H-1B Visa Labor Condition Application Approval Prediction

Using Classification Models

Team CD-10

Team Members

Juilee Thakur
MS in Data Science
Mumbai

Akshat Jain
MS in Data Science
Surat

Jaykumar Patel
MS in Data Science
Ahmedabad

Prathmesh Desai
MS in Information Systems
Mumbai

Introduction

- Labor Conditional Application, or LCA, is a required document that before filing the H-1B application with USCIS for whichever non-immigrant worker, Employers must apply with the US Department of Labor.
- LCA is essential to ensuring that you are paid a fair wage as a foreign employee and are not exploited by US businesses.
- Among the first stages to obtaining an H1B work permit in the US is having an LCA approved.
- An H1B Labor Condition Application (LCA) form contains all the pertinent details about the position being offered to the foreign worker, including the wage and location information.

Dataset

- We have captured a dataset from Data World that contains various observations about H-1B LCA applications filed in the year 2017.
- The dataset contains ~528000 observations and 40 columns before any data preprocessing.
- The attributes are a combination of both numerical data and categorical information.
- The main objective of our analysis identify meaningful relationships between the final status of the VISA and the candidate’s details.

Dataset Information

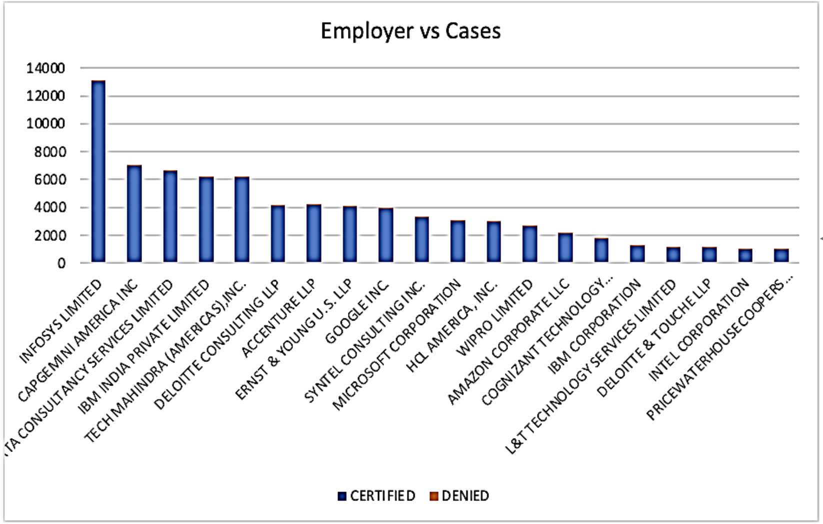
FIELD NAME	DESCRIPTION
CASE_NUMBER	Unique identifier assigned to each application submitted for processing to the Chicago National Processing Center.
CASE_STATUS	Status associated with the last significant event or decision. Valid values include “Certified,” “Certified-Withdrawn,” Denied,” and “Withdrawn”.
CASE_SUBMITTED	Date and time the application was submitted.
DECISION_DATE	Date on which the last significant event or decision was recorded by the Chicago National Processing Center.
VISA_CLASS	Indicates the type of temporary application submitted for processing. R = H-1B; A = E-3 Australian; C = H-1B1 Chile; S = H-1B1 Singapore. Also referred to as “Program” in prior years.
EMPLOYMENT_START_DATE	Beginning date of employment
EMPLOYMENT_END_DATE	Ending date of employment
EMPLOYER_NAME	Name of employer submitting labor condition application.
EMPLOYER_ADDRESS	Contact information of the Employer requesting temporary labor certification
EMPLOYER_CITY	
EMPLOYER_STATE	
EMPLOYER_POSTAL_CODE	
EMPLOYER_COUNTRY	
EMPLOYER_PROVINCE	
EMPLOYER_PHONE	
EMPLOYER_PHONE_EXT	
AGENT_ATTORNEY_NAME	Name of Agent or Attorney filing an H-1B application on behalf of the employer.
AGENT_ATTORNEY_CITY	City information for the Agent or Attorney filing an H-1B application on behalf of the employer.
AGENT_ATTORNEY_STATE	State information for the Agent or Attorney filing an H-1B application on behalf of the employer.

FIELD NAME	DESCRIPTION
JOB_TITLE	Title of the job
SOC_CODE	Occupational code associated with the job being requested for temporary labor condition, as classified by the Standard Occupational Classification (SOC) System.
SOC_NAME	Occupational name associated with the SOC_CODE
NAICS_CODE	Industry code associated with the employer requesting permanent labor condition, as classified by the North American Industrial Classification System (NAICS)
TOTAL_WORKERS	Total number of foreign workers requested by the Employer(s)
FULL_TIME_POSITION	Y = Full Time Position; N = Part Time Position
PREVAILING_WAGE	Prevailing Wage for the job being requested for temporary labor condition.
PW_UNIT_OF_PAY	Unit of Pay. Valid values include “Daily (DAI),” “Hourly (HR),” “Bi-weekly (BI),” “Weekly (WK),” “Monthly (MTH),” and “Yearly (YR)”
PW_SOURCE	Variables include "OES", "CBA", "DBA", "SCA" or "Other"
PW_SOURCE_YEAR	Year the Prevailing Wage Source was Issued
PW_SOURCE_OTHER	If "Other Wage Source", provide the source of wage
WAGE_RATE_OF_PAY_FROM	Employer's proposed wage rate
WAGE_RATE_OF_PAY_TO	Maximum proposed wage rate
WAGE_UNIT_OF_PAY	Unit of pay. Valid values include “Hour", "Week", "Bi-Weekly", "Month", or "Year"
H-1B_DEPENDENT	Y = Employer is H-1B Dependent; N = Employer is not H-1B Dependent
WILLFUL_VIOLATOR	Y = Employer has been previously found to be a Willful Violator; N = Employer has not been considered a Willful Violator
WORKSITE_CITY	City information of the foreign worker's intended area of employment
WORKSITE_COUNTY	County information of the foreign worker's intended area of employment
WORKSITE_STATE	State information of the foreign worker's intended area of employment
WORKSITE_POSTAL_CODE	Zip Code information of the foreign worker's intended area of employment
ORIGINAL_CERT_DATE	Original Certification Date for a Certified_Withdrawn application

