VISA APPROVAL PREDICTION

DOCUMENTATION BY:

TEAM NO:CSE-041

ABSTRACT:

Visa is more demanded now a days .Many applications are filed every year for the visa.

• The purpose of this project is to predict the visa approval on the basis of metadata provided using a predictive model developed using machine learning techniques.

• The metadata includes many aspects such as purpose ,area ,year ,job-title,worksite and many.

• We shall consider all aspects by which the application may be approved or otherwise rejected.

• In order to predict the status of the application we will give the dataset which contains the required data by which the machine learning algorithm can predict the approval of the application.

INTRODUCTION:

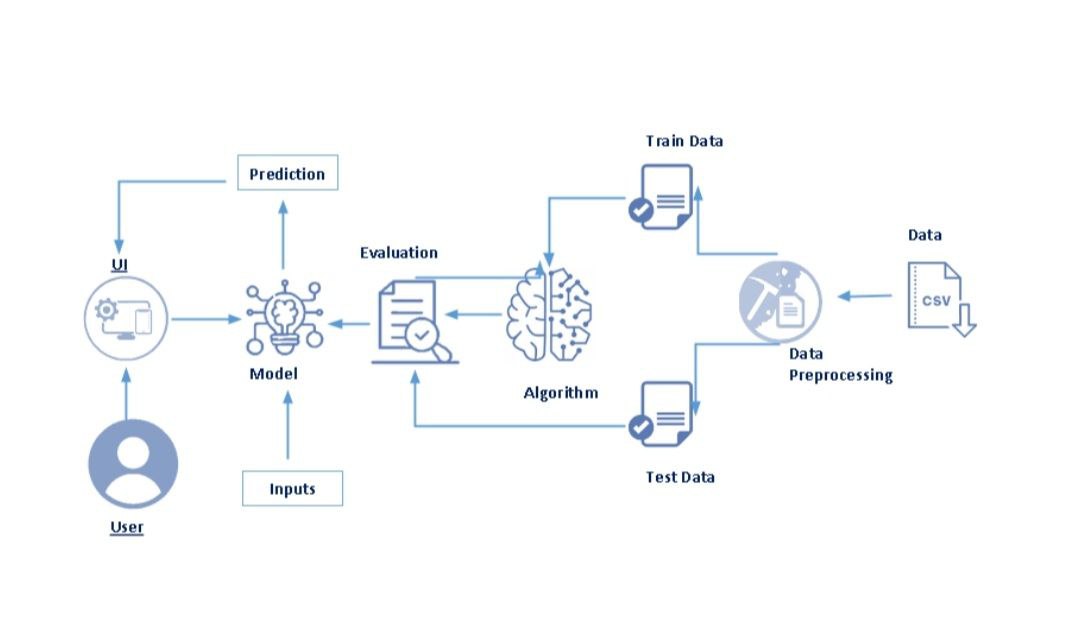
* In our project, we aim to predict the outcome of H-1B visa applications that are filed by many high-skilled foreign nationals every year. We framed the problem as a classification problem and applied Logistic Regression in order to output a predicted case status of the application. The input to our algorithm is the attributes of the applicant which will be further explained in the following parts.
* H-1B is a type of non-immigrant visa in the United States that allows foreign nationals to work in occupations that require specialized knowledge and a bachelor’s degree in the specific specialty. This visa requires the applicant to have a job offer from an employer in the US before they can file an application to the US immigration service (USCIS). USCIS grants 85,000 H-1B visas every year, even though the number of applicants far exceed that number.
* The selection process is claimed to be based on a lottery, hence how the attributes of the applicants affect the final outcome is unclear. We believe that this prediction algorithm could be a useful resource both for the future H-1B visa applicants and the employers who are considering to sponsor them. Overview To predict the outcome of H-1B visa applications based on the attributes of the applicant ,several machine learning models like Logistic Regression , random forest can be used. Finally, this can be integrated to a web application. Our aim from the project is to make use of pandas, matplotlib , & seaborn libraries from python to extract the libraries for machine learning for the Visa prediction.

