minimal overfitting, excellent generalization.
Undersampled	0.704	0.762	-0.058	Lowest validation F1 score, signs of underfitting.

Table 3: AdaBoost Model Comparison for Selection

✓ The **AdaBoost model trained with original dataset** is the best choice for further tuning because it has:
- The highest validation F1 score (0.816).
- Minimal overfitting, with a very small difference of 0.003 between training and validation performance.

- **Bagging**

Dataset	Training F1 Score	Validation F1 Score	Difference	Remarks
Original	0.987	0.774	**0.213**	High validation F1 score but larger difference indicating overfitting.
Oversampled	0.986	0.763	**0.223**	Slightly lower validation F1 score, similar overfitting as original.
Undersampled	0.976	0.693	**0.283**	Lowest validation F1 score and largest difference, indicating underfitting.

Table 4: Bagging Model Comparison for Selection

✓ The **Bagging model trained with original dataset** is the best choice for further tuning because it has:
- The highest validation F1 score (0.774).
- Lowest difference (0.213), indicating better generalization compared to the oversampled and undersampled models.

Hyperparameter Tuning

- Let's tune the performance of the 3 selected Homogeneous models – **Bagging (Original dataset), Adaboost (Original dataset), Gradient Boost Method (Original dataset).**
- After that, we shall feed the above tuned models, as base models, into **Stacking Estimator** (Heterogeneous Ensemble Model) to check if we can further tune the performance.
- We will use **RandomizedSearchCV** function to tune the models: -
  - ✓ RandomizedSearchCV is a hyperparameter optimization technique that helps in finding the best combination of hyperparameters for a machine learning model by randomly sampling from a specified range of values for each hyperparameter.
  - ✓ How it works: -
    I. **Define the Hyperparameter Space**: Specify a range or distribution of values for each hyperparameter you want to tune.
    II. **Specify the Number of Combinations**: Set a fixed number of random combinations to try (via the n_iter parameter).
    III. **Random Sampling**: Randomly samples n_iter combinations of hyperparameters from the specified hyperparameter space.
    IV. **Cross-Validation**: For each sampled combination, perform **k-fold cross-validation** to **evaluate the model's performance** using a scoring metric.
    V. **Select the Best Combination**: The combination of hyperparameters that gives the best cross-validation score is selected as the optimal set.

Tuning Adaboost Model (Original Dataset)

- We will use **AdaBoostClassifier** with random_state 1, only this time we would tune the model by passing **hyperparameters** in the **RandomSearchCV** function.
- Use the following **hyperparameter space**: -
  - n_estimators: 50 to 110 (step by 25)
  - learning_rate: 0.01, 0.1, 0.05
  - estimator: DecisionTreeClassifier with max_depth 2, DecisionTreeClassifier with max_depth 3
- Following is the best combination: -

```
Best parameters are {'n_estimators': 50, 'learning_rate': 0.1, 'estimator': DecisionTreeClassifier(max_depth=3, random_state=1)} with CV score=0.8236220098886333:
```

Figure 31: Best Hyperparameter Combination - Adaboost

- Creating Tuned Adaboost Model with above hyperparameters on Training dataset: -

