H1-B Visa Status Predictor

Developed by:

Srujan Shah - 823336313

Hitayu Patel - 823342137

Abstract – The foreign workers can legally work in the United States for the big companies only if they have H1B visa filed and accepted. The report explains the approach of predicting the status of the H1-B visa of the foreign workers using various attributes mentioned in the Data Dictionary (please refer section III), and the H1-B Visa prediction Dataset from Kaggle. The objective of the experiment is to understand the impact of the different parameters of the dataset on the output.

I. INTRODUCTION

The H1-B visa is a non-immigrant visa that allows US companies to employ graduate level workers in specialty occupations that require theoretical or technical expertise. This is one of the highly used visa categories, as many companies require foreign talent and rely on it for their further development. The rate of the H1-B Visa petition filing has increased in the recent times which can be seen in the Figure 2. because of the economy affecting industries such as IT and Research & Development hiring talented foreign workers.

This report gives an efficient approach to predict the H1-B visa status of the person before or after filing the petition. In the beginning, the report shows the relationships between different attributes through plotting. Later, we have described literature survey, feature selection, implementation of different models such as GaussianNB, k-Means and AdaBoost on the available data and Association Rule Mining to get the rules based on minimum support.

II. INITIAL SETUP

We have developed our whole project in the Spyder IDE using Python. There are many in-built libraries available in the Spyder which can be directly imported into the code and used. Python offers flexibility with minimal syntax and easy implementation. We have used sklearn library to split the data into training and testing datasets. We also used the same library to implement the GaussianNB model, k-Means model and AdaBoost model. We used numpy library to handle large and multi-dimensional arrays, pandas framework to handle the data structures and matplotlib.pyplot library to plot the outputs on the graphs. The installation of these libraries is done in the Anaconda Navigator and then they are imported into the program to execute. We have imported the following libraries:

1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4. from sklearn import train_test_split
5. from sklearn import scikitlearn
6. from sklearn.naive_bayes import GaussianNB
7. from sklearn.cluster import KMeans
8. from sklearn.ensemble import AdaBoostClassifier

III. DATA VISUALIZATION

The Kaggle Dataset has 27 features corresponding to the detail of the applicant for H1B Visa. However not all the features are relevant to the prediction (for example: Case ID does not contribute to the VISA prediction, however fields like Employer name, Occupation, etc are relevant to get the labour condition of the applicant).

At the beginning, we have five different output prediction labels in the main dataset. We have merged these labels and classified them into two main categories: 'CERTIFIED' and 'DENIED'. The labels are merged as shown below:

- 'CERTIFIED_WITHDRAWN' -> 'CERTIFIED'
- 'INVALIDATED' -> 'DENIED'
- 'REJECTED' -> 'DENIED'

Later, after applying the encoding, the main labels are classified as: 'CERTIFIED' and 'DENIED'. The chart for the labels is shown below:

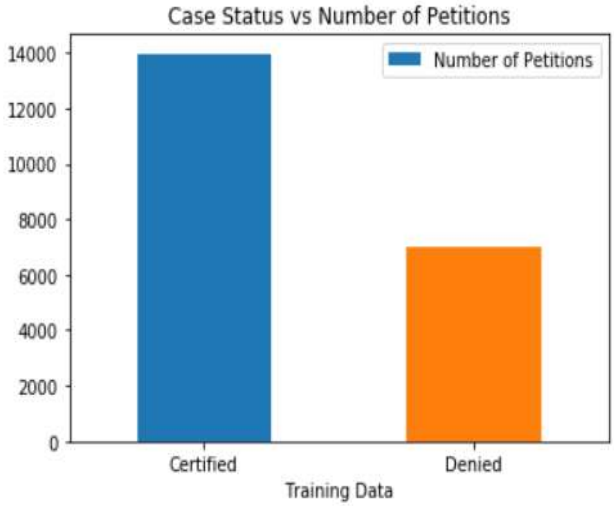

Figure 1. Case Status vs Number of Petitions

So, we visualized the data by plotting graphs for the attributes to the output status. The following are the graphs used for visualizing the data:

1. We checked the number of applications for each year from 2012-2017 from the dataset and found that the number of petitions filed increased gradually which can be observed in the following figure.

