

# H1B Visa Petition Status Prediction

\*

Yogita Suryavanshi  
*Department of Data Analytics*  
*San Jose State University*  
San Jose, United States of America  
yogitasuryavanshi@sjsu.edu

Preeti Khatri  
*Department of Data Analytics*  
*San Jose State University*  
San Jose, United States of America  
preeti.khatri@sjsu.edu

Jyoti Patel  
*Department of Data Analytics*  
*San Jose State University*  
San Jose, United States of America  
jyoti.patel@sjsu.edu

Hrushikesh Pokala  
*Department of Data Analytics*  
*San Jose State University*  
San Jose, United States of America  
hrushikesh.pokala@sjsu.edu

Saroj Saran  
*Department of Data Analytics*  
*San Jose State University*  
San Jose, United States of America  
saroj.saran@sjsu.edu

Priyanka Balla  
*Department of Data Analytics*  
*San Jose State University*  
San Jose, United States of America  
priyanka.balla@sjsu.edu

**Abstract**—The H-1B visa is a greatly desired work visa that is granted by the Immigration department of the United States which allows skilled foreign workers to enter the United States on a valid work permit. Every Year thousands of immigrant's petition for H-1B visas, these applications are usually made by the MNC companies on behalf of their employees. With tens of thousands of applications registered with the USCIS (US citizenship and immigration services), a certain annual limit is imposed on the number of H-1B visa issuance by conducting a lottery to go onto the process randomly and to select the petitions only after a considerable screening process. This selection process is highly competitive with only thousands of employees securing the H-1B visa among millions of applications. Requirements for an H-1B visa include a) A degree in Bachelor's or Master's Course (or any equivalent course from their foreign country) b) or work experience in the relevant field. c) or higher education and work experience in relevant field Based on the current trends of the visa requirements and profile score, we would like to analyze and predict the H-1B visa petition acceptance status by analyzing parameters like applicant salary, job profile, work experience, job type with the use of certain machine learning algorithms like Logistic Regression, Random Forest Classifier, Decision Tree, and Gaussian Naive Bayes Classifier. This project would aim in benefiting various groups of people to capture their success rates and prepare for the possible outcome. We have used Talend ETL for data cleaning, Python scikit- Learn for data modeling implementation. The designed Machine learning classifier models in this report provide a suitable prediction accuracy to serve the H1B applicants and their employer to measure their probability of getting the visa petition approved.

**Index Terms**—Machine Learning, Decision Forest, Random Forest, Logistic Regression, Naive Bayes, H1B Visa

## I. INTRODUCTION

H1B visa is the employment visa issued by the United States of America under the immigration act as a permit for highly

skilled foreign nationals to work legally on the soil of the United States of America, ca, this visa H1B issued through the United States Department of immigration following a selection process to identify the qualified and skilled personnel. This visa allows the foreign personnel to work on a temporary basis, this visa is valid for a term of three years which can be extended to up to six years. H1B being the most in-demand visa status makes its approval process complex thus involving a low rate of approval, with the increasing economy in the United States there comes an increasing demand for the visa applications which in turn affects the overall approval rates.

This visa approval strategy follows a strict selection process involving several selection factors like the employer petitioning for the person, person wage, Person's education, person's location, person's job description, etc. These factors are considered by the United States Department of Labor to decide upon the approval or rejection of the visa petition, upon approval of the visa petition his application are moved to the lottery pool to pick the desired set of applications for approval, in case the application is denied by the Department of Labour the applicants will have to reapply for the visa before the end of their current visa expiration.

As part of this research project we are aiming at designing a prediction model to enable the H1B visa petition status prediction, where we will focus on designing the model using Logistic regression, Random Forest, Decision Tree, and Gaussian Naive Bayes algorithms, we will discuss in detail about the various steps carried out to complete the modeling of this predictive system, Along with the predictive model our project will also focuses on designing web based application to take a real time data from users and provide them their visa status prediction and this data will further be used to learn.

## II. RELATED WORK

Since the beginning, the crucial aspect that every year nearly one-quarter of a million people are looking forward to is getting picked up in the VISA lottery. In this area there are a sizable number of types that people fall into. The majority of the VISA petitions fall in the H-1B category. The majority of the applicants are engineers. Since we were almost into the same category in the near future, we were looking forward to working on the analysis and trends of predictions.

While the goal is to secure an H1-B visa and have their work in the USA, the probability of getting picked up in a lottery was playing a crucial role in this area. The idea of creating a model or any algorithm which can predict the possibility by the historical data has been the primary motive of this project since the beginning. The model simply predicts if one can get a H1-B visa based on the primary details of them. As the lottery system has its own rules defined and parameters like area of the job, the study background, his salary, his work profile, and few others influence the probability of the selection. Various machine learning algorithms could be used to achieve the same but the ideal algorithm for this is under debate. Logistic regression has been one of the possible algorithms that has been effective in such scenarios and the analysis proved to be effective. The programming language python has been one of the effective languages for such machine learning model development.

The research shows the clustering concept is being effective in this scenario, especially quantization in signal processing. [1] Debabrata Swain explains that logistic regression has been the primary building block for the prediction model and has been effective using the methods of K-means clustering. The process helped in identifying the systematic data focal points and the difference between the clustering was significant.

[2] The other research section based on the decision tree usage, by Hartigan, demonstrated techniques that created a more accurate model. The systematic K-means clustering and decision trees provided more accurate results in the smaller datasets instead of a dataset of average size. The usage of pre-processing steps and creating new features in the ML model has been one other technique in this paper. The overall research revealed the fact that using unsupervised learning has been effective in this prediction model building. Clustering being the key component of an effective step that to be considered. The decision tree is also proven to be effective in the scenarios

[3] R. Serban mentions in his paper to discover the characteristics of persons living in the southern United States the findings of the analysis. The findings of the analysis demonstrate that the decision tree model yielded superior variable selection than the logistic regression model for predicting whether a person is likely to dwell in the south, at least from the data set we used.

[4] In the paper mentions that by tweaking the parameters of the random forest model, high-accuracy evaluation results can be obtained, and when compared to the single decision

tree algorithm, the proposed method's effectiveness evaluation findings are not only more accurate, but also more stable.

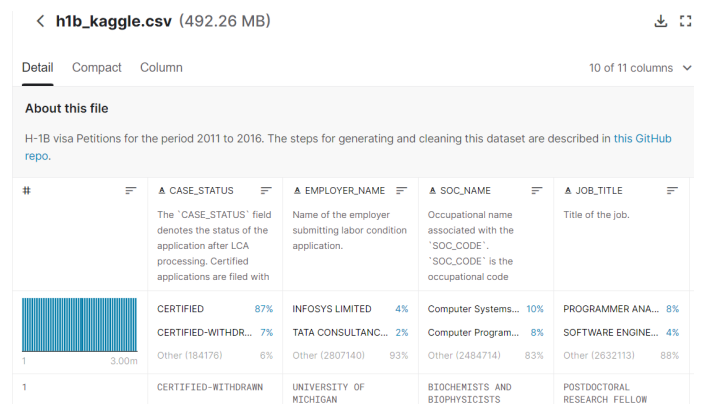
[5] The researcher proposes on the chatbot system, by suggesting a classification approach called intent classification to determine intent rather than user input; the researcher also wants to know the level of accuracy, precision, and recall on the evaluation findings of both methods. The Naive Bayes approach was used in this study to classify the data, and it was compared to the Logistic Regression method to establish the class intention. The Logistic Regression model has a greater level of accuracy, precision, and recall than the Naive Bayes model, according to the evaluation results.

[6] In this paper to recommend insurance products, random forest is used and compared to ID3, C4.5, Naive-Bayes, and Nearest-neighbor. The prediction error of random forest is 2.02 percent lower than ID3, 1.09 percent lower than C4.5, 1.67 percent lower than Naive-Bayes, and 5.97 percent lower than Nearest-neighbor, according to the findings of the experiment. As a result, recommending insurance products using random forests is very realistic.

[7] W.B. Zulfikar in her work proposed a classification model that used both a decision tree and a naive bayes classifier to speed up the process. During the evaluation phase, the accuracy and performance of both algorithms will be compared. As a consequence, we discovered that the decision tree has an accuracy value of exactly 66,65 percent and the naïve bayes classifier has an accuracy value of 79,95 percent. The other testing used a 100-data-testing method and a 400-data-training method. We discovered that the decision tree has exactly 78.5% accuracy while the naïve bayes classifier has 81.5% accuracy.

## III. DATASET

We have found a suitable Dataset for the project from Kaggle.com by author S. Naribole (2016) [8]. This dataset contains data of H-1B Visa Petitions from the year 2011-2016. There are about 3 million records of petitions in this dataset, refer to figure for the reference of the dataset from the Kaggle.com, we will discuss in detail about each column of the data in the dataset, refer Fig. 1.



