Technocolabs
Build With AI

INTERNSHIP REPORT

TEAM A

DECLARATION

I hereby declare that all the Internship tasks submitted are our own unaided work. All direct or indirect sources used are acknowledged as references. For the comparison of my work with existing sources, I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future Internship submitted. Further rights of reproduction and usage, however, are not granted here. This paper was not previously presented to another examination board and has not been published.

TEAM B

H1- B VISA PREDICTION WEBAPP

I. Definition

Project Overview

Engineers around the world dream of working in the United States, especially in the bay area, where the coolest advancements in technology happen every day. Many talented engineers from foreign countries work hard to get temporary visas to start their American dream. H-1B is a visa issued by the U.S. government to foreign nations in special occupations. Employers who are willing to sponsor a foreign national need to file a labor condition application (LCA) and get it approved in order to be considered for an H-1B lottery each year.

Problem Statement

Given the complex procedures and paperwork that are required, most companies delegate H-1B applications to an immigration attorney. Each application needs to adhere to a proper format and satisfy the conditions required for each job category in order to be successfully certified. The whole process is usually achieved by law professionals communicating with the applicant and the employer multiple times. A denied application not only incurs additional costs for each individual's time but could also potentially lower the applicant's chance to be successfully certified for the same position. Hence, it is imperative that each filed application satisfies requirements specified by USCIS and is comparable to other applications from the same industry.

For my capstone project, I created two different supervised classifier models (Random Forest and Logistic Regression), evaluated the performance of each of them, and decided the final model based on performance and feasibility. I started by figuring out the set of features that are common between data sources from different years, resolving naming conflict, and discarding infeasible and least useful features. After cleaning up records with empty values, I tried the initial implementation of these models and realized all models severely overfit a dominant class due to highly imbalanced training data. I random-sampled records from the dominant class to achieve about the same balance between the two classes, applied dimensionality reduction (PCA) to the dataset and implemented these models again with default parameters.

All the models turned out to be successful, and parameter tuning was applied to each of them to optimize performance further. Random Forest classifier was selected as the final model due to its high performance and efficiency. The final model was tested with manipulated data with random noise to demonstrate it is robust enough and not significantly affected by noisy and unseen data. The set of features is texts entered in different sections of a filed LCA. A successful model enables individuals filing an application to quickly check whether the information they enter in the application is strong enough to be considered by USCIS. This model does not replace an immigration attorney by any means but could be used as a quick sanity check after a proper application is created.

Metrics

Since the purpose of this model is to evaluate the likelihood of an unsuccessful outcome, a pessimistic model which returns a slightly lower chance of getting certified is much more tolerable than an overly optimistic one. An application with a false negative could be improved to increase confidence level, but a false positive result is certainly not acceptable. For this reason, my model is a high-precision model, which is calculated as follows:

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

However, this does not mean we can completely ignore recall and create a skewed model. To give more weights to precision than recall, an F-beta score with beta = 0.5 was used to evaluate the performance of each model.

$$F = \frac{1.25 * \text{Precision} * \text{Recall}}{0.25 * \text{Precision} + \text{Recall}}$$

II. Analysis

Data Exploration

United States Department of Labour discloses annual statistics for research and analysis purposes. Since this is actual data that was used by DOL to decide whether or not each application is certified, it was used as source data for all models. The disclosure data is broken down per year, each of which is an excel spreadsheet format. Each year’s spreadsheet has slightly different column names; some of them are merely renamed fields from previous years, and some are totally new fields added in newer years.

After selecting (renamed) columns that are present in all year's records, the concatenated data has 2462248 samples before going through the clean-up and sampling stage. The concatenated dataset consists of the following 40 columns.

FIELD NAME	DESCRIPTION
Case Number	Unique identifier assigned to each application submitted
Case Status	Status associated with the last decision (“Certified,” “Certified-Withdrawn,” Denied,” and “Withdrawn”)
Case Submitted	Date and time the application was submitted.
Decision Date	Date on which the decision was recoded
Visa Class	A H-1B visa technically has three more subsets. Possible values include 'H-1B’, ‘H-1B1 Chile’, ‘H-1B1 Singapore’, ’E-3 Australian’.
Employment Start Date	Start date of employment
Employment End Date	End date of employment
Employer Name	Name of employer submitting labor condition application.
Employer Address	Geographical information of the Employer requesting temporary labor certification
Employer City	
Employer State	
Employer Postal Code	
Employer Country	
Employer Province	
Employer Phone	Contact information of the Employer requesting temporary labor certification
Employer Phone Ext.	
Agent Attorney Name	Contact information of the attorney filing an H-1B application on behalf of the employer
Agent Attorney City	
Agent Attorney State	
Job Title	Title of the job
SOC Code	Occupational code associated with the job being requested for temporary labor condition, as classified by the Standard Occupational Classification (SOC) System.
NAICS Code	Industry code associated with the employer requesting permanent labor condition, as classified by the North American Industrial Classification System (NAICS)
Full Time Position	Boolean field (Y/N) indicating full-time and part-tome position
Prevailing Wage	Prevailing Wage for the job being requested for temporary labor condition (continuous integer values)
PW Unit of Pay	Unit of pay for prevailing wage (Hour, Week, Bi-Weekly, Month, Year).
PW Wage Level	Prevailing wage level ("I", "II", "III", "IV" or "N/A")
PW Wage Source	prevailing wage source (“OES”, “CAB”, “DBA”, “SCA”, “Other”)
PW Wage Source Year	Year the Prevailing Wage Source was Issued
PW Wage Source Other	Detailed text field for "Other” for wage source
Wage Rate Of Pay From	Wage offered by the employer for the position (continuous float value)
Wage Rate Of Pay To	Maximum wage the employer is willing to pay for the position (continuous float value)

Wage Rate Of Pay	Unit of pay for wage offered by employer (Hour, Week, Bi-Weekly, Month, Year).
H-1B Dependent	Boolean field (Y/N) indicating whether the employer is hugely depending on H1B employees
Willful Violator	Boolean field (Y/N) indicating whether the employer committed a willful failure or a misrepresentation of a material fact in the past. A willful violator employer must provide additional evidences for any LCA it files and could be subject to random investigations
Worksite City	Geographical information of worksite
Worksite Country	
Worksite State	
Worksite Postal Code	

Table 1: List of Common Features in Disclosed Data.