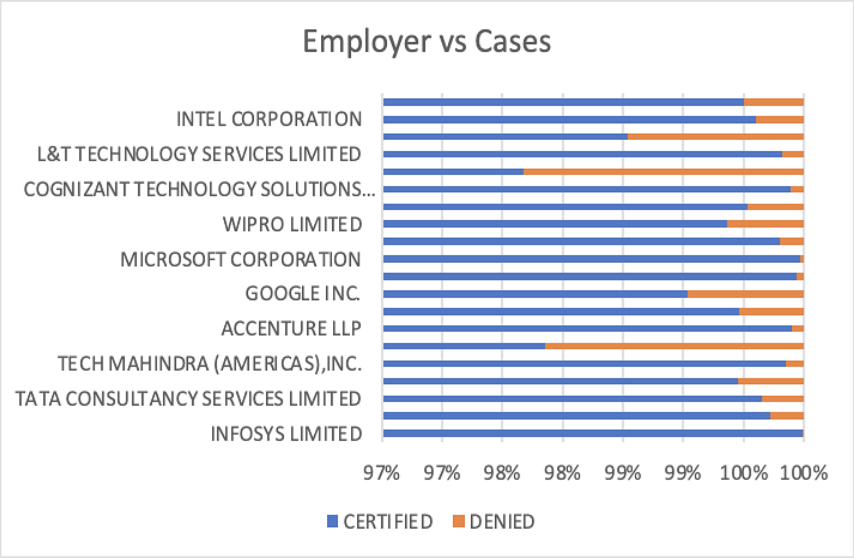
Exploratory Data Analysis (EDA)

- We identified the important features from the column names and observed the case result based on the features.
- For e.g. we noted that the company "Infosys" has the most H1-B applications and the company "Intel Corp" has the greatest success rate.
- We also looked at the number of applications per state and the success rate per state.
- We further performed an analysis of the job title, income, and nature of work of the applicant with respect to the number of applications and application status
- EDA helped us identify the most important features in our dataset which help determine the success rate for the LCA application

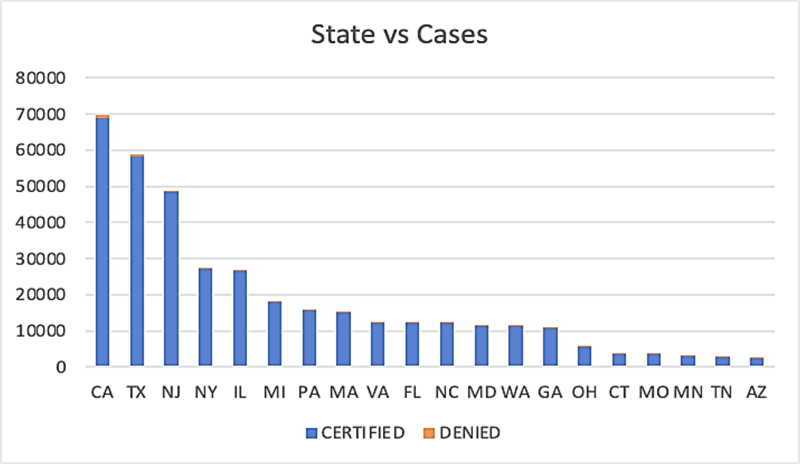
Exploratory Data Analysis (EDA)

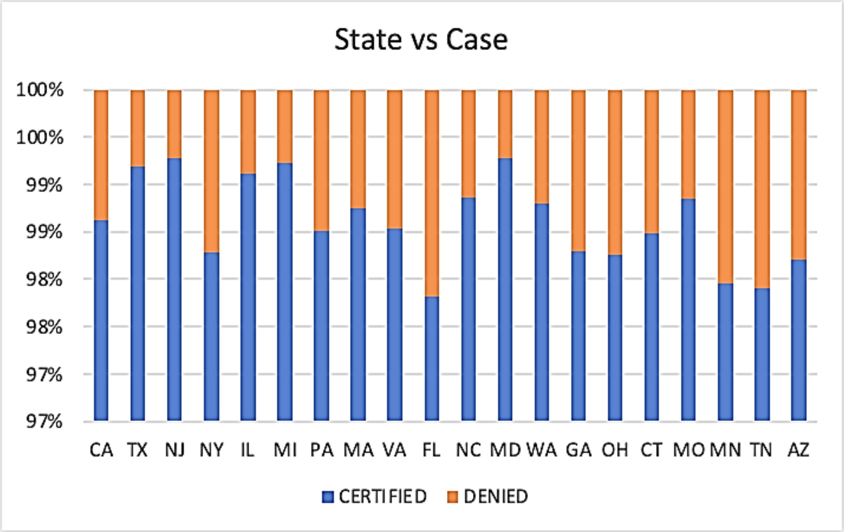


Infosys has the highest number of employee cases.



