Visa_Approval_Prediction

May 30, 2024

1 Objective : Visa Approval Classification

2 Exploratory Data Analysis (EDA) - Python

3 Insights - Patterns

4 Classification (Using the ML)

5 1. Load Python Modules

[1]:
```
#  Use Python's import statement to load modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
```

```
from tabulate import tabulate

from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder


from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.naive_bayes import CategoricalNB
from sklearn.naive_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier


from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imblearn.over_sampling import SMOTE
```

6 2. Read the Dataset from CSV file - Using Pandas

[2]:
```
file_path=r"Visa_Predection_Dataset.csv"
visa_df=pd.read_csv(file_path)
visa_df
```

[2]:
```
         case_id continent education_of_employee has_job_experience  \
0         EZYV01      Asia           High School                  N
1         EZYV02      Asia              Master's                  Y
2         EZYV03      Asia            Bachelor's                  N
3         EZYV04      Asia            Bachelor's                  N
4         EZYV05    Africa              Master's                  Y
…              …         …                     …                  …
```

```
25475  EZYV25476      Asia          Bachelor's                 Y
25476  EZYV25477      Asia         High School                 Y
25477  EZYV25478      Asia            Master's                 Y
25478  EZYV25479      Asia            Master's                 Y
25479  EZYV25480      Asia          Bachelor's                 Y

      requires_job_training  no_of_employees  yr_of_estab  \
0                         N            14513         2007
1                         N             2412         2002
2                         Y            44444         2008
3                         N               98         1897
4                         N             1082         2005
…                      …              …           …
25475                     Y             2601         2008
25476                     N             3274         2006
25477                     N             1121         1910
25478                     Y             1918         1887
25479                     N             3195         1960

      region_of_employment  prevailing_wage unit_of_wage full_time_position  \
0                     West         592.2029         Hour                  Y
1                Northeast       83425.6500         Year                  Y
2                     West      122996.8600         Year                  Y
3                     West       83434.0300         Year                  Y
4                    South      149907.3900         Year                  Y
…                     …              …          …                …
25475                South       77092.5700         Year                  Y
25476            Northeast      279174.7900         Year                  Y
25477                South      146298.8500         Year                  N
25478                 West       86154.7700         Year                  Y
25479              Midwest       70876.9100         Year                  Y

      case_status
0          Denied
1       Certified
2          Denied
3          Denied
4       Certified
…            …
25475   Certified
25476   Certified
25477   Certified
25478   Certified
25479   Certified

[25480 rows x 12 columns]
```

[3]:
```python
#drop - sensitive - non imp columns for data analysis
print(visa_df["case_id"].nunique())
visa_df.drop("case_id",axis=1,inplace=True)
# print columns names
print(visa_df.columns)
```

```
25480
Index(['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_status'],
      dtype='object')
```

7 3. Basic Inspection on given dataset

[4]:
```python
def basic_inspection_dataset(table):
    """Generates a basic inspection dataset from the given table."""

    print("top 5 rows - using head")
    print(table.head())
    print()

    print("bottom 5 rows using tail")
    print(table.tail())
    print()

    print("numbers of samples and columns")
    print(table.shape)
    print()

    print("numbers of samples ")
    print(len(table))
    print()

    print("numbers of entries in the data frame")
    print(table.size)
    print()

    print("Columns Names")
    print(table.columns)
    print()

    print("Columns dtypes")
    print(table.dtypes)
    print()

    print("Dataframe info")
```

```python
    print(table.info())
    print()

    print()
    print("check the missing value in each column")
    print(table.isnull().sum())

    print()
    print("check the missing value in each column")
    print(table.isna().sum())

    print()
    print("table describe")
    print(table.describe())

basic_inspection_dataset(visa_df)
```

```
top 5 rows - using head
  continent education_of_employee has_job_experience requires_job_training  \
0      Asia           High School                  N                     N
1      Asia              Master's                  Y                     N
2      Asia            Bachelor's                  N                     Y
3      Asia            Bachelor's                  N                     N
4    Africa              Master's                  Y                     N

   no_of_employees  yr_of_estab region_of_employment  prevailing_wage  \
0            14513         2007                 West         592.2029
1             2412         2002            Northeast       83425.6500
2            44444         2008                 West      122996.8600
3               98         1897                 West       83434.0300
4             1082         2005                South      149907.3900

  unit_of_wage full_time_position case_status
0         Hour                  Y      Denied
1         Year                  Y   Certified
2         Year                  Y      Denied
3         Year                  Y      Denied
4         Year                  Y   Certified

bottom 5 rows using tail
      continent education_of_employee has_job_experience  \
25475      Asia            Bachelor's                  Y
25476      Asia           High School                  Y
25477      Asia              Master's                  Y
25478      Asia              Master's                  Y
25479      Asia            Bachelor's                  Y
```

```
      requires_job_training  no_of_employees  yr_of_estab  \
25475                     Y             2601         2008
25476                     N             3274         2006
25477                     N             1121         1910
25478                     Y             1918         1887
25479                     N             3195         1960

      region_of_employment  prevailing_wage unit_of_wage full_time_position  \
25475                South         77092.57         Year                  Y
25476            Northeast        279174.79         Year                  Y
25477                South        146298.85         Year                  N
25478                 West         86154.77         Year                  Y
25479              Midwest         70876.91         Year                  Y

      case_status
25475   Certified
25476   Certified
25477   Certified
25478   Certified
25479   Certified

numbers of samples and columns
(25480, 11)

numbers of samples
25480

numbers of entries in the data frame
280280

Columns Names
Index(['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_status'],
      dtype='object')

Columns dtypes
continent                 object
education_of_employee     object
has_job_experience        object
requires_job_training     object
no_of_employees            int64
yr_of_estab                int64
region_of_employment      object
prevailing_wage          float64
unit_of_wage              object
full_time_position        object
```

```
case_status               object
dtype: object

Dataframe info
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25480 entries, 0 to 25479
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   continent              25480 non-null  object
 1   education_of_employee  25480 non-null  object
 2   has_job_experience     25480 non-null  object
 3   requires_job_training  25480 non-null  object
 4   no_of_employees        25480 non-null  int64
 5   yr_of_estab            25480 non-null  int64
 6   region_of_employment   25480 non-null  object
 7   prevailing_wage        25480 non-null  float64
 8   unit_of_wage           25480 non-null  object
 9   full_time_position     25480 non-null  object
 10  case_status            25480 non-null  object
dtypes: float64(1), int64(2), object(8)
memory usage: 2.1+ MB
None


check the missing value in each column
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
dtype: int64

check the missing value in each column
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
dtype: int64

table describe
       no_of_employees   yr_of_estab  prevailing_wage
count     25480.000000  25480.000000     25480.000000
mean       5667.043210   1979.409929     74455.814592
std       22877.928848     42.366929     52815.942327
min         -26.000000   1800.000000         2.136700
25%        1022.000000   1976.000000     34015.480000
50%        2109.000000   1997.000000     70308.210000
75%        3504.000000   2005.000000    107735.512500
max      602069.000000   2016.000000    319210.270000
```

7.0.1 Observations - dataset

- Have 25480 Sample with Varaibles 12
- There is no null values in the dataset

Categorical Variables:

- case_id
- continent
- education_of_employee
- has_job_experience
- requires_job_training
- unit_of_wage
- full_time_position
- case_status
- yr_of_estab
- region_of_employment

Numerical Variables:

- prevailing_wage
- no_of_employees

8 4. Handling Missing Values - Categorical - Variables

[5]:
```python
# check for missing values - for confirmation
visa_df.isnull().sum()
```

[5]:
```
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
dtype: int64
```

9 5. Categorical- UniVariate - Analysis -Using Pipeline

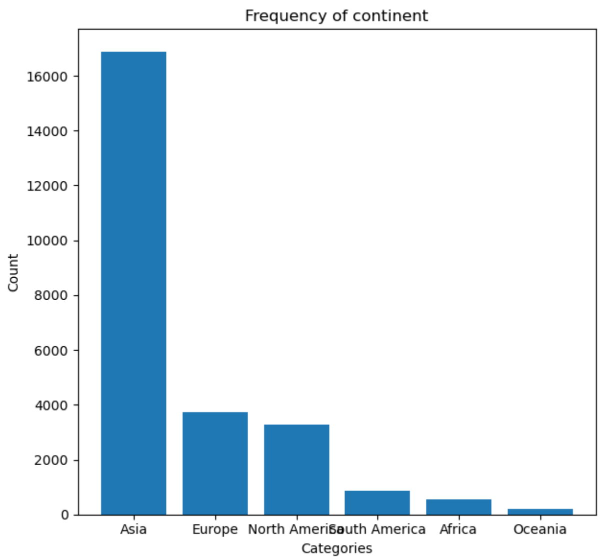
[6]:
```python
class BarPieChartTransformer(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        df=X.copy()
        # get cat columns
        cat_cols = df.select_dtypes(include='object').columns
        for cat_name in cat_cols:
            value_counts = df[cat_name].value_counts().reset_index()
            # Rename the columns
            value_counts.columns = ['Class', 'Frequency']

            # Print the result as a table
            print(f"{cat_name} frequency table")
            print(tabulate(value_counts, headers='keys', tablefmt='pretty'))

            # Calculate relative frequency
            total_count = value_counts['Frequency'].sum()
            value_counts['Relative Frequency %'] =␣
↪round((value_counts['Frequency'] / total_count)*100,2)

            # Print the result as a table
            print(f"{cat_name} Relative frequency table")
            print(tabulate(value_counts, headers='keys', tablefmt='pretty'))

            # Extract the values and index from value counts
            value_counts = df[cat_name].value_counts()
            values = value_counts.values
            labels = value_counts.index

            fig, axs = plt.subplots(1, 2, figsize=(12, 6))  # 1 row, 2 columns
```

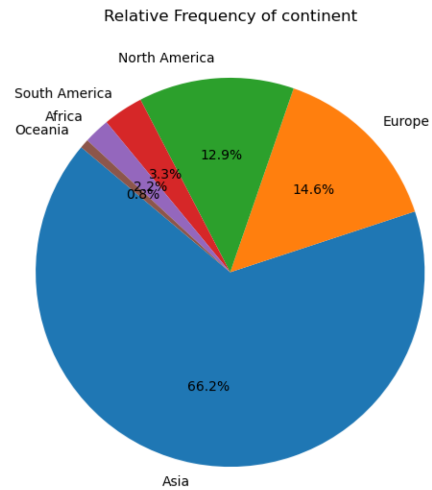
```
            # Create a bar graph
            axs[0].bar(labels, values)
            axs[0].set_title(f'Frequency of {cat_name}')
            axs[0].set_xlabel('Categories')  # Set x-label
            axs[0].set_ylabel('Count')       # Set y-label

            axs[1].pie(value_counts.values, labels=value_counts.index,␣
↪autopct='%1.1f%%', startangle=140)
            axs[1].set_title(f'Relative Frequency of {cat_name}')
            plt.tight_layout()
            # Show the plot
            plt.show()
```

[7]:

```
pipeline_cat_var = Pipeline([
    ('cat_univaraite_analysis', BarPieChartTransformer())
])

# Fit and transform your data using the pipeline
processed_data = pipeline_cat_var.fit_transform(visa_df)
```

```
continent frequency table
+---+---------------+-----------+
|   |     Class     | Frequency |
+---+---------------+-----------+
| 0 |     Asia      |   16861   |
| 1 |    Europe     |   3732    |
| 2 | North America |   3292    |
| 3 | South America |    852    |
| 4 |    Africa     |    551    |
| 5 |    Oceania    |    192    |
+---+---------------+-----------+
continent Relative frequency table
+---+---------------+-----------+----------------------+
|   |     Class     | Frequency | Relative Frequency % |
+---+---------------+-----------+----------------------+
| 0 |     Asia      |   16861   |        66.17         |
| 1 |    Europe     |   3732    |        14.65         |
| 2 | North America |   3292    |        12.92         |
| 3 | South America |    852    |         3.34         |
| 4 |    Africa     |    551    |         2.16         |
| 5 |    Oceania    |    192    |         0.75         |
+---+---------------+-----------+----------------------+
```



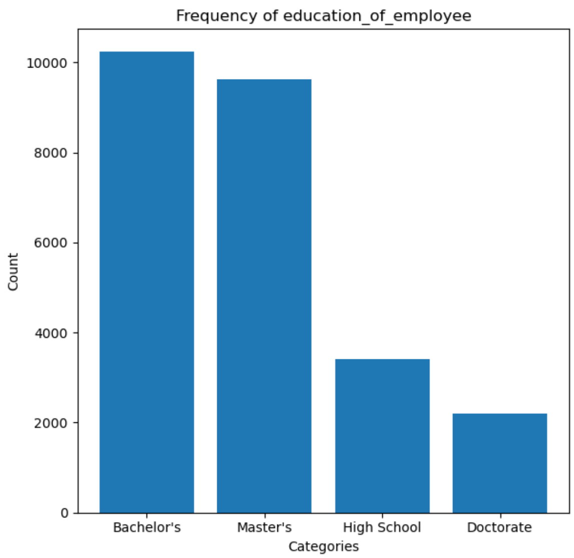


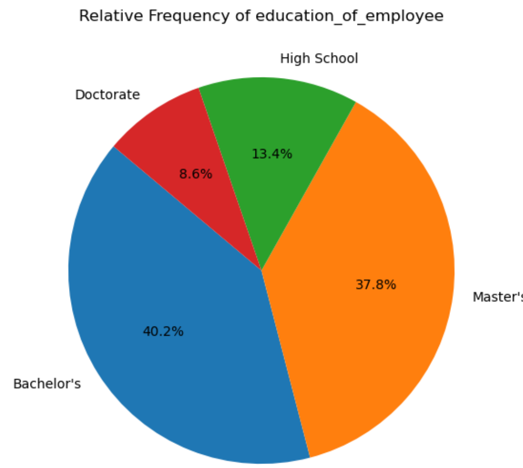
education_of_employee frequency table

	Class	Frequency
0	Bachelor's	10234
1	Master's	9634
2	High School	3420
3	Doctorate	2192

education_of_employee Relative frequency table

	Class	Frequency	Relative Frequency %
0	Bachelor's	10234	40.16
1	Master's	9634	37.81
2	High School	3420	13.42
3	Doctorate	2192	8.6