#	CASE_STATUS	EMPLOYER_NAME	SOC_NAME	JOB_TITLE
1	CERTIFIED	INFOSYS LIMITED	Computer Systems...	PROGRAMMER ANA...
2	CERTIFIED-WITHDR...	TATA CONSULTANC...	Computer Program...	SOFTWARE ENGINE...
3	Other (184176)	Other (2807140)	Other (2484774)	Other (2632113)

Fig. 1. Dataset from Kaggle

dataframe1.head()						
	Key	CASE_STATUS	EMPLOYER_NAME	SOC_NAME	JOB_TITLE	FULL_TIME_POSITION
0	1	CERTIFIED-WITHDRAWN	UNIVERSITYOFMICHIGAN	BIOCHEMISTS AND BIOPHYSICISTS	POSTDOCTORAL RESEARCH FELLOW	N
1	2	CERTIFIED-WITHDRAWN	GOODMANNETWORKSINC	CHIEF EXECUTIVES	CHIEF OPERATING OFFICER	Y
2	3	CERTIFIED-WITHDRAWN	PORTSAMERICAGROUPINC	CHIEF EXECUTIVES	CHIEF PROCESS OFFICER	Y
3	4	CERTIFIED-WITHDRAWN	GATESCORPORATIONAWHOLLYOWNEDSUBSIDIARYOFTOMKINSPLC	CHIEF EXECUTIVES	REGIONAL PRESIDENT, AMERICAS	Y
4	5	WITHDRAWN	PEABODYINVESTMENTSCORP	CHIEF EXECUTIVES	PRESIDENT MONGOLIA AND INDIA	Y

Fig. 2. Sample of the H1B dataset

PREVAILING_WAGE	YEAR	WORKSITE	CITY	COUNTRY	lon	lat
36067.0	2016.0	ANN ARBOR, MICHIGAN	ANN ARBOR	MICHIGAN	-83.743038	42.280826
242674.0	2016.0	PLANO, TEXAS	PLANO	TEXAS	-96.698886	33.019843
193066.0	2016.0	JERSEY CITY, NEW JERSEY	JERSEY CITY	NEW JERSEY	-74.077642	40.728158
220314.0	2016.0	DENVER, COLORADO	DENVER	COLORADO	-104.990251	39.739236
157518.4	2016.0	ST. LOUIS, MISSOURI	ST. LOUIS	MISSOURI	-90.199404	38.627003

Fig. 3. Sample of the H1B dataset

The fields used in our dataset are detailed below, refer fig Fig. 2 and Fig. 3 for raw dataset.

- 1) **Employer\_Name** – This column consists of all the employers' names having unique names listed for the H1 visa selection process in which the company submits the application.
- 2) **SOC\_Name** — This column holds all the Occupational names connected to an occupational code.
- 3) **Job\_Title** — This column consists of all the unique job titles present in the data set which must be eligible for the H1 visa selection process.
- 4) **Full\_Time\_Position** – This column consists of position requirements whether it would be full time or not for the job title applied. The “Y” variable indicates the position would be full time and “N” variable indicates position would be a part time job.
- 5) **Prevailing\_Wage** - This column holds the average wages value for that position which would be paid to sane employed workers for the asked occupation.
- 6) **Worksite** - This column consists of the information about the names of the Cities and States of the foreign workers considered for that area of employment.
- 7) **Case\_Status** – This column shows the status of visa filed application after LCA Processing is done. It holds six distinct values such as “CERTIFIED”, “CERTIFIED-WITHDRAWN”, “DENIED”, “WITHDRAWN”, “REJECTED”, “INVALIDATED”, “PENDING QUALITY AND COMPLIANCE REVIEW”.
- 8) **Year** – This column shows the year in which the filing of the H-1B visa petition was filed.

- 9) **City** - This column consists of the information about the names of the cities of the foreign workers considered for that area of employment.
- 10) **Country** - This column consists of the information about the names of the countries of the foreign workers considered for that area of employment.
- 11) **Lon** – This column shows the worksite's longitude.
- 12) **Lat** - This column shows the worksite's latitude.

#### IV. WORKFLOW

This section provides us the detailed workflow of this project, which described the different phases carried out to achieve the project goals, refer Fig. 4

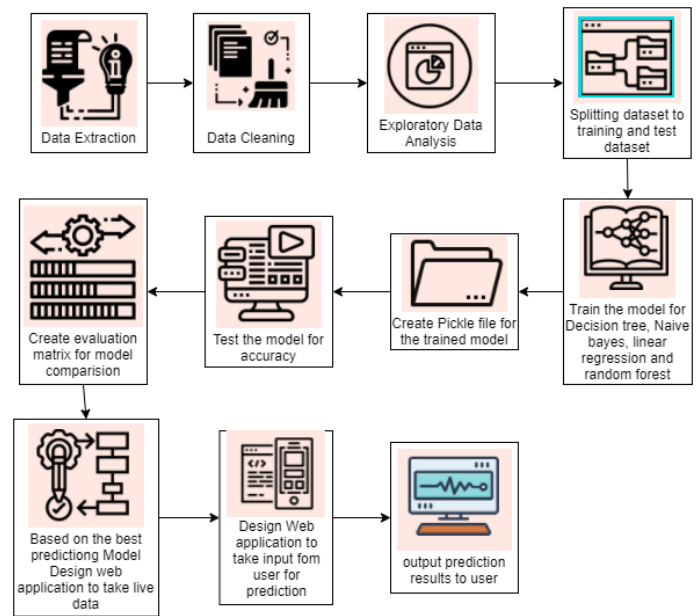


Fig. 4. Dataset Cleaning Through Talend

Workflow involves:

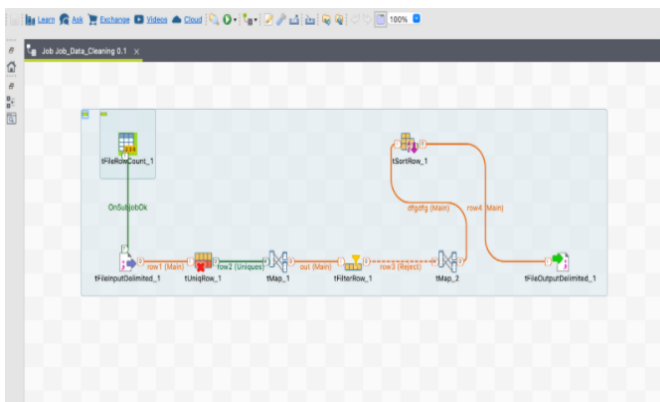
- 1) Stage 1: Data extraction and collection, for this project we have collected the data from Kaggle.com, data collected from Kaggle consist of 3 million records.
- 2) Stage 2: Data Cleaning, upon collection of data we will clean all the noisy data in the raw dataset, upon preliminary analysis it was found that the data collected had null values, missing data, duplicate data which will be handled in this phase.
- 3) Stage 3: Exploratory Analysis, before starting with the model designing and testing, we will perform a round of exploratory analysis to understand the data and its pattern.
- 4) Stage 4: Data splitting, raw data will then be split into training set and test set, where the training dataset will be used to train the machine learning model and test dataset will be used to test the designed model.

- 5) Stage 5: Model Training, the training set of data will be used to train the Machine learning model, here we will use Naive Bayes, Logistic Regression, Random Forest, and Decision tree machine learning models.
- 6) Stage 6: Pickle File Creation, upon model training we will create pickle file for these training datasets, this will help us with an easy testing on the pickle file where we do not have to run the training stage every time the model is tested with new data.
- 7) Stage 7: Model Testing: we will run the testing on the trained model to find the predictions for the test data.
- 8) Stage 8: Evaluation, we will design the evaluation matrices to compare the accuracy of all four models to find the model with better accuracy and performance.
- 9) Stage 9: Upon evaluating the prediction model and finding the best working model for the H1B prediction system, we aim at designing a web application to take petition request from users to feed this live data to our prediction model for their prediction.
- 10) Stage 10: Web application takes the input fed from user for prediction and then using the trained model which will be stored in pickle file, a prediction will be evaluation.
- 11) Stage 11: Upon prediction evaluation results will be displayed to the user, similarly the petition request fed to the web application will be preserved for future modeling and analysis.

- 1) Duplicate Records: The dataset taken from Kaggle have some duplicate values of petitions filed so we removed those duplicates using Talend jobs since we do not want redundant data in our model.
- 2) Missing Data: There were lot of cities which have no Longitude and latitude present in the data so we took the cities latitude and longitude details from google and used the Talend jobs to populate the missing values of latitude and longitude for those records as we need those 2 columns for further analysis.
- 3) Segregating Columns: The column 'Worksite' has values of City and Country together in a single column, which may not be accurately helpful in our analysis hence we have segregated that column into 2 separate columns named City and Country as both will be separately required for further analysis.
- 4) Null values: Certain petition records have null or N/A values in their feature fields like year and city column. , since such data add no value to our analysis and modeling these data are required to be removed for a smooth processing in model development.
- 5) Sorting: The date requires us to sort the column and using Talend we were able to sort the dataset based on the year on which the H1B visa petition has been filed for a particular employee.
- 6) Junk Values: Various columns of dataset such as "SOC\_NAME, JOB\_TITLE, FULL\_TIME\_POSITION, PREVAILING\_WAGE have a huge amount of junk data which some junk characters in it, which is handled by using Talend.

Fig. 6. Dataset Cleaning Through Talend

Fig. 6 and Fig. 7.



### H1B Visa petition Status Prediction

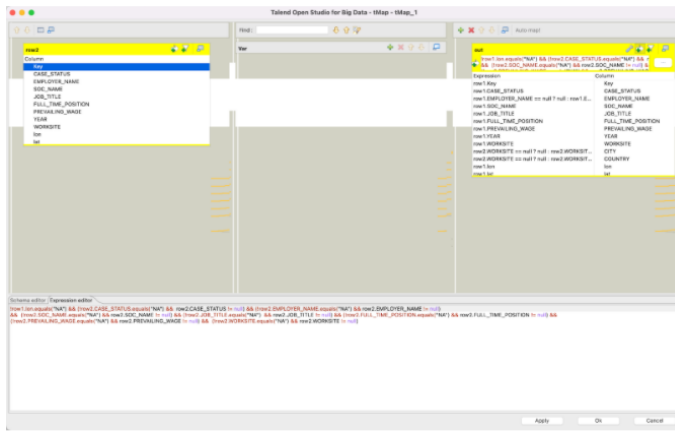


Fig. 7. Dataset Cleaning Through Talend

## VI. EXPLORATORY ANALYSIS

Exploratory Data Analysis is primarily carried out to understand and evaluate the data in hand to understand the data trends and its pattern before drawing conclusions. For example, these exploratory analysis benefits the project by locating the data irregularities and finding the outliers or unusual events and any noteworthy correlations between the variables. we have performed a preliminary analysis using python pandas to understand the dataset and its pattern, we have H-1B visa Petitions for the period 2011 to 2016 available in the dataset

### 1) Analysis 1

Analysis was to identify the count of filed petitions against the petition status between the year 2011 to 2016. Where the count of Certified petitions is the maximum with a count of 1650982 and the petition status with minimum count of 1 petition each for Invalidated and rejected visa status, refer Fig. 8.

