H-1B Visa Approval Rate in the United States Analysis and Prediction using Machine Learning

1st Hao Yu
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, United States
hay024@ucsd.edu, A14050325

2nd Kai Chuen Tan
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, United States
kctan@ucsd.edu, A59011493

***Abstract*—H-1B visa is a temporary non-immigrant visa that allows highly-skilled foreign workers to work with U.S. employers legally, and H-1B visas play an important role in building and shaping the U.S. economy. The ever-increasing number of H-1B applicants every year and more imposed non-immigrant visa restrictions in the U.S. have significantly impacted highly-skilled foreign workers' American Dreams negatively to work long term in the U.S. As a result, the uncertainty of getting an H-1B for foreign workers and graduated international students in the U.S. rises, and it causes uncertainty in foreign workers' legal status in the U.S. Therefore, in this project, statistical methods are applied to perform pre-processing on the 2019 H-1B data-set from the U.S. Department of Labor website, and several machine learning and classifications techniques are implemented including Naive Bayes and K-Nearest Neighbor (KNN) Classifications to determine the correlation between the H-1B approval rate and features of several major categories. The project aims to help international students better plan which professions to select to maximize their probability of getting an H1-B visa sponsorship based on their field of study and interests.**

***Index Terms*—H-1B Visa, H-1B Pre-processing Data, Naive Bayes Classification, K-Nearest Neighbor (KNN) Classification, Machine Learning**

I. Background and Introduction

During the pre-COVID 19 eras, the United States (U.S.) non-immigrant visa restrictions and rules had made job-searching more challenging for international students, who studied and graduated from their United States institution [1]. The recent COVID-19 pandemic not only worsen the chances of a graduated international student from receiving a full-time job opportunity in the U.S. but also shattered the American dreams of highly-skilled foreign workers with an expiring H-1B visa due to the suspension of H-1B issuance by the Trump's administration in early July 2020 [2]. After the current U.S. President, Joe Biden won the 2020 U.S. Presidential Election and took over the office, the H-1B visa suspension was lifted in late March 2021, and employers could hire highly-skilled foreign workers again [3].

To make the college education spending worthwhile, most of the international graduates not only wanted to receive a world-class tertiary education and experience the diversity and culture in the U.S. but also eager to receive certain fields of job opportunities that might not be available in their home country and to gain valuable long-term work experience in the U.S. Nevertheless, a temporary non-immigrant, employment-based working visa, H-1B, is required for international graduates to work legally in the U.S. for a long term, and the annual H-1B visa limit is only 85,000 [4]. Out of 85,000 H-1B visas, 20,000 H-1B visas are reserved for applicants who are holding a master's degree or higher, and the remaining 65,000 H-1B visas are for any applicants including those who are holding an advanced degree; among 65,000 H-1B visas, 6,800 H-1B visas (i.e., H-1B1 Chile, and H-1B1 Singapore) are reserved for 1,400 applicants from Chile and 5,400 applicants from Singapore, respectively, and 10,500 H-1B visas (i.e., E-3 Australian) are reserved for applicants from Australia [5]. If the H-1B1 Chile, H-1B1 Singapore, and E-3 Australian are not used up in a particular year, they will be added back to the 65,000 H-1B visas cap. In order to apply for an H-1B visa, a foreign worker must be hired by an employer in the U.S., and the employer must be willing to sponsor the foreign worker an H-1B visa by submitting an H-1B visa petition to the U.S. Immigration Department.

The purpose of the "*H-1B Visa Approval Rate in the United States Analysis and Prediction using Machine Learning*" project is to predict the probability of an international graduate or a foreign worker to get an H1-B visa application approved based on the degree-level completed, annual salary level, state of the worksite, job title, full-time position status, nationality, and company where the H1-B applicant works. Statistical methods will be applied to perform pre-analysis on the data, and several machine learning techniques will be implemented to determine the correlation between the result and features of each case. Eventually, the project would be able to help international students better plan which professions to select to maximize their probability of getting an H1-B visa sponsorship based on their field of study and interests.

II. Data-Set Description

The year 2019 H-1B data-set was downloaded from the U.S. Department of Labor (DOL) official website [6], and the data-set has 664,616 rows, which represents the total number of H-1B visa applicants, and 260 columns, which are known as categories including the case status, visa class, job title, etc. In this project, 8 out of 260 categories are selected from the data because the other categories are redundant like the

decision date, prevailing wage tracking number, preparer last name, etc. Hence, the 8 categories that are focused on in this project are case status, visa class, Standard Occupational Classification (SOC) title, full-time position status, employer name, first worksite state, pay wage level, and statutory basis.

III. Data Analysis and Visualization

The 9 main categories from the H-1B data-set as stated in the Data Description section are analyzed to gain some valuable insights from the data.

H-1B Visa Case Status Analysis: H-1B Case Status presents the outcome of an H-1B Visa application. The feature is extracted from the data-set, and there are 4 different possible classes or outcomes, which are Certified, Certified Withdrawn, Denied, and Withdrawn. In this project, only two main classes are taken into account, which are Certified and Denied, to do binary classification instead of multi-class classification. Hence, the two other classes (i.e., Certified-Withdrawn and Withdrawn) are converted into one of the two main classes as shown in **Fig. 1**. Certified-Withdrawn and Withdrawn are considered as Denied in this project because they are not considered as a fully approved H-1B application.

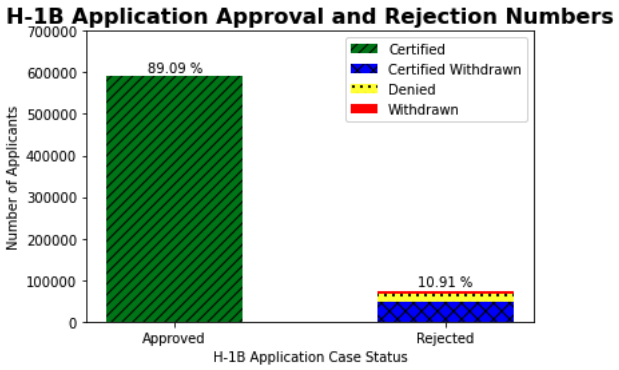


Fig. 1: Numbers of Certified and Denied H-1B Applications

Fig. 1 illustrates that 89.09 %-symbol of the 664,616 H-1B applications are certified, which are approximately 592,106, and only 10.91 %-symbol H-1B applications are denied in year 2019. Based on the stacked bar chart in **Fig. 1** above, H-1B applicants who filed and submitted their H-1B application, have a significantly high chance of getting their H-1B application certified.

H-1B Visa Class Analysis: H-1B Visa Class presents the type of H-1B visas that is submitted by an applicant for processing. As mentioned in I, H-1B Visas are divided into 4 different classes, which are the general H-1B, E-3 Australian, H-1B1 Chile, and H-1B1 Singapore. The percentages of certified and denied H-1B application for each visa class is shown below:

Fig. 2 presents that all the different H-1B visa classes have approximately the same probability of getting the H-1B visa certified, which are around 86.0 %-symbol to 89.0 %-symbol, although the general H-1B visa class has the highest number of certified H-1B visa (i.e., 578,640), and the H-1B1 Chile has

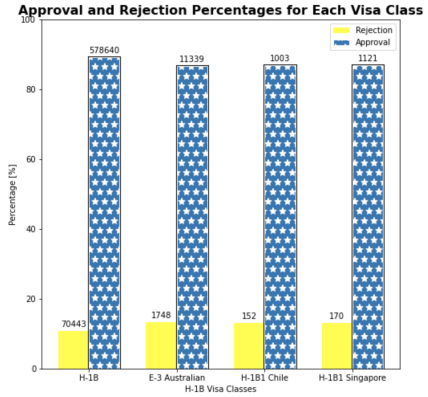


Fig. 2: Different H-1B Visa Classes' Certified and Denied H-1B Visas Percentages

the lowest number of certified H-1B visa (i.e., 1,003). Hence, the data shows that there is no bias towards any visa class during the H-1B application process.

Worksite State Analysis: The Worksite State category from the H-1B visa application dataset presents the state in the U.S. where the H-1B applicant works. This category data information is crucial to analyze the number of H-1B applicants in each state in the U.S. because it provides insight about which states have more employers that are willing to sponsor H-1B visas for foreign workers as shown in the treemap below:

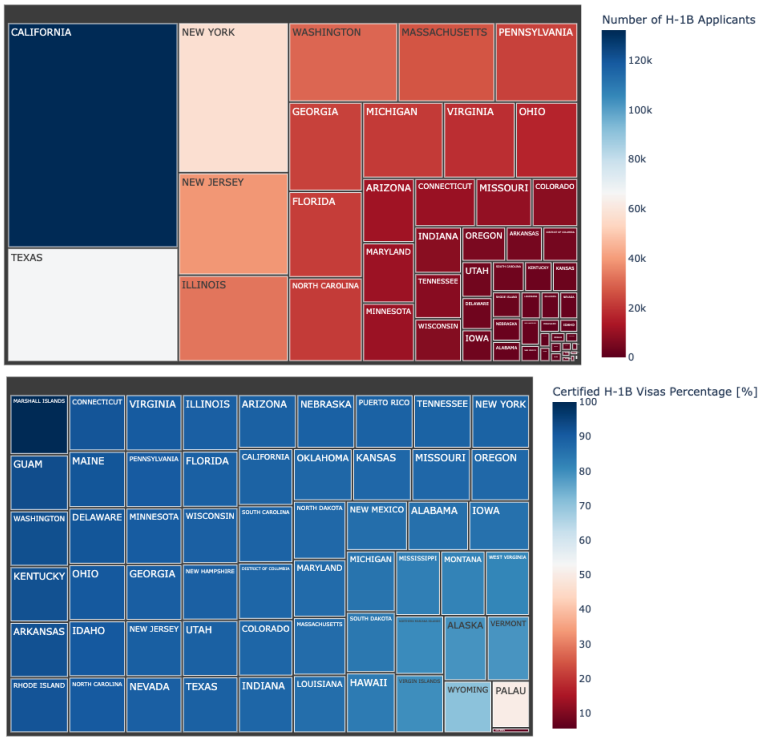


Fig. 3: Number of H-1B Applicants in Different States (Top) and Percentage of Certified H-1B in Different States(Bottom)

According to **Fig. 3** above, although California, Texas, and New York are the top three states in the U.S., where employers are willing to sponsor H-1B visas to highly-skilled foreign workers, most of the states have more than 80 %-symbol of getting an H-1B visa certified except for Alaska, Wyoming, Vermont and Palau states as illustrated in **Fig. 3**. The main reason that California state has the highest number of H-1B applicants is that majority of the Information Technology (I.T.) companies and the big tech companies like Facebook, Amazon, Apple, Netflix, Google, Tesla, and Microsoft Corporation are located there. Besides California, Texas and New York are second and third popular states, respectively, for highly-skilled foreign workers to work in the U.S.

Employer Name Analysis: Since knowing which employers hired and sponsored the most H-1B Visas for foreign workers are important to analyze, the employer name category information is extracted from the H-1B visa application dataset.

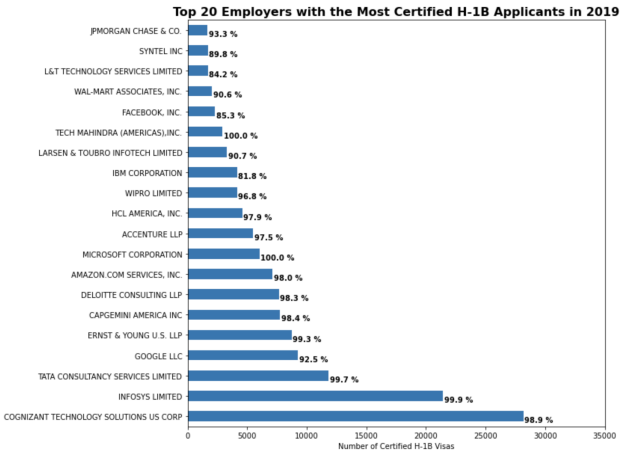


Fig. 4: Top 20 Companies that Sponsored the Most H-1B Visas

Fig. 4 shows that most of the employers who sponsored H-1B visas for foreign workers are I.T. companies like Google LLC, Amazon.com Services, Inc., Microsoft Corporation, Facebook, Inc., etc., and the top 3 employers are multinational consulting firms, which are Cognizant Technology Solutions US CORP, INFOSYS Limited, and TATA Consultancy Services Limited. Besides that, the H-1B visa application approval rate for the top 20 employers have more than 81.0 %-symbol probability of getting an H-1B visa application certified; Microsoft Corporation and Tech Mahindra (Americas), Inc. have all of their H-1B visa applications certified in the year 2019.

Job Position Type Analysis: Job position type presents the information about whether the foreign worker will be working as a full-time or a part-time employee for the employer who sponsored the H-1B visa. The data provides insight about whether U.S. employers are preferred to sponsor H-1B visas for full-time or part-time employees.

Fig. 5 above illustrates 98.3 %-symbol (i.e., 653,376 applicants) of the H-1B applications are working as a full-time

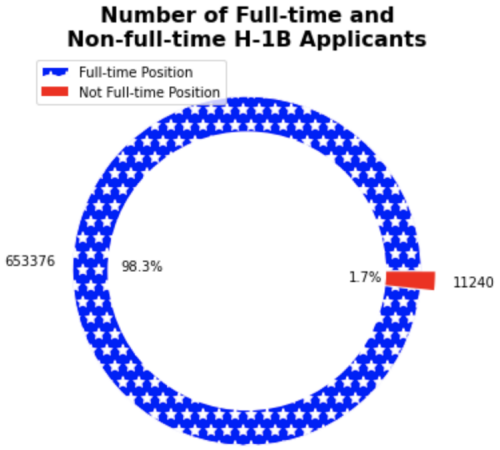


Fig. 5: Number of Full-time and Part-time H-1B Applicants

employee, and only 1.7 %-symbol (i.e., 11,240 applicants) of the H-1B applications are for the part-time positions. Hence, more employers preferred to sponsor H-1B visas for full-time employees rather than part-time employees.

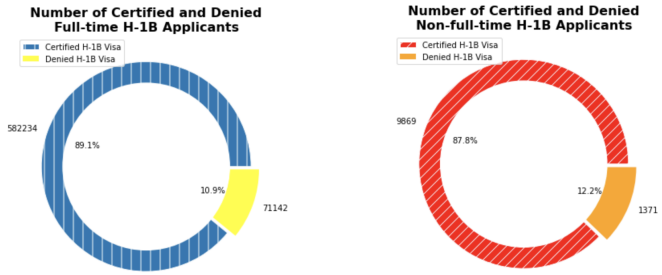


Fig. 6: Difference Between Full-time and Part-time H-1B Applicants

Although there are more H-1B visa applications for full-time positions than part-time positions, the percentage of certified H-1B visa application for full-time jobs are just 1.3 %-symbol higher than the percentage of certified H-1B visa application for part-time positions, which are 87.8 %-symbol (i.e., 9,869 out of 11,240 part-time H-1B applicants) as shown in **Fig. 6**. Therefore, the job position type does not affect much the probability of getting an H-1B visas application certified.

Prevailing Wage Level Analysis: Prevailing Wage Level is defined as the range of foreign workers' wages paid by the employer, and it is divided into four levels, which are Level I, Level II, Level III, and Level IV. H-1B Wage Level I (entry) annual salary ranges from USD 38k to USD 51k; H-1B Wage Level II (qualified) annual salary ranges from USD 51k to USD 65k; H-1B Wage Level III (experienced) annual salary ranges from USD 65k to USD 75k; H-1B Wage Level IV (Fully Competent) annual salary ranges from USD 78k to USD 90k [7]. The information from this category provides an insight about which Prevailing Wage Level has the most H-1B visa applications, and how it affects the probability of getting the H-1B application certified.

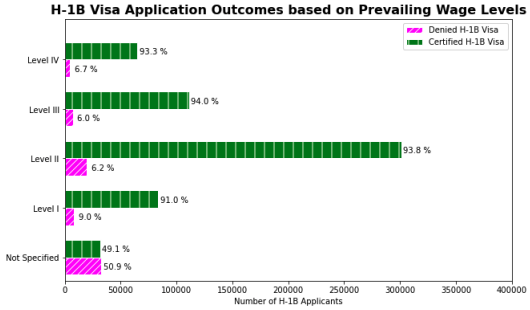


Fig. 7: Relationship between H-1B Case Status and Prevailing Wages Level

Fig. 7 shows that Prevailing Wage Level II has the largest number of certified H-1B applications, which is 301,168, and H-1B applications that do not specify prevailing wage level has the lowest number of certified H-1B visas (i.e., 31,417). As long as the prevailing wage level is specified in the H-1B application, all the prevailing wage levels have more than 90.0 %-symbol of getting an H-1B visa application certified. If the prevailing wage level is not specified in the H-1B visa application, it is more likely to get an H-1B application denied, which is 50.9 %-symbol. Hence, the **Fig. 7** presents that employers are more willing to sponsor H-1B visas for foreign workers who have more work experience, but the work experience or prevailing wage level does not affect the chances of getting an H-1B visa application approved.

Statutory Basis Analysis: Statutory basis presents information about whether the H-1B visa application has either an advanced degree, USD 60k annual salary or higher, both, or none. In this project, we only focused on whether the applicant has an advanced degree or not; hence, "USD 60k Annual Salary" and "None" are classified as an applicant without an advanced degree, and "Advanced Degree" and "Both" are classified as an applicant with an advanced degree

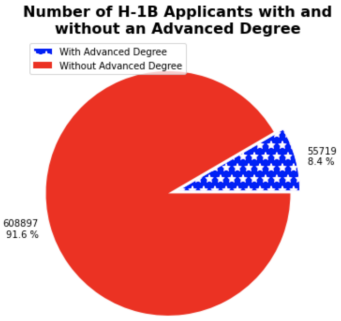


Fig. 8: Number of H-1B Applicants with and without Advanced Degree

Fig. 8 illustrates that the number of H-1B visa applications without an advanced degree is 10.9 times more than the number of H-1B visa applications with an advanced degree. The data also show most of the occupations do not really require an advanced degree to qualify for a job position.

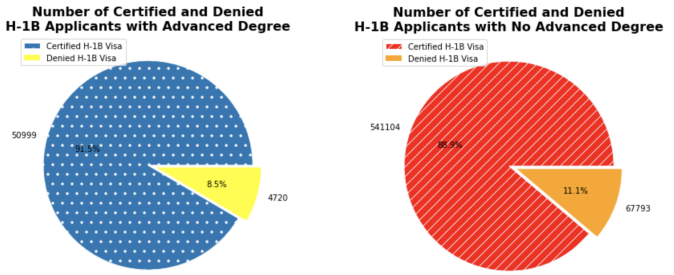


Fig. 9: Difference Between H-1B Applicants with and without Advanced Degree

According to the **Fig. 9**, H-1B applications with an advanced degree just have 2.6 %-symbol higher chances of getting an H-1B visa application certified than H-1B applications without an advanced degree. Nevertheless, both H-1B visa applications with and without an advanced degree have a very probability of getting the H-1B visa application certified, which are 91.5 %-symbol and 88.9 %-symbol, respectively. Therefore, an H-1B visa application with an advanced degree does not significantly improve the chances of getting an H-1B visa application certified.

Standard Occupational Classification (SOC) Title Analysis: Standard Occupational Classification (SOC) Title classifies foreign workers' job titles into a more general occupational category, for example, mechanical design engineer and mechanical systems control engineer are classified as mechanical engineers. SOC title from the data-set provides insight about which occupational categories have the most H-1B visa applications. The top 20 SOC titles with the most certified H-1B visa are filtered and plotted to analyze as shown in the figure below.

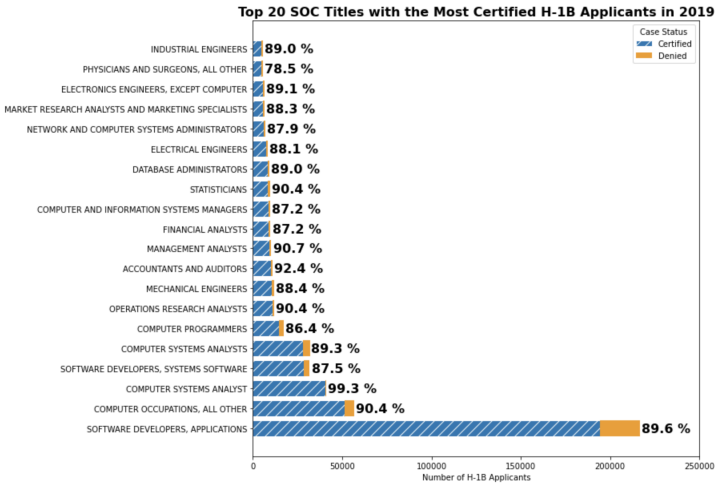


Fig. 10: Top 20 SOC Titles in H-1B Applications

Fig. 10 illustrates that the top 3 occupational categories that have the most H-1B visa applications are "Software Developers, Applications", "Computer Occupations, All Other", and "Computer System Analysis", and the top 20 occupational

categories are mostly in the field of I.T. "Electrical Engineer" is in the top 15 occupational categories with 7,316 certified H-1B visa applications, but it is 96.2 %-symbol less than the top one occupational category (i.e., Software Developers, Applications) number of certified H-1B visa applications (i.e., 194,060 certified H-1B visa applications). Hence, the data presents that there are significantly more job opportunities in the I.T. field for highly-skilled foreign workers. However, most of the top 20 SOC titles have more than 85.0 %-symbol chance of getting an H-1B visa application certified, except for the "Physicians and Surgeons, All Other" occupational category.