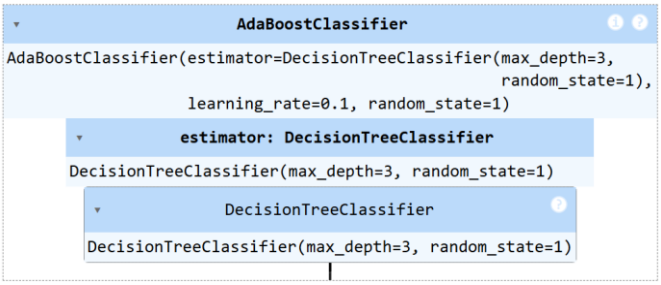


Figure 32: Tuned Adaboost Model

- Evaluating Performance Metrics on Training & Validation datasets: -

Training Performance Metrics - Adaboost | Original Data

	Accuracy	Recall	Precision	F1
0	0.753	0.874	0.782	0.825

Validation Performance Metrics - Adaboost | Original Data

	Accuracy	Recall	Precision	F1
0	0.753	0.868	0.785	0.825

Figure 33: Training & Validation Performance Metrics - Tuned Adaboost Model

- **Performance Summary: -**
  - **High recall (86.8%)** means that the model is effective at correctly identifying most of the denied applications, which is critical for ensuring visa approvals are accurate.
  - **Good precision (78.5%)** ensures that when the model predicts a denial, it is likely correct, reducing unnecessary rejections of valid applications.
  - **Good F1 score (82.5%) confirms that the model performs well in handling both certified and denied cases.**
  - **Consistent performance** across training and validation data indicates that the model is robust and can be **deployed without significant risk of performance degradation on real-world data.**
  - **Conclusion: -**
    - ✓ The AdaBoost model trained on the original dataset performs well, with a strong balance between precision and recall, making it suitable for predicting visa certification and denial.
    - ✓ Since visa denial is a critical decision, the high recall ensures that most applicants who should be denied are correctly identified, minimizing the risk of incorrect certifications.
    - ✓ With good generalization and no significant overfitting, this model can be further fine-tuned or deployed in production for assisting in the visa approval process.

Tuning Gradient Boost Model (Original Dataset)

- We will use **GradientBoostingClassifier** with random_state 1, only this time we would tune the model by passing **hyperparameters** in the **RandomSearchCV** function.
- Use the following **hyperparameter space**: -
  - init: AdaBoostClassifier with random_state=1, DecisionTreeClassifier with random_state 1,
  - n_estimators: 50 to 110 (step by 25)
  - learning_rate: 0.01, 0.1, 0.05
  - subsample: 0.7, 0.9
  - max_features: 0.5, 0.7, 1
- Following is the best combination: -

```
Best parameters are {'subsample': 0.7, 'n_estimators': 100, 'max_features': 0.5, 'learning_rate': 0.05, 'init': AdaBoostClassifier(random_state=1)} with
CV score=0.8233454002701398:
```

Figure 34: Best Hyperparameter Combination - Gradient Boost

- Creating Tuned Gradient Boost Model with above hyperparameters on Training dataset: -

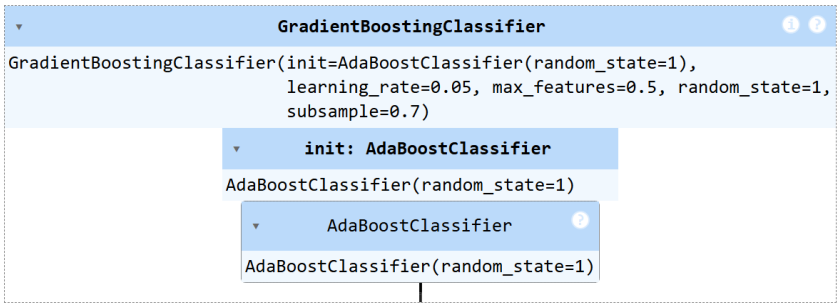


Figure 35: Tuned Gradient Boost Model

- Evaluating Performance Metrics on Training & Validation datasets: -

Training Performance Metrics - Gradient Boost | Original Data

	Accuracy	Recall	Precision	F1
0	0.751	0.875	0.780	0.825

Valdation Performance Metrics - Gradient Boost | Original Data

	Accuracy	Recall	Precision	F1
0	0.751	0.870	0.782	0.823

Figure 36: Training & Validation Performance Metrics - Tuned Gradient Boost Model

- **Performance Summary: -**
  - With a **High Recall (87%)**, the model correctly identifies most of the denied applications, minimizing the risk of incorrect visa certifications. This is critical in ensuring the accuracy of visa approvals.
  - **Good Precision (78.2%),** ensures that when the model predicts a denial, it is likely correct, reducing unnecessary rejections of valid applications.
  - **High F1 score (82.3%), confirms that the model performs well in handling both certified and denied cases**
  - **Good Generalization,** as there is minimal difference between training and validation metrics, suggests that the model can be deployed confidently, as its performance on new, unseen data is likely to be consistent with what is observed in validation.
  - **Conclusion: -**
    - ✓ The tuned Gradient Boost model performs well, with a strong balance between precision and recall, making it suitable for predicting visa certification and denial.
    - ✓ The high F1 strikes a good balance between handling false positives & false negatives, thus, the model performs well in handling both certified and denied cases.
    - ✓ With consistent performance across training and validation datasets, the model is robust and ready for deployment in a real-world visa approval system.