LITERATURE SURVEY:

* Data mining is the process of analysing data from different perspectives and extracting useful knowledge from it. It is the core of knowledge discovery process. The various steps involved in extracting knowledge from raw data. Different data mining techniques include classification, clustering, association rule mining, prediction and sequential patterns, neural networks, regression etc. Classification is the most commonly applied data mining technique, which employs a set of preclassified examples to develop a model that can classify the population of records at large. Fraud detection and credit risk applications are particularly well suited to classification technique. This approach frequently employs Decision tree based classification Algorithm. In classification, a training set is used to build the model as the classifier which can classify the data items into its appropriate classes. A test set is used to validate the model. We use the Logistic Regression that predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class. The class with the highest probability is considered as the most likely class. And the most likely class will be the output predicted for the visa approval. And also we have created an UI using the Flask for the visa approval prediction, this UI will allow the users to predict the visa approval status very easily and the User interface is user friendly not at least one complication in using the interface, and it can be used just by entering some necessary details into the UI.

THEORETICAL ANALYSIS:

1.Block diagram: diagrammatic overview of the project.



2. Designing Hardware and Software requirements of the project

* While selecting the algorithm that gives an accurate prediction we gone through lot of algorithms which gives the results abruptly accurate and from them we selected only one algorithm for the prediction problem that is Logistic Regression .
* For the Visa status prediction. Accuracy is defined as the ratio of the number of samples correctly classified by the classifier to the total number of samples for a given test data set. The formula is as follows Accuracy=TP+TN/TP+TN+FT+FN At first we got like lot of worst accuracies because we tried lot of algorithms for the best accurate algorithm , finally after all of that we tried the best suitable algorithm which gives the prediction accurately is Logistic Regression and developed it to use as a real time prediction problem for the visa status prediction.
* 1.Google colab notebook
* 2. Spyder Ide
* 3. Machine Learning Algorithms
* 4. Python (pandas, numpy , matplotlib , seaborn , sklearn)
* 5. HTML
* 6. Flask
* We developed this Visa Approval status prediction by using the Python language which is a interpreted and high level programming language and using the Machine Learning algorithms.
* For coding we used the Google colab notebook and the Spyder IDE.
* For creating an user interface for the prediction we used the Flask. It is a micro web framework written in Python. It is classified as a micro frame work because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions
* Used HTML scripting language to create a webpage by creating the templates to use in the functions of the Flask and HTML.

EXPERIMENTAL INVESTIGATION :

The dataset we used is derived from H-1B\_Kaggle .It contains more than 10L H-1B Visa data of users.It contained 7 features and 1 label.

FULL TIME POSITION: Positions are given in ”Full Time Position = Y; Part Time Position = N” format. We converted them to ”Full Time Position = 1; Part Time Position = 0” format.

YEAR: Year in which application was filed. We converted the data into one-hot-k representation.