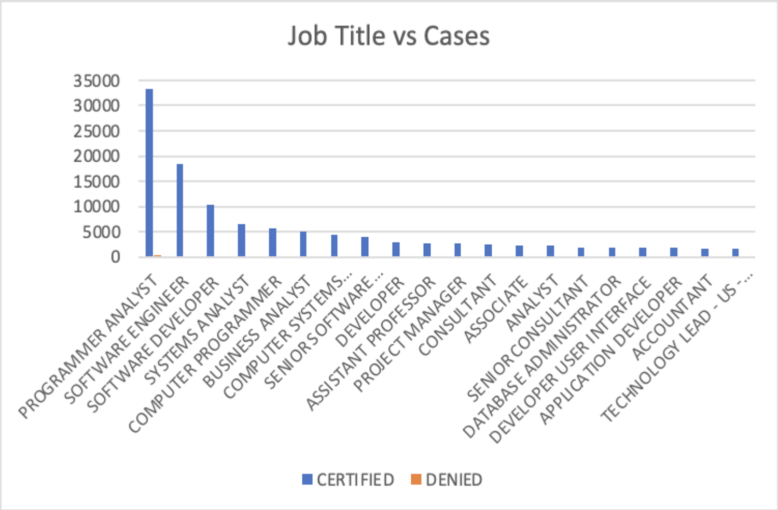
Exploratory Data Analysis (EDA)



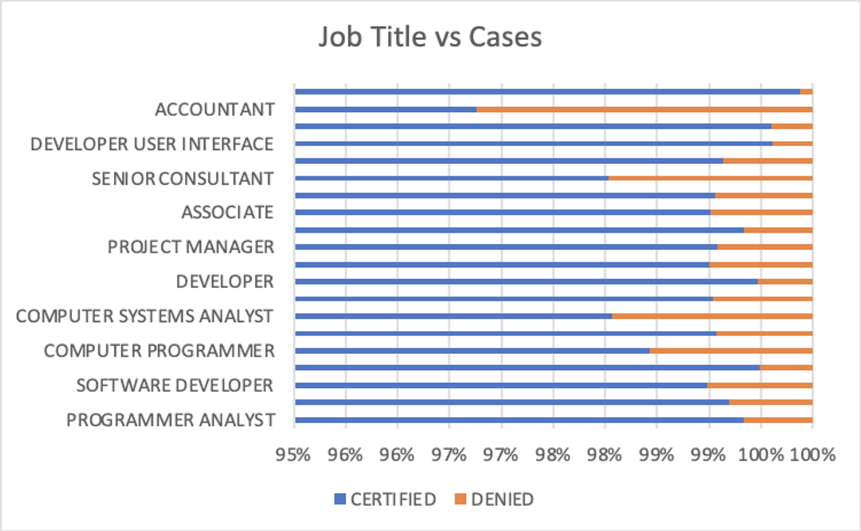


CA has the highest number of employee cases.

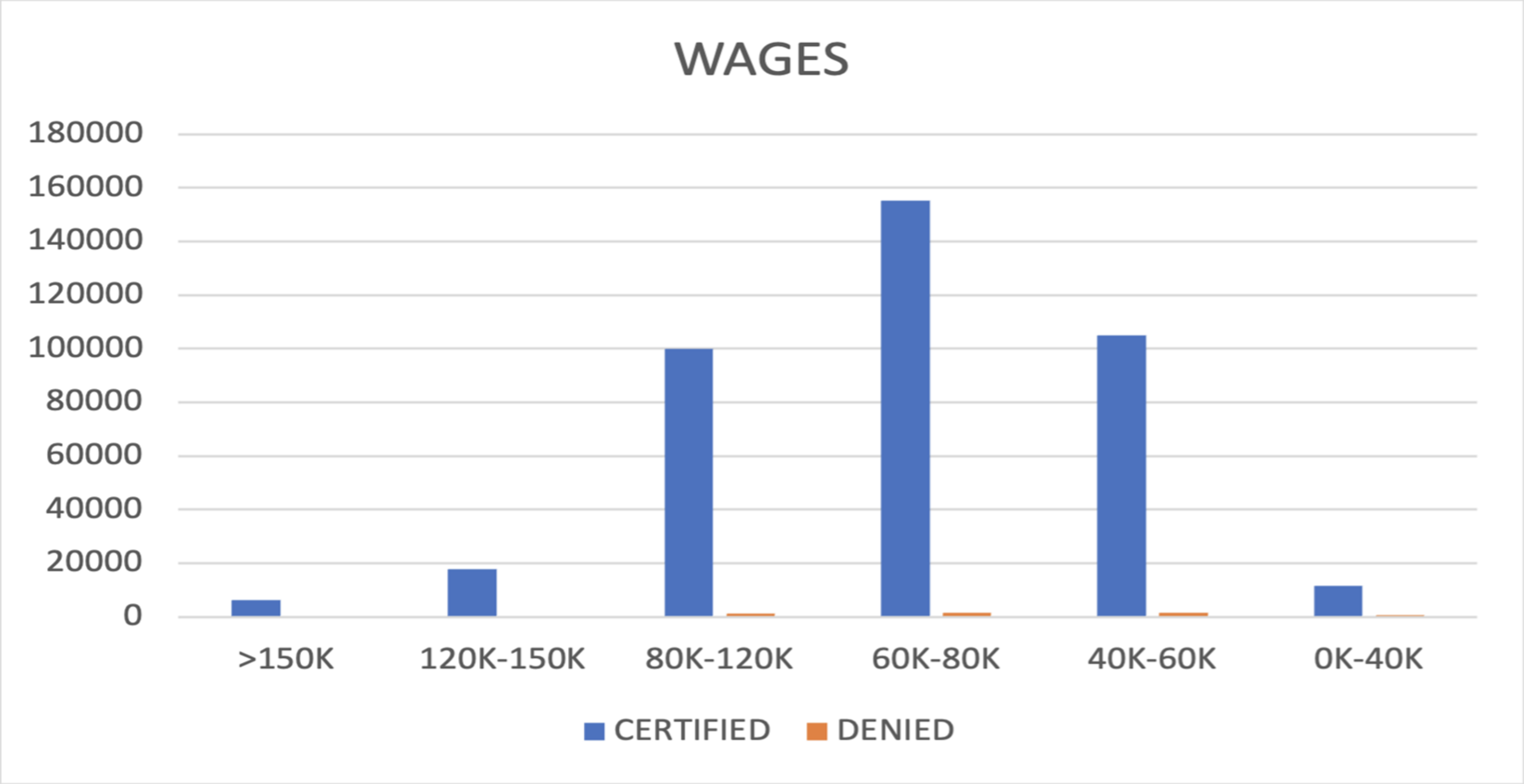
Exploratory Data Analysis (EDA)



Programmer analysts have the highest number of cases.



