# Machine Learning -2
(coded project)

by

Purnima MS

**Page of Contents:**

# 1.Exploratory Data Analysis:

## 1.1 Problem definition

*The OFLC processes numerous visa applications every year to hire foreign workers in the U.S.*
*Given the surge in applications, manually reviewing every case is cumbersome and time-consuming. Our goal is to build a machine learning model that predicts visa certification status.*
*This will help prioritize applications with a higher likelihood of approval.*
*By leveraging data attributes like education, job experience, wages, etc., we aim to identify key drivers of certification.*
*The data contains the different attributes of visa's certification details. The detailed data dictionary is given below fig.1.1.1*

| **Features Names** | **Description** |
|---|---|
| case_id | Unique ID of each visa application |
| continent | Continent of the employee |
| education_of_employee | Education level of the employee |
| has_job_experience | Whether the employee has job experience (Y/N) |
| requires_job_training | Whether the employee requires job training (Y/N) |
| no_of_employees | Number of employees in the employer's company |
| yr_of_estab | Year the employer's company was established |
| region_of_employment | Intended U.S. employment region of the foreign worker |
| prevailing_wage | Average wage paid to similar workers in the employment area |
| unit_of_wage | Unit of wage measurement (Hourly, Weekly, Monthly, Yearly) |
| full_time_position | Whether the position is full-time (Y/N) |
| case_status | Target variable: Whether the visa was certified or denied (0/1) |

*fig1.1.1*

### Libraries

*Utilized Google Colab as the development environment and incorporated 4 key libraries to facilitate data analysis and visualization. These libraries are:*

1. *NumPy - For Numerical computing*
2. *Pandas - For Data Manipulation*
3. *Seaborn - Visualization library for statistical plotting*
4. *Matplotlib - Visualization library for statistical plotting*
5. *Sklearn.model_selection (train_test_split) - Split the data into train and test*
6. *Statsmodels.api - To build model for prediction*
7. *Sklearn.metrics (f1_score, accuracy_score, recall_score, precision_score, confusion_matrix, roc_auc_score, precision_recall_curve, roc_curve, make_scorer) - To check model performance*
8. *from sklearn. import*
9. *from sklearn.tree import DecisionTreeClassifier,Randomforestclassifier*
10. *from sklearn.model_selection import GridSearchCV- To tune different model*
11. *from imblearn.over_sampling import SMOTE*
12. *from imblearn.under_sampling import RandomUnderSampler*
    *To do hyperparameter tuning*
13. *from sklearn.model_selection import RandomizedSearchCVs*
14. *from sklearn.ensemble import (AdaBoostClassifier, GradientBoostingClassifier,RandomForestClassifier, BaggingClassifier)*
15. *from xgboost import XGBClassifier*

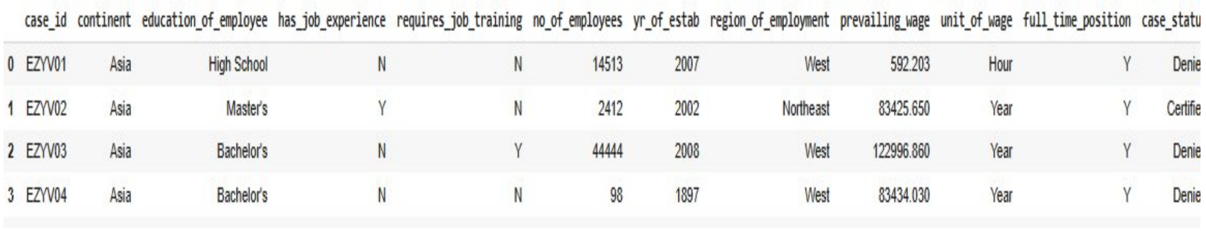
### Load the data

*Mounted Google Drive in the Colab environment and successfully incorporated it into my workflow. After transferring my dataset to a specific Google Drive folder, I read* ***'EasyVisa.csv'*** *file directly and loaded its contents into Pandas DataFrame.*

### Structure of the data

*To gain an initial understanding of the dataset's structure,*

- *Utilized the data.head () function to preview the first 5 rows of data seen in fig1.1.2. This allowed me to inspect the format, column names, and sample values within the dataset.*



| | case_id | continent | education_of_employee | has_job_experience | requires_job_training | no_of_employees | yr_of_estab | region_of_employment | prevailing_wage | unit_of_wage | full_time_position | case_statu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | EZYV01 | Asia | High School | N | N | 14513 | 2007 | West | 592.203 | Hour | Y | Denie |
| 1 | EZYV02 | Asia | Master's | Y | N | 2412 | 2002 | Northeast | 83425.650 | Year | Y | Certifie |
| 2 | EZYV03 | Asia | Bachelor's | N | Y | 44444 | 2008 | West | 122996.860 | Year | Y | Denie |
| 3 | EZYV04 | Asia | Bachelor's | N | N | 98 | 1897 | West | 83434.030 | Year | Y | Denie |