Tuning Bagging Model (Original Dataset)

- We will use **BaggingClassifier** with random_state 1, only this time we would tune the model by passing **hyperparameters** in the **RandomSearchCV** function.
- Use the following **hyperparameter space**: -
  - n_estimators: 50 to 110 (step by 25)
  - max_features: 0.05, 0.07,1
  - estimator: DecisionTreeClassifier with random_state 1 & class_weight balanced
- Following the best combination: -

```
Best parameters are {'n_estimators': 100, 'max_features': 0.7, 'estimator': DecisionTreeClassifier(class_weight='balanced', random_state=1)} with CV scor
e=0.8073170947365014:
```

Figure 37: Best Hyperparameter Combination - Bagging

- Creating Tuned Bagging Model with above hyperparameters on Training dataset: -

```
BaggingClassifier
BaggingClassifier(max_features=0.7, n_estimators=100, random_state=1)
```

Figure 38: Tuned Bagging Model

- Evaluating Performance Metrics on Training & Validation datasets: -

Training Performance Metrics - Bagging | Original Data

	Accuracy	Recall	Precision	F1
0	0.997	1.000	0.996	0.998

Validation Performance Metrics - Bagging | Original Data

	Accuracy	Recall	Precision	F1
0	0.728	0.886	0.751	0.813

Figure 39: Training & Validation Performance Metrics - Tuned Bagging Model

- **Performance Summary: -**
  - **Good Recall (88.6%)** means that the model is effective at correctly identifying most of the denied applications, which is critical for ensuring visa approvals are accurate.
  - **Good precision (75.1%)** ensures that when the model predicts a denial, it is likely correct, reducing unnecessary rejections of valid applications.
  - **Good F1 score (81.3%) confirms that the model performs well in handling both certified and denied cases.**

- The **large gap between the training and validation metrics** (F1 score: 0.998 (training) vs. 0.813 (validation)) indicates that the **model is overfitting**. This means that while the **model performs very well on the training data, it does not generalize well to new, unseen data.**
- **Conclusion: -**
  - ✓ The tuned Bagging model shows signs of overfitting, as evidenced by the large gap between training and validation performance metrics.
  - ✓ While the model achieves High Recall (88.6%) on the validation data, its relatively lower Precision (75.1%) and significant overfitting suggest that it may not be the best model for deployment.

Stacking

- **Stacking** is an **ensemble learning technique** that combines predictions from multiple base models (diverse models of different types) to produce a stronger overall model. Instead of simply averaging the predictions (like bagging or boosting), stacking involves training a **meta-model** that learns how to best combine the outputs of the base models.
- **How Stacking Works?**
  - **I.** Train Base Models: Multiple **diverse models** are trained independently on the same dataset. These base models can be of different types or even the same type but with different hyperparameters. **In our case, we shall use the above 3 tuned models are base models**.
  - II. Generate Meta-Features: The predictions from each base model are collected and used as input features to train the meta-model.
  - III. Train the Meta-Model: A **meta-model** is trained on the predictions generated by the base models. The meta-model learns how to combine the predictions of the base models to make the final prediction.
  - IV. Final Prediction: When predicting for new data, the base models first generate predictions, and these predictions are fed into the meta-model, which outputs the final prediction.
- We will use **XGBClassifier** as the **final estimator**, that would be used to combine the predictions of the base models (as input features) and learn how to optimally combine them to produce the final prediction.
- **Base models** (or simply, 'estimators') – **Tuned Bagging Model, Tuned Adaboost Model, Tuned Gradient Boost Model** (already built in the previous section).
- **StackingClassifier** function with Base models & Final Estimator (as specified above) is used, along with cv 5 (i.e. 5-fold cross validation), to build the model on Training dataset: -