IV. Logistical Prediction

After we have finished pre-processing the data, we perform statistical analysis on the data to predict the probability that an international student receives an H-1B visa. Since there are only two outcomes for the event of H-1B visa application (approved or denied), we decided to build logistical models to fit the data we have and utilize those logistical models to make predictions afterwards. In order to check the accuracy of our models, we split the data randomly into training data and testing data by a four over one distribution, and we call them X_{train} and X_{test}, accordingly. We also extract the results of corresponding cases in both X_{train} and X_{test}, can we call them y_{train} and y_{test}.

A. Models

Two models were applied in this project to fit the pre-processed data and make predictions about future possible cases, which are Naïve Bayes classifier model [8] and K-Nearest Neighbour classifier model [9].

1) Naïve Bayes Classifier: The first model implemented is a Naïve Bayes classifier model. Based on the Bayes' theorem [10],

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

a Naïve Bayes classifier is capable of generating strong independence relationships among features. In this particular project, we applied the Naïve Bayes classifier to generate logistical assumptions from the set

$$C = \{\textit{Denied}, \textit{Certified}\}$$

We take another step and mark the word *Denied* as 0, *Certified* as 1, which leads the set of the assumption result to be $C = \{0, 1\}$. In order to fit data of words into the Naïve Bayes classifier, a binary bag of word (BoW) [11] matrix needs to be built. We first collect all the unique words in the attributes of the selected features, and store them in a set D, where d denotes the size of the set. We then treat features in each case as words in an independent sentence, and we represent features in each case by a binary vector x_n, where

$$x_n \in \{0, 1\}^d$$

here, $x_n[d] = 1$ means the n-th element in X_{train} contains the d-th unique word in the word set D, and $x_n[d] = 0$ means the n-th element in X_{train} does not contain the d-th unique word in the word set D.

After we have presented every case in X_{train} as a d-dimensional binary vector, we then build the Naïve Bayes classifier and train it. To achieve this, we first build a $2\ by\ d$ matrix filled with zero and called it S. After that, we loop through all cases in X_{train} and create a d-dimensional binary vector, x, for each case. For each x, we get the corresponding y_{train} data, Y for that particular case and updated the matrix S we just created

$$S[Y, i] = S[Y, i] + x[i],\ for\ i \in \{0, 1, 2, ..., d-1\}$$

After we have finished updating the matrix S using X_{train} and y_{train}, we applie the maximum likelihood estimation (MLE) method to predict result for cases in X_{test} to get the analysis the accuracy of the model we built [CITE SOMETHING]. For each case in X_{test}, we converte it to a d-dimensional vector as we did for cases in X_{train}, and we call it x. For each x, we calculate the prediction result y_{pred} by

$$y_{pred} = \underset{C=\{0,1\}}{\operatorname{argmax}} \prod_{i=1}^{d} p(x[i] \mid Y = C)\ p(Y = C)$$

$$= \underset{C=\{0,1\}}{\operatorname{argmax}} \prod_{i=1}^{d} S[Y, i]\ p(Y = C)$$

However, some elements in x had a numerical value of 0, which means that particular case didn't contain a unique word in the set D, and it would lead y_{pred} to zero. To solve this problem, we calculate log of the expression of y_{pred} instead of y_{pred} itself

$$y_{pred} = \underset{C=\{0,1\}}{\operatorname{argmax}}\ \log \prod_{i=1}^{d} S[Y, i]\ p(Y = C)$$

$$y_{pred} = \underset{C=\{0,1\}}{\operatorname{argmax}} (\sum_{i=1}^{d} \log S[Y, i]\ + \log p(Y = C))$$

The value y_{pred} illustrates the result of prediction for a particular case, x, in X_{test}. By performing the same procedure to every case in X_{test}, we predict the results for all cases in X_{test} and collect the prediction results.

2) K-Nearest Neighbours Classifier: The second model implemented is a K-Nearest Neighbour (KNN) classifier model. The KNN algorithm is a data classification method that predicts which group a data point will become a member of based on the distance between that data point and k nearest data points in the available groups. Different from Naïve Bayes classifier, KNN is a lazy classification algorithm, which

means training doesn't take place while we implement KNN algorithm. However, the initial infrastructure processes for data using Naïve Bayes and KNN algorithms are similar. For both algorithms, we create the BoW, split the raw data into training and testing sets similar to what we did for when we implemented the Naïve Bayes classifier. After getting X_{train}, y_{train}, X_test, y_test, for each case in X_{set}, we get the binary vector, x_d, and we calculate the euclidean distances between x_d and all binary vectors in X_{train}. We sort the calculated distances, find k smallest distances and their corresponding cases. We record the logistical results of those cases, and vote the most common result. For example, if we choose k to be 3. Then for a random x_d in X_{set}, we will find the 3 cases that have the shortest distance from x_d. Assuming the three generated results are "$Denied$","$Accepted$","$Accepted$". then the prediction for x_d will be "Accepted". We perform the above action for every case in the test set, and get the y_{pred}, and we evaluate the accuracy of prediction by comparing y_{pred} and y_{test}.

B. Prediction

After performing the Naive Bayes analysis on the pre-processed data, the accuracy for predicting was calculated to be 87.90 %-symbol. The confusion matrix and table of conditional probability for prediction are shown below.

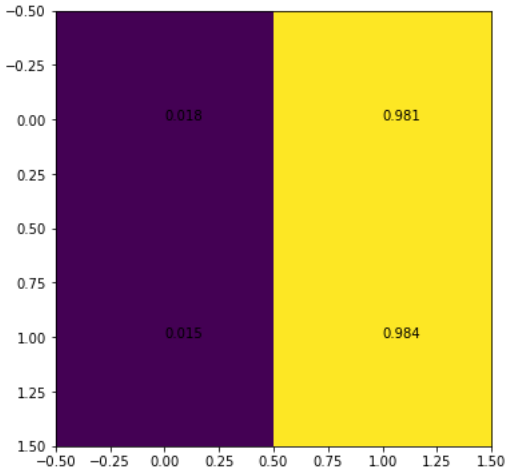


Fig. 11: Confusion Matrix for Naive Bayes Classification

Fig. 11 illustrates the confusion matrix for the result of prediction using the Naive Bayes classifier. The rows represent the ground truth result, where the first row indicates "Denied", and the second row indicates "Accepted". The columns represent the prediction result, where the first column shows "Denied", and the second column shows "Accepted". The number in the figure means the probability of truth/false - positive/negative. For example, the number in second row second column, 0.984, tells the probability that 98.4 %-symbol cases that are labeled "Accepted" also get predicted to be "Accepted" in this model.

	Predicted Denied	Predicted Accepted
True Denied	0.02	0.98
True Accepted	0.02	0.98

Fig. 12: Conditional Probability for Predicting Results

Fig. 12 represents the conditional probability explicitly for the Naive Bayes model.

Accuracy of applying the KNN algorithm was figured out to be 88.13 %-symbol when k is defined to be 5. The confusion matrix and table of conditional probability for prediction are shown below.

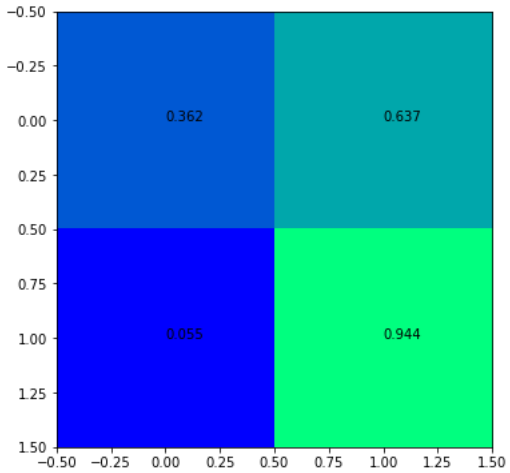


Fig. 13: Confusion Matrix for K-Nearest Neighbours Classification

Fig. 13 illustrates the confusion matrix for the result of prediction using the KNN classifier. The rows represent the ground truth result, where the first row indicates "Denied", and the second row indicates "Accepted". The columns represent the prediction result, where the first column shows "Denied", and the second column shows "Accepted". The number in the figure means the probability of truth/false - positive/negative.

	Predicted Denied	Predicted Accepted
True Denied	0.36	0.64
True Accepted	0.06	0.94

Fig. 14: Conditional Probability for Predicting Results

Fig. 14 represents the conditional probability explicitly for the KNN model.

Fig. 15 illustrates the correlation between the selected features and the application result. In this figure, the greenish colors indicate high correlation between subjects, and reddish colors indicate low correlation between subjects.

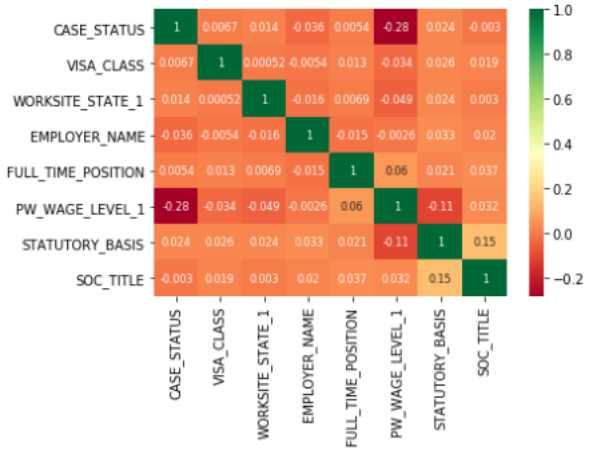


Fig. 15: Correlation Between the Selected Features and Ground Truth Results

C. Discussion of Prediction Results

In order to find the optimal number for the number of neighbours, k, multiple k was experimented, including 1, 3, 5, 7, and 9. Among the results produced, $k = 5$ provided the optimal accuracy. However, values of accuracy using other k are not ideal. For example, accuracy of prediction when k was defined to be 3 was found to be 41.02 %-symbol, which was assumed to be unacceptable.

Even though the accuracy found by applying the optimal k for the KNN algorithm, 88.13 %-symbol, leaded to a better result than the result generated by the Naive Bayes classifier, 87.90 %-symbol, it is not larger than 1 %-symbol, thus the advantage is considered to be trivial. By observing the conditional probability, we found that both models have a strong bias towards generating results labeled as "Approved". One possible explanation is this data set contains significantly more cases labeled as "Approved" than "Denied". By analyzing the raw data, only 10.85 %-symbol cases were labeled as "Denied", which indicated that number of cases labeled as "Approved" is 9 times the number of cases labeled as "Denied". Another possible explanation is there are no strong correlations between the features we selected and the results. From Fig. 15, we can tell that the correlation between subjects we selected and the final results are small. Two assumptions were made to explain the low correlations. Perhaps we didn't select the most outstanding features, perhaps results of applications of H-1B Visa was randomly distributed.

V. Conclusion

H-1B Visa is a significant component of the U.S. economy, and it has strong effect on international students who have career plans of working in the U.S. after graduation. This project analyzed on the 2019 H-1B cases and it focused on predicting the probability of getting certified for entering the H-1B lottery based on some features that are assumed to play an important role in the application. As a conclusion, even though this project failed find a dominant feature that will effect greatly effect the result of applications, two models used for predicting the results achieved an accuracy close to 90 percent. In addition, visualization on the raw data shows that the probability of getting certified was 89 percent. Combining the two results, this project would encourage any international students who desires to start a career in the U.S. to apply for the H-1B Visa due to its high certified probability and low bias towards a specific field.

References

[1] M. Krislov, "Why international students are good for colleges, Universities and America," Forbes, 22-Mar-2019. [Online]. Available: https://www.forbes.com/sites/marvinkrislov/2019/03/22/why-international-students-are-good-for-colleges-universities-and-america/?sh=612fb786f496. [Accessed: 23-Nov-2021].

[2] V. Gewin, "The visa woes that Shattered Scientists' american dreams," Nature News, 28-Sep-2020. [Online]. Available: https://www.nature.com/articles/d41586-020-02746-y. [Accessed: 23-Nov-2021].

[3] J. Fabian and G. Douglas, "Biden to Let Trump's H-1B Visa Ban Expire in Win for Tech," Bloomberg.com, 30-Mar-2021. [Online]. Available: https://www.bloomberg.com/news/articles/2021-03-30/biden-to-let-trump-s-h1-b-visa-ban-expire-in-win-for-tech-firms. [Accessed: 23-Nov-2021].

[4] S. Anderson, "2021 might be a decisive year for H-1B Visas," Forbes, 02-Jun-2021. [Online]. Available: https://www.forbes.com/sites/stuartanderson/2021/06/02/2021-might-be-a-decisive-year-for-h-1b-visas/?sh=1d708d6118df. [Accessed: 23-Nov-2021].

[5] "H-1B, H-1B1 and E-3 Specialty (Professional) Workers," United States Department of Labor. [Online]. Available: https://www.dol.gov/agencies/eta/foreign-labor/performance. [Accessed: 25-Nov-2021].

[6] "Performance data," United States Department of Labor. [Online]. Available: https://www.dol.gov/agencies/eta/foreign-labor/performance. [Accessed: 25-Nov-2021].

[7] Yekrangi amp; Associates. (2021, April 9). What are the wage levels for H1-B visa workers? Yekrangi; Associates. Retrieved December 4, 2021, from https://www.yeklaw.com/blog/2021/april/what-are-the-wage-levels-for-h1-b-visa-workers-/.

[8] McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." AAAI-98 workshop on learning for text categorization. Vol. 752. No. 1. 1998.

[9] Guo, Gongde, et al. "KNN model-based approach in classification." OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". Springer, Berlin, Heidelberg, 2003.

[10] Swinburne, Richard. "Bayes' Theorem." Revue Philosophique de la France Et de l 194.2 (2004).

[11] Sethy, Abhinav, and Bhuvana Ramabhadran. "Bag-of-word normalized n-gram models." Ninth Annual Conference of the International Speech Communication Association. 2008.

Jupyter_Notebook_Final_Project_Data_Visualization_and_Preprocessing

December 12, 2021

0.1 Pre-processing Data and Data Visualization

Load data from the excel file and perform pre-processing on the data:

[1]:
```
# Import necessary libraries
import pandas as pd
from math import *
import numpy as np
from copy import deepcopy
from collections import Counter
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

[4]:
```
# File names
file_name = "H-1B_Disclosure_Data_FY2019.xlsx"

# Read and store the data into an array
data = pd.read_excel(file_name)

# Print status
print ("Done loading the excel file")
```

```
Done loading the excel file
```

Determine the number of H-1B visa applications and categories available in this dataset:

[5]:
```
# Display number of H-1B visa applicants
print ("{} H-1B visa applicants are provided in this dataset.".
 ↪format(len(data)))
```

```
664616 H-1B visa applicants are provided in this dataset.
```

[6]:
```
data_columns = data.columns

print ("There are a total of {} catergories in this dataset, which are: \n".
 ↪format(len(data_columns)))
```

```
for i in range(len(data_columns)): print (str(i + 1) + " " + data_columns[i])
```

There are a total of 260 catergories in this dataset, which are:

```
1 CASE_NUMBER
2 CASE_STATUS
3 CASE_SUBMITTED
4 DECISION_DATE
5 ORIGINAL_CERT_DATE
6 VISA_CLASS
7 JOB_TITLE
8 SOC_CODE
9 SOC_TITLE
10 FULL_TIME_POSITION
11 PERIOD_OF_EMPLOYMENT_START_DATE
12 PERIOD_OF_EMPLOYMENT_END_DATE
13 TOTAL_WORKER_POSITIONS
14 NEW_EMPLOYMENT
15 CONTINUED_EMPLOYMENT
16 CHANGE_PREVIOUS_EMPLOYMENT
17 NEW_CONCURRENT_EMPLOYMENT
18 CHANGE_EMPLOYER
19 AMENDED_PETITION
20 EMPLOYER_NAME
21 EMPLOYER_BUSINESS_DBA
22 EMPLOYER_ADDRESS1
23 EMPLOYER_ADDRESS2
24 EMPLOYER_CITY
25 EMPLOYER_STATE
26 EMPLOYER_POSTAL_CODE
27 EMPLOYER_COUNTRY
28 EMPLOYER_PROVINCE
29 EMPLOYER_PHONE
30 EMPLOYER_PHONE_EXT
31 NAICS_CODE
32 AGENT_REPRESENTING_EMPLOYER
33 AGENT_ATTORNEY_LAW_FIRM_BUSINESS_NAME
34 AGENT_ATTORNEY_ADDRESS1
35 AGENT_ATTORNEY_ADDRESS2
36 AGENT_ATTORNEY_CITY
37 AGENT_ATTORNEY_STATE
38 AGENT_ATTORNEY_POSTAL_CODE
39 AGENT_ATTORNEY_COUNTRY
40 AGENT_ATTORNEY_PROVINCE
41 AGENT_ATTORNEY_PHONE
42 AGENT_ATTORNEY_PHONE_EXT
```

43 STATE_OF_HIGHEST_COURT
44 NAME_OF_HIGHEST_STATE_COURT
45 WORKSITE_WORKERS_1
46 SECONDARY_ENTITY_1
47 SECONDARY_ENTITY_BUSINESS_NAME_1
48 WORKSITE_ADDRESS1_1
49 WORKSITE_ADDRESS2_1
50 WORKSITE_CITY_1
51 WORKSITE_COUNTY_1
52 WORKSITE_STATE_1
53 WORKSITE_POSTAL_CODE_1
54 WAGE_RATE_OF_PAY_FROM_1
55 WAGE_RATE_OF_PAY_TO_1
56 WAGE_UNIT_OF_PAY_1
57 PREVAILING_WAGE_1
58 PW_UNIT_OF_PAY_1
59 PW_TRACKING_NUMBER_1
60 PW_WAGE_LEVEL_1
61 PW_OES_YEAR_1
62 PW_OTHER_SOURCE_1
63 PW_NON-OES_YEAR_1
64 PW_SURVEY_PUBLISHER_1
65 PW_SURVEY_NAME_1
66 WORKSITE_WORKERS_2
67 SECONDARY_ENTITY_2
68 SECONDARY_ENTITY_BUSINESS_NAME_2
69 WORKSITE_ADDRESS1_2
70 WORKSITE_ADDRESS2_2
71 WORKSITE_CITY_2
72 WORKSITE_COUNTY_2
73 WORKSITE_STATE_2
74 WORKSITE_POSTAL_CODE_2
75 WAGE_RATE_OF_PAY_FROM_2
76 WAGE_RATE_OF_PAY_TO_2
77 WAGE_UNIT_OF_PAY_2
78 PREVAILING_WAGE_2
79 PW_UNIT_OF_PAY_2
80 PW_TRACKING_NUMBER_2
81 PW_WAGE_LEVEL_2
82 PW_OES_YEAR_2
83 PW_OTHER_SOURCE_2
84 PW_NON-OES_YEAR_2
85 PW_SURVEY_PUBLISHER_2
86 PW_SURVEY_NAME_2
87 WORKSITE_WORKERS_3
88 SECONDARY_ENTITY_3
89 SECONDARY_ENTITY_BUSINESS_NAME_3
90 WORKSITE_ADDRESS1_3

91 WORKSITE_ADDRESS2_3
92 WORKSITE_CITY_3
93 WORKSITE_COUNTY_3
94 WORKSITE_STATE_3
95 WORKSITE_POSTAL_CODE_3
96 WAGE_RATE_OF_PAY_FROM_3
97 WAGE_RATE_OF_PAY_TO_3
98 WAGE_UNIT_OF_PAY_3
99 PREVAILING_WAGE_3
100 PW_UNIT_OF_PAY_3
101 PW_TRACKING_NUMBER_3
102 PW_WAGE_LEVEL_3
103 PW_OES_YEAR_3
104 PW_OTHER_SOURCE_3
105 PW_NON-OES_YEAR_3
106 PW_SURVEY_PUBLISHER_3
107 PW_SURVEY_NAME_3
108 WORKSITE_WORKERS_4
109 SECONDARY_ENTITY_4
110 SECONDARY_ENTITY_BUSINESS_NAME_4
111 WORKSITE_ADDRESS1_4
112 WORKSITE_ADDRESS2_4
113 WORKSITE_CITY_4
114 WORKSITE_COUNTY_4
115 WORKSITE_STATE_4
116 WORKSITE_POSTAL_CODE_4
117 WAGE_RATE_OF_PAY_FROM_4
118 WAGE_RATE_OF_PAY_TO_4
119 WAGE_UNIT_OF_PAY_4
120 PREVAILING_WAGE_4
121 PW_UNIT_OF_PAY_4
122 PW_TRACKING_NUMBER_4
123 PW_WAGE_LEVEL_4
124 PW_OES_YEAR_4
125 PW_OTHER_SOURCE_4
126 PW_NON-OES_YEAR_4
127 PW_SURVEY_PUBLISHER_4
128 PW_SURVEY_NAME_4
129 WORKSITE_WORKERS_5
130 SECONDARY_ENTITY_5
131 SECONDARY_ENTITY_BUSINESS_NAME_5
132 WORKSITE_ADDRESS1_5
133 WORKSITE_ADDRESS2_5
134 WORKSITE_CITY_5
135 WORKSITE_COUNTY_5
136 WORKSITE_STATE_5
137 WORKSITE_POSTAL_CODE_5
138 WAGE_RATE_OF_PAY_FROM_5

139 WAGE_RATE_OF_PAY_TO_5
140 WAGE_UNIT_OF_PAY_5
141 PREVAILING_WAGE_5
142 PW_UNIT_OF_PAY_5
143 PW_TRACKING_NUMBER_5
144 PW_WAGE_LEVEL_5
145 PW_OES_YEAR_5
146 PW_OTHER_SOURCE_5
147 PW_NON-OES_YEAR_5
148 PW_SURVEY_PUBLISHER_5
149 PW_SURVEY_NAME_5
150 WORKSITE_WORKERS_6
151 SECONDARY_ENTITY_6
152 SECONDARY_ENTITY_BUSINESS_NAME_6
153 WORKSITE_ADDRESS1_6
154 WORKSITE_ADDRESS2_6
155 WORKSITE_CITY_6
156 WORKSITE_COUNTY_6
157 WORKSITE_STATE_6
158 WORKSITE_POSTAL_CODE_6
159 WAGE_RATE_OF_PAY_FROM_6
160 WAGE_RATE_OF_PAY_TO_6
161 WAGE_UNIT_OF_PAY_6
162 PREVAILING_WAGE_6
163 PW_UNIT_OF_PAY_6
164 PW_TRACKING_NUMBER_6
165 PW_WAGE_LEVEL_6
166 PW_OES_YEAR_6
167 PW_OTHER_SOURCE_6
168 PW_NON-OES_YEAR_6
169 PW_SURVEY_PUBLISHER_6
170 PW_SURVEY_NAME_6
171 WORKSITE_WORKERS_7
172 SECONDARY_ENTITY_7
173 SECONDARY_ENTITY_BUSINESS_NAME_7
174 WORKSITE_ADDRESS1_7
175 WORKSITE_ADDRESS2_7
176 WORKSITE_CITY_7
177 WORKSITE_COUNTY_7
178 WORKSITE_STATE_7
179 WORKSITE_POSTAL_CODE_7
180 WAGE_RATE_OF_PAY_FROM_7
181 WAGE_RATE_OF_PAY_TO_7
182 WAGE_UNIT_OF_PAY_7
183 PREVAILING_WAGE_7
184 PW_UNIT_OF_PAY_7
185 PW_TRACKING_NUMBER_7
186 PW_WAGE_LEVEL_7

187 PW_OES_YEAR_7
188 PW_OTHER_SOURCE_7
189 PW_NON-OES_YEAR_7
190 PW_SURVEY_PUBLISHER_7
191 PW_SURVEY_NAME_7
192 WORKSITE_WORKERS_8
193 SECONDARY_ENTITY_8
194 SECONDARY_ENTITY_BUSINESS_NAME_8
195 WORKSITE_ADDRESS1_8
196 WORKSITE_ADDRESS2_8
197 WORKSITE_CITY_8
198 WORKSITE_COUNTY_8
199 WORKSITE_STATE_8
200 WORKSITE_POSTAL_CODE_8
201 WAGE_RATE_OF_PAY_FROM_8
202 WAGE_RATE_OF_PAY_TO_8
203 WAGE_UNIT_OF_PAY_8
204 PREVAILING_WAGE_8
205 PW_UNIT_OF_PAY_8
206 PW_TRACKING_NUMBER_8
207 PW_WAGE_LEVEL_8
208 PW_OES_YEAR_8
209 PW_OTHER_SOURCE_8
210 PW_NON-OES_YEAR_8
211 PW_SURVEY_PUBLISHER_8
212 PW_SURVEY_NAME_8
213 WORKSITE_WORKERS_9
214 SECONDARY_ENTITY_9
215 SECONDARY_ENTITY_BUSINESS_NAME_9
216 WORKSITE_ADDRESS1_9
217 WORKSITE_ADDRESS2_9
218 WORKSITE_CITY_9
219 WORKSITE_COUNTY_9
220 WORKSITE_STATE_9
221 WORKSITE_POSTAL_CODE_9
222 WAGE_RATE_OF_PAY_FROM_9
223 WAGE_RATE_OF_PAY_TO_9
224 WAGE_UNIT_OF_PAY_9
225 PREVAILING_WAGE_9
226 PW_UNIT_OF_PAY_9
227 PW_TRACKING_NUMBER_9
228 PW_WAGE_LEVEL_9
229 PW_OES_YEAR_9
230 PW_OTHER_SOURCE_9
231 PW_NON-OES_YEAR_9
232 PW_SURVEY_PUBLISHER_9
233 PW_SURVEY_NAME_9
234 WORKSITE_WORKERS_10