Fields “Prevailing Wage”, “Wage Rate of Pay From”, and “Wage Rate of Pay To” are continuous floating-point numbers. All other fields are categorical fields with possible values as described above.

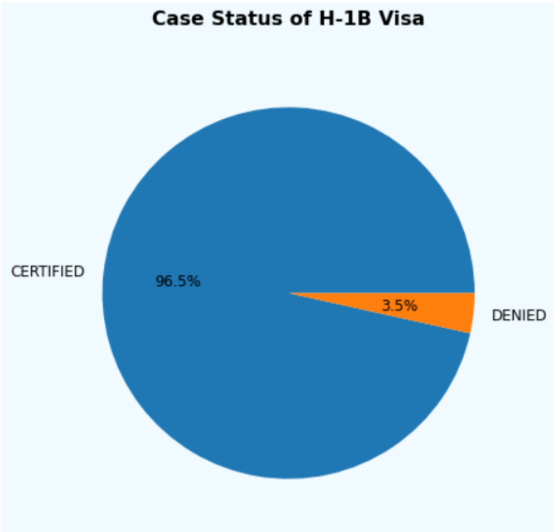
Exploratory Visualization

Below is the initial observation of features from the dataset.

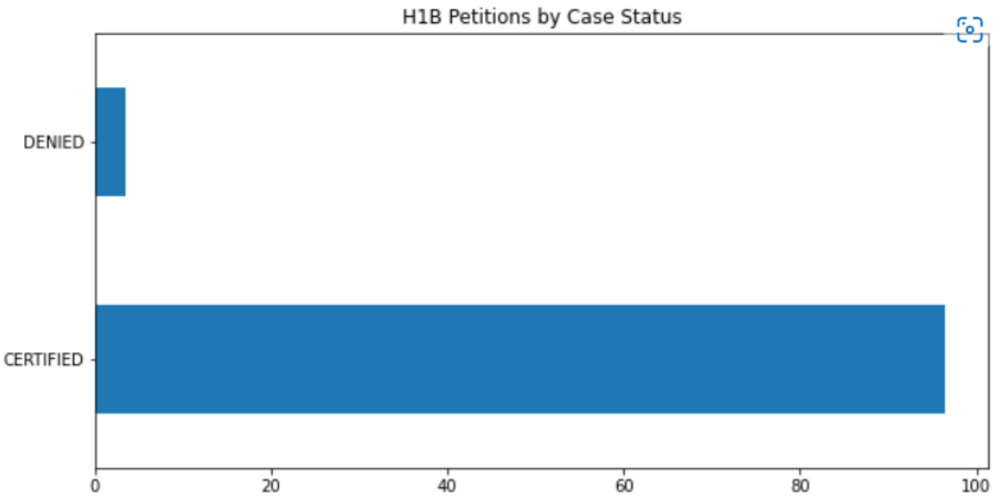
- While most categorical features (excluding geographical and contact information) have only a few values, the following features have more than 10 unique values:
- “SOC Code” and NAICS Code have 1250 and 3508 unique values, respectively.
- “Job Title” and “Employer Name” have 169572 and 148747 unique values, respectively, even after removing special characters and converting to uppercased strings.
- Three columns (“Wage Rate of Pay From”, “Wage Rate of Pay To”, and “Prevailing Wage”) representing wages are the only continuous features in the dataset. Given that feature “Wage Rate Of Pay To” is an optional field, it is missing from 21% of samples and set to 0 from 59% of samples.

Case Status –

As shown in Figure 1, the dataset is significantly imbalanced in that only 3.5 % of samples are labeled “DENIED”. A careful sampling technique is required to prevent the models from overfitting to the dominant class.



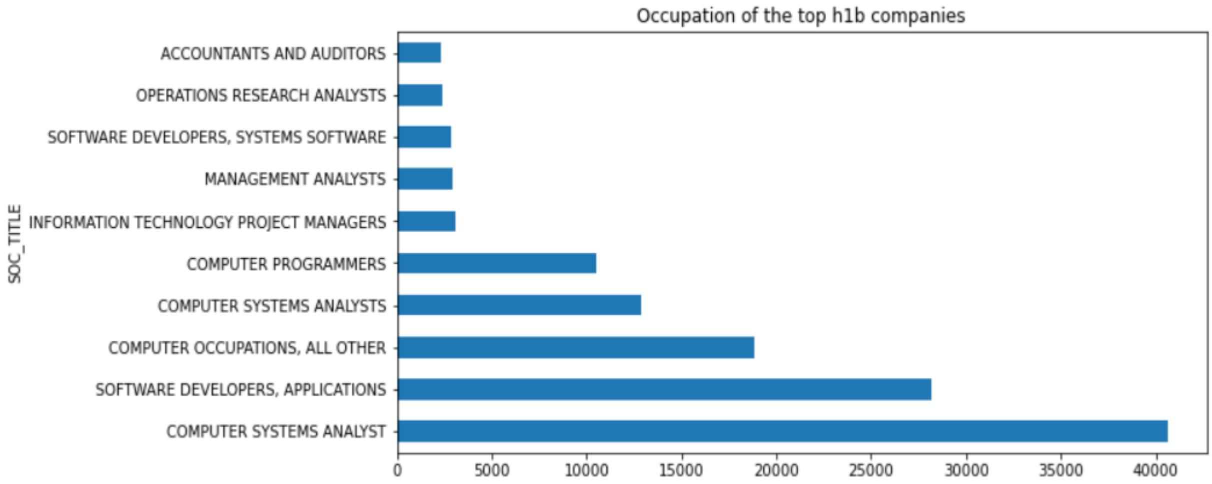
PIE chart showing different cases of CASE-STATUS



