# Predictive Modeling For H1b Visa Approval Using IBM Watson

## 1. INTRODUCTION

### 1.1 Overview

H1-B Visa is one type of non-immigrant temporary visa granted by USCIS (United States Citizenship and Immigration Service) for the foreign nationals. These petitions are filed by the employers for their employees. This visa is also filed by international students after they get admissions into universities. Since the number of applicants is very large than the number of selections and as the selection process is claimed to be as lottery there is no insight of how the attributes have influence over the outcome. So, we believe that a predictive model generated using all the past data can be a useful resource to predict the outcome for the applicants and the sponsors.

In the Guided Project, our goal is to predict the outcome of H-1B visa applications that are filed by many professional foreign nationals every year. Here, we framed the problem as a classification problem and applied it in order to output a predicted case status of the application. The input to our algorithm is the attributes of the applicant. This paper, for predicting the outcome of the approval of H-1B visa, the 2011-2016 H-1B dataset is used which contains more than 3 million petitions from the datasets. Histograms were utilized in order to eliminate the outliers. One-hot encoding was used to convert data into appropriate format. Finally, Random Forest algorithm was used to train the data and predict the final outcome, whether the petition is accepted or not.

### 1.2 Purpose

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 sponsoring them. In order to predict the case status of the applicants, we will be feeding the model with the dataset which contains the required fields by which the machine can classify the case status as certified or denied

## 2.LITERATURE SURVEY

### 2.1 Existing  Problem

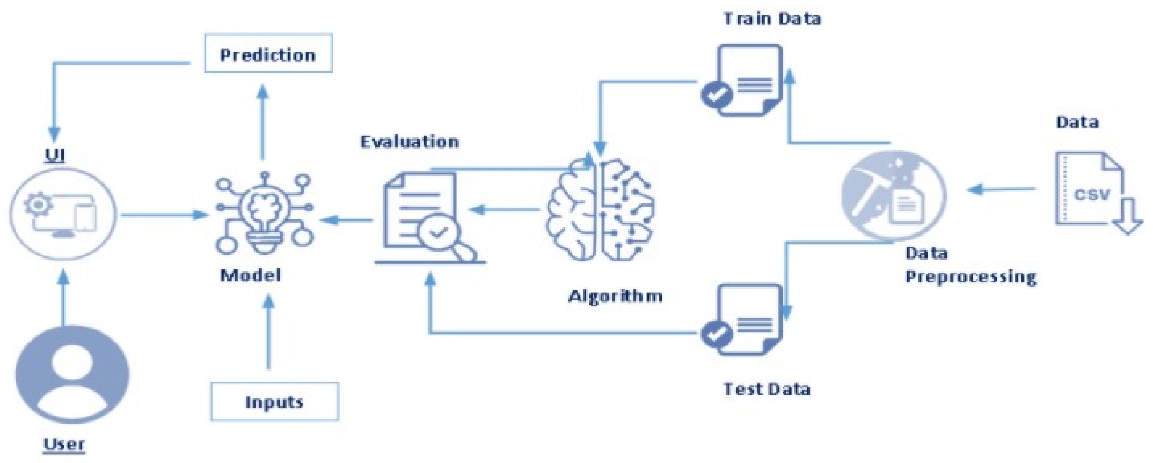
The first and greatest drawback of the H-1B visa is the fact that there is an annual limit on how many petitions are approved each year. While other visas also have a limit, they are not as easy to obtain and so that limit is rarely reached. The H-1B, on the other hand, annually receives almost three times the amounts of petitions than is allotted.

### 2.2 Proposed Solution

The goal is to explore the petitions filed and their outcomes for the past six years i.e., from 2011 to 2016, and to find a pattern to predict the outcome by using a predictive model developed using Machine Learning techniques. In order to predict the case status of the applicants, we will be feeding the model with the dataset which contains the required fields by which the machine can predict the certification status of the visa applications.

## 3.THEORITICAL ANALYSIS

### 3.1 Block Diagram



## 3.2 Hardware / Software designing

- IBM Watson Studio - IBM Watson Studio helps data scientists and analysts prepare data and build models at scale across any cloud.
- IBM Watson Machine Learning - IBM Watson Machine Learning helps data scientists and developers accelerate AI and machine learning deployment.
- IBM Cloud Object Storage - IBM Cloud Object Storage makes it possible to store practically limitless amounts of data, simply and cost effectively.
- Machine Learning Services - Machine learning as service is an umbrella term for collection of various cloud-based platforms that use machine learning tools to provide solutions that can help ML teams with: out-of-the box predictive analysis for various use cases, data pre-processing, model training and tuning.

## 4. EXPERIMENTAL INVESTIGATIONS

Dataset is downloaded from the Kaggle which has 9 features and 1 feature containing the class label. The total number of records available for us is more than 3 million points. The features provide the following information about our samples.

- EMPLOYER_NAME: Name of employer submitting application.
- SOC_NAME: Occupational name associated with the SOC CODE which is an occupational code associated with the job being requested for temporary labour condition, as classified by the Standard Occupational Classification (SOC) System.
- JOB_TITLE: Title of the job
- FULL_TIME_POSTION: There are 2 categories for this feature: Y= Full time position and N = Part Time Position
- PREVAILING_WAGE: the average wage paid to employees with similar qualifications in the intended area of employment.
- YEAR: The year of filing the petition

- WORKSITE: City and state of the applicant’s job.
- lon & lat: Exact geographical location of the worksite.

The 1 label in the dataset is divided into 7 classes:

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

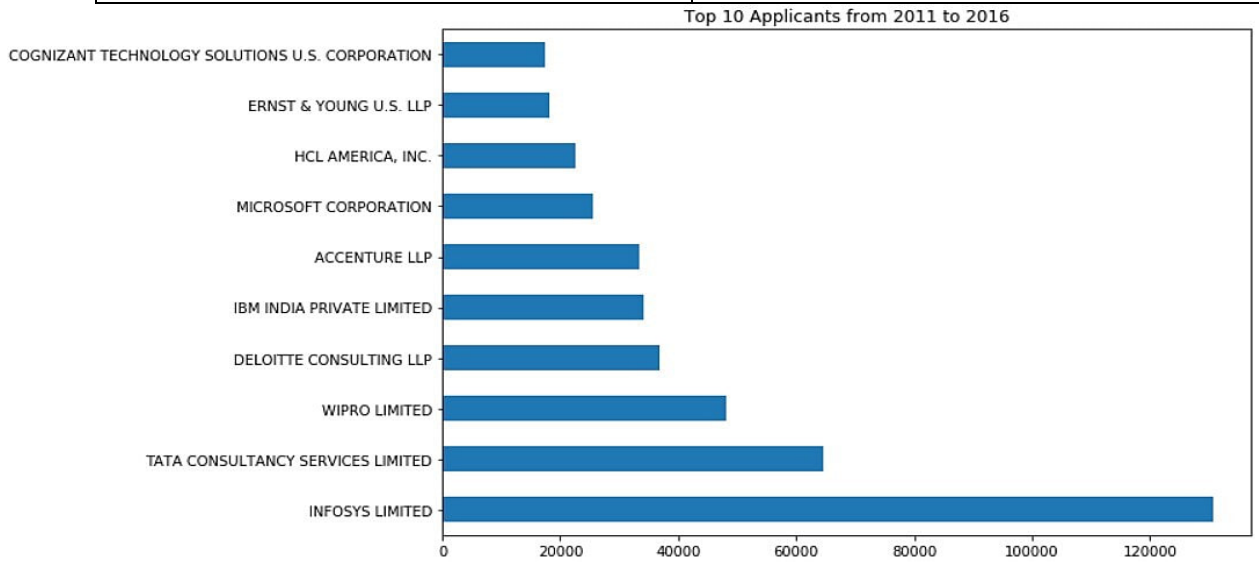
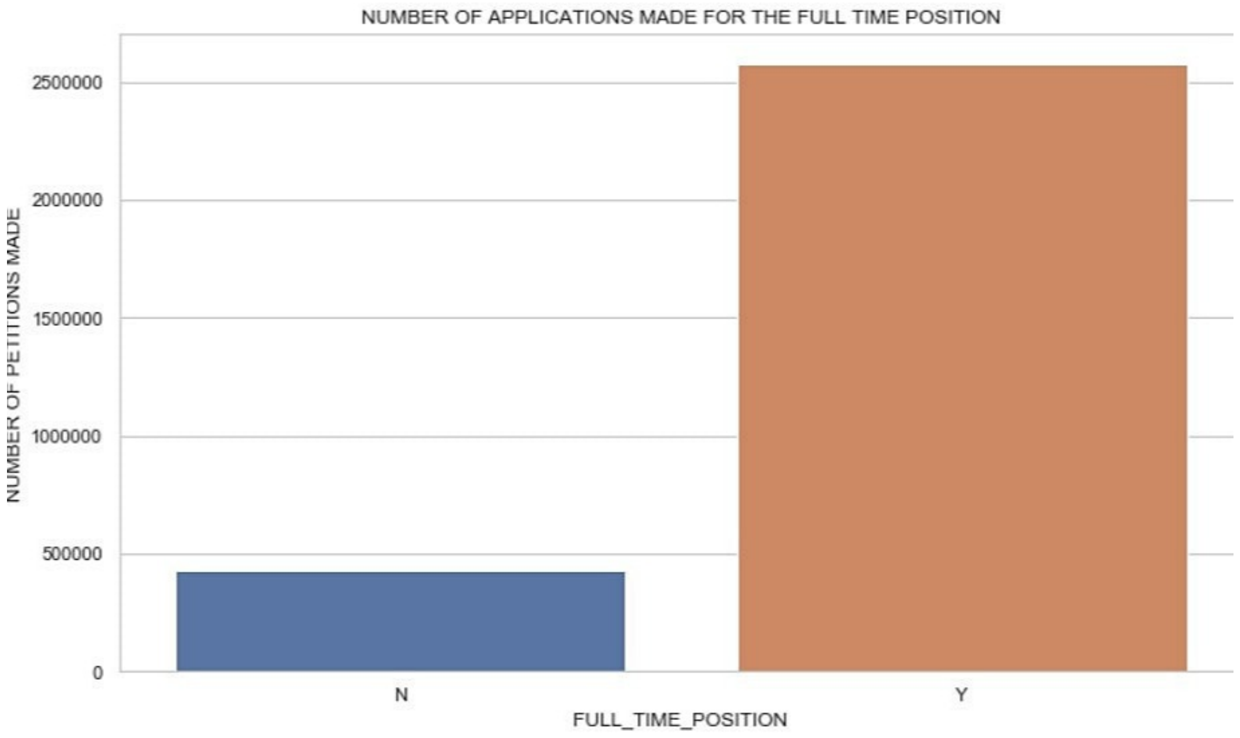


Fig here shows petitions filed per year has highly increased from 2011 to 2016 has approximately doubled its number



As can be seen from the Fig. the dataset is highly imbalanced, we apply some pre-processing techniques to create a dataset which has more relevant data to generate a model leaving all the noise in the data. We performed exploratory data analysis to get some facts which the data provides us and basing on them. we considered the relevance of relationship among the features and accordingly discarded few features and created some to remove the redundant information. We observed that features 'LON' and 'LAT' have missing values nearly 100000 points hence we remove both the features entirely. Also, we transformed few features into new features. The features of our final dataset after transformation is: CASE_STATUS, FULL_TIME_POSITION, PREVAILING_WAGE, YEAR, SOC_NAME.

After all the pre-processing steps we performed on our dataset to get the final transformed dataset we split the data into train and test. For the prediction task we use these 3 classifiers. They are Gaussian Naïve Bayes Classifier, Random-Forest classifier and XG-Boost.

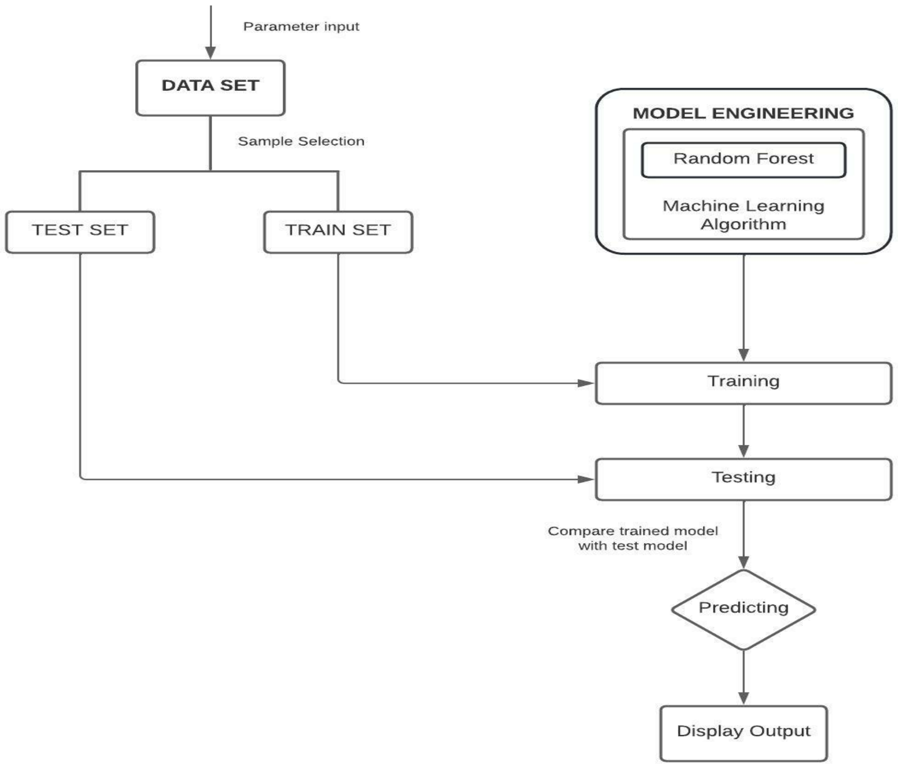
1. Naïve Bayes: Naive Bayes is a simple and interpretable model which assumes all features are conditionally independent given labels and are in Gaussian distribution. The function fit was used to fit the learning model on

the data and the function score was used to find out the F-score of this algorithm and to assess its performance.