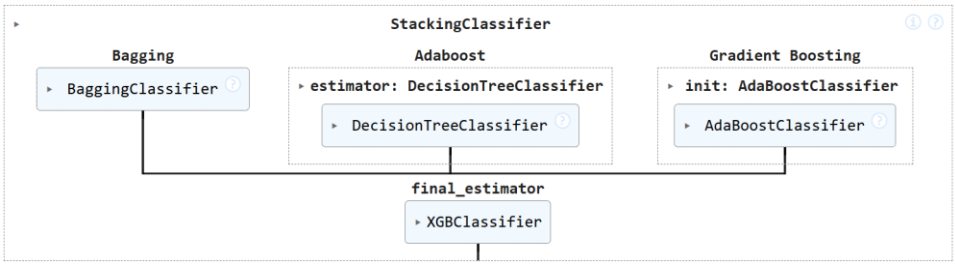


Figure 40: Stacking Model

- Evaluating Performance Metrics on Training & Validation datasets: -

Training Performance Metrics - Stacking | Original Data

	Accuracy	Recall	Precision	F1
0	0.747	0.841	0.793	0.816

Validation Performance Metrics - Stacking | Original Data

	Accuracy	Recall	Precision	F1
0	0.739	0.848	0.780	0.813

Figure 41: Training & Validation Performance Metrics - Stacking Model

- **Performance Summary: -**
  - **Good Recall (84.8%)** means that the model is effective at correctly identifying denied applications. This is critical for the visa approval process, as failing to identify denied cases (false negatives) could lead to incorrect certifications.
  - **Good precision (78%)** means that the model performs reasonably well in minimizing false positives (i.e., unnecessary rejections of valid applications).
  - **Good F1 score (81.3%) indicates a well-balanced model that maintains good performance across both classes (certified and denied).**
  - The minimal difference between training and validation metrics confirms that the **stacking model generalizes well to new, unseen data, making it a reliable model for deployment.**
  - **Conclusion: -**
    - ✓ The stacking model performs well, with high recall, good precision, and a high F1 score, making it suitable for predicting visa certification and denial.
    - ✓ Since the model shows **no significant overfitting** and generalizes well, it is a strong candidate for use in real-world visa decision-making.

Rubric Question 7: Model Performance Comparison and Final Model Selection

Training Dataset Performance Comparison

- Below is the summary of performance comparison of Training dataset

Training performance comparison:

	Bagging trained with Original-data	AdaBoost trained with Original-data	Gradient boosting trained with Original-data	Stacking Model trained with Original-data
Accuracy	0.997	0.753	0.751	0.747
Recall	1.000	0.874	0.875	0.841
Precision	0.996	0.782	0.780	0.793
F1	0.998	0.825	0.825	0.816

Figure 42: Training Dataset Performance Comparison

Validation Dataset Performance Comparison

- Below is the summary of performance comparison of Validation dataset

Validation performance comparison:

	Bagging trained with Original-data	AdaBoost trained with Original-data	Gradient boosting trained with Original-data	Stacking Model trained with Original-data
Accuracy	0.728	0.753	0.751	0.739
Recall	0.886	0.868	0.870	0.848
Precision	0.751	0.785	0.782	0.780
F1	0.813	0.825	0.823	0.813

Figure 43: Validation Dataset Performance Comparison

Final Model Selection

- Best Model: **AdaBoost** (Original dataset)
- **Reason** for Selection: -
  1. AdaBoost offers the best **balance between precision and recall** on both training and validation datasets, with an **F1 score of 0.825** on validation.
  2. It shows **no significant overfitting & generalizes well on unseen data**, as the training and validation performance metrics are almost identical. Our metric of interest, i.e. F1-score, is identical (0.825) in both training & validation datasets.
- **Strengths** of the Model: -
  1. **High Recall (0.868)** ensures that most denied applications are correctly identified, reducing the risk of incorrect visa approvals.
  2. **Good Precision (0.785)** ensures that when the model predicts a denial, it is correct most of the time, minimizing unnecessary rejections.
  3. **Good F1-score (0.825) indicates a well-balanced model** that maintains good performance across both classes (certified and denied) by striking a balance between minimizing False Positives & False Negatives.
  4. **No significant overfitting**, as indicated by similar training and validation performance metrics, thus, it generalizes well with unseen data & can be deployed to production.
- **High recall (0.868)** ensures that most denied applications are correctly identified, reducing the risk of incorrect visa approvals.
- To **Summarise: -**
  1. **AdaBoost is the best model** for predicting visa certification and denial, as it provides the most consistent and balanced performance across both training and validation datasets.
  2. While **Bagging** offers higher recall, its **severe overfitting** makes it less reliable for deployment.
  3. **Gradient Boosting** performs similarly to AdaBoost but **slightly underperforms in precision and F1 score**, making AdaBoost the preferred choice.