% Distribution of Samples with Each Class

SOC_TITLE –

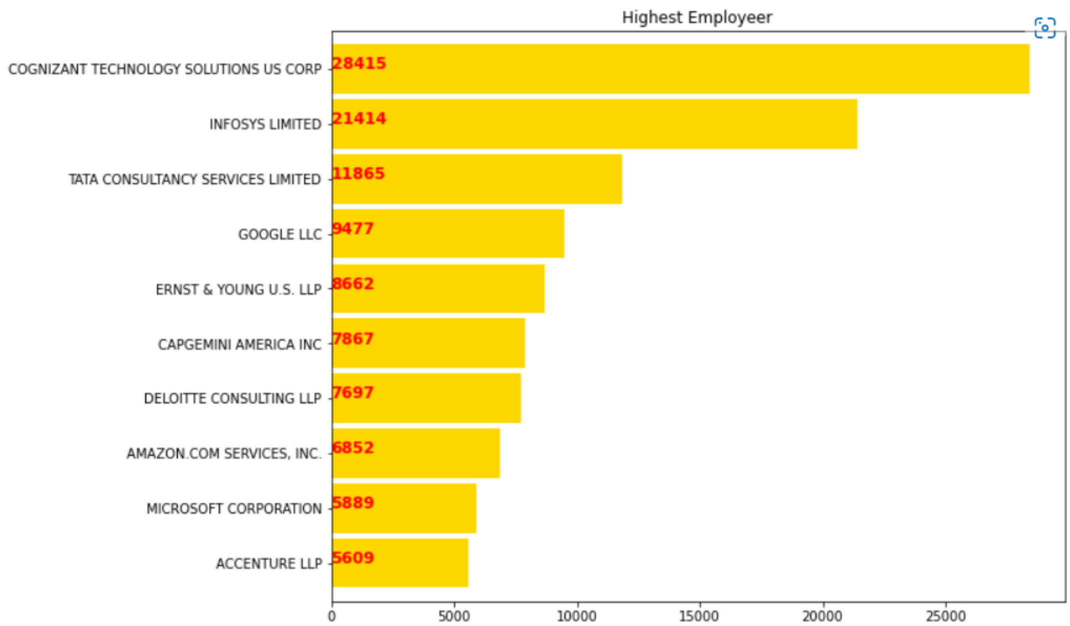
SOC title has details about the position, occupation field, and seniority of the applicant.



The top OCCUPATIONS of the H1-B's being filed by the employers.

EMPLOYER-NAME –

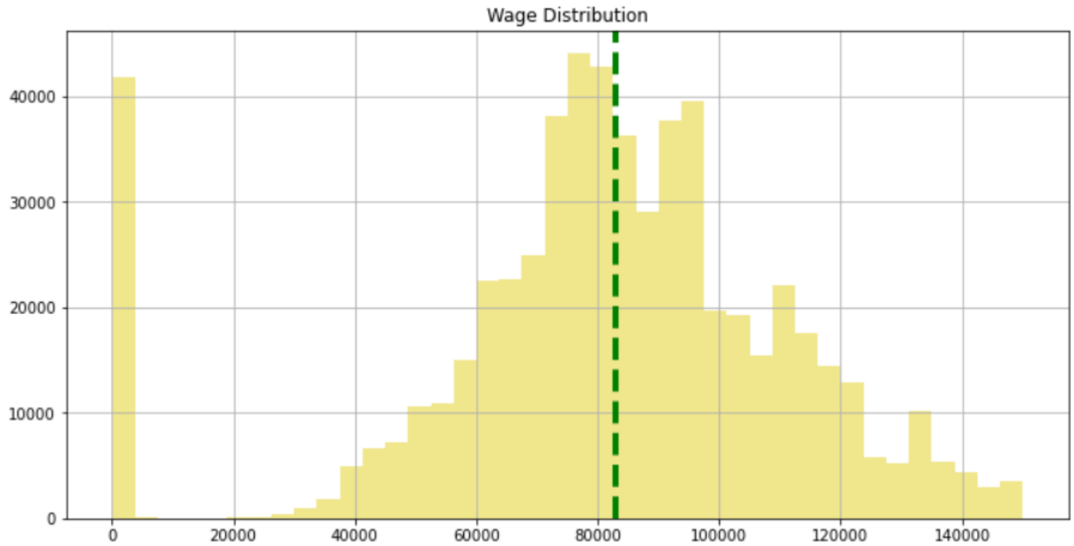
The employer's name submitted the visa application. We believe an employer's name is one of the important features to profile the visa application. As per the NY Times, some companies are manipulating the visa process by flooding the system.



Plot showing which Employees file the most petitions.

PREVAILING WAGE –

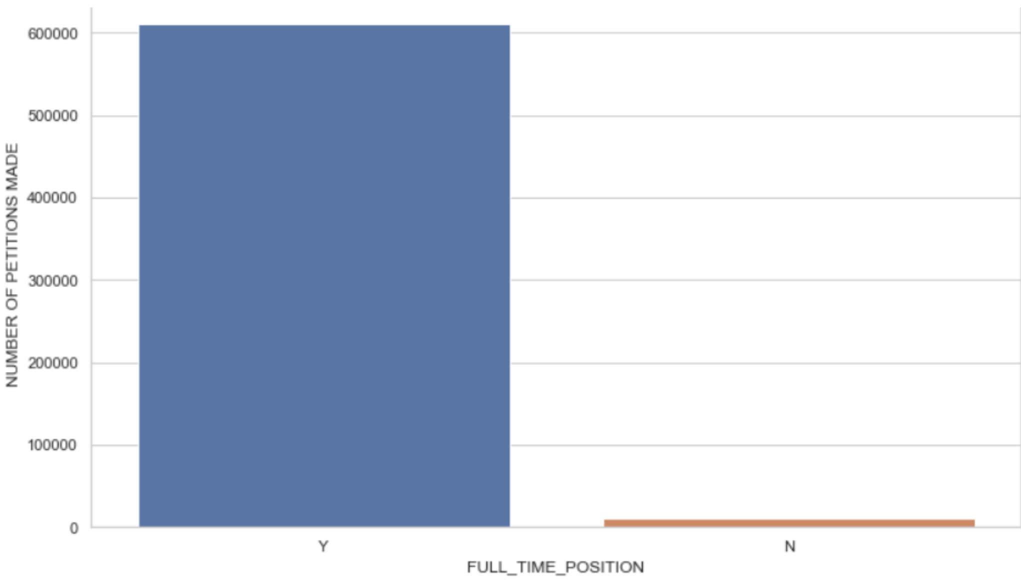
We can see in the below plot that the mean salary of the employees is nearly about 80000 dollars.