2. Random-Forest: It is an ensemble technique which uses bagging technique. It uses number of meta-classifiers on various sub samples of the dataset and then averages the prediction to improve the final predictive outcome. This classifier can also control over fitting by proper parameter tuning.

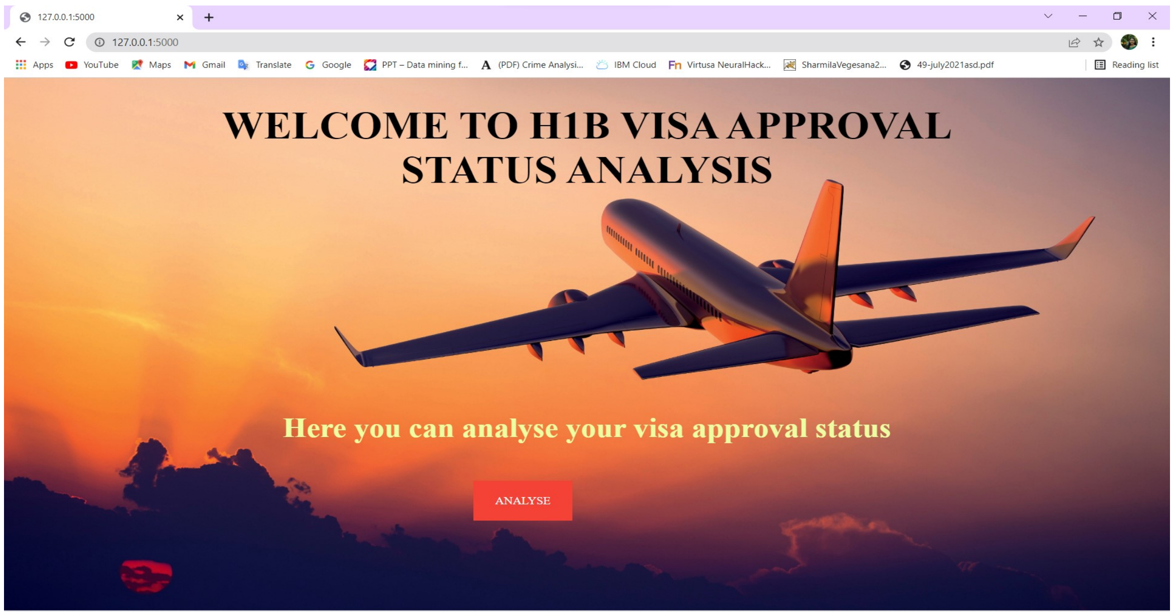
3. XG-Boost: XGBoost or Extreme Gradient Boost algorithm is an ensemble method. It uses ‘Bagging and Boosting’ techniques. In Bagging technique, trees are grown to their maximum extent and Boosting techniques uses trees with fewer splits. On aggregation of the two models, the final model gives us the outcome with less MSE (Mean Squared Error).

## 5. FLOWCHART



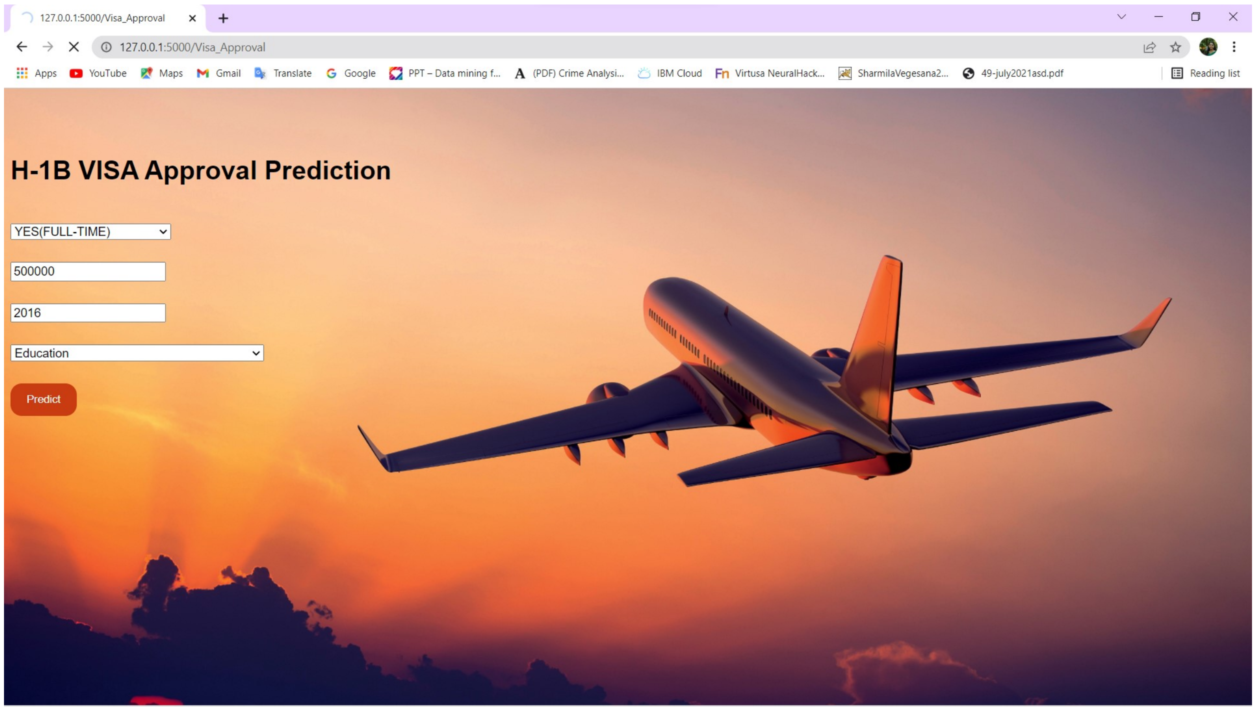
## 6. RESULT

The final result of the project is the predict the certification status of the visa applications.