*fig1.1.2*

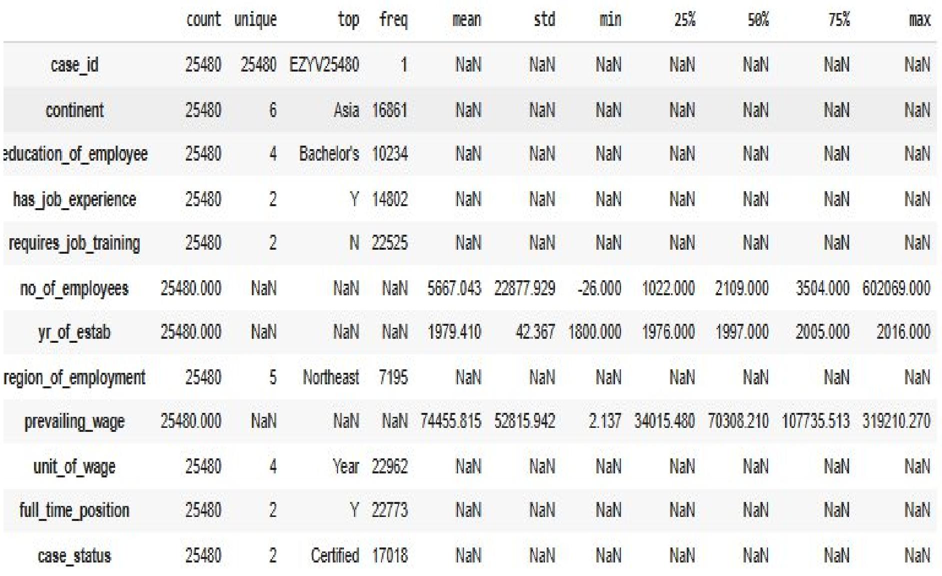
### Types of the data

Used the data.info() function for valuable insights of the dataset such as dimensions, column names, data types, and the presence of missing values. The dataset consists of *25480* entries across 12 columns with the following characteristics: ***Data Types****: The dataset includes* ***3 columns with float and int64*** *datatype, and* ***8 columns with object*** *datatype (categorical/text data).*

- ***Non-Null Counts****: All columns contain 25480 non-null entries, indicating no missing data.*
- ***The target variable****: case_status is the target variable which is of categorical type***.**

**Statistical Summary**
*To gain an initial understanding of the dataset's structure shows in fig 1.1.3,*

- ***no_of_employees****: Large variability in employer sizes, with some negative values possibly due to data errors.*
- ***yr_of_estab****: Companies span from 1800 to 2016, showing a wide range of employer establishment dates.*
- ***region_of_employment****: Northeast is the most common employment region, indicating regional employment trends.*
- ***prevailing_wage****: The wages range from 2 to over 300,000, suggesting significant variation in offered wages across jobs.*
- ***unit_of_wage****: Most records use yearly wage units, making it the dominant scale for analysis.*



| | count | unique | top | freq | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|---|---|---|
| case_id | 25480 | 25480 | EZYV25480 | 1 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| continent | 25480 | 6 | Asia | 16861 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| ducation_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 |
| no_of_employees | 25480.000 | NaN | NaN | NaN | 5667.043 | 22877.929 | -26.000 | 1022.000 | 2109.000 | 3504.000 | 602069.000 |
| yr_of_estab | 25480.000 | NaN | NaN | NaN | 1979.410 | 42.367 | 1800.000 | 1976.000 | 1997.000 | 2005.000 | 2016.000 |
| region_of_employment | 25480 | 5 | Northeast | 7195 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| prevailing_wage | 25480.000 | NaN | NaN | NaN | 74455.815 | 52815.942 | 2.137 | 34015.480 | 70308.210 | 107735.513 | 319210.270 |
| 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 |

*fig 1.1.3*

## 1.2 Univariate analysis

### Numerical variable:

***no_of_employees****-fig1.2.1*

- *The distribution is highly right-skewed, with most values around very low counts.*
- *There are a few extremely large values (outliers), causing a long tail up to 600,000.*
- *Most companies have fewer than 10,000 employees, suggesting many small businesses.*

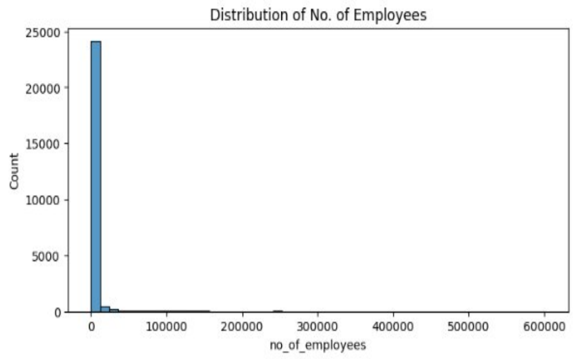


*fig1.2.1*

***prevailing_wage****- fig1.2.2*

- *The wage distribution is moderately right-skewed with a large spike at very low wages.*
- *Most wages lie below $100,000, but there's a long tail stretching to over $300,000.*
- *The presence of a spike at low wages may suggest reporting errors or lower-paid positions.*

***yr_of_estab****- fig1.2.3*

- *The establishment year data shows a concentration of records for more recent years.*
- *There's a sharp increase in establishments after 1950, peaking around 2000.*
- *Older establishment years (1800–1900) are rare and may be data entry artifacts.*

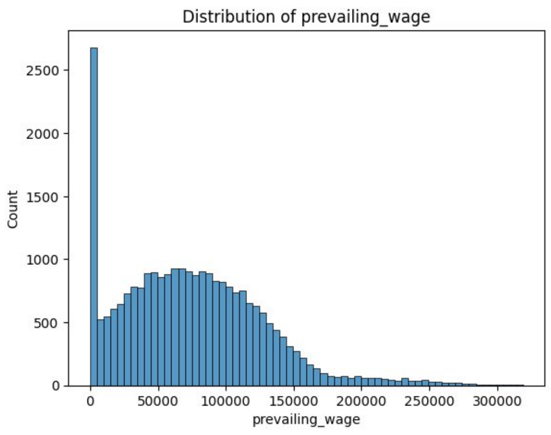


fig1.2.2

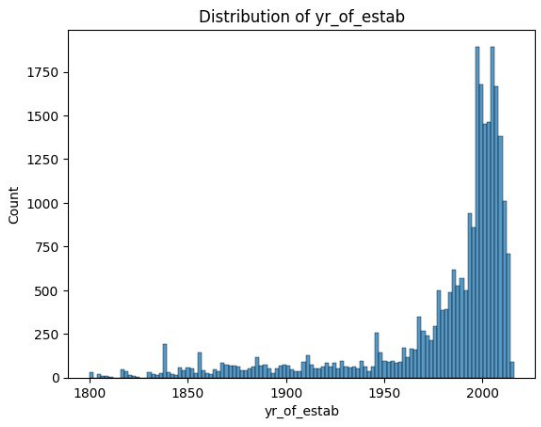


fig 1.2.3

**Categorical variables:**

**full_time_position:** *-fig 1.2.4*

- *A large majority of the positions are full-time, with over 22,000 entries marked as Y.*
- *There's a smaller group (~3,000) marked as N, indicating part-time or non-full-time positions.*
- *The dominance of full-time positions suggests that full-time roles are the standard or preferred employment type in this dataset.*