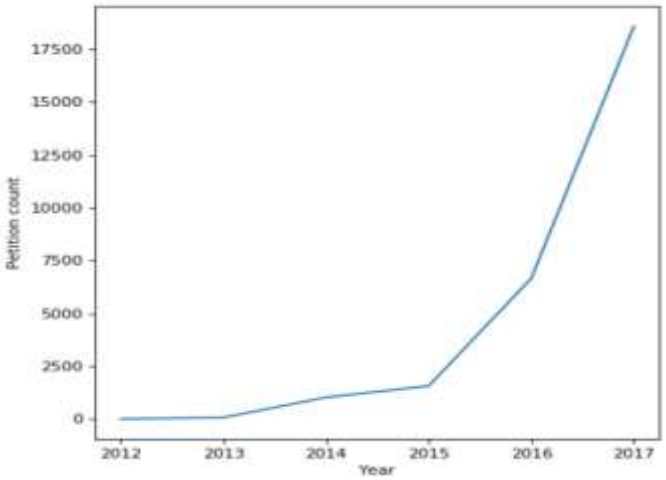

Figure 2. Year vs Number of petitions filed

2. We checked for the relation between the FULL_TIME_POSITION with the CASE_STATUS and plotted the graph as shown below.

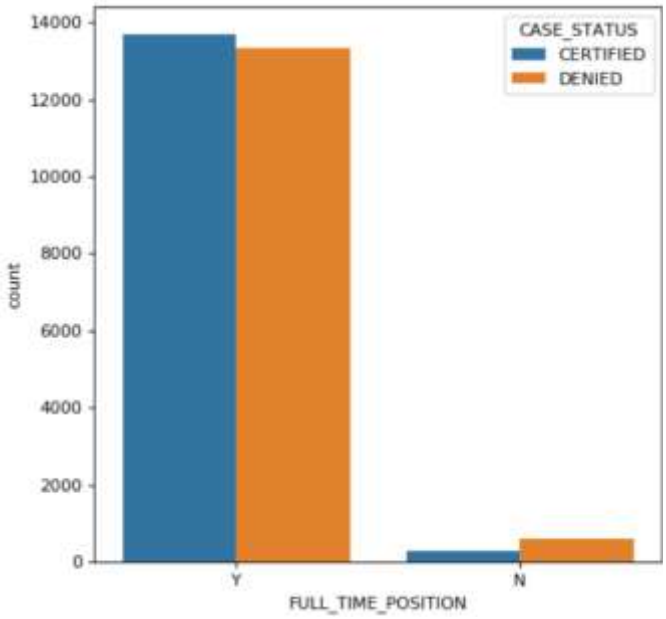

Figure 3. Full time positions vs number of petitions

3. From the above figure it cannot be concluded if the attribute is important for the prediction or not. So, we took the FULL_TIME_POSITION as "NO" and plotted a graph which proved that the attribute is important for the prediction of the CASE_STATUS as shown in the following figure.

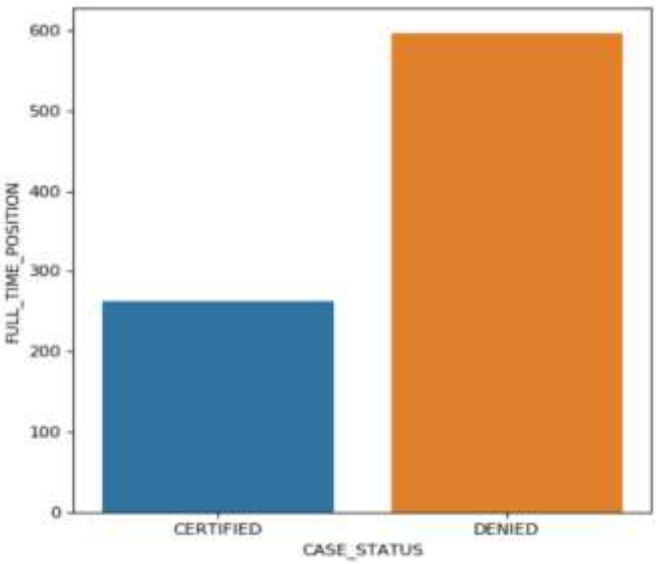

Figure 4. Full-time positions="No" vs Case tatus

- From the above figure, it can be concluded that the person not having full-time job have more chances of getting denied for the H1-b Visa.

4. Then, we checked for the attribute PREVAILING_WAGE with the CASE_STATUS. By observing the graph, we found that the attribute significantly impacts the predictions and is to be considered for the prediction. The graph is shown below.