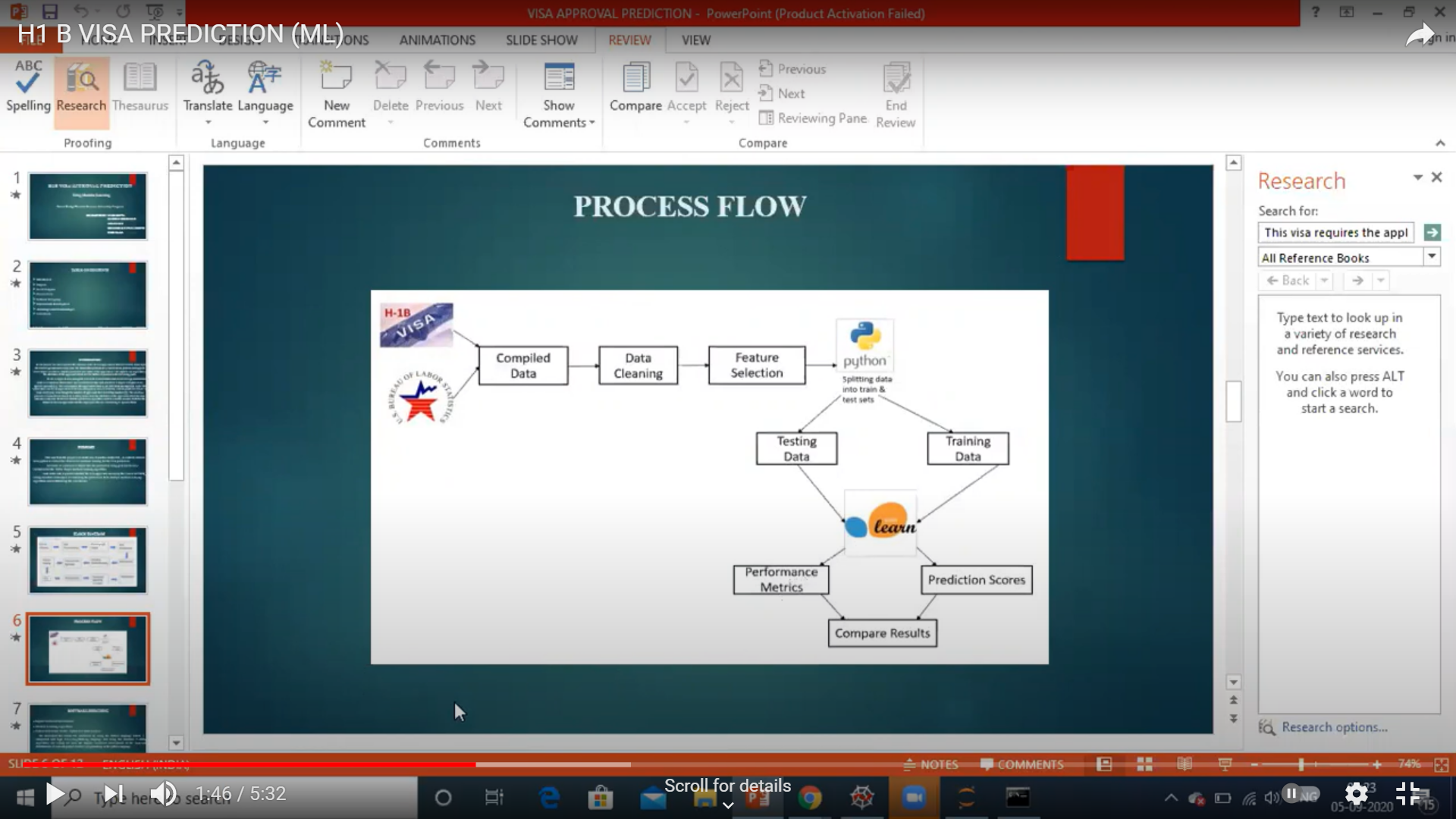
PREVAILING WAGE: Prevailing wage is the average wage paid to employees with similar qualifications in the intended area of employment. We discarded the outlier terms and used the rest of the data as it was. APPS PER EMPLOYER\_NAME: We created a feature for the number of H-1B applications per employer, and discarded data points that are petitioned by an employer that has less than 4 applications. Although this processing step undesirably gets rid of applications filed by small companies, it significantly helps with cleaning up the misspelled company names. We created a feature for the success rate per employer.

APPS PER SOC\_NAME: SOC stands for Standard Occupational Classification System, which is a federal occupational classification system. We created a feature for the number of H-1B applications per SOC type, and discarded data points with SOC types that appear less than 4 times in the data. This processing step undesirably gets rid of applications with uncommon jobs, but helps with cleaning up .

WORKSITE: Data is given in the ”City, State” format. We only included ”State” and converted the data into one-hot-k representation.

After the pre-processing steps described above, we split the dataset into train and test data.

FLOWCHART:



CODE:

from google.colab import drive

drive.mount('/content/drive')

Importing required libraries:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from matplotlib.pyplot import plot

from collections import Counter as p

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score,confusion\_matrix

Reading the datasets:

 dataset = pd.read\_csv("/content/drive/MyDrive/h1b\_kaggle.csv.zip")

Understanding the dataset

dataset.describe()

dataset.head()

dataset.info()

dataset.shape#no.of columns and rows

dataset.columns#different columns present in the dataset

dataset["YEAR"].value\_counts()

dataset["YEAR"].unique()

dataset["YEAR"].mode()

dataset.shape[0]#no.of applications

dataset.CASE\_STATUS.value\_counts()#knowing the no.of each case status

dataset['EMPLOYER\_NAME'].mode()

dataset["EMPLOYER\_NAME"].value\_counts()

dataset["SOC\_NAME"].value\_counts()

dataset["JOB\_TITLE"].value\_counts()

dataset.shape

#plotting graph for the case status

plt.figure(figsize=(10,7))

dataset.CASE\_STATUS.value\_counts().plot(kind='barh')

dataset.sort\_values('CASE\_STATUS')

plt.title("NUMBER OF APPLICATIONS")

plt.show()

dataset.YEAR.value\_counts()#no.of applications per year

#plotting graph for no.of applications per year

dataset.YEAR.value\_counts().plot(kind = 'bar')