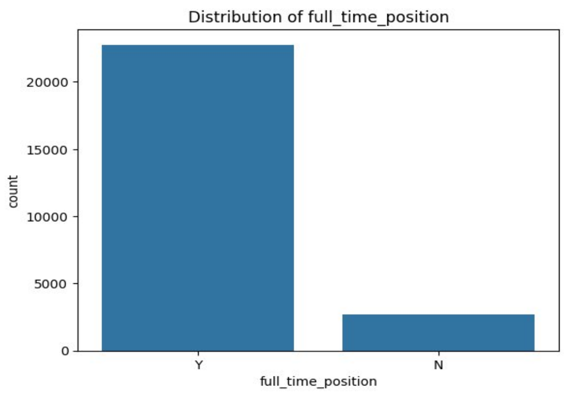


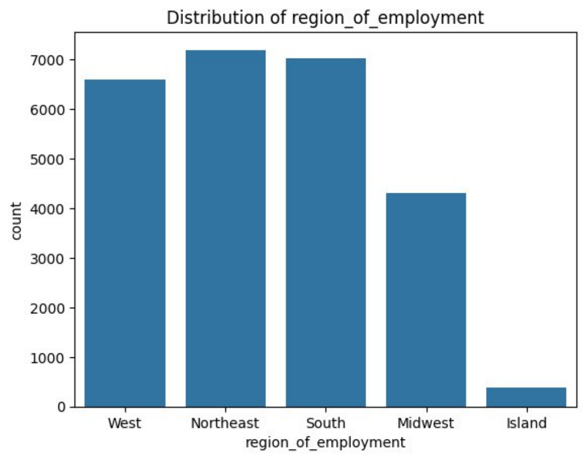
*fig 1.2.4*

***region_of_employment****-fig1.2.5*

- *Northeast and South regions have the highest employment counts, both over 7000.*
- *The Island region shows significantly lower employment, under 500.*
- *Employment distribution is uneven, with a stark contrast between major regions and the Island.*

***Continent****-fig 1.2.6*

- *Asia dominates the dataset with the highest count of entries (~16,861), indicating a significant representation.*
- *Europe and North America have moderate and comparable frequencies, suggesting a balanced representation.*
- *Africa, Oceania, and South America show minimal counts, highlighting underrepresentation in the dataset.*



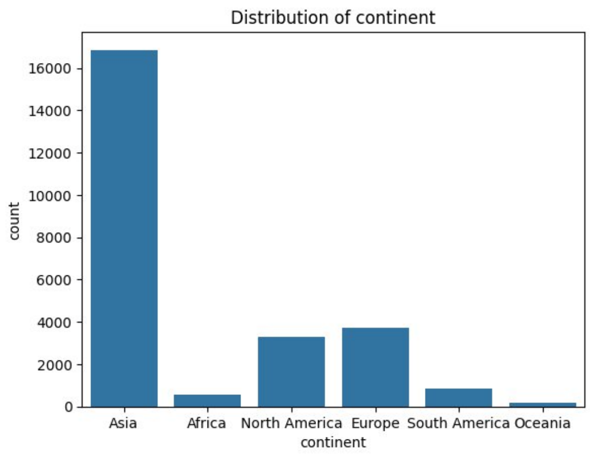


*fig1.2.5*

*fig 1.2.6*

**1.3 Bivariate analysis**
**Relationship between all variables**

- *In fig 1.3.1 Most companies have low employee counts, were founded more recently, and offer lower wages (right-skewed distributions).*
- *The no_of_employees and prevailing_wage variables are heavily right-skewed, indicating few companies with very high values.*
- *No strong linear relationships are observed between these variables, but newer companies (yr_of_estab) may offer slightly higher wages.*
- *Histograms along the diagonal of the pair plots confirm the skewness and distribution of each variable, highlighting their concentration in the lower ranges.*