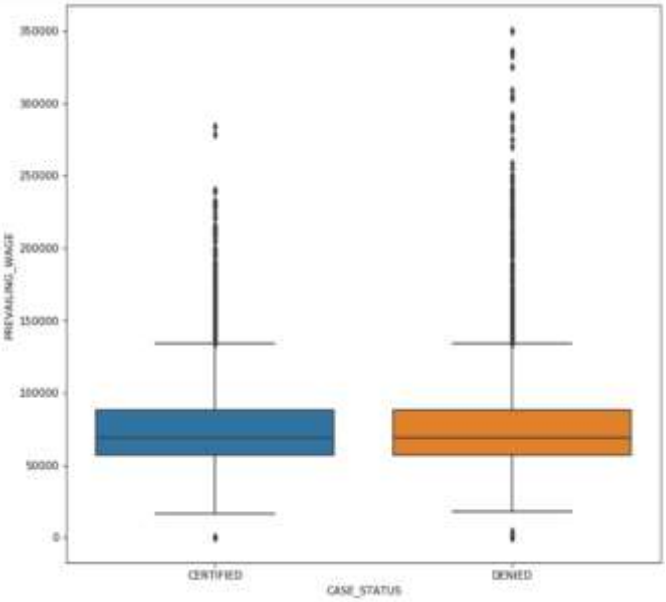

Figure 5. Prevailing Wage vs Case Status

5. We checked for the SOC_NAME in which the occupation field is given with the CASE_STATUS and found that this attribute also impacts the output prediction of the status.

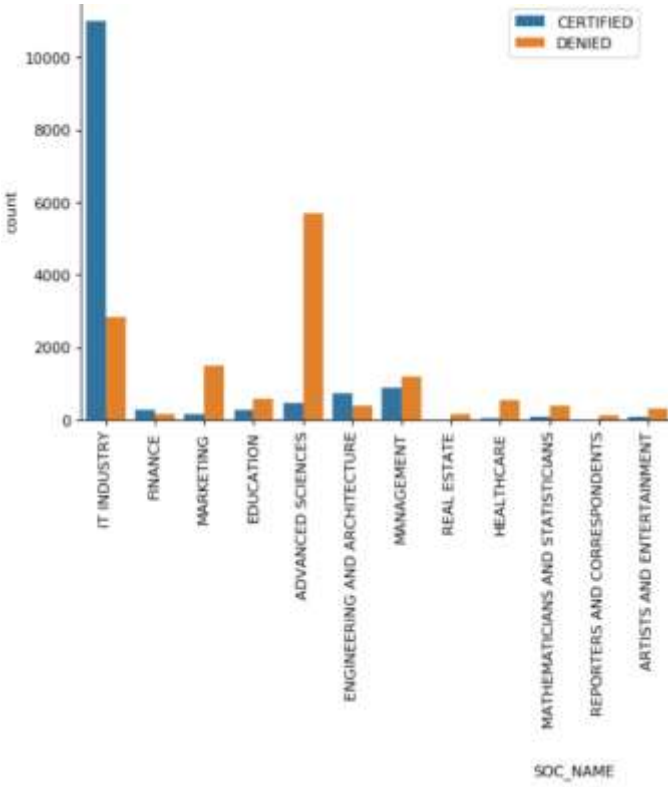

Figure 6. Soc_Name vs Case_Status

6. Lastly, we compared the WORKSITE_STATE with the CASE_STATUS and found that the attribute is impacts the outcome of the status as shown below.

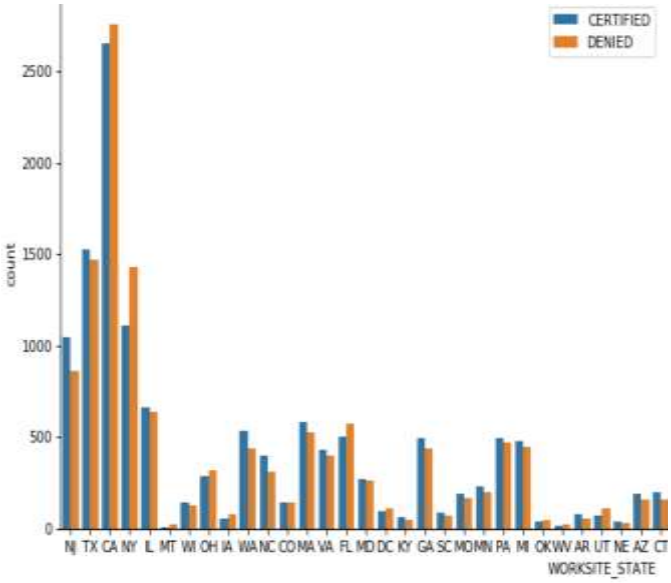

Figure 7. Work state vs Case Status

- Hence, by visualizing the above graphs, we can get the attributes which are important for the prediction of the output status. The table below shows the relevant attributes along with its description.

Table 1. Relevant fields used for Prediction

Fields used for Prediction	Description
FULL_TIME_POSITION	Y = Full Time Position; N = Part Time Position
PREVAILING_WAGE	Prevailing Wage for the job being requested
CASE_SUBMITTED_YEAR	Year when Visa Application was submitted
EMPLOYER_NAME	Name of employer submitting application.
SOC_NAME	Occupational name associated with the SOC_CODE
WORKSITE_STATE	State information of the foreign worker's intended area of employment
CASE_STATUS	Status associated with the last significant event or decision. Valid values include "Certified," "Certified_Withdrawn," Denied," and "Withdrawn".