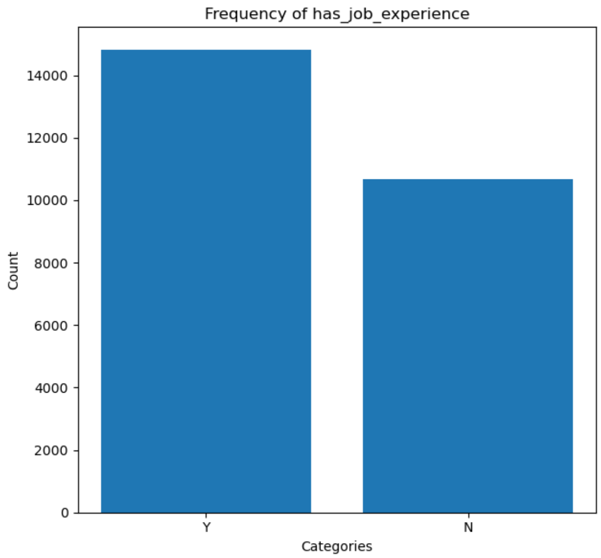


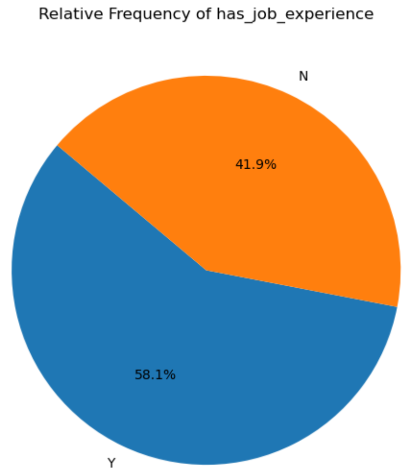
has_job_experience frequency table

	Class	Frequency
0	Y	14802
1	N	10678

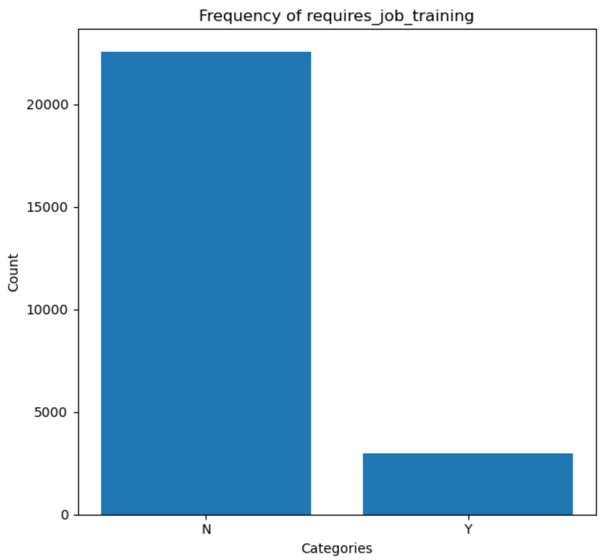
has_job_experience Relative frequency table

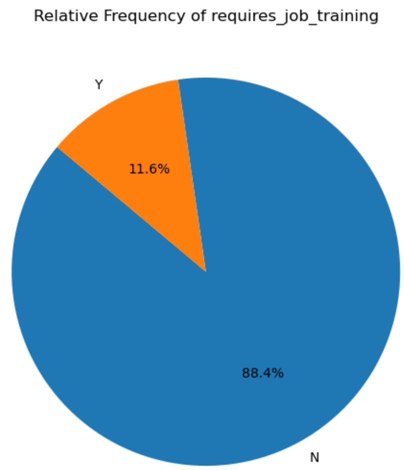
	Class	Frequency	Relative Frequency %
0	Y	14802	58.09
1	N	10678	41.91






```
requires_job_training frequency table
+---+-------+-----------+
|   | Class | Frequency |
+---+-------+-----------+
| 0 |   N   |   22525   |
| 1 |   Y   |   2955    |
+---+-------+-----------+
requires_job_training Relative frequency table
+---+-------+-----------+----------------------+
|   | Class | Frequency | Relative Frequency % |
+---+-------+-----------+----------------------+
| 0 |   N   |   22525   |         88.4         |
| 1 |   Y   |   2955    |         11.6         |
+---+-------+-----------+----------------------+
```



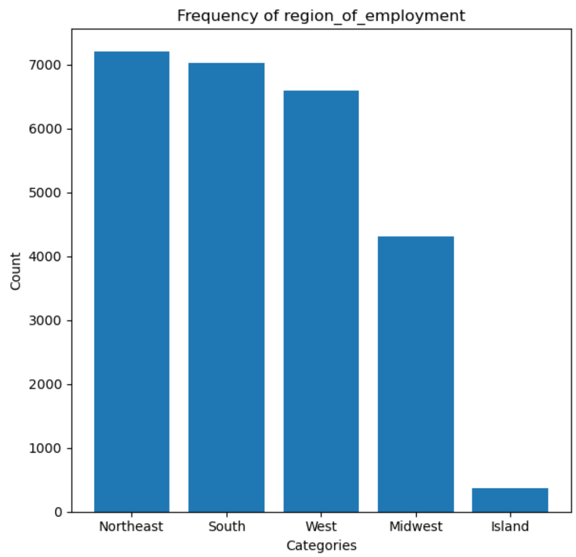


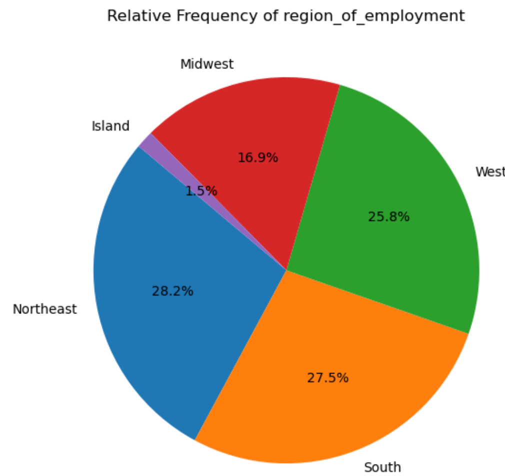
region_of_employment frequency table

	Class	Frequency
0	Northeast	7195
1	South	7017
2	West	6586
3	Midwest	4307
4	Island	375

region_of_employment Relative frequency table

	Class	Frequency	Relative Frequency %
0	Northeast	7195	28.24
1	South	7017	27.54
2	West	6586	25.85
3	Midwest	4307	16.9
4	Island	375	1.47



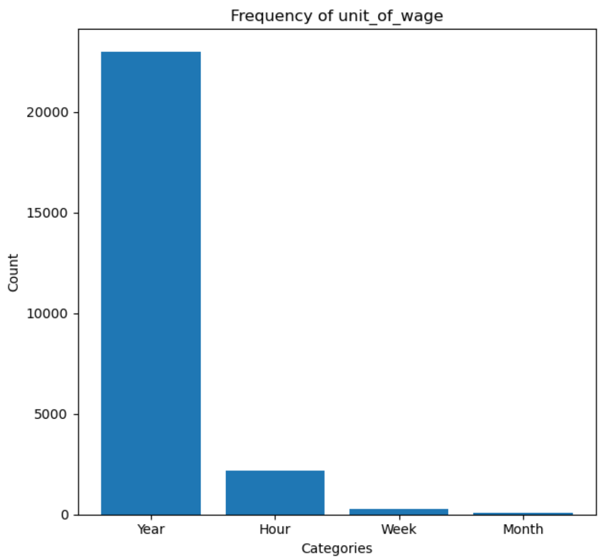


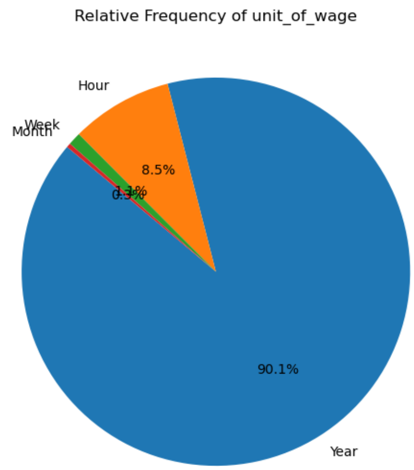
unit_of_wage frequency table

	Class	Frequency
0	Year	22962
1	Hour	2157
2	Week	272
3	Month	89

unit_of_wage Relative frequency table

	Class	Frequency	Relative Frequency %
0	Year	22962	90.12
1	Hour	2157	8.47
2	Week	272	1.07
3	Month	89	0.35



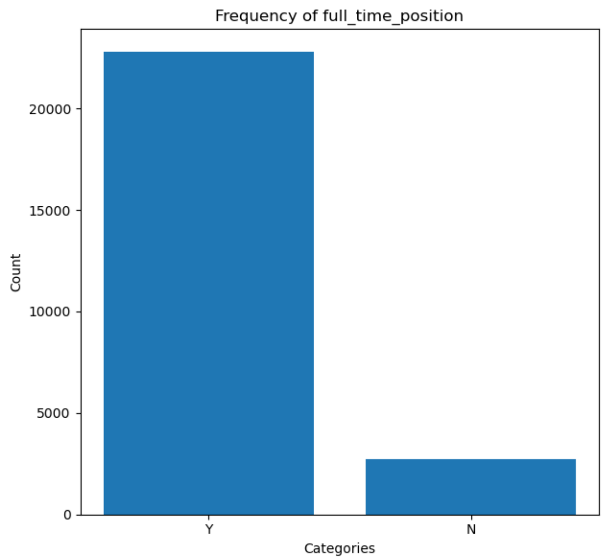


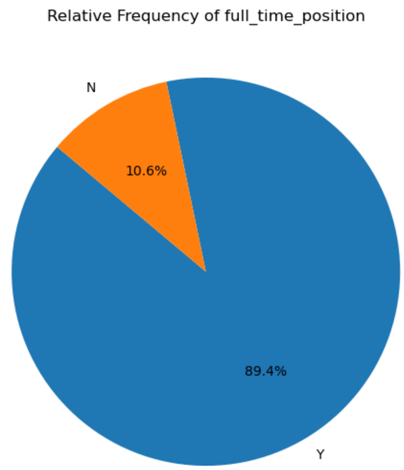
full_time_position frequency table

	Class	Frequency
0	Y	22773
1	N	2707

full_time_position Relative frequency table

	Class	Frequency	Relative Frequency %
0	Y	22773	89.38
1	N	2707	10.62



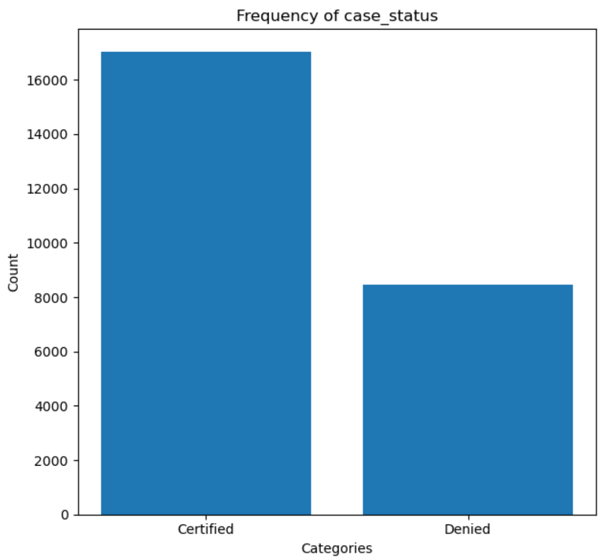


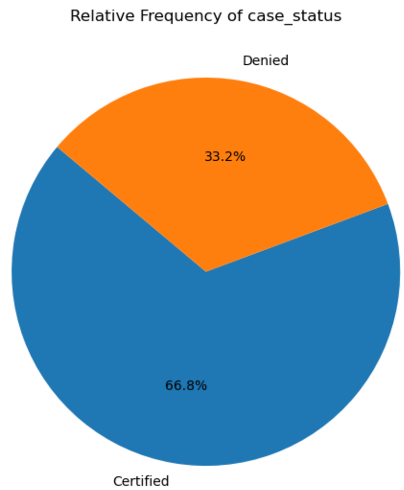
case_status frequency table

	Class	Frequency
0	Certified	17018
1	Denied	8462

case_status Relative frequency table

	Class	Frequency	Relative Frequency %
0	Certified	17018	66.79
1	Denied	8462	33.21





10 6. Handling Missing Values in Numerical Columns

[8]:
```
visa_df.isnull().sum()
```

[8]:
```
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
dtype: int64
```

[9]:
```
visa_df.describe()
```

[9]:

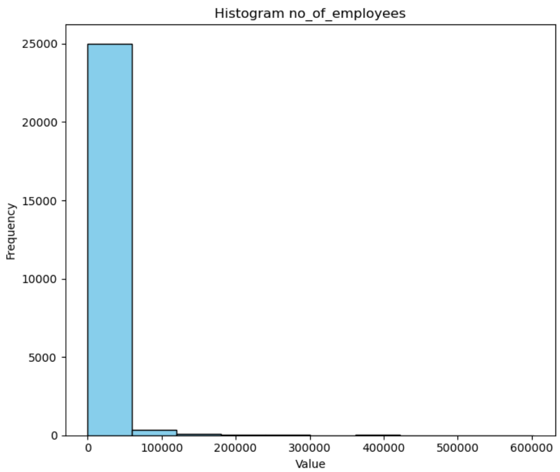
	no_of_employees	yr_of_estab	prevailing_wage
count	25480.000000	25480.000000	25480.000000
mean	5667.043210	1979.409929	74455.814592
std	22877.928848	42.366929	52815.942327
min	-26.000000	1800.000000	2.136700
25%	1022.000000	1976.000000	34015.480000
50%	2109.000000	1997.000000	70308.210000
75%	3504.000000	2005.000000	107735.512500

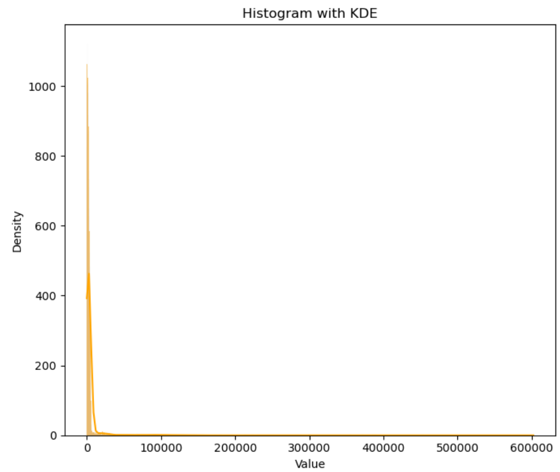
```
max      602069.000000   2016.000000    319210.270000
```

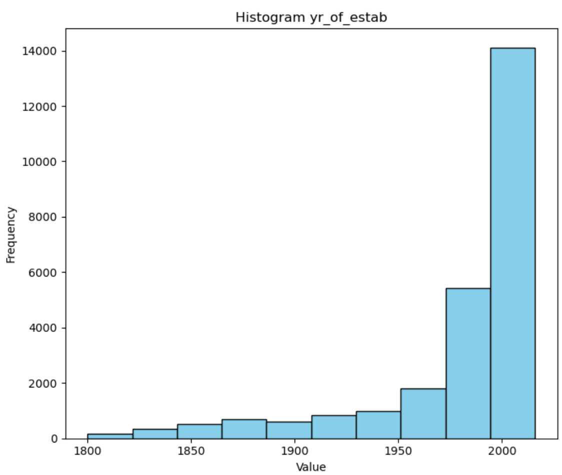
11 7. Numerical - UniVariate - Analysis - Using -Pipeline