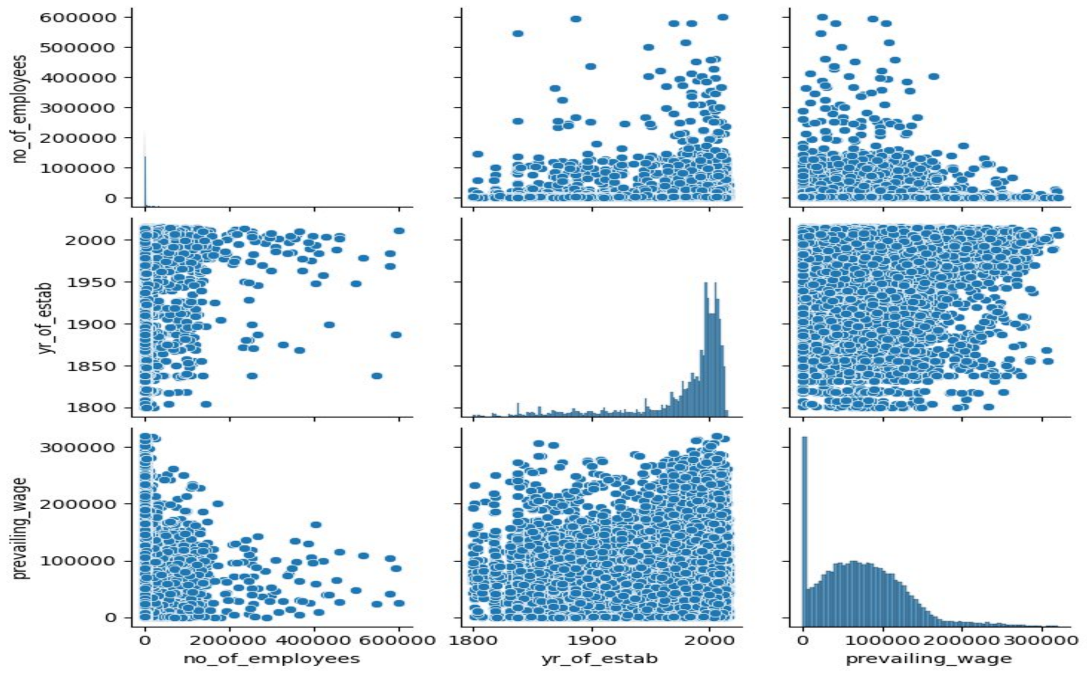
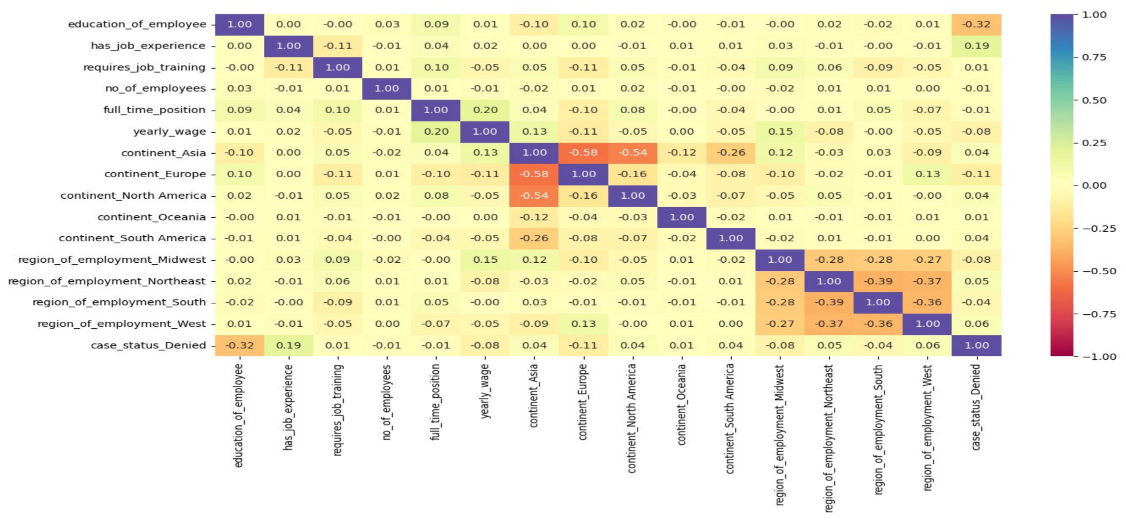


*fig 1.3.1*

**Top of Form**
**Bottom of Form**

**Correlation between all variables**

- *In fig1.3.2, Most variables show weak correlations (near zero), indicating little linear relationship between pairs.*
- *Education of employees is negatively correlated with case status denied (-0.32), suggesting higher education may reduce the chance of denial.*



| | education_of_employee | has_job_experience | requires_job_training | no_of_employees | full_time_position | yearly_wage | continent_Asia | continent_Europe | continent_North America | continent_Oceania | continent_South America | region_of_employment_Midwest | region_of_employment_Northeast | region_of_employment_South | region_of_employment_West | case_status_Denied |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| education_of_employee | 1.00 | 0.00 | -0.00 | 0.03 | 0.09 | 0.01 | -0.10 | 0.10 | 0.02 | -0.00 | -0.01 | -0.00 | 0.02 | -0.02 | 0.01 | -0.32 |
| has_job_experience | 0.00 | 1.00 | -0.11 | -0.01 | 0.04 | 0.02 | 0.00 | 0.00 | -0.01 | 0.01 | 0.01 | 0.03 | -0.01 | -0.00 | -0.01 | 0.19 |
| requires_job_training | -0.00 | -0.11 | 1.00 | 0.01 | 0.10 | -0.05 | 0.05 | -0.11 | 0.05 | -0.01 | -0.04 | 0.09 | 0.06 | -0.09 | -0.05 | 0.01 |
| no_of_employees | 0.03 | -0.01 | 0.01 | 1.00 | 0.01 | -0.01 | -0.02 | 0.01 | 0.02 | -0.01 | -0.00 | -0.02 | 0.01 | 0.01 | 0.00 | -0.01 |
| full_time_position | 0.09 | 0.04 | 0.10 | 0.01 | 1.00 | 0.20 | 0.04 | -0.10 | 0.08 | -0.00 | -0.04 | -0.00 | 0.01 | 0.05 | -0.07 | -0.01 |
| yearly_wage | 0.01 | 0.02 | -0.05 | -0.01 | 0.20 | 1.00 | 0.13 | -0.11 | -0.05 | 0.00 | -0.05 | 0.15 | -0.08 | -0.00 | -0.05 | -0.08 |
| continent_Asia | -0.10 | 0.00 | 0.05 | -0.02 | 0.04 | 0.13 | 1.00 | -0.58 | -0.54 | -0.12 | -0.26 | 0.12 | -0.03 | 0.03 | -0.09 | 0.04 |
| continent_Europe | 0.10 | 0.00 | -0.11 | 0.01 | -0.10 | -0.11 | -0.58 | 1.00 | -0.16 | -0.04 | -0.08 | -0.10 | -0.02 | -0.01 | 0.13 | -0.11 |
| continent_North America | 0.02 | -0.01 | 0.05 | 0.02 | 0.08 | -0.05 | -0.54 | -0.16 | 1.00 | -0.03 | -0.07 | -0.05 | 0.05 | -0.01 | -0.00 | 0.04 |
| continent_Oceania | -0.00 | 0.01 | -0.01 | -0.01 | -0.00 | 0.00 | -0.12 | -0.04 | -0.03 | 1.00 | -0.02 | 0.01 | -0.01 | -0.01 | 0.01 | 0.01 |
| continent_South America | -0.01 | 0.01 | -0.04 | -0.00 | -0.04 | -0.05 | -0.26 | -0.08 | -0.07 | -0.02 | 1.00 | -0.02 | 0.01 | -0.01 | 0.00 | 0.04 |
| region_of_employment_Midwest | -0.00 | 0.03 | 0.09 | -0.02 | -0.00 | 0.15 | 0.12 | -0.10 | -0.05 | 0.01 | -0.02 | 1.00 | -0.28 | -0.28 | -0.27 | -0.08 |
| region_of_employment_Northeast | 0.02 | -0.01 | 0.06 | 0.01 | 0.01 | -0.08 | -0.03 | -0.02 | 0.05 | -0.01 | 0.01 | -0.28 | 1.00 | -0.39 | -0.37 | 0.05 |
| region_of_employment_South | -0.02 | -0.00 | -0.09 | 0.01 | 0.05 | -0.00 | 0.03 | -0.01 | -0.01 | -0.01 | -0.01 | -0.28 | -0.39 | 1.00 | -0.36 | -0.04 |
| region_of_employment_West | 0.01 | -0.01 | -0.05 | 0.00 | -0.07 | -0.05 | -0.09 | 0.13 | -0.00 | 0.01 | 0.00 | -0.27 | -0.37 | -0.36 | 1.00 | 0.06 |
| case_status_Denied | -0.32 | 0.19 | 0.01 | -0.01 | -0.01 | -0.08 | 0.04 | -0.11 | 0.04 | 0.01 | 0.04 | -0.08 | 0.05 | -0.04 | 0.06 | 1.00 |