Exploratory Data Analysis (EDA)



Data Preprocessing

- Removing attributes that shall not be used for analysis and classification
- Multi-class classification to Binary class classification
- Feature value re-categorization and re-evaluation
- Handling missing data
- Handling imbalanced dataset
- Encoding categorical data
- Feature Scaling

Data Preprocessing

- Out of all the attributes, we would be using below attributes classification as per our EDA process.
- The original dataset also included details regarding other VISA types,withdrawn cases which needed to be eliminated prior to forming our training data set.
- CASE_STATUS, EMPLOYER_NAME, EMPLOYER_STATE, JOB_TITLE, SOC_NAME, TOTAL_WORKERS, FULL_TIME_POSITION, PREVAILING_WAGE, H-1B_DEPENDENT, WILLFUL_VIOLATOR

Multi-class to Binary class creation

```
+------------------+------+
|       CASE_STATUS| count|
+------------------+------+
|         CERTIFIED|468970|
|CERTIFIED-WITHDRAWN| 36171|
|         WITHDRAWN| 16016|
|            DENIED|  6989|
+------------------+------+
```

- Grouping CERTIFIED and CERTIFIED-WITHDRAWN cases
- Removing the WITHDRAWN case
- Final result –

[CERTIFIED, DENIED]

```
df['CASE_STATUS'].value_counts()
```

```
CERTIFIED    505141
DENIED         6989
Name: CASE_STATUS, dtype: int64
```

Feature Value re-categorization and re-evaluation

- `VISA_CLASS: ['H-1B' 'E-3 Australian' 'H-1B1 Singapore' 'H-1B1 Chile']`
- `EMPLOYER_COUNTRY: ['UNITED STATES OF AMERICA' 'CANADA' 'CAMBODIA' 'AUSTRALIA' nan 'CHINA']`

```
df['CASE_STATUS'].value_counts()
```

```
CERTIFIED     505141
IED             6989
e: CASE_STATUS, dtype: int64
```

- Rescaling WAGE attribute based on Unit of Pay, i.e, converting all the wages into yearly wages.

```
# rescale pay
def rescale_pay(wage, unit):
    if unit == 'Month':
        wage *= 12
    if unit == 'Week':
        wage *= 52
    if unit == 'Bi-Weekly':
        wage *= 26
    if unit == 'Hour':
        wage *= 40 * 52
    return wage
```

```
df['PREVAILING_WAGE'] = df[['PREVAILING_WAGE', 'PW_UNIT_OF_PAY']]\
                        .apply(lambda x: rescale_pay(x['PREVAILING_WAGE'], x['PW_UNIT_OF_PAY']), axis=1)
```

Feature Value re-categorization and re-evaluation

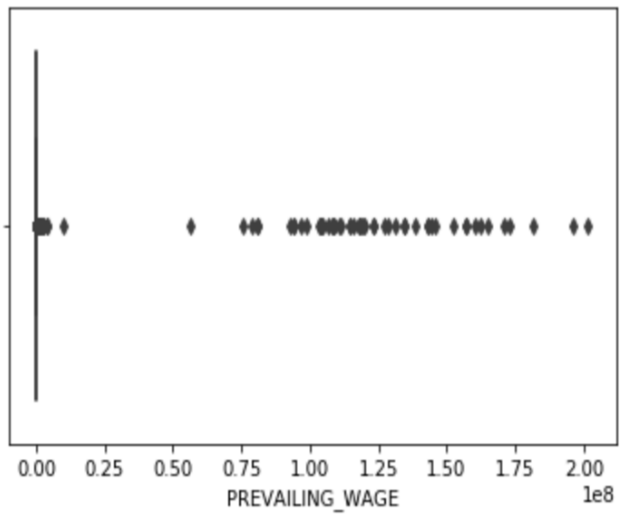
- Prevailing Wage

```
print(np.nanpercentile(df.PREVAILING_WAGE,98))
print(np.nanpercentile(df.PREVAILING_WAGE,2))
```

```
144961.0
37794.0
```

```
sb.boxplot(df['PREVAILING_WAGE'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed7fd2c4d0>
```



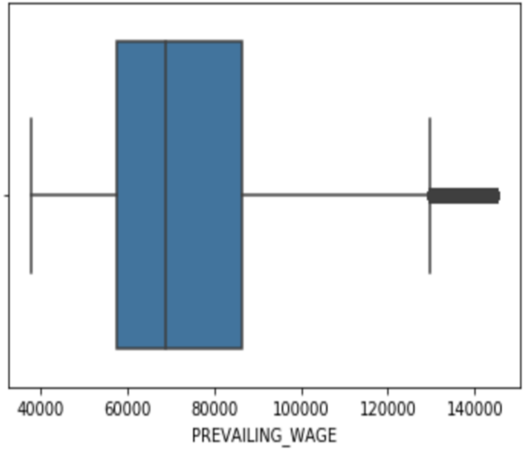
Feature Value re-categorization and re-evaluation