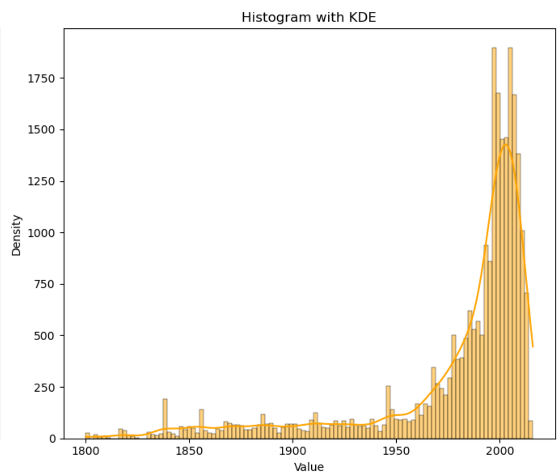
[10]:
```
class HistBoxChartTransformer(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        df=X.copy()
        # getting num cols
        num_cols = df.select_dtypes(exclude='object').columns
        for con_var in num_cols:

            # Create a figure and axes object
            fig, axes = plt.subplots(1, 2, figsize=(14, 6))

            # Plot histogram without KDE on the left
            axes[0].hist(df[con_var], color='skyblue', edgecolor='black')
            axes[0].set_xlabel('Value')
            axes[0].set_ylabel('Frequency')
            axes[0].set_title(f'Histogram {con_var}')

            # Plot histogram with KDE on the right
            sns.histplot(data=df, x=con_var, kde=True, color='orange',␣
 ↪edgecolor='black', ax=axes[1])
            axes[1].set_xlabel('Value')
            axes[1].set_ylabel('Density')
            axes[1].set_title('Histogram with KDE')

            # Adjust layout
            plt.tight_layout()

            # Show the combined plot
            plt.show()
```

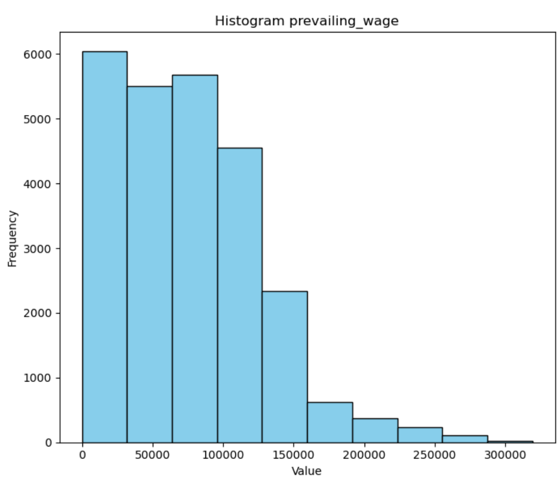
[11]:
```
pipeline_num_var = Pipeline([
    ('num_uni_variate_analysis', HistBoxChartTransformer())
])

# Fit and transform your data using the pipeline
processed_data = pipeline_num_var.fit_transform(visa_df)
```

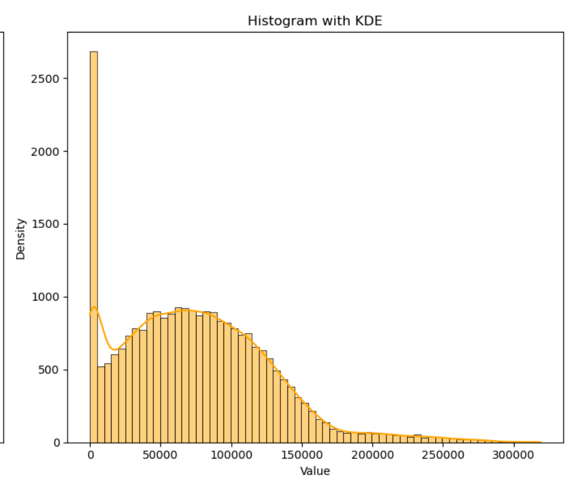

Histogram no_of_employees
Frequency
Value


Histogram with KDE
Density
Value


Histogram yr_of_estab
Frequency
Value


Histogram with KDE
Density
Value


Histogram prevailing_wage
Frequency
Value


Histogram with KDE
Density
Value

12 8. Numerical - Variables -Outliers Analysis - fillit

13 9. Bi Variate Analyis

13.1 9.1 cat to target(cat)

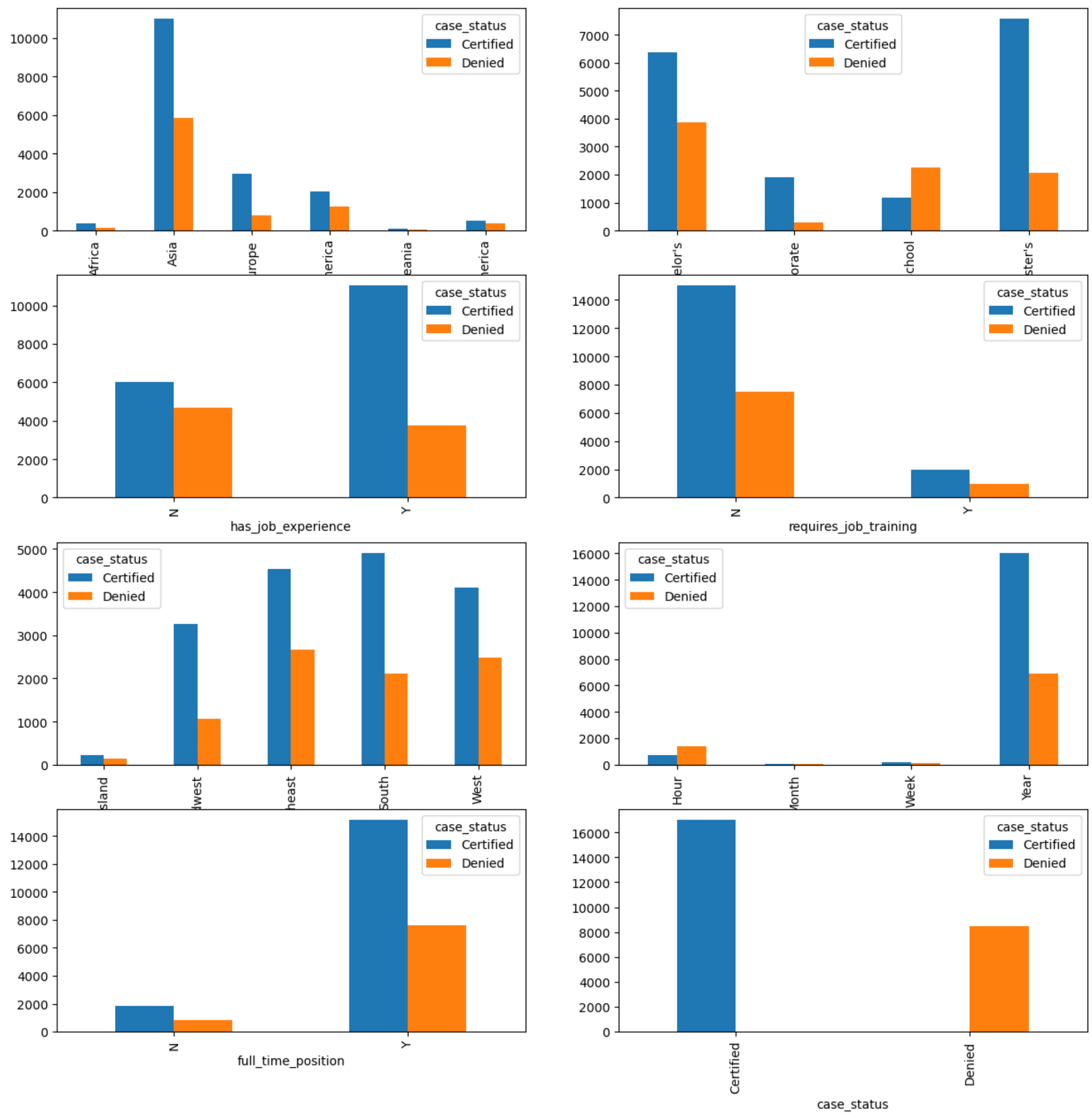
[12]:
```
cat_vars = visa_df.select_dtypes(include="object").columns
print(cat_vars)
```

```
Index(['continent', 'education_of_employee', 'has_job_experience',
       'requires_job_training', 'region_of_employment', 'unit_of_wage',
       'full_time_position', 'case_status'],
      dtype='object')
```

[13]:
```
target="case_status"
fig,ax = plt.subplots(4,2,figsize=(15,15))
for axi,x in zip(ax.flat,cat_vars):
    col1=visa_df[x]
    col2=visa_df[target]
    result=pd.crosstab(col1,col2)
    print(result)
    print("===============================")
    result.plot(kind='bar',ax=axi)
```

```
case_status    Certified  Denied
continent
Africa               397     154
Asia               11012    5849
Europe              2957     775
North America       2037    1255
Oceania              122      70
South America        493     359
===============================
case_status            Certified  Denied
education_of_employee
Bachelor's                  6367    3867
Doctorate                   1912     280
High School                 1164    2256
Master's                    7575    2059
===============================
case_status         Certified  Denied
has_job_experience
N                        5994    4684
Y                       11024    3778
===============================
case_status            Certified  Denied
```

```
requires_job_training
N                           15012    7513
Y                            2006     949
===============================
case_status           Certified  Denied
region_of_employment
Island                      226     149
Midwest                    3253    1054
Northeast                  4526    2669
South                      4913    2104
West                       4100    2486
===============================
case_status   Certified  Denied
unit_of_wage
Hour                747    1410
Month                55      34
Week                169     103
Year              16047    6915
===============================
case_status         Certified  Denied
full_time_position
N                        1855     852
Y                       15163    7610
===============================
case_status  Certified  Denied
case_status
Certified        17018       0
Denied               0    8462
===============================
```



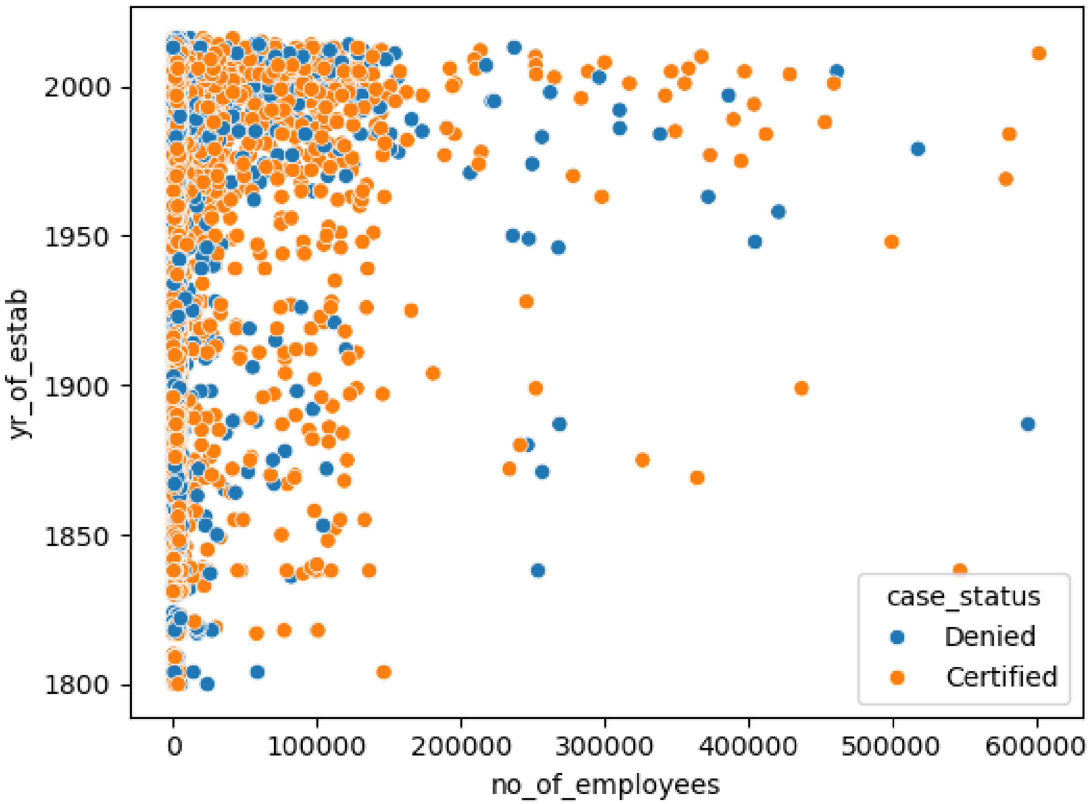
13.2 9.2 Num vs Num

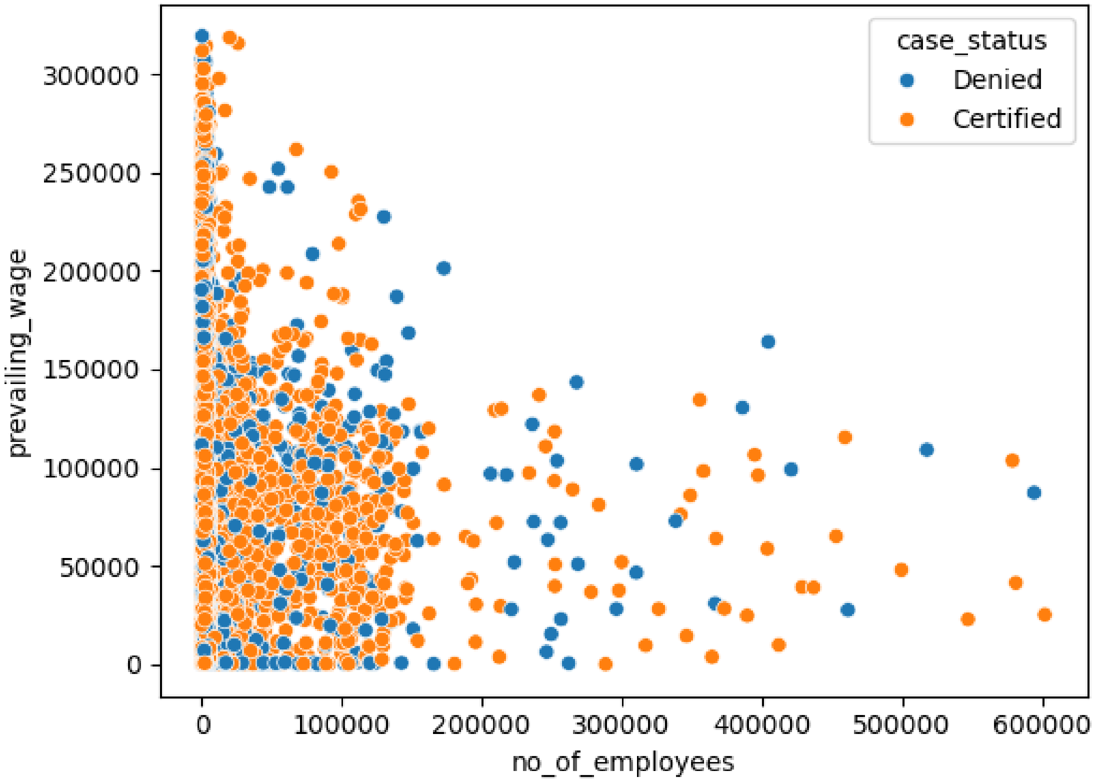
[14]:
```python
num_cols1 = visa_df.select_dtypes(exclude="object").columns.to_list()
num_cols2 = num_cols1.copy()
num_cols2
```