*fig1.3.2*

- *Yearly wage and full-time position have a slight positive correlation (0.20), hinting that full-time roles tend to offer higher pay.*

**1.5 Observations and Insights**

- *Most companies are small-sized with low-to-moderate wages, though some outliers exist with high wages and large employee counts.*
- *Companies founded after 1950 show a sharp increase, peaking in the 2000s.*
- *Certified applications consistently outnumber denied ones, with the South and West regions leading in volume.*
- *Asia dominates in application volume, and full-time positions are significantly more common than part-time roles.*

- *Despite some mild correlations (like higher wages for full-time positions), most features are largely independent.*
- *Visualizations highlight patterns by region and show no strong relationships between numeric features.*

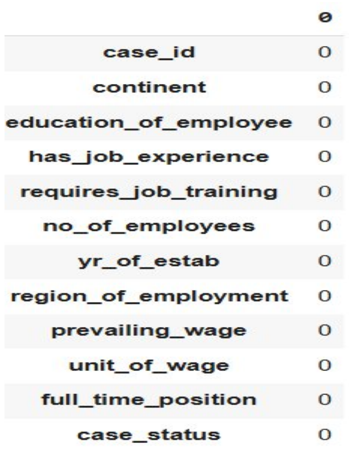
## 2.Data preprocessing:

### 2.1 Duplicate value check

- *Checked by using duplicated () and sum of those result it returns no duplicate values in the data and the output is **np.int64(0)***

### 2.2 Missing value treatment

- *Checked by using isnull () and sum of those result it returns no missing values in the data and output is fig2.2.1*



| | 0 |
|---|---|
| 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 |

*fig2.2.1*

### 2.3 Outlier treatment

- *In fig 2.3.1 box plots of no_of_employees feature have most companies have very few employees, but some have an extremely large number (outliers).*
- *yr_of_esta feature has most establishments were founded in the late 20th to early 21st century, with some much older or newer outliers.*

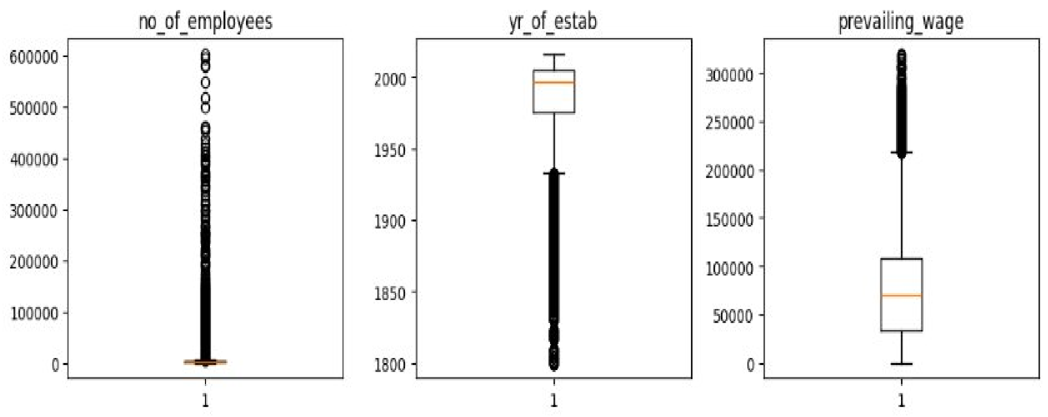


*fig 2.3.1*

- *prevailing_wage feature has most wages are in the lower to mid-range, but some are significantly higher (outliers).*