- Prevailing Wage

```
df.loc[df.PREVAILING_WAGE > 144961, 'PREVAILING_WAGE']= 144961
df.loc[df.PREVAILING_WAGE < 37794, 'PREVAILING_WAGE']= 37794
```

```
sb.boxplot(df['PREVAILING_WAGE'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed7d3f2e90>
```




```
df['PREVAILING_WAGE'].mean()
```

```
74283.97675386834
```

```
df['PREVAILING_WAGE'].median()
```

```
68827.0
```

```
df['PREVAILING_WAGE'].max()
```

```
144961.0
```

```
df['PREVAILING_WAGE'].min()
```

```
37794.0
```

Handling missing data

- For H-1B Dependent, Willful Violator, Full-time position, Job Title, SOC Name, replace missing data with Mode of respective columns
- Removing Employer Name, removing entire entry
- Additionally, removing Visa Class, Employer City and Unit of Pay columns

```
CASE_STATUS                0
VISA_CLASS                 0
EMPLOYER_NAME             43
EMPLOYER_CITY             14
EMPLOYER_STATE            15
EMPLOYER_COUNTRY      107712
JOB_TITLE                  3
SOC_NAME                   1
TOTAL_WORKERS              0
FULL_TIME_POSITION         4
PREVAILING_WAGE            0
PW_UNIT_OF_PAY            35
H-1B_DEPENDENT         10301
WILLFUL_VIOLATOR       10302
dtype: int64
```

Handling missing and unrequired data

- Initial result

```
CASE_STATUS                 0
VISA_CLASS                  0
EMPLOYER_NAME              43
EMPLOYER_CITY              14
EMPLOYER_STATE             15
EMPLOYER_COUNTRY       107712
JOB_TITLE                   3
SOC_NAME                    1
TOTAL_WORKERS               0
FULL_TIME_POSITION          4
PREVAILING_WAGE             0
PW_UNIT_OF_PAY             35
H-1B_DEPENDENT          10301
WILLFUL_VIOLATOR        10302
dtype: int64
```

- Final result

```
CASE_STATUS           0
EMPLOYER_NAME         0
EMPLOYER_STATE        0
JOB_TITLE             0
SOC_NAME              0
TOTAL_WORKERS         0
FULL_TIME_POSITION    0
PREVAILING_WAGE       0
H-1B_DEPENDENT        0
WILLFUL_VIOLATOR      0
dtype: int64
```

Handling Imbalanced Dataset

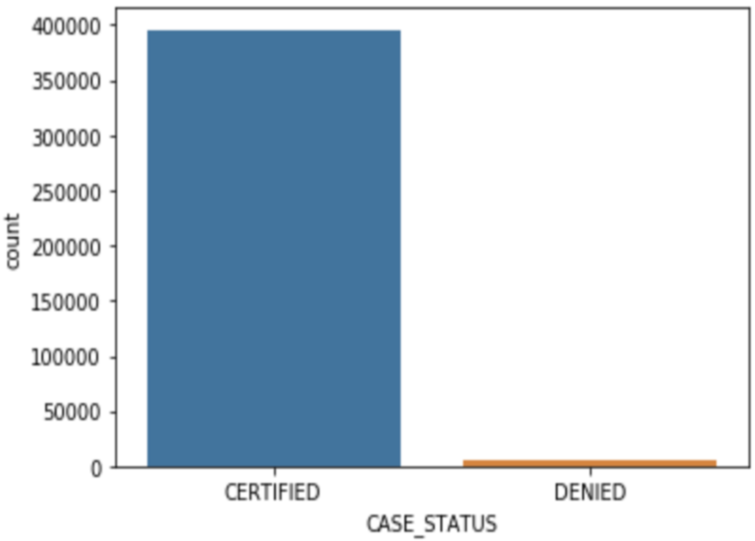
- We noted that there was a huge variance between the number cases that were “certified” and “denied”, which would lead to an imbalanced data set.

```
df['CASE_STATUS'].value_counts()
```

```
CERTIFIED    395583
DENIED         5026
Name: CASE_STATUS, dtype: int64
```

```
sb.countplot(df['CASE_STATUS'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed73657990>
```

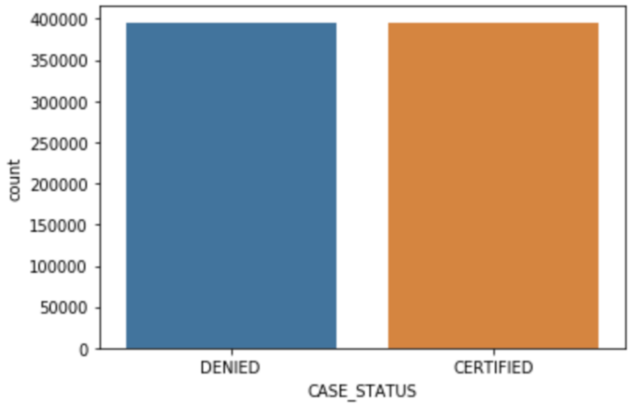


Handling Imbalanced Dataset

Over sampling

```
sb.countplot(df_os['CASE_STATUS'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fed7e850590>



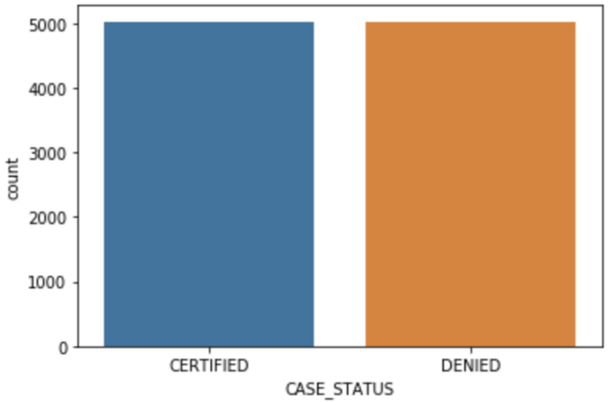