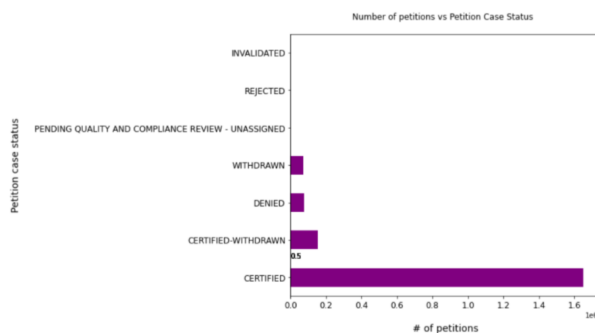


Fig. 8. Number of petitions vs Petition case status

### 2) Analysis 2

The purpose of this study is to identify the top 10 companies that have applied for more count of H1B visas for their employees. The following Figure gives

us the list of top 10 companies and the number of petitions they have received over the period 2011 to 2016. The same is depicted in the horizontal bar chart, refer Fig. 9 for detailed information.

Top 10 Employers filing for petition

INFOSYSLIMITED	26131
IBMINDIAPRIVATELIMITED	20480
WIPROLIMITED	18795
TATACONSULTANCYSERVICESLIMITED	17630
DELOITTECONSULTINGLLP	16113
ACCENTURELLP	15275
HCLAMERICAINC	11884
TECHMAHINDRAAMERICASINC	10589
IBMCORPORATION	9511
COGNIZANTTECHNOLOGYSOLUTIONSCORPORATION	8940

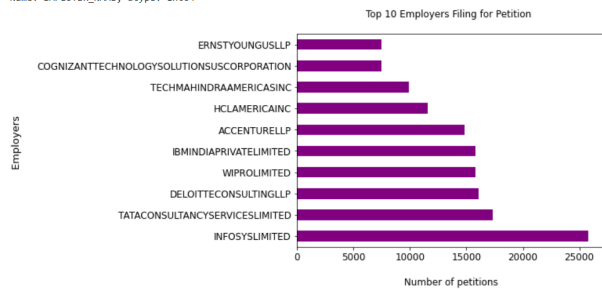


Fig. 9. Chart for top 10 Employers filing petition

### 3) Analysis 3

This analysis was carried out to identify the number of petitions filed by the employers throughout the course of a calendar year and the details are shown in the visual graph, refer Fig. 10. The year with the maximum petitions was identified to be 2016, while the year with the fewest number of petitions were observed for the year 2011.

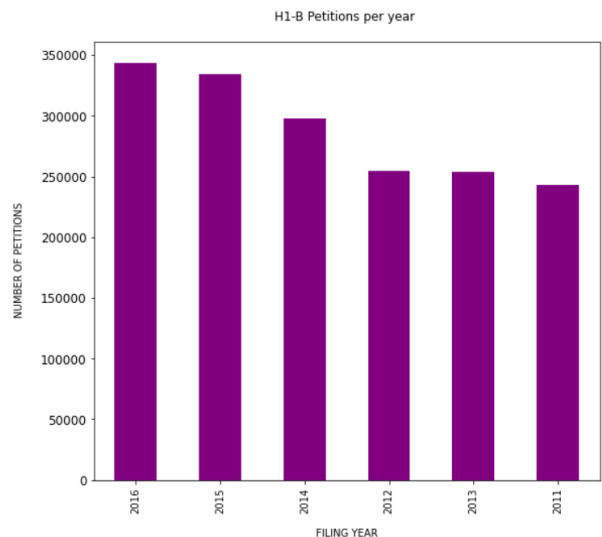


Fig. 10. H1B petitions per year

According to year-by-year petition acceptance rates, 2016 had the highest rate of petitions compared to 2011. Whereas for the year-by-year rejection rates, 2016 had less denial or rejection compared to 2011. for



each year from 2011 till 2016 we have a year-by-year line graph to represent the acceptance and rejection rates, refer Fig. 11 for the acceptance graph and Fig. 12 for the rejection graph.

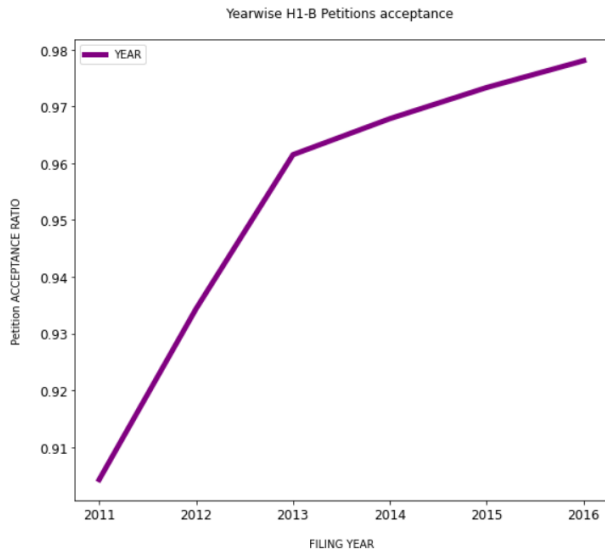


Fig. 11. H1B petitions acceptance yearwise

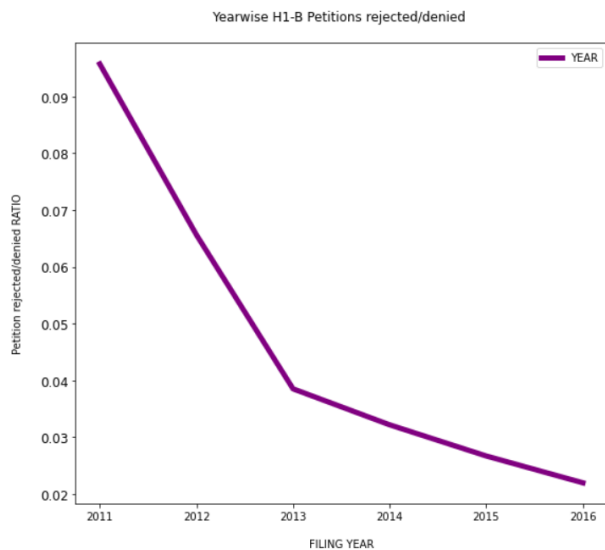


Fig. 12. H1B petitions rejected yearwise

#### 4) Analysis 4

The analysis was conducted to identify the top ten jobs for which the h1b visa petition was most frequently requested. In light of the fact that computer system analytics has received the most petitions. As per the chart, Computer Systems analysts stand first among the positions in demand for H1B. Below are the list of positions with count of H1B petitions for each, refer Fig. 13.

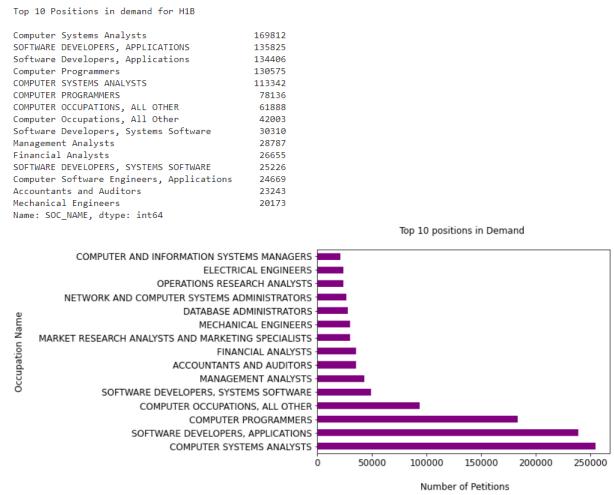


Fig. 13. Median salary of employees yearwise

#### 5) Analysis 5

This analysis was to identify the median salary of the employees, where from 2011 to 2016 the median wage of employees was plotted against the number of petitions filed each year. When observed in all the years the median wage of the employees is above 6K for all the petitioned employees, refer Fig. 14.

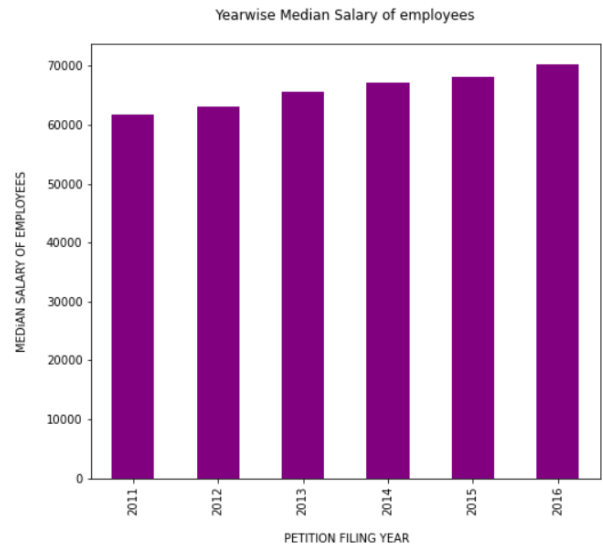


Fig. 14. Median salary of employees year-wise

## VII. FEATURE ENGINEERING

This preprocessed steps which converts the raw data set into features which are used further in ML Algorithms mostly to predict the models, such as predictive models.

### A. Experiment Design

The experimental design phase rs the various aspects of Analysis such as statistical Analysis of data which includes the process of conducting statistical operations on the data by

collecting, exploring, and analyzing data to identify trends and patterns. There are several types of statistical analysis, and we have used predictive analysis and exploratory data analysis for this project.

For predictive analysis we will be using Random Forest, Naive Bayes, Decision Tree and Logistic Regression to predict our data, Predictive analysis will be covered in detail in the coming modeling phase. whereas exploratory data analysis was covered in detail in the above section.