WELCOME TO H1B VISA APPROVAL STATUS ANALYSIS

Here you can analyse your visa approval status



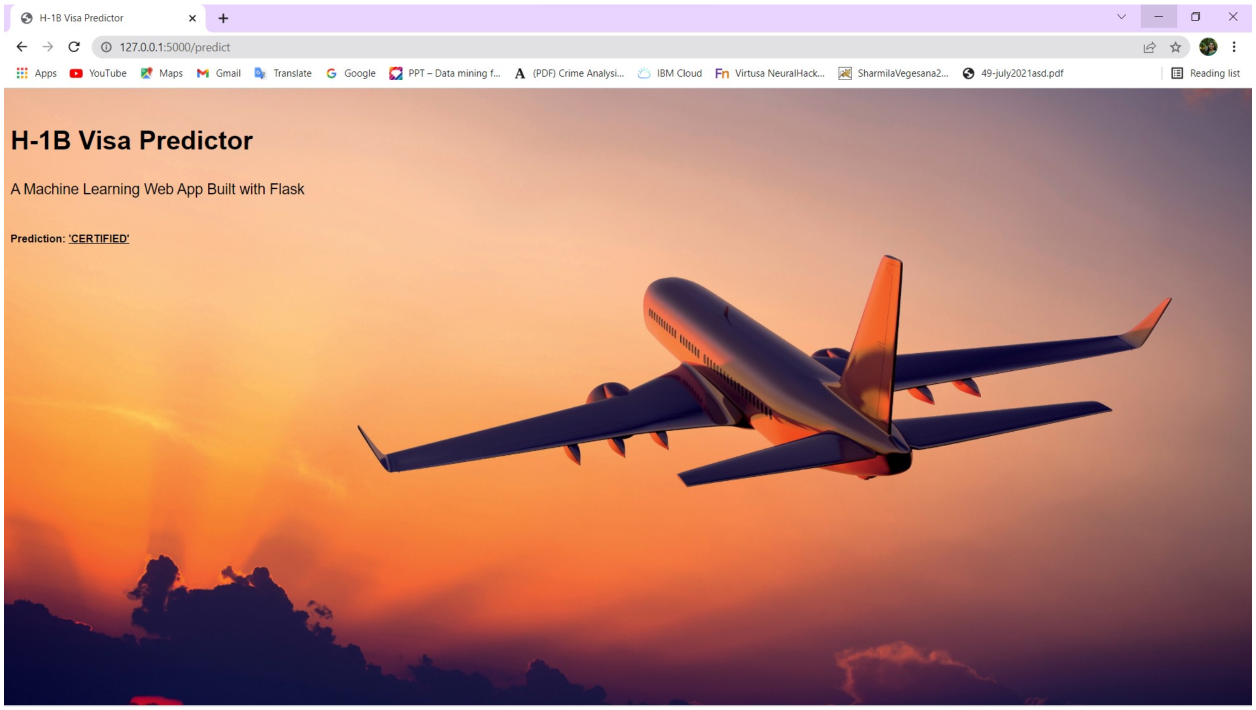
H-1B VISA Approval Prediction

YES(FULL-TIME)

500000

2016

Education



H-1B Visa Predictor

A Machine Learning Web App Built with Flask

**Prediction: 'CERTIFIED'**

## 7. ADVANTAGES & DISADVANTAGES

The main advantage of this proposed application is reduction of time. One can infer from these applications to know the case status of the application as certified or denied. we don’t need to check all data to know accepted or rejected.it take necessary data and make prediction. Hence it helps to reduce huge amount time for checking each and every application.

**Disadvantages**

- Need more datasets, to increase the accuracy of the algorithms.
- A large amount of data is used in the process of training and learning. So these use of data should be of good quality, unbiased.
- The proposed application can only be used by Employee or Applicant for prediction.
- The proposed application is Web-based, hence cannot be used in Mobile devices.
- The result of the application depends upon the accuracy of the algorithms

## 8. APPLICATIONS

Checking visa Approval takes huge amount of effort and time. This application will be more useful for students or job seeker who applied for h1b visa and Employee who is responsible for checking visa approval.

## 9.CONCLUSION

In this work, Gaussian Naive Bayes, Random Forest Classifier and XGBoost Classifier were considered for determining the status of H1-B visa applications. Random Forest Classifier performed the best in terms of accuracy, precision and F1 score over others. We achieved a best of 86.808% classification accuracy. Naïve Bayes classifier has performance of 51.92% accuracy. This leads to conclusion that how much important is feature selection and feature transformation is. Our results showed that the most predictive features are EMPLOYER SUCCESS RATE and PREVAILING WAGE. One can infer from these results that the chance of being certified increases with the amount of wage and how successful your sponsor was in the previous H1B applications.

## 10. FUTURE SCOPE

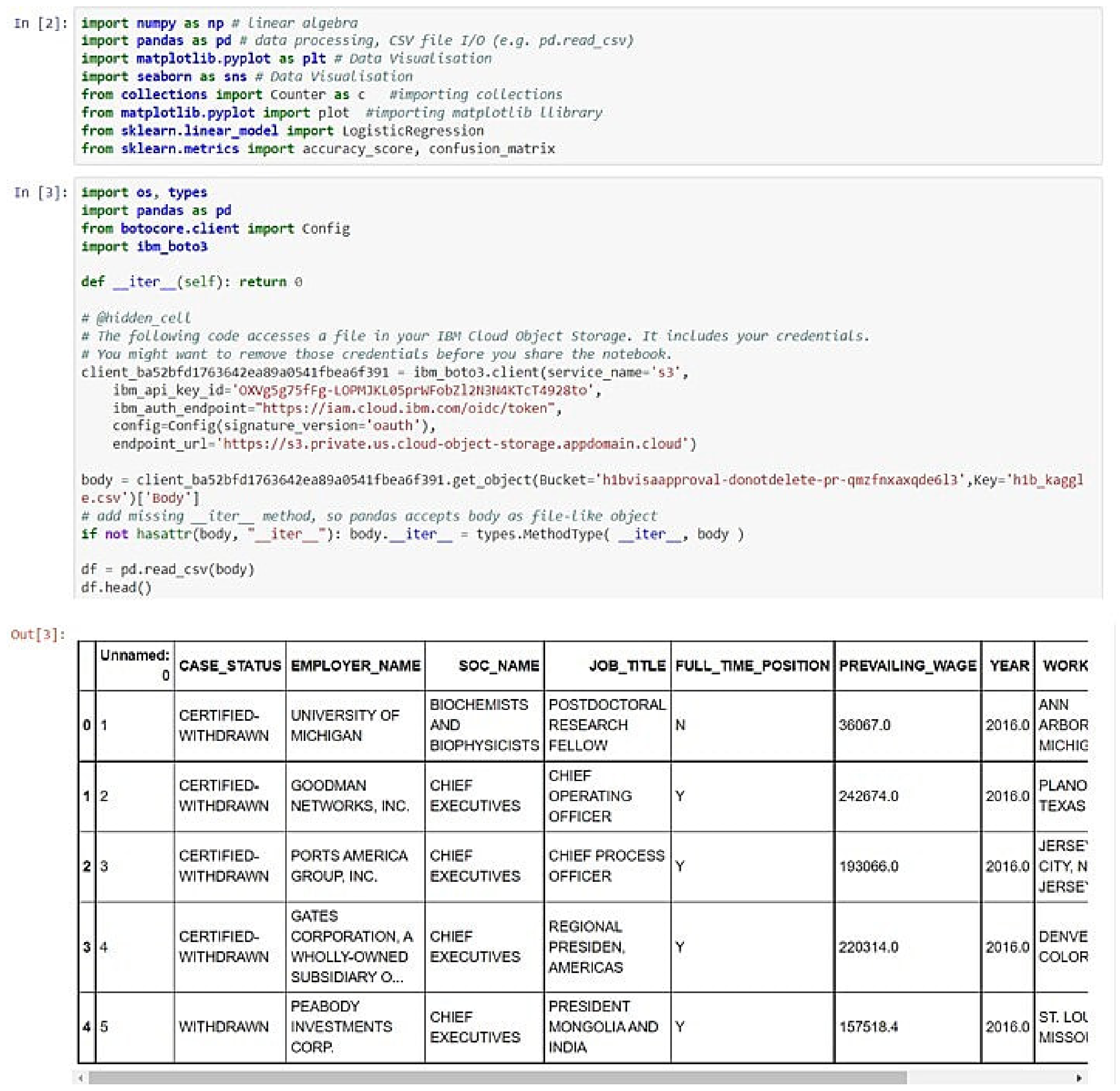
Supplemental data concerning the Standard Occupational Classification (SOC)   can be gathered and used in coordination with this data set to obtain a more comprehensive analysis of how the H-1B Visa selection process works. By using the wage evaluations and ranges under SOC, the wage attribute in this data set can be correctly put in to a range of salaries which can then be used to classify the visa petitions based on occupation roles rather than location wise. In addition, other classification algorithms other than the discriminative models can be experimented with this testbed and their performances can also be analyzed.