## 2.4 Feature engineering

- *Dropped the column case_id as it is an insignificant feature in dataset.*
- *Maps categorical string values to numerical representations for several features. "High School" to "Doctorate" are mapped to 1-4 for education_of_employee. For has_job_experience, requires_job_training, and full_time_position, "Y" is mapped to 0 and "N" to 1, effectively transforming these binary categorical features into numerical ones.*
- *Retain only entries where the 'no_of_employees' is a non-negative value. This action effectively removes any rows containing invalid or negative employee counts.*
- *Standardizes prevailing_wage to a yearly_wage by applying multipliers based on the original unit_of_wage.*
- *After this calculation, it removes the original unit_of_wage, prevailing_wage, and yr_of_estab columns from the dataset.*
- *These transformation ensures all wages are comparable on an annual basis and streamlines the dataset by dropping now redundant or specified columns.*

## 2.5 Data preparation for modeling

- *X (Independent variables): These are all the columns (features) that you will use to predict the target. You dropped the column 'case_status" because it's the one you're trying to predict.*
- *y (Dependent variable): This is the column you want to predict, which is 'case_status"*
- *Converted categorical variables into numeric format using one-hot encoding (dummy variables).*
- *Used drop_first=True to avoid the dummy variable trap (multicollinearity).*
- *Ensured all features are of float type for compatibility with modeling algorithms.*
- *Splitting data into training and validation into 65:25, and test set into 80:20*
- *Splitting the data in 70:20 ratio for train to test data.*

## 3. Model building -Original data

### 3.1 Choose the appropriate metric for model evaluation.

***False Positive (Type I Error):***
***Prediction**: Visa will be certified*
***Reality**: Visa is denied*
***Consequence**: The applicant is falsely shortlisted, possibly leading to wasted administrative effort and missed opportunities for genuinely eligible candidates.*

***False Negative (Type II Error):***
***Prediction**: Visa will be denied*
***Reality**: Visa is actually certified*
***Consequence**: A genuinely eligible applicant is overlooked, resulting in lost talent and a negative impact on workforce planning.*

***Which one is more expensive?***

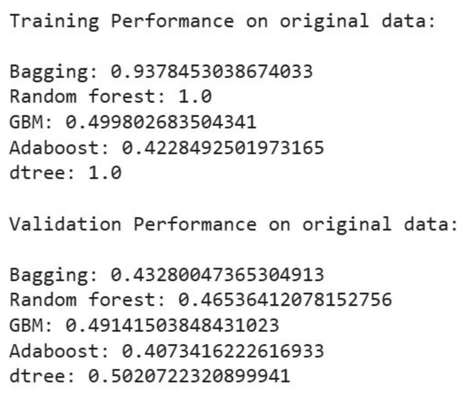
***False Negatives (Type II errors)** are generally more costly.*
*Qualified applicants are missed, reducing workforce competitiveness and causing operational disruptions for employers who lose out on needed talent.*

***How to reduce false negatives?***

***Greater the Recall, lower the false negatives***
*Focus on improving Recall (or Sensitivity) in the classification model to ensure that certified visas are correctly identified.*

***3.2 Build models on original data**.*

- *In fig 3.2.1, Random Forest and Decision Tree models exhibit perfect training precision (1.0), indicating severe overfitting to the training data.*
- *Bagging also shows extremely high training precision (0.93), suggesting it's likely overfitting as well, though slightly less than the individual tree and random forest.*
- *In contrast, GBM and Adaboost have much lower training precision (around 0.42), which points to underfitting or a more generalized learning approach on the training set.*
- *The large drop in precision from training to validation for Bagging, Random Forest, and Decision Tree confirms their significant overfitting, while GBM and Adaboost maintain more consistent (though lower) performance.*



```
Training Performance on original data:

Bagging: 0.9378453038674033
Random forest: 1.0
GBM: 0.499802683504341
Adaboost: 0.4228492501973165
dtree: 1.0

Validation Performance on original data:

Bagging: 0.43280047365304913
Random forest: 0.46536412078152756
GBM: 0.49141503848431023
Adaboost: 0.4073416222616933
dtree: 0.5020722320899941
```

*fig 3.2.1*

***Comment on the model performance:***

- ***Models with high training but low validation performance (like Decision Trees, Random Forest, and Bagging)*** *are likely memorizing the historical data, but they fail to predict new visa applications accurately.*
- ***GBM's closer training-validation performance*** *suggests it might be more robust for real-world prediction, despite the moderate accuracy.*
- ***AdaBoost's low scores*** *suggest it may not be the best model choice here.*

## 4. Model Building - Oversampled Data

### 4.1 Oversample the train data

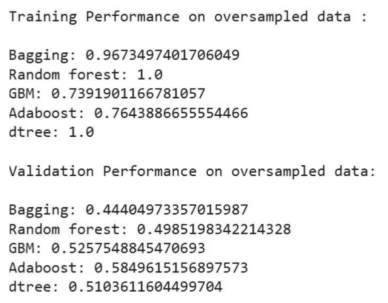
We see clear overfitting and data imbalance issues (visa-certified vs denied cases), using resampling techniques to balance the classes and reduce bias

**SMOTE (Synthetic Minority Over-sampling Technique)**
**SMOTE generates synthetic data points for the minority class (e.g., "denied" cases if they're rare). This helps the model learn the decision boundaries better and not favor the majority class.**

- *Identify the minority class.*
- *Create synthetic examples by interpolating between actual minority samples.*
- *Combine these synthetic samples with the original dataset and retrain the model.*

### 4.2 Build models on oversampled data

- *In fig 4.2.1, Random Forest and Decision Tree still show perfect training precision (1.0), indicating they continue to severely overfit the oversampled training data.*
- *Bagging also maintains very high training precision (0.96), suggesting significant overfitting, similar to the original data.*



```
Training Performance on oversampled data :

Bagging: 0.9673497401706049
Random forest: 1.0
GBM: 0.7391901166781057
Adaboost: 0.7643886655554466
dtree: 1.0

Validation Performance on oversampled data:

Bagging: 0.44404973357015987
Random forest: 0.4985198342214328
GBM: 0.5257548845470693
Adaboost: 0.5849615156897573
dtree: 0.5103611604499704
```