### B. Feature Selection

Feature selection is the process to identify the desired features for the modeling, which involves the reduction in the count of input variables in the process to develop a predictive model. The input variables used in our data set consists of 12 columns. The sample of our dataset is below, refer Fig. 15 and Fig. 16.

```
dataframe1.head()
```

	Key	CASE_STATUS	EMPLOYER_NAME	SOC_NAME	JOB_TITLE	FULL_TIME_POSITION	PR
0	1	CERTIFIED-WITHDRAWN	UNIVERSITYOFMICHIGAN	BIOCHEMISTS AND BIOPHYSICISTS	POSTDOCTORAL RESEARCH FELLOW		N
1	2	CERTIFIED-WITHDRAWN	GOODMANNETWORKSINC	CHIEF EXECUTIVES	CHIEF OPERATING OFFICER		Y
2	3	CERTIFIED-WITHDRAWN	PORTSAMERICAGROUPINC	CHIEF EXECUTIVES	CHIEF PROCESS OFFICER		Y
3	4	CERTIFIED-WITHDRAWN	GATESCORPORATIONAWHOLLYOWNEDSUBSIDIARYOFTOMKINSPLC	CHIEF EXECUTIVES	REGIONAL PRESIDENT AMERICAS		Y
4	5	WITHDRAWN	PEABODYINVESTMENTSCORP	CHIEF EXECUTIVES	PRESIDENT MONGOLIA AND INDIA		Y

Fig. 15. Sample of the H1B dataset

PREVAILING_WAGE	YEAR	WORKSITE	CITY	COUNTRY	lon	lat
36067.0	2016.0	ANN ARBOR, MICHIGAN	ANN ARBOR	MICHIGAN	-83.743038	42.280826
242674.0	2016.0	PLANO, TEXAS	PLANO	TEXAS	-96.698886	33.019843
193066.0	2016.0	JERSEY CITY, NEW JERSEY	JERSEY CITY	NEW JERSEY	-74.077642	40.728158
220314.0	2016.0	DENVER, COLORADO	DENVER	COLORADO	-104.990251	39.739236
157518.4	2016.0	ST. LOUIS, MISSOURI	ST. LOUIS	MISSOURI	-90.199404	38.627003

Fig. 16. Sample of the H1B dataset

The featured columns we are considering including are

- 1) Employer\_Name – This column consists of all the employers' names having unique names listed for the H1 visa selection process in which the company submits the application.
- 2) SOC\_Name — This column holds all the Occupational names connected to an occupational code.
- 3) Job\_Title — This column consists of all the unique job titles present in the data set which must be eligible for the H1 visa selection process.
- 4) Full\_Time\_Position – This column consists of position requirements whether it would be full time or not for the

job title applied. The “Y” variable indicates the position would be full time and “N” variable indicates position would be a part time job.

- 5) Prevailing\_Wage - This column holds the average wages value for that position which would be paid to sane employed workers for the asked occupation.
- 6) Worksite - This column consists of the information about the names of the Cities and States of the foreign workers considered for that area of employment.
- 7) Case\_Status – This column shows the status of visa filed application after LCA Processing is done. It holds six distinct values such as “CERTIFIED”, “CERTIFIED-WITHDRAWN”, “DENIED”, “WITHDRAWN”, “REJECTED”, “INVALIDATED”, “PENDING QUALITY AND COMPLIANCE REVIEW”.
- 8) Year – This column shows the year in which the filing of the H-1B visa petition was filed.
- 9) City - This column consists of the information about the names of the cities of the foreign workers considered for that area of employment.
- 10) Country - This column consists of the information about the names of the countries of the foreign workers considered for that area of employment.
- 11) Lon – This column shows the worksite's longitude.
- 12) Lat - This column shows the worksite's latitude.

### Preliminary Steps Requirement:

- 1) Selecting the data set from any public source available
- 2) Importing the data sets in the required tool.
- 3) Checking out for the missing values present in the data set.
- 4) Look for the Categorical Values in the data set.
- 5) Dividing the dataset into parts such as Training data set and Test data Set.
- 6) Feature the Scaling in which will apply limitations to the input variables so that later that can be compared with the common rules

### C. Feature Creation

In this process, we looked through all the features which are required to predict the model. All the present features were jumbled either by adding/multiply/subtracting which creates new derived features holding more prediction value.

We have used two of the feature creation.

- 1) **One Hot Encoding:** This method converts the categorical field into numerical value for wage, employer, soc and job. This was achieved by identifying the unique values of the columns. Also, one problem occurred in converting the multi class classification into binary classification for Case\_Status. So, feature encoding was performed to filter the Case\_Status to keep only 'CERTIFIED' or 'DECLINED'.

Refer Figure Fig. 17, which displays the data columns which are having categorical data and these data can

further be categorized into some basic format by using feature encoding.

```
# one-hot encoding for every possible and needed column
print("Dataframe with confirmed or denied cases :\n")
print("*****")
print(training_df.info())
print("*****")
print("Unique values count of each columns :\n")
print("Case Status ", training_df.CASE_STATUS.nunique())
print("Unique Employers ", training_df.EMPLOYER_NAME.nunique())
print("Unique SOCs ", training_df.SOC_NAME.nunique())
print("Unique Job Titles ", training_df.JOB_TITLE.nunique())
print("Unique Employment Type ", training_df.FULL_TIME_POSITION.nunique())
print("Unique Prevailing Wages ", training_df.PREVAILING_WAGE.nunique())
print("Unique Year ", training_df.YEAR.nunique())
print("Unique Worksite State ", training_df.WORKSITE.nunique())
print("Unique CITY State ", training_df.CITY.nunique())
print("Unique COUNTRY State ", training_df.COUNTRY.nunique())
```

Fig. 17. Feature Encoding of Categorical Value

Refer Figure Fig. 18, which displays the Case\_Status column which is having categorical data which is further converted into categorical data using feature encoding.

```
print(Temp_dataframe['CASE_STATUS'].unique())
Temp_dataframe = Temp_dataframe.loc[Temp_dataframe['CASE_STATUS'].isin(['CERTIFIED', 'DENIED'])] #filtering
print(Temp_dataframe['CASE_STATUS'].unique())
['CERTIFIED' 'DENIED' 'REJECTED']
['CERTIFIED' 'DENIED']
```

Fig. 18. Feature Encoding of Case Status Column

Refer Figure Fig. 19, which displays the unique values in each column of the dataset, this helps to identify the columns which are having huge distinct columns that needs to be categorized for easy modeling and testing.

```
*****
Unique values count of each columns :

Case Status      2
Unique Employers 65633
Unique SOCs      924
Unique Job Titles 48121
Unique Employment Type  2
Prevailing Wages 22598
Unique Year      6
Unique Worksite State 2467
Unique CITY State 2148
Unique COUNTRY State 52
```

Fig. 19. Unique column Names through encoding

- 2) **Categorization of features:** In this part, as looking in above one hot encoding code, the “Prevailing Wages” had huge unique vale of 22598, creating the problem to train the model so we categorized it into five columns with respect to the wages such as “VERY LOW holds wage under 50000”, “LOW holds wage greater than 50000 and less than equals to 70000”, “MEDIUM holds wage greater than 70000 and less than equals to 90000”, “HIGH holds wage greater than 90000 and less than equals to 150000”, “VERY HIGH” holds wage greater than equals to 150000.

Also, we categorized the “Grant Status” with respect to acceptance ratio into 6 different columns such as “AR with acceptance ratio equals to -1”, “VLA” with acceptance ratio greater than 0.0 and ;0.20”, “LA with acceptance ratio greater than 0.20 and ;0.40”, “MA with

acceptance ratio greater than 0.40 and less than 0.60”, “HA with acceptance ratio greater than 0.60 and less than 080”, “VHA” with acceptance ratio greater than equals to 0.80” making it easier to run the algorithms and predict the model, refer figure Fig. 20.

## Categorizing the Features

```
def Prevailing_wage_categorization(Prevailing_wage):
    if Prevailing_wage <=50000:
        return "VERY LOW"
    elif Prevailing_wage >50000 and Prevailing_wage <= 70000:
        return "LOW"
    elif Prevailing_wage >70000 and Prevailing_wage <= 90000:
        return "MEDIUM"
    elif Prevailing_wage >90000 and Prevailing_wage<=150000:
        return "HIGH"
    elif Prevailing_wage >=150000:
        return "VERY HIGH"

def Grant_status_Categorization(acceptance_ratio):
    if acceptance_ratio == -1:
        return "AR"
    elif acceptance_ratio >=0.0 and acceptance_ratio<0.20:
        return "VLA"
    elif acceptance_ratio>=0.20 and acceptance_ratio<0.40:
        return "LA"
    elif acceptance_ratio>=0.40 and acceptance_ratio<0.60:
        return "MA"
    elif acceptance_ratio>=0.60 and acceptance_ratio<0.80:
        return "HA"
    elif acceptance_ratio>=0.80:
        return "VHA"
def worksite_collector(worksite):
    return worksite.split(', ')[1]
```

Fig. 20. Categorization of Features Of Prevailing Wages

## VIII. MACHINE LEARNING MODELS

As part of modeling for this project we will focus on four different algorithms to understand the performance accuracy of all four models, and then upon comparing these model’s accuracy we identify the best performing model for this prediction.

We will also have a look into the “Pickled File” version for all these 4 algorithms which we did. Since the training dataset was in huge quantity, we have exported the pickled file so that training data can be stored, this will allow us to run the testing on the new test data where we do not have to train the model every time, we wish to run the test, thus saving us the rework and making the modeling memory efficient.

For training the model the data has been split into 80% of training data and 20% of test data. Thus, the training data contains 139827 records and the test data consists of 34957 records.

