ECM3001

Data Analytics And Visualisation

J Component Report

A project report titled

CURBING THE PANDEMIC USING VISA PREDICION

By

19BLC1017	Priyanshu Prasad
19BLC1012	KPS Shivratna
19BLC1133	Ujjwal Gupta

Google sites link:
https://sites.google.com/vitstudent.ac.in/ecm3001jcomponent/ecm3001_project-review-2

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMPUTER ENGENEERING

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Submitted to: **Dr. C.Sweetlin Hemalatha**

DECLARATION BY THE CANDIDATE

I hereby declare that the Report entitled "**Curbing the Pandemic using Visa Prediction"** submitted by me to VIT Chennai is a record of bonafide work undertaken by me under the supervision of **Dr. Sweetlin Hemalatha, Senior Assistant Professor, SCOPE, VIT Chennai.**

Chennai

05/12/2021.

ACKNOWLEDGEMENT

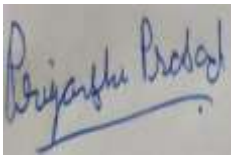
We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. C Sweetlin Hemalatha,** School of Computer Science and Engineering for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

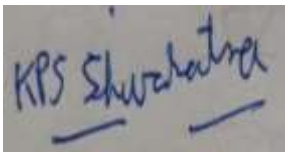
We are extremely grateful to **Dr. Sivasubramanian. A,** Dean of the School of Electronics Engineering (SENSE), VIT University Chennai, for extending the facilities of the School towards our project and for his unstinting support.

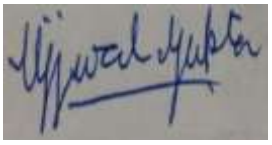
We express our thanks to our **Head of The Department Dr. Vetrivelan. P (for B.Tech-ECE)** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the courses till date.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.







Priyanshu Prasad **(19BLC1017)**

KPS Shivratna **(19BLC1012)**

Ujjwal Gupta **(19BLC1133)**

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Curbing the Pandemic using Visa Prediction"** is a bonafide work of **PRIYANSHU PRASAD (19BLC1017)** , **KPS SHIVRATNA (19BLC1012) and UJJWAL GUPTA (19BLC1133)** carried out the "J"-Project work under my supervision and guidance for ECM 3001 Data Analytics And Visualisation.

Dr.C.Sweetlin Hemalatha

School of Computer Science and Engineering

VIT University, Chennai

Chennai – 600 127.

TABLE OF CONTENTS

INTRODUCTION

Abstract

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 Gaussian Naive Bayes, Logistic Regression, Decision Tree Model,Random Forests and Artificial Neural Networks 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 or higher in the specific specialty [1]. 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 [2]. 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.

Problem Statement and Objective

The report tries to show the dependency of the decision on the attributes of the application. The attributes of the application here serve as an input and the output is the predicted decision. The data used here has the below meta-data:

Name of the employer: Name of employer submitting labor condition application.

Category of the job or SOC Name: Occupational name associated with the requested job under temporary labor condition. Standard Occupational Classification system defines the codes and the names associated with them.

Job title: the requested job title in the petition.

Employment Type: Full time employment (Y)or a part-time employment (N).

Year of Filing: Year when petition is filed (in between 2011-2016).

Prevailing wage: Prevailing Wage for the job being requested for temporary labor condition.

Location of work : State information of theforeign worker's intended area of employment

Output can be any of the two values: 1. Certified, 2.Denied. The data used for the problem has been imported from KAGGLE the link of which is : https://www.kaggle.com/nsharan/h-1b-visa

and the raw form of it held 3 million records in form of filed petitions. The distribution is in accordance to case status:

TABLE1. DISTRIBUTION OF STATUS LABELS

CERTIFIED	2615623
CERTIFIED-WITHDRAWN	202659
DENIED	94346
WITHDRAWN	89799
PENDING QUALITY AND COMPLIANCE REVIEW - UNASSIGNED	15
REJECTED	2
INVALIDATED	1

LITERATURE SURVEY

S.No.	Title	Journal/Year of publication	Dataset used	Methodologies used	Metrics used	Interpretation of Results
1	Predicting the outcome of H-1B visa using ANN algorithm Link: https://www.ijrte.org/wp-content/uploads/papers/v9i1/A2917059120.pdf	May,2020	h1b_kaggle.csv	Artificial Neural Networks	No of epochs,F1 score,Precision	The F1 Score calculated was 0.965.Precision was 0.93.No of epochs were 2000.
2	A predictive model for H1-B Visa Petition Link : http://www.ijirset.com/upload/2018/october/37_A%20Predictive.pdf	October,2018	us_perm_visas.csv	Naïve Bayes Classifier,Random Forest,XG-Boost	Precision score Recall,F1-Score,Accuracy	The accuracy of each model was between 84% to 87%.The f1 score was between 0.73 to 0.80. The precision score was between 0.87 to 0.90.
3	**An allotment of H1B work visa in .** **USA using machine learning** Link: https://www.researchgat	August,2018	OFLC Database : H1b2015_2017.csv	Decision tree using C5.0 algorithm, Neural Network,S	Time,F1 score,TPR,TNR	Accuracy:92-95% Precision-99%-91% Time-60-169 seconds

	e.net/publication/328488339_An_allotment_of_H1B_work_visa_in_USA_using_machine_learning			VM,Naïve Bayes etc		
4	Prediction of H1-B VISA Application Using Classification Link: http://journalstd.com/gallery/49-july2021asd.pdf	July,2021	h1b_kaggle.csv	Random Forests(deciding the no of decision trees to be built,entropy function)	Accuracy	The accuracy obtained was : 80.8%
5	**Predictive analytics for classification of immigration visa applications: A discriminative machine learning approach** Link: https://krex.k-state.edu/dspace/bitstream/handle/2097/38822/SharmilaVegesana2018.pdf?sequence=1&isAllowed=y	February, 2018	Data Downloaded from : U.S Bureau of Labor Statistics.	KNN,Naïve bayes,Random forests	Accuracy, F1 Score, Performance Time	The accuracy observed was between 92%-96%. The F1 score was between :0.76-0.90 The performance time was between :3.6 seconds(random forests) to 156.366 seconds(knn)
6	**Predicting the Outcome of H-1B Visa Applications CS229 Term Project Final Report**	April-2017	H-1B Visa Petitions 2011-2016 dataset	Logistic Regression ,SVM, Neural Networks	Precision, Recall,Training and	The training accuracy observed was

	Link: http://cs229.stanford.edu/proj2017/final-reports/5208701.pdf				Testing Accuracy	between 94%-98%. The testing accuracy observed was between 72%-82%. The precision observed was between 73%-81%. The Recall observed was between 96.7%-99.9%.
7	**Data Preprocessing and Analysis for H-1b Visa Petitions.** Link : http://www.iosrjen.org/Papers/Conf.19017-2019/Volume-3/9.%2049-54.pdf	May-2019	OFLC Database : 2011-2016	Data Splitting of 82000 rows into training and testing data Applying neural networks (l2 regularisation)	Split by 80% training and 20 % testing. Accuracy	98% training accuracy. 82% testing accuracy
8	H-1B VISA PETITION ANALYSIS 2016-17 Link:	September-2020	OFLC Database : FY-2017 H1-B filing data	EDA	Median salary, Total petitions	Anesthesiologist-s are the top earners.

	http://rstudio-pubs-static.s3.amazonaws.com/337170_83d3630f6e0f4f2082ba8464e9dfa7e8.html					California and Washington are the states with highest median wages. Infosys Limited filed more than twice as many h1b visa applications as Tata Consultancy did in during both the years.
9	Predicting filed H1-B Visa Petitions' Status. Link: https://www.academia.edu/37419860/IRJET-_Predicting_filed_H1-B_Visa_Petitions_Status	August 2018	us_perm_visas.csv	SVM,Naïve Bayes,Gaussian Naïve Bayes,Logistic Regression	Precision, Recall,F1 Score	Barcharts of precision,recall and f1 score suggests that Logistic Regression outperformed by a slight margin

Requirements and proposed system

❖ Initial Data Analysis

STEP1 : Importing the required libraries

The python libraries used in this project are :

- **Pandas**

  A Python library used for working with arrays. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

- **Numpy**

  Mainly used for **data analysis**. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, and Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

- **Sklearn**

  It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

- **Matplotlib**

  Matplotlib is a comprehensive library for **creating static, animated, and interactive visualizations in Python**. Matplotlib makes easy things easy and hard things possible. Create publication quality plots. Make interactive figures that can zoom, pan, update.

SCREENSHOT OF THE IMPLEMENTATION

```
##basic library - Pandas and Numpy
import pandas as pd
import numpy as np

|

## Imports for different type of classfiers
from sklearn import tree # <- Decision- Trees
from sklearn import svm # <- Support Vector Machines
import sklearn.linear_model as linear_model # <- Logisitic Regression - Sigmoid Function on the Linear Regression
from sklearn.ensemble import RandomForestClassifier # <- Random Forest Classifier
from sklearn.neural_network import MLPClassifier # <- Neural Networks
from sklearn.naive_bayes import GaussianNB # <- Gaussian Naive-Bayes Classifier

## Imports for recursive feature elimination
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

## Imports for splitting the data into training and test data
from sklearn.model_selection import train_test_split

## Imports for evaluating the different classifier models selected
import sklearn.metrics as metrics
from sklearn import preprocessing

## Data Visualisation
import matplotlib.pyplot as plt
%matplotlib inline
```

Step 2: Import the dataset and load it into a pandas dataframe for further cleaning and Analysis

In [2]:
```
## Input the data's absolute/relative path from the user
df= pd.read_csv("h1b_kaggle.csv")
```

In [25]:
```
print(len(df))
```

```
3002458
```

In [4]:
```
pd.set_option('display.max_colwidth', -1)
pd.options.mode.chained_assignment = None
```

```
<ipython-input-4-f0d25484541f>:1: FutureWarning: Passing a negative integer is deprecated in version 1.0 and will not be suppor
ted in future version. Instead, use None to not limit the column width.
  pd.set_option('display.max_colwidth', -1)
```

In [5]:
```
df.head()
```

Out[5]:

	Unnamed: 0	CASE_STATUS	EMPLOYER_NAME	SOC_NAME	JOB_TITLE	FULL_TIME_POSITION	PREVAILING_WAGE	YEAR	WORKSITE	lon	lat
0	1	CERTIFIED-WITHDRAWN	UNIVERSITY OF MICHIGAN	BIOCHEMISTS AND BIOPHYSICISTS	POSTDOCTORAL RESEARCH FELLOW	N	36067.0	2016.0	ANN ARBOR, MICHIGAN	-83.743038	42.280826
1	2	CERTIFIED-WITHDRAWN	GOODMAN NETWORKS, INC.	CHIEF EXECUTIVES	CHIEF OPERATING OFFICER	Y	242674.0	2016.0	PLANO, TEXAS	-96.698886	33.019843
2	3	CERTIFIED-WITHDRAWN	PORTS AMERICA GROUP, INC.	CHIEF EXECUTIVES	CHIEF PROCESS OFFICER	Y	193066.0	2016.0	JERSEY CITY, NEW JERSEY	-74.077642	40.728158
3	4	CERTIFIED-WITHDRAWN	GATES CORPORATION, A WHOLLY-OWNED SUBSIDIARY OF TOMKINS PLC	CHIEF EXECUTIVES	REGIONAL PRESIDEN, AMERICAS	Y	220314.0	2016.0	DENVER, COLORADO	-104.990251	39.739236
4	5	WITHDRAWN	PEABODY INVESTMENTS CORP.	CHIEF EXECUTIVES	PRESIDENT MONGOLIA AND INDIA	Y	157518.4	2016.0	ST. LOUIS, MISSOURI	-90.199404	38.627003

Step 3: CREATE A NEW COLUMN -COVIDHhotspot

A ‘1’ in the COVID hotspot indicates that the place is a Covid Hotspot and entry is **restricted** . A ‘0’ in the COVID hotspot indicates that the place is not a Covid Hotspot and entry is **allowed**.We can update this column periodically as the pandemic situation evolves.

```
In [67]: df['COVID-hotspot']=np.random.randint(0,2,df.shape[0])
```

```
In [70]: df.head()
```

Out[70]:

_STATUS	EMPLOYER_NAME	SOC_NAME	JOB_TITLE	FULL_TIME_POSITION	PREVAILING_WAGE	YEAR	WORKSITE	lon	lat	COVID-hotspot
RTIFIED-DRAWN	UNIVERSITY OF MICHIGAN	BIOCHEMISTS AND BIOPHYSICISTS	POSTDOCTORAL RESEARCH FELLOW	N	36067.0	2016.0	ANN ARBOR, MICHIGAN	-83.743038	42.280826	0
RTIFIED-DRAWN	GOODMAN NETWORKS, INC.	CHIEF EXECUTIVES	CHIEF OPERATING OFFICER	Y	242674.0	2016.0	PLANO, TEXAS	-96.698886	33.019843	1
RTIFIED-DRAWN	PORTS AMERICA GROUP, INC.	CHIEF EXECUTIVES	CHIEF PROCESS OFFICER	Y	193066.0	2016.0	JERSEY CITY, NEW JERSEY	-74.077642	40.728158	0
RTIFIED-DRAWN	GATES CORPORATION, A WHOLLY-OWNED SUBSIDIARY OF TOMKINS PLC	CHIEF EXECUTIVES	REGIONAL PRESIDEN, AMERICAS	Y	220314.0	2016.0	DENVER, COLORADO	-104.990251	39.739236	1
DRAWN	PEABODY INVESTMENTS CORP.	CHIEF EXECUTIVES	PRESIDENT MONGOLIA AND INDIA	Y	157518.4	2016.0	ST. LOUIS, MISSOURI	-90.199404	38.627003	0

DATA ANALYSIS

ANALYSIS 1: Finding the Case Status v/s Number of Petitions of the visa

Petition

```
plot_status_numberinit = df['CASE_STATUS'].value_counts().plot(title = 'CASE STATUS vs NUMBER OF PETITIONS', \
                                                               kind = 'barh', color = 'green')
plot_status_numberinit.set_xlabel("CASE STATUS")
plot_status_numberinit.set_ylabel("NUMBER OF PETITIONS")
plt.show()
print(df['CASE_STATUS'].value_counts())
```


```
CERTIFIED                                  2615623
CERTIFIED-WITHDRAWN                        202659
DENIED                                     94346
WITHDRAWN                                  89799
PENDING QUALITY AND COMPLIANCE REVIEW - UNASSIGNED    15
REJECTED                                   2
INVALIDATED                                1
Name: CASE_STATUS, dtype: int64
```

Changing the current visa status as - Rejected ,Certified or Denied

```
table_2 = df.loc[df['CASE_STATUS'].isin(["CERTIFIED", "DENIED", "REJECTED"])]
```

```
table_2['YEAR'] = table_2['YEAR'].astype(int)
table_2["EMPLOYER_NAME"] = table_2['EMPLOYER_NAME'].str.upper()
table_2['SOC_NAME'] = table_2['SOC_NAME'].str.upper()
table_2['JOB_TITLE'] = table_2['JOB_TITLE'].str.upper()
table_2['FULL_TIME_POSITION'] = table_2['FULL_TIME_POSITION'].str.upper()#datatype conversion for the year column
```

```
table_2.head()
```

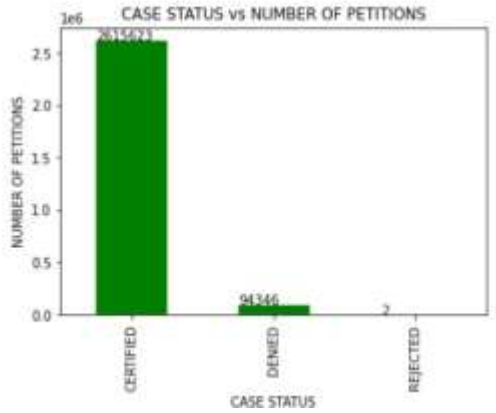
Unnamed: 0	CASE_STATUS	EMPLOYER_NAME	SOC_NAME	JOB_TITLE	FULL_TIME_POSITION	PREVAILING_WAGE	YEAR	WORKSITE	lon	
19	CERTIFIED	QUICKLOGIX LLC	CHIEF EXECUTIVES	CEO	Y	187200.0	2016	SANTA CLARA, CALIFORNIA	-121.955236	37.3541
20	CERTIFIED	MCCHRYSTAL GROUP, LLC	CHIEF EXECUTIVES	PRESIDENT, NORTHEAST REGION	Y	241842.0	2016	ALEXANDRIA, VIRGINIA	-77.046921	38.8048
23	CERTIFIED	LOMICS, LLC	CHIEF EXECUTIVES	CEO	Y	99986.0	2016	SAN DIEGO, CALIFORNIA	-117.161084	32.7157
24	CERTIFIED	UC UNIVERSITY HIGH SCHOOL EDUCATION INC	CHIEF EXECUTIVES	CHIEF FINANCIAL OFFICER	Y	99986.0	2016	CHULA VISTA, CALIFORNIA	-117.084196	32.6400
26	CERTIFIED	QUICKLOGIX, INC.	CHIEF EXECUTIVES	CEO	Y	187200.0	2016	SANTA CLARA, CALIFORNIA	-121.955236	37.3541

ANALYSIS 2:

Row Counts v/s Case Status of the visa petition

```
In [10]: plot_status_number = table_2['CASE_STATUS'].value_counts().plot(title = 'CASE STATUS vs NUMBER OF PETITIONS', \
                                                                        kind = 'bar', color = 'green')
         plot_status_number.set_xlabel("CASE STATUS")
         plot_status_number.set_ylabel("NUMBER OF PETITIONS")
         for p in plot_status_number.patches:
             plot_status_number.annotate(str(p.get_height()), (p.get_x() * 1.0050, p.get_height() * 1.005))
         plot_status_number
```

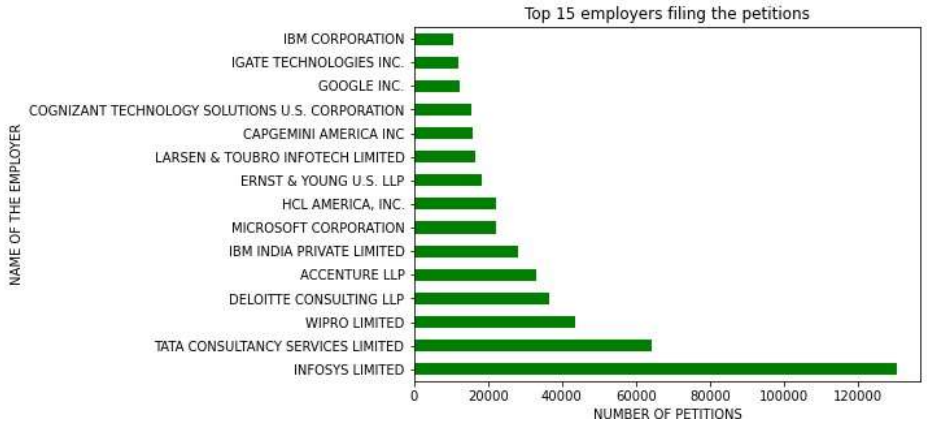
```
Out[10]: <AxesSubplot:title={'center':'CASE STATUS vs NUMBER OF PETITIONS'}, xlabel='CASE STATUS', ylabel='NUMBER OF PETITIONS'>
```



ANALYSIS 2: The top 15 employers filing the H1-B visa petitions

```
plot_status_topemp= table_2['EMPLOYER_NAME'].value_counts().head(15).plot.barh(title = "Top 15 employers filing the petitions", 
                                                    color = 'green', figsize = (7, 5))
plot_status_topemp.set_ylabel("NAME OF THE EMPLOYER")
plot_status_topemp.set_xlabel("NUMBER OF PETITIONS")
plot_status_topemp
print(table_2['EMPLOYER_NAME'].value_counts().head(15))
```

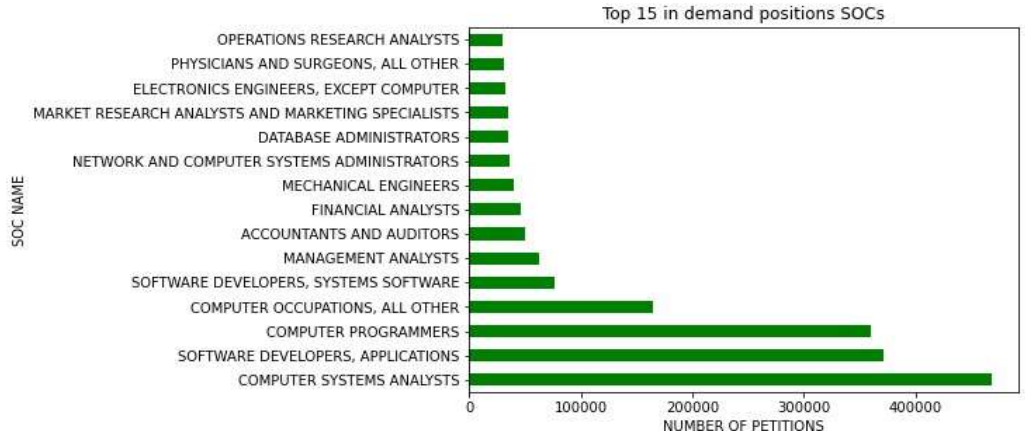
```
INFOSYS LIMITED                                 130241
TATA CONSULTANCY SERVICES LIMITED               64358
WIPRO LIMITED                                   43679
DELOITTE CONSULTING LLP                         36667
ACCENTURE LLP                                   32983
IBM INDIA PRIVATE LIMITED                       28166
MICROSOFT CORPORATION                           22373
HCL AMERICA, INC.                               22330
ERNST & YOUNG U.S. LLP                          18217
LARSEN & TOUBRO INFOTECH LIMITED                16724
CAPGEMINI AMERICA INC                           16032
COGNIZANT TECHNOLOGY SOLUTIONS U.S. CORPORATION 15448
GOOGLE INC.                                     12545
IGATE TECHNOLOGIES INC.                         12196
IBM CORPORATION                                 10690
Name: EMPLOYER_NAME, dtype: int64
```



ANALYSIS 3: The top 15 SOC names for which H1-B visas are raised

```
plot_status_topsoc= table_2['SOC_NAME'].value_counts().head(15).plot.barh(title = "Top 15 in demand positions SOCs", \
                                                                          color = 'green', figsize = (7, 5))
plot_status_topsoc.set_ylabel("SOC NAME")
plot_status_topsoc.set_xlabel("NUMBER OF PETITIONS")
plot_status_topsoc
print(table_2['SOC_NAME'].value_counts().head(15))
```

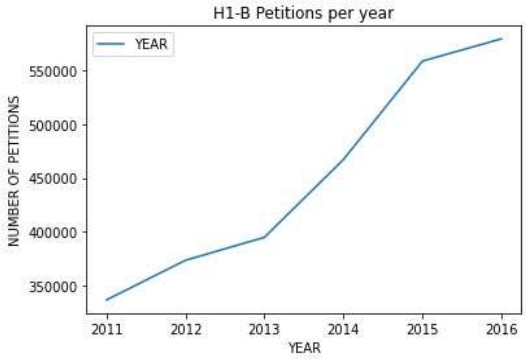
```
COMPUTER SYSTEMS ANALYSTS                          469300
SOFTWARE DEVELOPERS, APPLICATIONS                  372125
COMPUTER PROGRAMMERS                               360575
COMPUTER OCCUPATIONS, ALL OTHER                    164659
SOFTWARE DEVELOPERS, SYSTEMS SOFTWARE               75806
MANAGEMENT ANALYSTS                                 62096
ACCOUNTANTS AND AUDITORS                            49780
FINANCIAL ANALYSTS                                  46730
MECHANICAL ENGINEERS                                39844
NETWORK AND COMPUTER SYSTEMS ADMINISTRATORS         36219
DATABASE ADMINISTRATORS                             35303
MARKET RESEARCH ANALYSTS AND MARKETING SPECIALISTS  34433
ELECTRONICS ENGINEERS, EXCEPT COMPUTER              31782
PHYSICIANS AND SURGEONS, ALL OTHER                  30641
OPERATIONS RESEARCH ANALYSTS                        30328
Name: SOC_NAME, dtype: int64
```

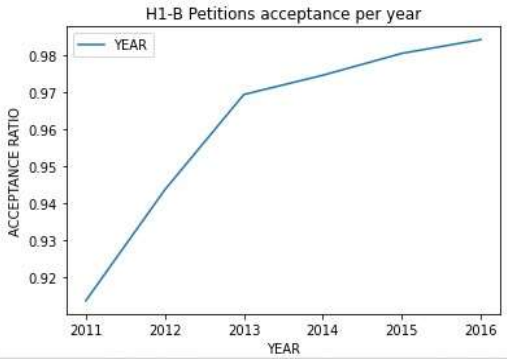


ANALYSIS 4: Acceptance rate of the H1-B Visa petitions through different years

```
dfplot_status_fyear = pd.DataFrame(table_2['YEAR'].value_counts())
dfplot_status_fyear = dfplot_status_fyear.sort_values(['YEAR'])
plot_status_fyear = dfplot_status_fyear.plot(title = 'H1-B Petitions per year', kind = 'line')
plot_status_fyear.set_xlabel('YEAR')
plot_status_fyear.set_ylabel('NUMBER OF PETITIONS')
plt.show()

dfstatus_acceptance_peryear = pd.DataFrame(table_2[table_2['CASE_STATUS'] == 'CERTIFIED'].YEAR.value_counts() / table_2.YEAR.valu
dfstatus_acceptance_peryear = dfstatus_acceptance_peryear.sort_values(['YEAR'])
status_acceptance_peryear = dfstatus_acceptance_peryear.plot(title = 'H1-B Petitions acceptance per year', kind = 'line')
status_acceptance_peryear.set_xlabel('YEAR')
status_acceptance_peryear.set_ylabel('ACCEPTANCE RATIO')
plt.show()
```

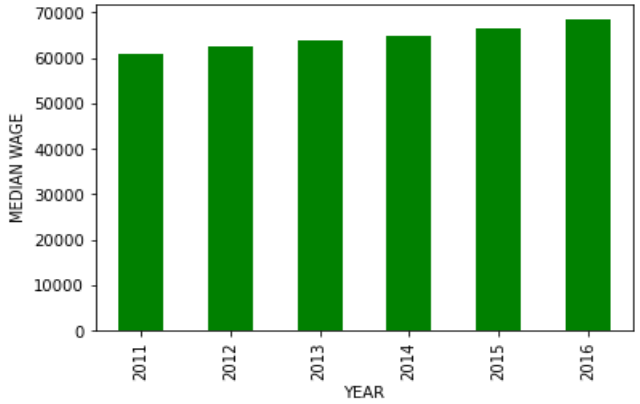




ANALYSIS 5: Median Wage of employees every year

```
dfsalaries_trends_year = table_2.loc[:,['PREVAILING_WAGE', 'YEAR']].groupby(['YEAR']).agg(['median'])

plot_salaries_trends_year = dfsalaries_trends_year.plot(kind = 'bar', color = 'g', legend = None)
plot_salaries_trends_year.set_xlabel('YEAR')
plot_salaries_trends_year.set_ylabel('MEDIAN WAGE')
plt.show()
dfsalaries_trends_year
```



	PREVAILING_WAGE
	median
YEAR	
2011	60882.0
2012	62462.0
2013	63939.0
2014	64958.0
2015	66394.0
2016	68411.0

DATA PREPROCESSING

STEP1: Filter the rows and keep the ones with case status as 'CERTIFIED' or 'DECLINED

```
print(table_2['CASE_STATUS'].unique())
table_2 = table_2.loc[table_2['CASE_STATUS'].isin(["CERTIFIED", "DENIED"])] #filtering
```

```
['CERTIFIED' 'DENIED']
```

STEP2: Remove rows with null values for EMPLOYER_NAME, SOC_NAME, JOB_TITLE, FULL_TIME_POSITION, PREVAILING_WAGE

```
table_2.isnull().sum(axis = 0)
```

```
Unnamed: 0            0
CASE_STATUS           0
EMPLOYER_NAME         18
SOC_NAME              15893
JOB_TITLE             10
FULL_TIME_POSITION    1
PREVAILING_WAGE       53
YEAR                  0
WORKSITE              0
lon                   97071
lat                   97071
dtype: int64
```

```
table_3 = table_2.dropna(axis=0, how='any', subset = ['EMPLOYER_NAME', 'SOC_NAME', 'JOB_TITLE',
                                                      'FULL_TIME_POSITION', 'PREVAILING_WAGE'])
```

STEP3: Find the number of certified and denied of all the needed columns with their count

```
: print(table_2.shape)
  print(table_3.shape)
```

```
(2709969, 11)
(2694002, 11)
```

```
: table_3.CASE_STATUS.value_counts()
```

```
: CERTIFIED    2600241
  DENIED       93761
  Name: CASE_STATUS, dtype: int64
```

STEP4: Downsampling the Data to match the ratio of certified and denied samples

```
table_temp_2_Dx = table_3[table_3['CASE_STATUS'] == 'DENIED']
#table_temp_2_Dx.duplicated(features_for_dup_removal).value_counts()
```

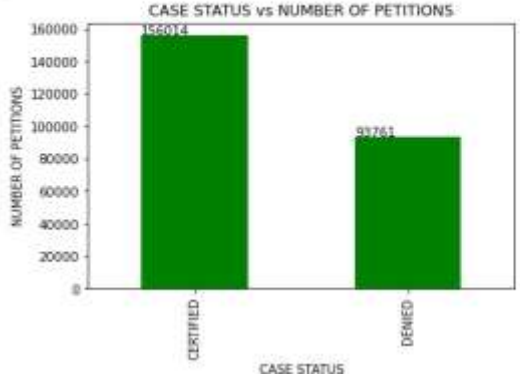
```
table_temp_2_Cx = table_3[table_3['CASE_STATUS'] == 'CERTIFIED']
#table_temp_2_Cx.duplicated(features_for_dup_removal).value_counts()
```

```
Input_Certified, Input_Certified_extra, y_certified, y_certified_extra = train_test_split(table_3[table_3.CASE_STATUS == 'CERTIFI
                                                                                            table_temp_2_Cx.CASE_STATUS, train_size

#Input_Certified is the needed x axis data
#Input_certified_extra is the eliminitated attributes data
#Same applied for the Y axis but as the values are "Certified" throughout, it doesn't matter
```

```
training_dataframe = Input_Certified.append(table_temp_2_Dx)
```

```
## plot the distribution of the certified and denied samples after downsampling
plot_after_ds = training_dataframe['CASE_STATUS'].value_counts().plot(title = 'CASE STATUS vs NUMBER OF PETITIONS', \
                                                                      kind = 'bar', color = 'green')
plot_after_ds.set_xlabel("CASE STATUS")
plot_after_ds.set_ylabel("NUMBER OF PETITIONS")
for p in plot_after_ds.patches:
    plot_after_ds.annotate(str(p.get_height()), (p.get_x() * 1.0050, p.get_height() * 1.005))
plt.show()
```



STEP5: Finding and keeping only the Unique Values

```
print("Case Status ",training_dataframe.CASE_STATUS.nunique())
print("Unique Employers ",training_dataframe.EMPLOYER_NAME.nunique())
print("Prevailing Wages ",training_dataframe.PREVAILING_WAGE.nunique())
print("Unique SOCs ", training_dataframe.SOC_NAME.nunique())
print("Unique Job Titles ",training_dataframe.JOB_TITLE.nunique())
print("Unique Filing Year ",training_dataframe.YEAR.nunique())
print("Unique Worksite State ",training_dataframe.WORKSITE.nunique())
print("Unique Employment Type ", training_dataframe.FULL_TIME_POSITION.nunique()
```

```
Case Status  2
Unique Employers  80566
Prevailing Wages  24804
Unique SOCs  983
Unique Job Titles  53272
Unique Filing Year  6
Unique Worksite State  8637
Unique Employment Type  2
```

STEP6: Feature Categorisation

On the bases of wage into :

- **VERY LOW**
- **LOW**
- **MEDIUM**
- **HIGH**
- **VERY HIGH**

On the bases of Visa acceptance rate into :

- **NAR (no acceptance rate)**
- **VLA (Very Low Acceptance Rate)**
- **LA (Low Acceptance Rate)**
- **MA (Medium Acceptance Rate_**
- **HA (High Acceptance Rate)**
- **VHA (Very High Acceptance Rate)**

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

```
def categorisation_visagrant(ratio_of_acceptance):
    if ratio_of_acceptance == -1:
        return "AR"
    elif ratio_of_acceptance >=0.0 and ratio_of_acceptance<0.20:
        return "VLA"
    elif ratio_of_acceptance>=0.20 and ratio_of_acceptance<0.40:
        return "LA"
    elif ratio_of_acceptance>=0.40 and ratio_of_acceptance<0.60:
        return "MA"
    elif ratio_of_acceptance>=0.60 and ratio_of_acceptance<0.80:
        return "HA"
    elif ratio_of_acceptance>=0.80:
        return "VHA"
```

pd.get_dummies to get the final data frame categorical variables as 0 or 1 :

```
final_df_train = pd.get_dummies(training_dataframe, columns=['FILING_YEAR', 'WORKSITE', 'FULL_TIME_POSITION', 'WAGE_CATEGORY', 'E
                                                             'JOB_ACCEPTANCE', 'SOC_ACCEPTANCE' ], drop_first=True)
final_df_train.head()
```

	CASE_STATUS	FILING_YEAR_2012	FILING_YEAR_2013	FILING_YEAR_2014	FILING_YEAR_2015	FILING_YEAR_2016	WORKSITE_ALASKA	WORKSITE_ARIZ(
39	0	0	0	0	0	1	0	
47	0	0	0	0	0	1	0	
66	1	0	0	0	0	1	0	
70	1	0	0	0	0	1	0	
91	0	0	0	0	0	1	0	

5 rows × 78 columns

Recursive Feature Elimination (RFE) to get the most relevant variables:

```
rfe = RFE(model, 30)
fit = rfe.fit(final_df_train.iloc[:,1:], final_df_train.iloc[:,0])
support_rfe = rfe.support_
length_cols = list(final_df_train.iloc[:,1:].columns.values)
list_selected = []
for index in range(len(length_cols)):
    if support_rfe[index] == True:
        list_selected.append(length_cols[index])
    else:
        pass
print(list_selected)
print(rfe.ranking_)     # ref.ranking_ returns an array with positive integer values
                        # to indicate the attribute ranking with a lower score indicating a higher ranking
```

```
C:\Users\ojjwa\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
['FILING_YEAR_2012', 'FILING_YEAR_2013', 'FILING_YEAR_2014', 'FILING_YEAR_2015', 'FILING_YEAR_2016', 'WORKSITE_ALASKA', 'WORK
SITE_DISTRICT OF COLUMBIA', 'WORKSITE_KANSAS', 'WORKSITE_KENTUCKY', 'WORKSITE_MAINE', 'WORKSITE_MISSISSIPPI', 'WORKSITE_NA',
'WORKSITE_NORTH DAKOTA', 'WORKSITE_OKLAHOMA', 'WORKSITE_SOUTH DAKOTA', 'FULL_TIME_POSITION_1', 'WAGE_CATEGORY_VERY HIGH', 'EM
PLOYER_ACCEPTANCE_HA', 'EMPLOYER_ACCEPTANCE_LA', 'EMPLOYER_ACCEPTANCE_MA', 'EMPLOYER_ACCEPTANCE_VHA', 'EMPLOYER_ACCEPTANCE_VL
A', 'JOB_ACCEPTANCE_HA', 'JOB_ACCEPTANCE_LA', 'JOB_ACCEPTANCE_MA', 'JOB_ACCEPTANCE_VHA', 'JOB_ACCEPTANCE_VLA', 'SOC_ACCEPTANC
E_LA', 'SOC_ACCEPTANCE_MA', 'SOC_ACCEPTANCE_VLA']
[ 1  1  1  1  1  1 23 14 33 11 38 25  1  9 18 15 37 22 48  6  1  1 29  1
 45 41 46 28  1 40 35  1  7 20  5 27  3 24 19  1 30  1 10 36 31 21  4  1
 26 43 47 42 44 17 13  8 34  1 16 39  1 12  1  1  1  1  1  1  1  1  1  1
  2  1  1 32  1]
```

DATA SPLITTING

Splitting into training and test data (20% testing data and 80% training data)

```
_test, y_train, y_test = train_test_split(final_df_train.iloc[:,1:], final_df_train.iloc[:, 0], test_size = 0.20, random_state=1)
_train==1].shape
est==1].shape
```

```
(31200,)
```

```
X_train.head()
```

	FILING_YEAR_2012	FILING_YEAR_2013	FILING_YEAR_2014	FILING_YEAR_2015	FILING_YEAR_2016	WORKSITE_ALASKA	WORKSITE_ARIZONA	WOR
363034	0	0	0	0	1	0	0	
2794144	0	0	0	0	0	0	0	
2298633	1	0	0	0	0	0	0	
256430	0	0	0	0	1	0	0	
2924319	0	0	0	0	0	0	0	

5 rows × 77 columns

MODEL BUILDING AND RESULT ANALYSIS

MODEL1 -DECISION TREES

A **decision tree** is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements .

Decision Tree Model

```
dtree = tree.DecisionTreeClassifier()
dtree = dtree.fit(X_train, y_train)
```

```
y_pred = dtree.predict(X_test)

y_prob = dtree.predict_proba(X_test)

print("test", y_test[:10])
print("pred", y_pred[:10])
print()

print(metrics.confusion_matrix(y_test,y_pred))
print(metrics.classification_report(y_test, y_pred))
```

```
test 1069287    1
608355     0
873557     1
2063899    0
879488     1
2280957    1
564762     0
2784493    1
2531141    1
718033     0
Name: CASE_STATUS, dtype: int64
pred [1 0 1 0 1 1 0 1 1 1]

[[14833  3922]
 [ 2223 28977]]
              precision    recall  f1-score   support

           0       0.87      0.79      0.83     18755
           1       0.88      0.93      0.90     31200

    accuracy                           0.88     49955
   macro avg       0.88      0.86      0.87     49955
weighted avg       0.88      0.88      0.88     49955
```

INFERENCE:

The F1 score for:

0 (VISA REJECTED) was : 0.83

1 (VISA ACCEPTED) was :0.90

Precision was found to be :
0 (VISA REJECTED) was : 0.87
1 (VISA ACCEPTED) was :0.88

Recall was found to be :
0 (VISA REJECTED) was : 0.79
1 (VISA ACCEPTED) was :0.93

Accuracy is found to be : 88%

▪MODEL2 –LOGISTIC REGRESSION .

The **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of 1.

Logistic Regression Model

```
lr_clf = linear_model.LogisticRegression()
lr_clf.fit(X_train, y_train)
```

```
C:\Users\ujjwa\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763:
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

```
y_pred_lr = lr_clf.predict(X_test)

probs = lr_clf.predict_proba(X_test)

print("test", y_test[:10])
print("pred", y_pred_lr[:10])

print(metrics.confusion_matrix(y_test,y_pred_lr))
print(metrics.classification_report(y_test, y_pred_lr))
)
```

```
test 1069287    1
608355     0
873557     1
2063899    0
879488     1
2280957    1
564762     0
2784493    1
2531141    1
718033     0
Name: CASE_STATUS, dtype: int64
pred [1 0 1 0 1 1 0 1 1 1]
[[14545  4210]
 [ 1238 29962]]
              precision    recall  f1-score   support

           0       0.92      0.78      0.84     18755
           1       0.88      0.96      0.92     31200

    accuracy                           0.89     49955
   macro avg       0.90      0.87      0.88     49955
weighted avg       0.89      0.89      0.89     49955
```

INFERENCE:

The F1 score for:
0 (VISA REJECTED) was : 0.84
1 (VISA ACCEPTED) was :0.92

Precision was found to be :
0 (VISA REJECTED) was : 0.92
1 (VISA ACCEPTED) was :0.88

Recall was found to be :
0 (VISA REJECTED) was : 0.78
1 (VISA ACCEPTED) was :0.96

Accuracy is found to be : 89%

MODEL3 –RANDOM FOREST

Random forests are an ensemble learning method for classification,regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned .

Random Forest Classifier

```
rf = RandomForestClassifier(n_estimators = 75, random_state = 50)
# Train the model on training data
rf.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=75, random_state=50)
```

```
y_pred_rf =  rf.predict(X_test)
probs = rf.predict_proba(X_test)

print("test", y_test[:10])
print("pred", y_pred[:10])
print(metrics.confusion_matrix(y_test,y_pred_rf))
print(metrics.classification_report(y_test, y_pred_rf))
```

```
test 1069287    1
608355     0
873557     1
2063899    0
879488     1
2280957    1
564762     0
2784493    1
2531141    1
718033     0
Name: CASE_STATUS, dtype: int64
pred [1 0 1 0 1 1 0 1 1 1]
[[14722  4033]
 [ 1907 29293]]
              precision    recall  f1-score   support

           0       0.89      0.78      0.83     18755
           1       0.88      0.94      0.91     31200

    accuracy                           0.88     49955
   macro avg       0.88      0.86      0.87     49955
weighted avg       0.88      0.88      0.88     49955
```

INFERENCE:

The F1 score for:
0 (VISA REJECTED) was : 0.83
1 (VISA ACCEPTED) was :0.91

Precision was found to be :
0 (VISA REJECTED) was : 0.89
1 (VISA ACCEPTED) was :0.88

Recall was found to be :
0 (VISA REJECTED) was : 0.78
1 (VISA ACCEPTED) was :0.94

Accuracy is found to be : 88%

MODEL4 –ARTIFICIAL NEURAL NETWORKS

To find the output of the neuron, First we must take the weighted sum of all the inputs, weighted by the *weights* of the *connections* from the inputs to the neuron. We add a *bias* term to this sum. This weighted sum is sometimes called the *activation*. This weighted sum is then passed through a (usually nonlinear) activation function to produce the output. The initial inputs are external data, such as images and documents. The ultimate outputs accomplish the task, such as recognizing an object in an image

Artificial Neural Networks

```
mlp = MLPClassifier(hidden_layer_sizes=(20,20,20,20,20), max_iter=1000)
mlp.fit(X_train, y_train)
```

```
MLPClassifier(hidden_layer_sizes=(20, 20, 20, 20, 20), max_iter=1000)
```

```
y_pred_mlp = mlp.predict(X_test)
confusion = metrics.confusion_matrix(y_test, y_pred_mlp)
print(confusion)
print(metrics.classification_report(y_test, y_pred_mlp))
d
```

```
[[14562  4193]
 [ 1272 29928]]
              precision    recall  f1-score   support

           0       0.92      0.78      0.84     18755
           1       0.88      0.96      0.92     31200

    accuracy                           0.89     49955
   macro avg       0.90      0.87      0.88     49955
weighted avg       0.89      0.89      0.89     49955
```

INFERENCE:

The F1 score for:
0 (VISA REJECTED) was : 0.84
1 (VISA ACCEPTED) was :0.92

Precision was found to be :
0 (VISA REJECTED) was : 0.92
1 (VISA ACCEPTED) was :0.88

Recall was found to be :
0 (VISA REJECTED) was : 0.78
1 (VISA ACCEPTED) was :0.96

Accuracy is found to be : 89%

MODEL5–GAUSSIAN NAÏVE BAYES CLASSIFIER

Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data..... Naive Bayes are a **group of supervised machine learning classification algorithms** based on the Bayes theorem.It is used to find the distribution of the data.

Gaussian Naive Bayes Classifier

```
gaus_clf = GaussianNB()
gaus_clf.fit(X_train, y_train)
```

```
GaussianNB()
```

```
y_pred_glb = gaus_clf.predict(X_test)
confusion = metrics.confusion_matrix(y_test, y_pred_glb)
print(confusion)
print(metrics.classification_report(y_test, y_pred_glb))
```

```
[[ 7777 10978]
 [ 2608 28592]]
              precision    recall  f1-score   support

           0       0.75      0.41      0.53     18755
           1       0.72      0.92      0.81     31200

    accuracy                           0.73     49955
   macro avg       0.74      0.67      0.67     49955
weighted avg       0.73      0.73      0.71     49955
```

INFERENCE:

The F1 score for:
0 (VISA REJECTED) was : 0.53
1 (VISA ACCEPTED) was :0.81

Precision was found to be :
0 (VISA REJECTED) was : 0.75
1 (VISA ACCEPTED) was :0.72

Recall was found to be :
0 (VISA REJECTED) was : 0.41
1 (VISA ACCEPTED) was :0.92

Accuracy is found to be : 73%

Conclusion

In terms of Accuracy the best performing model was **Logistic Regression** and **Artificial Neural Network** with a consolidated accuracy of **89 %.**

Gaussian Naïve Bayes Classifier was the worst performing model with an accuracy of **73%.**

References

- https://www.ijrte.org/wp-content/uploads/papers/v9i1/A2917059120.pdf
- http://www.ijirset.com/upload/2018/october/37_A%20Predictive.pdf
- https://www.researchgate.net/publication/328488339_An_allotment_of_H1B_work_visa_in_USA_using_machine_learning
- http://journalstd.com/gallery/49-july2021asd.pdf
- https://krex.k-state.edu/dspace/bitstream/handle/2097/38822/SharmilaVegesana2018.pdf?sequence=1&isAllowed=y
- http://cs229.stanford.edu/proj2017/final-reports/5208701.pdf
- http://www.iosrjen.org/Papers/Conf.19017-2019/Volume-3/9.%2049-54.pdf
- http://rstudio-pubs-static.s3.amazonaws.com/337170_83d3630f6e0f4f2082ba8464e9dfa7e8.html
- https://www.academia.edu/37419860/IRJET-_Predicting_filed_H1-B_Visa_Petitions_Status