IV. DATA PREPROCESSING AND CONVERSION

A. Data Pre-processing and Cleaning:

1. Filling the missing values in the dataset:
   In the dataset, there are many places where the data is empty. The counts of the fields with NaN values before pre-processing were as follows:

Table 2. Filling the missing values

Fields with NaN	**count**
EMPLOYER_NAME	4
EMPLOYER_STATE	1
FULL_TIME_POSITION	2
PW_UNIT_OF_PAY	31
PW_SOURCE	32
PW_SOURCE_YEAR	30
H-1B_DEPENDENT	882
WILLFUL_VIOLATOR	882

For all the above fields, we replaced the NaN value with the 'mode' of the data. The mode corresponds to the most occurring value in the feature. Apart from the above, we have replaced the NaN values in the PREVAILING_WAGE field with the mean of the salaries and the SOC_NAME (Occupation) field with "Others".

2. Then, we checked the balancing of the dataset so that the number of samples of "CERTIFIED" and "DENIED" class are approximately close to each other. The data is balanced in both: training and testing datasets.

B. Data Conversion and Feature Extraction:

1. Data Conversion:
   We will use the pre-processed data obtained after data pre-processing for our model implementation. The data is converted into the numerical values from the categorical values to feed into the models to predict the results. We used the "One Hot Encoding" approach to encode the data into the numerical values to be used for the implementation of the models. The dataset is encoded using this approach so that we have the numerical values for all the columns of the dataset.
2. Feature Creation:
   We did the data normalization on the following important attributes of the dataset which are already converted into the numerical values using the encoding technique:
   1. "CASE_SUBMITTED_YEAR_RANGE" attribute:
   The above attribute contains the year when the visa petition was filed. We have made it binary with the values being 'in or before 2017' and 'after 2012'. After the conversion, the complexity to predict the status decreases.
   2. "PREVAILING_WAGE" attribute:
   The wages of the foreign workers also vary on a large scale. We have clustered them into five groups based on the range of the wages. This also reduces the complexity of the algorithm to predict the status.
   3. "SOC_NAME" attribute:
   We have the occupation of the person by whom or for whom the H1-B visa petition is being filed. We have clustered different values of the column into 8 cluster based on the type of the occupation. The following are the values and the corresponding clusters for the particular value:

Table 3. Grouping "SOC_NAME" into Clusters

SOC_NAME	FIELD
COMPUTER OCCUPATION GRAPHIC DESIGNERS ANALYSTS	IT INDUSTRY
DESIGNERS |ENTERTAINMENT FASHION DESIGNERS MULTIMEDIA ARTISTS AND ANIMATORS	ARTISTS AND ENTERTAINMENT
ACCOUNTANTS|BUSINESS OPERATIONS SPECIALIST|CHIEF EXECUTIVES|CURATORS|EVENT PLANNERS|FIRST LINE SUPERVISORS|HUMAN RESOURCES|IT MANAGERS|MANAGEMENT|MANAGERS|PUBLIC RELATIONS	MANAGEMENT
MARKETING SALES AND RELATED WORKERS	MARKETING
ACTUARIES | FINANCE	FINANCE
COACHES AND SCOUTS COUNSELLERS | EDUCATION FITNESS TRAINERS | LIBRARIANS LOGISTICIANS | SURVEYERS	EDUCATION
ENGINEERS | COMMUNICATIONS LAB TECHNICIANS| MECHANICS CONSTRUCTION |ARCHITECTURE	ENGINEER AND ARCHITECTS
AGRICULTURE | ANIMAL HUSBANDRY | FOOD PREPARATION WORKERS	FOOD AND AGRICULTURE
SCIENTIST | DOCTORS | INTERNIST	ADVANCED SCIENCES

V. MODEL IMPLEMENTATION

There are two different datasets used for training and testing. The dataset "File-1" is used for training purpose and "File-2" is used for testing purpose in all the models. We ran the datasets on three models by changing their respective hyper-parameters.

A. k-Means Model

k-Means clustering is an unsupervised learning technique in which it aims to partition n observations into k clusters where each observation belongs to a cluster with the nearest mean. For an unknown observation, the algorithm puts it in one of the clusters formed.

1. *Hyper-Parameters:*

- Number of clusters (k): This defines the number of clusters used to form as well as number of centroids to generate.
- n_init: This defines about how many times the k-Means algorithm will be run with different centroid seeds.
- max_iter: This defines the maximum number of iterations for the algorithm for a single run.