The Fig. 21 explains the detailed code for python code of splitting the data into train and test data by importing “train\_test\_split” from scikit-learn.

```
In [57]: 1 x_train, x_test, y_train, y_test = train_test_split(final_dataframe_training.iloc[:,1:], final_dataframe_training.iloc[:,
2 x_test[-1].stage
3 print("The number of records in the training data is:", len(x_train))
4 print("The number of records in the testing data is:", len(y_test))
5

The number of records in the training data is: 139827
The number of records in the testing data is: 34957
```

Fig. 21. Splitting Data

### A. Random Forest Classifier

Our first Machine learning algorithm model is Random Forest, Random Forest classifier is a supervised learning algorithm which works best both for classification and



Formula used to solve Classification Problem using Gini Index-

$$Gini = 1 - \sum_{i=1}^C (p_i)^2$$

$$MSE = \frac{1}{N} \sum_{i=1}^N (f_i - y_i)^2$$

```
In [69]: #pickled random forest prediction
rfst = randf.predict(x_test)
print(rfst)
probability = randf.predict_proba(x_test)

print("Test", Y_test[:10])
print("Prediction", rfst[:10])

print(metrics.confusion_matrix(Y_test, rfst))
print(metrics.classification_report(Y_test, rfst))

[[0 0 0 ... 1 0 1]
 Test 673293      0
 903730      0
 1280316      0
 1457299      1
 1785441      1
 401513       0
 1754095      1
 1745278      0
 1482800      1
 27798        0
 Name: CASE_STATUS, dtype: int64
 Prediction [0 0 0 1 1 0 1 0 1 0]
 [[11822  3247]
 [ 1758 18130]]

              precision    recall  f1-score   support

      0       0.87       0.78       0.83       15069
      1       0.85       0.91       0.88       19888

 accuracy         0.86
 macro avg         0.86
 weighted avg         0.86
```

Fig. 26. Testing with Pickle File for Random Forest

Absolute Error, Mean Squared Error and Root Mean Squared Error, we are evaluating above listed metrics to identify the different parameters which can help us identify the quality of the prediction models, these parameters will also help us compare the different models performance so we can identify the better working model.

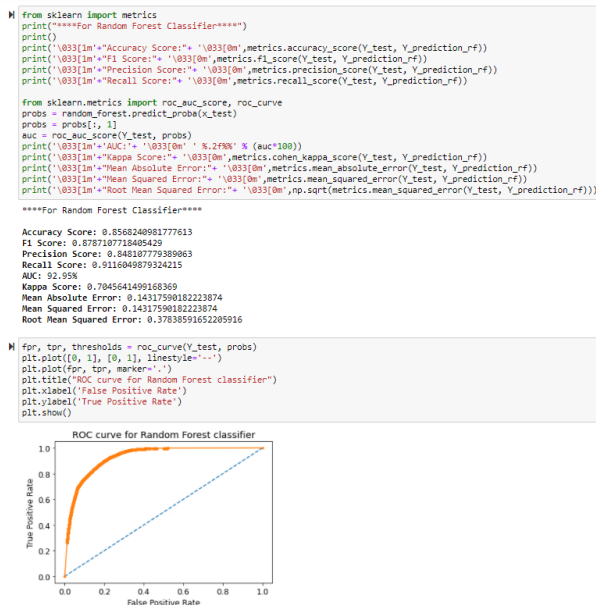


Fig. 27. ROC curve for Random Forest

## B. Gaussian Naive Bayes Classifier

Our Second Machine learning algorithm model is Gaussian Naive Bayes, Naive Bayes classifier is a supervised learning

algorithm which was elaborated more in advance to for the Gaussian Naive Bayes. It is one of the most used supervised learning algorithms. It is based on Bayes Theorem, helping in figuring the conditional probabilities of state of two events placed on the occurrence of probabilities of each single event, ciphering those probabilities as incredibly useful. This is useful for data sets which are continuous in nature and normal distribution,[11].

This algorithm consists of two parts of probabilities which directly calculate the result from your training data

- 1) Calculation of probability of a single class.
- 2) Calculation of conditional probability for every class for given each value x.

Formula used for Gaussian distribution:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Fig. 28. Naive Bayes Formula

Where, mu denotes mean of the distribution  
sigma denotes standard deviation of the distribution

Refer Fig. 29, When training the model for Gaussian Naive Bayes Classification the Gaussian NB Classifier is imported from the naive bayes library of scikit-learn. The model is trained using x\_train and Y\_train of data. The detailed explanation of training the Gaussian Naive Bayes classifier is represented in Fig. 29.

```
In [61]: # Gaussian Naive Bayes Classifier

In [62]: gaussian_classification = GaussianNB()
gaussian_classification.fit(x_train, Y_train)

Out[62]: GaussianNB()

In [71]: Y_prediction_gnbc = gaussian_classification.predict(x_test)
confusion = metrics.confusion_matrix(Y_test, Y_prediction_gnbc)
print("Test", Y_test[:10])
print("Prediction", Y_prediction_gnbc[:10])
print(metrics.confusion_matrix(Y_test, Y_prediction_gnbc))
print(metrics.classification_report(Y_test, Y_prediction_gnbc))

Test 673293      0
 903730      0
 1280316      0
 1457299      1
 1785441      1
 401513       0
 1754095      1
 1745278      0
 1482800      1
 27798        0
 Name: CASE_STATUS, dtype: int64
 Prediction [1 1 0 1 1 0 1 1 1 1]
 [[ 7388  7681]
 [2010 17878]]

              precision    recall  f1-score   support

      0       0.79       0.49       0.60       15869
      1       0.70       0.90       0.79       19888

 accuracy         0.74
 macro avg         0.69
 weighted avg         0.71
```

Fig. 29. Accuracy calculation through Naive Bayes Theorem

From Fig. 29, while testing the Gaussian Naive Bayes Classifier the testing data is predicted and the probability of the testing data is predicted. The confusion matrix and the classification report of the classifier is found out. The following are the results identified from the confusion matrix:

- 1) When the class is 0 and the predicted label is 0 the sample count is 7388
- 2) When the class is 0 and the predicted label is 0 the sample count is 7681

- So, there were 15,069 points in the first class where the model was able to identify 7388 out of those correctly in label 0 and 7681 were marked as label 1. In the second class out of 19,888 points the model was able to identify 17878 correctly in label 1 and 2010 were marked as label 0.

In the classification report, the precision for the first class is 79% and the precision for the second class is 70%. Thus, the precision is relatively low for both the classes when compared to other models. The recall for the first class is 49% and the recall for the second class is 90% . The recall of the second class is higher compared to the first class but since the recall for class one is below 50% , it is below average. The f1 score of the first class is 60% and the f1 score of the second class is 79%.

**The accuracy of Gaussian Naive Bayes algorithm is 72%.**

### Creating Pickle File for Naive Bayes:

```
pickle.dump(gaussian_classification,open('gaussian_classification.pickle','wb'))  
with open('gaussian_classification.pickle','rb') as g:  
    gf = pickle.load(g)  
  
print(open('gaussian_classification.pickle','rb').read()[140])  
  
b'\x00\x04\x95\xbf\x0b\x00\x00\x00\x00\x00\x00\x00\x8c\x13learn.naive_bayes\x94\x8c\nGausse'
```

Fig. 30. Export Pickled file of Gaussian Naive Bayes

The name of the pickled file is “gaussian\_classification.pickle”.

### Testing and Evaluation of Gaussian Naive Bayes Classifier:

The accuracy of the pickled prediction for Gaussian Naive Bayes Classification Algorithm represented in the Fig. 31 is 72% which is the same as that of the original prediction, this evaluation on pickle file was performed to just cross validate the prediction accuracy of the original prediction with the prediction on the pickle file.

Refer Fig. 32 for the evaluation metrics for the Gaussian Naive Bayes algorithm, where we are evaluating the Accuracy Score, F1 Score, Precision Score, Recall Score, AUC, Kappa Score, Mean Absolute Error, Mean Squared Error and Root Mean Squared Error, we are evaluating above listed metrics to identify the different parameters which can help us identify the quality of the prediction models, these parameters will also help us compare the different models performance so we can identify the better working model.

### C. Logistic Regression

Our Third Machine learning algorithm model is Logistic Regression, Logistic Regression classifier is a supervised

```
In [73]: #pickled gaussian classification prediction
gauf = gf.predict(x_test)
print(gauf)
probability = gf.predict_proba(x_test)

print("Test", Y_test[:10])
print("Prediction", gauf[:10])

print(metrics.confusion_matrix(Y_test,gauf))
print(metrics.classification_report(Y_test,gauf))

[1 1 0 ... 0 1 1]
Test 673293      0
903730      0
1280316      0
1457299      1
1785441      1
401513      0
1754095      1
1745278      0
1482800      1
27798      0
Name: CASE_STATUS, dtype: int64
Prediction [1 1 0 1 1 0 1 1 1 1]
[[ 7388  7681]
 [ 2010 17878]]

              precision      recall  f1-score   support

0               0.79         0.49         0.60       15069
1               0.70         0.90         0.79       19888

    accuracy                                0.72       34957
    macro avg              0.74         0.69         0.70       34957
    weighted avg           0.74         0.72         0.71       34957
```

