A Project Report

On

H1-B Visa approval using classification

ABSTRACT

In our project,our aim is to predict the outcome of H-1B Visa applications that are applied by many skilled foreign nationals every year. The report addresses the approach to predict the case status of the filed H1-B Visa petitions using various data such as employer name, job category, job title, location of job, filing year, and prevailing wage.We framed the problem as a Classification problem and we applied the algorithms like Logistic Regression,K-Nearest Neighbours,Decision Tree,Random Forest and Naïve Bayes to predict the case status of the application that is applied for H-1B visa.The inputs to our algorithm are the attributes of the applicant.

H-1B is a type of non-immigrant visa in the United States that allows foreign countries to work in specified jobs which requires specialized knowledge and a Bachelor Degree or higher in the specific area.This visa requires the applicant to have an offer letter from an employer who is from 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 are more than the count. The selection process is claimed to be based on a lottery,hence how the attributes of the applicants affect the final outcome is unclear.

As H-1B visa is one of the highly important one.This approach can be applied by both individual and the employer in between applying for the visa,so 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.

1. INTRODUCTION

H-1B Visa is the guide of authorization on a travel permit that gives a permit to the holder to move in, leave or stay in the country for a predetermined timeframe. There are distinctive kinds of foreigner visas, the required structures, and the means in the worker visa process contingent upon the nation one needs to move. Moving to America is a vital and complex decision.

For an outside national to apply for H1B visa, a US business must offer an occupation and request to for H-1B visa with the US movement office. This is the most widely recognized visa status connected to and held by universal understudies once they finish school/advanced education (Masters,Ph.D.) and work in a full-time position. The help begins the movement methodology by recording an interest to for the remote inhabitant's purpose with U.S. Residency and Colonization Facilities (USCIS). The Office of Foreign Labour Certification (OFLC) creates program information that is helpful data about the movement programs including the H1-B visa.

It is intended to carry outside experts with professional educations and specific aptitudes to fill occupations when qualified Americans can't be found. Be that as it may, as of late, worldwide outsourcing organizations have ruled the program, winning a huge number of visas and pressing out numerous American organizations, including littler new companies. To take one vital case, in India, the quantity of first degrees presented in science and designing rose from 176 thousand of every 1990 to 455 thousand of every 2000. Second, the Act of 1990 set up the H-1B visa package for impermanent labourers in "claim to fame occupations".

The rules defines "claim to fame occupation" as requiring hypothetical and common-sense use of a collection of exceptionally particular learning in a field of human undertaking including, yet not constrained to, design, building, arithmetic, physical sciences, sociologies, solution and wellbeing, instruction, law, bookkeeping, business fortes, religious philosophy, and expressions of the human experience.

Furthermore, candidates are required to have achieved a four year certification or it's identical as a base. Firms that desire to contract non-natives on H1B visas have needs to file a Labour Condition Application (LCA) . In LCA's for H-1B specialists, the business must bear witness to that the firm will pay the non-foreigner the more prominent of the genuine remuneration paid to different representatives in a similar activity or the common pay for that occupation, and the firm will give working conditions to the non- migrant that don't make the working states of alternate workers be unfavourably affected.

By then, planned H-1B non-outsiders must exhibit to the US Citizenship and Immigration Services Bureau (USCIS) in America.In spite of the fact that H1B visa contributed a considerable measure to the economy of USA by bringing the skilled non-natives, it additionally influences American work [1]. They lose their employment, as firms incline toward modest work when contrasted with American's. The objective of the H1B program is to connect a work hole in the U.S without influencing U.S specialists.

At to start with,the structure of H1B is to fill work hole however current structure encourages businesses to augment the work hole as there aren't any qualified U.S specialists and they are procuring modest remote labourers as H1B program. There are 3 primary targets of the H1B program: Section 1) To connect a work hole without dislodging U.S specialists forever. Section 2) Review the present structure of H1B program, concentrating on the way toward acquiring H1B visa and what it enables its holders to do. It gives two classifications of the run the show: Qualification of remote specialists.

a) The framework guarantees that outsiders don't dislodge U.S specialists.

Section 3) Effect of H1B structure on compensation framework. Paying non-natives not exactly or equivalent to U.S representatives to make a disincentive U.S specialists

So, from a Machine Learning perspective, this challenge posses a possibility of an underlying pattern that can be identified by training a binary classifier. This implies, we just need to focus on the target column of our dataset and extract relevant features that might form a pattern which can be recognised by our classifier.

The main objectives of current work are as follows:

1) Detail investigation of effectively existing machine learning systems and upgraded Machine Learning approaches for a better forecast.

2) Approve different models in the wake of observing at on premise insights estimations for using sensible endorsement framework.

3) Finally, we prepare a proposed model with the marked data to foresee future petitions as a right one or mishandle and then validate it by using suitable validation technique

2. Data Collection

H-1B Visa for the Classifier was acquired from an open Kaggle dataset, a collection of 12000 articles that span various genres .

By the analysis done by some organisations, U.S receives more than 1,00,000 applications from all nations every year.But many of them are rejected because those visas are given in lottery method.So our main aim is to build a machine learning model that predicts the likelihood of H1-B visa application.Hence with this aim in mind we decided to create a model with H1-B visa petitions 2011-2016 dataset from the Kaggle.

H-1B Visa database contains approximately 3 million records overall. The columns in the dataset include case status, employer name, worksite coordinates, job title, prevailing wage, occupation code, and year filed.

Figure 1 H-1B Visa

The certified data is now clubbed with the non certified news data. The ratio of the certified to non-certified news was estimated on the basis of empirical evidence with an attempt to replicated the state of published news in the real world. I used 27,07,835 certified applicants with 85178 non-certified applicants for a total of around 28,00,000 entries as my complete dataset.

The dataset of H1-B visa is first filtered by removing Not A Number(NaN) , rows with no entries , Columns which are not useful and the attributes other than certified which belongs to CASE_STATUS column are to be defined under Denied status.

After filtering, the certified and non-certified they are clubbed together into a dataset. This dataset is shuffled and divided into ‘train-dataset’ and ‘test_dataset’ in the ratio 80%:20%.

3. DATA REPRESENTATION

The dataset that we are using is **H-1B visa petitions 2011-2016** which is derived from the Kaggle dataset. H-1B Visa database contains approximately 3 million records overall.The following are the steps used to solve the problem.

First five records of the dataset are as follows

```
In [4]: data.head()
```

Out[4]:

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

Figure 2: Head of H1-B visa Dataset

There are total of 11 columns in our dataset each consisting of some specific records.The values which are stored and meaning of each column are explained below

- Unnamed :0 Column -It is a not named column,which stores ID of the row
- CASE_STATUS - It indicates the status of the application

- EMPLOYER_NAME -The name of the employer as registered in H1-B visa application
- SOC_NAME - The occupation code for the employment
- JOB_TITLE - The job title for the employment
- FULL_TIME_POSITION - Indicates whether the application is for full time or for the part time employment
- PREVAILING_WAGE - The most frequent wage for a corresponding role as filled in the visa application
- YEAR -The application year
- WORKSITE -The address of the employer worksite
- LON -Longitude of employer worksite
- LAT -Latitude of employer worksite

Description of each column of the dataset is given below:

```
In [9]: data.info()
        data.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2877765 entries, 0 to 3002444
Data columns (total 11 columns):
 #   Column               Dtype
---  ------               -----
 0   Unnamed: 0           int64
 1   CASE_STATUS          object
 2   EMPLOYER_NAME        object
 3   SOC_NAME             object
 4   JOB_TITLE            object
 5   FULL_TIME_POSITION   object
 6   PREVAILING_WAGE      float64
 7   YEAR                 float64
 8   WORKSITE             object
 9   lon                  float64
 10  lat                  float64
dtypes: float64(4), int64(1), object(6)
memory usage: 263.5+ MB
```

Out[9]:

	Unnamed: 0	PREVAILING_WAGE	YEAR	lon	lat
count	2.877765e+06	2.877765e+06	2.877765e+06	2.877765e+06	2.877765e+06
mean	1.489381e+06	1.451666e+05	2.013877e+03	-9.212937e+01	3.815896e+01
std	8.659624e+05	5.307856e+06	1.675226e+00	1.965994e+01	4.674872e+00
min	1.000000e+00	0.000000e+00	2.011000e+03	-1.578583e+02	1.343719e+01
25%	7.404220e+05	5.460000e+04	2.012000e+03	-1.118999e+02	3.416536e+01
50%	1.484061e+06	6.512500e+04	2.014000e+03	-8.615807e+01	3.910312e+01
75%	2.236965e+06	8.151500e+04	2.015000e+03	-7.551381e+01	4.088374e+01
max	3.002445e+06	6.997607e+09	2.016000e+03	1.457298e+02	6.483778e+01

Figure 3 : Description of H1-B visa Dataset

In this dataset our target column is **CASE_STATUS** as it consists of the information whether a visa is certified or not.And in the CASE_STATUS there are 6 unique values that represents whether an application is approved or not.Those unique values are as follows.

```
In [8]: data.CASE_STATUS.value_counts()

Out[8]: CERTIFIED                                    2512114
        CERTIFIED-WITHDRAWN                          195721
        DENIED                                        85161
        WITHDRAWN                                     84752
        PENDING QUALITY AND COMPLIANCE REVIEW - UNASSIGNED       15
        INVALIDATED                                       1
        REJECTED                                          1
        Name: CASE_STATUS, dtype: int64
```

Figure 4: Values of CASE_STATUS column from H1-B visa Dataset

1.Data Normalisation:

Then we performed the data normalisation in which the main aim is to remove the unwanted or unusable entries or unusable columns.At first we merged all the records other than Certified are merged under the Denied status and then we removed LAT and LON columns.Then we created a new column called as NEW_EMPLOYER and then we dropped EMPLOYER_NAME,Worksite and Unnamed:0 so as to normalise the data.The above process is shown below.

```
In [10]: import warnings
         warnings.filterwarnings("ignore")
         data.CASE_STATUS[data['CASE_STATUS']=='REJECTED'] = 'DENIED'
         data.CASE_STATUS[data['CASE_STATUS']=='INVALIDATED'] = 'DENIED'
         data.CASE_STATUS[data['CASE_STATUS']=='PENDING QUALITY AND COMPLIANCE REVIEW - UNASSIGNED'] = 'DENIED'
         data.CASE_STATUS[data['CASE_STATUS']=='CERTIFIED-WITHDRAWN'] = 'CERTIFIED'
         data = data.drop(data[data.CASE_STATUS == 'WITHDRAWN'].index)

In [11]: data.CASE_STATUS.value_counts()

Out[11]: CERTIFIED    2707835
         DENIED         85178
         Name: CASE_STATUS, dtype: int64
```

Figure 5 : Considering all values other than Certified as Denied

```
In [12]: data = data.drop('lat', axis = 1)
         data = data.drop('lon', axis = 1)

In [13]: data.head()
```

Out[13]:

	Unnamed: 0	CASE_STATUS	EMPLOYER_NAME	SOC_NAME	JOB_TITLE	FULL_TIME_POSITION	PREVAILING_WAGE	YEAR	WORKSITE
0	1	CERTIFIED	UNIVERSITY OF MICHIGAN	BIOCHEMISTS AND BIOPHYSICISTS	POSTDOCTORAL RESEARCH FELLOW	N	36067.0	2016.0	ANN ARBOR, MICHIGAN
1	2	CERTIFIED	GOODMAN NETWORKS, INC.	CHIEF EXECUTIVES	CHIEF OPERATING OFFICER	Y	242674.0	2016.0	PLANO, TEXAS
2	3	CERTIFIED	PORTS AMERICA GROUP, INC.	CHIEF EXECUTIVES	CHIEF PROCESS OFFICER	Y	193066.0	2016.0	JERSEY CITY, NEW JERSEY
3	4	CERTIFIED	GATES CORPORATION, A WHOLLY-OWNED SUBSIDIARY O...	CHIEF EXECUTIVES	REGIONAL PRESIDEN, AMERICAS	Y	220314.0	2016.0	DENVER, COLORADO
5	6	CERTIFIED	BURGER KING CORPORATION	CHIEF EXECUTIVES	EXECUTIVE V P, GLOBAL DEVELOPMENT AND PRESIDEN...	Y	225000.0	2016.0	MIAMI, FLORIDA

```
In [14]: data['NEW_EMPLOYER'] = np.nan
         data.shape

Out[14]: (2793013, 10)
```

Figure 6: Dropping LAT and LON columns

```
In [15]: data['EMPLOYER_NAME'] = data['EMPLOYER_NAME'].str.lower()
         data.NEW_EMPLOYER[data['EMPLOYER_NAME'].str.contains('university')] = 'university'
         data['NEW_EMPLOYER']= data.NEW_EMPLOYER.replace(np.nan, 'non university', regex=True)
```

```
In [16]: data.head()
```

Out[16]:

	Unnamed: 0	CASE_STATUS	EMPLOYER_NAME	SOC_NAME	JOB_TITLE	FULL_TIME_POSITION	PREVAILING_WAGE	YEAR	WORKSITE	NEW_EMP
0	1	CERTIFIED	university of michigan	BIOCHEMISTS AND BIOPHYSICISTS	POSTDOCTORAL RESEARCH FELLOW	N	36067.0	2016.0	ANN ARBOR, MICHIGAN	ur
1	2	CERTIFIED	goodman networks, inc.	CHIEF EXECUTIVES	CHIEF OPERATING OFFICER	Y	242674.0	2016.0	PLANO, TEXAS	non ur
2	3	CERTIFIED	ports america group, inc.	CHIEF EXECUTIVES	CHIEF PROCESS OFFICER	Y	193066.0	2016.0	JERSEY CITY, NEW JERSEY	non ur
3	4	CERTIFIED	gates corporation, a wholly-owned subsidiary o...	CHIEF EXECUTIVES	REGIONAL PRESIDEN, AMERICAS	Y	220314.0	2016.0	DENVER, COLORADO	non ur
5	6	CERTIFIED	burger king corporation	CHIEF EXECUTIVES	EXECUTIVE V P, GLOBAL DEVELOPMENT AND PRESIDEN...	Y	225000.0	2016.0	MIAMI, FLORIDA	non ur

Figure 7 : Replacing Values with other one

```
In [17]: ## Splitting city and state and capturing state in another variable
         data['state'] = data.WORKSITE.str.split(',').str[-1]
```

```
In [18]: data.head()
```

Out[18]:

	Unnamed: 0	CASE_STATUS	EMPLOYER_NAME	SOC_NAME	JOB_TITLE	FULL_TIME_POSITION	PREVAILING_WAGE	YEAR	WORKSITE	NEW_EMP
0	1	CERTIFIED	university of michigan	BIOCHEMISTS AND BIOPHYSICISTS	POSTDOCTORAL RESEARCH FELLOW	N	36067.0	2016.0	ANN ARBOR, MICHIGAN	ur
1	2	CERTIFIED	goodman networks, inc.	CHIEF EXECUTIVES	CHIEF OPERATING OFFICER	Y	242674.0	2016.0	PLANO, TEXAS	non ur
2	3	CERTIFIED	ports america group, inc.	CHIEF EXECUTIVES	CHIEF PROCESS OFFICER	Y	193066.0	2016.0	JERSEY CITY, NEW JERSEY	non ur
3	4	CERTIFIED	gates corporation, a wholly-owned subsidiary o...	CHIEF EXECUTIVES	REGIONAL PRESIDEN, AMERICAS	Y	220314.0	2016.0	DENVER, COLORADO	non ur
5	6	CERTIFIED	burger king corporation	CHIEF EXECUTIVES	EXECUTIVE V P, GLOBAL DEVELOPMENT AND PRESIDEN...	Y	225000.0	2016.0	MIAMI, FLORIDA	non ur

```
In [19]: data = data.drop('EMPLOYER_NAME', axis = 1)
         data = data.drop('WORKSITE', axis = 1)
         data = data.drop('Unnamed: 0', axis = 1)
```

Figure 8 : Splitting city and state and capturing state to another variable

```
In [20]: data.head()
```

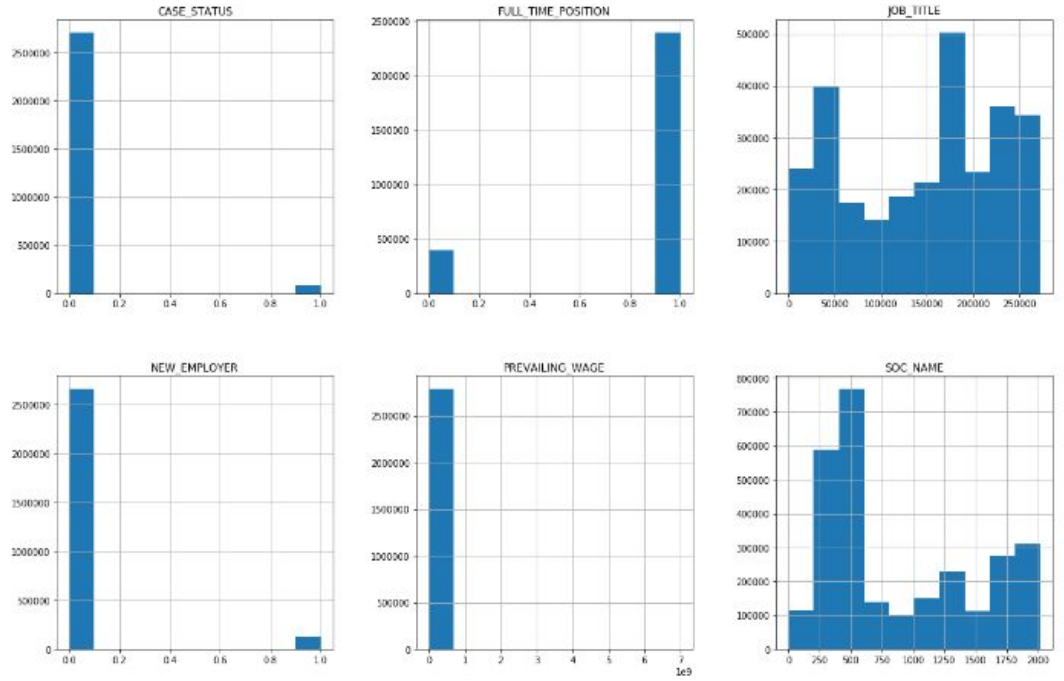
Out[20]:

	CASE_STATUS	SOC_NAME	JOB_TITLE	FULL_TIME_POSITION	PREVAILING_WAGE	YEAR	NEW_EMPLOYER	state
0	CERTIFIED	BIOCHEMISTS AND BIOPHYSICISTS	POSTDOCTORAL RESEARCH FELLOW	N	36067.0	2016.0	university	MICHIGAN
1	CERTIFIED	CHIEF EXECUTIVES	CHIEF OPERATING OFFICER	Y	242674.0	2016.0	non university	TEXAS
2	CERTIFIED	CHIEF EXECUTIVES	CHIEF PROCESS OFFICER	Y	193066.0	2016.0	non university	NEW JERSEY
3	CERTIFIED	CHIEF EXECUTIVES	REGIONAL PRESIDEN, AMERICAS	Y	220314.0	2016.0	non university	COLORADO
5	CERTIFIED	CHIEF EXECUTIVES	EXECUTIVE V P, GLOBAL DEVELOPMENT AND PRESIDEN...	Y	225000.0	2016.0	non university	FLORIDA

Figure 9: Head of dataset after Normalisation

2.<u>Data Visualization:</u> Then we performed data visualisation between the columns of the dataset and we plotted a histogram as follows

```
In [24]: data.hist(figsize=(20,20))
         plt.show()
```



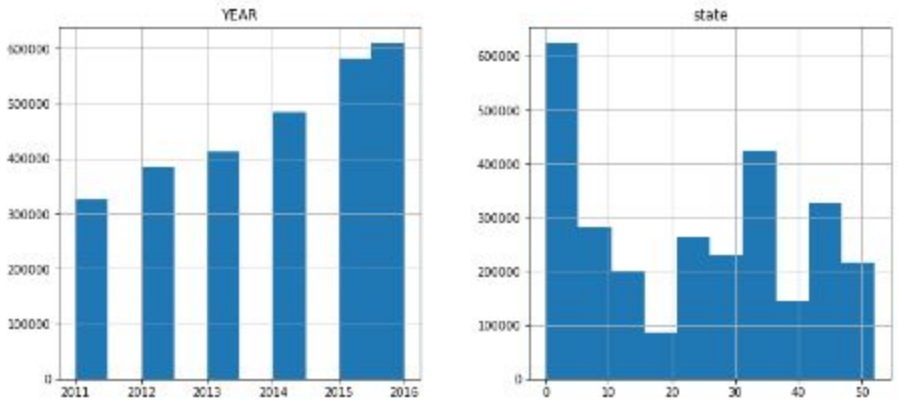


Figure 10: Histogram drawn for columns

Then we plotted the countplot between CASE_STATUS and count to know how many applications are certified and how many are rejected.The resultant graph is as follows:

```
In [29]: sns.countplot(x=data['CASE_STATUS'], data=data)
         plt.show()
```

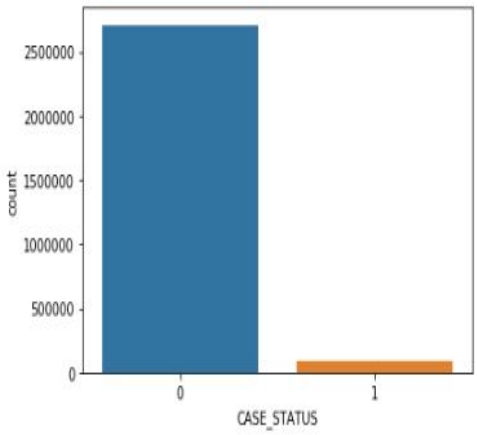


Figure 11: Count plot drawn to know how many applications accepted or not

We can see that there are more number of certified class like more than 25,00,000 and there are very few denied class records,which tells that certified visa applications are much more than rejected visa applications.

4.Models

Several different Classifying Models were implemented to accurately predict if the H-1B visa for an applicant is cerfied or not certified.The training dataset and test dataset from above data preprocessing is used to train and check the reliability of our model. The Accuracy and F1 Scores on the test set are reported for each model. Here, the F1 Score is a more reliable testing parameter as the ratio of the Certified to Non-Certified is not 0.5, rather it was estimated on the basis of empirical evidence to replicate the state of published news in the real world.

1. Logistic Regression

A Logistic Regression algorithm is used to create a binary classifier that is optimised on our training dataset. The Logistic Regression function from sklearn library is used to create and train our classifier. We did not used any parameters of the function.

The model was trained after dimensionality reduction. Then the model was tested for accuracy on the test dataset and a Confusion Matrix was plotted along with test Accuracy, Precision, Recall and F1Score was reported.

Results :

The training and then prediction for the test set was performed using the same total dataset. For every iteration the dataset was split randomly into Training

and Test set then the dimensionality reduction was carried out on the training and the test set. The model performance in terms of the Accuracy, Precision. Recall and F1 Score was recorded and the final results are as follows.

```
In [25]: from sklearn.model_selection import train_test_split
         x = data.drop('CASE_STATUS', axis=1)
         y = data.CASE_STATUS
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=7)

In [26]: from sklearn.linear_model import LogisticRegression
         lg=LogisticRegression()
         lg.fit(x_train,y_train)
         pred=lg.predict(x_test)

In [27]: from sklearn.metrics import classification_report, confusion_matrix
         print('Classification Report:')
         print(classification_report(y_test, pred))
         print('\n')
         print('Confusion Matrix:')
         print(confusion_matrix(y_test, pred))

Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98    541610
           1       0.97      0.02      0.03     16993

    accuracy                           0.97    558603
   macro avg       0.97      0.51      0.51    558603
weighted avg       0.97      0.97      0.96    558603


Confusion Matrix:
[[541601      9]
 [ 16715    278]]

In [28]: from sklearn.metrics import accuracy_score
         print(accuracy_score(y_test, pred))

0.9700610272411713
```

Figure 12 :Precision,Recall,F1 score,Confusion matrix,Accuracy for Logistic Regression without Undersampling

F1 score,Recall and Confusion Matrix for Certified status is very low when compared to Non Certified.So we performed UnderSampling using NearMiss algorithm and the results are as follows

```
In [30]: from imblearn.under_sampling import NearMiss
         nm = NearMiss()
         x_res, y_res = nm.fit_resample(x, y)

In [31]: x_res.shape
Out[31]: (170356, 7)

In [32]: from sklearn.linear_model import LogisticRegression
         X_train, X_test, Y_train, Y_test = train_test_split(x_res,y_res, test_size=0.2, random_state=42)
         clf = LogisticRegression().fit(X_train, Y_train)
         Y_Test_Pred = clf.predict(X_test)

In [33]: from sklearn.metrics import classification_report, confusion_matrix
         print('Classification Report:')
         print(classification_report(Y_test,Y_Test_Pred))
         print('\n')
         print('Confusion Matrix:')
         print(confusion_matrix(Y_test, Y_Test_Pred))

Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.70      0.69     16942
           1       0.69      0.67      0.68     17130

    accuracy                           0.68     34072
   macro avg       0.68      0.68      0.68     34072
weighted avg       0.68      0.68      0.68     34072


Confusion Matrix:
[[11783  5159]
 [ 5608 11522]]

In [34]: from sklearn.metrics import accuracy_score
         print(accuracy_score(Y_test, Y_Test_Pred))

0.6839927212960789
```

Figure 13 :Precision,Recall,F1 score,Confusion matrix,Accuracy for Logistic Regression with Undersampling

After performing the UnderSampling the precision,recall,F1 Score for Non-Certified increased exponentially but in case of Certified there is decrease in above parameters.And the total accuracy of this model is dropped from 97% to 68%.So this algorithm is not efficient to work with.So we performed K-Nearest Neighbours algorithm.

2. K-Nearest Neighbors Algorithm

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive

problems.KNN algorithm assumes the similarity between the new case/data and available case and put the new case into the category that is most similar to the available categories.KNN algorithm stores all the available data and classifies a new data point based on the similarity.This means when new data appears then it can be easily classified into a well suite category by using KNN algorithm.KNN is non-parametric algorithm,which means it does not makes any assumptions on underlying data.It is also called as a Lazy Learner Algorithm because it does not learn from the training set immediately instead it stores that dataset and at the time of classification,it performs an action on the dataset.KNN algorithm at the time of the training,it just stores the dataset and when it gets the new data,then it classifies that data into a category that is much similar to the new data.The parameters of the function used :

- **n_neighbors**: It specifies the number of neighbors to use by default for k-neighbors queries . It takes integer values as parameter and the default value=5

To find the exact k-value we applied the elbow method and plotted the graph for different k values vs Error rate

```
#choosing the right K value
error_rate = []
for i in range(1, 20):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(X_train, Y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != Y_test))
```

```
plt.figure(figsize =(10, 6))
plt.plot(range(1, 20), error_rate, color ='blue',
                linestyle ='dashed', marker ='o',
         markerfacecolor ='red', markersize = 10)

plt.title('Error Rate vs K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

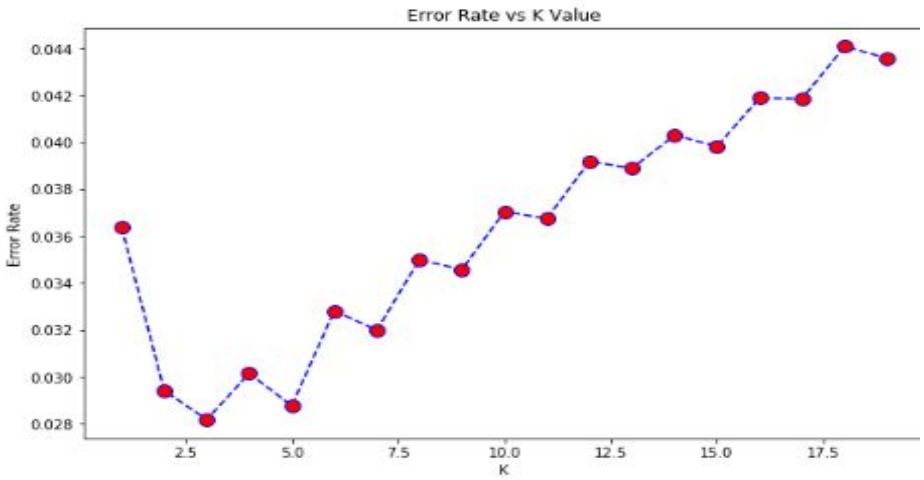


Figure 14 : Plot of error rate vs K-Value

By looking at the plot we can observe that at k=3 the error rate is very low,so we train the knn algorithm by n_neighbors=3.

Results :

The training and then prediction for the test set was performed using the same total dataset. For every iteration the dataset was split randomly into Training and Test set then the dimensionality reduction was carried out on the training and the test set. The model performance in terms of the Accuracy, Precision, Recall and F1 Score are as follows.

```
In [66]: from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors = 3)
         knn.fit(X_train, Y_train)
         k_pred = knn.predict(X_test)

In [67]: print('Classification Report:')
         print(classification_report(Y_test,pred))
         print('\n')
         print('Confusion Matrix:')
         print(confusion_matrix(Y_test, k_pred))

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     16942
           1       0.99      0.95      0.97     17130

    accuracy                           0.97     34072
   macro avg       0.97      0.97      0.97     34072
weighted avg       0.97      0.97      0.97     34072


Confusion Matrix:
[[16758   184]
 [  776 16354]]

In [68]: print(accuracy_score(Y_test, k_pred))

0.9718243719182906
```

Figure 15 :Precision,Recall,F1 score,Confusion matrix,Accuracy for KNN

3. Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A decision tree is a flowchart-like structure in which each internal node represents a “test” on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents

a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

Tree based learning algorithms are considered to be one of the best and mostly used supervised learning methods. Tree based methods empower predictive models with high accuracy, stability and ease of interpretation. Unlike linear models, they map non-linear relationships quite well. They are adaptable at solving any kind of problem at hand (classification or regression). The parameters of the function used :

- **Criterion:** The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.
  It takes these parameters:*{"gini", "entropy"}, default="gini"*

Result :

The training and then prediction for the test set was performed using the same total dataset. For every iteration the dataset was split randomly into Training and Test set then the dimensionality reduction was carried out on the training and the test set. The model performance in terms of the Accuracy, Precision. Recall and F1 Score was recorded and the final results are as follows.

```
In [38]: from sklearn.tree import DecisionTreeClassifier
         dtc = DecisionTreeClassifier(criterion='entropy')
         dtc.fit(X_train, Y_train)
         d_pred = dtc.predict(X_test)

In [39]: print('Classification Report:')
         print(classification_report(Y_test,d_pred))
         print('\n')
         print('Confusion Matrix:')
         print(confusion_matrix(Y_test,d_pred))

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     16942
           1       0.99      0.96      0.97     17130

    accuracy                           0.98     34072
   macro avg       0.98      0.98      0.98     34072
weighted avg       0.98      0.98      0.98     34072



Confusion Matrix:
[[16835   107]
 [  740 16390]]

In [40]: print(accuracy_score(Y_test,d_pred))

0.9751408781404085
```

Figure 16 :Precision,Recall,F1 score,Confusion matrix,Accuracy for Decision Tree Classifier

4. Random Forest

The random forest algorithm creats a forest with a number of Decision Trees. It is a type of Ensemble machine learning algorithm, which use a divide-and-conquer approach. The main principle behind ensemble algorithms is **boosting**, that is a group of weak learners (single estimator or a decision tree) can work together to form a strong learner (group of estimators or a forest) to classify the data. The random decision forests can correct for the decision trees' habit of overfitting to the training dataset. Hence, random forest algorithm comprises of **bagging** (Bootstrap aggregating), which is the approach to reduce overfitting by combining the classifications of randomly generated training sets, together with the random selection of features to construct a collection of decision forests.

The Random Forest Classifier is created using the RandomForestClassifier function of sklearn library. The parameters of the function used :

- **n_estimators** : The number of decision trees in the forest, We selected 11.

The model was trained using the tfidf vector after dimensionality reduction. The model can also show the feature importance of all the tokens based on the gini importance criterion.

The feature importance graph is similar to the feature variance plot after the dimensionality reduction.

Then the model was tested for accuracy on the test dataset and a Confusion Matrix was plotted along with test Accuracy, Precision, Recall and F1Score was reported.

Results :

The training and then prediction for the test set was performed using the same total dataset. For every iteration the dataset was split randomly into Training and Test set then the dimensionality reduction was carried out on the training and the test set. The model performance in terms of the Accuracy, Precision, Recall and F1 Score are as follows.

```
In [41]: from sklearn.ensemble import RandomForestClassifier
         rf=RandomForestClassifier(n_estimators=11)
         rf.fit(X_train,Y_train)
         r_pred=rf.predict(X_test)

In [42]: print('Classification Report:')
         print(classification_report(Y_test,r_pred))
         print('\n')
         print('Confusion Matrix:')
         print(confusion_matrix(Y_test,r_pred))

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     16942
           1       0.99      0.96      0.98     17130

    accuracy                           0.98     34072
   macro avg       0.98      0.98      0.98     34072
weighted avg       0.98      0.98      0.98     34072


Confusion Matrix:
[[16820   122]
 [  646 16484]]

In [84]: print(accuracy_score(Y_test,r_pred))

0.9774594975346326
```

Figure 17 :Precision,Recall,F1 score,Confusion matrix,Accuracy for Random Forest

Then we applied Grid Search to the above model and we got the best parameters as follows:

```
In [57]: rf1=RandomForestClassifier()
         param_grid = {
             'n_estimators': [9,10,11,12,13],
             'max_features': ['auto', 'sqrt', 'log2'],
             'max_depth' : [4,5,6,7,8],
             'criterion' :['gini', 'entropy']
         }

In [58]: from sklearn.model_selection import GridSearchCV
         CV_rf1 = GridSearchCV(estimator=rf1, param_grid=param_grid, cv= 5)
         CV_rf1.fit(X_train, Y_train)

Out[58]: GridSearchCV(cv=5, estimator=RandomForestClassifier(),
                      param_grid={'criterion': ['gini', 'entropy'],
                                  'max_depth': [4, 5, 6, 7, 8],
                                  'max_features': ['auto', 'sqrt', 'log2'],
                                  'n_estimators': [9, 10, 11, 12, 13]})

In [59]: CV_rf1.best_params_

Out[59]: {'criterion': 'gini',
          'max_depth': 8,
          'max_features': 'sqrt',
          'n_estimators': 12}
```

Figure 18 :Precision,Recall,F1 score,Confusion matrix,Accuracy for Random Forest after applying GridSearch and Elbow method

Then Accuracy, Precision, Recall and F1 Score after applying above methods are as follows:

```
In [71]: rf2=RandomForestClassifier(n_estimators=12,max_features='auto',criterion='gini',max_depth= 8)
         rf2.fit(X_train,Y_train)
         r1_pred=rf2.predict(X_test)

In [72]: print('Classification Report:')
         print(classification_report(Y_test,r1_pred))
         print('\n')
         print('Confusion Matrix:')
         print(confusion_matrix(Y_test,r1_pred))

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.95      0.93     16942
           1       0.95      0.91      0.93     17130

    accuracy                           0.93     34072
   macro avg       0.93      0.93      0.93     34072
weighted avg       0.93      0.93      0.93     34072



Confusion Matrix:
[[16057   885]
 [ 1610 15520]]

In [73]: print(accuracy_score(Y_test,r1_pred))

0.9267727166001409
```

Figure 19 : Final Precision,Recall,F1 score,Confusion matrix,Accuracy for Random Forest

5. Naive Bayes

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

$$P(A/B) = (P(B/A)P(A)) / P(B)$$

Using Bayes theorem, we can find the probability of A happening, given that B has occurred. Here, B is the evidence and A is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

This model was trained using the tfidf vector after dimensionality reduction. Then the model was tested for accuracy on the test dataset and a Confusion Matrix was plotted along with test Accuracy, Precision, Recall and F1Score was repored.

Result :

The training and then prediction for the test set was performed 10 times using the same total dataset. For every iteration the dataset was split randomly into Training and Test set then and dimensionality reduction was carried out on the training and the test set. The model performance in terms of the Accuracy, Precision, Recall and F1 Score are as follows.

```
In [44]: from sklearn.naive_bayes import GaussianNB
         nb=GaussianNB()
         nb.fit(X_train,Y_train)
         n_pred=nb.predict(X_test)

In [45]: print('Classification Report:')
         print(classification_report(Y_test,n_pred))
         print('\n')
         print('Confusion Matrix:')
         print(confusion_matrix(Y_test,n_pred))

Classification Report:
              precision    recall  f1-score   support

           0       0.51      0.99      0.68     16942
           1       0.92      0.06      0.12     17130

    accuracy                           0.53     34072
   macro avg       0.72      0.53      0.40     34072
weighted avg       0.72      0.53      0.40     34072


Confusion Matrix:
[[16848    94]
 [16052  1078]]

In [47]: print(accuracy_score(Y_test,n_pred))

0.5261211552007513
```

Figure 20 :Precision,Recall,F1 score,Confusion matrix,Accuracy for Naïve Bayes

The accuracy scores of all the classification models we performed are as follows:

```
Accuracy scores of each model
LogisticRegression:  0.6839927212960789
Knn                :  0.9718243719182906
Decision tree      :  0.9751408781404085
Random forest      :  0.9774594975346326
GaussianNB         :  0.5261211552007513

Highest accuracy is for Random Forest
```

5.Conclusion

The aim of this project was to test different data representation methods and train various models using this data. We collected our real data from sources like H-1B Visa Petitions 2011-2016 dataset was aquired from Kaggle.The resulting transformed data was trained on the models mentioned in Section 4, and the following results were extrapolated from this experiment.

Algorithm	Precision	Recall	F-score
LogisticRegression	0.68	0.68	0.68
KNeighborsClassifier	0.97	0.97	0.97
DecisionTree	0.98	0.98	0.98
RandomForest	0.98	0.98	0.98
GaussianNB	0.72	0.53	0.40

Figure 21 :Precision,Recall,F1 score of all algorithms applied

The ratio of Certified to Non-Certified in the dataset were biased to resemble the real world scenario. Hence, in situation like these the model **accuracy** is really not the best metric to base our results. We generally test the performace of a model trained a biased dataset using **F1-Score**, which is nothing but the harmonic mean

of **precision** and **recall**. The above scores are given for prediction of the fake news article.

We observe that the **Random Forest** performed the best in accurately predicting the visa certified status using a TF-IDF data representation giving a F1-Score of 98.0%. A close second best performer was the Decision Tree model with an F1-Score of 98.0% with slight difference in Accuracy.

To know that which model is performing better,we performed the statistical test for 4 iterations in which we had compared between Random Forest and Decision Tree by using McNemar test and the results are as follows:

```
#mcnemar's statistical test b/w random forest and decision tree for 4 iterations
from mlxtend.evaluate import mcnemar_table
from mlxtend.evaluate import mcnemar
rfc_score=[]
dt_score=[]
for i in [9,15,42,98] :
    X_train1, X_test1, Y_train1, Y_test1 = train_test_split(x_res,y_res, test_size=0.2, random_state=i)
    rf.fit(X_train1,Y_train1)
    r_avg=rf.predict(X_test1)
    r_pred_avg=accuracy_score(Y_test1,r_avg)
    rfc_score.append(r_pred_avg)
    dtc.fit(X_train1, Y_train1)
    d_avg = dtc.predict(X_test1)
    d_pred_avg=accuracy_score(Y_test1,d_avg)
    dt_score.append(d_pred_avg)
    tb = mcnemar_table(y_target=Y_test,
                   y_model1=r_avg,
                   y_model2=d_avg)

    print(tb)
    chi2, p = mcnemar(ary=tb, exact=True)
    print('chi-squared:', chi2)
    print('p-value:', p)
    print()
```

```
[[16988   105]
 [   95 16884]]
chi-squared: 95
p-value: 0.5246223557223909

[[16932   103]
 [   98 16939]]
chi-squared: 98
p-value: 0.7779207003754158

[[33172    58]
 [  119   723]]
chi-squared: 58
p-value: 5.294914244749848e-06

[[16989    73]
 [   84 16926]]
chi-squared: 73
p-value: 0.42491156013385617
```

Figure 22 :Statistical test between RandomForest and Decision Tree using McNemar Test

As compared to the p-value's in different iterations,the values are mostly more than the significant threshold value which is 0.05,So we cannot reject the null hypothesis and assume that there is no significant difference between the two predictive models.

So now based on performing the Random forest and Decision tree for four iterations we got the accuracy results as follows:

```
In [64]: #this is the accuracy of random forest and decision tree on 4 iterations
         print("random forest: ",rfc_score)
         print("decition tree: ",dt_score)

         random forest:  [0.9782225874618455, 0.9782812866870157, 0.977077952571026, 0.9787802301009627]
         decition tree:  [0.9765790091570792, 0.9754930734914299, 0.9752876262033341, 0.9769312045081004]

In [61]: #this is the mean accuracy of random forest and decision tree
         print("mean accuracy of random forest is: ",np.mean(rfc_score))
         print("mean accuracy of decision tree is: ",np.mean(dt_score))

         mean accuracy of random forest is:  0.9780905142052125
         mean accuracy of decision tree is:  0.9760727283399859
```

Figure 23 :Mean accuracy of RandomForest and Decision Tree.

Here we observe that the **Random Forest** performed the best in accurately predicting the visa status with a mean accuracy of 0.97809 compared to Decision tree's mean accuracy of 0.97607.