2. *Testing Model:* k-Means Model

- We are varying the size of k as [2,3,4] along with the number of times to run the algorithm and maximum number of iterations with the algorithm as "auto".

- The following table shows the accuracy on the test dataset while using different values of the hyper-parameters:

Table 4. Accuracy table for k-Means

Number of clusters (k)	n_init	Max Iterations	Accuracy on Test Data
2	10	300	**0.637**
3	10	300	0.3142
4	10	300	0.1723
5	10	300	0.1251
2	30	500	0.637
3	30	500	0.3142
4	30	500	0.1268
5	30	500	0.0849

3. *Observations:*

- By comparing the values in the above table, we can conclude that the accuracy of the model decreases with the increase in the value of k if the other two hyper-parameters remain constant.
- The above statement can also be proved by observing the graph of value of k VS accuracy as shown below:

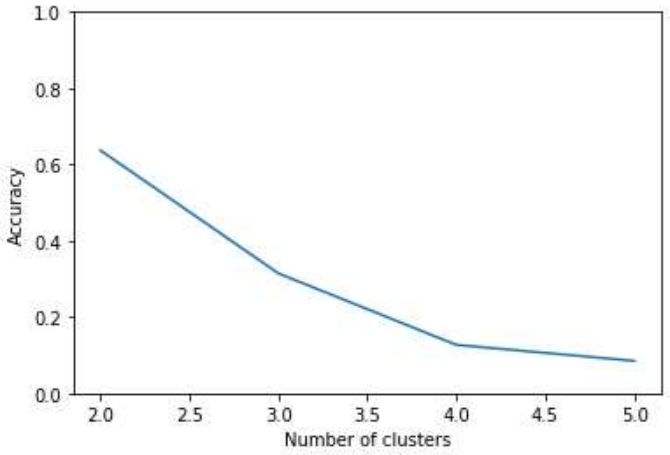

Figure 8. Number of clusters vs Accuracy

- When the other two parameters are increased or decreased, there is not much difference in the accuracy.
- We have also plotted ROC curve for the model which appears as below:

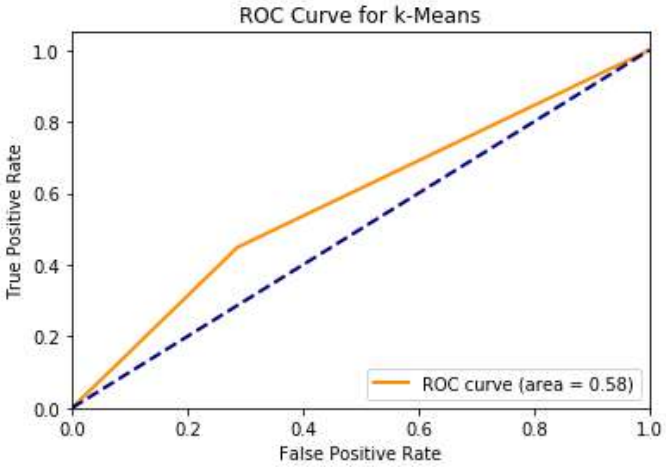

Figure 9. k-Means ROC curve

- In the above figure, as the area under the curve is less, we can conclude that the ratio of True Positive to False positive is 65:35 which is not so good and the accuracy is comparatively poor.
- The best of k-Means model gave the following observations:
    1. Confusion Matrix: [[10624 3342]
       [4264 2719]]
    2. Class Precision Score: [0.714 0.449]
    3. Class Recall Rate: [0.761 0.340]

B. Gaussian Naïve Bayes Model

The Naïve Bayes Model is supervised machine learning technique which predicts the labels considering all the feature values continuous and independent of one another. The GaussianNB Model considers that the distribution of the feature values follows Gaussian distribution.

1. *Hyper-Parameters:*

- priors – It defines the prior probabilities of the classes used for the prediction of the labels.
- var_smoothing – This is the portion of the largest variance of all features added to the variances for calculation stability.

2. *Testing Model:* GaussianNB Model

- For this model, there are very small number of hyper-parameters for the cross-validation of the data and also, the impact of these hyper-parameters is not much on the prediction of the labels.
- However, we tried changing the values of the hyper-parameters by assigning the probabilities using prior and changing the variances as shown in the table below:

Table 5. Accuracy table for GaussianNB

Priors	Var_smoothing	Accuracy
None	1e-09	**0.8015**
None	1e+05	0.7858
None	1e+09	0.7927

3. *Observations:*

- As shown in the above table, we can conclude that there is not much impact of the different values of the hyper-parameters for predicting the status in our dataset.
- The average accuracy almost remains same even though the values of var_smoothing are changed.
- But, the average accuracy of the GaussianNB Model is more as compared to the k-Means Model's best accuracy with two clusters.
- The ROC curve for this model is as below:

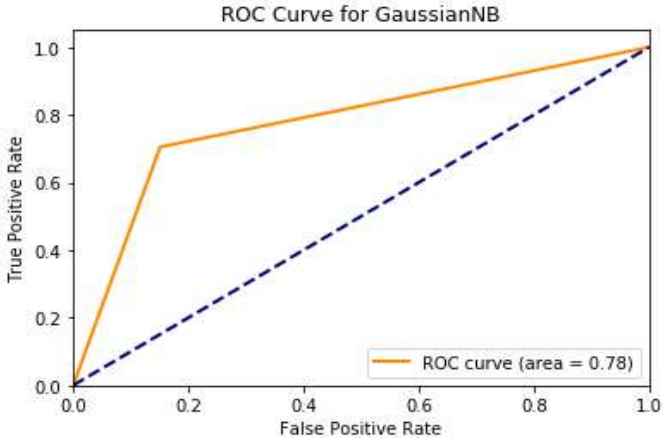

Figure 10. GaussianNB ROC curve

- The area under the curve for GaussianNB model is more than the area under the curve for k-Means model which indeed proves that the current model is previous than the k-Means model.
- The best of GaussianNB model gave the following observations:
  1. Confusion Matrix: [[11932 2034]
     [2125 4858]]
  2. Class Precision Score: [0.8488 0.7048]
  3. Class Recall Rate: [0.8544 0.6957]

C. *AdaBoost Model*

AdaBoost Model is supervised machine learning technique used to boost the performance of the other machine learning algorithm. It works the best on the weak learners of the algorithm.

1. *Training Parameters:*

- n_estimators: It is the maximum number of estimators at which boosting is terminated.
- base_estimator: It is the estimator from which the boosted ensemble is built. It can be any machine learning technique on which the boosting can be applied and the default is DecisionTreeClassifire (max_depth = 1).
- Learning_rate: It shrinks the contribution of each classifier by the learning_rate.

2. *Testing Model:* AdaBoost Model

We are varying the number of estimators along with the base_estimator and the learning rate. The accuracies with different values of hyper-parameters are as shown in the below figure:

Table 6. Accuracy table for AdaBoost

n_estimators	base_estimator	learning_rate	Accuracy
25	default	1	0.8282
50	default	1	0.8285
100	default	1	0.8286
200	default	1	0.8286
25	svm	0.5	0.8456
50	svm	0.5	0.8465
100	svm	0.5	**0.8506**
200	svm	0.5	0.8503

3. *Observations:*

- As shown in the above table, it can be seen that the highest accuracy is obtained when the base_estimator is kept as SVM model and the learning rate is 0.5.
- It is observed that the changes in the number of estimators of the model does not have much impact on the accuracy on the test data. This statement can also be proved by observing the below figure:

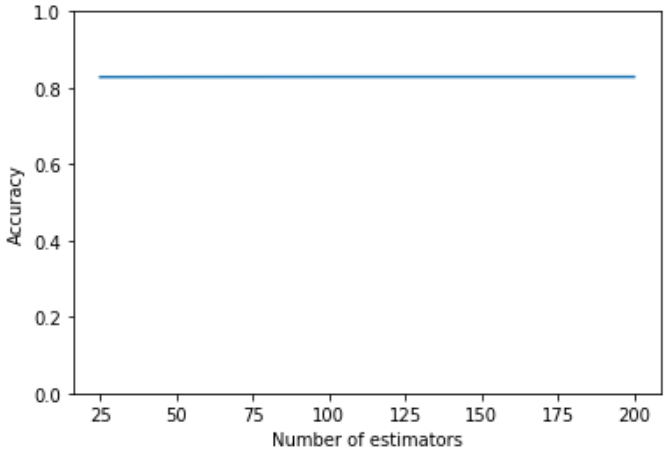

Figure 11. Numer_of_estimators vs Accuracy

- The ROC curve for the model appears as below when the model is tested with the test-dataset:

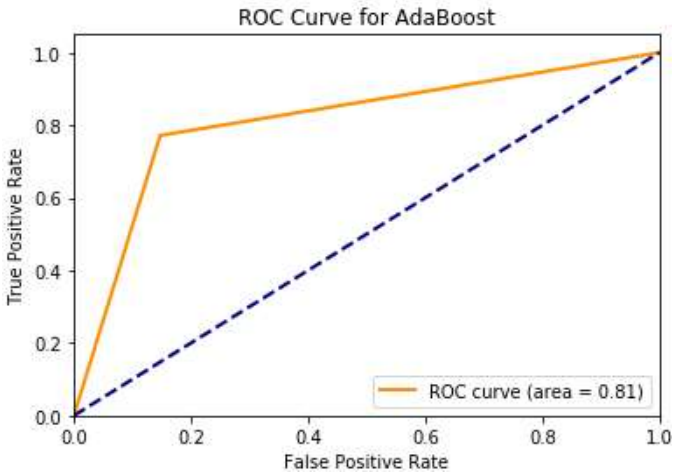

Figure 12. AdaBoost ROC curve

- As observed from the above figure, we can say that the area under the curve (0.81) is the highest for this model as compared to the area for k-Means (0.58) and for GaussianNB (0.78).
- The best of Adaboost model gave the following observations:
  1. Confusion Matrix: [[12563 1403]
     [2186 4797]]
  2. Class Precision Score: [0.8517 0.7738]
  3. Class Recall Rate: [0.8995 0.6869]

VI. ASSOCIATION RULE MINING

We also applied the Apriori algorithm to perform Association rule mining to determine the patterns among the attributes to predict the labels in the dataset. As the dataset is too large for the application of the algorithm, we took a smaller dataset to apply the algorithm. We were able to find the rules for the dataset which can be later applied to the whole dataset.

Firstly, we got the frequent itemset from the attributes: "SOC_NAME", "PREVAILING_WAGE", "H1B_DEPENDENT", "WORKSITE_STATE" and "CASE_STATUS" by taking the Support value as 0.01.

Then, we determined the association rules from the frequent itemsets using Support and Lift measures. There are some rules obtained from the frequent itemset as there are a lot of rules originally obtained for the dataset.

The below figures show some of the frequent itemset and the association rules for the attributes. (To see all the itemsets and Association rules, kindly run the python code.)

0	19.16
1	35734.0
2	38970.0
3	39853.0
4	40706.0
5	43110.0
6	45302.0
7	45540.0
8	46821.0
9	47244.0
10	49800.0
11	51730.0
12	52915.0
13	54600.0
14	58053.0
15	59197.0
16	59405.0
17	63170.0
18	63877.0
19	66331.0
20	76502.0
21	77875.0
22	90376.0
23	96845.0
24	116605.0
25	ACCOUNTANTS
26	ANALYSTS
27	AR
28	CA
29	CERTIFIED
30	COMPUTER OCCUPATION
31	DC

Figure 13. Frequent itemsets of the attributes

	rule
3525	{AR} => {ACCOUNTANTS, CERTIFIED, N, 45302.0}
491	{40706.0} => {OR, CERTIFIED}
3773	{N, COMPUTER OCCUPATION} => {DENIED, NY, 54600.0}
3772	{54600.0, N} => {DENIED, NY, COMPUTER OCCUPATION}
3769	{DENIED, 54600.0} => {NY, N, COMPUTER OCCUPATION}
3768	{DENIED, NY} => {54600.0, N, COMPUTER OCCUPATION}
3767	{DENIED, N} => {NY, 54600.0, COMPUTER OCCUPATION}
490	{OR} => {40706.0, CERTIFIED}
3068	{116605.0, N} => {NY, ANALYSTS}
481	{ACCOUNTANTS, 40706.0} => {OR}
3765	{54600.0} => {DENIED, NY, N, COMPUTER OCCUPATION}
494	{OR, CERTIFIED} => {40706.0}
495	{40706.0, CERTIFIED} => {OR}
496	{OR} => {40706.0, N}
497	{40706.0} => {OR, N}
3069	{NY, ANALYSTS} => {116605.0, N}
482	{ACCOUNTANTS, OR} => {40706.0}
480	{OR} => {ACCOUNTANTS, 40706.0}
3030	{CA, CERTIFIED, Y} => {96845.0}
3781	{DENIED, NY, COMPUTER OCCUPATION} => {54600.0, N}
3048	{116605.0} => {NY, ANALYSTS, CERTIFIED}
3786	{NY, 54600.0, COMPUTER OCCUPATION} => {DENIED, N}
3785	{54600.0, N, COMPUTER OCCUPATION} => {DENIED, NY}
3054	{116605.0, CERTIFIED} => {NY, ANALYSTS}
934	{59405.0} => {GA, CERTIFIED}
3784	{NY, N, COMPUTER OCCUPATION} => {DENIED, 54600.0}
3780	{DENIED, NY, 54600.0} => {N, COMPUTER OCCUPATION}
479	{40706.0} => {ACCOUNTANTS, OR}
3779	{DENIED, N, COMPUTER OCCUPATION} => {NY, 54600.0}
3061	{NY, ANALYSTS, CERTIFIED} => {116605.0}

Figure 14. Association rules based on the frequent itemsets

VII. RESULTS

A. Time taken by every model to predict the output label where base_estimator for AdaBoost is binaryTreeClassifier (default).