Fig. 31. Testing of pickle File for Gaussian Naive Bayes

```

from sklearn import metrics
print('====For Gaussian Naive Bayes Classifier====')
print()
print('0331[In]=Accuracy Score: '+'0331[Out]=metrics.accuracy_score(Y_test, Y_prediction_gmbc)')
print('0331[In]=F1 Score: '+'0331[Out]=metrics.f1_score(Y_test, Y_prediction_gmbc)')
print('0331[In]=Precision Score: '+'0331[Out]=metrics.precision_score(Y_test, Y_prediction_gmbc)')
print('0331[In]=Recall Score: '+'0331[Out]=metrics.recall_score(Y_test, Y_prediction_gmbc)')

from sklearn.metrics import roc_auc_score, roc_curve
Y_gmb_score = gaussian_classification_predict_proba(x_test)
Y_gmb_score = Y_gmb_score[:, 1]
auc = roc_auc_score(Y_test, Y_gmb_score)
print('0331[In]=AUC: '+'0331[Out]='+'%.2f'% (auc*100))
print('0331[In]=Kappa Score: '+'0331[Out]=metrics.cohen_kappa_score(Y_test, Y_prediction_gmbc)')
print('0331[In]=Mean Absolute Error: '+'0331[Out]=metrics.mean_absolute_error(Y_test, Y_prediction_gmbc)')
print('0331[In]=Mean Squared Error: '+'0331[Out]=metrics.mean_squared_error(Y_test, Y_prediction_gmbc)')
print('0331[In]=Mean Negative Log Likelihood Error: '+'0331[Out]='+'%.2f'% (metrics.mean_squared_error(Y_test, Y_prediction_gmbc)))

====For Gaussian Naive Bayes Classifier====

Accuracy Score: 0.72273736933947421
F1 Score: 0.78676260259280393
Precision Score: 0.6894791635354959
Recall Score: 0.898934035711987
AUC: 83.108
Kappa Score: 0.40780571218566966
Mean Absolute Error: 0.2772263606052579
Mean Squared Error: 0.2772263606052579
Root Mean Squared Error: 0.526522845283342

M Y_gmb_score = gaussian_classification_predict_proba(x_test[:, 1], 1)
for tpr, thresholds = roc_curve(Y_test, Y_gmb_score)
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, linewidth=2)
plt.plot([0,1], [0,1], 'k-')
plt.cParams['font.size'] = 12
plt.title('ROC curve for Gaussian Naive Bayes Classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

```

Fig. 32. Evaluation Metrics for Gaussian Naive Bayes

learning algorithm which is mainly used for classification tasks. The outcome of prediction is categorical, like True/False, 0/1, Yes/No for the group of independent variables. It helps in giving the prediction of the probability of the event.

```
Out[53]: LogisticRegression()

In [55]: Y_prediction_lr = logistic_classification.predict(x_test)

probability = logistic_classification.predict_proba(x_test)

print("Test", Y_test[:10])
print("Prediction", Y_prediction_lr[:10])

print(metrics.confusion_matrix(Y_test, Y_prediction_lr))
print(metrics.classification_report(Y_test, Y_prediction_lr))

Test 673293      0
903736      0
1208316      0
1457299      1
1785441      1
401513      0
1754095      1
1745278      0
1462000      1
27798      0
Name: CASE_STATUS, dtype: int64
Prediction: [0 0 1 1 0 1 0 1 0]
[[11799 3270]
 [1133 18755]]

      precision    recall  f1-score   support

      0         0.91      0.78      0.84      15069
      1         0.85      0.94      0.89      19888

 accuracy          0.88      0.86      0.87      34957
 macro avg          0.88      0.87      0.87      34957
 weighted avg          0.88      0.87      0.87      34957
```

Fig. 33. Accuracy calculation through LR

- 1) When the class is 0 and the predicted label is 0 the sample count is 11799
- 2) When the class is 0 and the predicted label is 0 the sample count is 3270
- 3) When the class is 1 and the predicted label is 0 the sample count is 1133
- 4) When the class is 1 and the predicted label is 1 the sample count is 18755

So there were 15,069 points in the first class where the model was able to identify 11722 out of those correctly in label 0 and 3270 were marked as label 1. In the second class out of 19,888 points the model was able to identify 18755 correctly in label 1 and 1133 were marked as label 0.

The recall of the second class is higher compared to the first class but since the recall for both the classes are above 50% , it is good. The f1 score of the first class is 84% and the f1 score of the second class is 89%.

```
import pickle
pickle.dump(logistic_classification.open('logistic_classification.pickle', 'wb'))
with open('logistic_classification.pickle', 'rb') as f:
    lr = pickle.load(f)

print(open('logistic_classification.pickle', 'rb').read())[40])
```

Fig. 34. Pickled File Logistic Regression

The accuracy of the pickled prediction for the Logistic Regression algorithm represented in the Fig. 35 is 87% which is the same as that of the original prediction, this evaluation on pickle file was performed to just cross validate the prediction accuracy of the original prediction with the prediction on the pickle file.

```
In [63]: #pickled logistic regression prediction
lgr = lr.predict(x_test)
print(lgr)
probability = lr.predict_proba(x_test)

print("Test", Y_test[:10])
print("Prediction", lgr[:10])

print(metrics.confusion_matrix(Y_test, lgr))
print(metrics.classification_report(Y_test, lgr))

[0 0 0 ... 1 0 1]
Test 673293      0
903730      0
1280316      0
1457299      1
1785441      1
401513      0
1754095      1
1745278      0
1482800      1
27798      0
Name: CASE_STATUS, dtype: int64
Prediction [0 0 0 1 1 0 1 0 1 0]
[[11799  3270]
 [ 1133 18755]]

              precision      recall   f1-score      support

              0              0.91              0.78              0.84              15069
              1              0.85              0.94              0.89              19888

    accuracy              0.87              34957
    macro avg              0.88              0.86              0.87              34957
    weighted avg              0.88              0.87              0.87              34957
```

Fig. 35. Testing of Pickle File for Logistic Regression:



Mean Squared Error, we are evaluating above listed metrics to identify the different parameters which can help us identify the quality of the prediction models, these parameters will also help us compare the different models performance so we can identify the better working model.



Fig. 36. Evaluation Metrics for Logistic Regression

#### D. Decision Tree

Our fourth Machine learning algorithm model is Decision Tree, Decision Tree classifier is a supervised learning algorithm, which is used for both classification and regression problems solving. The aim to use this algorithm is to design the training model which predicts the all-target variables values through easy decision rules gathered from earlier training data. It does the predictions of labels which get from the root node of the tree. The values are compared of the root element and the gathered element. Based on the comparison, it follows the respective branch belonging to that value and jumps to the next point.[10]

Algorithm to calculate the Decision Tree Model is ID3 (Iterative Dichotomiser), where ID3 uses Entropy and Information Gain metrics to calculate the node.

Entropy[H(S)], measures the total amount of unpredictability in the data sets.

$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

Fig. 37. Entropy H(S)

S = The data set for which entropy is being calculated.  
p(x) = quantity of number of elements present in class x.  
X = Set of class in S

Information Gain [IG(A)], is a measure calculated between entropy difference present before and after in set S with respect to attribute A.

$$IG(S, A) = H(S) - \sum_{t \in T} p(t)H(t) = H(S) - H(S|A).$$

Fig. 38. Information Gain IG(A)

where, H(S) denotes entropy present in set S.  
T denotes subsets generated by splitting set S in attribute A  
p(T) denotes the amount of the number of factors in t to the amount of number of factors in set S.  
H(t) denotes entropy in subset S

When training the model for Decision Tree the Decision Tree Classifier is imported from the tree library of scikit-learn. The model is trained using x\_train and Y\_train of data. The detailed explanation of training the Decision Tree classifier is represented in the figure below, refer Fig. 39,

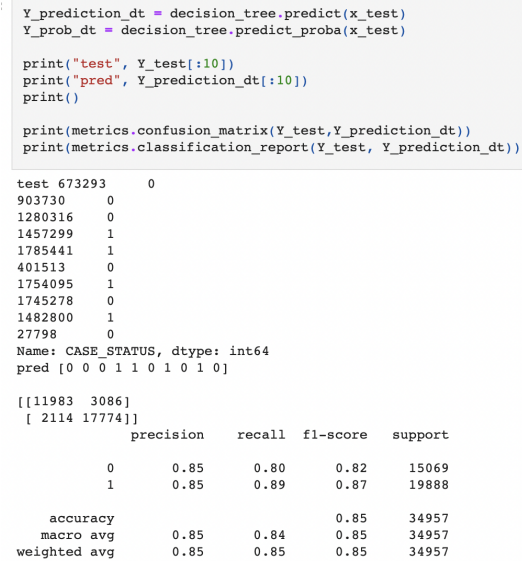


Fig. 39. Accuracy calculation through Decision Tree

From the Fig. 39, while testing the Decision Tree Classifier the testing data is predicted and the probability of the testing data is predicted. The confusion matrix and the classification report of the classifier is found out. The following are the results identified from the confusion matrix:

- 1) When the class is 0 and the predicted label is 0 the sample count is 11984.
- 2) When the class is 0 and the predicted label is 0 the sample count is 3085.



- So there were 15,069 points in the first class where the model was able to identify 11984 out of those correctly in label 0 and 3085 were marked as label 1. In the second class out of 19,888 points the model was able to identify 17784 correctly in label 1 and 2104 were marked as label 0.

**The accuracy of the Decision Tree algorithm is 85%.**

```
pickle.dump(decision_tree, open('decision_tree.pickle', 'wb'))  
with open('decision_tree.pickle', 'rb') as d:  
    dt = pickle.load(d)  
  
print(open('decision_tree.pickle', 'rb').read()[140])  
  
b'\x00\x04\x95\xca\x03\x00\x00\x00\x00\x00\x08\xc3\x5aklearn.tree._classes\x94\x8c\x16Dec'
```

The name of the pickled file is “ decision tree.pickle”.

The accuracy of the pickled prediction for Decision Tree algorithm represented in the figure Fig. 41 is 85% which is the same as that of the original prediction, this evaluation on pickle file was performed to just cross validate the prediction accuracy of the original prediction with the prediction on the pickle file.