## 11. BIBILOGRAPHY

1) Prof. S. Sarkar IIT Kharagpur, Introduction to machine learning NPTEL :“https://www.youtube.com/playlist?list=PLYihddLF   CgYuWNL55Wg8ALkm6u 8U7gps

2) 5. Swain, D., Chakraborty, K., Dombe, A., Ashture, A., & Valakunde, N. (2018, December). Prediction of H1B Visa Using Machine Learning Al- gorithms. In 2018 International Conference on Advanced Computation and Telecommunication (ICACAT) (pp. 1-7). IEEE.

# APPENDIX



In [2]:
```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # Data Visualisation
import seaborn as sns # Data Visualisation
from collections import Counter as c   #importing collections
from matplotlib.pyplot import plot  #importing matplotlib library
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [3]:
```python
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_ba52bfd1763642ea89a0541fbea6f391 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='OXVg5g75fFg-LOPMJKL05prWFobZl2N3N4KTcT4928to',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

body = client_ba52bfd1763642ea89a0541fbea6f391.get_object(Bucket='h1bvisaapproval-donotdelete-pr-qmzfnxaxqde6l3',Key='h1b_kaggl
e.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df = pd.read_csv(body)
df.head()
```

Out[3]:

| | Unnamed: 0 | CASE_STATUS | EMPLOYER_NAME | SOC_NAME | JOB_TITLE | FULL_TIME_POSITION | PREVAILING_WAGE | YEAR | WORK |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CERTIFIED-WITHDRAWN | UNIVERSITY OF MICHIGAN | BIOCHEMISTS AND BIOPHYSICISTS | POSTDOCTORAL RESEARCH FELLOW | N | 36067.0 | 2016.0 | ANN ARBOR MICHIG |
| 1 | 2 | CERTIFIED-WITHDRAWN | GOODMAN NETWORKS, INC. | CHIEF EXECUTIVES | CHIEF OPERATING OFFICER | Y | 242674.0 | 2016.0 | PLANO TEXAS |
| 2 | 3 | CERTIFIED-WITHDRAWN | PORTS AMERICA GROUP, INC. | CHIEF EXECUTIVES | CHIEF PROCESS OFFICER | Y | 193066.0 | 2016.0 | JERSE CITY, N JERSE |
| 3 | 4 | CERTIFIED-WITHDRAWN | GATES CORPORATION, A WHOLLY-OWNED SUBSIDIARY O... | CHIEF EXECUTIVES | REGIONAL PRESIDEN, AMERICAS | Y | 220314.0 | 2016.0 | DENVE COLOR |
| 4 | 5 | WITHDRAWN | PEABODY INVESTMENTS CORP. | CHIEF EXECUTIVES | PRESIDENT MONGOLIA AND INDIA | Y | 157518.4 | 2016.0 | ST. LOU MISSO |

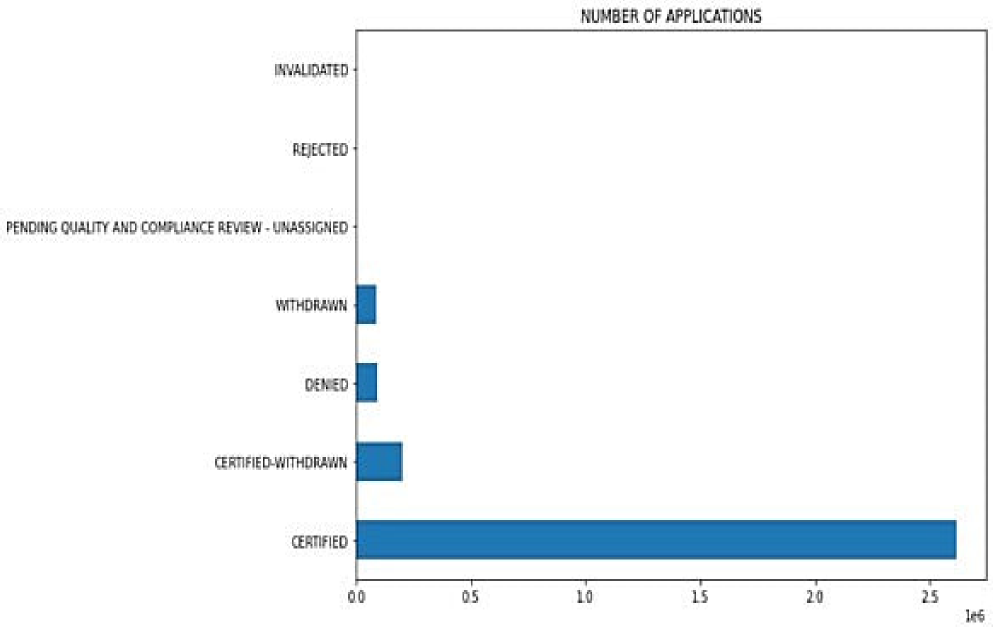
```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3002458 entries, 0 to 3002457
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: 252.0+ MB
```

```
In [6]: df.CASE_STATUS.value_counts()
```

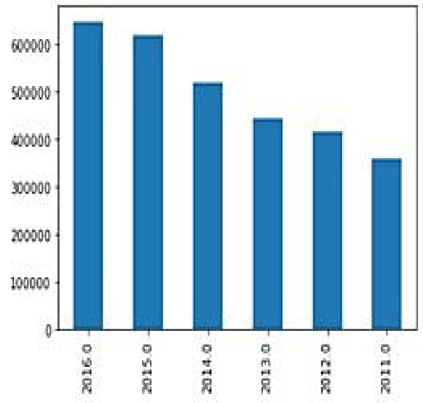
```
Out[6]: CERTIFIED                                             2615623
        CERTIFIED-WITHDRAWN                                    202659
        DENIED                                                  94346
        WITHDRAWN                                               89799
        PENDING QUALITY AND COMPLIANCE REVIEW - UNASSIGNED         15
        REJECTED                                                    2
        INVALIDATED                                                 1
        Name: CASE_STATUS, dtype: int64
```