```
df_os['CASE_STATUS'].value_counts()
```

```
CERTIFIED    395583
DENIED       395583
Name: CASE_STATUS, dtype: int64
```

Under sampling

```
sb.countplot(df_us['CASE_STATUS'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fed6fd50f90>




```
df_us['CASE_STATUS'].value_counts()
```

```
DENIED       5026
CERTIFIED    5026
Name: CASE_STATUS, dtype: int64
```

Encoding Categorical Data

- We encoded all categorical type data in order to create a training dataset which is free of "labels"
- Employer_Name, Employer_State, Job_Title, SOC_Name, Full_Time_Position, H-1B_Dependent, Willful_Violator have Categorical Values
- Encoded values of each column into binary input by using One Hot Encoder
- Result -> 9 to 119401 columns

```
# Encoding Categorical data
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

```
column_transformer = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0,1,2,3,4,5,6])],\
                                       remainder='passthrough')
```

```
X = column_transformer.fit_transform(X)
```

```
X.shape
```

```
(791166, 119401)
```

Feature Scaling

- Employer_Name, Employer_State, Job_Title, SOC_Name, Full_Time_Position, H-1B_Dependent, Willful_Violator are encoded as 0 / 1
- Prevailing_Wage and Total_Workers have numerical data >>> 0/1
- Hence, featured scaled those attributes to have range of those values within -1 and 1

```python
# Feature Scaling
from sklearn.preprocessing import StandardScaler
```

```python
std_sclr = StandardScaler(with_mean=False)
X[:,[-1, -2]] = std_sclr.fit_transform(X[:,[-1, -2]])
```

Classification Models

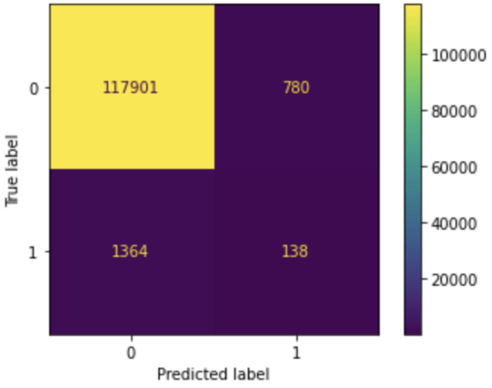
- Decision Tree Classifier
- Random Forest Classifier
- Naive Bayes Classifier
- KNN Classifier

DECISION TREE CLASSIFIER

Unsampled

```
accuracy_score(y_test, y_pred)
```

0.9821605385121024




```
precision_score(y_test, y_pred)
```

0.1503267973856209

```
recall_score(y_test, y_pred)
```

0.09187749667110519

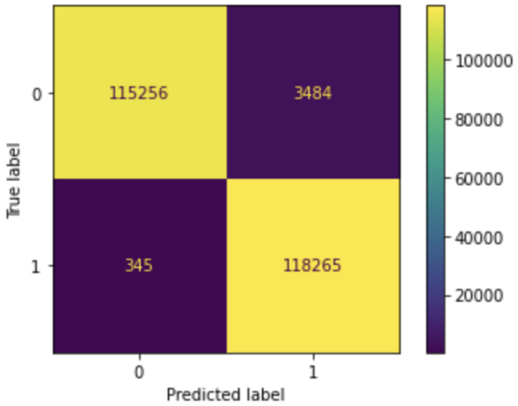
```
f1_score(y_test, y_pred)
```

0.1140495867768595

OverSampled

```
accuracy_score(y_test, y_pred)
```

0.9838677059195281




```
precision_score(y_test, y_pred)
```

0.9713837485318154

```
recall_score(y_test, y_pred)
```

0.99709130764691

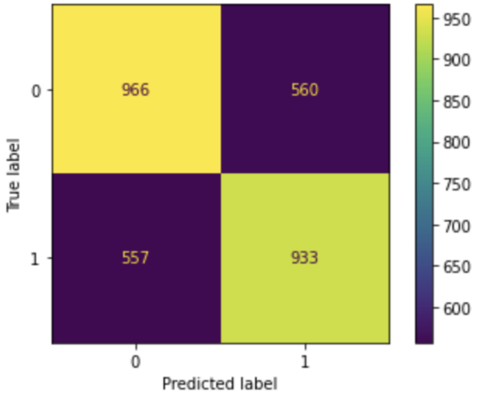
```
f1_score(y_test, y_pred)
```

0.984069662463232

UnderSampled

```
accuracy_score(y_test, y_pred)
```

0.6296419098143236




```
precision_score(y_test, y_pred)
```

0.6249162759544541

```
recall_score(y_test, y_pred)
```

0.6261744966442953

```
f1_score(y_test, y_pred)
```

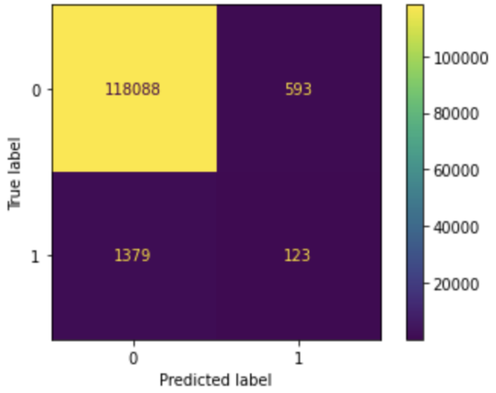
0.6255447536037545

RANDOM FOREST CLASSIFIER

Unsampled