This covers our machine learning modeling for H1B prediction, and upon observing the performance of all four models we can draw certain conclusions.

```
In [76]: #pickled decision tree prediction
decit = dt.predict(x_test)
print(decit)
probability = dt.predict_proba(x_test)

print("Test", Y_test[:10])
print("Prediction", decit[:10])

print(metrics.confusion_matrix(Y_test,decit))
print(metrics.classification_report(Y_test,decit))

[0 0 0 ... 1 0 1]
Test 673293      0
903730      0
1280316      0
1457299      1
1785441      1
401513      0
1754095      1
1745278      0
1482800      1
27798      0
Name: CASE_STATUS, dtype: int64
Prediction [0 0 0 1 1 0 1 0 1 0]
[[11983 3086]
 [ 2114 17774]]
               precision      recall   f1-score      support

0               0.85         0.80         0.82         15069
1               0.85         0.89         0.87         19888

accuracy                                0.85         34957
macro avg               0.85         0.84         0.85         34957
weighted avg            0.85         0.85         0.85         34957
```

```

print("""For Decision Tree""")
print()
print('||03||In="Accuracy Score:" * ||03||@m,metrics.accuracy_score(y_test, y_prediction_dt))
print('||03||In="F1 Score:" * ||03||@m,metrics.f1_score(y_test, y_prediction_dt))
print('||03||In="Precision Score:" * ||03||@m,metrics.precision_score(y_test, y_prediction_dt))
print('||03||In="Recall Score:" * ||03||@m,metrics.recall_score(y_test, y_prediction_dt))

from sklearn.metrics import roc_auc_score, roc_curve
y_prob_dt = decision_tree.predict_proba(x_test)
y_prob_dt = y_prob_dt[:, 1]
auc = roc_auc_score(y_test, y_prob_dt)
print('||03||In="AUC:" * ||03||@m, "%2f%%" % (auc*100))
print('||03||In="Kappa Score:" * ||03||@m,metrics.cohen_kappa_score(y_test, y_prediction_dt))
print('||03||In="Mean Absolute Error:" * ||03||@m,metrics.mean_absolute_error(y_test, y_prediction_dt))
print('||03||In="Mean Squared Error:" * ||03||@m,metrics.mean_squared_error(y_test, y_prediction_dt))
print('||03||In="Root Mean Squared Error:" * ||03||@m,np.sqrt(metrics.mean_squared_error(y_test, y_prediction_dt))

"""For Decision Tree"""

Accuracy Score: 0.8517212280430243
F1 Score: 0.8728084515609865
Precision Score: 0.8524857375731122
Recall Score: 0.8041860991054947
AUC: 90.12%
Kappa Score: 0.6953413185937751
Mean Absolute Error: 0.14826787195697572
Mean Squared Error: 0.14826787195697572
Root Mean Squared Error: 0.3850567384077816

# y_prob_dt = decision_tree.predict_proba(x_test)[: , 1]
# fpr, tpr, thresholds = roc_curve(y_test, y_prob_dt)
# plt.figure(figsize=(6,4))
# plt.plot(fpr, tpr, linewidth=2)
# plt.plot([0,1], [0,1], 'k-')
# plt.rcParams['font.size'] = 12
# plt.title('ROC curve for Decision Tree')
# plt.xlabel('false Positive Rate')
# plt.ylabel('true Positive Rate')
# plt.show()

ROC curve for Decision Tree

```

We have implemented these 4 algorithms in our model and after seeing the results of accuracy, the best algorithms with the highest accuracy rate are Logistic Regression.

The effectiveness of the analytical or machine learning model can be assessed using evaluation measures. Machine learning models or algorithms need to be tested in order to ensure the success of any project. A model or method is put to the test that used a collection of several evaluation metrics.

Below is the comparison table for the models implemented and their respective metrics.

Algorithms Metrics	Logistic Regression	Random Forest Classifier	Gaussian Naive Bayes Classifier	Decision Tree
Accuracy	87.39%	85.60%	72.20%	85.00%
F1 Score	0.89	0.87	0.78	0.87
Precision Score	0.85	0.84	0.69	0.85
Recall Score	0.94	0.91	0.89	0.89
AUC	94.50%	92.95%	83.16%	90.12%
Kappa Score	0.73	0.7	0.47	0.695
MAE	0.12	0.14	0.27	0.15
MSE	0.12	0.14	0.27	0.15
RMSE	0.35	0.37	0.53	0.38

Fig. 43. Evaluation Metrics for the Model Comparison

- Accuracy is the simplest intuitive performance measure because it is just a ratio of accurately predicted observations to the total predictions. We have the highest accuracy for Logistic regression algorithm that is 87.39%. And the least accuracy of 72.2% for Gaussian Naive Bayes classifier.
- The F1 score is one of the essential metrics to know the performance of the model. If you have a high F1 score, you're able to accurately recognize key challenges and avoid being disturbed 83.16.16 by error rates. When an F1 score is 1, the model is deemed ideal, but when it is 0 it is considered a complete failure. When comparing all the four models, the logistic regression has the F1 score as 0.9 which is a near value to 1.
- Precision and recall scores have a slight difference. Precision is the ratio of True positives and all positives, whereas the recall is the successfully anticipated positive observations to all actual class measurements: yes. We need to minimize the false positives and false negatives. These scores are perfect when the value is 1. So here, the logistic regression model has the values nearby 1 that is 0.85 and 0.94.
- The ROC curve can be summarized using the AUC, which measures a classifier's ability to differentiate between classes. The model's ability to differentiate between true and false classes improves with increasing AUC. Having the higher AUC, better the performance. From the table, we can notice that the AUC is high for Logistic Regression with 94.5%. Random forest classifier stands second with 92%.
- MAE is the relative difference in between projected and the actual values. MSE is calculated by averaging the squared difference between original and anticipated values.
- RMS Error is simply the square root of RMS error. Standard deviation is a measure of the variability of residuals. A good RMSE score should be between 0.2 - 0.5 which implies a model can determine accurate predictions. All the models present in the table have the RMSE fall under this range.

- When two raters are rated the same quantity, Cohen's Kappa is a statistical metric used to identify how often the raters are in agreement. When the score is above 0.75 it is excellent when between the range 0.4-0.75 it is considered good and below 0.4 is poor. When compared, all are considered to be good models.
- From the evaluation matrix from Fig. 43, considering all the evaluation metrics parameters, with a prediction accuracy of 87.39% which is maximum compared to all the models, we can conclude that the Logistic regression works better compared to all the machine learning models.

## X. WEB APPLICATION

The primary motivation was to find the best algorithm, but we have formulated the analysis to be useful for the evaluation of any petition using a web application. The web application is developed in python using the Flask framework. App has the basic routes which are concise to the landing page and prediction page. The landing page incorporates the user with a form which can be filled in with the required information needed. The applicant is provided with the sample data which they can edit. The form shows the sample data that is being filled, refer Fig. 44 for the GUI appearance of the web application to take the petition inputs from users.

We have used the Jupyter Notebook to implement the python code for machine learning implementation and then used the JavaScript, CSS, HTML and flask application to design and deploy this web application form which will help any user who is seeking a prediction help to understand where the user application stand and what is the expected status from the trained model, We have used the best performing model in this web application petition status prediction, to make the prediction performance faster the current web application is designed to run the test data prediction on the pickle file.

The form data will be posted to the prediction route which then uses the data to the model and get the prediction accuracy. The prediction is being evaluated by the various models and the best accuracy model is being used to evaluate the form details. The result would be finally evaluated as a PICK or NO PICK based on the model evaluation, refer Fig. 45 for the result interface representing the prediction result.

The advantage of having this kind of a web application developed over our prediction algorithms will help us gather new data from the users which can be stored for further learning thus facilitating u with a real time data collection and real time test data predictions. This will help us manage the velocity of data flowing in for the model predictions.

## XI. INNOVATION

This web application which enabled us to access the real time data from users which can be further stored for future learning, this was our innovative learning's, this helped us give a reason to our machine learning modeling project since this project not only proves to be a prediction model analysis but

also can be pushed in action to make a prediction for new applicants.

## XII. SIGNIFICANCE TO THE REAL WORLD

The project is based upon the idea to allow every employee seeking for a H1B visa status approval and their employer who is looking forward to on boarding their talented employees a well ahead prediction for their petitions based on the employees current status i.e., the features associated with the employee visa application, this will help them identify the success rates for the petition with given input features as with thousands of applications petitions every year the process is getting complex with a minimal success rate for petition approval, in such scenario this project can help such applicant to take every possible precautionary measures to make their petition request a success by pre-evaluating their petition status well in hand.

## XIII. LESSONS LEARNT

The concepts which we have learned in this course duration consists of many parts. It helped in achieving our goal in implementing the model with the best accuracy. The learning consists of-

- Complete exploratory analysis helped to understand our dataset more which is the critical step for the project.
- Supervised and Unsupervised learning which are different techniques which get applied for different data sets.
- Different Pandas and Scikit libraries of python which helped in solving the problems related to classification and finding accuracy.
- The concepts of SVM help in distinguishing between classification and regression task problems and predict the model based on that. It can be applied to linear problems or nonlinear problems.
- The various kinds of algorithms were covered such as Decision Tree, Logistic Regression, Random Forest Classification, and Gaussian Bayes which helped in predicting the accuracy of models by using trained and test dataset.
- Able to acquire knowledge on the Evaluation metrics, these are used to measure the quality and performance of the machine learning algorithms that we used on the dataset.
- Utilized most of the concepts of Machine Learning which are explained in class.
- We were able to implement the real time petition data capturing through the web application.
- We also learnt about the documentation tool LaTeX overleaf, which helped us to document the report in IEEE conference format.
- We were able to understand the importance of blog and Elevator pitch video, and how these would help us promote our project from a fewer audience to a wider range of audience.

The screenshot shows a web application titled "H1-B Visa Prediction". It contains several input fields: "Job Title" with the value "LEGAL MANAGER", "Full Time" with the value "N", "SOC Name" with the value "GENERAL AND OPERATIONS MANAGERS", "Employer Name" with the value "ANDERSONYEHPC", "Year" with the value "2016", "Wage" with the value "57033.6", "Worksite" with the value "SANTA MONICA, CALIFORNIA", "City" with the value "SANTA MONICA", and "Country" with the value "CALIFORNIA". At the bottom of the form is a green button labeled "Get Prediction".

Fig. 44. Web application to get User petition input

The screenshot shows a web application titled "PREDICTION RESULTS". Below the title, it says "The prediction for the given petition is a NO PICK".