```
235 SECONDARY_ENTITY_10
236 SECONDARY_ENTITY_BUSINESS_NAME_10
237 WORKSITE_ADDRESS1_10
238 WORKSITE_ADDRESS2_10
239 WORKSITE_CITY_10
240 WORKSITE_COUNTY_10
241 WORKSITE_STATE_10
242 WORKSITE_POSTAL_CODE_10
243 WAGE_RATE_OF_PAY_FROM_10
244 WAGE_RATE_OF_PAY_TO_10
245 WAGE_UNIT_OF_PAY_10
246 PREVAILING_WAGE_10
247 PW_UNIT_OF_PAY_10
248 PW_TRACKING_NUMBER_10
249 PW_WAGE_LEVEL_10
250 PW_OES_YEAR_10
251 PW_OTHER_SOURCE_10
252 PW_NON-OES_YEAR_10
253 PW_SURVEY_PUBLISHER_10
254 PW_SURVEY_NAME_10
255 H-1B_DEPENDENT
256 WILLFUL_VIOLATOR
257 SUPPORT_H1B
258 STATUTORY_BASIS
259 MASTERS_EXEMPTION
260 PUBLIC_DISCLOSURE
```

There are 260 catergories in this dataset, and some of the categories are redundant to this project. For example, categories like *AGENT_REPRESENTING_EMPLOYER* and *NAME_OF_HIGHEST_STATE_COURT* are not likely to affect the final result stored in the *CASE_STATUS* column, and catergories like *SOC_CODE* and *SOC_TITLE* are relative to each other.

As a result, in this project, we will primary focus on the following catergories:

[7]:
```
# focus = [3, 26, 27, 38, 41, 43, 47, 55, 108, 111, 113, 115, 125]
# focus = [3, 27, 38, 41, 43, 47, 55, 108, 111, 113, 115, 125]
focus = [2, 6, 7, 9, 10, 20, 52, 60, 258]

for i, ele in enumerate(focus):
    print (str(i + 1) + " " + data_columns[ele - 1])

print ("\nThere are {} focused catergories, in total.".format(len(focus)))
```

```
1 CASE_STATUS
2 VISA_CLASS
3 JOB_TITLE
4 SOC_TITLE
5 FULL_TIME_POSITION
```

```
6 EMPLOYER_NAME
7 WORKSITE_STATE_1
8 PW_WAGE_LEVEL_1
9 STATUTORY_BASIS

There are 9 focused catergories, in total.
```

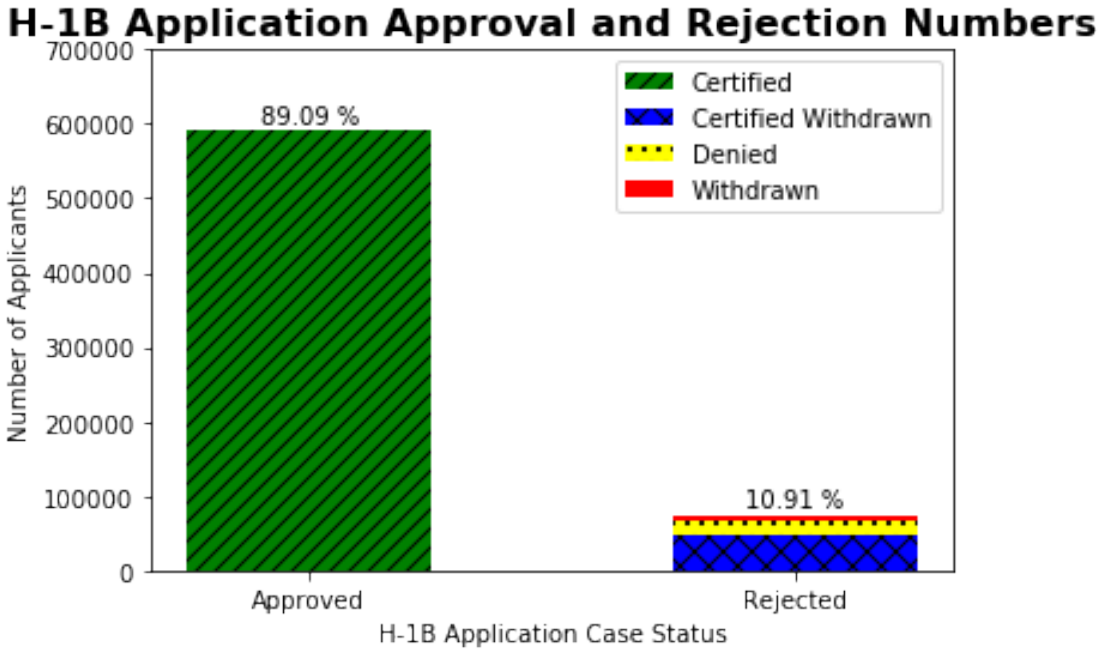
Data Visualization: Case Status

[8]:
```
# Explore CASE_STATUS Column
# Deep copy the case status list
CASE_STATUS = deepcopy(list(data["CASE_STATUS"]))
# Convert to array
CASE_STATUS_array = np.array(CASE_STATUS)

# Determine the how many different classes in the Case Status Category
CASE_STATUS_classes = np.unique(CASE_STATUS_array)
print("Classes of the Case Status: ", CASE_STATUS_classes)

# Count how many applicants are denied and approved
case_dict = Counter(CASE_STATUS_array)

# Display how many applicants are certified, certified withdrawn, denied, and␣
 ↪withdrawn.
#print(case_dict)
certified_Num = Counter(case_dict).most_common(len(Counter(case_dict)))[0][1]
certified_withdrawn_Num = Counter(case_dict).
 ↪most_common(len(Counter(case_dict)))[1][1]
denied_Num = Counter(case_dict).most_common(len(Counter(case_dict)))[2][1]
withdrawn_Num = Counter(case_dict).most_common(len(Counter(case_dict)))[3][1]

# Turn multiple classes classification to a binary class classification
# Denied included DENIED, CERTIFIED-WITHDRAWN, WITHDRAWN
# Certified is CERTIFIED
case_dict = {"DENIED" : 0, "CERTIFIED" : 1, "CERTIFIED-WITHDRAWN" : 0,␣
 ↪"WITHDRAWN" : 0}

# Determine the number of denied and certified
for i in range(len(CASE_STATUS)):

    case = CASE_STATUS[i]
    CASE_STATUS[i] = case_dict[case]

case = np.array(deepcopy(CASE_STATUS))

# Calculate the percentage of certified denied applicants
percent_certified = np.count_nonzero(case == 1) / len(case) * 100
```

```
# Calculate the percentage of denied denied applicants
percent_denied = np.count_nonzero(case == 0) / len(case) * 100

# Display results
print ("Percentage of Certified Applicants: ", percent_certified, " %")
print ("Percentage of Denied Applicants: ", percent_denied, " %")
print ("Number of Certified Applicants: ", np.count_nonzero(case == 1))
print ("Number of Denied Applicants: ", len(case) - np.count_nonzero(case == 1))


#### For Data Visualization
case_status_label = ["Approved", "Rejected"]
number_of_approved_applicants = np.array([certified_Num, 0])
number_of_certified_withdrawn_applicants = np.array([0,␣
↪certified_withdrawn_Num])
number_of_denied_applicants = np.array([0, denied_Num])
number_of_withdrawn_applicants = np.array([0, withdrawn_Num])

# Plot bars in a stack method
fig, ax = plt.subplots()

ax.bar(case_status_label, number_of_approved_applicants, width=0.5, color =␣
↪'green', hatch='///')
ax.bar(case_status_label, number_of_certified_withdrawn_applicants, bottom =␣
↪number_of_approved_applicants, width=0.5, color = 'blue', hatch='xx')
ax.bar(case_status_label, number_of_denied_applicants, bottom =␣
↪number_of_approved_applicants + number_of_certified_withdrawn_applicants,␣
↪width=0.5, color = 'yellow', hatch='..')
ax.bar(case_status_label, number_of_withdrawn_applicants, bottom =␣
↪number_of_approved_applicants + number_of_certified_withdrawn_applicants +␣
↪number_of_denied_applicants, width=0.5, color = 'red')
ax.set_xlabel("H-1B Application Case Status")
ax.set_ylabel("Number of Applicants")
ax.set_ylim([0, 700000])
ax.legend(["Certified", "Certified Withdrawn", "Denied", "Withdrawn"])

# Add labels to each bar.
for i, p in enumerate(ax.patches):

    if i == 0:
        h, w, x = p.get_height(), p.get_width(), p.get_x()
        xy = (x + w / 2., h + 20000)
        text = f'{percent_certified:0.2f} %'
        ax.annotate(s=text, xy=xy, ha='center', va='center')

    if i == 3:
        h, w, x = p.get_height(), p.get_width(), p.get_x()
```

```
        xy = (x + w / 2., h + 50000)
        text = f'{percent_denied:0.2f} %'
        ax.annotate(s=text, xy=xy, ha='center', va='center')

plt.title("H-1B Application Approval and Rejection Numbers", fontweight='bold',␣
 ↪fontsize = 16)
plt.show()
```

```
Classes of the Case Status:  ['CERTIFIED' 'CERTIFIED-WITHDRAWN' 'DENIED'
'WITHDRAWN']
Percentage of Certified Applicants:  89.08948926899141  %
Percentage of Denied Applicants:  10.910510731008582  %
Number of Certified Applicants:  592103
Number of Denied Applicants:  72513
```



Data Visualization: Visa Class

[9]:
```
# Explore VISA_CLASS Column
# Deep copy the visa class list
VISA_CLASS = deepcopy(list(data["VISA_CLASS"]))

# Convert to array
VISA_CLASS_array = np.array(VISA_CLASS)

# Determine the how many different classes in the Visa Class Category
VISA_CLASS_classes = np.unique(VISA_CLASS_array)
```

```
print("Classes of the Visa Class: ", VISA_CLASS_classes)
print("\n")

# Count how many applicants are denied and approved
visa_dict = Counter(VISA_CLASS)
# Display how many applicants are H-1B, E-3 Australian, H-1B1 Singapore, and␣
 ↪H-1B1 Chile.
print("Number of Applicants for each Visa Class:")
print(visa_dict)
print("\n")

# Create a visa dictionary that stores each visa class approved # and rejected #
# for example, {H-1B: [Rejected #, Approved#]
#               E-3 Australian: [Rejected #, Approved#]...}
visa_dict = {i: [0, 0] for i in set(VISA_CLASS)}
# Loop through the VISA_CLASS column to make sure there is no blank space in␣
 ↪the list
for idx in range(0, len(VISA_CLASS)):

    # Extract visa type information
    visa_type = VISA_CLASS[idx]

    if type(visa_type) == str:

        # Update the VISA_CLASS
        VISA_CLASS[idx] = visa_type

    else:

        # if empty, it is a None type
        VISA_CLASS[idx] = None

# Total applicant counts
total_applicants = 0

# Counts the number of approved and number of rejection for each class.
for idx in range(0, len(CASE_STATUS)):

    # Extract the visa type and case status
    visa_class, status = VISA_CLASS[idx], CASE_STATUS[idx]

    # Incement the count
    visa_dict[visa_class][status] += 1
    total_applicants += 1

# Display each visa class' rejected and approved counts
print("Visa Class: [Rejected Count, Approved Count]")
```

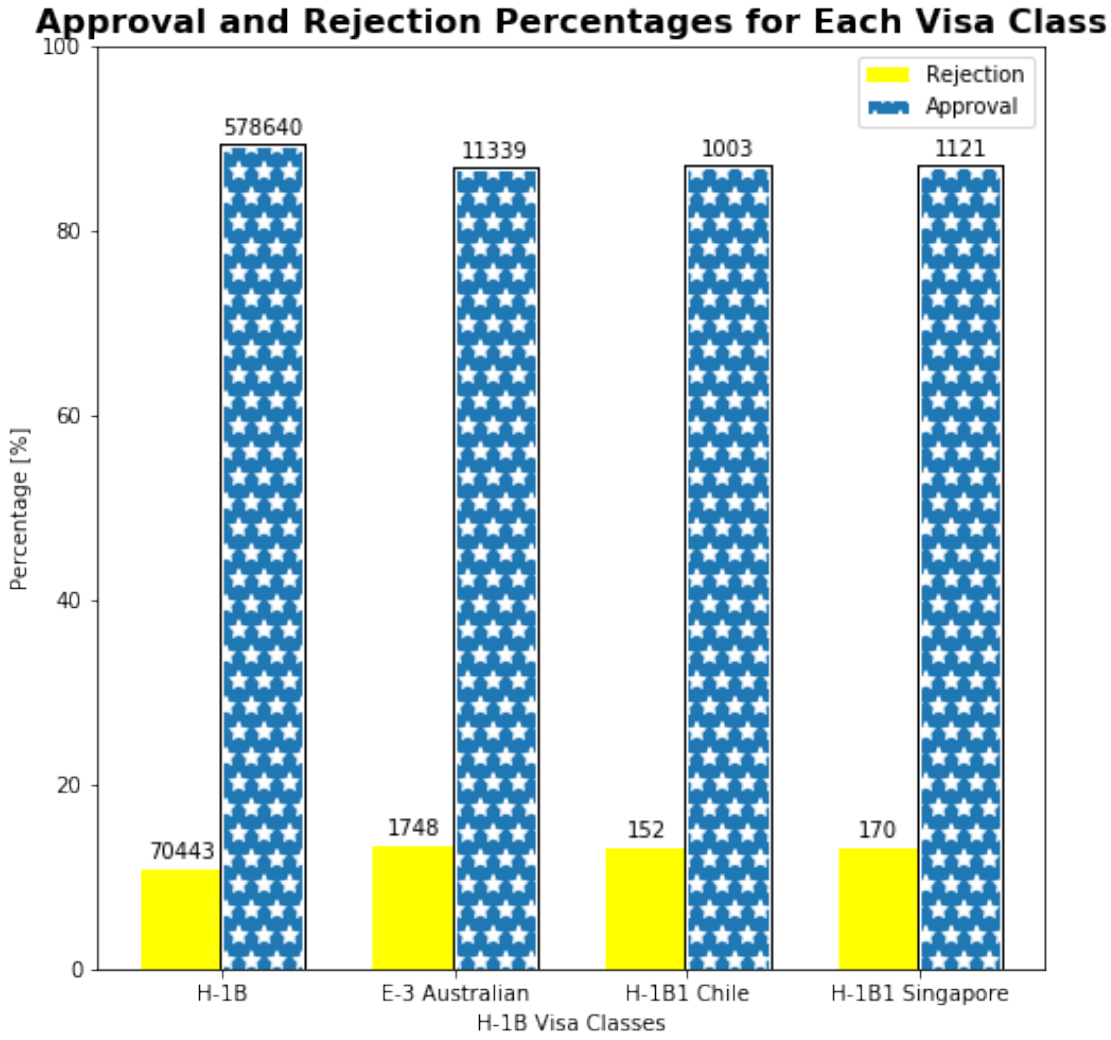
```
print(visa_dict)

# Define rejection and approved visa percentage list
rej, pas = [], []

# Total H-1B applicants
H_1B_tot_applicants = visa_dict["H-1B"][1] + visa_dict["H-1B"][0]
# H-1B approved percentage
H_1B_apr_perc = visa_dict["H-1B"][1] / H_1B_tot_applicants * 100
# H-1B rejected percentage
H_1B_rjt_perc = visa_dict["H-1B"][0] / H_1B_tot_applicants * 100
# Append to the list
rej.append(H_1B_rjt_perc)
pas.append(H_1B_apr_perc)

print(H_1B_apr_perc, H_1B_rjt_perc)

# Total E-3 applicants
E_3_tot_applicants = visa_dict["E-3 Australian"][1] + visa_dict["E-3␣
 ↪Australian"][0]
# E_3 approved percentage
E_3_apr_perc = visa_dict["E-3 Australian"][1] / E_3_tot_applicants * 100
# E_3 rejected percentage
E_3_rjt_perc = visa_dict["E-3 Australian"][0] / E_3_tot_applicants * 100
# Append to the list
rej.append(E_3_rjt_perc)
pas.append(E_3_apr_perc)

print(E_3_apr_perc, E_3_rjt_perc)

# Total H-1B1 Chile applicants
H_1B1_C_tot_applicants = visa_dict["H-1B1 Chile"][1] + visa_dict["H-1B1␣
 ↪Chile"][0]
# H-1B1 Chile approved percentage
H_1B1_C_apr_perc = visa_dict["H-1B1 Chile"][1] / H_1B1_C_tot_applicants * 100
# H-1B1 Chile rejected percentage
H_1B1_C_rjt_perc = visa_dict["H-1B1 Chile"][0] / H_1B1_C_tot_applicants * 100
# Append to the list
rej.append(H_1B1_C_rjt_perc)
pas.append(H_1B1_C_apr_perc)

print(H_1B1_C_apr_perc, H_1B1_C_rjt_perc)