Test Dataset Performance of the Final Model

- Now that we have selected our final model, lets test the performance against the Test dataset (tantamount to unseen data): -

Test Performance Metrics - Adaboost | Original Data

	Accuracy	Recall	Precision	F1
0	0.741	0.871	0.771	0.818

Figure 44: Performance Metric Summary on Test Dataset - Final Model (Adaboost – Original dataset)

- Let's compare the Test dataset performance with Training dataset performance: -

Final Model Performance Comparison:

	AdaBoost Performance on Train dataset	AdaBoost Performance on Test dataset
Accuracy	0.753	0.741
Recall	0.874	0.871
Precision	0.782	0.771
F1	0.825	0.818

Figure 45: Final Model Performance Comparison - Training vs Test Dataset

- **Final Model Performance Summary**: -
  1. Accuracy: -
     - ✓ Training Accuracy: 0.753 | Test Accuracy: 0.741
     - ✓ The model achieves 75.3% accuracy on the training dataset and 74.1% accuracy on the test dataset.
     - ✓ The slight drop in accuracy from training to test indicates that the model generalizes well to unseen data without significant overfitting.
     - ✓ Accuracy, while useful, may not be the best metric in this context due to potential class imbalance (i.e., more visa certifications than denials).
  2. Recall: -
     - ✓ Training Accuracy: 0.874 | Test Accuracy: 0.871
     - ✓ The model achieves a recall of 87.4% on training and 87.1% on test, means that the model correctly identifies 87% of all actual denied applications
     - ✓ The minimal drop in recall between training and test datasets indicates that the model performs consistently and generalizes well.
  3. Precision: -
     - ✓ Training Accuracy: 0.782 | Test Accuracy: 0.771
     - ✓ Precision is 78.2% on training and 77.1% on test, means that when the model predicts a visa application as 'denied', it is correct 77% of the time
     - ✓ The slight decrease in precision from training to test again suggests good generalization without significant overfitting.
  4. **F1 Score (Metric of Interest!): -**
     - ✓ Training Accuracy: 0.825 | Test Accuracy: 0.818
     - ✓ The F1 score is 82.5% on training and 81.8% on test, reflecting a well-balanced model in terms of precision and recall.
     - ✓ The minimal difference (0.007) between training and test F1 scores indicates that the model is neither overfitting nor underfitting.
     - ✓ A high F1 score is particularly important in this context because it ensures that both false negatives (missed denials) and false positives (incorrect denials) are reasonably controlled i.e. the model maintains a good balance between correctly identifying denied cases and avoiding unnecessary denials of valid applications.
- To summarize, the **AdaBoost model demonstrates strong performance** in predicting visa certification and denial. Its high recall ensures that most denied applications are correctly identified, while its good precision minimizes false positives. The consistent performance across both training and test datasets shows that the **model generalizes well and is suitable for deployment in a real-world visa approval system.**

Rubric Question 8: Actionable Insights & Recommendations

- Before we get to actional insights & recommendations, lets delve into the Feature importances (Feature importance indicates how much each feature contributes to the model's predictive performance) of the final model: -