Prediction Accuracy : 86.91

Fig. 45. Web application prediction Result

## XIV. TECHNICAL DIFFICULTIES

- 1) We had faced difficulty in loading our input dataset CSV file into the Jupyter Notebook since the file was extremely large and it was difficult to read the file. We finally resolved the issue by changing the settings of the Jupyter Notebook to accommodate and handle large files.
- 2) During Data Cleaning there were cities which didn't have longitude and latitude therefore we had to locate the latitude and longitude details of these cities and then integrated it with data to populate the correct values.
- 3) While adding files to the GitHub there were certain files which were larger in size and we couldn't add those files to the GitHub initially. We resolved this issue by adding the files to GitHub LFS which supports large file size through Git Bash.

## XV. PROJECT MILESTONES AND TEAM MEMBER ROLES

Our project kick-started 21st September and we were able to complete the project development and push the final code to the GitHub by 28th November.

Refer Fig. 46 which briefs the project milestones and the timelines.

Milestones	Start Date	End Date	Status
H1B Visa Petition Prediction			Completed
Data Sources	21/9/2021	28/9/2021	Completed
Project proposal report	29/9/2021	1/10/2021	Completed
Data Collection	2/10/2021	5/10/2021	Completed
Data Cleansing - using ETL tools	6/10/2021	9/10/2021	Completed
Data Processing	10/10/2021	14/10/2021	Completed
Data Analysis	15/10/2021	21/10/2021	Completed
Data Modelling	22/10/2021	26/10/2021	Completed
Project intermediate status Report	27/10/2021	29/10/2021	Completed
Development - using Python scripts	28/10/2021	6/11/2021	Completed
Development - using ML algorithms	7/11/2021	18/11/2021	Completed
Testing/Validation	19/11/2021	28/11/2021	Completed
Documentation	28/11/2021	3/12/2021	Completed
Presentation	22/11/2021	3/12/2021	Completed

Fig. 46. Project Milestones

Refer Fig. 47 which briefs the team members and their roles in the project tasks.

Stage	Members
Data Preparation	Yogita, Saroj
Data Cleaning	Yogita, Saroj
Data Analysis	Preeti, Priyanka, Hrushikesh
Data modelling & Algorithms	Jyoti, Yogita, Hrushikesh, Preeti, Priyanka, Saroj
Development, Validation	Jyoti, Yogita, Hrushikesh, Preeti, Priyanka, Saroj
Reporting	Jyoti, Yogita, Hrushikesh, Preeti, Priyanka, Saroj

Fig. 47. Team Members and Their Roles

## XVI. VERSION CONTROLLING

We have maintained all the file including the input dataset, Jupyter Notebook, web application files in the GitHub so we are able to trace all the changes and also at the same time maintaining a file versioning which can be reverted to older version in case of any issue, we also have placed a backup file in personal shared drive for the project use.

Refer Fig. 48 for the GitHub repository created to maintain the file versioning for the project.

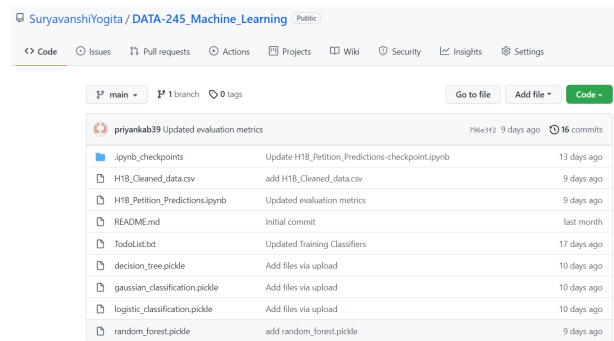


Fig. 48. GitHub Repository for the project

Refer Fig. 49 files where checking by users through GithUB desktop version, we can revert to old changes in case of any issue though the version ID revert.

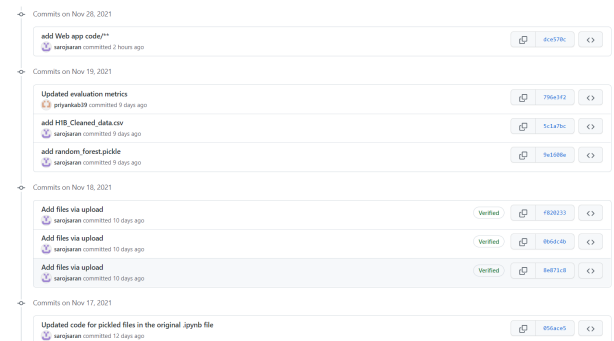


Fig. 49. GitHub File Versioning

## XVII. CONCLUSION

To conclude this project design, with the Exploratory Analysis the project was able to provide an insight into the data trends by helping the perspective employees and their employers with their petition filing process and their trends. We were able to achieve the desired predictive model with the accuracy as expected and we were able to identify the suitable model for our prediction analysis and modeling. We found that the logistic regression worked better when compared to the other models Random Forest, Decision Tree and Gaussian Naïve Bayes algorithms. Logistic regression gave us a prediction accuracy of 87.39%.

## ACKNOWLEDGMENT

We would like to thank our professor, Dr. Vishnu S. Pendyala for his guidance and supervision throughout the course and giving us this opportunity to understand the project aspects and its use in real world, we would also like to extend our appreciation to Teaching Assistant Darshit Jagetiya for his support.

## XVIII. APPENDIX

Table. I, provides us the summary of all the task carried out in the project.

TABLE I  
PROJECT RUBRIC

Criteria	Reference/ Remark
Visualization	Exploratory Analysis <a href="#">view</a>
Significance to the real world	Significance to the real world <a href="#">view</a>
Saving the model for quick demo	Pickle Files, these files are also placed in GitHub <a href="#">view</a>
Code Walkthrough	Will be Carried out during presentation
Report	Followed overleaf IEEE Conference
Prospects of winning competition / publication	Our research project enables the users to request for a prediction and the same data can be stored and used for future learning.
Version Control	Version Controlling through GitHub <a href="#">view</a>
Lessons learned	ML Learning and Lessons Learnt <a href="#">view</a>
Velocity	Created web applications to get Real time petition data from users <a href="#">view</a>
Innovation	extended the modeling to creating a web application for users <a href="#">view</a>
Evaluation of performance	created evaluation metrics for all models <a href="#">view</a>
Teamwork	Whole team contributed to the project success <a href="#">view</a>
Technical Difficulty	Listed technical difficulty <a href="#">view</a>
Paired Programming	We have scheduled zoom video calls at regular intervals to sit together to program the model, refer to the Minutes of Meeting Project Data245_Group5.docx for details
Practiced agile / scrum	We have scheduled zoom video calls at regular intervals to follow weekly sprint, refer to the Minutes of Meeting Project Data245_Group5.docx for details
Used Grammarly	Report was properly framed and corrected using Grammarly, refer screenshot attached Grammarly.png
Used LaTeX	This report was created using a LaTeX online editor Overleaf IEEE Conference format, code is pushed to the GitHub.
Used creative presentation techniques	Used Prezi for the presentation slides
Literature Survey	Literature Survey is covered under related work <a href="#">view</a>
GitHub Link	<a href="https://github.com/SuryavanshiYogita/DATA-245_Machine_Learning">https://github.com/SuryavanshiYogita/DATA-245_Machine_Learning</a>

## REFERENCES

- [1] D. Swain, K. Chakraborty, A. Dombe, A. Ashture and N. Valakunde, "Prediction of H1B Visa Using Machine Learning Algorithms," 2018 International Conference on Advanced Computation and Telecommunication (ICACAT), 2018, pp. 1-7, doi: 10.1109/ICACAT.2018.8933628.
- [2] Hartigan, J.A. and M.A. Wong, Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 1979. 28(1): p. 100-108.
- [3] R. Serban, A. Kupraszewicz and G. Hu, "Predicting the characteristics of people living in the South USA using logistic regression and decision tree," 2011 9th IEEE International Conference on Industrial Informatics, 2011, pp. 688-693, doi: 10.1109/INDIN.2011.6034974.
- [4] Q. Zhou, W. Lan, Y. Zhou and G. Mo, "Effectiveness Evaluation of Anti-bird Devices based on Random Forest Algorithm," 2020 7th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS), 2020, pp. 743-748, doi: 10.1109/ICCSS52145.2020.9336891.
- [5] M. Y. Helmi Setyawan, R. M. Awangga and S. R. Efendi, "Comparison Of Multinomial Naive Bayes Algorithm And Logistic Regression For Intent Classification In Chatbot," 2018 International Conference on Applied Engineering (ICAE), 2018, pp. 1-5, doi: 10.1109/INCAE.2018.8579372.
- [6] Y. Guo, Y. Zhou, X. Hu and W. Cheng, "Research on Recommendation of Insurance Products Based on Random Forest," 2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), 2019, pp. 308-311, doi: 10.1109/MLBDBI48998.2019.00069.
- [7] W. B. Zulfikar, Y. A. Gerhana and A. F. Rahmania, "An Approach to Classify Eligibility Blood Donors Using Decision Tree and Naive Bayes Classifier," 2018 6th International Conference on Cyber and IT Service Management (CITSM), 2018, pp. 1-5, doi: 10.1109/CITSM.2018.8674353.
- [8] S. Naribole, "H-1B Visa Petitions 2011-2016", 2016, Accessed on: Aug. 18, 2021. [Online]. Available: <https://www.kaggle.com/nsharan/h-1b-visa>
- [9] W. Koehrsen, "An Implementation and Explanation of the Random Forest in Python", Aug 30, 2018, Accessed on: Nov. 2, 2021. [Online]. Available: <https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76>
- [10] N. S. Chauhan, 2020, Accessed on: Nov. 12, 2021. [Online]. Available: <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>
- [11] "Normal distribution", Accessed on: Nov. 3, 2021. [Online]. Available: [https://en.wikipedia.org/wiki/Normal\\_distribution](https://en.wikipedia.org/wiki/Normal_distribution)