# Total H-1B1 Singapore applicants
H_1B1_S_tot_applicants = visa_dict["H-1B1 Singapore"][1] + visa_dict["H-1B1␣
 ↪Singapore"][0]
# H-1B1 Singapore approved percentage
```

```
H_1B1_S_apr_perc = visa_dict["H-1B1 Singapore"][1] / H_1B1_S_tot_applicants *␣
↪100
# H-1B1 Singapore rejected percentage
H_1B1_S_rjt_perc = visa_dict["H-1B1 Singapore"][0] / H_1B1_S_tot_applicants *␣
↪100
# Append to the list
rej.append(H_1B1_S_rjt_perc)
pas.append(H_1B1_S_apr_perc)

print(H_1B1_S_apr_perc, H_1B1_S_rjt_perc)

# Visa class labels
visa_class_labels = ["H-1B", "E-3 Australian", "H-1B1 Chile", "H-1B1 Singapore"]

# Define rejection and approved visa numbers list
rej_Num, pas_Num = [], []

# Append the numbers to the list
for label in visa_class_labels:
    rej_Num.append(visa_dict[label][0])
    pas_Num.append(visa_dict[label][1])

# Combined into an array
rej_pas_Num_array = np.array([rej_Num,pas_Num])
rej_pas_Num_array = rej_pas_Num_array.T
# Flatten the array
rej_pas_Num_array = rej_pas_Num_array.flatten()

# Indices for each visa class
ind = np.arange(len(visa_class_labels))

# Plot bars
fig, ax = plt.subplots(figsize = (8, 8))

# Set Bar Graph width
bar_width = 0.35

# Plot the bar graph
ax.bar(ind, rej, bar_width, label = "Rejection", color = 'yellow', zorder = 0)
ax.bar(ind + bar_width, pas, bar_width, label = "Approval", hatch = '*', fill =␣
↪"white", edgecolor = 'white', zorder = 2, lw = 1)
ax.set_title("Approval and Rejection Percentages for Each Visa Class",␣
↪fontweight='bold', fontsize = 16)
ax.set_ylabel("Percentage [%]")
ax.set_xlabel("H-1B Visa Classes")
ax.set_ylim([0, 100])
ax.set_xticks(ind + bar_width / 2)
```

```
ax.set_xticklabels(visa_class_labels)
ax.legend(loc = "best")

# Annotate the number of applicant for each class and case status bar
for i, p in enumerate(ax.patches):

    # If index is less than 3
    if i <= 3:

        height = p.get_height()
        ax.annotate('{}'.format(rej_pas_Num_array[i*2]),
                    xy=(p.get_x() + p.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

    # If more than 3
    else:

        height = p.get_height()
        ax.annotate('{}'.format(rej_pas_Num_array[i - (len(rej_pas_Num_array) -␣
 ↪1 - i)]),
                    xy=(p.get_x() + p.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

# Draw black edge for the approved bar
ax.bar(ind + bar_width, pas, bar_width, edgecolor = 'black', zorder = 1, lw = 3)

plt.show()
```

```
Classes of the Visa Class:  ['E-3 Australian' 'H-1B' 'H-1B1 Chile' 'H-1B1
Singapore']


Number of Applicants for each Visa Class:
Counter({'H-1B': 649083, 'E-3 Australian': 13087, 'H-1B1 Singapore': 1291,
'H-1B1 Chile': 1155})


Visa Class: [Rejected Count, Approved Count]
{'H-1B': [70443, 578640], 'E-3 Australian': [1748, 11339], 'H-1B1 Singapore':
[170, 1121], 'H-1B1 Chile': [152, 1003]}
89.14730473606612 10.852695263933889
86.64323374340948 13.356766256590511
86.83982683982684 13.16017316017316
```

```
86.83191324554609 13.168086754453912
```



Data Visualization: Worksite State

[10]:
```
# WORKSITE_STATE
# Explore VISA_CLASS Column
# Deep copy the visa class list
WORKSITE_STATE_1 = deepcopy(list(data["WORKSITE_STATE_1"]))

# Convert to array
WORKSITE_STATE_1_array = np.array(WORKSITE_STATE_1)

# Loop through the WORKSITE_STATE_1 column to make sure there is no blank space␣
 ↪in the list
for idx in range(0, len(WORKSITE_STATE_1)):
```

```
    # Extract work state information
    work_state = WORKSITE_STATE_1[idx]

    if type(work_state) == str:

        # Update the WORKSITE_STATE_1
        WORKSITE_STATE_1[idx] = work_state

    else:

        # if empty, it is a None type
        WORKSITE_STATE_1[idx] = "OTHERS"

# Determine the how many different states in the WORKSITE_STATE_1 column
WORKSITE_states = np.unique(WORKSITE_STATE_1_array)
print("Worksite States :\n\n", WORKSITE_states)
print("\n")

states_dict = {"OTHERS" : 0,
               "ALABAMA" : 1, "AL" : 1,
               "ALASKA" : 2, "AK" : 2,
               "ARIZONA" : 3, "AZ" : 3,
               "ARKANSAS" : 4, "AR" : 4,
               "CALIFORNIA" : 5, "CA" : 5,
               "COLORADO" : 6, "CO" : 6,
               "CONNECTICUT" : 7, "CT" : 7,
               "DELAWARE" : 8, "DE" : 8,
               "DISTRICT OF COLUMBIA": 9, "DC": 9,
               "FLORIDA" : 10, "FL": 10,
               "GEORGIA" : 11, "GA": 11,
               "GUAM": 12, "GU": 12,
               "HAWAII" : 13, "HI" : 13,
               "IDAHO": 14, "ID" : 14,
               "ILLINOIS": 15, "IL": 15,
               "INDIANA": 16, "IN" : 16,
               "IOWA": 17, "IA" : 17,
               "KANSAS": 18, "KS": 18,
               "KENTUCKY": 19, "KY": 19,
               "LOUISIANA": 20, "LA": 20,
               "MAINE": 21, "ME" : 21,
               "MARYLAND": 22, "MD": 22,
               "MASSACHUSETTS": 23, "MA": 23,
               "MICHIGAN": 24, "MI": 24,
               "MINNESOTA": 25, "MN": 25,
               "MISSISSIPPI": 26, "MS": 26,
               "MISSOURI": 27, "MO": 27,
```

```
                "MONTANA": 28, "MT": 28,
                "NEBRASKA": 29, "NE": 29,
                "NEVADA": 30, "NV": 30,
                "NEW HAMPSHIRE": 31, "NH": 31,
                "NEW JERSEY": 32, "NJ": 32,
                "NEW MEXICO": 33, "NM": 33,
                "NEW YORK": 34, "NY": 34,
                "NORTH CAROLINA": 35, "NC": 35,
                "NORTH DAKOTA": 36, "ND": 36,
                "NORTHERN MARIANA ISLANDS": 37, "MP": 37,
                "OHIO": 38, "OH": 38,
                "OKLAHOMA": 39, "OK": 39,
                "OREGON": 40, "OR": 40,
                "PENNSYLVANIA": 41, "PA": 41,
                "PUERTO RICO": 42, "PR": 42,
                "RHODE ISLAND": 43, "RI": 43,
                "SOUTH CAROLINA": 44, "SC": 44,
                "SOUTH DAKOTA": 45, "SD": 45,
                "TENNESSEE": 46, "TN": 46,
                "TEXAS": 47, "TX": 47,
                "UTAH": 48, "UT": 48,
                "VERMONT": 49, "VT": 49,
                "VIRGINIA": 50, "VA": 50,
                "VIRGIN ISLANDS": 51, "VI": 51,
                "WASHINGTON": 52, "WA": 52,
                "WEST VIRGINIA": 53, "WV": 53,
                "WISCONSIN": 54, "WI":54,
                "WYOMING": 55, "WY": 55,
                "PALAU": 56, "PW": 56,
                "MARSHALL ISLANDS": 57}


# Determine the number of denied and certified
for i in range(len(WORKSITE_STATE_1)):

    # Get state (string)
    state = WORKSITE_STATE_1[i]
    # Output state id
    WORKSITE_STATE_1[i] = states_dict[state]

    # Convert to state in string or None
    WORKSITE_STATE_1[i] = list(states_dict.keys())[list(states_dict.values()).
 ↪index(WORKSITE_STATE_1[i])]

# Count how many applicants are there for different worksite states
new_states_dict = Counter(WORKSITE_STATE_1)
```

```
# Display how many applicants are H-1B, E-3 Australian, H-1B1 Singapore, and␣
 ↪H-1B1 Chile.
print("Number of Applicants in Different Worksite States:\n\n")
print(new_states_dict)
print("\n")

# Determine the how many different states in the WORKSITE_STATE_1
#U_States = np.unique(u_states_array)
print("Different Worksite States:\n\n", new_states_dict.keys())
print("\n")

# Convert to dataframe
df_states = pd.DataFrame(list(new_states_dict.items()))
df_states.columns = ['Worksite States', 'Number of H-1B Applicants']
#print(df_states)

fig = px.treemap(df_states, path=['Worksite States'],
                 values='Number of H-1B Applicants',
                 color='Number of H-1B Applicants',
                 color_continuous_scale='RdBu')
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```

```
Worksite States :

 ['AK' 'AL' 'ALABAMA' 'ALASKA' 'AR' 'ARIZONA' 'ARKANSAS' 'AZ' 'CA'
 'CALIFORNIA' 'CO' 'COLORADO' 'CONNECTICUT' 'CT' 'DC' 'DE' 'DELAWARE'
 'DISTRICT OF COLUMBIA' 'FL' 'FLORIDA' 'GA' 'GEORGIA' 'GU' 'GUAM' 'HAWAII'
 'HI' 'IA' 'ID' 'IDAHO' 'IL' 'ILLINOIS' 'IN' 'INDIANA' 'IOWA' 'KANSAS'
 'KENTUCKY' 'KS' 'KY' 'LA' 'LOUISIANA' 'MA' 'MAINE' 'MARSHALL ISLANDS'
 'MARYLAND' 'MASSACHUSETTS' 'MD' 'ME' 'MI' 'MICHIGAN' 'MINNESOTA'
 'MISSISSIPPI' 'MISSOURI' 'MN' 'MO' 'MONTANA' 'MP' 'MS' 'MT' 'NC' 'ND'
 'NE' 'NEBRASKA' 'NEVADA' 'NEW HAMPSHIRE' 'NEW JERSEY' 'NEW MEXICO'
 'NEW YORK' 'NH' 'NJ' 'NM' 'NORTH CAROLINA' 'NORTH DAKOTA'
 'NORTHERN MARIANA ISLANDS' 'NV' 'NY' 'OH' 'OHIO' 'OK' 'OKLAHOMA' 'OR'
 'OREGON' 'PA' 'PALAU' 'PENNSYLVANIA' 'PR' 'PUERTO RICO' 'PW'
 'RHODE ISLAND' 'RI' 'SC' 'SD' 'SOUTH CAROLINA' 'SOUTH DAKOTA' 'TENNESSEE'
 'TEXAS' 'TN' 'TX' 'UT' 'UTAH' 'VA' 'VERMONT' 'VI' 'VIRGIN ISLANDS'
 'VIRGINIA' 'VT' 'WA' 'WASHINGTON' 'WEST VIRGINIA' 'WI' 'WISCONSIN' 'WV'
 'WY' 'WYOMING' 'nan']


Number of Applicants in Different Worksite States:


Counter({'CALIFORNIA': 132251, 'TEXAS': 66718, 'NEW YORK': 56903, 'NEW JERSEY':
38861, 'ILLINOIS': 32812, 'WASHINGTON': 29850, 'MASSACHUSETTS': 26522,
```

```
'PENNSYLVANIA': 22683, 'GEORGIA': 22650, 'FLORIDA': 21898, 'NORTH CAROLINA':
21310, 'MICHIGAN': 21181, 'VIRGINIA': 18783, 'OHIO': 16413, 'ARIZONA': 11530,
'MARYLAND': 10802, 'MINNESOTA': 10493, 'CONNECTICUT': 10178, 'MISSOURI': 9385,
'COLORADO': 7697, 'INDIANA': 7529, 'TENNESSEE': 7320, 'WISCONSIN': 6949,
'OREGON': 5263, 'ARKANSAS': 4356, 'DISTRICT OF COLUMBIA': 4220, 'UTAH': 3783,
'DELAWARE': 3384, 'IOWA': 3375, 'SOUTH CAROLINA': 3334, 'KENTUCKY': 2883,
'KANSAS': 2681, 'RHODE ISLAND': 2541, 'NEBRASKA': 2302, 'ALABAMA': 1994,
'LOUISIANA': 1883, 'OKLAHOMA': 1770, 'NEVADA': 1734, 'NEW HAMPSHIRE': 1697, 'NEW
MEXICO': 1072, 'MISSISSIPPI': 914, 'IDAHO': 842, 'WEST VIRGINIA': 536, 'MAINE':
529, 'HAWAII': 515, 'NORTH DAKOTA': 433, 'VERMONT': 401, 'GUAM': 355, 'SOUTH
DAKOTA': 280, 'MONTANA': 189, 'ALASKA': 172, 'WYOMING': 160, 'PUERTO RICO': 116,
'NORTHERN MARIANA ISLANDS': 113, 'VIRGIN ISLANDS': 50, 'OTHERS': 18, 'PALAU': 2,
'MARSHALL ISLANDS': 1})


Different Worksite States:

 dict_keys(['NEW YORK', 'ILLINOIS', 'GEORGIA', 'CALIFORNIA', 'UTAH', 'OREGON',
'TEXAS', 'TENNESSEE', 'NEW JERSEY', 'MINNESOTA', 'MARYLAND', 'FLORIDA',
'WASHINGTON', 'HAWAII', 'ARIZONA', 'VIRGINIA', 'OHIO', 'PENNSYLVANIA',
'CONNECTICUT', 'DISTRICT OF COLUMBIA', 'NORTH CAROLINA', 'DELAWARE', 'MICHIGAN',
'NEVADA', 'KENTUCKY', 'MISSISSIPPI', 'MASSACHUSETTS', 'SOUTH DAKOTA',
'COLORADO', 'INDIANA', 'GUAM', 'WISCONSIN', 'RHODE ISLAND', 'NEBRASKA',
'ARKANSAS', 'ALABAMA', 'PUERTO RICO', 'KANSAS', 'NORTHERN MARIANA ISLANDS',
'IDAHO', 'NEW HAMPSHIRE', 'MISSOURI', 'LOUISIANA', 'VIRGIN ISLANDS', 'OKLAHOMA',
'SOUTH CAROLINA', 'IOWA', 'ALASKA', 'WYOMING', 'NORTH DAKOTA', 'NEW MEXICO',
'WEST VIRGINIA', 'MAINE', 'MONTANA', 'VERMONT', 'OTHERS', 'PALAU', 'MARSHALL
ISLANDS'])
```

[11]:
```python
# Create a state count dictionary that stores each states approved # and␣
 ↪rejected #
# for example, {California: [Rejected #, Approved#, Total # of Applicant in the␣
 ↪State, Percentage of Approved]
#               Ohio: [Rejected #, Approved#]...}
state_count_dict = {i: [0, 0, 0, 0.0] for i in set(WORKSITE_STATE_1)}

# Total applicant counts
total_applicants = 0

# Counts the number of approved and number of rejection for each worksite state.
for idx in range(0, len(CASE_STATUS)):

    # Extract the work state and case status
    wrk_state, c_status = WORKSITE_STATE_1[idx], CASE_STATUS[idx]
```

```
    # Incement the count
    state_count_dict[wrk_state][c_status] += 1
    state_count_dict[wrk_state][2] += 1
    state_count_dict[wrk_state][3] = state_count_dict[wrk_state][1] /␣
 ↪state_count_dict[wrk_state][2] * 100

    total_applicants += 1

# Display each worksite state rejected and approved counts, total applicants in␣
 ↪the state, and approved percentage
print("Worksite State: [Rejected Count, Approved Count, Total H-1B Applicants␣
 ↪in the State, Certified H-1B Visas Percentage]:\n")
print(state_count_dict)

# Convert to dataframe
df_states_count = pd.DataFrame.from_dict(state_count_dict, orient='index')
df_states_count.columns = ['Number of Denied H-1B Visas', 'Number of Certified␣
 ↪H-1B Visas', 'Total H-1B Applicants in the State', 'Certified H-1B Visas␣
 ↪Percentage [%]']
#print(df_states_count)
```

```
Worksite State: [Rejected Count, Approved Count, Total H-1B Applicants in the
State, Certified H-1B Visas Percentage]:

{'OKLAHOMA': [214, 1556, 1770, 87.909604519774], 'NEW MEXICO': [147, 925, 1072,
86.28731343283582], 'MISSOURI': [1149, 8236, 9385, 87.75705913692062],
'LOUISIANA': [256, 1627, 1883, 86.40467339352098], 'NEVADA': [183, 1551, 1734,
89.44636678200692], 'MARSHALL ISLANDS': [0, 1, 1, 100.0], 'FLORIDA': [2327,
19571, 21898, 89.37345876335738], 'MASSACHUSETTS': [3531, 22991, 26522,
86.68652439484201], 'MARYLAND': [1414, 9388, 10802, 86.90983151268283], 'SOUTH
DAKOTA': [44, 236, 280, 84.28571428571429], 'VIRGIN ISLANDS': [10, 40, 50,
80.0], 'PUERTO RICO': [13, 103, 116, 88.79310344827587], 'IOWA': [481, 2894,
3375, 85.74814814814815], 'COLORADO': [923, 6774, 7697, 88.00831492789398],
'ALASKA': [36, 136, 172, 79.06976744186046], 'ARIZONA': [1254, 10276, 11530,
89.12402428447528], 'WEST VIRGINIA': [99, 437, 536, 81.52985074626866],
'WISCONSIN': [748, 6201, 6949, 89.2358612750036], 'WYOMING': [47, 113, 160,
70.625], 'HAWAII': [84, 431, 515, 83.68932038834951], 'ALABAMA': [281, 1713,
1994, 85.90772316950851], 'MICHIGAN': [3166, 18015, 21181, 85.05264151834191],
'IDAHO': [78, 764, 842, 90.73634204275535], 'NEW YORK': [6399, 50504, 56903,
88.75454721192204], 'INDIANA': [907, 6622, 7529, 87.95324744321955], 'VERMONT':
[85, 316, 401, 78.80299251870323], 'DELAWARE': [290, 3094, 3384,
91.43026004728132], 'GEORGIA': [2353, 20297, 22650, 89.61147902869757],
'TENNESSEE': [823, 6497, 7320, 88.7568306010929], 'RHODE ISLAND': [211, 2330,
2541, 91.6961826052735], 'KANSAS': [327, 2354, 2681, 87.80305856023871], 'UTAH':
[409, 3374, 3783, 89.18847475548507], 'OREGON': [655, 4608, 5263,
87.55462663879916], 'PALAU': [1, 1, 2, 50.0], 'VIRGINIA': [1800, 16983, 18783,
90.41686631528509], 'CALIFORNIA': [14921, 117330, 132251, 88.71766565092135],
```

```
'WASHINGTON': [2226, 27624, 29850, 92.5427135678392], 'ILLINOIS': [3475, 29337,
32812, 89.40936242837985], 'SOUTH CAROLINA': [377, 2957, 3334,
88.69226154769046], 'PENNSYLVANIA': [2210, 20473, 22683, 90.25702067627739],
'NORTH DAKOTA': [54, 379, 433, 87.52886836027713], 'MINNESOTA': [1033, 9460,
10493, 90.15534165634233], 'ARKANSAS': [338, 4018, 4356, 92.24058769513314],
'OHIO': [1511, 14902, 16413, 90.79388289770304], 'GUAM': [25, 330, 355,
92.95774647887323], 'OTHERS': [17, 1, 18, 5.555555555555555], 'KENTUCKY': [220,
2663, 2883, 92.36906000693722], 'MISSISSIPPI': [157, 757, 914,
82.82275711159738], 'NEW HAMPSHIRE': [183, 1514, 1697, 89.2162639952858],
'CONNECTICUT': [863, 9315, 10178, 91.52092749066614], 'NORTH CAROLINA': [2003,
19307, 21310, 90.60065696855936], 'NEW JERSEY': [4043, 34818, 38861,
89.59625331309024], 'MAINE': [45, 484, 529, 91.49338374291115], 'MONTANA': [34,
155, 189, 82.01058201058201], 'NORTHERN MARIANA ISLANDS': [22, 91, 113,
80.53097345132744], 'TEXAS': [7253, 59465, 66718, 89.12887076950749], 'DISTRICT
OF COLUMBIA': [506, 3714, 4220, 88.00947867298578], 'NEBRASKA': [252, 2050,
2302, 89.0529973935708]}
```

[12]:
```
# Display Certified H-1B Visas Percentage in Different Worksite States Treemap
fig = px.treemap(df_states_count, path=[df_states_count.index],
                 values='Certified H-1B Visas Percentage [%]',
                 color='Certified H-1B Visas Percentage [%]',
                 color_continuous_scale='RdBu')
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```

[13]:
```
# Display Number of Certified H-1B Visas in Different Worksite States Treemap
fig = px.treemap(df_states_count, path=[df_states_count.index],
                 values='Number of Certified H-1B Visas',
                 color='Number of Certified H-1B Visas',
                 color_continuous_scale='RdBu')
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```