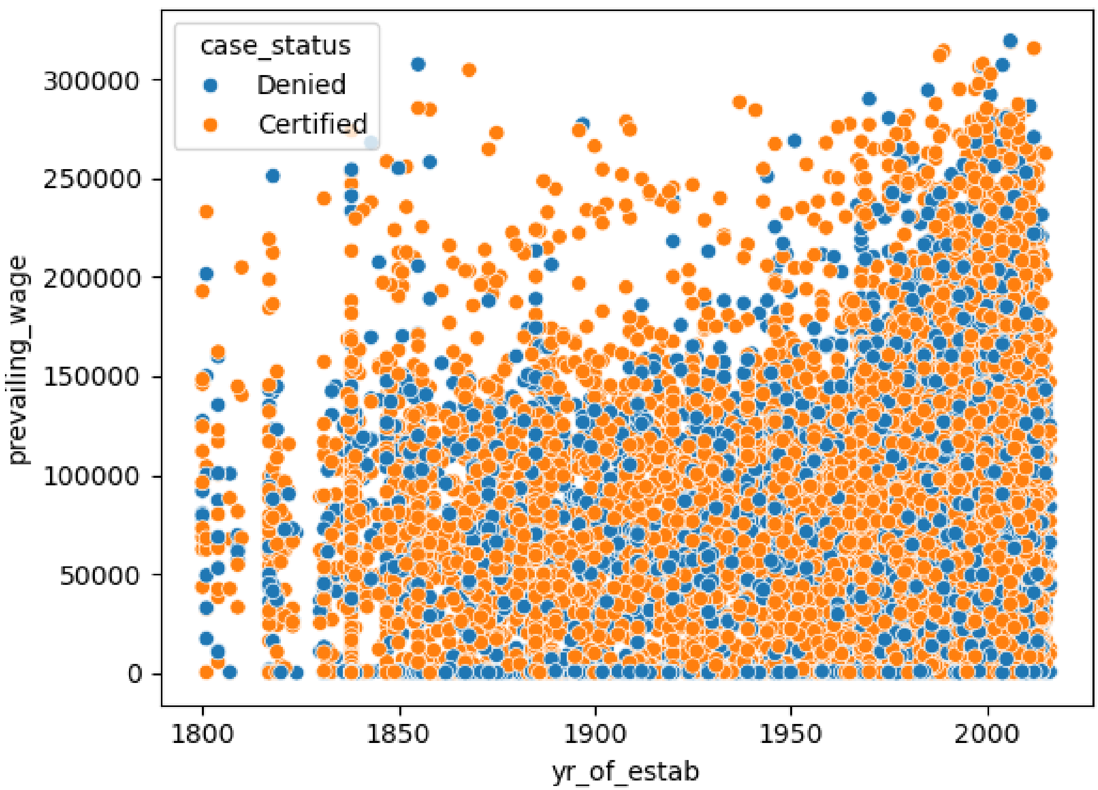
[14]: ['no_of_employees', 'yr_of_estab', 'prevailing_wage']

[15]:
```python
for i in num_cols1:
    for j in num_cols2:
        if i == j:
            pass
```

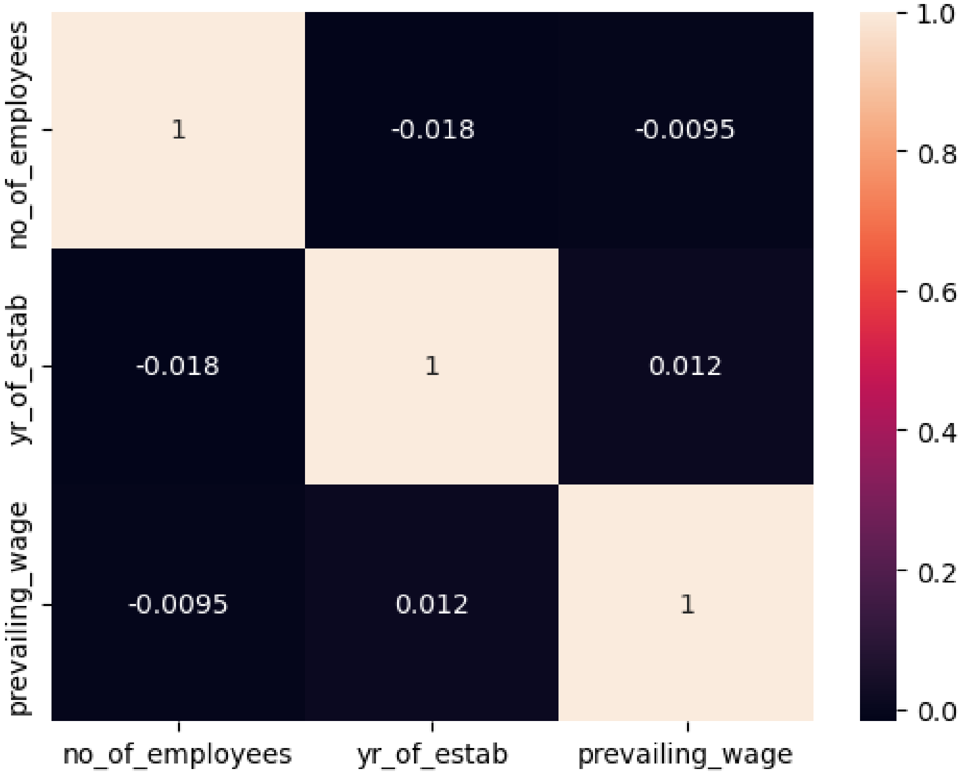
```
        else:
            sns.scatterplot(x=i,y=j,hue=target,data=visa_df)
            plt.show()
    num_cols2.pop(0)
```



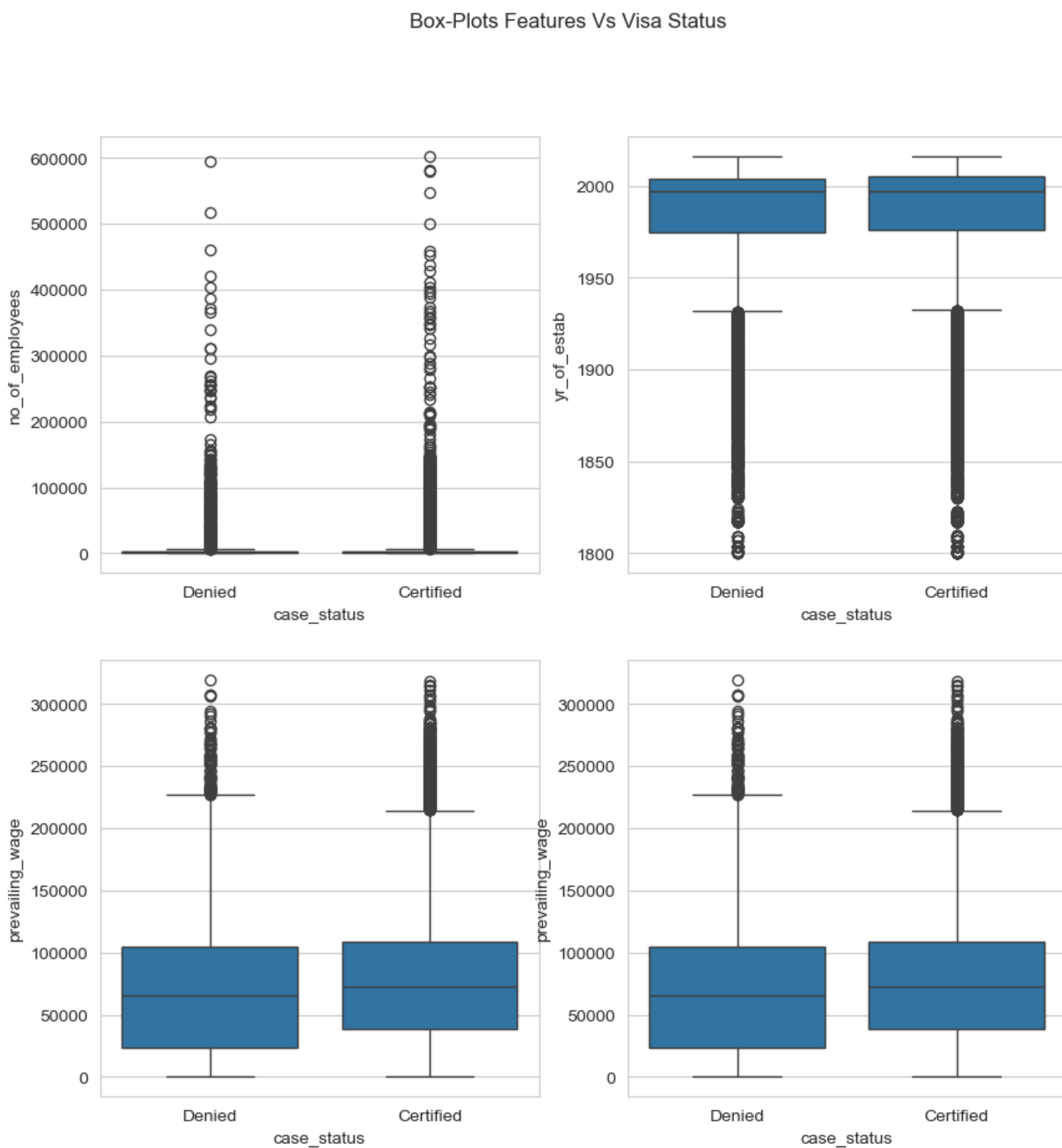

case_status
Denied
Certified
prevailing_wage
no_of_employees
0
50000
100000
150000
200000
250000
300000
0
100000
200000
300000
400000
500000
600000



[16]:
```
corr_mat=visa_df.corr(numeric_only=True)
sns.heatmap(corr_mat,annot=True)
plt.show()
```

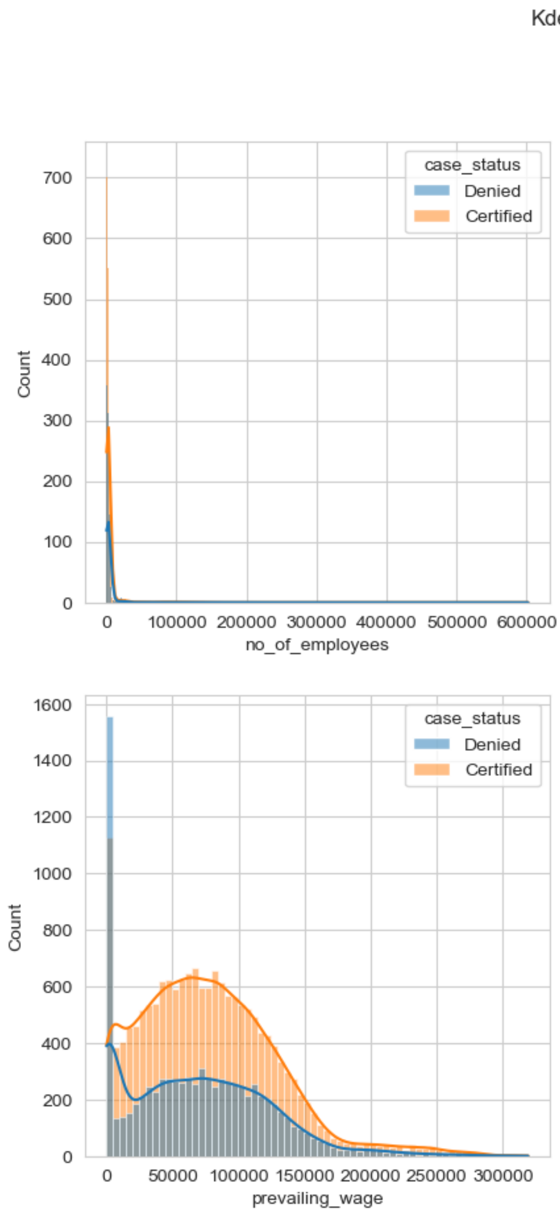


[17]:
```
output_var=target
sns.set_style("whitegrid")
fig, axes = plt.subplots(2, 2, figsize=(10, 10))
fig.suptitle('Box-Plots Features Vs Visa Status')
sns.boxplot(ax=axes[0, 0], x=output_var, y='no_of_employees', data=visa_df)
sns.boxplot(ax=axes[0, 1], x=output_var, y='yr_of_estab', data=visa_df)
sns.boxplot(ax=axes[1, 0], x=output_var, y='prevailing_wage', data=visa_df)
sns.boxplot(ax=axes[1, 1], x=output_var, y='prevailing_wage', data=visa_df)
plt.show()
```



[18]:
```
sns.set_style("whitegrid")
fig, axes = plt.subplots(2, 2, figsize=(10, 10))
fig.suptitle('Kde-Plots')
sns.histplot(ax=axes[0, 0], hue=output_var, x='no_of_employees',␣
    ↪data=visa_df,kde=True)
sns.histplot(ax=axes[0, 1], hue=output_var, x='yr_of_estab',␣
    ↪data=visa_df,kde=True)
sns.histplot(ax=axes[1, 0], hue=output_var, x='prevailing_wage',␣
    ↪data=visa_df,kde=True)
```

```
sns.histplot(ax=axes[1, 1], hue=output_var, x='prevailing_wage',␣
 ↪data=visa_df,kde=True)
plt.show()
```