```
accuracy_score(y_test, y_pred)
```

0.9835916893404225




```
precision_score(y_test, y_pred)
```

0.1717877094972067

```
recall_score(y_test, y_pred)
```

0.08189081225033289

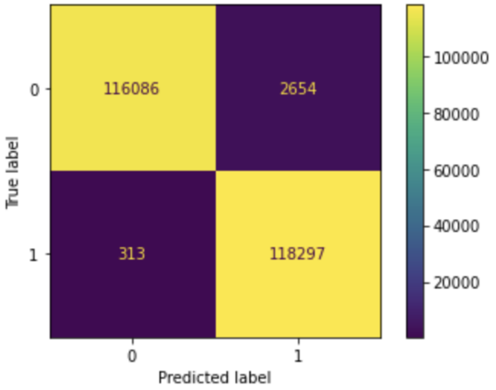
```
f1_score(y_test, y_pred)
```

0.11091073038773672

OverSampled

```
accuracy_score(y_test, y_pred)
```

0.9874994733515905




```
precision_score(y_test, y_pred)
```

0.9780572297872693

```
recall_score(y_test, y_pred)
```

0.9973610994013995

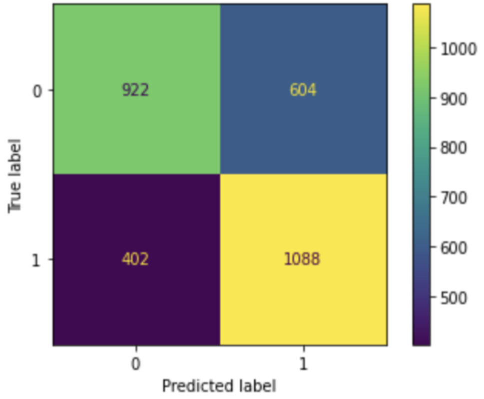
```
f1_score(y_test, y_pred)
```

0.9876148454882056

UnderSampled

```
accuracy_score(y_test, y_pred)
```

0.66445623342175




```
precision_score(y_test, y_pred)
```

0.6430260047281324

```
recall_score(y_test, y_pred)
```

0.7302013422818792

```
f1_score(y_test, y_pred)
```

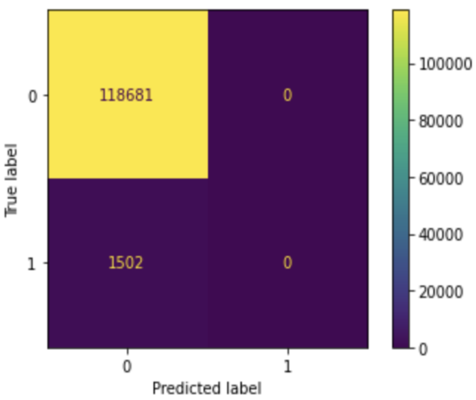
0.6838466373350094

NAIVE BAYES CLASSIFIER

Unsampled

```
accuracy_score(y_test, y_pred)
```

0.9875023921852508




```
precision_score(y_test, y_pred)
```

0.0

```
recall_score(y_test, y_pred)
```

0.0

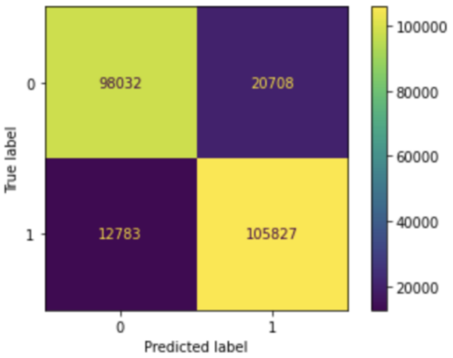
```
f1_score(y_test, y_pred)
```

0.0

OverSampled

```
accuracy_score(y_test, y_pred)
```

0.8588961449336423




```
precision_score(y_test, y_pred)
```

0.8363456751096534

```
recall_score(y_test, y_pred)
```

0.8922266250737712

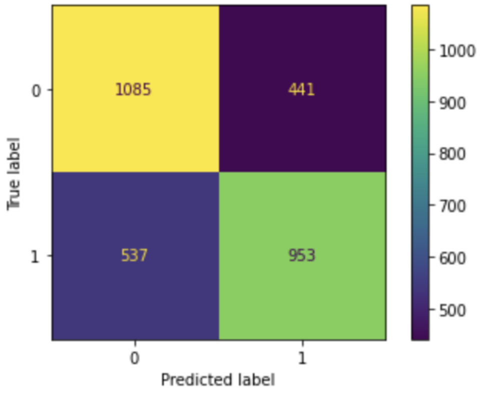
```
f1_score(y_test, y_pred)
```

0.8633828958371575

UnderSampled

```
accuracy_score(y_test, y_pred)
```

0.6757294429708223




```
precision_score(y_test, y_pred)
```

0.6836441893830703

```
recall_score(y_test, y_pred)
```

0.6395973154362417

```
f1_score(y_test, y_pred)
```

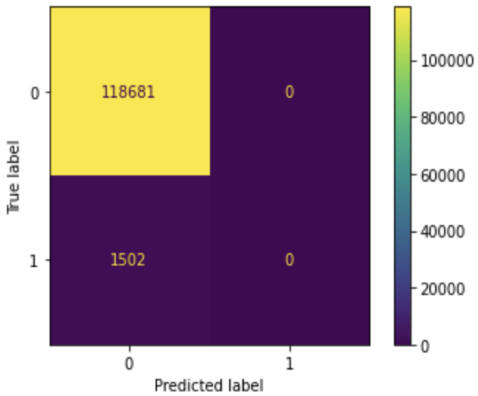
0.660887656033287

NAIVE BAYES CLASSIFIER

Unsampled