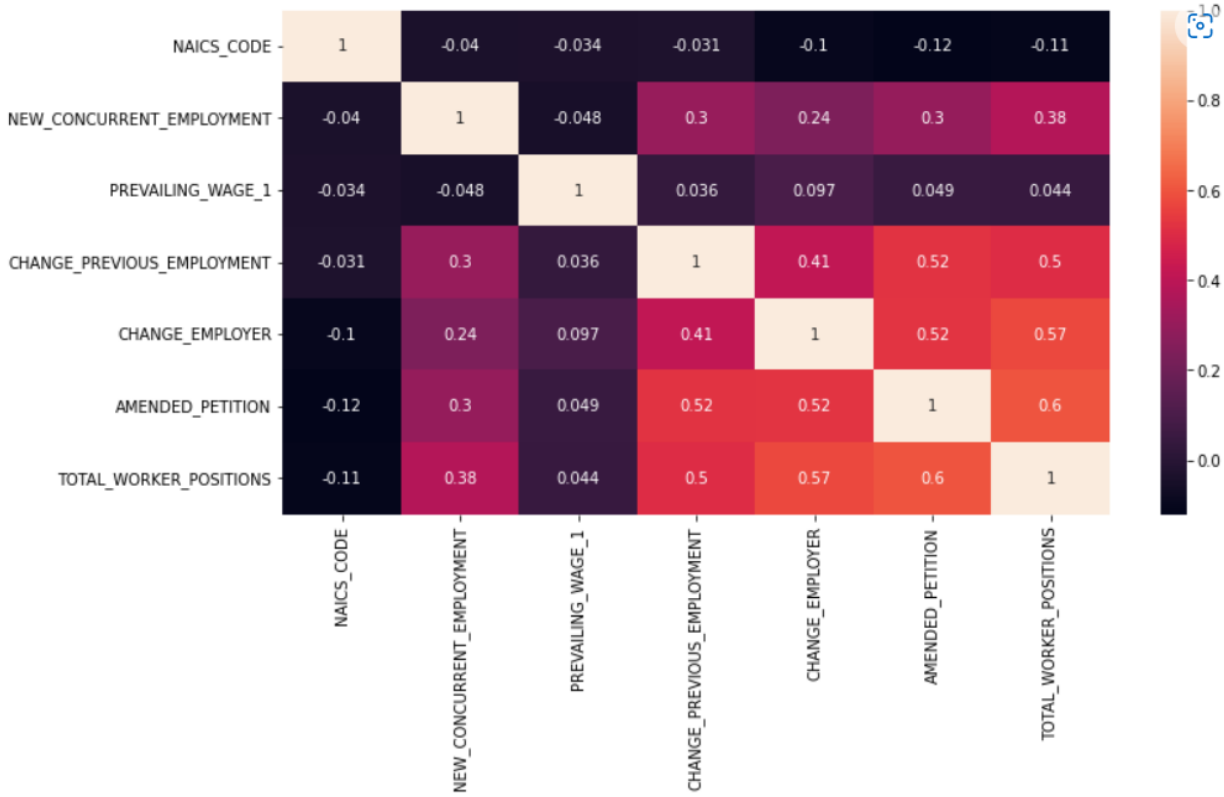
Plot showing the Wage Distribution among all the employees.

FULL-TIME POSITION –

Here we can see how many of the applicants are working as full-time.



Let's see the correlation of the features of the dataset among each other, by plotting a heatmap using a correlation matrix.



We can see in the above heatmap that Total Worker Positions are nicely correlated with AMENDED_ PETITION, CHANGE_EMPLOYER, and CHANGE_PREVIOUS_EMPLOYMENT.

On the other hand, we can also see that NAICS_CODE and PREVAILING_WAGE_1 are not correlated enough with the other features in the dataset.

III. METHODOLOGY-

DATA PRE-PROCESSING -

As discussed above, the feature “Case Status” is the label that all models take and try to predict. Although there are four classes in total (“Certified”, “Certified-Withdrawn”, “Denied”, “Withdrawn”), there are actually only two true classes. “Certified-Withdrawn” labeled values were removed since these are certified applications that were voluntarily withdrawn, and samples with the label “Withdrawn” were also excluded from the input data since they did not go through the case decision process. After making these changes, there were 1479869 samples, 3.5 % with the label “Denied” and 96.5 % with the label “Certified”.

Excluded Features -

After carefully reviewing the values of all features, the following were excluded from the dataset:

- Attorney name, attorney city, attorney state: Common reasons for LCA denial include filing incomplete LCA, failure to pay filing fees, and insufficient job description. My initial assumption was that these trivial mistakes would significantly decrease if the application is handled by an immigration attorney. However, as seen in the above figures, samples with both labels showed almost the same distribution of whether the attorney's name is missing. This tells us that whether these fields are empty doesn’t really make a difference in terms of case decision, so features “Agent Attorney City”, “Agent Attorney Name”, and “Agent Attorney State” are excluded.
- Case number, case submitted date, case decision date: As disclosed on the DoL website, most applications got processed in a timely manner, and there wasn’t much variation in terms of turnaround time. A case number is a unique identifier for each case, which doesn’t help us in generalizing a pattern.
- Employer’s street address, city, state, phone, phone extension, postal code, province, country: Additional geographical and contact information of the employer are unique to each employer, and LCA decision is not dependent on location.
- Worksite city, state, postal code: Similarly, additional geographical information for the worksite doesn’t add any value to the LCA decision process.