14 10. Data Transformation

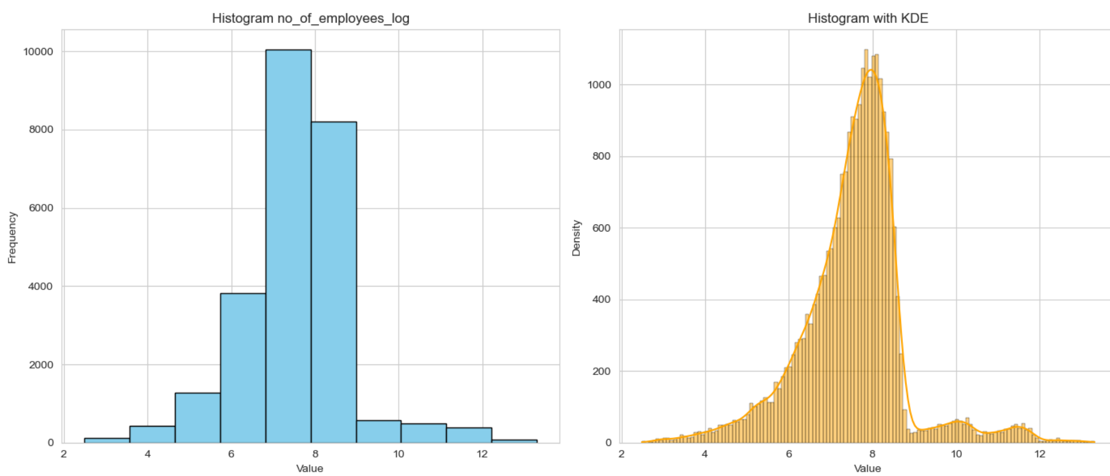
[19]:
```
visa_df.select_dtypes(exclude='object').columns
```

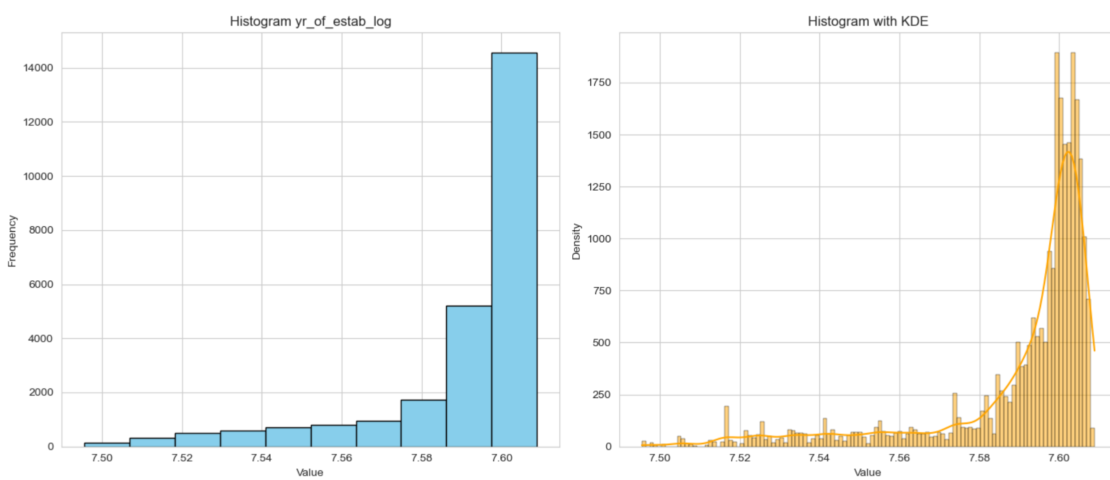
[19]:
```
Index(['no_of_employees', 'yr_of_estab', 'prevailing_wage'], dtype='object')
```

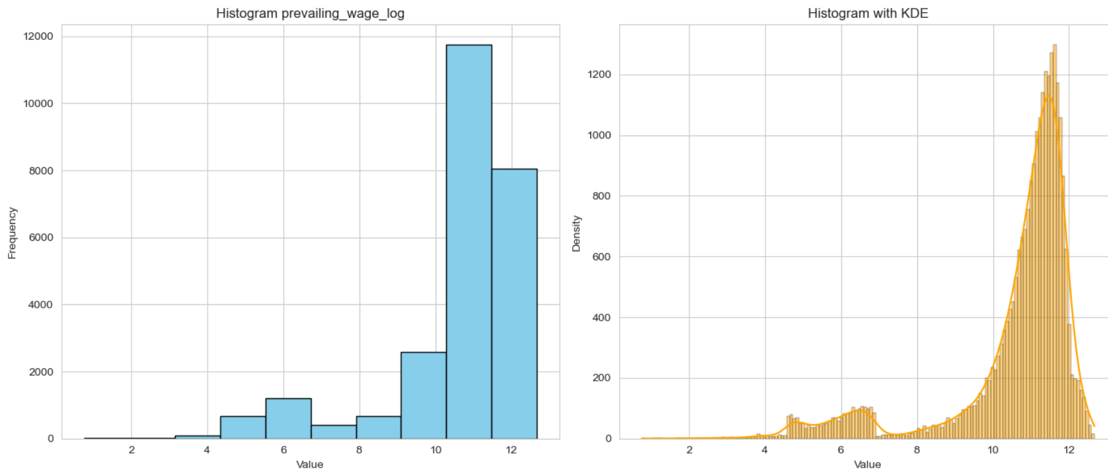
[20]:

```
visa_df["no_of_employees_log"]=np.log(visa_df["no_of_employees"])
visa_df["yr_of_estab_log"]=np.log(visa_df["yr_of_estab"])
visa_df["prevailing_wage_log"]=np.log(visa_df["prevailing_wage"])


visa_num_df = visa_df[['no_of_employees_log','yr_of_estab_log','prevailing_wage_log']].copy()
# Fit and transform your data using the pipeline
processed_data = pipeline_num_var.fit_transform(visa_num_df)
```

```
C:\Users\srishanm\AppData\Local\anaconda3\Lib\site-
packages\pandas\core\arraylike.py:396: RuntimeWarning: invalid value encountered
in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

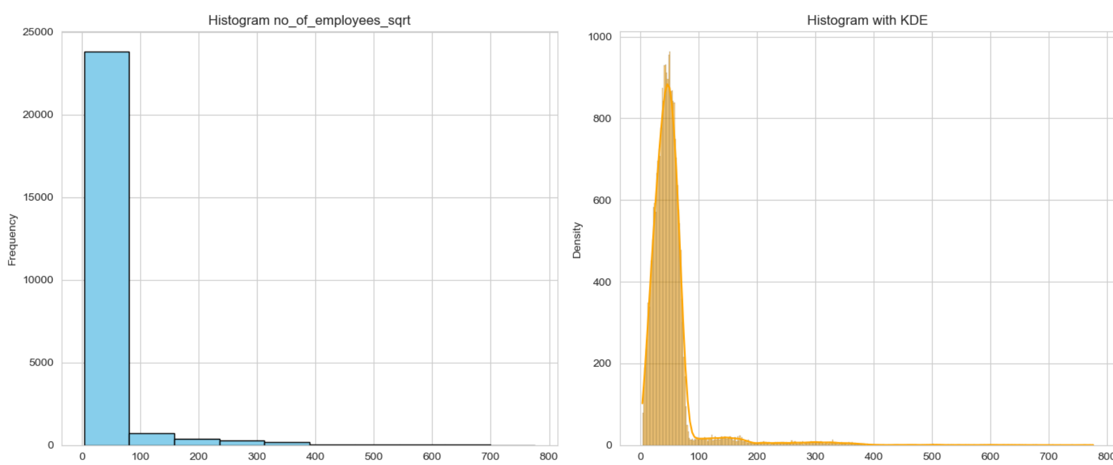


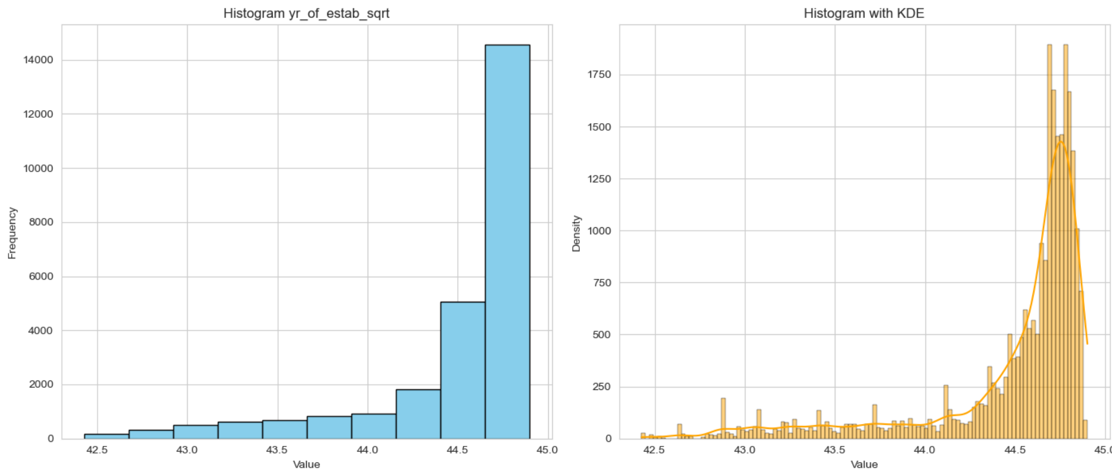


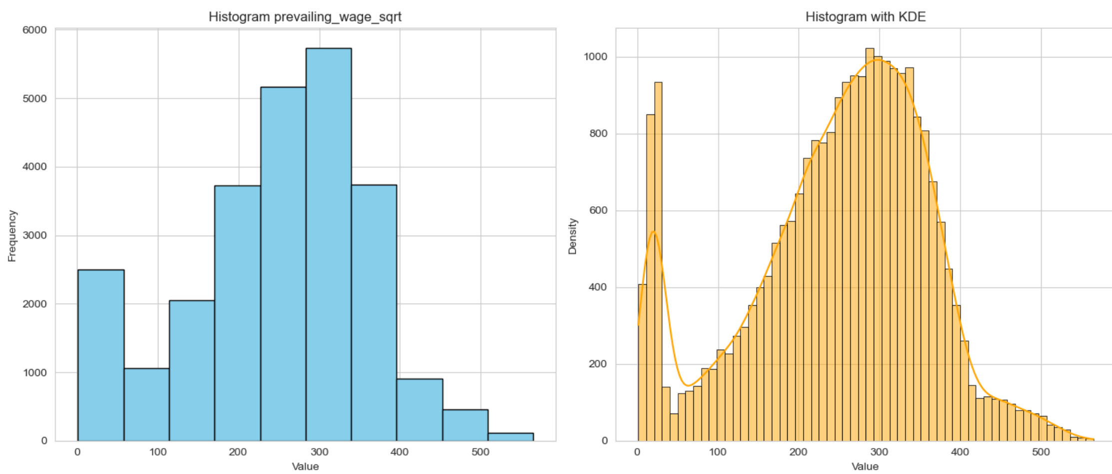



```
[21]: visa_df["no_of_employees_sqrt"]=np.sqrt(visa_df["no_of_employees"])
visa_df["yr_of_estab_sqrt"]=np.sqrt(visa_df["yr_of_estab"])
visa_df["prevailing_wage_sqrt"]=np.sqrt(visa_df["prevailing_wage"])


visa_num_df = visa_df[['no_of_employees_sqrt','yr_of_estab_sqrt',␣
↪'prevailing_wage_sqrt']].copy()
# Fit and transform your data using the pipeline
processed_data = pipeline_num_var.fit_transform(visa_num_df)
```

```
C:\Users\srishanm\AppData\Local\anaconda3\Lib\site-
packages\pandas\core\arraylike.py:396: RuntimeWarning: invalid value encountered
in sqrt
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

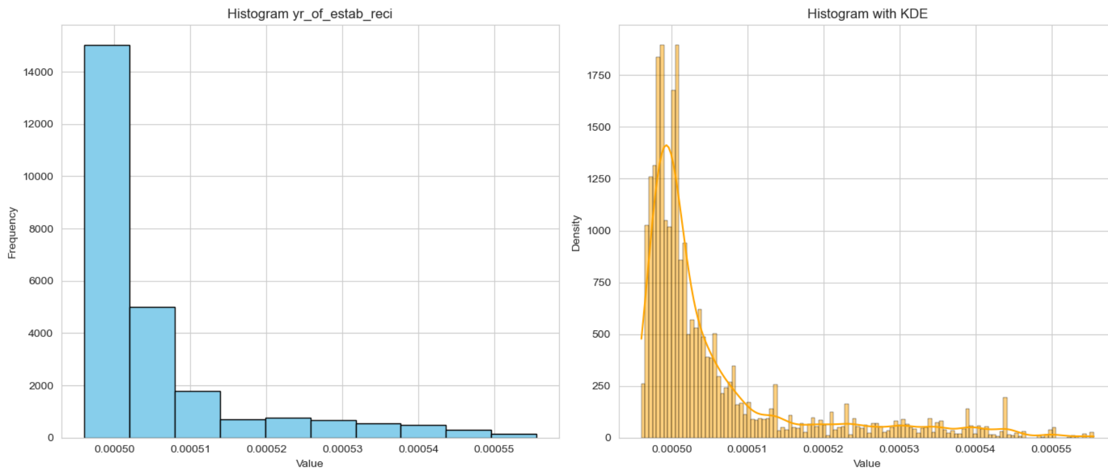






[22]:
```
visa_df["yr_of_estab_reci"]=1/(visa_df["yr_of_estab"])
```

[23]:
```
visa_num_df = visa_df[['yr_of_estab_reci']].copy()
# Fit and transform your data using the pipeline
processed_data = pipeline_num_var.fit_transform(visa_num_df)
```