[14]:
```
# Display Number of Denied H-1B Visas in Different Worksite States Treemap
df_states_count = df_states_count[df_states_count['Number of Denied H-1B␣
↪Visas']!=0]
fig = px.treemap(df_states_count, path=[df_states_count.index],
                 values='Number of Denied H-1B Visas',
                 color='Number of Denied H-1B Visas',
                 color_continuous_scale='RdBu')
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```

Data Visualization: Employer Name

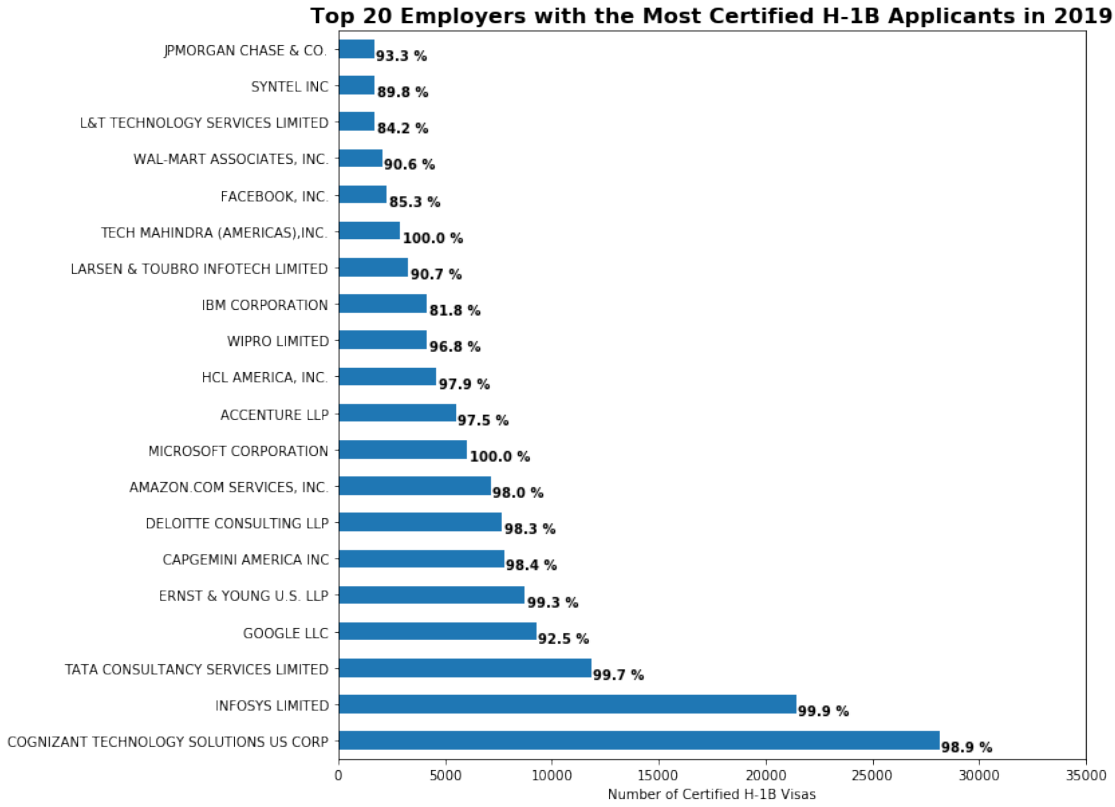
[15]:
```
# Explore EMPLOYER_NAME Column
# Deep copy the EMPLOYER_NAME list
```

```
EMPLOYER_NAME = deepcopy(list(data["EMPLOYER_NAME"]))

# Loop through the EMPLOYER_NAME column to make sure there is no blank space in␣
 ↪the list
for idx in range(0, len(EMPLOYER_NAME)):

    # Extract employer name information
    employer_name = EMPLOYER_NAME[idx]

    if type(employer_name) == str:

        # Update the EMPLOYER_NAME
        EMPLOYER_NAME[idx] = employer_name

    else:

        # if empty, it is a None type
        EMPLOYER_NAME[idx] = "OTHERS"

# Count how many applicants are there in different employers
employer_names_dict = Counter(EMPLOYER_NAME).most_common(100)[:][:]
print("Number of Applicants for each Employer:")
print(employer_names_dict)
print("\n")


# Create a employer name count dictionary that stores approved # and rejected #
# for example, {Tesla: [Rejected #, Approved#, Total # of Applicant, Percentage␣
 ↪of Approved]
#               Apple: [Rejected #, Approved#, Total # of Applicant, Percentage␣
 ↪of Approved]...}
employer_names_count_dict = {i: [0, 0, 0, 0.0] for i in set(EMPLOYER_NAME)}

# Total applicant counts
total_applicants = 0

# Counts the number of approved and number of rejection for each employer name.
for idx in range(0, len(CASE_STATUS)):

    # Extract the employer name and case status
    emplyr_nm, c_status = EMPLOYER_NAME[idx], CASE_STATUS[idx]

    # Incement the count
    employer_names_count_dict[emplyr_nm][c_status] += 1
    employer_names_count_dict[emplyr_nm][2] += 1
```

```
    employer_names_count_dict[emplyr_nm][3] =␣
 ↪employer_names_count_dict[emplyr_nm][1] /␣
 ↪employer_names_count_dict[emplyr_nm][2] * 100

    total_applicants += 1

# Convert to dataframe
df_employer_names_count = pd.DataFrame.from_dict(employer_names_count_dict,␣
 ↪orient='index')
df_employer_names_count.columns = ['Number of Denied H-1B Visas', 'Number of␣
 ↪Certified H-1B Visas', 'Total H-1B Applicants in the State', 'Certified H-1B␣
 ↪Visas Percentage [%]']

# Sorted dataframe from largest to smallest
sorted_df = df_employer_names_count['Number of Certified H-1B Visas'].
 ↪sort_values(ascending=False)
emplyr_cert_perc = df_employer_names_count.loc[sorted_df.index[:]]['Certified␣
 ↪H-1B Visas Percentage [%]']

# Plot horizontal bar
plt.figure(figsize=(10,10))
ax1 = df_employer_names_count['Number of Certified H-1B Visas'].
 ↪sort_values(ascending=False).head(20).plot(kind='barh')
ax1.set_ylabel("")
ax1.set_title('Top 20 Employers with the Most Certified H-1B Applicants in␣
 ↪2019', fontweight='bold', fontsize = 16)
ax1.set_xlim([0, 35000])
ax1.set_xlabel("Number of Certified H-1B Visas")

# Annotate the horizontal bar with Certified H-1B Visas Percentage [%]
for i, num_cert in enumerate(sorted_df[:20]):

    ax1.text(num_cert + 100, i - 0.3, f'{emplyr_cert_perc[i]:0.1f} %',␣
 ↪color='black', fontweight='bold')

plt.show()
```

```
Number of Applicants for each Employer:
[('COGNIZANT TECHNOLOGY SOLUTIONS US CORP', 28475), ('INFOSYS LIMITED', 21448),
('TATA CONSULTANCY SERVICES LIMITED', 11868), ('GOOGLE LLC', 10009), ('ERNST &
YOUNG U.S. LLP', 8801), ('CAPGEMINI AMERICA INC', 7878), ('DELOITTE CONSULTING
LLP', 7793), ('AMAZON.COM SERVICES, INC.', 7267), ('MICROSOFT CORPORATION',
6040), ('ACCENTURE LLP', 5653), ('IBM CORPORATION', 5057), ('HCL AMERICA, INC.',
4688), ('WIPRO LIMITED', 4291), ('LARSEN & TOUBRO INFOTECH LIMITED', 3624),
('TECH MAHINDRA (AMERICAS),INC.', 2915), ('FACEBOOK, INC.', 2680), ('WAL-MART
ASSOCIATES, INC.', 2277), ('IBM INDIA PRIVATE LIMITED', 2171), ('L&T TECHNOLOGY
SERVICES LIMITED', 2034), ('SYNTEL INC', 1888), ('JPMORGAN CHASE & CO.', 1786),
```

('APPLE INC.', 1708), ('INTEL CORPORATION', 1683), ('COMPUNNEL SOFTWARE GROUP, INC.', 1666), ('RANDSTAD TECHNOLOGIES, LLC', 1579), ('AMAZON WEB SERVICES, INC.', 1547), ('PRICEWATERHOUSECOOPERS ADVISORY SERVICES LLC', 1452), ('MPHASIS CORPORATION', 1403), ('SALESFORCE.COM, INC.', 1402), ('UBER TECHNOLOGIES, INC.', 1350), ('NTT DATA, INC.', 1279), ('VIRTUSA CORPORATION', 1208), ('LINKEDIN CORPORATION', 1205), ('MASTECH DIGITAL TECHNOLOGIES, INC., A MASTECH DIGITAL, INC. COMPANY', 1182), ('CAPITAL ONE SERVICES, LLC', 1179), ('MINDTREE LIMITED', 1173), ('PRICEWATERHOUSECOOPERS LLP', 1132), ('VMWARE, INC.', 1097), ('PAYPAL, INC.', 1012), ('DELOITTE & TOUCHE LLP', 1003), ('SYNECHRON, INC.', 964), ('HEXAWARE TECHNOLOGIES, INC.', 954), ('TESLA, INC.', 901), ('GOLDMAN SACHS & CO. LLC', 886), ('KPMG LLP', 861), ('BANK OF AMERICA N.A.', 856), ('POPULUS GROUP LLC', 853), ('CISCO SYSTEMS, INC.', 847), ('CUMMINS INC.', 813), ('CERNER CORPORATION', 789), ('HCL GLOBAL SYSTEMS INC', 775), ('UST GLOBAL INC.', 774), ('COMCAST CABLE COMMUNICATIONS, LLC', 772), ('V-SOFT CONSULTING GROUP, INC', 769), ('EBAY INC.', 730), ('KFORCE INC.', 723), ('TECH MAHINDRA (AMERICAS) INC.', 712), ('AVCO CONSULTING INC', 676), ('POLARIS CONSULTING & SERVICES LTD', 651), ('CITIBANK, N.A.', 640), ('MARLABS, INC', 636), ('QUEST GLOBAL SERVICES-N.A., INC.', 617), ('ORACLE AMERICA, INC.', 605), ('HTC GLOBAL SERVICES INC.', 604), ('OATH HOLDINGS INC.', 592), ('COLLABORATE SOLUTIONS INC', 591), ('UNIVERSITY OF MICHIGAN', 588), ('NTT DATA SERVICES, LLC', 569), ('INTUIT INC.', 568), ('GOLDMAN SACHS SERVICES LLC', 567), ('A2Z DEVELOPMENT CENTER, INC.', 561), ('SLK AMERICA, INC.', 561), ('CITIUSTECH INC', 552), ('JPMORGAN CHASE & CO.', 549), ('TECH MAHINDRA (AMERICAS), INC.', 548), ('GOOGLE INC.', 545), ('ADOBE INC.', 534), ('FACEBOOK INC.', 532), ('ZS ASSOCIATES, INC.', 524), ('T-MOBILE USA, INC.', 519), ('SAP AMERICA, INC.', 519), ('THE MATHWORKS, INC.', 516), ('EXPEDIA, INC.', 513), ('PROKARMA, INC.', 511), ('ANTHEM, INC.', 502), ('BLACKROCK FINANCIAL MANAGEMENT, INC.', 501), ('NATSOFT CORPORATION', 499), ('INFOSHARE SYSTEMS, INC.', 496), ('THE BOSTON CONSULTING GROUP, INC.', 484), ('AMERICAN EXPRESS TRAVEL RELATED SERVICES COMPANY, INC.', 481), ('CIBER GLOBAL LLC', 478), ('ATOS SYNTEL INC', 473), ('PERSISTENT SYSTEMS, INC.', 472), ('MICRON TECHNOLOGY, INC.', 471), ('THE BOARD OF TRUSTEES OF THE LELAND STANFORD, JR. UNIVERSITY', 466), ('DELOITTE & TOUCHE LLP', 465), ('COGNIZANT WORLDWIDE LIMITED', 465), ('MARVELL TECH LLC', 462), ('OPTUM SERVICES, INC.', 451), ('CGI TECHNOLOGIES AND SOLUTIONS INC.', 451)]



Data Visualization: Full Time Position

[16]:
```
# Explore FULL_TIME_POSITION Column
# Deep copy the FULL_TIME_POSITION list
FULL_TIME_POSITION = deepcopy(list(data["FULL_TIME_POSITION"]))

# Loop through the FULL_TIME_POSITION column to make sure there is no blank␣
 ↪space in the list
for idx in range(0, len(FULL_TIME_POSITION)):

    # Extract full time position information
    full_time_pst = FULL_TIME_POSITION[idx]

    if type(full_time_pst) == str:

        # Update the FULL_TIME_POSITION
        FULL_TIME_POSITION[idx] = full_time_pst

    else:

        # if empty, it is a N/A
        FULL_TIME_POSITION[idx] = "N/A"
```

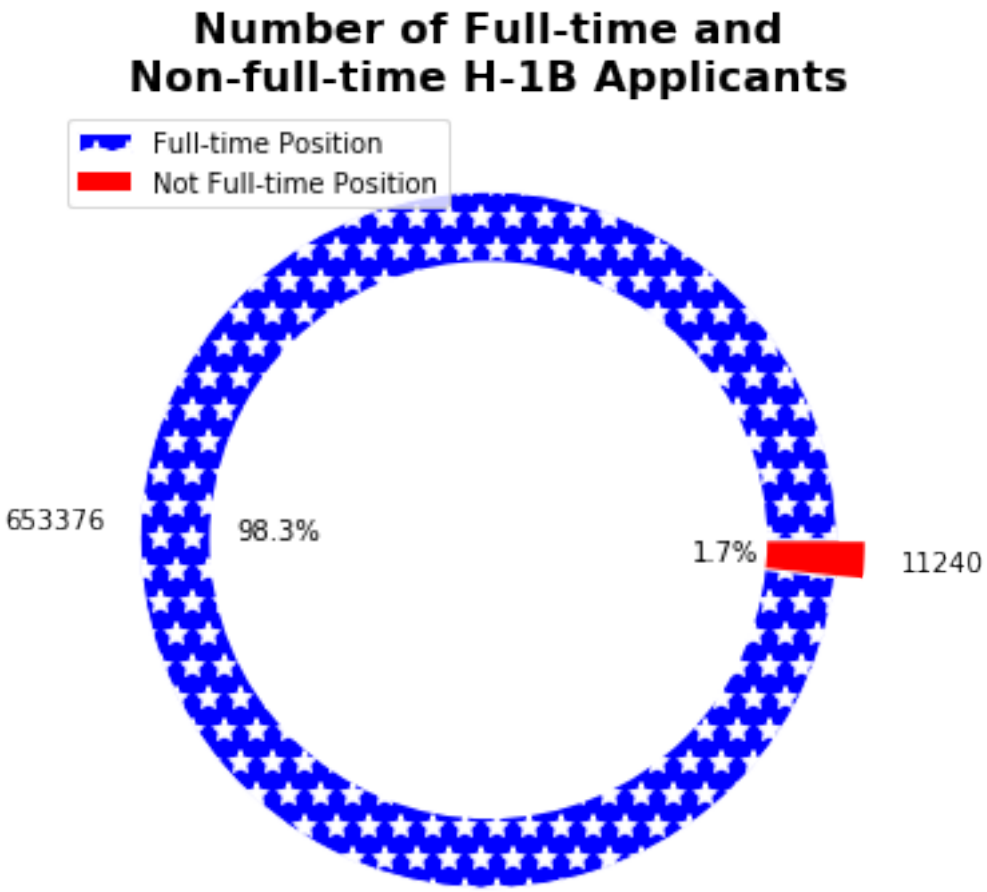
```
# Count how many applicants are full time
full_time_pst_dict = Counter(FULL_TIME_POSITION)
print("Full-position Statistics:")
print(full_time_pst_dict)
print("\n")

# Create a full-time position count dictionary that stores approved # and␣
 ↪rejected #
# for example, {"Y": [Rejected #, Approved#, Total # of Applicant, Percentage␣
 ↪of Approved]
#               "N": [Rejected #, Approved#, Total # of Applicant, Percentage␣
 ↪of Approved]}
full_time_pst_dict = {i: [0, 0, 0, 0.0] for i in set(FULL_TIME_POSITION)}

# Total applicant counts
total_applicants = 0

# Counts the number of approved and number of rejection for full-time and not␣
 ↪full-time positions.
for idx in range(0, len(CASE_STATUS)):

    # Extract the full-time position info and case status
    ft_pst, c_status = FULL_TIME_POSITION[idx], CASE_STATUS[idx]

    # Incement the count
    full_time_pst_dict[ft_pst][c_status] += 1
    full_time_pst_dict[ft_pst][2] += 1
    full_time_pst_dict[ft_pst][3] = full_time_pst_dict[ft_pst][1] /␣
 ↪full_time_pst_dict[ft_pst][2] * 100

    total_applicants += 1

# {'Full-time position':[Rejected #, Certified #, Total #, Percentage of␣
 ↪Certified H-1B applications]
# 'Not Full-time position':[Rejected #, Certified #, Total #, Percentage of␣
 ↪Certified H-1B applications]}
print("Full-time Position Status [Y/N] : [Rejected #, Certified #, Total #,␣
 ↪Percentage of Certified H-1B applications]")
print(full_time_pst_dict)

# List that store total number of full-time H-1B and non-full-time H-1B␣
 ↪applicants
# [Certified Full time #, Denied Full time #, Certified non-Full time #, Denied␣
 ↪non-Full time #]
tot_ft_pt = [full_time_pst_dict["Y"][2], full_time_pst_dict["N"][2]]
```

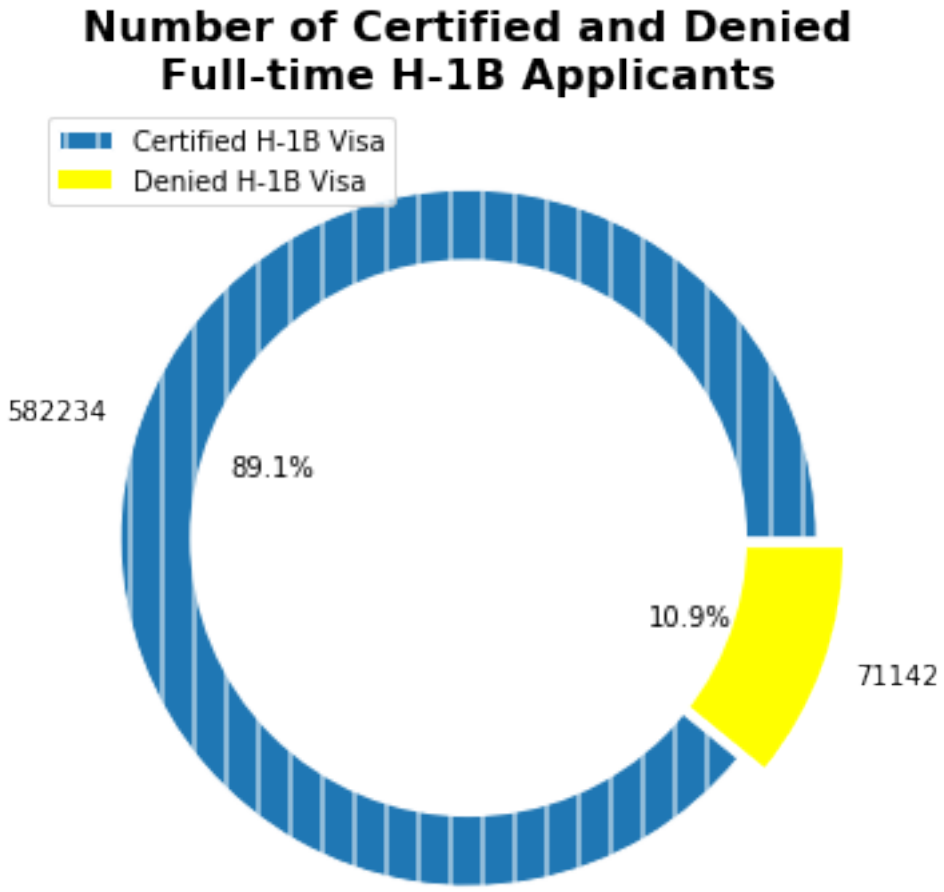
```
tot_ft_pt_labels = [str(full_time_pst_dict["Y"][2]),␣
 ↪str(full_time_pst_dict["N"][2])]

# Donut Colors
#colors = ['#FF0000', '#0000FF', '#FFFF00', '#ADFF2F', '#FFA500']
colors = ["blue", "red"]

# Explosion Donut Chart Parameters
explode = (0.00, 0.08)

# Set figure size
plt.figure(figsize=(6,6))

# Pie Chart
pie_chart1 = plt.pie(tot_ft_pt, colors = colors, labels = tot_ft_pt_labels,␣
 ↪autopct = '%1.1f%%', explode=explode)
pie_chart1[0][0].set_hatch("*")
pie_chart1[0][0].set_edgecolor("white")

# Draw a circle at the center of the pie chart
centre_circle = plt.Circle((0, 0), 0.8, fc='white')
fig = plt.gcf()

# Adding Circle in Pie chart
fig.gca().add_artist(centre_circle)

# Adding Title of chart
plt.title('Number of Full-time and\nNon-full-time H-1B Applicants',␣
 ↪fontweight='bold', fontsize = 16)