*fig 4.2.1*

- *GBM and Adaboost's training precision improved (to 0.73 and 0.76 respectively) compared to the original data, indicating they are now learning more from the training set due to oversampling.*
- *Despite the training improvements for some models, all models still show a substantial drop in recall from training to validation, confirming persistent overfitting issues on the oversampled dataset, with GBM and Adaboost showing the least drop among them*

### *4.3* Model performance across oversampled data models

- *Across the oversampled data models, Adaboost shows the best generalization performance with the highest validation recall value(0.76), indicating it handles the new data better.*
- *Conversely, Decision Tree, Random Forest, and Bagging exhibit significant overfitting, achieving near-perfect training precision but dropping drastically on validation*

## 5. Model Building - Undersampled Data

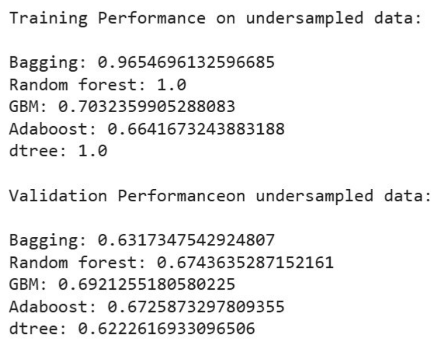
### 5.1 Undersample the train data

***Reduces the size of the majority class by removing some of its samples.***

- *Identify the majority class.*
- *Randomly remove samples to match the size of the minority class.*
- *Retrain the model on the balanced data.*
- *Helps the model avoid bias towards the majority class, although it may reduce overall data diversity.*

**5.2 Build models**

- *In fig 5.2.1 Random Forest and Decision Tree again show perfect training recall(1.0), indicating extreme overfitting to the undersampled training data.*
- *Bagging also exhibits very high training precision (0.96), suggesting strong overfitting tendencies similar to the other tree-based models.*
- *GBM achieves the highest validation recall(0.70) among all models, followed by Adaboost (0.66), suggesting they generalize slightly better than the other techniques on this undersampled dataset.*



```
Training Performance on undersampled data:

Bagging: 0.9654696132596685
Random forest: 1.0
GBM: 0.7032359905288083
Adaboost: 0.6641673243883188
dtree: 1.0

Validation Performanceon undersampled data:

Bagging: 0.6317347542924807
Random forest: 0.6743635287152161
GBM: 0.6921255180580225
Adaboost: 0.6725873297809355
dtree: 0.6222616933096506
```

*fig 5.2.1*

**5.3 Model performance across different undersampled data models:**

- *All models, particularly Decision Tree, Random Forest, and Bagging, exhibit substantial overfitting, with near-perfect training precision but significantly lower validation recall.*
- *GBM performs best in generalization: Gradient Boosting Machine (GBM) achieves the highest validation recall (0.69), indicating it generalizes comparatively better to unseen data.*
- *Adaboost is a close second and also shows relatively better performance with the second-highest validation recall (0.67).*
- *Tree-based models struggle: Individual Decision Tree, Random Forest, and Bagging models show the poorest generalization with validation recall.*

**6. Model Performance Improvement using Hyperparameter Tuning**

**6.1 Choose 3 models after tuning with proper reasoning**

Based on random undersampling and Oversampling results, the following **three models** for final consideration for hyperparamter tunning, along with reasoning:

- **Gradient Boosting Algorithm on Oversampled data**
  *GBM to the original data, indicating they are now learning more from the training set due to oversampling.*
- **Gradient Boosting Algorithm on Undersampled data**
- *GBM performs best in generalization: Gradient Boosting Machine (GBM) achieves the highest validation recall, indicating it generalizes comparatively better to unseen data.*
- **Adaboost Algorithm on Undersampled data**
  *Adaboost is a close second and also shows relatively better performance with the second-highest validation recall.*

## 6.2 The performance of Tuned models

**Training and validation performance comparision-***fig6.2.2 & fig6.2.3*

*Gradient Boosting Tuned with Hyperparameter (Original data)*

- *Training Recall: 0.535*
- *Validation Recall: 0.509*

  ***Lower recall compared to other techniques.***

*Gradient Boosting Tuned with Hyperparameter (Undersampled data)*

- *Training Recall: 0.717*
- *Validation Recall: 0.710*

  ***Significant improvement in recall due to undersampling.***

*Gradient Boosting Tuned with Hyperparameter (Oversampled data)*

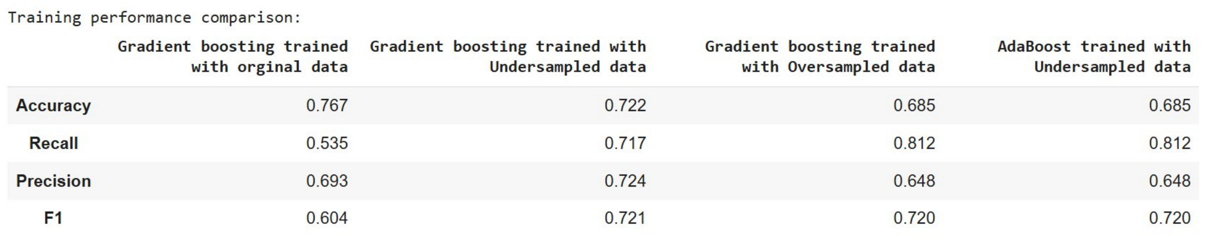
- *Training Recall: 0.812*
- *Validation Recall: 0.730*