15 11. Standization - Normalization

[24]:
```
scaler = StandardScaler()

# Fit and transform the scaler on the selected columns
scaled_columns = scaler.
↪fit_transform(visa_df[['no_of_employees_log','yr_of_estab_log',␣
↪'prevailing_wage_sqrt']])

# Replace the original columns with the scaled columns
visa_df[['no_of_employees_log_stand','yr_of_estab_log_stand',␣
↪'prevailing_wage_sqrt_stand']] = scaled_columns

print(visa_df)
```

```
      continent education_of_employee has_job_experience  \
0          Asia           High School                  N
1          Asia              Master's                  Y
2          Asia            Bachelor's                  N
3          Asia            Bachelor's                  N
4        Africa              Master's                  Y
…           …                   …                 …
25475      Asia            Bachelor's                  Y
25476      Asia           High School                  Y
25477      Asia              Master's                  Y
25478      Asia              Master's                  Y
25479      Asia            Bachelor's                  Y

      requires_job_training  no_of_employees  yr_of_estab  \
0                         N            14513         2007
```

```
1                        N          2412         2002
2                        Y         44444         2008
3                        N            98         1897
4                        N          1082         2005
…                      …             …          …
25475                    Y          2601         2008
25476                    N          3274         2006
25477                    N          1121         1910
25478                    Y          1918         1887
25479                    N          3195         1960

      region_of_employment  prevailing_wage unit_of_wage full_time_position  \
0                     West         592.2029         Hour                  Y
1                Northeast       83425.6500         Year                  Y
2                     West      122996.8600         Year                  Y
3                     West       83434.0300         Year                  Y
4                    South      149907.3900         Year                  Y
…                      …             …          …                 …
25475                South       77092.5700         Year                  Y
25476            Northeast      279174.7900         Year                  Y
25477                South      146298.8500         Year                  N
25478                 West       86154.7700         Year                  Y
25479              Midwest       70876.9100         Year                  Y

       … no_of_employees_log  yr_of_estab_log  prevailing_wage_log  \
0      …            9.582800         7.604396             6.383849
1      …            7.788212         7.601902            11.331711
2      …           10.701985         7.604894            11.719914
3      …            4.584967         7.548029            11.331812
4      …            6.986566         7.603399            11.917773
…    …              …                …                …
25475  …            7.863651         7.604894            11.252762
25476  …            8.093768         7.603898            12.539593
25477  …            7.021976         7.554859            11.893407
25478  …            7.559038         7.542744            11.363901
25479  …            8.069342         7.580700            11.168700

       no_of_employees_sqrt  yr_of_estab_sqrt  prevailing_wage_sqrt  \
0                120.469913         44.799554             24.335219
1                 49.112117         44.743715            288.834987
2                210.817457         44.810713            350.709082
3                  9.899495         43.554563            288.849494
4                 32.893768         44.777226            387.178757
…                      …                …                 …
25475             51.000000         44.810713            277.655488
25476             57.218878         44.788391            528.369937
25477             33.481338         43.703547            382.490327
25478             43.794977         43.439613            293.521328
```

```
25479            56.524331         44.271887           266.227177

       yr_of_estab_reci  no_of_employees_log_stand  yr_of_estab_log_stand  \
0              0.000498                   1.633418               0.643017
1              0.000500                   0.200370               0.529088
2              0.000498                   2.527131               0.665769
3              0.000527                  -2.357543              -1.931516
4              0.000499                  -0.439774               0.597480
…                   …                         …                      …
25475          0.000498                   0.260612               0.665769
25476          0.000499                   0.444369               0.620254
25477          0.000524                  -0.411498              -1.619582
25478          0.000530                   0.017367              -2.172923
25479          0.000510                   0.424864              -0.439305

       prevailing_wage_sqrt_stand
0                       -1.990753
1                        0.357791
2                        0.907183
3                        0.357920
4                        1.231004
…                             …
25475                    0.258526
25476                    2.484668
25477                    1.189375
25478                    0.399402
25479                    0.157052

[25480 rows x 21 columns]
```

16 12. Convert Cat - to - Numerical Columns

[25]:
```
cat_vars = visa_df.select_dtypes(include='object').columns
cat_vars
```

[25]:
```
Index(['continent', 'education_of_employee', 'has_job_experience',
       'requires_job_training', 'region_of_employment', 'unit_of_wage',
       'full_time_position', 'case_status'],
      dtype='object')
```

[26]:
```
from sklearn.preprocessing import LabelEncoder
for var in cat_vars:
    le = LabelEncoder()
    visa_df[var]=le.fit_transform(visa_df[var])
```

17 13. SMOTE for Balancing Data

[27]:
```
visa_df.columns
```

[27]:
```
Index(['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_status', 'no_of_employees_log',
       'yr_of_estab_log', 'prevailing_wage_log', 'no_of_employees_sqrt',
       'yr_of_estab_sqrt', 'prevailing_wage_sqrt', 'yr_of_estab_reci',
       'no_of_employees_log_stand', 'yr_of_estab_log_stand',
       'prevailing_wage_sqrt_stand'],
      dtype='object')
```

[28]:
```
visa_df.dropna(inplace=True)
```

[29]:
```
Y=visa_df["case_status"]
X=visa_df[[ 'continent', 'education_of_employee', 'has_job_experience',
       'requires_job_training',
       'region_of_employment', 'unit_of_wage',
       'full_time_position', 'no_of_employees_log_stand',␣
 ↪'yr_of_estab_log_stand',
       'prevailing_wage_sqrt_stand']]
print(len(Y),len(X))
print(X.columns)
```

```
25447 25447
Index(['continent', 'education_of_employee', 'has_job_experience',
       'requires_job_training', 'region_of_employment', 'unit_of_wage',
       'full_time_position', 'no_of_employees_log_stand',
       'yr_of_estab_log_stand', 'prevailing_wage_sqrt_stand'],
      dtype='object')
```

[30]:
```
X, Y = SMOTE().fit_resample(X, Y)
print(X.columns)
print(len(Y),len(X))
```

```
Index(['continent', 'education_of_employee', 'has_job_experience',
       'requires_job_training', 'region_of_employment', 'unit_of_wage',
       'full_time_position', 'no_of_employees_log_stand',
       'yr_of_estab_log_stand', 'prevailing_wage_sqrt_stand'],
      dtype='object')
34002 34002
```

[31]:
```
Y.value_counts()
```

[31]:
```
case_status
1    17001
```

```
0    17001
Name: count, dtype: int64
```

18 ML Models

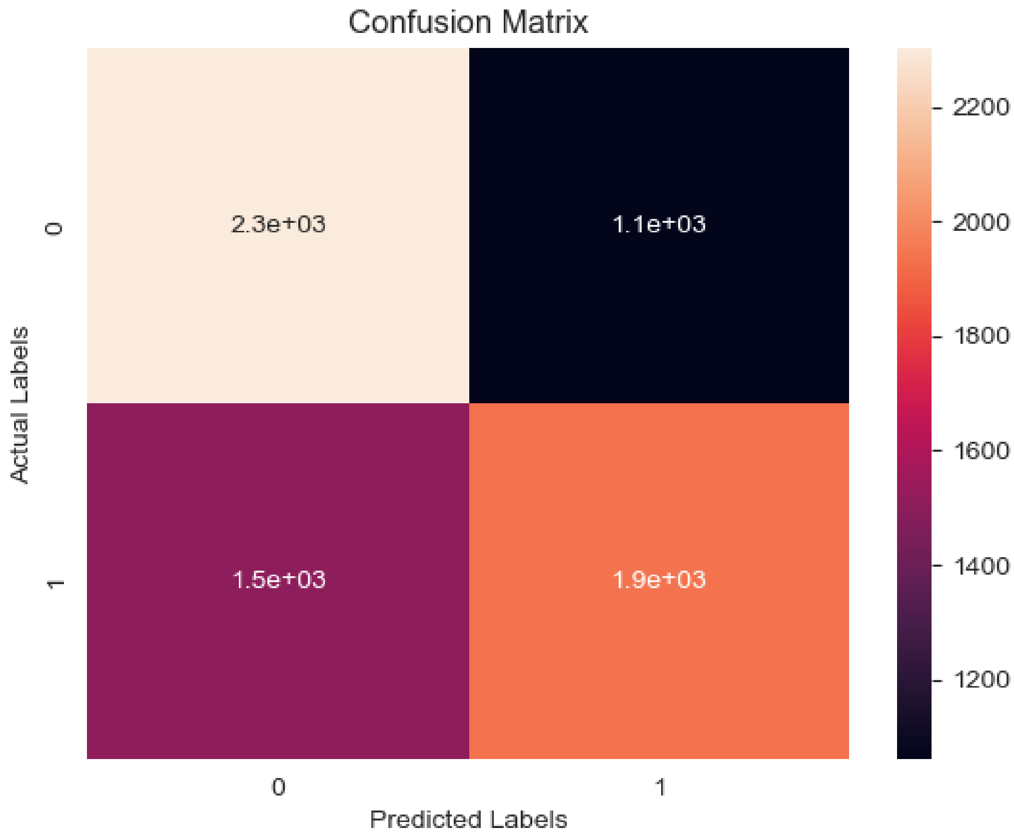
[32]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.
 ↪2,random_state = 42)
```

[33]:
```
def draw_heatmap(conf_matrix):
    sns.heatmap(conf_matrix, annot=True)
    plt.xlabel('Predicted Labels')
    plt.ylabel('Actual Labels')
    plt.title('Confusion Matrix')
    plt.show()
```

18.1 Logistic Regression

[34]:
```
lg_model = LogisticRegression(solver='saga', max_iter=500, random_state=42)
lg_model.fit(X_train, Y_train)

print("Model - Logistic Regression")
score = lg_model.score(X_train, Y_train)
print('accuracy train score overall :', score)
score = lg_model.score(X_test, Y_test)
print('accuracy test score overall :', score)

y_pred = lg_model.predict(X_test)
print(classification_report(Y_test, y_pred))
print(confusion_matrix(Y_test, y_pred))
conf_matrix = confusion_matrix(Y_test, y_pred)
draw_heatmap(conf_matrix)
```

```
Model - Logistic Regression
accuracy train score overall : 0.632182640344105
accuracy test score overall : 0.6228495809439788
              precision    recall  f1-score   support

           0       0.60      0.68      0.64      3364
           1       0.65      0.56      0.60      3437

    accuracy                           0.62      6801
   macro avg       0.63      0.62      0.62      6801
weighted avg       0.63      0.62      0.62      6801

[[2302 1062]
 [1503 1934]]
```

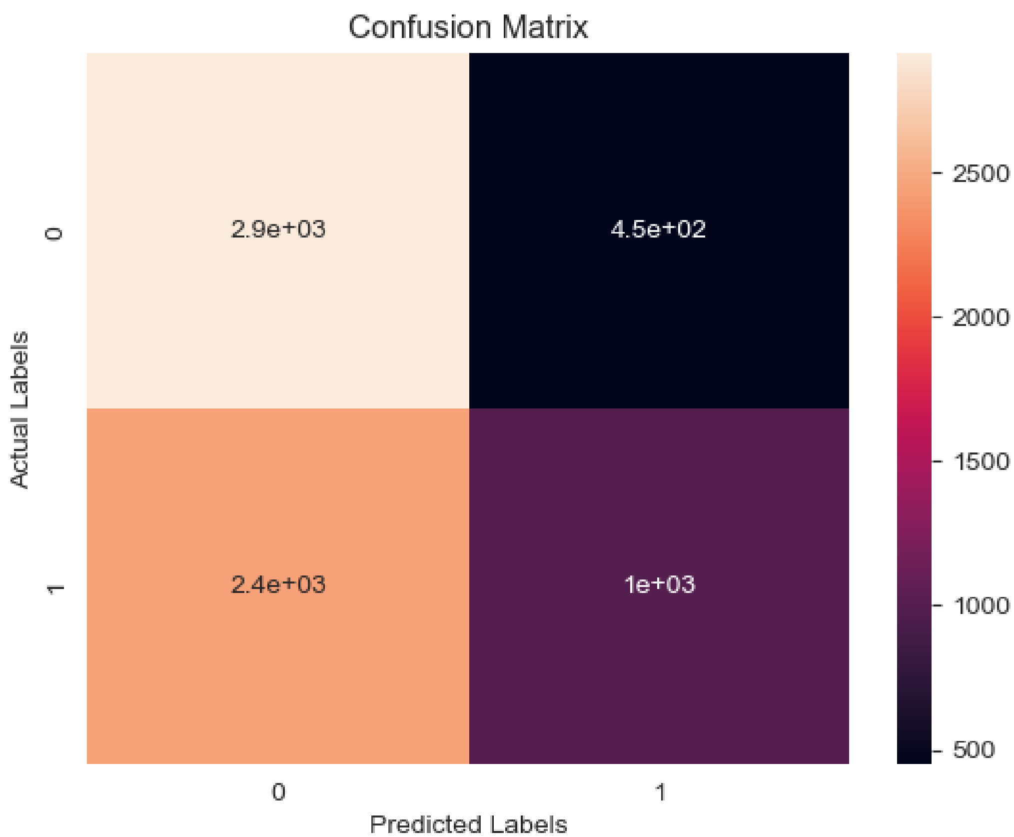


18.2 GaussianNB

[35]:
```
from sklearn.naive_bayes import GaussianNB, CategoricalNB
gnb_model = GaussianNB()
gnb_model.fit(X_train,Y_train)

print("Model-GaussianNB")
print("train score",gnb_model.score(X_train,Y_train))
print("test score",gnb_model.score(X_test,Y_test))

y_pred = gnb_model.predict(X_test)
print(classification_report(Y_test, y_pred))
print(confusion_matrix(Y_test, y_pred))
conf_matrix = confusion_matrix(Y_test, y_pred)
draw_heatmap(conf_matrix)
```

```
Model-GaussianNB
train score 0.5835447226204918
test score 0.5744743420085282
              precision    recall  f1-score   support
```

```
              0       0.54      0.87      0.67      3364
              1       0.69      0.29      0.41      3437

       accuracy                           0.57      6801
      macro avg       0.62      0.58      0.54      6801
   weighted avg       0.62      0.57      0.54      6801

[[2912  452]
 [2442  995]]
```

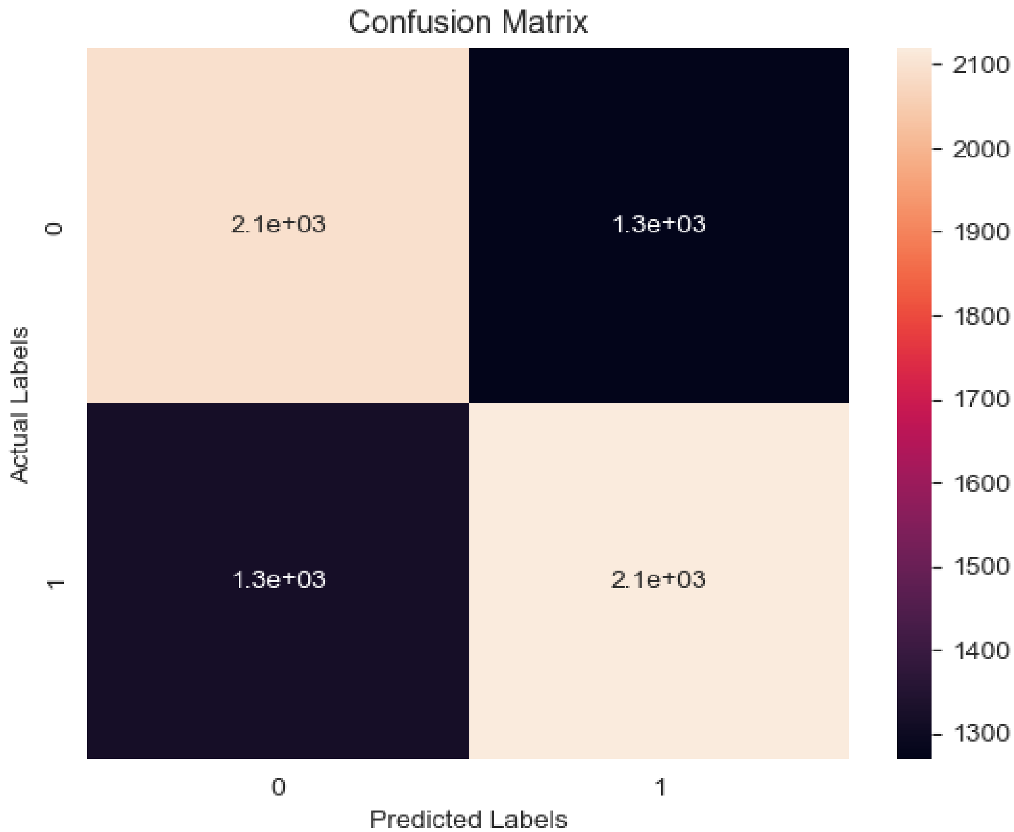


19 Suport Vector Machine - Classifier

[36]:
```
from sklearn.svm import SVC
# Initialize the SVM classifier
svm_linear_classifier = SVC(kernel='linear', random_state=42)