# Add Legends
plt.legend(["Full-time Position", "Not Full-time Position"], loc="upper left")

# Displaing Chart
plt.show()

# List that store total number of full-time certified H-1B and full-time denied␣
 ↪H-1B applicants
# [Full time Certified # Full time Denied #]
cert_rjt_ft = [full_time_pst_dict["Y"][1], full_time_pst_dict["Y"][0]]
cert_rjt_ft_labels = [str(full_time_pst_dict["Y"][1]),␣
 ↪str(full_time_pst_dict["Y"][0])]

# Donut Colors
#colors = ['#FF0000', '#0000FF', '#FFFF00', '#ADFF2F', '#FFA500']
colors = ['#1f77b4', "yellow"]
```

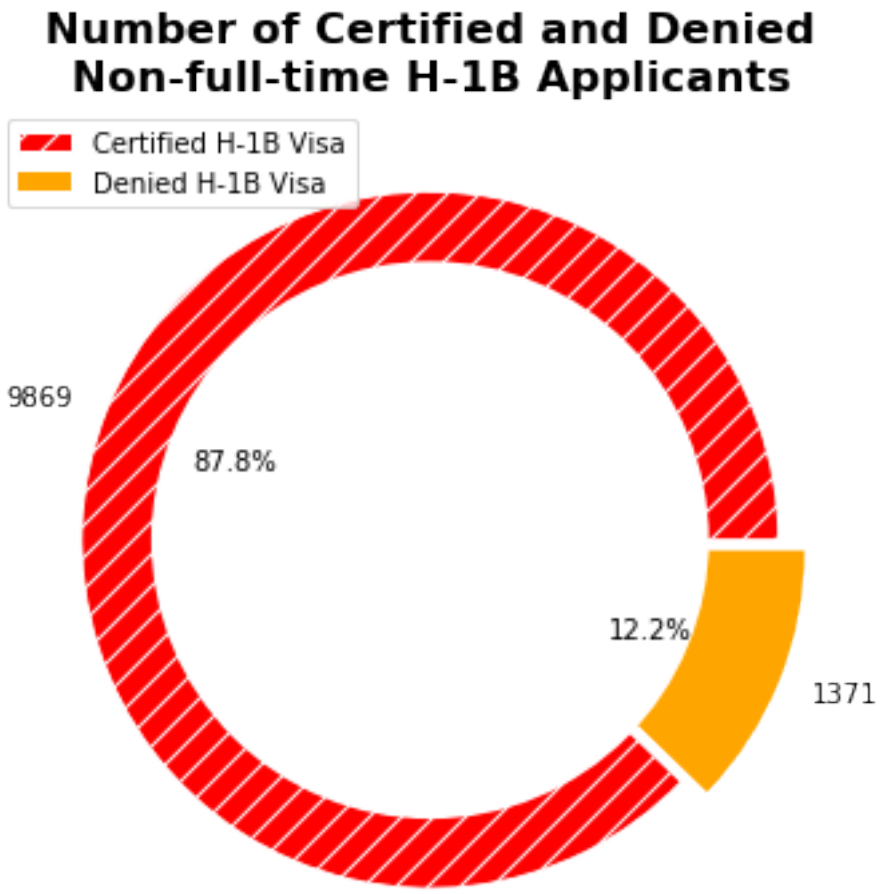
```
# Set figure size
plt.figure(figsize=(6,6))

# Pie Chart
pie_chart2 = plt.pie(cert_rjt_ft, colors = colors, labels = cert_rjt_ft_labels,␣
 ↪autopct = '%1.1f%%', explode=explode)
pie_chart2[0][0].set_hatch("|")
pie_chart2[0][0].set_edgecolor("white")

# Draw a circle at the center of the pie chart
centre_circle = plt.Circle((0, 0), 0.8, fc='white')
fig = plt.gcf()

# Adding Circle in Pie chart
fig.gca().add_artist(centre_circle)

# Adding Title of chart
plt.title('Number of Certified and Denied\nFull-time H-1B Applicants',␣
 ↪fontweight='bold', fontsize = 16)

# Add Legends
plt.legend(["Certified H-1B Visa", "Denied H-1B Visa"], loc="upper left")

# Displaing Chart
plt.show()

# List that store total number of non full-time certified H-1B and non␣
 ↪full-time denied H-1B applicants
# [non-Full time Certified # non-Full time Denied #]
cert_rjt_nft = [full_time_pst_dict["N"][1], full_time_pst_dict["N"][0]]
cert_rjt_nft_labels = [str(full_time_pst_dict["N"][1]),␣
 ↪str(full_time_pst_dict["N"][0])]

# Donut Colors
#colors = ['#FF0000', '#0000FF', '#FFFF00', '#ADFF2F', '#FFA500']
colors = ['red', "orange"]

# Set figure size
plt.figure(figsize=(6,6))

# Pie Chart
pie_chart3 = plt.pie(cert_rjt_nft, colors = colors, labels =␣
 ↪cert_rjt_nft_labels, autopct = '%1.1f%%', explode=explode)
pie_chart3[0][0].set_hatch("//")
pie_chart3[0][0].set_edgecolor("white")

# Draw a circle at the center of the pie chart
```

```
centre_circle = plt.Circle((0, 0), 0.8, fc='white')
fig = plt.gcf()

# Adding Circle in Pie chart
fig.gca().add_artist(centre_circle)

# Adding Title of chart
plt.title('Number of Certified and Denied\nNon-full-time H-1B Applicants',␣
 ↪fontweight='bold', fontsize = 16)

# Add Legends
plt.legend(["Certified H-1B Visa", "Denied H-1B Visa"], loc="upper left")

# Displaing Chart
plt.show()
```

```
Full-position Statistics:
Counter({'Y': 653376, 'N': 11240})


Full-time Position Status [Y/N] : [Rejected #, Certified #, Total #, Percentage
of Certified H-1B applications]
{'N': [1371, 9869, 11240, 87.80249110320285], 'Y': [71142, 582234, 653376,
89.1116294446077]}
```


Number of Full-time and
Non-full-time H-1B Applicants
Full-time Position
Not Full-time Position
653376
98.3%
1.7%
11240


Number of Certified and Denied
Full-time H-1B Applicants
Certified H-1B Visa
Denied H-1B Visa
582234
89.1%
10.9%
71142



Data Visualization: Prevailing Wage Level

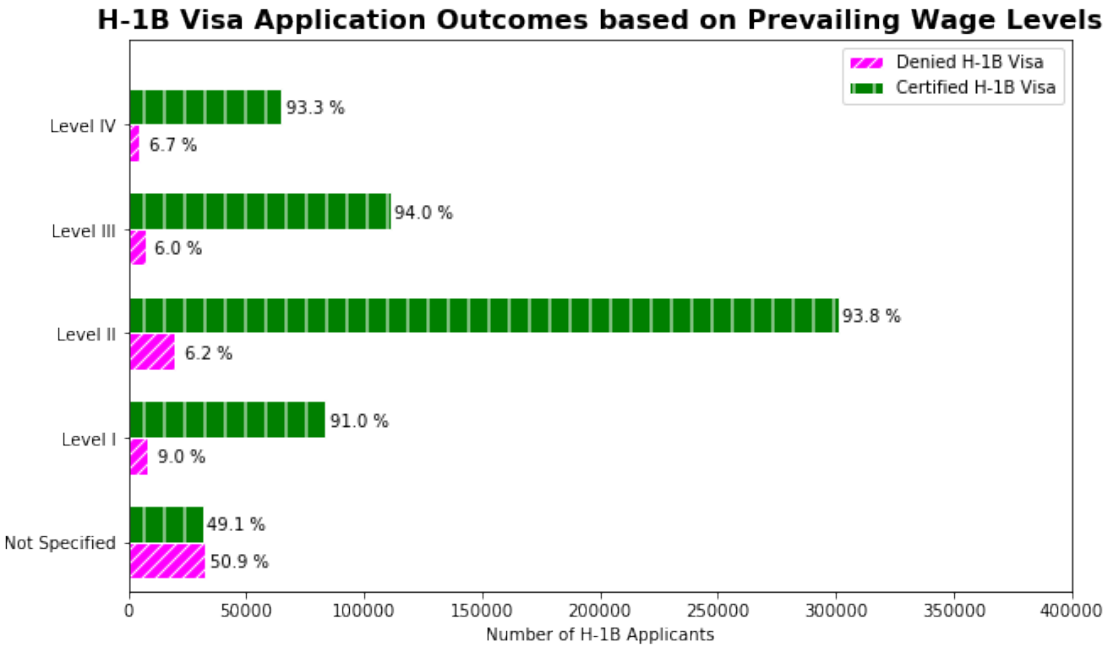
[17]:
```
# Explore PW_WAGE_LEVEL_1 Column
# Deep copy the PW_WAGE_LEVEL_1 list
PW_WAGE_LEVEL_1 = deepcopy(list(data["PW_WAGE_LEVEL_1"]))

# Loop through the PW_WAGE_LEVEL_1 column to make sure there is no blank space␣
↪in the list
for idx in range(0, len(PW_WAGE_LEVEL_1)):

    # Extract Prevailing Wage Level information
    pw_level = PW_WAGE_LEVEL_1[idx]

    if type(pw_level) == str:

        # Update the PW_WAGE_LEVEL_1
```

```
        PW_WAGE_LEVEL_1[idx] = pw_level

    else:

        # if empty, it is a N/A
        PW_WAGE_LEVEL_1[idx] = "Not Specified"

# Count how many applicants are there in different prevailing wage level
# H-1B Wage Level 1 (Entry) : $38k to $51k annual salary
# H-1B Wage Level 2 (Qualified) : $51k to $65k salary
# H-1B Wage Level 3 (Experienced): $65k to $75k salary
# H-1B Wage Level 4 (Fully Competent): $78k to $90k salary
# Source: https://www.yeklaw.com/blog/2021/april/
 ↪what-are-the-wage-levels-for-h1-b-visa-workers-/
pw_level_dict = Counter(PW_WAGE_LEVEL_1)
print("Prevailing Wage Level Information:")
print(pw_level_dict)
print("\n")

# Create a prevailing wage level count dictionary that stores approved # and␣
 ↪rejected #
# for example, {"Level I": [Rejected #, Approved#, Total # of Applicant,␣
 ↪Percentage of Approved, Percentage of Denied]
#               "Level II": [Rejected #, Approved#, Total # of Applicant,␣
 ↪Percentage of Approved, Percentage of Denied]}
pw_level_count_dict = {i: [0, 0, 0, 0.0, 0.0] for i in set(PW_WAGE_LEVEL_1)}

# Total applicant counts
total_applicants = 0

# Counts the number of approved and number of rejection for different␣
 ↪prevailing wage levels.
for idx in range(0, len(CASE_STATUS)):

    # Extract the prevailing wage level info and case status
    pw_lvl, c_status = PW_WAGE_LEVEL_1[idx], CASE_STATUS[idx]

    # Incement the count
    pw_level_count_dict[pw_lvl][c_status] += 1
    pw_level_count_dict[pw_lvl][2] += 1
    pw_level_count_dict[pw_lvl][3] = pw_level_count_dict[pw_lvl][1] /␣
 ↪pw_level_count_dict[pw_lvl][2] * 100
    pw_level_count_dict[pw_lvl][4] = pw_level_count_dict[pw_lvl][0] /␣
 ↪pw_level_count_dict[pw_lvl][2] * 100

    total_applicants += 1
```

```
# Print dictionary
print("Prevailing Wage Levels: [Rejected #, Certified #, Total #, Percentage of␣
 ↪Certified, Percentage of Denied]")
print(pw_level_count_dict)
print(pw_level_count_dict["Not Specified"][0])

# Prevailing Wage(PW) Level Labels
pw_level_labels = ["Not Specified", "Level I", "Level II", "Level III", "Level␣
 ↪IV"]
rej_num, pas_num, tot_num, pas_perc, rej_perc= [], [], [], [], []

# Append to list
for label in pw_level_labels:

    rej_num.append(pw_level_count_dict[label][0])
    pas_num.append(pw_level_count_dict[label][1])
    tot_num.append(pw_level_count_dict[label][2])
    pas_perc.append(pw_level_count_dict[label][3])
    rej_perc.append(pw_level_count_dict[label][4])

# Horizontal bar plot parameters
ind = np.arange(len(pw_level_labels))
width = 0.35

# Plot horizontal Bar
fig, ax = plt.subplots(figsize=(10,6))
ax.barh(ind, rej_num, width, color='magenta', label='Denied H-1B Visa', hatch␣
 ↪='///', edgecolor = "white")
ax.barh(ind + width, pas_num, width, color='green', label='Certified H-1B␣
 ↪Visa', hatch ='|', edgecolor = "white")
ax.set_title("H-1B Visa Application Outcomes based on Prevailing Wage Levels",␣
 ↪fontweight='bold', fontsize = 16)
ax.set_xlim([0, 400000])
ax.set_xlabel("Number of H-1B Applicants")
ax.set(yticks= ind + width / 2, yticklabels=pw_level_labels, ylim=[2*width - 1,␣
 ↪len(pw_level_labels)])
ax.legend()

# Combined into an array
rej_pas_Perc_array = np.array([rej_perc, pas_perc])
rej_pas_Perc_array = rej_pas_Perc_array.T
# Flatten the array
rej_pas_Perc_array = rej_pas_Perc_array.flatten()
```

```
# Annotate the horizontal bar with Certified and Denied H-1B Visas Percentage␣
 ↪[%]
for i, p in enumerate(ax.patches):

    # If index is less than 4
    if i <= 4:

        width = p.get_width()
        ax.annotate(f'{rej_pas_Perc_array[i*2]:0.1f} %',
                    xy=(width, p.get_y() + p.get_height() / 2),
                    xytext=(20, -6),  # 3 points horizontal offset
                    textcoords="offset points",
                    ha='center', va='bottom')

    # If more than 4
    else:

        width = p.get_width()
        ax.annotate(f'{rej_pas_Perc_array[i - (len(rej_pas_Perc_array) - 1 -␣
 ↪i)]:0.1f} %',
                    xy=(width, p.get_y() + p.get_height() / 2),
                    xytext=(20, -6),  # 3 points horizontal offset
                    textcoords="offset points",
                    ha='center', va='bottom')
```

```
Prevailing Wage Level Information:
Counter({'Level II': 320990, 'Level III': 118271, 'Level I': 91913, 'Level IV':
69395, 'Not Specified': 64047})


Prevailing Wage Levels: [Rejected #, Certified #, Total #, Percentage of
Certified, Percentage of Denied]
{'Level III': [7135, 111136, 118271, 93.96724471764, 6.032755282360004], 'Level
IV': [4678, 64717, 69395, 93.25888032278982, 6.741119677210174], 'Not
Specified': [32630, 31417, 64047, 49.053039174356336, 50.94696082564367], 'Level
I': [8248, 83665, 91913, 91.02629660657361, 8.973703393426392], 'Level II':
[19822, 301168, 320990, 93.82472974235958, 6.175270257640425]}
32630
```



Data Visualization: Statutory Basis

[18]:
```
# Explore STATUTORY_BASIS Column
# Deep copy the STATUTORY_BASIS list
STATUTORY_BASIS = deepcopy(list(data["STATUTORY_BASIS"]))

# Loop through the STATUTORY_BASIS column to make sure there is no blank space␣
 ↪in the list
for idx in range(0, len(STATUTORY_BASIS)):

    # Extract statutory basis information
    stb = STATUTORY_BASIS[idx]

    if type(stb) == str:

        # Update the STATUTORY_BASIS
        STATUTORY_BASIS[idx] = stb

    else:

        # if empty, it is a N/A
        STATUTORY_BASIS[idx] = "N/A"

# Count how many applicants have an advanced degree
stb_dict = Counter(STATUTORY_BASIS)
print("Statutory Basis Statistics:")
```

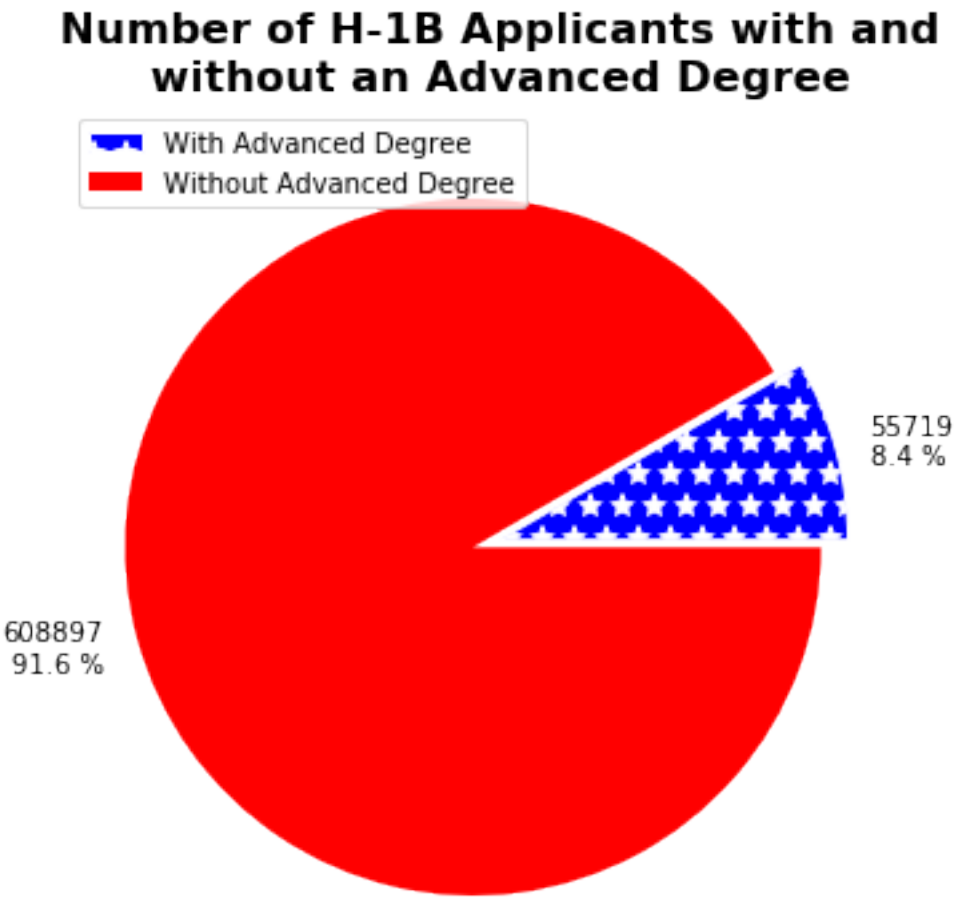
```
print(stb_dict)
print("\n")

# Convert it to whether it has advanced degree or no advanced degree like␣
 ↪masters and PhDs
for idx in range(0, len(STATUTORY_BASIS)):

    # Extract statutory basis information
    stb = STATUTORY_BASIS[idx]

    if stb == "BOTH" or  stb == "DEGREE":

        # Update the STATUTORY_BASIS
        STATUTORY_BASIS[idx] = "Y"

    else:

        # if empty, it is a N/A
        STATUTORY_BASIS[idx] = "N"

# Count how many applicants have an advanced degree
new_stb_dict = Counter(STATUTORY_BASIS)
print("Advanced Degrees:")
print(new_stb_dict)
print("\n")


# Create a advanced degree count dictionary that stores approved # and rejected␣
 ↪#
# for example, {"Y": [Rejected #, Approved#, Total # of Applicant, Percentage␣
 ↪of Approved]
#               "N": [Rejected #, Approved#, Total # of Applicant, Percentage␣
 ↪of Approved]}
new_stb_dict = {i: [0, 0, 0, 0.0] for i in set(STATUTORY_BASIS)}

# Total applicant counts
total_applicants = 0

# Counts the number of approved and number of rejection for full-time and not␣
 ↪full-time positions.
for idx in range(0, len(CASE_STATUS)):