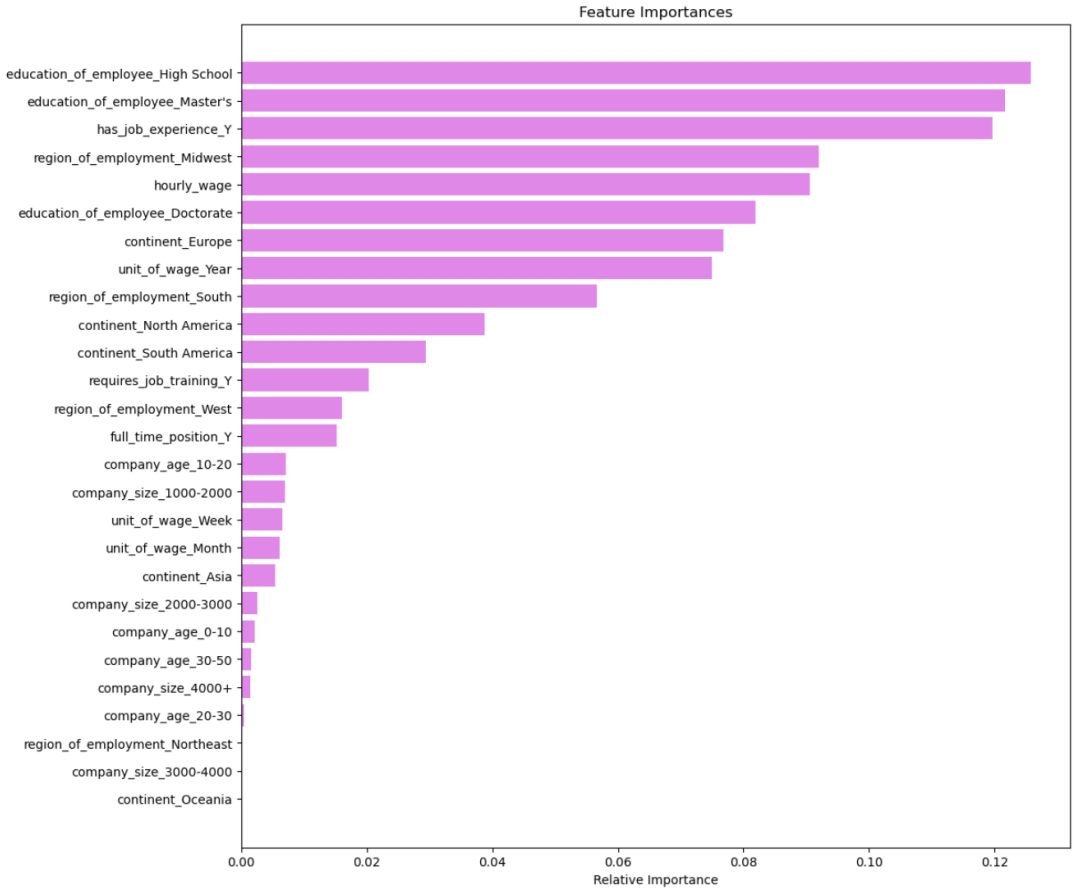


Figure 46: Feature Importances - Final Model

	Importance
education_of_employee_High School	0.126
education_of_employee_Master's	0.122
has_job_experience_Y	0.120
region_of_employment_Midwest	0.092
hourly_wage	0.091
education_of_employee_Doctorate	0.082
continent_Europe	0.077
unit_of_wage_Year	0.075
region_of_employment_South	0.057
continent_North America	0.039

Figure 47: Feature Importance Table (Top 10) - Final Model

- Let's analyse the Feature importances (top 10, i.e. significant ones) in context to the business problem: -