# Train the SVM classifier
svm_linear_classifier.fit(X_train, Y_train)
print("model-Suport Vector Machine - kernel - linear -Classifier")
```

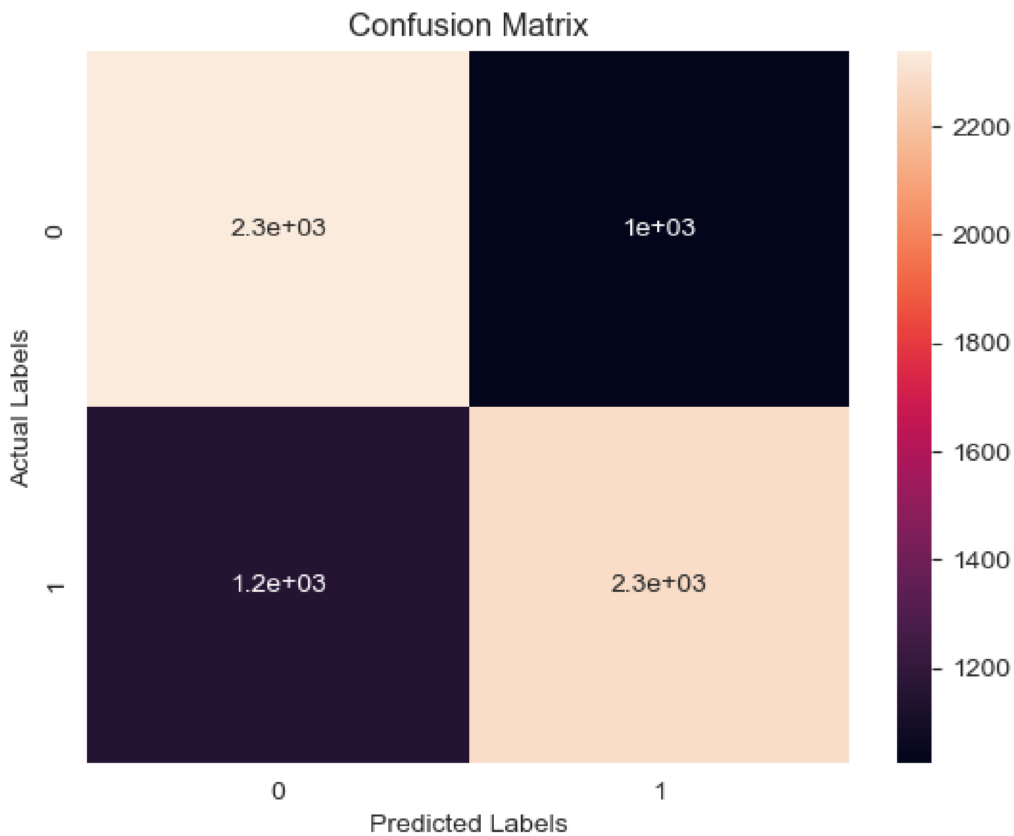
```python
y_pred = svm_linear_classifier.predict(X_train)
# Calculate the accuracy of the model
accuracy = accuracy_score(Y_train, y_pred)
print("Train Accuracy:", accuracy)

# Predict the classes for test set
y_pred = svm_linear_classifier.predict(X_test)
# Calculate the accuracy of the model
accuracy = accuracy_score(Y_test, y_pred)
print("Test Accuracy:", accuracy)

print(classification_report(Y_test, y_pred))
print(confusion_matrix(Y_test, y_pred))
conf_matrix = confusion_matrix(Y_test, y_pred)
draw_heatmap(conf_matrix)
```

```
model-Suport Vector Machine - kernel - linear -Classifier
Train Accuracy: 0.6231388551891475
Test Accuracy: 0.6194677253345097
              precision    recall  f1-score   support

           0       0.61      0.62      0.62      3364
           1       0.63      0.62      0.62      3437

    accuracy                           0.62      6801
   macro avg       0.62      0.62      0.62      6801
weighted avg       0.62      0.62      0.62      6801

[[2094 1270]
 [1318 2119]]
```



[37]:
```
svm_rbf_classifier = SVC(kernel='rbf', random_state=42)

# Train the SVM classifier
svm_rbf_classifier.fit(X_train, Y_train)
print("model-Suport Vector Machine - Kernel -rbf - Classifier")
y_pred = svm_rbf_classifier.predict(X_train)
# Calculate the accuracy of the model
accuracy = accuracy_score(Y_train, y_pred)
print("Train Accuracy:", accuracy)

# Predict the classes for test set
y_pred = svm_rbf_classifier.predict(X_test)
# Calculate the accuracy of the model
accuracy = accuracy_score(Y_test, y_pred)
print("Test Accuracy:", accuracy)

print(classification_report(Y_test, y_pred))
print(confusion_matrix(Y_test, y_pred))
conf_matrix = confusion_matrix(Y_test, y_pred)
draw_heatmap(conf_matrix)
```

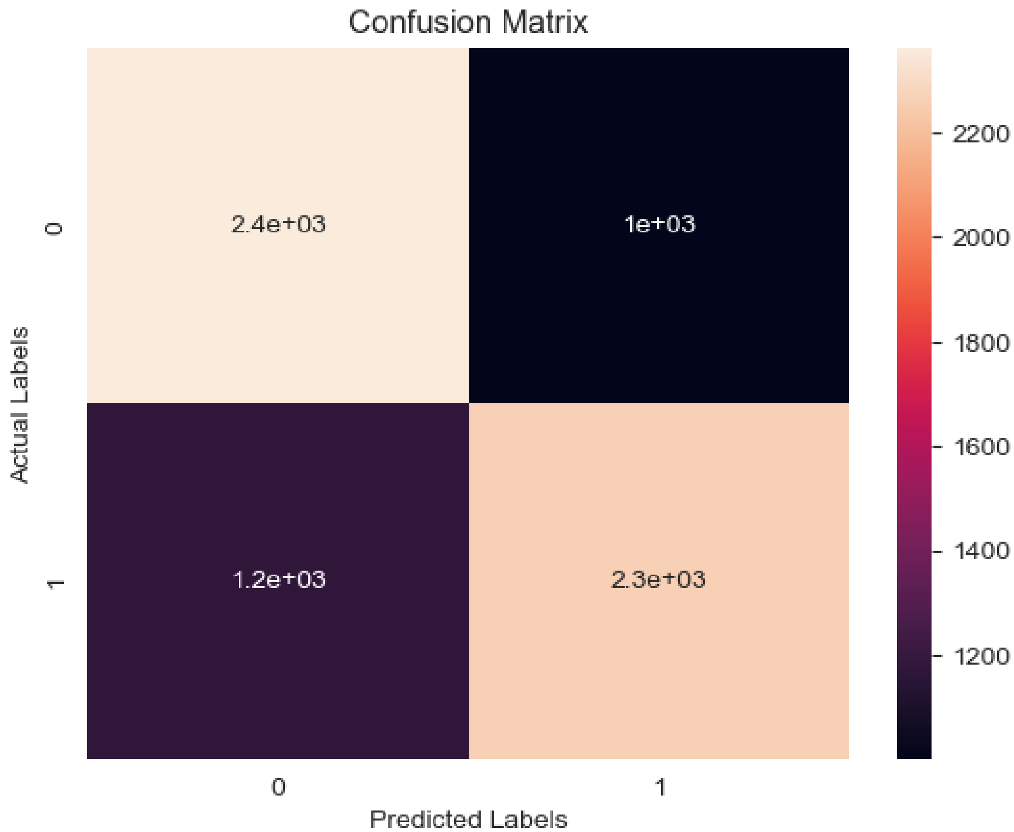
```
model-Suport Vector Machine - Kernel -rbf - Classifier
Train Accuracy: 0.6752692915701629
Test Accuracy: 0.6800470519041317
              precision    recall  f1-score   support

           0       0.67      0.70      0.68      3364
           1       0.69      0.67      0.68      3437

    accuracy                           0.68      6801
   macro avg       0.68      0.68      0.68      6801
weighted avg       0.68      0.68      0.68      6801

[[2338 1026]
 [1150 2287]]
```



[38]:
```
svm_poly_classifier = SVC(kernel='poly', random_state=42)

# Train the SVM classifier
svm_poly_classifier.fit(X_train, Y_train)
print("model-Suport Vector Machine - Kernel -poly - Classifier")
```

```python
y_pred = svm_poly_classifier.predict(X_train)
# Calculate the accuracy of the model
accuracy = accuracy_score(Y_train, y_pred)
print("Train Accuracy:", accuracy)

# Predict the classes for test set
y_pred = svm_poly_classifier.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(Y_test, y_pred)
print("Test Accuracy:", accuracy)

print(classification_report(Y_test, y_pred))
print(confusion_matrix(Y_test, y_pred))
conf_matrix = confusion_matrix(Y_test, y_pred)
draw_heatmap(conf_matrix)
```

```
model-Suport Vector Machine - Kernel -poly - Classifier
Train Accuracy: 0.6733575971471637
Test Accuracy: 0.6796059403028967
              precision    recall  f1-score   support

           0       0.67      0.70      0.68      3364
           1       0.69      0.66      0.67      3437

    accuracy                           0.68      6801
   macro avg       0.68      0.68      0.68      6801
weighted avg       0.68      0.68      0.68      6801

[[2361 1003]
 [1176 2261]]
```

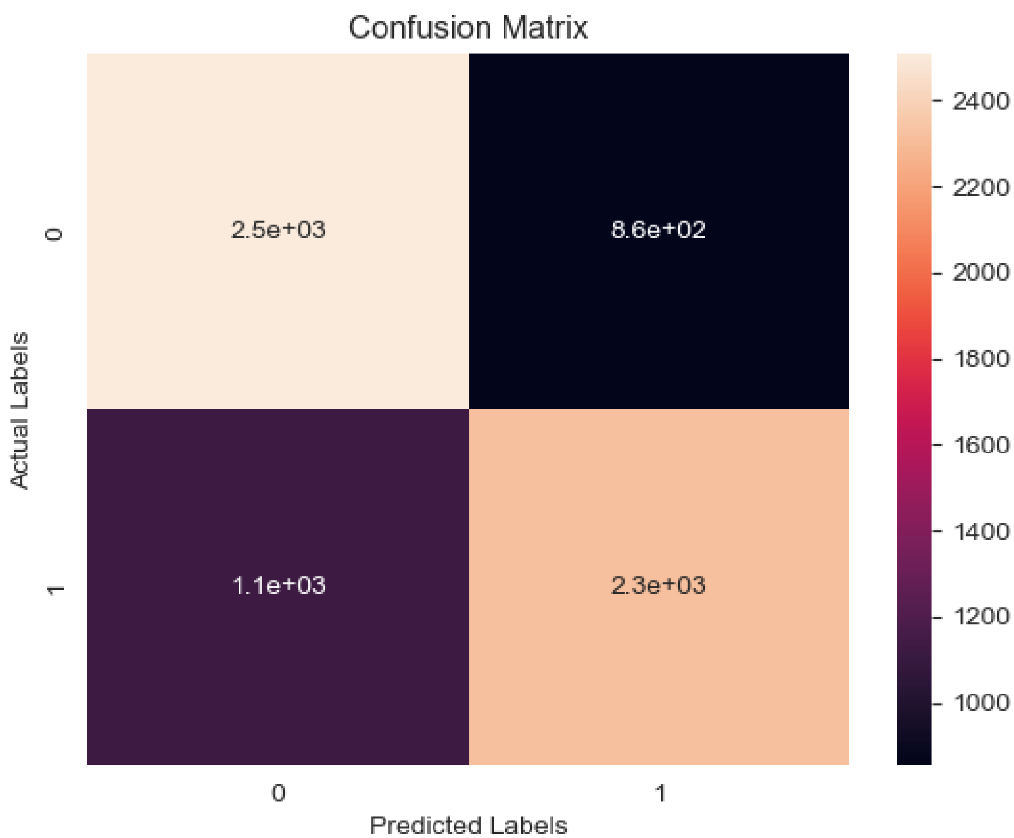


19.1 Decision Tree

[39]:
```
dt_clf = DecisionTreeClassifier(max_leaf_nodes=20,random_state=42)
dt_clf.fit(X_train, Y_train)
print("Model-Decion Tree")

accuracy=dt_clf.score(X_train, Y_train)
print(f"train score: {accuracy}")

accuracy=dt_clf.score(X_test, Y_test)
print(f"test score: {accuracy}")

y_pred=dt_clf.predict(X_test)
print(classification_report(Y_test, y_pred))
print(confusion_matrix(Y_test, y_pred))
conf_matrix = confusion_matrix(Y_test, y_pred)
draw_heatmap(conf_matrix)
```

```
Model-Decion Tree
train score: 0.7029888607036506
test score: 0.709748566387296
```

```
              precision    recall  f1-score   support

           0       0.69      0.75      0.72      3364
           1       0.73      0.67      0.70      3437

    accuracy                           0.71      6801
   macro avg       0.71      0.71      0.71      6801
weighted avg       0.71      0.71      0.71      6801