```
In [7]: plt.figure(figsize=(10,7))
        df.CASE_STATUS.value_counts().plot(kind='barh')
        df.sort_values('CASE_STATUS')
        plt.title("NUMBER OF APPLICATIONS")
        plt.show()
```

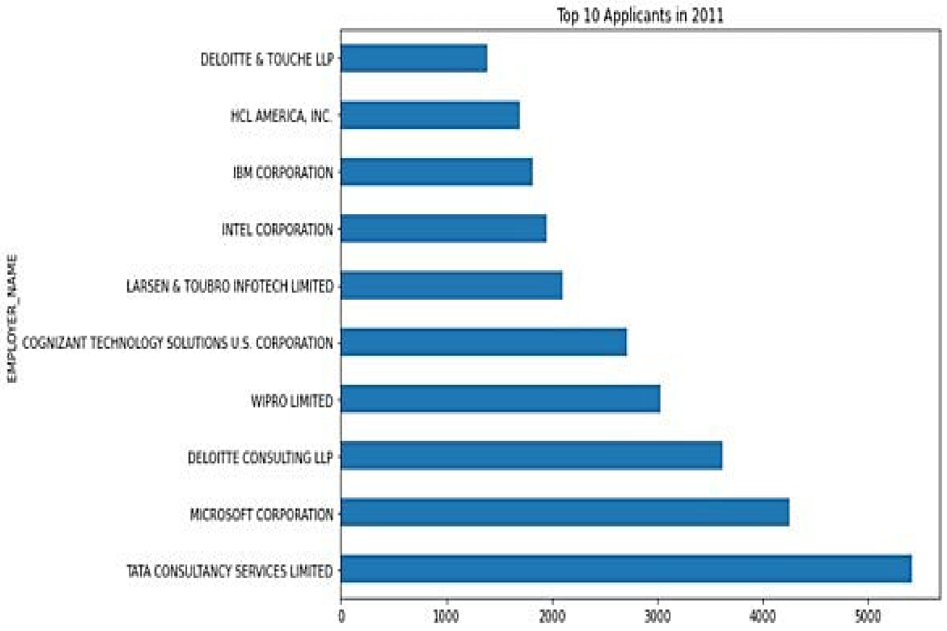


In [8]:
```
df.YEAR.value_counts().plot(kind = 'bar')
```

Out[8]:
```
<AxesSubplot:>
```



In [9]:
```
plt.figure(figsize=(10,7))

ax1 = df['EMPLOYER_NAME'][df['YEAR'] == 2011].groupby(df['EMPLOYER_NAME']).count().sort_values(ascending=False).head(10).plot(ki
nd='barh', title = "Top 10 Applicants in 2011")
ax1.set_label("")
plt.show()
```



## Removing Outliers

```
In [15]: df = df[df['PREVAILING_WAGE'] <= 500000]
         by_emp_year = df[['EMPLOYER_NAME', 'YEAR', 'PREVAILING_WAGE']][df['EMPLOYER_NAME'].isin(top_emp)]
         by_emp_year = by_emp_year.groupby([df['EMPLOYER_NAME'],df['YEAR']])
```

## Checking for Null values

```
In [16]: df.isnull().sum()

Out[16]: Unnamed: 0                 0
         CASE_STATUS                0
         EMPLOYER_NAME             42
         SOC_NAME               17698
         JOB_TITLE                 26
         FULL_TIME_POSITION         0
         PREVAILING_WAGE            0
         YEAR                       0
         WORKSITE                   0
         lon                   107089
         lat                   107089
         dtype: int64
```

```
In [17]: df['SOC_NAME'] = df['SOC_NAME'].fillna(df['SOC_NAME'].mode()[0])
```

```
In [18]: df.isnull().sum()

Out[18]: Unnamed: 0                 0
         CASE_STATUS                0
         EMPLOYER_NAME             42
         SOC_NAME                   0
         JOB_TITLE                 26
         FULL_TIME_POSITION         0
         PREVAILING_WAGE            0
         YEAR                       0
         WORKSITE                   0
         lon                   107089
         lat                   107089
         dtype: int64
```

## Label Encoding Case status

```
In [19]: df['CASE_STATUS'] = df['CASE_STATUS'].map({'CERTIFIED' : 0, 'CERTIFIED-WITHDRAWN' : 1, 'DENIED' : 2,'WITHDRAWN' : 3,'PENDING QUA
         LITY AND COMPLIANCE REVIEW - UNASSIGNED' : 4,'REJECTED' : 5, 'INVALIDATED' : 6})
```

```
In [21]: import sys
         df['SOC_NAME1'] = 'others'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('computer','software')] = 'it'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('chief','management')] = 'manager'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('mechanical')] = 'mechanical'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('database')] = 'database'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('sales','market')] = 'scm'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('financial')] = 'finance'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('public','fundraising')] = 'pr'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('education','law')] = 'administrative'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('auditors','compliance')] = 'audit'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('distribution','logistics')] = 'scm'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('recruiters','human')] = 'hr'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('agricultural','farm')] = 'agri'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('construction','architectural')] = 'estate'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('forencsic','health')] = 'medical'
         df['SOC_NAME1'][df['SOC_NAME'].str.contains('teachers')] = 'education'
```

```
In [22]: df = df.drop(['Unnamed: 0', 'EMPLOYER_NAME', 'SOC_NAME','JOB_TITLE','WORKSITE', 'lon','lat'], axis = 1)
```

```
In [23]: df.head()
```

Out[23]:

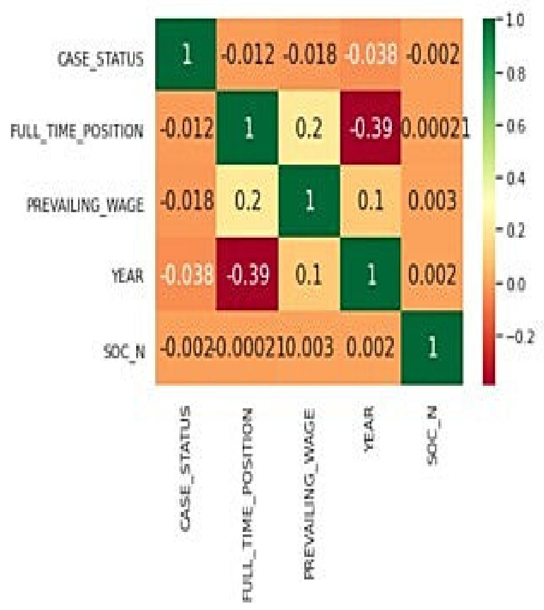
| | CASE_STATUS | FULL_TIME_POSITION | PREVAILING_WAGE | YEAR | SOC_NAME1 |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 36067.0 | 2016.0 | others |
| 1 | 1 | 1 | 242674.0 | 2016.0 | others |
| 2 | 1 | 1 | 193066.0 | 2016.0 | others |
| 3 | 1 | 1 | 220314.0 | 2016.0 | others |
| 4 | 3 | 1 | 157518.4 | 2016.0 | others |

```
In [24]: from sklearn import preprocessing
         le = preprocessing.LabelEncoder()
         le.fit(df.SOC_NAME1)
         # print list(le.classes_)
         df['SOC_N']=le.transform(df['SOC_NAME1'])
```

```
In [25]: df = df.drop(['SOC_NAME1'], axis=1)
```

```
In [26]: sns.heatmap(df.corr(), annot=True, cmap="RdYlGn", annot_kws={"size":15})
```

```
Out[26]: <AxesSubplot:>
```



```
In [34]: accuracy = accuracy_score(y_test,y_pred_rf)
         accuracy

Out[34]: 0.8687839870929605
```

```
In [35]: import pickle
         pickle.dump(rf,open('Visarf.pkl','wb'))
```

```
In [36]: pip install ibm_watson_machine_learning

         Requirement already satisfied: ibm_watson_machine_learning in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (1.0.
         189)
         Requirement already satisfied: pandas<1.4.0,>=0.24.2 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_w
         atson_machine_learning) (1.2.4)
         Requirement already satisfied: ibm-cos-sdk==2.7.* in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_wats
         on_machine_learning) (2.7.0)
         Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine
         _learning) (0.8.9)
         Requirement already satisfied: certifi in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine_
         learning) (2021.10.8)
         Requirement already satisfied: requests in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine
         _learning) (2.25.1)
         Requirement already satisfied: packaging in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machin
         e_learning) (20.9)
         Requirement already satisfied: lomond in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine_l
         earning) (0.3.3)
         Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm_watson_machine_
         learning) (1.26.6)
```

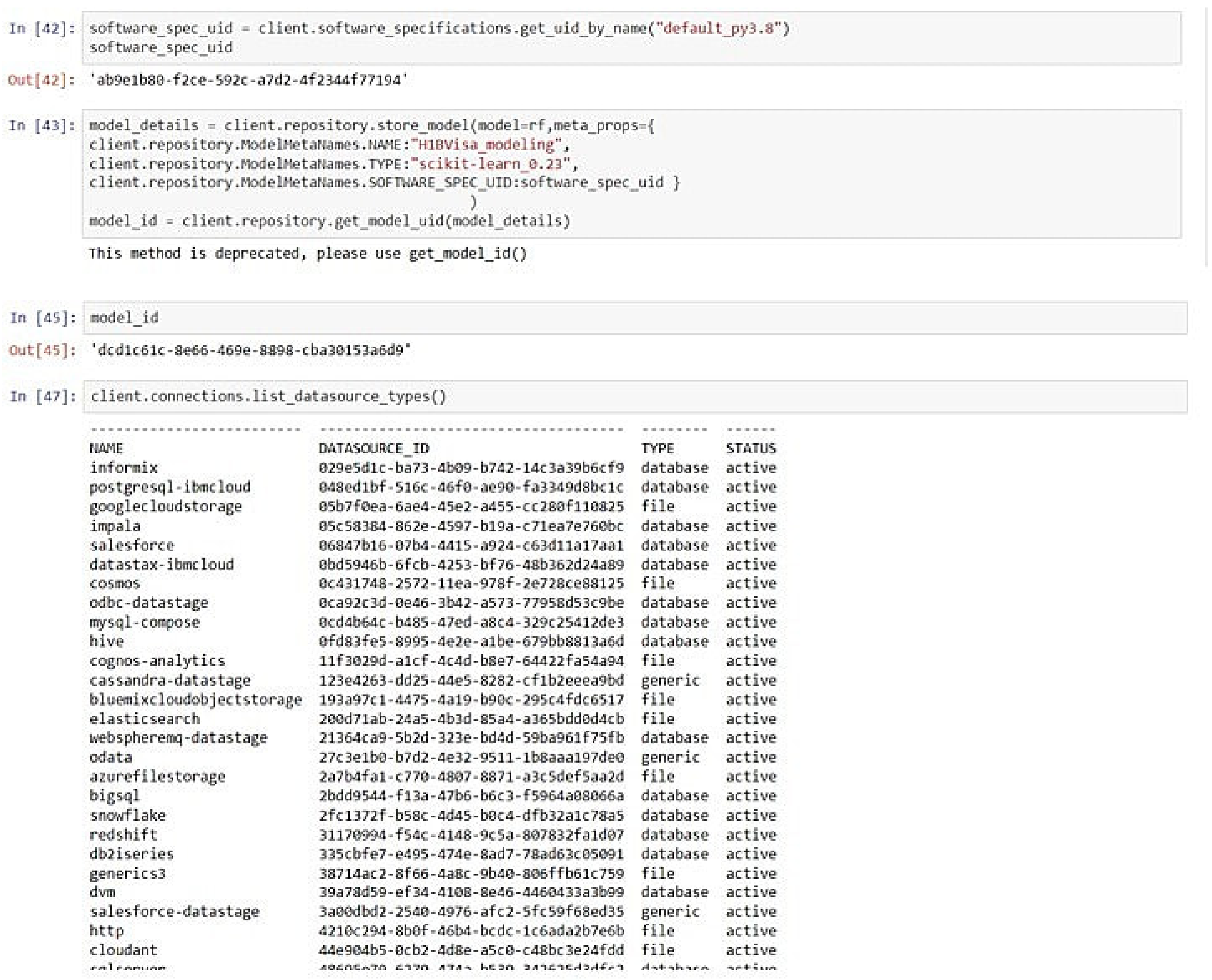
```
In [37]: from ibm_watson_machine_learning import APIClient
         wml_credentials = {
                             "url": "https://us-south.ml.cloud.ibm.com",
                             "apikey":"TC_5gO5FshF_PS3XRD0DpP4Dg-PYgbbWssRd5G7dwyPY"
                           }
         client = APIClient(wml_credentials)
```

```
In [38]: def guid_from_space_name(client, space_name):
             space = client.spaces.get_details()
             #print(space)
             return(next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])
```

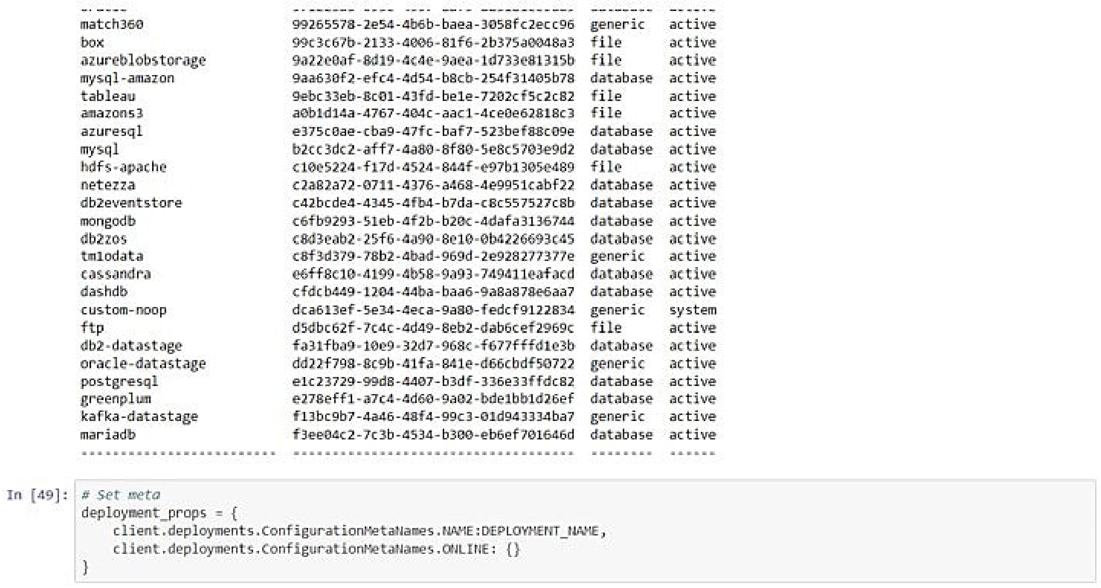
```
In [39]: space_uid = guid_from_space_name(client, 'models')
         print("space UID = " + space_uid)

         space UID = fd717ce6-173f-4961-bad7-015df1f71da4
```

```
In [40]: client.set_default_space(space_uid)

Out[40]: 'SUCCESS'
```

```
In [41]: client.software_specifications.list()

         ----------------------------  ------------------------------------  ----
         NAME                          ASSET_ID                              TYPE
         default_py3.6                 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
         pytorch-onnx_1.3-py3.7-edt    069ea134-3346-5748-b513-49120e15d288  base
         scikit-learn_0.20-py3.6       09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
         spark-mllib_3.0-scala_2.12    09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
         pytorch-onnx_rt22.1-py3.9     0b848dd4-e681-5599-be41-b5f6fccc6471  base
         ai-function_0.1-py3.6         0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
         shiny-r3.6                    0e6e79df-875e-4f24-8ae9-62dcc2148306  base
         tensorflow 2.4-py3.7-horovod  1092590a-307d-563d-9b62-4eb7d64b3f22  base
```



```
In [42]: software_spec_uid = client.software_specifications.get_uid_by_name("default_py3.8")
         software_spec_uid

Out[42]: 'ab9e1b80-f2ce-592c-a7d2-4f2344f77194'

In [43]: model_details = client.repository.store_model(model=rf,meta_props={
         client.repository.ModelMetaNames.NAME:"H1BVisa_modeling",
         client.repository.ModelMetaNames.TYPE:"scikit-learn_0.23",
         client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid }
                                                        )
         model_id = client.repository.get_model_uid(model_details)

         This method is deprecated, please use get_model_id()

In [45]: model_id

Out[45]: 'dcd1c61c-8e66-469e-8898-cba30153a6d9'

In [47]: client.connections.list_datasource_types()

         ------------------------  ------------------------------------  --------  ------
         NAME                      DATASOURCE_ID                         TYPE      STATUS
         informix                  029e5d1c-ba73-4b09-b742-14c3a39b6cf9  database  active
         postgresql-ibmcloud       048ed1bf-516c-46f0-ae90-fa3349d8bc1c  database  active
         googlecloudstorage        05b7f0ea-6ae4-45e2-a455-cc280f110825  file      active
         impala                    05c58384-862e-4597-b19a-c71ea7e760bc  database  active
         salesforce                06847b16-07b4-4415-a924-c63d11a17aa1  database  active
         datastax-ibmcloud         0bd5946b-6fcb-4253-bf76-48b362d24a89  database  active
         cosmos                    0c431748-2572-11ea-978f-2e728ce88125  file      active
         odbc-datastage            0ca92c3d-0e46-3b42-a573-77958d53c9be  database  active
         mysql-compose             0cd4b64c-b485-47ed-a8c4-329c25412de3  database  active
         hive                      0fd83fe5-8995-4e2e-a1be-679bb8813a6d  database  active
         cognos-analytics          11f3029d-a1cf-4c4d-b8e7-64422fa54a94  file      active
         cassandra-datastage       123e4263-dd25-44e5-8282-cf1b2eeea9bd  generic   active
         bluemixcloudobjectstorage 193a97c1-4475-4a19-b90c-295c4fdc6517  file      active
         elasticsearch             200d71ab-24a5-4b3d-85a4-a365bdd0d4cb  file      active
         webspheremq-datastage     21364ca9-5b2d-323e-bd4d-59ba961f75fb  database  active
         odata                     27c3e1b0-b7d2-4e32-9511-1b8aaa197de0  generic   active
         azurefilestorage          2a7b4fa1-c770-4807-8871-a3c5def5aa2d  file      active
         bigsql                    2bdd9544-f13a-47b6-b6c3-f5964a08066a  database  active
         snowflake                 2fc1372f-b58c-4d45-b0c4-dfb32a1c78a5  database  active
         redshift                  31170994-f54c-4148-9c5a-807832fa1d07  database  active
         db2iseries                335cbfe7-e495-474e-8ad7-78ad63c05091  database  active
         generics3                 38714ac2-8f66-4a8c-9b40-806ffb61c759  file      active
         dvm                       39a78d59-ef34-4108-8e46-4460433a3b99  database  active
         salesforce-datastage      3a00dbd2-2540-4976-afc2-5fc59f68ed35  generic   active
         http                      4210c294-8b0f-46b4-bcdc-1c6ada2b7e6b  file      active
         cloudant                  44e904b5-0cb2-4d8e-a5c0-c48bc3e24fdd  file      active
```



```
         match360                  99265578-2e54-4b6b-baea-3058fc2ecc96  generic   active
         box                       99c3c67b-2133-4006-81f6-2b375a0048a3  file      active
         azureblobstorage          9a22e0af-8d19-4c4e-9aea-1d733e81315b  file      active
         mysql-amazon              9aa630f2-efc4-4d54-b8cb-254f31405b78  database  active
         tableau                   9ebc33eb-8c01-43fd-be1e-7202cf5c2c82  file      active
         amazons3                  a0b1d14a-4767-404c-aac1-4ce0e62818c3  file      active
         azuresql                  e375c0ae-cba9-47fc-baf7-523bef88c09e  database  active
         mysql                     b2cc3dc2-aff7-4a80-8f80-5e8c5703e9d2  database  active
         hdfs-apache               c10e5224-f17d-4524-844f-e97b1305e489  file      active
         netezza                   c2a82a72-0711-4376-a468-4e9951cabf22  database  active
         db2eventstore             c42bcde4-4345-4fb4-b7da-c8c557527c8b  database  active
         mongodb                   c6fb9293-51eb-4f2b-b20c-4dafa3136744  database  active
         db2zos                    c8d3eab2-25f6-4a90-8e10-0b4226693c45  database  active
         tm1odata                  c8f3d379-78b2-4bad-969d-2e928277377e  generic   active
         cassandra                 e6ff8c10-4199-4b58-9a93-749411eafacd  database  active
         dashdb                    cfdcb449-1204-44ba-baa6-9a8a878e6aa7  database  active
         custom-noop               dca613ef-5e34-4eca-9a80-fedcf9122834  generic   system
         ftp                       d5dbc62f-7c4c-4d49-8eb2-dab6cef2969c  file      active
         db2-datastage             fa31fba9-10e9-32d7-968c-f677fffd1e3b  database  active
         oracle-datastage          dd22f798-8c9b-41fa-841e-d66cbdf50722  generic   active
         postgresql                e1c23729-99d8-4407-b3df-336e33ffdc82  database  active
         greenplum                 e278eff1-a7c4-4d60-9a02-bde1bb1d26ef  database  active
         kafka-datastage           f13bc9b7-4a46-48f4-99c3-01d943334ba7  generic   active
         mariadb                   f3ee04c2-7c3b-4534-b300-eb6ef701646d  database  active
         ------------------------  ------------------------------------  --------  ------

In [49]: # Set meta
         deployment_props = {
             client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
             client.deployments.ConfigurationMetaNames.ONLINE: {}
         }
```