Features	Importance	Insight	Business Implication/Recommendation
Education of Employee – High School	~12.6%	✓ Employees with only a high school education may have a higher likelihood of denial because they may not meet the skill requirements for specialized roles requiring visas.	✓ Employers should prioritize candidates with higher educational qualifications when applying for visas to improve approval chances.
Education of Employee – Master's	~12.2%	✓ Employees with a master's degree may have better chances of approval, as they are likely considered highly skilled.	✓ Employers should highlight advanced degrees (Master's and above) in visa applications, emphasizing the specialized skills of the employee.
Has Job Experience	~12%	✓ Job experience significantly impacts visa certification decisions. Candidates with relevant work experience are more likely to have their visas approved.	✓ Employers should clearly document the candidate's prior work experience in similar roles, demonstrating the employee's qualifications and necessity for the job.
Region of Employment – Midwest	~9.2%	✓ The region where the job is located impacts the visa outcome. Midwest regions might have a higher demand for skilled labour, influencing visa approval.	✓ Employers in the Midwest should emphasize local labour shortages and justify the need for foreign workers to improve visa approval rates.
Hourly Wage	~9.1%	✓ Wage levels play a critical role in visa certification. Higher wages likely signal higher skill levels, leading to better chances of approval.	✓ Employers should ensure they offer competitive wages that meet or exceed industry standards and clearly document this in visa applications.
Education of Employee – Doctorate	~8.2%	✓ Similar to candidates with Master's degrees, those with Doctorate degrees are likely viewed favourably due to their specialized expertise.	✓ Employers hiring employees with a Doctorate should emphasize the high level of expertise required for the role.
Continent of Employment – Europe	~7.7%	✓ The continent where the job is located affects visa decisions. Jobs in Europe may have different visa criteria, influencing approval chances.	✓ Employers should clearly explain local demand and provide evidence for the necessity of hiring foreign workers in Europe.
Unit of Wage – Year	~7.5%	✓ The unit of wage payment affects the visa outcome. Annual wages may indicate full-time, long-term positions, improving approval chances.	✓ Employers should prefer presenting wages on an annual basis (rather than hourly or weekly) to signal long-term employment commitments.
Region of Employment – South	~5.7%	✓ Similar to the Midwest, employment in the southern region is a significant factor.	✓ Employers in the South should justify the need for foreign workers by emphasizing the shortage of local talent.
Continent of Employment – North America	~3.9%	✓ Employment in North America has a moderate impact on visa decisions	✓ Employers in North America should follow standard practices, such as offering competitive wages and documenting job-specific skills, to improve visa approval chances.

Table 5: Insights & Recommendations against Important Features (Top 10)

- **General Actionable Insights: -**
  1. **Emphasize Skill and Qualification:**
     - Since education and job experience are highly important, employers should prioritize hiring candidates with advanced degrees (Master's or Doctorate) and relevant experience.
     - Properly document these qualifications in visa applications to strengthen the case.
  2. **Offer Competitive Compensation:**
     - Wage-related features, such as hourly wage and unit of wage (annual), are significant. Offering wages that are competitive and aligned with industry standards can improve the likelihood of visa certification.
     - Present wages on an annual basis to indicate long-term employment and stability.

3. **Region-Specific Strategies:**
   - The region of employment matters, with Midwest and South regions showing higher importance. Employers in these regions should justify the need for foreign talent by providing evidence of local labour shortages.
   - Tailor visa applications to highlight regional economic conditions and job market demand.
4. **Job Experience is Key**:
   - Candidates with prior relevant job experience have a better chance of approval. Employers should highlight the candidate's experience, especially if it matches the job role and industry.
5. **Avoid Applying for Lower-Skilled Positions:**
   - Since education at the high school level negatively influences visa outcomes, employers should avoid applying for visas for positions that can be filled by lower-skilled or entry-level employees. Focus on roles that genuinely require specialized skills.
6. **Overbooking Strategy**: Leverage the model's predictions to safely overbook rooms based on the likelihood of cancellations.

- **Recommendations for Employers**: -
  1. **Focus on High-Skilled Roles**: Apply for visas for positions requiring high skills, advanced education, and relevant experience.
  2. **Provide Detailed Justifications**: Clearly explain why the role cannot be filled by local talent and why the specific candidate is uniquely qualified for the job.
  3. **Highlight Competitive Wages**: Ensure that offered wages meet or exceed prevailing wage standards for the role and region, and present wages on an annual basis where possible.
  4. **Region-Specific Applications**: Tailor applications based on the region of employment. In high-demand regions like the Midwest and South, emphasize local labour shortages and the specialized skills of the candidate.
- **To conclude**, the feature importance analysis shows that **education level**, **job experience**, **wage**, and **region of employment** are critical factors influencing visa approval outcomes. Employers should focus on hiring highly educated and experienced candidates, offer competitive compensation, and provide region-specific justifications in visa applications. By addressing these factors, employers can significantly improve their chances of visa certification.