 Analyzing top 10 applicants

 top\_emp = list(dataset['EMPLOYER\_NAME'][dataset['YEAR'] >= 2015].groupby(dataset['EMPLOYER\_NAME']).count().sort\_values(ascending=False).head(10).index)

byempyear = dataset[['EMPLOYER\_NAME', 'YEAR', 'PREVAILING\_WAGE']][dataset['EMPLOYER\_NAME'].isin(top\_emp)]

byempyear = byempyear.groupby([dataset['EMPLOYER\_NAME'], dataset['YEAR']])

 #plotting graph for top 10 applicantsplt.figure(figsize=(12,7))

markers=['o','v','^','<','>','d','s','p','\*','h','x','D','o','v','^','<','>','d','s','p','\*','h','x','D']

for company in top\_emp:

    tmp = byempyear.count().loc[company]

    plt.plot(tmp.index.values, tmp["PREVAILING\_WAGE"].values, label=company, linewidth=2,marker=markers[top\_emp.index(company)])

plt.xlabel("Year")

plt.ylabel("Number of Applications")

plt.legend()

plt.title('Number of Applications of Top 10 Applicants')

plt.show()

**Data preprocessing(1)**

#checking and removing outliers from the dataset

dataset = dataset[dataset['PREVAILING\_WAGE'] <= 500000]

by\_emp\_year = dataset[['EMPLOYER\_NAME', 'YEAR', 'PREVAILING\_WAGE']][dataset['EMPLOYER\_NAME'].isin(top\_emp)]

by\_emp\_year = by\_emp\_year.groupby([dataset['EMPLOYER\_NAME'],dataset['YEAR']])

 dataset.PREVAILING\_WAGE.max()

 dataset.isnull().any()

#checking for null values in the dataset

dataset.isnull().sum()

#print(dataset['EMPLOYER\_NAME'].mode())

dataset['EMPLOYER\_NAME'] = dataset['EMPLOYER\_NAME'].fillna(dataset['EMPLOYER\_NAME'].mode().iloc[0])

#dataset['SOC\_NAME'].mode

dataset['SOC\_NAME'] = dataset['SOC\_NAME'].fillna(dataset['SOC\_NAME'].mode().iloc[0])

#dataset['JOB\_TITLE'].mode()

dataset['JOB\_TITLE'] = dataset['JOB\_TITLE'].fillna(dataset['JOB\_TITLE'].mode().iloc[0])

#dataset['FULL\_TIME\_POSITION'].mode()

dataset['FULL\_TIME\_POSITION'] = dataset['FULL\_TIME\_POSITION'].fillna(dataset['FULL\_TIME\_POSITION'].mode().iloc[0])

#dataset['PREVAILING\_WAGE'].mean()

dataset['PREVAILING\_WAGE'].fillna(dataset['PREVAILING\_WAGE'].mean(), inplace = True)

dataset['PREVAILING\_WAGE'] = dataset['PREVAILING\_WAGE'].round()

#dataset['YEAR'].mode()

dataset['YEAR'] = dataset['YEAR'].fillna(dataset['YEAR'].mode().iloc[0])

#dataset['CASE\_STATUS'].mode()

dataset['CASE\_STATUS'] = dataset['CASE\_STATUS'].fillna(dataset['CASE\_STATUS'].mode().iloc[0])

 dataset.isnull().any()

**Data preprocessing(2)**

Label encoding:changing the string values of columns into numerical values

dataset.CASE\_STATUS.value\_counts()

dataset['CASE\_STATUS'] = dataset['CASE\_STATUS'].map({'CERTIFIED' : 0, 'CERTIFIED-WITHDRAWN' : 1, 'DENIED' : 2, 'WITHDRAWN' : 3,

                                           'PENDING QUALITY AND COMPLIANCE REVIEW - UNASSIGNED' : 4, 'REJECTED' : 5, 'INVALIDATED' : 6})

dataset.head()

 dataset['FULL\_TIME\_POSITION'] = dataset['FULL\_TIME\_POSITION'].map({'N' : 0, 'Y' : 1})

dataset.head()

dataset.SOC\_NAME.value\_counts()

import sys

dataset['SOC\_NAME1'] = 'others'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('computer','software')] = 'it'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('chief','management')] = 'manager'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('mechanical')] = 'mechanical'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('database')] = 'database'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('sales','market')] = 'scm'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('financial')] = 'finance'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('public','fundraising')] = 'pr'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('education','law')] = 'administrative'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('auditors','compliance')] = 'audit'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('distribution','logistics')] = 'scm'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('recruiters','human')] = 'hr'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('agricultural','farm')] = 'agri'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('construction','architectural')] = 'estate'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('forencsic','health')] = 'medical'

dataset['SOC\_NAME1'][dataset['SOC\_NAME'].str.contains('teachers')] = 'education'

dataset.head()

dataset.columns

 dataset= dataset.drop(['SOC\_NAME1'], axis=1)

dataset.head()

dataset.isnull().any()

Model building

x = dataset.drop(['CASE\_STATUS'], axis=1)

y = dataset['CASE\_STATUS']

 from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.2,random\_state =0)

 from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.fit\_transform(x\_test)

 from sklearn.linear\_model import LogisticRegression

log = LogisticRegression()

log.fit(x\_train,y\_train)

 ypred =  log.predict(x\_test)

  ypred

  y\_test

 from sklearn.metrics import accuracy\_score

accuracy = accuracy\_score(ypred,y\_test)

 accuracy

from sklearn.ensemble import RandomForestRegressor

reg\_rf = RandomForestRegressor()

reg\_rf.fit(x\_train, y\_train)

y\_pred = reg\_rf.predict(x\_test)

from sklearn import metrics

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

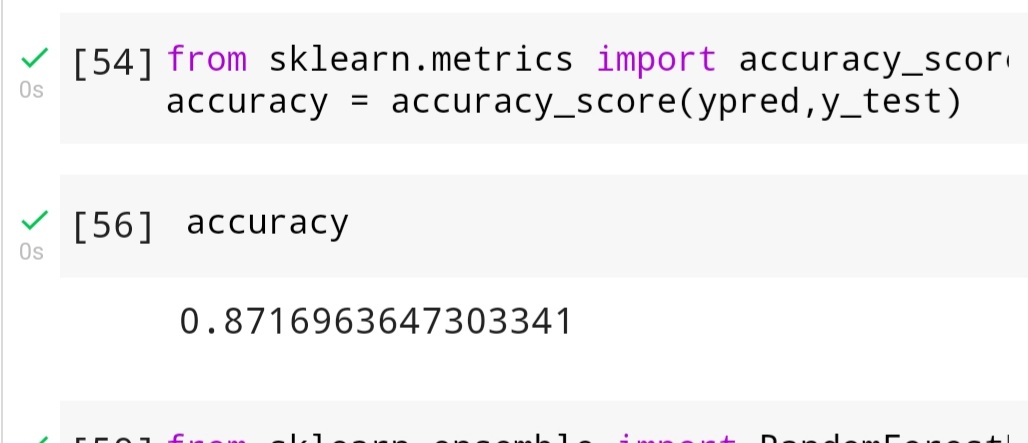
print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

 rf\_accuracy=metrics.r2\_score(y\_test, y\_pred)\*100

 rf\_accuracy

RESULT:

We used the Logistic Regression algorithm to predict the visa approval status. The logistic regression algorithm performs the best, with an accuracy of 87%.



ADVANTAGES AND DISADVANTAGES

**Advantages:**

H-1B visa benefit , and perhaps the main reason for its popularity, is the board requirements associated with qualifying for the visa.

Duration of Stay.

Portability.

Anyone Can Apply.

Dual Intent (pursue legal permanent residency) while under H-1B non-immigrant status.

**Disadvantages**:

Lottery.

Extensions.

Fees.

CONCLUSION:

In order to predict the outcome of H-1B visa applications based on the attributes of the applicant, several machine learning models like Naive Bayes , random forest can be used. Finally , this can be integrated to a web application.

FUTURE SCOPE :

In further Logistic Regression algorithm can be applied on other data sets available for visa approvals to further investigate its accuracy. A rigorous analysis of other machine learning algorithms other than this can also be done in future to investigate the power of machine learning algorithms for visa status prediction.