```
accuracy_score(y_test, y_pred)
```

0.9875023921852508




```
precision_score(y_test, y_pred)
```

0.0

```
recall_score(y_test, y_pred)
```

0.0

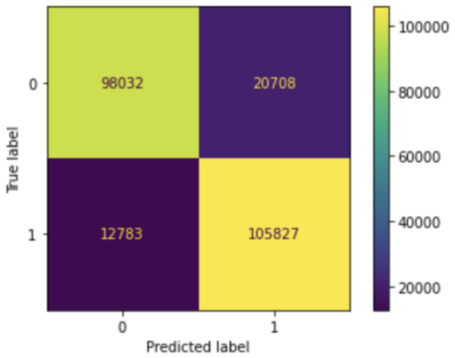
```
f1_score(y_test, y_pred)
```

0.0

OverSampled

```
accuracy_score(y_test, y_pred)
```

0.8588961449336423




```
precision_score(y_test, y_pred)
```

0.8363456751096534

```
recall_score(y_test, y_pred)
```

0.8922266250737712

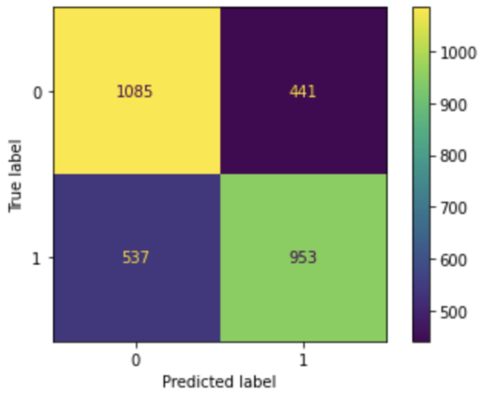
```
f1_score(y_test, y_pred)
```

0.8633828958371575

UnderSampled

```
accuracy_score(y_test, y_pred)
```

0.6757294429708223




```
precision_score(y_test, y_pred)
```

0.6836441893830703

```
recall_score(y_test, y_pred)
```

0.6395973154362417

```
f1_score(y_test, y_pred)
```

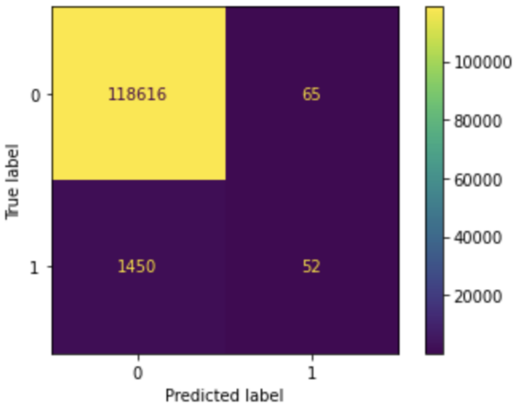
0.660887656033287

KNN CLASSIFIER

Unsampled

```
accuracy_score(y_test, y_pred)
```

0.9873942238086918




```
precision_score(y_test, y_pred)
```

0.4444444444444444

```
recall_score(y_test, y_pred)
```

0.03462050599201065

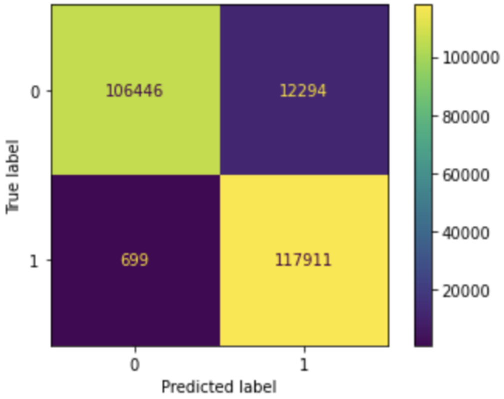
```
f1_score(y_test, y_pred)
```

0.06423718344657195

OverSampled

```
accuracy_score(y_test, y_pred)
```

0.9452580577206657




```
precision_score(y_test, y_pred)
```

0.9055796628393686

```
recall_score(y_test, y_pred)
```

0.99410673636287

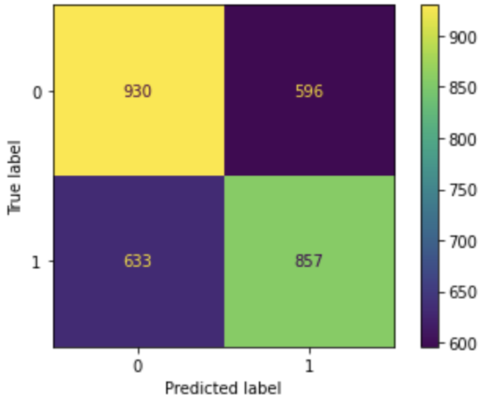
```
f1_score(y_test, y_pred)
```

0.9477804794727006

UnderSampled

```
accuracy_score(y_test, y_pred)
```

0.5925066312997348




```
precision_score(y_test, y_pred)
```

0.5898141775636614

```
recall_score(y_test, y_pred)
```

0.5751677852348993

```
f1_score(y_test, y_pred)
```

0.582398912674142

Accuracy & F1-score comparison (Oversampled data)

Classification Model	Accuracy	F1 score
Decision Tree	98.39%	0.9841
Random Forest	98.75%	0.9876
Naive Bayes	85.89%	0.8634
KNN	94.53%	0.9478

Conclusion

- We made a comparitive study between the results obtained for undersampled and oversampled dataset. With Imbalanced class data, accuracy was achieved, however, it would fail for minority classes as seen from its Precision, Recall and F1 score.

- With balanced data, in the case of Under Sampled Class data, all the models failed to achieve reasonable accuracy

- In case of Over Sampled Class data, all the models have higher accuracy compared to Imbalanced class and under sampled class

References

- Ian Greenleigh, “Data world”, https://data.world/ian/h-1-b-disclosure-data-fy17/workspace/file?filename=H-1B_FY17_Record_Layout.pdf, 2017.
- Kumar, “What is H1B LCA ? Why file it? Salary, Processing times – DOL”, February 2022, https://redbus2us.com/what-is-h1b-lca-why-file-it-salary-processing-timesdol

THANK YOU!