    # Extract the advanced degree and case status
    adv_deg, c_status = STATUTORY_BASIS[idx], CASE_STATUS[idx]

    # Incement the count
    new_stb_dict[adv_deg][c_status] += 1
```

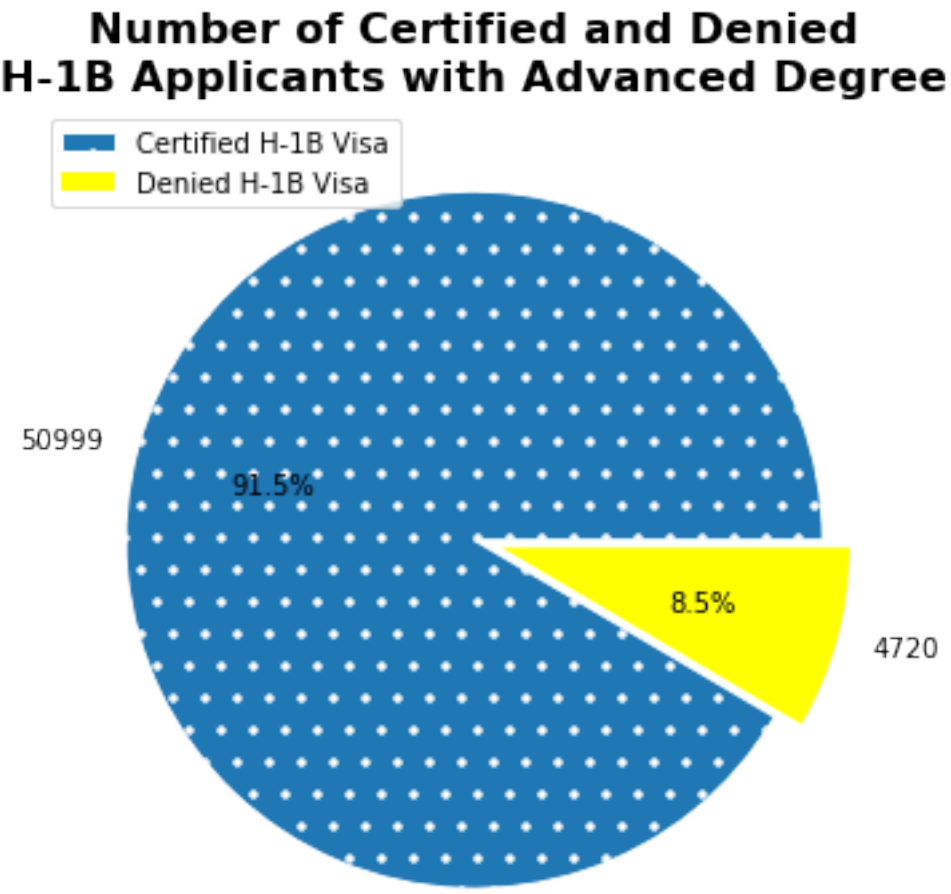
```
    new_stb_dict[adv_deg][2] += 1
    new_stb_dict[adv_deg][3] = new_stb_dict[adv_deg][1] /␣
 ↪new_stb_dict[adv_deg][2] * 100

    total_applicants += 1

# {'Advanced Degree':[Rejected #, Certified #, Total #, Percentage of Certified␣
 ↪H-1B applications]
# 'No Advanced Degre':[Rejected #, Certified #, Total #, Percentage of␣
 ↪Certified H-1B applications]}
print("Advanced Degree [Y/N] : [Rejected #, Certified #, Total #, Percentage of␣
 ↪Certified H-1B applications]")
print(new_stb_dict)

# List that store total number of H-1B applicants with advanced degree
# [Advanced Degree #, No Advanced Degree #]
tot_adv_deg = [new_stb_dict["Y"][2], new_stb_dict["N"][2]]
tot_adv_deg_labels = [str(new_stb_dict["Y"][2]) + "\n" +␣
 ↪str(round(new_stb_dict["Y"][2]/(new_stb_dict["Y"][2] +␣
 ↪new_stb_dict["N"][2])*100, 1)) + " %",
                      str(new_stb_dict["N"][2]) + "\n" +␣
 ↪str(round(new_stb_dict["N"][2]/(new_stb_dict["Y"][2] +␣
 ↪new_stb_dict["N"][2])*100, 1)) + " %"]

# Donut Colors
#colors = ['#FF0000', '#0000FF', '#FFFF00', '#ADFF2F', '#FFA500']
colors = ["blue", "red"]

# Explosion Donut Chart Parameters
explode = (0.00, 0.08)

# Set figure size
plt.figure(figsize=(6,6))

# Pie Chart
pie_chart1 = plt.pie(tot_adv_deg, colors = colors, labels = tot_adv_deg_labels,␣
 ↪explode=explode)
pie_chart1[0][0].set_hatch("*")
pie_chart1[0][0].set_edgecolor("white")

# Adding Title of chart
plt.title('Number of H-1B Applicants with and\nwithout an Advanced Degree',␣
 ↪fontweight='bold', fontsize = 16)

# Add Legends
```

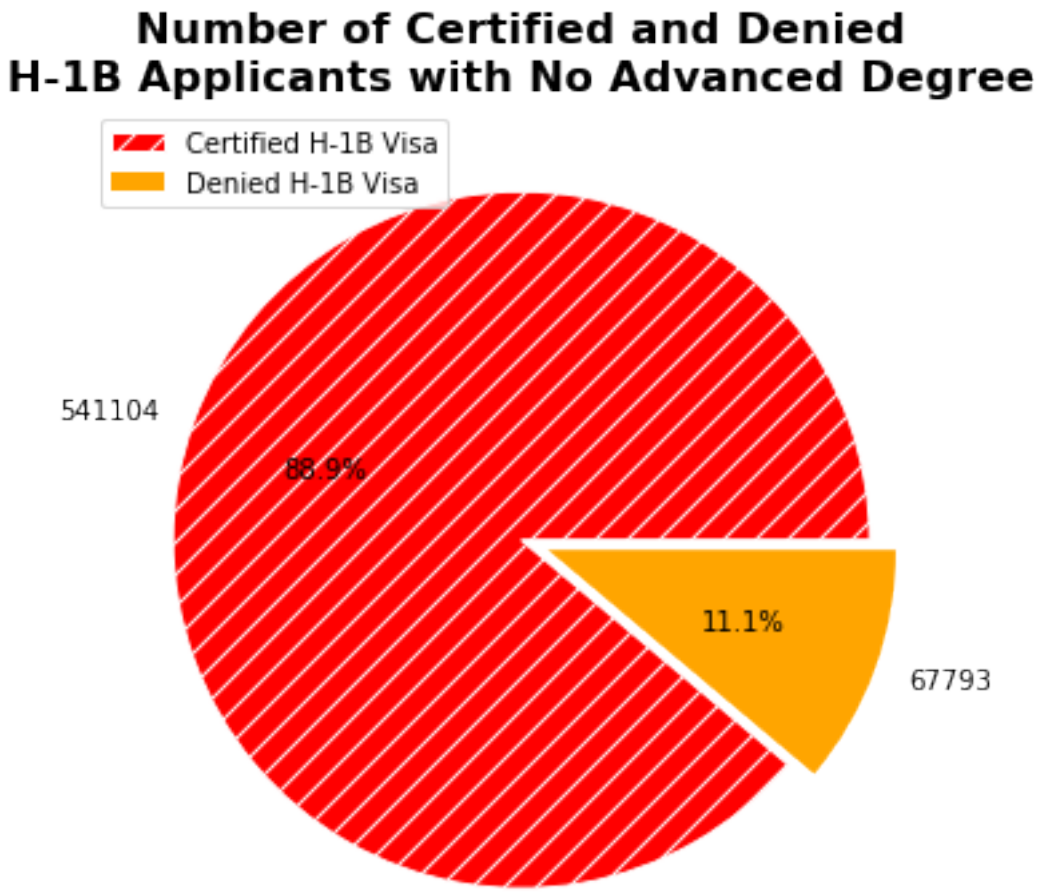
```
plt.legend(["With Advanced Degree", "Without Advanced Degree"], loc="upper␣
 ↪left")

# Displaing Chart
plt.show()

# List that store total number of advanced degree certified H-1B and no␣
 ↪advanced degree denied H-1B applicants
# [Advanced Degree Certified # Advanced Degree Denied #]
cert_rjt_adv_deg = [new_stb_dict["Y"][1], new_stb_dict["Y"][0]]
cert_rjt_adv_deg_labels = [str(new_stb_dict["Y"][1]), str(new_stb_dict["Y"][0])]

# Donut Colors
#colors = ['#FF0000', '#0000FF', '#FFFF00', '#ADFF2F', '#FFA500']
colors = ['#1f77b4', "yellow"]

# Set figure size
plt.figure(figsize=(6,6))

# Pie Chart
pie_chart2 = plt.pie(cert_rjt_adv_deg, colors = colors, labels =␣
 ↪cert_rjt_adv_deg_labels, autopct = '%1.1f%%', explode=explode)
pie_chart2[0][0].set_hatch(".")
pie_chart2[0][0].set_edgecolor("white")

# Adding Title of chart
plt.title('Number of Certified and Denied\nH-1B Applicants with Advanced␣
 ↪Degree', fontweight='bold', fontsize = 16)

# Add Legends
plt.legend(["Certified H-1B Visa", "Denied H-1B Visa"], loc="upper left")

# Displaing Chart
plt.show()

# List that store total number of certified and denied H-1B applicants with no␣
 ↪advanced degree
# [No Advanced Degree Certified #, No Advanced Degree Denied #]
cert_rjt_no_adv_deg = [new_stb_dict["N"][1], new_stb_dict["N"][0]]
cert_rjt_no_adv_deg_labels = [str(new_stb_dict["N"][1]),␣
 ↪str(new_stb_dict["N"][0])]

# Donut Colors
#colors = ['#FF0000', '#0000FF', '#FFFF00', '#ADFF2F', '#FFA500']
colors = ['red', "orange"]
```

```
# Set figure size
plt.figure(figsize=(6,6))

# Pie Chart
pie_chart3 = plt.pie(cert_rjt_no_adv_deg, colors = colors, labels =␣
 ↪cert_rjt_no_adv_deg_labels, autopct = '%1.1f%%', explode=explode)
pie_chart3[0][0].set_hatch("//")
pie_chart3[0][0].set_edgecolor("white")

# Adding Title of chart
plt.title('Number of Certified and Denied\nH-1B Applicants with No Advanced␣
 ↪Degree', fontweight='bold', fontsize = 16)

# Add Legends
plt.legend(["Certified H-1B Visa", "Denied H-1B Visa"], loc="upper left")

# Displaing Chart
plt.show()
```

```
Statutory Basis Statistics:
Counter({'N/A': 456626, 'WAGE': 152271, 'BOTH': 54427, 'DEGREE': 1292})


Advanced Degrees:
Counter({'N': 608897, 'Y': 55719})


Advanced Degree [Y/N] : [Rejected #, Certified #, Total #, Percentage of
Certified H-1B applications]
{'N': [67793, 541104, 608897, 88.86626145308648], 'Y': [4720, 50999, 55719,
91.52892191173567]}
```


Number of H-1B Applicants with and without an Advanced Degree
With Advanced Degree
Without Advanced Degree
55719
8.4 %
608897
91.6 %


Number of Certified and Denied
H-1B Applicants with Advanced Degree
Certified H-1B Visa
Denied H-1B Visa
50999
91.5%
8.5%
4720



Data Visualization: Job Title

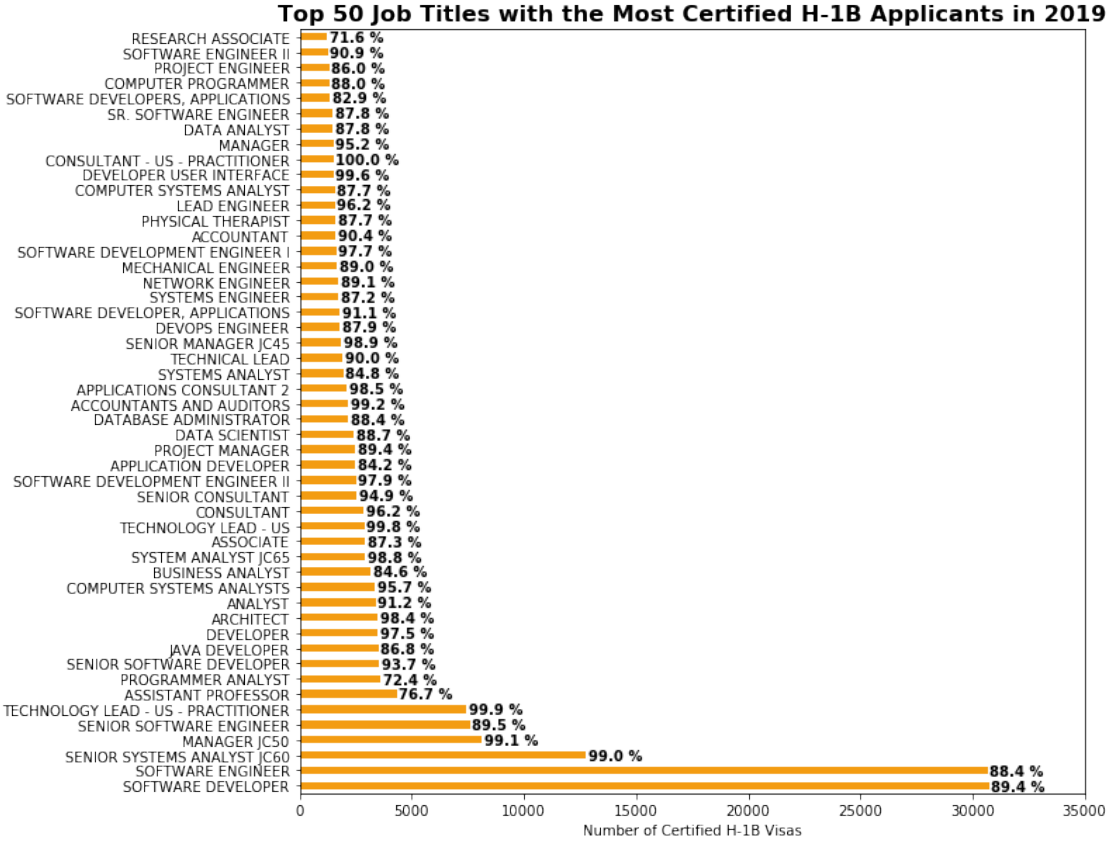
[19]:
```
# Explore JOB_TITLE Column
# Deep copy the JOB_TITLE list
JOB_TITLE = deepcopy(list(data["JOB_TITLE"]))

# Loop through the JOB_TITLE column to make sure there is no blank space in the␣
 ↪list
for idx in range(0, len(JOB_TITLE)):

    # Extract job title information
    job_title = JOB_TITLE[idx]

    if type(job_title) == str:

        # Update the JOB_TITLE
        JOB_TITLE[idx] = job_title
```

```
    else:

        # if empty, it is Blank
        JOB_TITLE[idx] = "Blank"

# Count how many applicants are there in different job titles
job_title_dict = Counter(JOB_TITLE).most_common(100)[:][:]
print("Number of Applicants for each Job Title:")
print(job_title_dict)
print("\n")


# Create a job title count dictionary that stores approved # and rejected #
# for example, {Engineer: [Rejected #, Approved#, Total # of Applicant,␣
 ↪Percentage of Approved]
#               Doctor: [Rejected #, Approved#, Total # of Applicant,␣
 ↪Percentage of Approved]...}
job_title_count_dict = {i: [0, 0, 0, 0.0] for i in set(JOB_TITLE)}

# Total applicant counts
total_applicants = 0

# Counts the number of approved and number of rejection for each job title.
for idx in range(0, len(CASE_STATUS)):

    # Extract the job title and case status
    jb_title, c_status = JOB_TITLE[idx], CASE_STATUS[idx]

    # Incement the count
    job_title_count_dict[jb_title][c_status] += 1
    job_title_count_dict[jb_title][2] += 1
    job_title_count_dict[jb_title][3] = job_title_count_dict[jb_title][1] /␣
 ↪job_title_count_dict[jb_title][2] * 100

    total_applicants += 1

# Convert to dataframe
df_job_title_count = pd.DataFrame.from_dict(job_title_count_dict,␣
 ↪orient='index')
df_job_title_count.columns = ['Number of Denied H-1B Visas', 'Number of␣
 ↪Certified H-1B Visas', 'Total H-1B Applicants in the State', 'Certified H-1B␣
 ↪Visas Percentage [%]']

# Sorted dataframe from largest to smallest
sorted_df = df_job_title_count['Number of Certified H-1B Visas'].
 ↪sort_values(ascending=False)
```

```
jb_cert_perc = df_job_title_count.loc[sorted_df.index[:]]['Certified H-1B Visas␣
↪Percentage [%]']

# Plot horizontal bar
plt.figure(figsize=(10,10))
ax1 = df_job_title_count['Number of Certified H-1B Visas'].
↪sort_values(ascending=False).head(50).plot(kind='barh', color='#F39C12')
ax1.set_ylabel("")
ax1.set_title('Top 50 Job Titles with the Most Certified H-1B Applicants in␣
↪2019', fontweight='bold', fontsize = 16)
ax1.set_xlim([0, 35000])
ax1.set_xlabel("Number of Certified H-1B Visas")

# Annotate the horizontal bar with Certified H-1B Visas Percentage [%]
for i, num_cert in enumerate(sorted_df[:50]):

    ax1.text(num_cert + 100, i - 0.3, f'{jb_cert_perc[i]:0.1f} %',␣
↪color='black', fontweight='bold')

plt.show()
```

```
Number of Applicants for each Job Title:
[('SOFTWARE ENGINEER', 34691), ('SOFTWARE DEVELOPER', 34366), ('SENIOR SYSTEMS
ANALYST JC60', 12897), ('SENIOR SOFTWARE ENGINEER', 8482), ('MANAGER JC50',
8205), ('TECHNOLOGY LEAD - US - PRACTITIONER', 7434), ('ASSISTANT PROFESSOR',
5632), ('PROGRAMMER ANALYST', 4983), ('JAVA DEVELOPER', 4062), ('SENIOR SOFTWARE
DEVELOPER', 3800), ('BUSINESS ANALYST', 3754), ('ANALYST', 3724), ('DEVELOPER',
3580), ('ARCHITECT', 3540), ('COMPUTER SYSTEMS ANALYSTS', 3518), ('ASSOCIATE',
3335), ('CONSULTANT', 2975), ('SYSTEM ANALYST JC65', 2956), ('APPLICATION
DEVELOPER', 2944), ('TECHNOLOGY LEAD - US', 2888), ('PROJECT MANAGER', 2771),
('DATA SCIENTIST', 2701), ('SENIOR CONSULTANT', 2670), ('SOFTWARE DEVELOPMENT
ENGINEER II', 2572), ('DATABASE ADMINISTRATOR', 2466), ('SYSTEMS ANALYST',
2306), ('ACCOUNTANTS AND AUDITORS', 2193), ('APPLICATIONS CONSULTANT 2', 2147),
('TECHNICAL LEAD', 2120), ('DEVOPS ENGINEER', 2033), ('SYSTEMS ENGINEER', 1983),
('SOFTWARE DEVELOPER, APPLICATIONS', 1954), ('NETWORK ENGINEER', 1925),
('MECHANICAL ENGINEER', 1874), ('SENIOR MANAGER JC45', 1857), ('PHYSICAL
THERAPIST', 1814), ('ACCOUNTANT', 1786), ('COMPUTER SYSTEMS ANALYST', 1776),
('RESEARCH ASSOCIATE', 1723), ('DATA ANALYST', 1685), ('SR. SOFTWARE ENGINEER',
1671), ('SOFTWARE DEVELOPMENT ENGINEER I', 1667), ('LEAD ENGINEER', 1636),
('SOFTWARE DEVELOPERS, APPLICATIONS', 1617), ('MANAGER', 1578), ('PROJECT
ENGINEER', 1540), ('DEVELOPER USER INTERFACE', 1523), ('COMPUTER PROGRAMMER',
1511), ('CONSULTANT - US - PRACTITIONER', 1505), ('VICE PRESIDENT', 1407),
('SOFTWARE ENGINEER II', 1383), ('DATA ENGINEER', 1368), ('SOFTWARE DEVELOPMENT
ENGINEER', 1366), ('BUSINESS SYSTEMS ANALYST', 1341), ('FINANCIAL ANALYST',
1289), ('RESEARCH SCIENTIST', 1265), ('SALESFORCE DEVELOPER', 1230), ('.NET
DEVELOPER', 1211), ('ELECTRICAL ENGINEER', 1208), ('QUALITY ENGINEER', 1176),
('DESIGN ENGINEER', 1173), ('PRODUCT MANAGER', 1100), ('ENGINEER', 1098),
```

('POSTDOCTORAL FELLOW', 1075), ('SOFTWARE ENGINEER III', 1066), ('SOFTWARE APPLICATION DEVELOPER', 1040), ('CONSULTANT - US', 1016), ('SENIOR ENGINEER', 1014), ('SYSTEMS ANALYST JC65', 998), ('TECHNICAL TEST LEAD - US - PRACTITIONER', 986), ('APPLICATIONS CONSULTANT 3', 969), ('STAFF ACCOUNTANT', 956), ('TECHNICAL ARCHITECT', 955), ('POSTDOCTORAL RESEARCH ASSOCIATE', 934), ('PROGRAMMER ANALYST - II', 922), ('TECHNOLOGY LEAD - US - PROFESSIONAL', 916), ('TEST CONSULTANT 2', 910), ('ETL DEVELOPER', 903), ('RESEARCH FELLOW', 887), ('STAFF SOFTWARE ENGINEER', 875), ('SOLUTION ARCHITECT', 867), ('SENIOR BUSINESS ANALYST', 860), ('BUSINESS INTELLIGENCE ANALYST', 853), ('SOLUTIONS ARCHITECT', 851), ('TECHNICAL CONSULTANT', 851), ('MEMBER OF TECHNICAL STAFF', 833), ('COMPUTER SYSTEMS ANALYST 2', 833), ('LEAD SOFTWARE ENGINEER', 828), ('POSTDOCTORAL ASSOCIATE', 815), ('SOFTWARE QUALITY ASSURANCE ENGINEER', 792), ('SR. SOFTWARE DEVELOPER', 789), ('SYSTEM ADMINISTRATOR', 787), ('PRINCIPAL SOFTWARE ENGINEER', 783), ('PROCESS ENGINEER', 781), ('BUSINESS SYSTEMS ANALYST 2', 781), ('IT PROJECT MANAGER', 776), ('SOFTWARE TEST ENGINEER', 771), ('SENIOR DEVELOPER', 769), ('TEST ENGINEER', 766), ('HOSPITALIST', 754)]



Data Visualization: Standard Occupational Classification (SOC) Title

[20]:
```
# Explore SOC_TITLE Column
# Deep copy the SOC_TITLE list
SOC_TITLE = deepcopy(list(data["SOC_TITLE"]))