- Prevailing wage source other: This is an additional text field for the “other” option for the prevailing wage source. Since there are 10663 unique values, it’s not feasible to one-hot encode this field.
- Prevailing wage source year: This is the year the prevailing wage source was issued. Not useful since applicants would pull the latest source for most applications.
- PW_WAGE_LEVEL: This represents the level of prevailing wage (I, II, III, IV), but data for the year 2019 is missing this column and it cannot be inferred from other years’ data.

Feature Selection -

- Given there are significantly more samples with the “Certified” label, any sample in this class that’s missing any of the selected features was excluded.
- **CASE_STATUS:** As explained at the beginning of the data preprocessing section, samples with “Withdrawn” status were excluded and the class “Certified-Expired” was renamed to “Certified”. This will be the label used for all models.
- **SOC_CODE:** It is a federal statistical standard code for classifying occupations by categories. A valid SOC code is “xx-xxxx”, but there are quite a few values with invalid values. Only values in the format of “xx.xxxx”, “xxxxxx”, and “xxxxx” were corrected by replacing “.” with”-“ or adding it to the second index, and the rest were assigned null. For samples with denied labels, the value for “Job Title” was filled in for nulls.
- **NAICS_CODE:** If the values are present, most of them are either 5-6 digit integers or floating-point numbers (integers followed by trailing “.0x” mostly due to incorrect format in Excel). These values are cased to float and then to int to make sure only digits are present. If this fails, null was filled in. For samples with denied labels, values for “Employer Name” was filled in for nulls.
- **VISA_CLASS:** For samples with denied labels that are missing these values, “H-1B” was filled in, which 99.7% of samples have.
- **SECONDARY_ENTITY**: Whether the applicant will be placed in a secondary location. This feature is assumed to be helpful since the majority of Consultancy companies that are believed to outsource software services which have a reputation to flood the visa processing.
- **AGENT_REPRESENTING_EMPLOYER**: If another firm is representing the employer and its application. We plan to model this feature to see if an agency has high rejection rates as compared to another.
- **CONTINUED_EMPLOYMENT**: If this is a re-new visa application.

- **CHANGE_PREVIOUS_EMPLOYMENT**: If an application will continue without changes in job duties NEW_CONCURRENT_EMPLOYMENT: If the applicant will have an additional employer.
- **CHANGE_EMPLOYER**: If an applicant will get the visa with a new employer.
- **AMENDED_PETITION**: If an applicant will work with the same employer with changes in duties.
- **FULL-TIME POSITION:** For samples with the denied label, "N" was filled in for nulls.
- **H1B_DEPENDENT:** For samples with the denied label, "N" was filled in for nulls.
- **PREVAILING_WAGE:** For consistency, the wage for each year was corrected to yearly wage using "PW Unit of Pay", assuming a 40-hour/week shift for 4 weeks, 12 months. Also, all wages before 2018 were adjusted to today's wage by applying the inflation rate. For samples with the denied label, the median value was filled in for nulls.
- **PW_WAGE_SOURCE:** prevailing wage source (OES, CAB, DBA, SCA, etc). For samples with the denied labels, "Other" was filled in.
- **WILLFUL_VIOLATOR:** For samples with the denied label, "N" was assigned for nulls.
- **WAGE_UNIT_OF_PAY:** For samples with the denied label, "Year" was filled in, which 95% of samples have.
- **WAGE_LOWER_THAN_PREVAILING_WAGE:** The derived field is discussed in the feature engineering subsection. "False" is assigned for samples with denied labels and missing wages.
- **TOTAL_WORKER_POSITIONS**: Total amount of workers in the company filing the application.

FEATURE ENGINEERING -

First of all, let's categorize the features which have many unique values into different bins based on some common keywords. So let's transform SOC_TITLE, JOB_TITLE, and EMPLOYER_NAME.

SOC_TITLE:

```
df1.OCCUPATION.value_counts(dropna = False)
```

```
Computer Occupations          388873
Others                         71634
Architecture & Engineering     57176
Financial Occupation           23297
Medical Occupations            10905
Management Occupation          10480
Advance Sciences                8712
Education Occupations           6957
Administrative Occupation       3329
Business Occupation             1695
Mathematical Occupations         522
Marketing Occupation             190
Name: OCCUPATION, dtype: int64
```

JOB_TITLE :

```
df1.JOB_TITLE_NEW.value_counts(dropna = False)
```

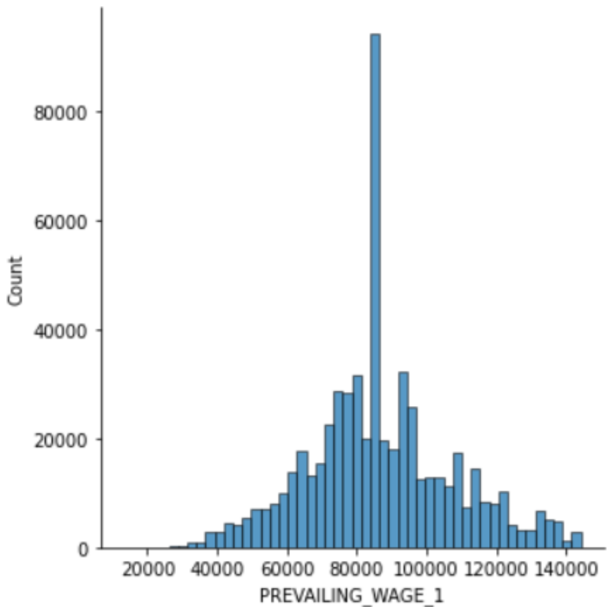
```
IT & SOFTWARE ENGINEERS                        222879
SENIOR TEAM                                     91655
others                                          71135
Manager & DIRECTORS                             45663
BUSINESS TEAM                                   31233
DATABASE & SCIENTISTS                           24256
MECHANICAL & CIVIL ENGINEER                     23990
ARCHITECT                                       17922
EDUCATIONAL ORGANISATION                        17906
ELECTRONICS & ELECTRONICS ENGINEERS TEAM        13001
MEDICAL TEAM                                     9247
MARKETING TEAM                                   7111
FINANCE TEAM                                     6203
LAW TEAM                                         1569
Name: JOB_TITLE_NEW, dtype: int64
```

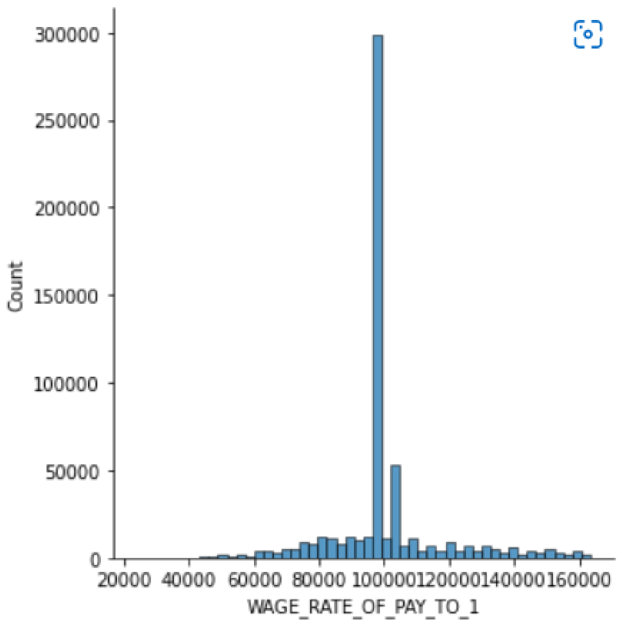
EMPLOYER_NAME :

```
df1.EMPLOYER_BRANCH.value_counts(dropna = False)
```

```
TECH SOLUTIONS                     276869
others                             198802
CONSULTING COMPANIES                54101
TOP TECH                            39148
FINANCE AND MEDICAL SOLUTIONS       13196
ELECTRONIC & LOGISTICS SERVICES     10492
RESEARCH LABS & NETWORK              6928
AUTOMOTIVE & ELECTRICAL              5226
BANKING COMPANIES                    4517
PRODUCT &ENTERPRISE COMPANIES        4383
UNIVERSITY                           4023
BUSINESS SOLUTIONS                   3782
Name: EMPLOYER_BRANCH, dtype: int64
```

Now, let's check the distribution of numeric features and if we find some outliers or the distribution is skewed then we need to fix this by using the Inter-Quantile Range Method.





Now let's check the other features we have converted the values of "FULL_TIME_POSITION", "AGENT_REPRESENTING_EMPLOYER", "SECONDARY_ENTITY_1", "H1B_DEPENDENT", "WILLFUL_VIOLATOR" from "Y" to 1 and from "N" to 0.

And then fill out all the null values by using mode also we converted the values of "NEW_CONCURRENT_EMPLOYMENT", "CHANGE_PREVIOUS_EMPLOYMENT", “TOTAL_WORKER_POSITIONS” and "AMENDED_PETITION" to ‘0’, ‘1’, ‘>1’.

Now, let’s fix ‘CONTINUED_EMPLOYMENT’ by replacing some unwanted values with valid numbers.

```
df1['CONTINUED_EMPLOYMENT'] = df1['CONTINUED_EMPLOYMENT'].replace(['001','01'],'1')
df1['CONTINUED_EMPLOYMENT'] = df1['CONTINUED_EMPLOYMENT'].replace(['00'],'0')
df1['CONTINUED_EMPLOYMENT'] = df1['CONTINUED_EMPLOYMENT'].replace(['02'],'2')
df1['CONTINUED_EMPLOYMENT'] = df1['CONTINUED_EMPLOYMENT'].replace(['B'],'1')
df1['CONTINUED_EMPLOYMENT'] = df1['CONTINUED_EMPLOYMENT'].replace(['0',],'1')
df1['CONTINUED_EMPLOYMENT'].value_counts()
```

Finally, after cleaning and transforming all the features, let’s encode them to perform modeling. All selected categorical features were label encoded using LabelEncoder() function from pandas further, we need to scale the numeric features “Prevailing_Wage_1” and “Wage_Rate_From_To_1” using StandardScaler().

MODELLING -

STEP - 1:

So w.r.t our WebApp we divided our dataset into 2 parts :

1. Based on Employee’s Skillset Information.
2. Based on Employee’s Wage-related Information.

```
df2 = df1[['CASE_STATUS', 'SECONDARY_ENTITY_1', 'AGENT_REPRESENTING_EMPLOYER', 'FULL_TIME_POSITION','WILLFUL_VIOLATOR',
           'NEW_CONCURRENT_EMP', 'CHANGE_PREVIOUS_EMP', 'AMENDED_PETITION_BIN', 'TOTAL_WORKER_POSITIONS_BIN',
           'OCCUPATION', 'JOB_TITLE_NEW', 'EMPLOYER_BRANCH']]
```

```
df3 = df1[['CASE_STATUS', 'WAGE_RATE_OF_PAY_FROM_1', 'WAGE_UNIT_OF_PAY_1', 'PREVAILING_WAGE_1', 'WAGE_RATE_OF_PAY_TO_1']]
df3.head()
```

STEP - 2:

Now let’s split the data into a training set and a test set, in order to train the data and then finally evaluate it on the test set. We’ll choose test_size = 0.20 for both the datasets.

```
y2 = df2.CASE_STATUS
X2 = df2.drop('CASE_STATUS', axis = 1)

seed = 7
test_size = 0.20
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=test_size, random_state=seed)
X_train2.columns
```

```
Index(['AGENT_REPRESENTING_EMPLOYER_N', 'SECONDARY_ENTITY_1_N',
       'FULL_TIME_POSITION_N', 'WILLFUL_VIOLATOR_N', 'NEW_CONCURRENT_EMP_N',
       'CHANGE_PREVIOUS_EMP_N', 'AMENDED_PETITION_N',
       'TOTAL_WORKER_POSITIONS_N', 'OCCUPATION_N', 'JOB_TITLE_N',
       'EMPLOYER_BRANCH_N'],
      dtype='object')
```

```
y3 = df3.CASE_STATUS
X3 = df3.drop('CASE_STATUS', axis = 1)

seed = 7
test_size = 0.20
X_train3, X_test3, y_train3, y_test3 = train_test_split(X3, y3, test_size=test_size, random_state=seed)
X_train3.columns
```

```
Index(['WAGE_RATE_OF_PAY_FROM_1', 'PREVAILING_WAGE_1', 'WAGE_RATE_OF_PAY_TO_1',
       'WAGE_UNIT_OF_PAY_1_N'],
      dtype='object')
```

STEP - 3:

Using the cleaned data, now let's implement the aforementioned models with default parameters on these 2 datasets. As a first attempt, the dominant class (Certified) was used without any resampling,

LOGISTIC REGRESSION

```
LR2 = LogisticRegression()
LR2.fit(X_train2, y_train2)
y_pred2 = LR2.predict(X_test2.to_numpy())
```

```
print(confusion_matrix(y_test2, y_pred2))
print(classification_report(y_test2, y_pred2))
```

```
[[     0    999]
 [     0 114821]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       999
           1       0.99      1.00      1.00    114821

    accuracy                           0.99    115820
   macro avg       0.50      0.50      0.50    115820
weighted avg       0.98      0.99      0.99    115820
```

```
metrics.accuracy_score(y_test2, y_pred2)
```

```
0.9913745467104127
```

```
LR3 = LogisticRegression()
LR3.fit(X_train3, y_train3)
y_pred3 = LR3.predict(X_test3.to_numpy())
```

```
print(confusion_matrix(y_test3, y_pred3))
print(classification_report(y_test3, y_pred3))
```

```
[[     0    999]
 [     0 114821]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       999
           1       0.99      1.00      1.00    114821

    accuracy                           0.99    115820
   macro avg       0.50      0.50      0.50    115820
weighted avg       0.98      0.99      0.99    115820
```

```
metrics.accuracy_score(y_test3, y_pred3)
```

```
0.9913745467104127
```

As we can clearly see that the accuracy of the model is excellent but on the other hand, if we evaluate the classification report and confusion matrix, we find that the f1-score w.r.t the majority class is 1.00 whereas the f1-score w.r.t the minority class is just 0. So we can deduce that these models completely overfit the majority class as the dataset is hugely imbalanced. Hence, we need to apply some resampling methods to balance the data in terms of both classes.

STEP - 4:

Let's oversample both the datasets using SMOTE (This is a type of data augmentation for the minority class and is referred to as the **Synthetic Minority Oversampling Technique**, or **SMOTE** for short.)

We'll be taking sampling_strategy = 0.5 so that we can oversample the data to the point where the count of the minority class is half of the majority class.

OVERSAMPLING USING SMOTE

```
from imblearn.over_sampling import SMOTE
```

```
sm2 = SMOTE(sampling_strategy = 0.5)
X2, y2 = sm.fit_resample(X_train2, y_train2)
```

```
sm3 = SMOTE(sampling_strategy = 0.5)
X3, y3 = sm.fit_resample(X_train3, y_train3)
```

STEP - 5:

Now that we have oversampled the data, let's apply the models again to these resampled datasets.

LOGISTIC REGRESSION

```
LR2 = LogisticRegression()
LR2.fit(X2, y2)
y_pred2 = LR2.predict(X_test2)
```

```
print(confusion_matrix(y_test2, y_pred2))
print(classification_report(y_test2, y_pred2))
```

```
[[   755    244]
 [61759 53062]]
              precision    recall  f1-score   support

           0       0.01      0.76      0.02       999
           1       1.00      0.46      0.63    114821

    accuracy                           0.46    115820
   macro avg       0.50      0.61      0.33    115820
weighted avg       0.99      0.46      0.63    115820
```

```
metrics.accuracy_score(y_test2, y_pred2)
```

```
0.4646606803660853
```

```
LR3 = LogisticRegression()
LR3.fit(X3, y3)
y_pred3 = LR3.predict(X_test3)
```

```
print(confusion_matrix(y_test3, y_pred3))
print(classification_report(y_test3, y_pred3))
```

```
[[   486    513]
 [43653 71168]]
              precision    recall  f1-score   support

           0       0.01      0.49      0.02       999
           1       0.99      0.62      0.76    114821

    accuracy                           0.62    115820
   macro avg       0.50      0.55      0.39    115820
weighted avg       0.98      0.62      0.76    115820
```

```
metrics.accuracy_score(y_test3, y_pred3)
```

```
0.6186668969089967
```

From the above models, we can see that the accuracy of both the models is significantly low but the recall and f1-score are pretty decent w.r.t the previous results.

Now let's apply RandomForest Classifier to these datasets.

```
RandomF2 = RandomForestClassifier(n_estimators = 100)
RandomF2.fit(X2, y2)
y_pred = RandomF2.predict(X_test2)
```

```
print(confusion_matrix(y_test2, y_pred2))
print(classification_report(y_test2, y_pred2))
```

```
[[   693    306]
 [56244 58577]]
              precision    recall  f1-score   support

           0       0.01      0.69      0.02       999
           1       0.99      0.51      0.67    114821

    accuracy                           0.97    123315
   macro avg       0.58      0.50      0.50    123315
weighted avg       0.94      0.97      0.95    123315
```

```
metrics.accuracy_score(y_test2, y_pred)
```

```
0.9652029355715038
```

```
RandomF3 = RandomForestClassifier(n_estimators = 100)
RandomF3.fit(X3, y3)
y_pred3 = RandomF3.predict(X_test3)
```

```
print(confusion_matrix(y_test3, y_pred3))
print(classification_report(y_test3, y_pred3))
```

```
[[   201    798]
 [  8486 106335]]
              precision    recall  f1-score   support

           0       0.02      0.20      0.04       999
           1       0.99      0.93      0.96    114821

    accuracy                           0.92    115820
   macro avg       0.51      0.56      0.50    115820
weighted avg       0.98      0.92      0.95    115820
```

```
metrics.accuracy_score(y_test3, y_pred3)
```

```
0.9198411327922639
```

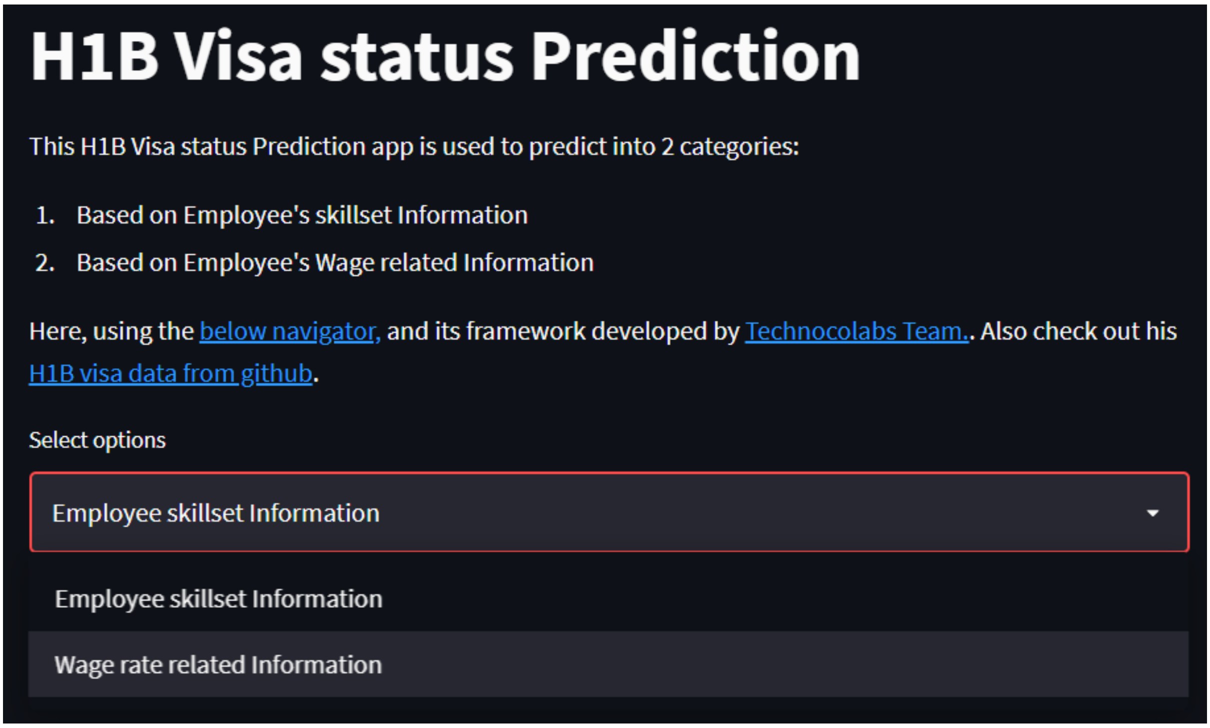
MODEL DEPLOYMENT -

As we can see in the above reports provided by the Random Forest Model, the accuracy seems better than Logistic Regression and the recall and f1-score are also not too low. Now that we've finalized the models for both datasets, let's create interfaces for them. For this, we'll use **Streamlit** (It is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time). We'll use MultiApp Function present in Streamlit to join the 2 pages.

Here is the implementation of that code:

```
import streamlit as st
from multiapp import MultiApp
from apps import app_with_wage_info, app_without_wage_info # import your app modules here

app = MultiApp()

st.markdown("""
# H1B Visa status Prediction

This H1B Visa status Prediction app is used to predict into 2 categories:

1. Based on Employee's skillset Information
2. Based on Employee's Wage related Information

Here, using the [below navigator,](https://github.com/upraneelnihar/streamlit-multiapps) ar

""")

# Add all your application here

app.add_app("Employee skillset Information", app_without_wage_info.app)
app.add_app("Wage rate related Information", app_with_wage_info.app)

# The main app
app.run()
```

Now, let's see how our Web App looks like:



On clicking **Select Options,** we can select 2 options:

1. Employee skillset Information
2. Wage rate-related Information

Based on the user's choice, he'll be redirected to the selected page. Finally, we deployed our web app on **Heroku** (It is a cloud platform as a service supporting several programming languages).

IV. CONCLUSION -

We created and evaluated a supervised classifier that predicts H-1B visa case status, based on the texts entered in the application. We started by defining the problem statement and identified which features are present in the data, and explored the dataset by making use of a few helpful plots to find the distribution and trend of each feature.

We found the logistic model to be predicting high accuracy but hides the true negatives as it tries to fit the data. So, we believe that visa outcome is not as dependent on employer and job profile as we presumed in our null hypothesis, it has an element of random behavior in the decision. We captured individual company names, job titles, and job categories to see if they are useful measures in modeling the accuracy. The result was a drop in total accuracy but a higher level of predicting true negatives. We also evaluated random forests through the same process, then defined a benchmark, and implemented the initial versions of both the models, all of which were unsuccessful due to highly imbalanced data. After balancing out samples from both classes, We evaluated results from the models, tuned parameters for a more accurate result, and picked the final models based on the evaluation score and feasibility. And then finally deployed the model using Streamlit and Heroku.

REFERENCES -

https://github.com/HellBrazer/H1-B-Visa-Prediction-WebApp

https://h1bvisapredictionapp.herokuapp.com/

https://www.youtube.com/playlist?list=PLtqF5YXg7GLmCvTswG32NqQypOuYkPRUE

https://www.youtube.com/playlist?list=PLqYFiz7NM_SN2ZbhnbfwG4kTZ6oCh0aOM

<u>TEAM-A MEMBERS</u> -

ARPIT KUMAR LARIYA (TEAM LEAD)

RISHABH GOYAL

SHUBHANGI VAJPAI

AMRATANSHU VARSHNEY

NITISH KUMAR S J

<u>SUPERVISED BY</u> -

YASIN SHAH

<u>MENTOR</u> -

KARISHMA KUNWAR