[[2508  856]
 [1118 2319]]
```

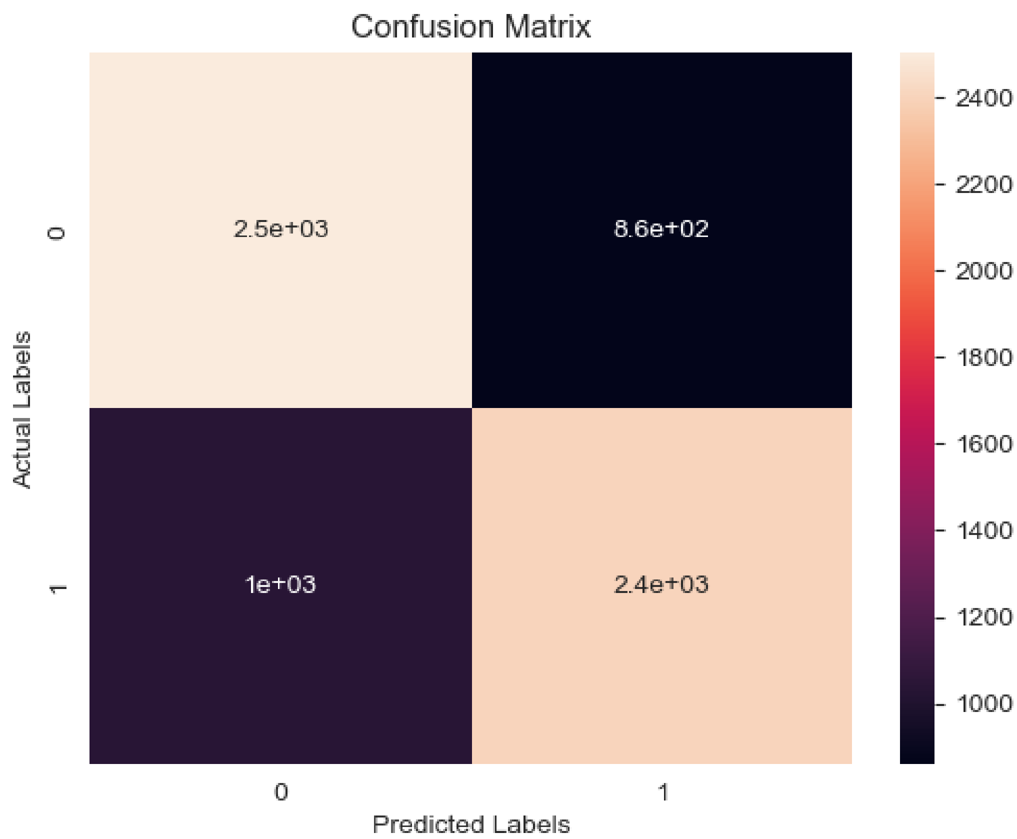


19.2 Random Forest

[40]:
```
rf_clf= RandomForestClassifier(n_estimators = 1000, random_state = 42,␣
  ↪max_leaf_nodes=20)
rf_clf.fit(X_train, Y_train)
print("Model- Random Forest Tree")

accuracy=rf_clf.score(X_train, Y_train)
```

```
print(f"train score: {accuracy}")

accuracy=rf_clf.score(X_test, Y_test)
print(f"test score: {accuracy}")

y_pred=rf_clf.predict(X_test)
print(classification_report(Y_test, y_pred))
print(confusion_matrix(Y_test, y_pred))
conf_matrix = confusion_matrix(Y_test, y_pred)
draw_heatmap(conf_matrix)
```

```
Model- Random Forest Tree
train score: 0.7099003713098783
test score: 0.7209233936185855
              precision    recall  f1-score   support

           0       0.71      0.74      0.73      3364
           1       0.74      0.70      0.72      3437

    accuracy                           0.72      6801
   macro avg       0.72      0.72      0.72      6801
weighted avg       0.72      0.72      0.72      6801

[[2502  862]
 [1036 2401]]
```

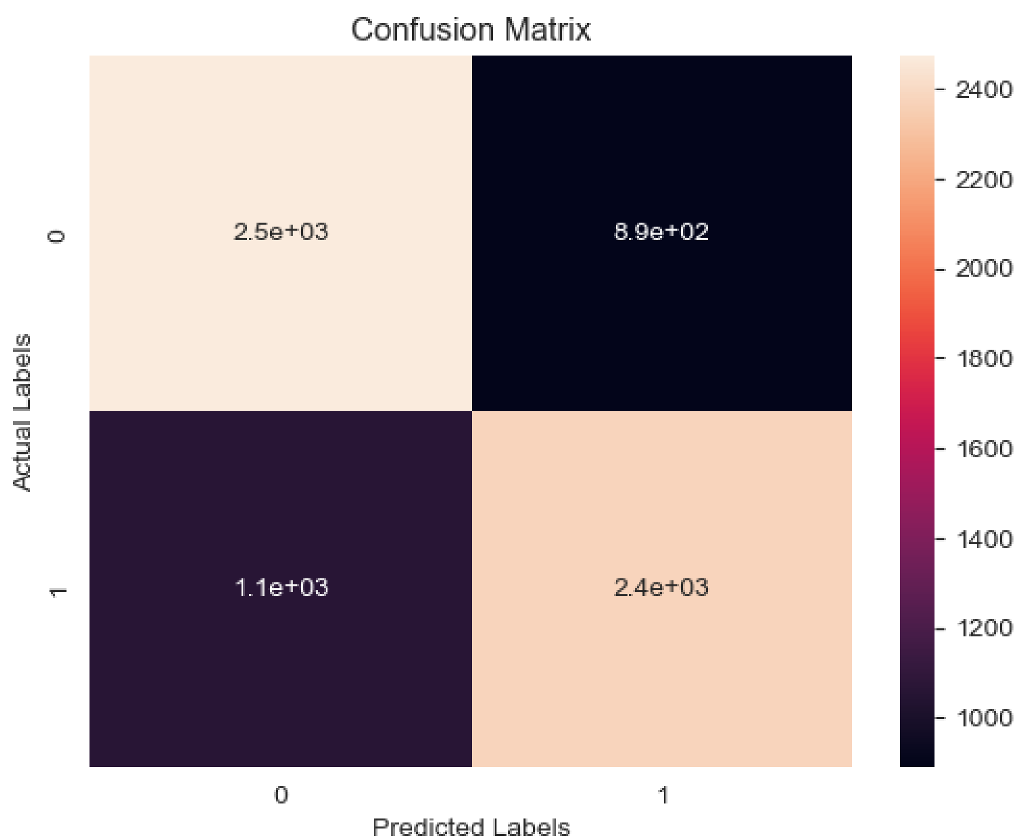


19.3 AdaBoost

[41]:
```
base_classifier = DecisionTreeClassifier(max_depth=1)
adaboost_clf = AdaBoostClassifier( n_estimators=50, random_state=42)

# Train the AdaBoost classifier
adaboost_clf.fit(X_train, Y_train)

print("Model-AdaBoost")
print("train score",adaboost_clf.score(X_train, Y_train))

# Predict on the test set
y_pred = adaboost_clf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(Y_test, y_pred)
print(f"test score: {accuracy}")

print(classification_report(Y_test, y_pred))
print(confusion_matrix(Y_test, y_pred))
```

```
conf_matrix = confusion_matrix(Y_test, y_pred)
draw_heatmap(conf_matrix)
```

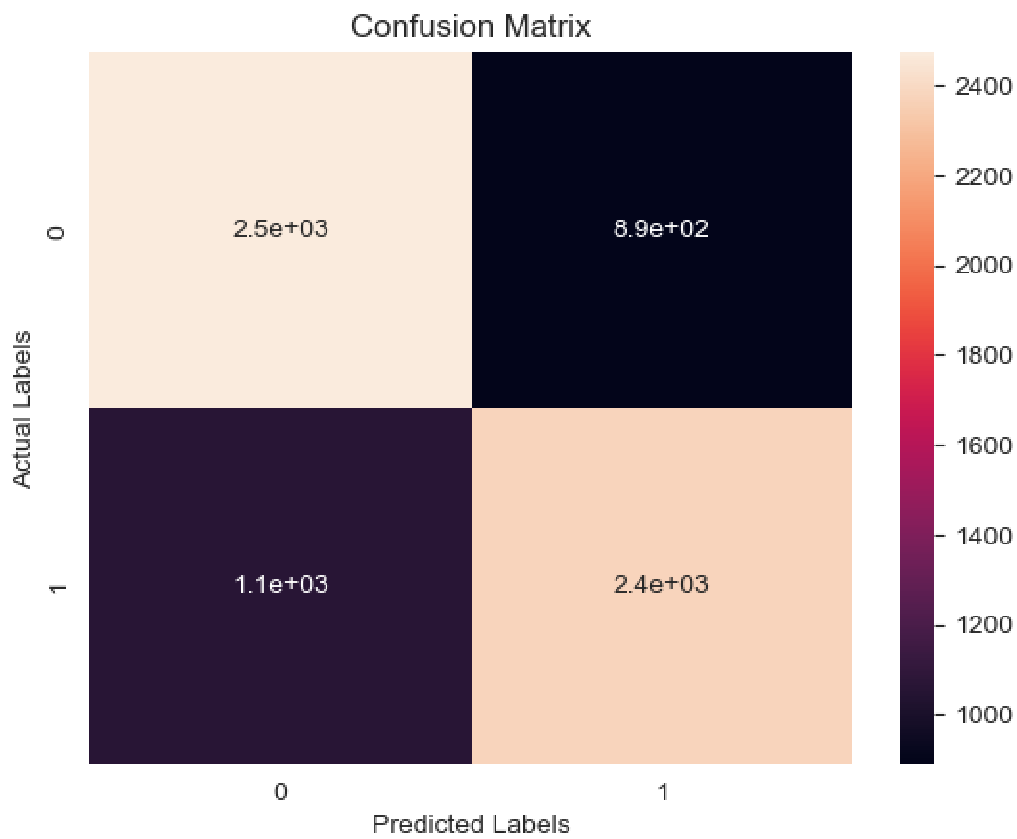
```
Model-AdaBoost
train score 0.703356494246535
test score: 0.7131304219967651
              precision    recall  f1-score   support

           0       0.70      0.74      0.72      3364
           1       0.73      0.69      0.71      3437

    accuracy                           0.71      6801
   macro avg       0.71      0.71      0.71      6801
weighted avg       0.71      0.71      0.71      6801

[[2473  891]
 [1060 2377]]
```



19.4 GradientBoostingClassifier

[42]:
```
gdb_clf = GradientBoostingClassifier(n_estimators=20, learning_rate=0.1,␣
 ↪max_depth=1, random_state=42)
gdb_clf.fit(X_train, Y_train)
print("model-Gradient Boosting Classifier")

accuracy = gdb_clf.score(X_train, Y_train)
print("Train Accuracy:", accuracy)

accuracy = gdb_clf.score(X_test, Y_test)
print("Test Accuracy:", accuracy)

print(classification_report(Y_test, y_pred))
print(confusion_matrix(Y_test, y_pred))
conf_matrix = confusion_matrix(Y_test, y_pred)
draw_heatmap(conf_matrix)
```

```
model-Gradient Boosting Classifier
Train Accuracy: 0.6283592514981067
Test Accuracy: 0.6196147625349213
              precision    recall  f1-score   support

           0       0.70      0.74      0.72      3364
           1       0.73      0.69      0.71      3437

    accuracy                           0.71      6801
   macro avg       0.71      0.71      0.71      6801
weighted avg       0.71      0.71      0.71      6801

[[2473  891]
 [1060 2377]]
```

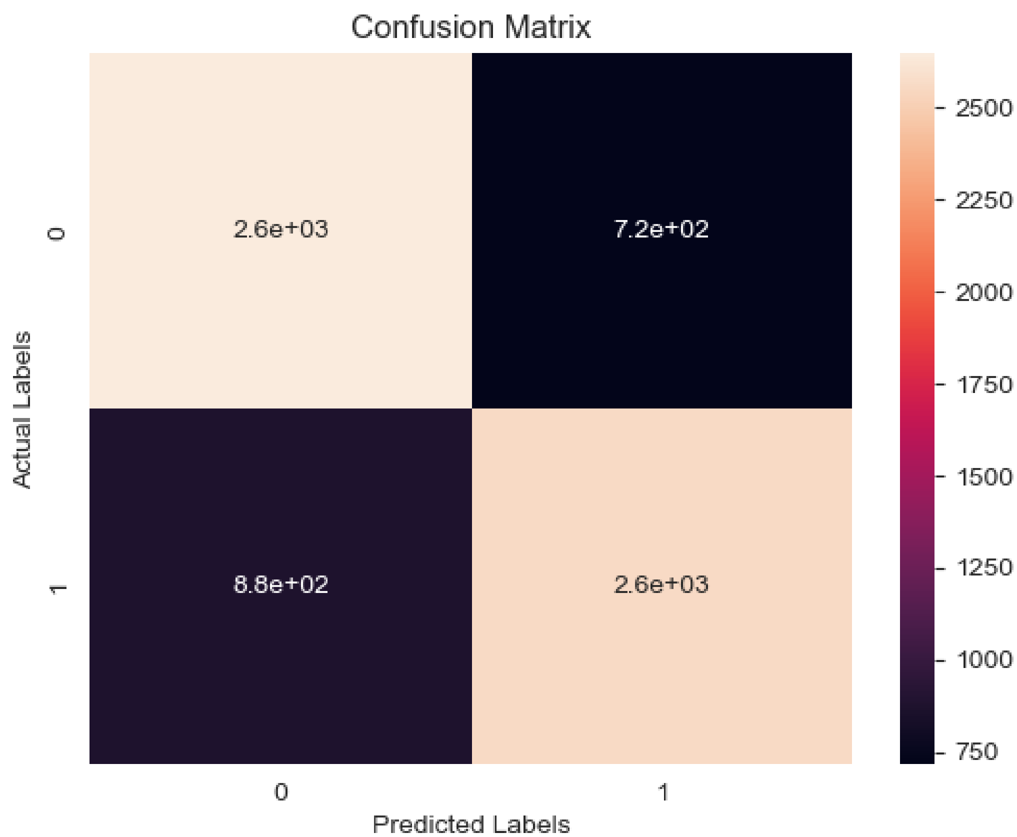


19.5 XGBClassifier

[43]:
```python
from xgboost import XGBClassifier
xgmodel = XGBClassifier()
xgmodel.fit(X_train, Y_train)


print("model- XGB Classifier")
# Make predictions on the test set
y_pred = xgmodel.predict(X_train)
accuracy = accuracy_score(Y_train, y_pred)
print("Train Accuracy:", accuracy)
# Evaluate the model

# Make predictions on the test set
y_pred = xgmodel.predict(X_test)
accuracy = accuracy_score(Y_test, y_pred)
print("Test Accuracy:", accuracy)

print(classification_report(Y_test, y_pred))
```

```
print(confusion_matrix(Y_test, y_pred))
conf_matrix = confusion_matrix(Y_test, y_pred)
draw_heatmap(conf_matrix)
```

```
model- XGB Classifier
Train Accuracy: 0.837101577147899
Test Accuracy: 0.7656227025437435
              precision    recall  f1-score   support

           0       0.75      0.79      0.77      3364
           1       0.78      0.75      0.76      3437

    accuracy                           0.77      6801
   macro avg       0.77      0.77      0.77      6801
weighted avg       0.77      0.77      0.77      6801

[[2646  718]
 [ 876 2561]]
```



[ ]: