ABSTRACT

This work focuses on the data science issue of using supervised machine learning for classification to forecast the outcome of previous immigrant visa applications. H-1B visas are non-immigrant visas that enable foreign nationals to serve in specific jobs in the United States. The trained models aid in the correlation of the decision to the application's properties. Because the H1-B visa classification is one of the most common, this approach can be used by both the individual and the employer in between submitting an application for a visa and receiving a final decision to be informed of the result before it occurs. The United States Citizenship and Immigration Service (USCIS) currently issues a quota of 85,000 visas per year, despite the fact that the number of applicants exceeds this figure. Therefore, H1B applicants and potential employers can use this model to determine the likelihood of approval both before and after the petition is filed . This study aims to utilize the metadata presented to determine the likelihood of obtaining a visa acceptance.

1. Introduction

1.1 Overview

Visa is the guide of approval on an authorization to travel that gives a grant to the holder to move in, leave or stay in the country for a predetermined time frame. Depending on the country to which one wishes to relocate, different types of foreigner visas, necessary structures, and methods for obtaining a worker visa exist. Moving to America is an essential and complex choice. The H1-B visa is a form of non-immigrant visa that requires US businesses to hire graduate-level employees in specialist occupations that include theoretical or professional experience in fields like information technology, banking, accounting, construction, engineering, mathematics, research, and medicine. This is one of the most commonly used visa types, and it is used by businesses that often need international talent.

1.2 Purpose

We hope to foresee the results of H-1B visa applications presented by a large number of highly qualified foreign nationals every year. This is a classification issue that is used to determine the application's expected case status. The applicant's characteristics are the contribution to our algorithm. The H-1B visa used in the United States requires foreign citizens to serve in jobs that necessitate advanced knowledge and a bachelor's degree in the field relevant specialization. This prediction algorithm, we conclude, may be a valuable resource for both prospective H-1B visa applicants and employers. We hope to foresee the results of a significant number of highly eligible foreign nationals sendH-1B visa applications per year. This project aims to forecast the case status of an employer's applicant to recruit non-immigrant employees under the H-1B visa program.

2. Literature Survey

The dataset we're looking at is called 'H1B Disclosure Dataset' on Kaggle, and it's a processed version of the original data. The data analysis helps us to identify the top occupations, states, employers, and industries that lead to the most H1B visa applications.

This chapter outlines the data set that was utilized for this research endeavor, as well as current debates concerning it. There's also some background on the algorithms that were employed in this study.

The main goal of this project is to forecast the status of visa applicants in terms of certification. The definition of classification will be used for this. When all of the attributes are given, the classifying model is trained on historical data from this application domain, and the likelihood of rows belonging to one class or status over another is calculated. We referred to the PREDICTION OF H1B VISA USING MACHINE LEARNING ALGORITHMS paper.

We came across two studies that are noteworthy. For both posts, we used the same Kaggle dataset. The first research analyzed and visualized the distribution of H-1B applications based on various input features such as place, wage, year, and work type. They only gave forecast accuracies for a selection of job categories rather than an average, despite using a K-means clustering and decision tree-based prediction methodology. Overall, this research provided us with important information on how our findings were disseminated. The case state was predicted using AdaBoost, Random Forest, Logistic Regression, and Naive Bayes in the second article. We applied some of their feature pre-processing techniques before feeding the functions into our own machine learning algorithms since they were clever.

Students at UC Berkeley are working on a project that seeks to forecast how long it would take to secure a work visa for a certain job title and firm. They employed KNN as the primary model to forecast the 'Quickest Certification Rate' for both vocations and organizations.

This project's aim is to forecast the case status of an employer's application to recruit nonimmigrant employees under the H-1B visa program. Employers can only recruit non-immigrant employees if their LCA petition has been accepted. The approved LCA petition is then included in the Petition for a Nonimmigrant Worker application for H-1B visa job authorizations.

3. Problem Identification and Objectives

3.1 Existing Problem

This project is about predicting the result of visa applications by using machine learning algorithms. Or this project, the dataset with H-1B visa applications has been taken and predicting is considered as supervised learning. It is a classification model has our final output will give us whether the visa is granted or not. The model is trained using some data and predicted with a classifier.

The H-1B visa, which was created under section 101(a)(15)(H) of the Immigration and Nationality Act, permits U.S. firms to temporarily recruit foreign employees in specialist occupations. Each fiscal year, the United States government issues around 85000 visas to those with a bachelor's degree or above. The master's category includes the remaining 20000 persons.

The project's main aim is to predict the acceptance of visa with more accuracy despite of uncertainty. It will be helpful for people to check whether their application gets accepted.

The dataset we're looking at is available on Kaggle under the name 'H1B Disclosure Dataset', which is processed dataset from the original data. From data analysis performed on this data allow us to find top contributing occupations, states, employers, and industries with the largest number of H1B applications for a visa. UC Berkeley students working on this project aims to estimate the time it would take to obtain a work visa for a given job title and employer. They made use of KNN as the main predictive model and the 'Fastest Certification Rate' in both occupations and organizations.

3.2 Proposed Method

The main objective of the project is to build a supervised classification model to classify the visa applications.

The visa petitions are classified into classes from the case status attribute of dataset. 80% of the data is used to train the 20% of the data and a model is to evaluate the data. The learning models' accuracy, precision, and recall see which model works best for predicting a visa petition's outcome.

The dataset we used for this problem is downloaded from Kaggle. It contained 10 features.

FULL_TIME_POSITION: Full time position = "Y" and Part time position = "N" are the two types of positions. "Full Time Position = 1; Part Time Position = 0" was changed to "Full Time Position = 1; Part Time Position = 0."

PREVAILING_WAGE: Prevailing pay is the standard wage paid to workers with similar credentials in the chosen field. We are currently utilizing this functionality.

CASE_SUBMITTED_YEAR: The year when the application was submitted.

CASE_SUBMITTED_MONTH: The month when the application was submitted.

CASE_SUBMITTED_DAY: The day was the application got submitted.

PW_SOURCE_YEAR: This is the year when the average wage paid to employees.

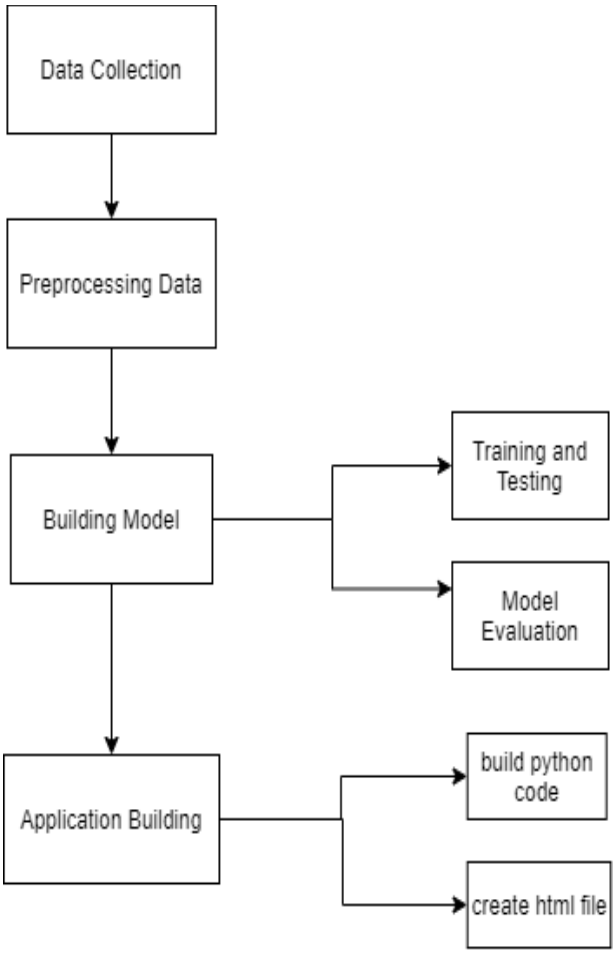
DECISION_DAY: The day when application got approved.

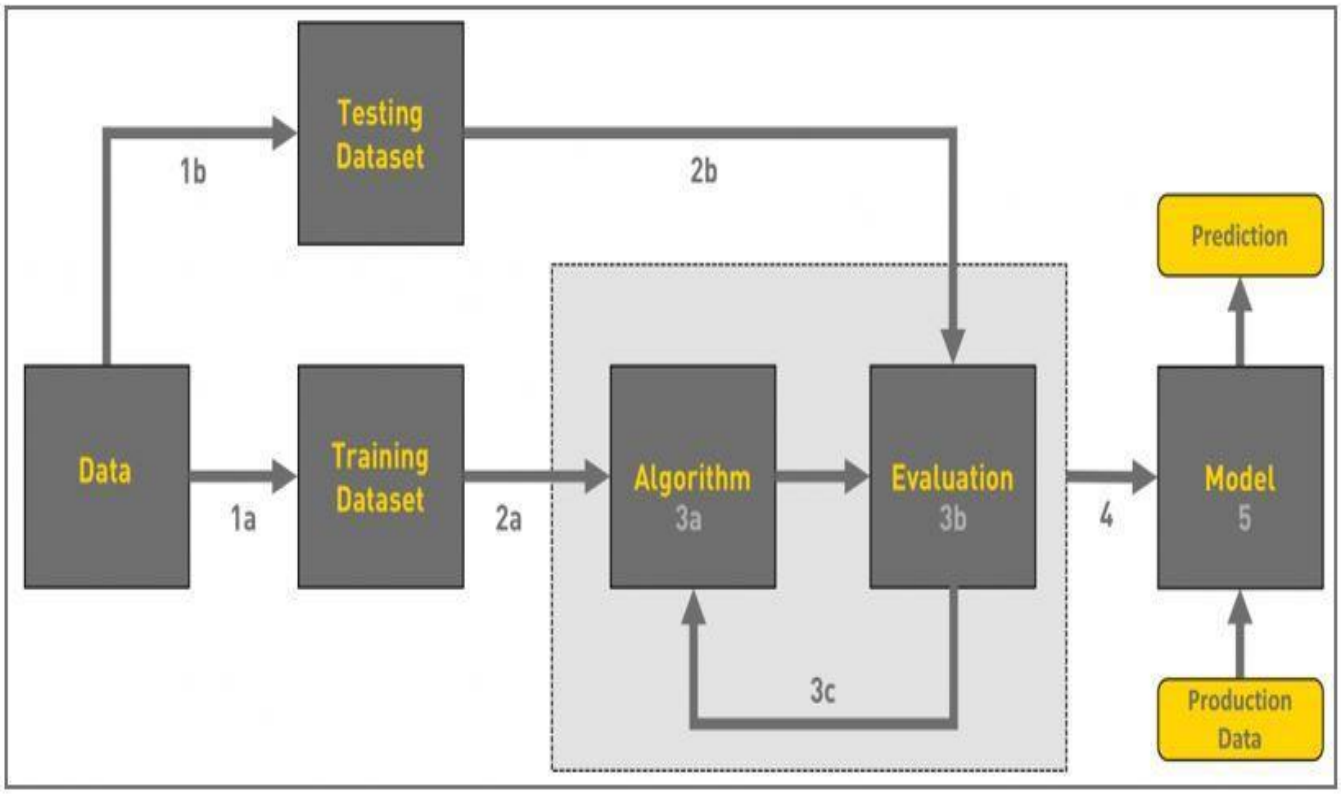
DECISION_YEAR: The year when application got approved.

DECISION_MONTH: The month when application got approved.

CASE_STATUS: This feature give us a complete prediction about either the application has been approved or denied.

4. System Design




Testing Dataset
1b
2b
Prediction
Data
1a
Training Dataset
2a
Algorithm
3a
Evaluation
3b
4
Model
5
3c
Production Data

5. Methodology

Machine Learning:

Machine Learning is a computer science and artificial intelligence technology that allows algorithms to learn and develop without having to be directly programmed. It employs a series of algorithms that can navigate data and learn on their own. Selecting the best algorithm could seem to be a challenge. There are several different types of supervised and unsupervised machine learning algorithms, each with its own learning approach. There is no one-size-fits-all method or methodology. Experimentation is insufficient when it comes to using the best algorithm. Even the most seasoned data scientists are unable to predict whether an algorithm would perform without first giving it a try. The size and type of data you're dealing with play a role in algorithm choosing. We decided to use the Naive Bayes technique because our dataset is huge and we have a classification challenge. It is possible to use either the Naive Bayes or the Support Vector Machine method. We need to follow the following steps:

1. Data Pre-processing
2. Training the model
3. Testing our prepared model
4. Model Evaluation
5. Application Building

Data Pre-processing:

Data preprocessing is the stage in any Machine's Learning process when the data is transformed, or encoded, to make it easier for the machine to parse. To put it another way, the algorithm can now quickly comprehend the findings' features. Data preprocessing is the process of preparing raw data to make it suitable for a machine learning algorithm. It is the first and most crucial phase in the building of a machine learning model.

While working on a machine learning project, we don't always come across clean and prepared data. Furthermore, prior to completing any data-related activity, the data must be cleaned and formatted. As a consequence, we make use of the data mining technique.

Important Steps:

1. Importing Libraries:

We need to import certain predefined Python libraries in order to do data preprocessing with Python. This libraries are used for a variety of tasks. For data preprocessing, we can use the following three libraries:

a. **Numpy:** The Numpy Python library is used to implement some kind of techniques known. It is the most important Python package for scientific calculations. Big, multidimensional arrays and matrices could also be applied.

b. **Matplotlib:** Its a 2D plotting library of python that requires the import of a sub-library called pyplot. This library is used in the code to map some kind of graph.

c. **Pandas:** It is one of the most well-known Python libraries for importing and handling datasets. It's a data manipulation and research library that's free to use.

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
```

2. Train Test Split:

One of the most critical phase of Machine Learning is the Train Test Split. It's important because the model must be tested before it can be deployed. And the assessment must be performed on unseen data, for all incoming data is unseen until it is deployed
. The key goal of the train test break is to turn the original data set into a new data set.
The train test split's key concept is to split the initial data collection into two parts:

a) train : It contains training data and training labels
b) test : It contains testing data and testing labels.

3. Handling Missing data:

The next step in the data preprocessing process is to deal with missing data in the datasets. If any of the data in our dataset is incomplete, it may pose a significant problem for our machine learning model. As a result, handling missing values in the dataset is needed.
There are primarily two

a. **approaches to dealing with lost data:** We can delete the missing values manually.

b. **By removing a certain row:** The first method is widely used to deal with null values. We simply remove the null values from the same row or column in this manner. However, this method is inefficient, and deleting data can result in the loss of knowledge, resulting in an inaccurate performance.

c. **Using the mean** as a guide: We will measure the mean of the column or row that includes any missing values and position it in the place of the missing value in this manner. This strategy is beneficial for features that include numeric details.

4. Encoding categorical data:

In machine learning models, both the input and output variables must be numeric. This means that you'll need to transform categorical data to numbers before fitting and assessing a standard. Encoding is a required pre-processing step when working with categorical data for machine learning algorithms.

a. **Label encoding** is the process of converting labels to numeric format so that machines can read them. Machine learning algorithms will then make more informed judgments about how to use these markings. It is an important pre-processing stage for the structured dataset in supervised learning.
b. **One hot encoding** makes for a more expressive representation of categorical data. Many machine learning algorithms are unable to deal explicitly with categorical results. The divisions must be numerically translated. This is expected for both categorical input and output variables.

Experimental Analysis:

Naive Bayes:

It's a classification technique based on Bayes' Theorem and the predictor independence principle. In simple terms, a Naive Bayes classifier assumes that the presence of one function in a class has no bearing on the presence of any other feature. The Naive Bayes model is simple to build and is very effective when dealing with huge data sets. Because of its simplicity, Naive Bayes is thought to outperform even the most complex categorization systems. By considering their highest likelihood estimates, it determines P(x/y=0), P(x/y=1), and P(y) in the mutual probability of the data. When creating a forecast, it evaluates both P(y=1) and P(y=0) on the Bayes law and compares the two.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

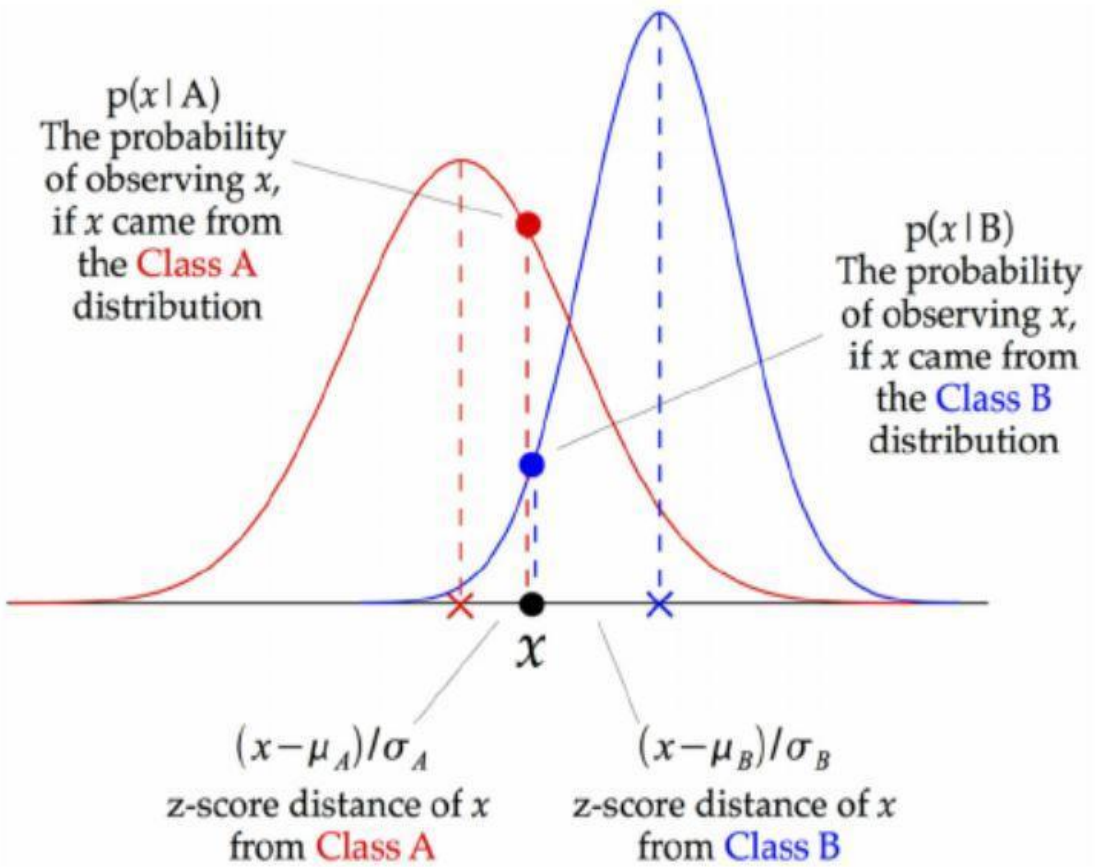
Gaussian Naive Bayes:

One typical assumption when working with continuous data is that the continuous values associated with each class would follow a regular (or Gaussian) distribution.

The probability of the traits is estimated to be-

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Gaussian Naive Bayes accepts continuous valued features and models them all as Gaussian (normal) distributions. To build a basic model, assume the data is represented by a Gaussian distribution with no covariance (independent dimensions) between the dimensions. This model can be fitted by simply calculating the mean and standard deviation of the points within each mark, which is all that is required to describe a distribution of this kind.



A Gaussian Naive Bayes (GNB) classifier is depicted in the diagram above. The z-score interval between each data point and each class mean, i.e. the distance from the class mean separated by the standard deviation of that class, is determined at each data point. As a result, we can see that the Gaussian Naive Bayes technique is somewhat different and can be used effectively.

Decision Tree Classifier:

A basic and commonly used classification method is the Decision Tree Classifier. It implements a basic principle to solve the classification problem.

A collection of well-crafted questions about the test record's properties are asked by the Decision Tree Classifier. Every time it receives a response, it asks a follow-up question before a consensus about the record's class mark is reached.

THEORITICAL ANALYSIS:

BLOCK DIAGRAM

```
In [94]: import matplotlib.pyplot as plt
         %matplotlib inline

In [95]: plt.plot(dataset['CASE_STATUS'],dataset['FULL_TIME_POSITION'],'g')

Out[95]: [<matplotlib.lines.Line2D at 0x24e397705e0>]
```

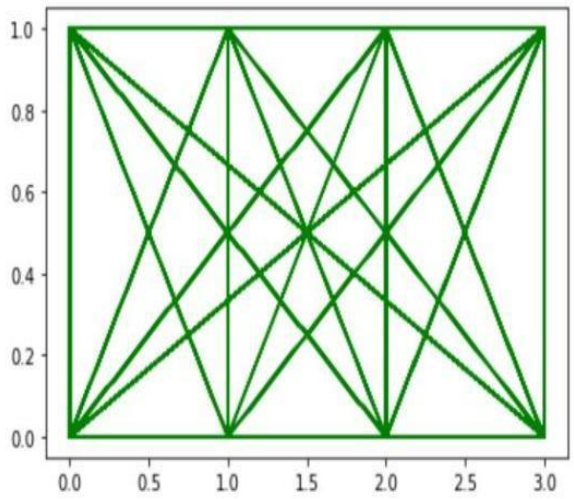


Figure 4: Matplot between "CASE_STATUS" and "FULL_TIME_POSITION"

HARDWARE/SOFTWARE DESIGNING

During the data pre-processing, we used Label Encoding only on 2 features, "FULL_TIME_POSITION", and "CASE_STATUS". And then simply spitted the data into train and test set in 80:20 ratio.

6. Implementation

Working of Naive Bayes:

Step 1: Construct a frequency table from the data collection.
Step 2: Find the odds and build a Likelihood graph
Step 3: Next, determine the posterior likelihood for each class using the Naive Bayesian method. The consequence of estimation is the class with the greatest posterior likelihood.

Model Building:

Training and Testing:

A training set is used to create a model, while a test or validation set is used to ensure accuracy. As a consequence, we use training data to fit the model and testing data to validate it. The training set is used to build the model, whereas the testing set is used to test it. A model is built via training, and research is done to guarantee that the model is accurate. Our dataset was divided into two parts: a testing set and a deep learning data preprocessing test set. This is an important step in data preparation since it improves the precision of our machine learning system.

Assume we've trained our machine learning model with one dataset and then tested it with a new dataset. Then our model would have a difficulties understanding the relationships between the models.

If we train our model well and its training accuracy is good, however we give it a new dataset, the model's efficiency will drop. As a result, we still strive to build a machine learning model that works well for both the training and test datasets.

```
In [99]: 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)

In [100]: x_train.shape

Out[100]: (422507, 9)

In [101]: x_test.shape

Out[101]: (105627, 9)

In [102]: y_train.shape

Out[102]: (422507, 1)

In [103]: y_test.shape

Out[103]: (105627, 1)
```

StandardScaler:

The mean is removed and each feature/variable is scaled to unit variance using StandardScaler. This process is carried out in a feature-by-feature manner. Since StandardScaler requires the calculation of the empirical mean and standard deviation of each f
unction, it may be affected by outliers (if they occur in the dataset).

StandardScaler is a scaler. After subtracting the mean, scale to unit variance, StandardScaler standardizes a function. Divide all of the values by the standard deviation to get unit variance. The StandardScaler function produces a distribution with a standard deviation of one.

```
In [104]: from sklearn.preprocessing import StandardScaler
          sc=StandardScaler()
          x_train=sc.fit_transform(x_train)
          x_test=sc.fit_transform(x_test)
```

```
In [105]: x_train
```

```
Out[105]: array([[ 0.33774548,  0.17368846,  0.47094157, ...,  0.15334608,
                  -0.22726029,  0.21278833],
                 [ 0.58293419, -0.13869274,  0.47094157, ...,  0.15334608,
                  -0.49766614, -0.01436499],
                 [ 1.80887774, -1.07583633,  0.47094157, ...,  0.15334608,
                   0.00673936, -0.01436499],
                 ...,
                 [ 0.46033984, -1.07583633,  0.47094157, ...,  0.15334608,
                   0.36358258, -0.01436499],
                 [-0.64300936, -0.13869274,  0.47094157, ...,  0.15334608,
                   1.22405004,  0.21278833],
                 [-0.2752263 , -0.45107394,  0.47094157, ..., -6.52119714,
                  -2.23959949, -0.01436499]])
```

Model evaluation:

The aim of model evaluation is to predict a model's generalization accuracy on future data. Methods for ensuring the efficiency of a model.
The evaluation of a model is a crucial phase in its development. It assists in the selection of the best model to represent our data as well as the prediction of how well that model will perform in the future. To avoid overfitting, both methods utilize a test range to assess model output.

Building a Decision Tree:

In a decision tree classifier, constructing an optimum decision tree is a key challenge.
A specified set of attributes can be used to build a variety of decision trees. While some of the trees are more reliable than others, despite the exponential scale of the search space, finding the best tree is computationally difficult.
However, a number of strong algorithms have been devised to quickly generate a somewhat accurate, suboptimal decision tree.
These algorithms generally form a decision tree by making a series of locally optimum judgments on which characteristic to utilize for data partitioning using a greedy strategy.
Hunt's method, ID3, C4.5, CART, and SPRINT are examples of greedy decision tree induction techniques.

ID3 Algorithm for Building a Decision Tree

ID3 stands for Iterative Dichotomiser 3, and it gets its name from the fact that the algorithm separates features into two or three groups at each step
iteratively, which means repeatedly.
ID3 is a top-down greedy solution to building a decision tree that was invented by Ross Quinlan.

In simple terms, the top-down technique entails building the tree from the top down, whereas the greedy method entails selecting the optimal function available at the time to produce a node at each iteration.
ID3 is usually used to solve classification problems with only a few items.

ID3 Metrics

During the development of a Decision tree, the ID3 algorithm selects the best feature at each stage.
ID3 uses Information Gain to determine how well a particular characteristic separates or categorizes the target groups by estimating the reduction in entropy. The feature with the largest Information Gain is picked as the best. Entropy is a measure of disorder in the goal function of a dataset, and the Entropy of a dataset is the measure of disorder in the dataset's goal function.
Entropy is 0 if all values in the target column are homogenous (identical), and 1 if all binary classification groups in the target column have the same number of values (there are two classes in target column).

ID3 Steps:

Calculate the Information Gain for each characteristic.
Divide dataset S into subsets using the function with the maximum Information Gain, given that not all rows belong to the same class.
Create a decision tree node using the function that provides the greatest information.

If all rows belong to the same class, make the current node a leaf node with the class as its name.

Continue until the decision tree has all leaf nodes or we run out of features.

Application Building:

Building a python code:

We will utilize a flask application to develop an application since we must first generate python code. Flask is a lightweight and compact Python web interface that provides helpful tools and capabilities for developing online applications in Python. Because a web application can be readily created with only a single Python file, it gives developers greater freedom and makes the platform more accessible to novice developers.

Backend:

```
In [60]: import numpy as np
         import pandas as pd

In [61]: dataset=pd.read_csv(r'C:\Users\hp\Desktop\Batch-10 Visa Approval Prediction\dataset\1. Master H1B Dataset.csv',encoding='latin1')

         C:\Users\hp\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3071: DtypeWarning: Columns (25) have mixed types.Spec
         ify dtype option on import or set low_memory=False.
           has_raised = await self.run_ast_nodes(code_ast.body, cell_name,

In [62]: dataset.head(5)

Out[62]:
```

	CASE_SUBMITTED_DAY	CASE_SUBMITTED_MONTH	CASE_SUBMITTED_YEAR	DECISION_DAY	DECISION_MONTH	DECISION_YEAR	VISA_CLASS	EMPLOY
0	24	2	2016	1	10	2016	H1B	PROD
1	4	3	2016	1	10	2016	H1B	DFS
2	10	3	2016	1	10	2016	H1B	TECH
3	28	9	2016	1	10	2016	H1B	INFO
4	22	2	2015	2	10	2016	H1B	CORP

5 rows × 27 columns

```
In [63]: dataset.drop('PW_UNIT_OF_PAY',axis=1,inplace=True)

In [64]: dataset.drop('WORKSITE_POSTAL_CODE',axis=1,inplace=True)

In [65]: dataset.drop('NAICS_CODE',axis=1,inplace=True)

In [66]: dataset.drop('TOTAL_WORKERS',axis=1,inplace=True)

In [67]: dataset.drop('WAGE_UNIT_OF_PAY',axis=1,inplace=True)

In [68]: dataset.drop('WAGE_RATE_OF_PAY_FROM',axis=1,inplace=True)

In [69]: dataset.drop('WAGE_RATE_OF_PAY_TO',axis=1,inplace=True)

In [70]: dataset.drop('H-1B_DEPENDENT',axis=1,inplace=True)

In [71]: dataset.drop('PW_SOURCE',axis=1,inplace=True)

In [72]: dataset.drop('WILLFUL_VIOLATOR',axis=1,inplace=True)
```

In [73]:
```
dataset.drop('VISA_CLASS',axis=1,inplace=True)
```

In [74]:
```
dataset.drop('EMPLOYER_STATE',axis=1,inplace=True)
```

In [75]:
```
dataset.drop('EMPLOYER_COUNTRY',axis=1,inplace=True)
```

In [76]:
```
dataset.drop('EMPLOYER_NAME',axis=1,inplace=True)
```

In [77]:
```
dataset.drop('SOC_NAME',axis=1,inplace=True)
```

In [78]:
```
dataset.drop('PW_SOURCE_OTHER',axis=1,inplace=True)
```

In [79]:
```
dataset.drop('WORKSITE_STATE',axis=1,inplace=True)
```

In [80]:
```
dataset.isnull().any()
```

Out[80]:
```
CASE_SUBMITTED_DAY      False
CASE_SUBMITTED_MONTH    False
CASE_SUBMITTED_YEAR     False
DECISION_DAY            False
DECISION_MONTH          False
DECISION_YEAR           False
FULL_TIME_POSITION       True
PREVAILING_WAGE         False
PW_SOURCE_YEAR           True
CASE_STATUS             False
dtype: bool
```

In [81]:
```
dataset.head(2)
```

Out[81]:

	CASE_SUBMITTED_DAY	CASE_SUBMITTED_MONTH	CASE_SUBMITTED_YEAR	DECISION_DAY	DECISION_MONTH	DECISION_YEAR	FULL_TIME_POSITION
0	24	2	2016	1	10	2016	Y
1	4	3	2016	1	10	2016	Y

In [82]:
```
dataset['FULL_TIME_POSITION'].fillna(dataset['FULL_TIME_POSITION'].mode()[0],inplace=True)
dataset['PW_SOURCE_YEAR'].fillna(dataset['PW_SOURCE_YEAR'].mode()[0],inplace=True)
```

In [83]:
```
dataset.isnull().any()
```

Out[83]:
```
CASE_SUBMITTED_DAY      False
CASE_SUBMITTED_MONTH    False
CASE_SUBMITTED_YEAR     False
DECISION_DAY            False
DECISION_MONTH          False
DECISION_YEAR           False
FULL_TIME_POSITION      False
PREVAILING_WAGE         False
PW_SOURCE_YEAR          False
CASE_STATUS             False
dtype: bool
```

In [84]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dataset['FULL_TIME_POSITION']= le.fit_transform(dataset['FULL_TIME_POSITION'])
dataset['CASE_STATUS']= le.fit_transform(dataset['CASE_STATUS'])
```

In [85]:
```
dataset['FULL_TIME_POSITION'].unique()
```

Out[85]:
```
array([1, 0])
```

In [86]:
```
dataset['CASE_STATUS'].unique()
```

Out[86]:
```
array([1, 3, 0, 2])
```

In [87]:
```
dataset.head(5)
```

Out[87]:

	CASE_SUBMITTED_DAY	CASE_SUBMITTED_MONTH	CASE_SUBMITTED_YEAR	DECISION_DAY	DECISION_MONTH	DECISION_YEAR	FULL_TIME_PO
0	24	2	2016	1	10	2016	
1	4	3	2016	1	10	2016	
2	10	3	2016	1	10	2016	
3	28	9	2016	1	10	2016	
4	22	2	2015	2	10	2016	

In [88]:
```
x=dataset.iloc[:,0:9].values
y=dataset.iloc[:,9:10].values
```

In [89]:
```
x.shape
```

Out[89]:
```
(528134, 9)
```

In [90]:
```
y.shape
```

Out[90]:
```
(528134, 1)
```

In [91]:
```
x
```

Out[91]:
```
array([[2.40000e+01, 2.00000e+00, 2.01600e+03, ..., 1.00000e+00,
        5.91970e+04, 2.01500e+03],
       [4.00000e+00, 3.00000e+00, 2.01600e+03, ..., 1.00000e+00,
        4.98000e+04, 2.01500e+03],
       [1.00000e+01, 3.00000e+00, 2.01600e+03, ..., 1.00000e+00,
        7.65020e+04, 2.01500e+03],
       ...,
       [3.00000e+01, 6.00000e+00, 2.01700e+03, ..., 1.00000e+00,
        7.94980e+04, 2.01600e+03],
       [3.00000e+01, 6.00000e+00, 2.01700e+03, ..., 1.00000e+00,
        1.18352e+05, 2.01600e+03],
       [3.00000e+01, 6.00000e+00, 2.01700e+03, ..., 1.00000e+00,
        4.91300e+04, 2.01600e+03]])
```

In [92]:
```
y
```

Out[92]:
```
array([[1],
       [1],
       [1],
       ...,
       [3],
       [ ]
```

Application Building:

Flask is a simple Python platform for web applications that manages URL routing and page rendering.

Since it doesn't actually have functionality including form validation, database isolation, authorization, and so on, Flask is referred to as a micro system. Instead, Flask plugins, which are Python bundles, have these capabilities. The extensions work hand in hand with Flask and tend to be an integral part of the framework. The coding allows us to support a simple web application as if it were a website.
The Flask Framework searches for HTML files in the templates folder. Build a templates folder and place all of your HTML files in
it. The flask framework's render template() function was imported. render template() searches the templates folder for a template (HTML file). The template you requested will then be rendered.

The return value has been changed to render template ("fin.html"). This allows us to see our HTML code.

Now go to your localhost and have a look at the changes: http://localhost:5000/ is a URL that you may use.

Frontend:

```
# -*- coding: utf-8 -*-
"""
Created on Mon April 26 17:03:46 2021

@author: Pinka
"""

from flask import Flask,render_template,request
import pickle
import numpy as np
model=pickle.load(open(r"C:/Users/hp/Desktop/Batch-10 Visa Approval Prediction/flask/dtc2.pkl",'rb'))



app=Flask(__name__)
@app.route('/')
def home():
    return render_template("fin.html")
@app.route('/login',methods=['POST'])
def login():
    file=request.form['ap']
    if(file=="Y"):
        s1=1
    if(file=="N"):
        s1=0




    file1=request.form['ag']
    file2=request.form['bp']
    file3=request.form['subday']
    file4=request.form['submonth']
    file5=request.form['subyear']
    file6=request.form['decday']
    file7=request.form['decyear']
    file8=request.form['decmonth']

    total=[[int(file3),int(file4),int(file5),int(file6),int(file8),int(file7),s1,int(file1),int(file2)]]
    y_pred=[]
    y_pred = [[3]]

    y_pred=model.predict(np.array(total))
```

```
    if (y_pred==[[0]]):
        return render_template("fin.html",showcase=" Your VISA is DENIED  ")

    if(y_pred==[[1]]):
         return render_template("fin.html",showcase="Your VISA is CERTIFIED WITHDRAWN")
    if(y_pred==[[2]]):
         return render_template("fin.html",showcase="Your VISA is CERTIFIED")
    else:
         return render_template("fin.html",showcase="Your VISA is WITHDRAWN")

if(__name__)=='__main__':
     app.run(debug=False)
```

Creating an html code:

Without any HTML, the application merely shows a basic message. Because online apps primarily employ HTML to present information to visitors, we must now focus on including HTML files in our app that can be seen in a web browser.

We need to write the html code for our application so that can predict the results using the inputs provided within short duration of time.

```
<html>
<head>
</head>
<style>
body {
  background-image: url("https://st4.depositphotos.com/1764573/20372/i/1600/depositphotos_203720964-stock-photo-blur-passport-white-background-concept.jpg");
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: cover;
  color: black;
  font-weight: bolder;
  font-size: 20;
}
</style>
```

```
<body>
<center>
<h1><b style="color:black">VISA Approval Prediction</b></h1>
<form action = "/login" method = "post">
<label for = "position">Is it a full time position</label>
<select name ="ap">
<option value ="Y">Y</option>
<option value ="N">N</option>
<select/>
```

```
<br>
<p> Enter prevailing wage</p>
<p><input type = "text" name = "ag"/></p>
<p> Enter prevailing source year</p>
<p><input type = "text" name = "bp"/></p>
```

```
<p> Enter case submitted dat</p>
<p><input type = "text" name = "subday"/></p>
<p> Enter case submitted month</p>
<p><input type = "text" name = "submonth"/></p>
<p> Enter case submitted year</p>
<p><input type = "text" name = "subyear"/></p>
<p> Enter decision day</p>
<p><input type = "text" name = "decday"/></p>
<p> Enter decision month</p>
<p><input type = "text" name = "decmonth"/></p>
<p> Enter decision year</p>
<p><input type = "text" name = "decyear"/></p>



<br>

<p> <input type = "submit" value = "SUMBIT"/></p>
</center>
</form>
<center>
<b> {{showcase}} </b>
</center>
</body>
</html>
```

7. System Testing

After splitting the data in ratio 80:20, we applied Naive Bayes on the data to predict the outcome. While using the naive Bayes algorithm we used the default version of it without changing any hyperparameter tuning and received an Accuracy of 0.842 i.e. 84.2%

```
In [50]: from sklearn.naive_bayes import GaussianNB
         naive = GaussianNB()

In [51]: naive.fit(x_train,y_train)

         C:\Users\hp\anaconda3\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-v
         n a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
           return f(**kwargs)

Out[51]: GaussianNB()

In [52]: y_pred=naive.predict(x_test)

In [53]: from sklearn.metrics import accuracy_score
         acc=accuracy_score(y_test,y_pred)

In [54]: acc

Out[54]: 0.8420763630511138
```

Since the Accuracy obtained was not adequate so, we then applied Decision Tree on the data to predict the outcome. While using the Decision Tree algorithm we used the ID3 Algorithm version of it without changing anything and received an Accuracy of 0.973 i.e. 97.3%

```
In [47]: from sklearn.tree import DecisionTreeClassifier
         dtc = DecisionTreeClassifier (random_state = 0)
         dtc.fit(x_train,y_train)

Out[47]: DecisionTreeClassifier(random_state=0)

In [48]: y_pred  = dtc.predict(x_test)

In [49]: from sklearn.metrics import accuracy_score
         acc=accuracy_score(y_test,y_pred)

In [50]: acc

Out[50]: 0.9734253552595454
```

Since we got a better accuracy by using/implementing Decision Tree Algorithm so we are going forward and showing the project using Decision Tree Algorithm only.

8. Output


VISA Approval Prediction
Is it a full time position Y
Enter prevailing wage
67870
Enter prevailing source year
2016
Enter case submitted dat
27
Enter case submitted month
9
Enter case submitted year
2016
Enter decision day
3
Enter decision month
10
Enter decision year
2016
SUMBIT
Your VISA is CERTIFIED


VISA Approval Prediction
Is it a full time position Y
Enter prevailing wage
90376
Enter prevailing source year
2016
Enter case submitted dat
28
Enter case submitted month
9
Enter case submitted year
2016
Enter decision day
1
Enter decision month
10
Enter decision year
2016
SUMBIT
Your VISA is WITHDRAWN


VISA Approval Prediction
Is it a full time position Y
Enter prevailing wage
59197
Enter prevailing source year
2015
Enter case submitted dat
24
Enter case submitted month
2
Enter case submitted year
2016
Enter decision day
1
Enter decision month
10
Enter decision year
2016
SUMBIT
Your VISA is CERTIFIED WITHDRAWN


VISA Approval Prediction
Is it a full time position Y
Enter prevailing wage
76219
Enter prevailing source year
2016
Enter case submitted dat
29
Enter case submitted month
6
Enter case submitted year
2017
Enter decision day
30
Enter decision month
6
Enter decision year
2017
SUMBIT
Your VISA is DENIED

Conclusions

The time it takes to filter out and shortlist the final applicants is considerable. So, if we could predict the outcome, it would be extremely beneficial for applicants to assess and prepare their future accordingly without wasting valuable time, especially during their prime working years. Companies would benefit as well, as they would be able to make effective strategic plans with the aid of these forecasts.

It was possible to predict this outcome with a reasonable degree of precision by taking into account different parameters and developing appropriate machine learning models.

The initial purpose of the project was to forecast the outcome of H-1B visa petitions, which are filed by a huge number of foreign citizens. We can forecast specific outcomes and assist millions of people by utilizing the benefits of cutting-edge technology such as machine learning.