  ***Highest recall overall, suggesting oversampling helps the model to better capture the minority class.***

*AdaBoost Tuned with Hyperparameter (Undersampled data)*

- *Training Recall: 0.812*
- *Validation Recall: 0.730*

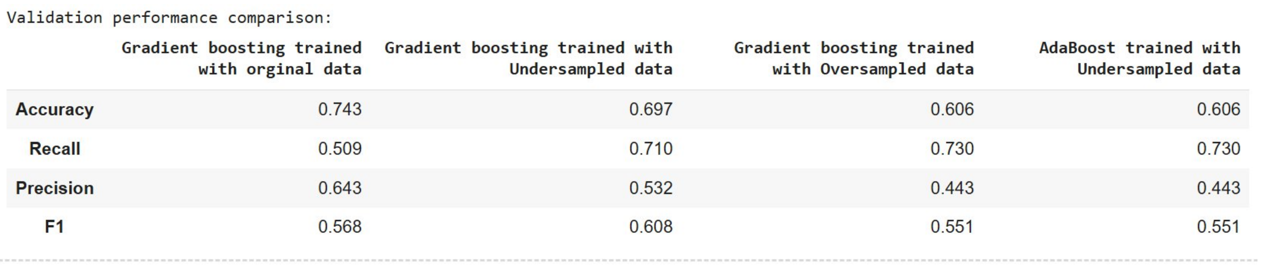
  ***Same as oversampled Gradient Boosting, indicating strong recall performance with undersampling.***



```
Training performance comparison:
```

| | Gradient boosting trained with orginal data | Gradient boosting trained with Undersampled data | Gradient boosting trained with Oversampled data | AdaBoost trained with Undersampled data |
|---|---|---|---|---|
| **Accuracy** | 0.767 | 0.722 | 0.685 | 0.685 |
| **Recall** | 0.535 | 0.717 | 0.812 | 0.812 |
| **Precision** | 0.693 | 0.724 | 0.648 | 0.648 |
| **F1** | 0.604 | 0.721 | 0.720 | 0.720 |

*fig6.2.2*



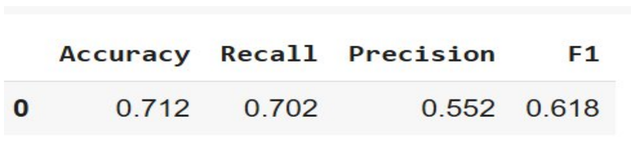
```
Validation performance comparison:
```

| | Gradient boosting trained with orginal data | Gradient boosting trained with Undersampled data | Gradient boosting trained with Oversampled data | AdaBoost trained with Undersampled data |
|---|---|---|---|---|
| **Accuracy** | 0.743 | 0.697 | 0.606 | 0.606 |
| **Recall** | 0.509 | 0.710 | 0.730 | 0.730 |
| **Precision** | 0.643 | 0.532 | 0.443 | 0.443 |
| **F1** | 0.568 | 0.608 | 0.551 | 0.551 |

*fig6.2.3*

***Test Set Performance*** *(Gradient Boosting Tuned with Hyperparameter (Undersampled data)*

- In fig6.2.4, *Recall on the test set is* ***0.702****, which is consistent with the* **validation recall (0.710)** *observed earlier for Gradient Boosting with undersampled data.*
- *This confirms that undersampling continues to improve recall even on the test set, showing the model's good generalization for minority class detection.*



| | Accuracy | Recall | Precision | F1 |
|---|---|---|---|---|
| **0** | 0.712 | 0.702 | 0.552 | 0.618 |

*fig6.2.4*

**7. Actionable Insights & Recommendations**

7.1 Insights from the analysis conducted

- *Gradient Boosting (Original) gives better precision and accuracy—good for minimizing false positives.*
- *On validation, undersampling with Gradient Boosting is a reasonable trade-off, achieving higher recall and a good F1*

- *Original data gives the lowest recall (~0.5).*
- *Undersampling boosts recall substantially (~0.7 in validation).*
- *Oversampling pushes recall even higher (~0.73 in validation).*
- *AdaBoost (with undersampling) achieves recall similar to oversampling, showing that undersampling + AdaBoost is highly effective for recall.*
- **Feature importance-***fig 7.1.1* - The analysis of visa application data reveals that education_of_employee, yearly_wage, has_job_experience, and are the most significant factors influencing visa outcomes.

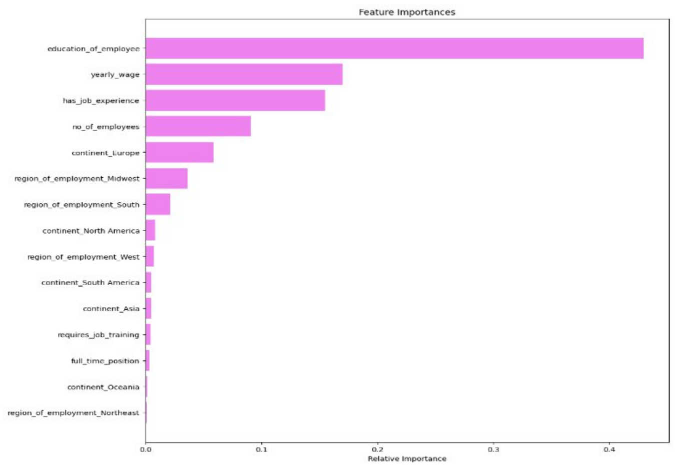


*fig 7.1.1*

**7.2 Actionable business recommendations**

- *Use the model to quickly sort visa applications for faster processing ("likely approved" vs. "needs review").*
- *Because the model has good recall (~0.7), use it for things like fraud or risk detection where don't want to miss any real case*
- *The precision is not very high (~0.55), so you'll get some false positives. Figure out if these false positives are okay for your business or if they cause problems.).*
- *Continuously enhance data quality to ensure the model's accuracy and reliability.*
- *Regularly track model performance and retrain it with new data to maintain its effectiveness.*
- *Use model insights to identify key approval drivers and refine visa application guidelines for better quality submissions.*