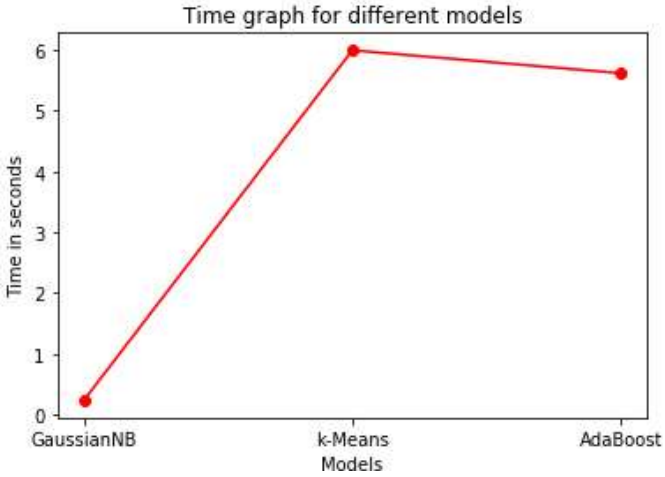

Figure 15. Time comparison where base_estimator = "Default"

B. Time taken by all the models where base_estimator for AdaBoost is SVM.

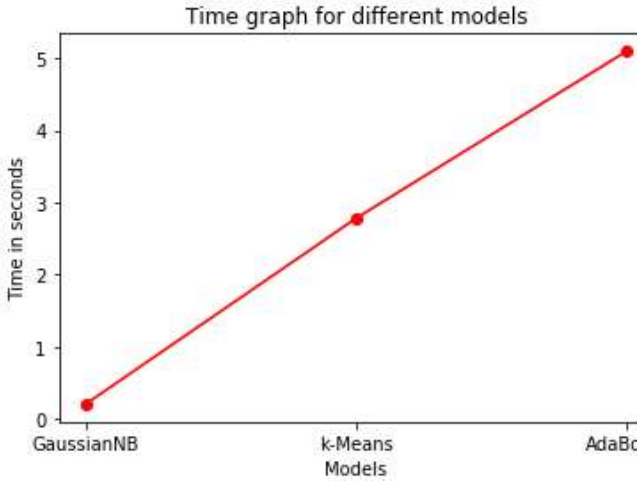

Figure 16. Time graph where base_estimator = "SVM"

C. Comparing the accuracy of different models.

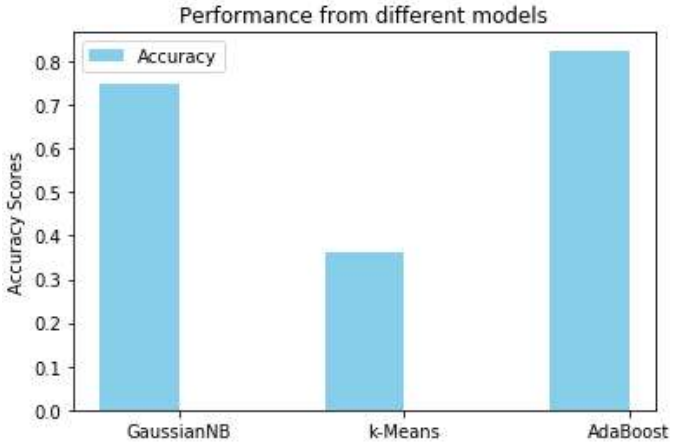

Figure 17. Accuracy Comparison among models

CONCLUSION:

Comparing the results obtained using the different models, we can conclude that:

- The accuracy for the k-Means model is the least among all the models while the AdaBoost has the highest accuracy for our dataset.
- From the figures of time comparisons, we can say that the highest time is taken by the k-Means model to predict the status but if SVM model is used as the base_estimator in the AdaBoost algorithm then it will take the highest time among all the models.
- So, it is best to use the **AdaBoost model** as it gives the highest accuracy and less time for prediction with the base_estimator as default value.
- For future enhancements, we plan to apply more models for prediction and an efficient algorithm to get Association rules of the whole dataset.

REFERENCES

[1] H1-B Visa Prediction Dataset, Retrieved from https://www.kaggle.com/trivedicharmi/h1b-disclosure-dataset#H1B%20Disclosure%20Data%20Project%20Report%20-%20Kaggle.pdf

[2] Implementation of Scikit Learn, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[3] Implementation of Machine Learning Models using Scikit Learn Library.

[4] Apriori Algorithm Implementation, from *Hands-On Unsupervised Learning with Python by Stefan Jansen.*

[5] Implementation of graphs using Seaborn and Matplotlib Libraries of Python.