# Loop through the SOC_TITLE column to make sure there is no blank space in the␣
 ↪list
for idx in range(0, len(SOC_TITLE)):

    # Extract soc title information
    soc_title = SOC_TITLE[idx]

    if type(soc_title) == str:

        # Update the SOC_TITLE
        SOC_TITLE[idx] = soc_title

    else:

        # if empty, it is Blank
        SOC_TITLE[idx] = "Blank"

# Count how many applicants are there in different soc titles
soc_title_dict = Counter(SOC_TITLE).most_common()[:][:]
print("Number of Applicants for each SOC Title:")
print(soc_title_dict)
print("\n")


# Create a soc title count dictionary that stores approved # and rejected #
# for example, {Engineer: [Rejected #, Approved#, Total # of Applicant,␣
 ↪Percentage of Approved]
#               Doctor: [Rejected #, Approved#, Total # of Applicant,␣
 ↪Percentage of Approved]...}
soc_title_count_dict = {i: [0, 0, 0, 0.0] for i in set(SOC_TITLE)}

# Total applicant counts
total_applicants = 0

# Counts the number of approved and number of rejection for each job title.
for idx in range(0, len(CASE_STATUS)):

    # Extract the soc title and case status
    sc_title, c_status = SOC_TITLE[idx], CASE_STATUS[idx]

    # Incement the count
    soc_title_count_dict[sc_title][c_status] += 1
    soc_title_count_dict[sc_title][2] += 1
```

```
    soc_title_count_dict[sc_title][3] = soc_title_count_dict[sc_title][1] /␣
 ↪soc_title_count_dict[sc_title][2] * 100

    total_applicants += 1

print(soc_title_count_dict["SOFTWARE DEVELOPERS, APPLICATIONS"])

# Convert to dataframe
df_soc_title_count = pd.DataFrame.from_dict(soc_title_count_dict,␣
 ↪orient='index')
df_soc_title_count.columns = ['Number of Denied H-1B Visas', 'Number of␣
 ↪Certified H-1B Visas', 'Total H-1B Applicants', 'Certified H-1B Visas␣
 ↪Percentage [%]']

# Sorted dataframe from largest to smallest
sorted_df = df_soc_title_count['Total H-1B Applicants'].
 ↪sort_values(ascending=False)
sc_cert_perc = df_soc_title_count.loc[sorted_df.index[:]]['Certified H-1B Visas␣
 ↪Percentage [%]']
```

```
Number of Applicants for each SOC Title:
[('SOFTWARE DEVELOPERS, APPLICATIONS', 216551), ('COMPUTER OCCUPATIONS, ALL
OTHER', 56626), ('COMPUTER SYSTEMS ANALYST', 40870), ('COMPUTER SYSTEMS
ANALYSTS', 32096), ('SOFTWARE DEVELOPERS, SYSTEMS SOFTWARE', 31724), ('COMPUTER
PROGRAMMERS', 17128), ('OPERATIONS RESEARCH ANALYSTS', 11949), ('MECHANICAL
ENGINEERS', 11698), ('ACCOUNTANTS AND AUDITORS', 10990), ('MANAGEMENT ANALYSTS',
10279), ('FINANCIAL ANALYSTS', 9820), ('STATISTICIANS', 9567), ('COMPUTER AND
INFORMATION SYSTEMS MANAGERS', 9455), ('DATABASE ADMINISTRATORS', 9114),
('ELECTRICAL ENGINEERS', 8308), ('NETWORK AND COMPUTER SYSTEMS ADMINISTRATORS',
7039), ('MARKET RESEARCH ANALYSTS AND MARKETING SPECIALISTS', 6569),
('ELECTRONICS ENGINEERS, EXCEPT COMPUTER', 6487), ('MEDICAL SCIENTISTS, EXCEPT
EPIDEMIOLOGISTS', 5672), ('PHYSICIANS AND SURGEONS, ALL OTHER', 5456),
('INDUSTRIAL ENGINEERS', 5403), ('CIVIL ENGINEERS', 4987), ('INFORMATION
SECURITY ANALYSTS', 4797), ('COMPUTER AND INFORMATION RESEARCH SCIENTISTS',
4407), ('ENGINEERS, ALL OTHER', 4075), ('BIOLOGICAL SCIENTISTS, ALL OTHER',
3964), ('FINANCIAL SPECIALISTS, ALL OTHER', 3930), ('INFORMATION TECHNOLOGY
PROJECT MANAGERS', 3438), ('BIOCHEMISTS AND BIOPHYSICISTS', 3328), ('COMPUTER
NETWORK ARCHITECTS', 3221), ('MARKETING MANAGERS', 3004), ('SALES ENGINEERS',
2896), ('CHEMISTS', 2444), ('HEALTH SPECIALTIES TEACHERS, POSTSECONDARY', 2316),
('GRAPHIC DESIGNERS', 2242), ('WEB DEVELOPERS', 2113), ('PHYSICAL THERAPISTS',
2047), ('LOGISTICIANS', 2013), ('SOFTWARE QUALITY ASSURANCE ENGINEERS AND
TESTERS', 1996), ('LAWYERS', 1972), ('FINANCIAL MANAGERS', 1900),
('ARCHITECTURAL AND ENGINEERING MANAGERS', 1867), ('GENERAL AND OPERATIONS
MANAGERS', 1680), ('COMPUTER SYSTEMS ENGINEERS/ARCHITECTS', 1644), ('ARCHITECTS,
EXCEPT LANDSCAPE AND NAVAL', 1522), ('COMPUTER HARDWARE ENGINEERS', 1469),
('BIOMEDICAL ENGINEERS', 1443), ('BUSINESS TEACHERS, POSTSECONDARY', 1405),
('MATERIALS ENGINEERS', 1397), ('ECONOMISTS', 1376), ('COMMERCIAL AND INDUSTRIAL
```

DESIGNERS', 1291), ('COMPLIANCE OFFICERS', 1272), ('CHEMICAL ENGINEERS', 1259), ('ENGINEERING TEACHERS, POSTSECONDARY', 1246), ('INTERNISTS, GENERAL', 1208), ('DENTISTS, GENERAL', 1206), ('PUBLIC RELATIONS SPECIALISTS', 1176), ('HUMAN RESOURCES SPECIALISTS', 1166), ('PHYSICISTS', 1089), ('MANAGERS, ALL OTHER', 1032), ('CONSTRUCTION MANAGERS', 1031), ('ELEMENTARY SCHOOL TEACHERS, EXCEPT SPECIAL EDUCATION', 997), ('BIOLOGICAL TECHNICIANS', 995), ('ARCHITECTURAL AND CIVIL DRAFTERS', 946), ('MEDICAL AND CLINICAL LABORATORY TECHNOLOGISTS', 943), ('COST ESTIMATORS', 941), ('COMPUTER SCIENCE TEACHERS, POSTSECONDARY', 923), ('SECONDARY SCHOOL TEACHERS, EXCEPT SPECIAL AND CAREER/TECHNICAL EDUCATION', 888), ('MATHEMATICAL SCIENCE TEACHERS, POSTSECONDARY', 848), ('MATERIALS SCIENTISTS', 800), ('SOFTWARE DEVELOPERS, APPLICATIONS, NON R&D', 790), ('SALES MANAGERS', 788), ('MULTIMEDIA ARTISTS AND ANIMATORS', 760), ('FAMILY AND GENERAL PRACTITIONERS', 679), ('TRAINING AND DEVELOPMENT SPECIALISTS', 655), ('ACTUARIES', 653), ('NATURAL SCIENCES MANAGERS', 645), ('FOOD SCIENTISTS AND TECHNOLOGISTS', 642), ('INDUSTRIAL PRODUCTION MANAGERS', 633), ('BUSINESS OPERATIONS SPECIALISTS, ALL OTHER', 622), ('MICROBIOLOGISTS', 619), ('TRANSPORTATION, STORAGE, AND DISTRIBUTION MANAGERS', 618), ('HEALTHCARE PRACTITIONERS AND TECHNICAL WORKERS, ALL OTHER', 613), ('PHARMACISTS', 612), ('ENVIRONMENTAL ENGINEERS', 594), ('MEDICAL AND HEALTH SERVICES MANAGERS', 592), ('MATHEMATICIANS', 564), ('INTERIOR DESIGNERS', 554), ('INSTRUCTIONAL COORDINATORS', 538), ('BUDGET ANALYSTS', 528), ('CHIEF EXECUTIVES', 523), ('FOREIGN LANGUAGE AND LITERATURE TEACHERS, POSTSECONDARY', 497), ('SPEECH-LANGUAGE PATHOLOGISTS', 469), ('ECONOMICS TEACHERS, POSTSECONDARY', 466), ('CREDIT ANALYSTS', 464), ('SOIL AND PLANT SCIENTISTS', 443), ('PEDIATRICIANS, GENERAL', 410), ('GEOSCIENTISTS, EXCEPT HYDROLOGISTS AND GEOGRAPHERS', 407), ('COMPUTER NETWORK SUPPORT SPECIALISTS', 404), ('PSYCHIATRISTS', 404), ('BIOLOGICAL SCIENCE TEACHERS, POSTSECONDARY', 397), ('SOCIAL SCIENCE RESEARCH ASSISTANTS', 395), ('SOFTWARE DEVELOPERS, APPLICATIONS, R&D', 386), ('HEALTH DIAGNOSING AND TREATING PRACTITIONERS, ALL OTHER', 386), ('ADVERTISING AND PROMOTIONS MANAGERS', 376), ('VETERINARIANS', 369), ('NETWORK AND COMPUTER SYSTEMS ADMINISTRATOR', 367), ('HUMAN RESOURCES MANAGERS', 344), ('LANDSCAPE ARCHITECTS', 341), ('OCCUPATIONAL THERAPISTS', 338), ('MIDDLE SCHOOL TEACHERS, EXCEPT SPECIAL AND CAREER/TECHNICAL EDUCATION', 337), ('ART DIRECTORS', 336), ('TECHNICAL WRITERS', 329), ('MANAGEMENT ANALYST', 327), ('ART, DRAMA, AND MUSIC TEACHERS, POSTSECONDARY', 324), ('JUDICIAL LAW CLERKS', 323), ('EDITORS', 321), ('AEROSPACE ENGINEERS', 313), ('PRODUCERS AND DIRECTORS', 312), ('PURCHASING MANAGERS', 305), ('FASHION DESIGNERS', 280), ('ENVIRONMENTAL SCIENTISTS AND SPECIALISTS, INCLUDING HEALTH', 278), ('EDUCATION ADMINISTRATORS, POSTSECONDARY', 276), ('SECURITIES, COMMODITIES, AND FINANCIAL SERVICES SALES AGENTS', 273), ('PETROLEUM ENGINEERS', 272), ('TEACHERS AND INSTRUCTORS, ALL OTHER', 270), ('URBAN AND REGIONAL PLANNERS', 261), ('PARALEGALS AND LEGAL ASSISTANTS', 251), ('COMMUNICATIONS TEACHERS, POSTSECONDARY', 250), ('PHYSICS TEACHERS, POSTSECONDARY', 244), ('COACHES AND SCOUTS', 244), ('COMPUTER USER SUPPORT SPECIALISTS', 240), ('MECHANICAL ENGINEERS, R&D', 240), ('CHEMISTRY TEACHERS, POSTSECONDARY', 237), ('ELECTRICAL ENGINEERS, R&D', 235), ('EDUCATIONAL, GUIDANCE, SCHOOL, AND VOCATIONAL COUNSELORS', 222), ('MEETING, CONVENTION, AND EVENT PLANNERS', 217), ('MENTAL HEALTH COUNSELORS', 217), ('ENGINEERING TECHNICIANS, EXCEPT DRAFTERS, ALL OTHER', 216), ('POLITICAL

SCIENCE TEACHERS, POSTSECONDARY', 216), ('SOCIAL AND COMMUNITY SERVICE MANAGERS', 213), ('REPORTERS AND CORRESPONDENTS', 210), ('STATISTICAL ASSISTANTS', 209), ('SURVEY RESEARCHERS', 207), ('EDUCATION TEACHERS, POSTSECONDARY', 201), ('SOFTWARE DEVELOPER, APPLICATIONS', 198), ('HEALTH AND SAFETY ENGINEERS, EXCEPT MINING SAFETY ENGINEERS AND INSPECTORS', 186), ('PSYCHOLOGY TEACHERS, POSTSECONDARY', 186), ('PUBLIC RELATIONS AND FUNDRAISING MANAGERS', 183), ('COMPUTER PROGRAMMER', 183), ('PURCHASING AGENTS, EXCEPT WHOLESALE, RETAIL, AND FARM PRODUCTS', 180), ('INTERPRETERS AND TRANSLATORS', 180), ('REHABILITATION COUNSELORS', 176), ('PHYSICAL SCIENTISTS, ALL OTHER', 174), ('POSTSECONDARY TEACHERS, ALL OTHER', 169), ('PHILOSOPHY AND RELIGION TEACHERS, POSTSECONDARY', 165), ('EPIDEMIOLOGISTS', 162), ('DESIGNERS, ALL OTHER', 161), ('COMPUTER SYSTEMS ANALYSTS, NON R&D', 160), ('SURGEONS', 155), ('ASTRONOMERS', 153), ('KINDERGARTEN TEACHERS, EXCEPT SPECIAL EDUCATION', 145), ('SOCIAL SCIENTISTS AND RELATED WORKERS, ALL OTHER', 141), ('SOCIOLOGISTS', 139), ('WRITERS AND AUTHORS', 139), ('ENGLISH LANGUAGE AND LITERATURE TEACHERS, POSTSECONDARY', 134), ('EXERCISE PHYSIOLOGISTS', 134), ('HEALTH EDUCATORS', 133), ('REGISTERED NURSES', 133), ('OPTOMETRISTS', 131), ('ARCHITECTURE TEACHERS, POSTSECONDARY', 129), ('TRAINING AND DEVELOPMENT MANAGERS', 126), ('ANIMAL SCIENTISTS', 126), ('SECONDARY SCHOOL TEACHERS, EXCEPT SPECIAL AND', 125), ('SALES REPRESENTATIVES, WHOLESALE AND MANUFACTURING, TECHNICAL AND SCIENTIFIC PRODUCTS', 122), ('CIVIL ENGINEERS, R&D', 119), ('MENTAL HEALTH AND SUBSTANCE ABUSE SOCIAL WORKERS', 119), ('AGRICULTURAL SCIENCES TEACHERS, POSTSECONDARY', 119), ('HISTORY TEACHERS, POSTSECONDARY', 118), ('OBSTETRICIANS AND GYNECOLOGISTS', 118), ('ELEMENTARY SCHOOL TEACHERS, EXCEPT SPECIAL', 115), ('FINANCIAL EXAMINERS', 114), ('SOFTWARE DEVELOPERS, SYSTEMS SOFTWARE, NON R&D', 112), ('PSYCHOLOGISTS, ALL OTHER', 112), ('CURATORS', 112), ('DENTISTS, ALL OTHER SPECIALISTS', 112), ('CLINICAL, COUNSELING, AND SCHOOL PSYCHOLOGISTS', 111), ('SPECIAL EDUCATION TEACHERS, KINDERGARTEN AND ELEMENTARY SCHOOL', 111), ('COMPENSATION, BENEFITS, AND JOB ANALYSIS SPECIALISTS', 110), ('MINING AND GEOLOGICAL ENGINEERS, INCLUDING MINING SAFETY ENGINEERS', 110), ('ATMOSPHERIC AND SPACE SCIENTISTS', 109), ('AIRLINE PILOTS, COPILOTS, AND FLIGHT ENGINEERS', 108), ('LODGING MANAGERS', 107), ('PRESCHOOL TEACHERS, EXCEPT SPECIAL EDUCATION', 107), ('PERSONAL FINANCIAL ADVISORS', 104), ('MEDIA AND COMMUNICATION WORKERS, ALL OTHER', 103), ('NURSE PRACTITIONERS', 103), ('AGRICULTURAL ENGINEERS', 102), ('MARINE ENGINEERS AND NAVAL ARCHITECTS', 101), ('GEOGRAPHY TEACHERS, POSTSECONDARY', 96), ('RECREATION AND FITNESS STUDIES TEACHERS, POSTSECONDARY', 96), ('LIFE SCIENTISTS, ALL OTHER', 95), ('AREA, ETHNIC, AND CULTURAL STUDIES TEACHERS, POSTSECONDARY', 95), ('FILM AND VIDEO EDITORS', 95), ('ANESTHESIOLOGISTS', 94), ('HEALTHCARE SOCIAL WORKERS', 91), ('OCCUPATIONAL HEALTH AND SAFETY SPECIALISTS', 91), ('SOCIOLOGY TEACHERS, POSTSECONDARY', 89), ('SPECIAL EDUCATION TEACHERS, SECONDARY SCHOOL', 89), ('DIETITIANS AND NUTRITIONISTS', 88), ('SOFTWARE DEVELOPERS, SYSTEMS SOFTWARE, R&D', 85), ('LEGAL SUPPORT WORKERS, ALL OTHER', 82), ('ADMINISTRATIVE SERVICES MANAGERS', 81), ('FOREIGN LANGUAGE AND LITERATURE TEACHERS,', 79), ('LIFE, PHYSICAL, AND SOCIAL SCIENCE TECHNICIANS, ALL OTHER', 77), ('FARMERS, RANCHERS, AND OTHER AGRICULTURAL MANAGERS', 76), ('ENVIRONMENTAL SCIENCE TEACHERS, POSTSECONDARY', 76), ('GEOGRAPHERS', 75), ('BUSINESS INTELLIGENCE ANALYSTS', 73), ('HYDROLOGISTS', 73), ('SELF-ENRICHMENT EDUCATION TEACHERS', 73),

('INSURANCE UNDERWRITERS', 72), ('POLITICAL SCIENTISTS', 70), ('ATMOSPHERIC, EARTH, MARINE, AND SPACE SCIENCES TEACHERS, POSTSECONDARY', 69), ('NURSING INSTRUCTORS AND TEACHERS, POSTSECONDARY', 66), ('CHEFS AND HEAD COOKS', 66), ('ZOOLOGISTS AND WILDLIFE BIOLOGISTS', 65), ('MOLECULAR AND CELLULAR BIOLOGISTS', 65), ('FOOD SERVICE MANAGERS', 63), ('COMPUTER PROGRAMMERS, NON R&D', 62), ('SALES REPRESENTATIVES, SERVICES, ALL OTHER', 62), ('CARTOGRAPHERS AND PHOTOGRAMMETRISTS', 61), ('SPECIAL EDUCATION TEACHERS, MIDDLE SCHOOL', 61), ('HEALTHCARE PRACTITIONERS AND TECHNICAL WORKERS,', 60), ('EDUCATION ADMINISTRATORS, ELEMENTARY AND SECONDARY SCHOOL', 59), ('LIBRARIANS', 59), ('PHYSICIAN ASSISTANTS', 57), ('ATHLETIC TRAINERS', 57), ('FUNDRAISERS', 56), ('VALIDATION ENGINEERS', 56), ('INDUSTRIAL-ORGANIZATIONAL PSYCHOLOGISTS', 56), ('CLERGY', 55), ('SOCIAL WORK TEACHERS, POSTSECONDARY', 55), ('CHILD, FAMILY, AND SCHOOL SOCIAL WORKERS', 54), ('CIVIL ENGINEERING TECHNICIANS', 53), ('MUSIC DIRECTORS AND COMPOSERS', 53), ('MATHEMATICAL SCIENCE OCCUPATIONS, ALL OTHER', 52), ('GEOLOGICAL AND PETROLEUM TECHNICIANS', 52), ('EDUCATION ADMINISTRATORS, ALL OTHER', 49), ('NUCLEAR ENGINEERS', 48), ('MECHANICAL DRAFTERS', 48), ('LAW TEACHERS, POSTSECONDARY', 48), ('MANUFACTURING ENGINEERS', 47), ('SOCIAL SCIENCES TEACHERS, POSTSECONDARY, ALL OTHER', 46), ('SALES REPRESENTATIVES, WHOLESALE AND MANUFACTURING, EXCEPT TECHNICAL AND SCIENTIFIC PRODUCTS', 46), ('WHOLESALE AND RETAIL BUYERS, EXCEPT FARM PRODUCTS', 45), ('MIDDLE SCHOOL TEACHERS, EXCEPT SPECIAL AND', 45), ('RECREATION WORKERS', 45), ('EDUCATION ADMINISTRATORS, PRESCHOOL AND CHILDCARE CENTER/PROGRAM', 43), ('ELECTRONICS ENGINEERS, EXCEPT COMPUTER, R&D', 43), ('SPECIAL EDUCATION TEACHERS, ALL OTHER', 43), ('SECURITIES, COMMODITIES, AND FINANCIAL SERVICES', 42), ('AGRICULTURAL INSPECTORS', 42), ('PROPERTY, REAL ESTATE, AND COMMUNITY ASSOCIATION MANAGERS', 41), ('MARRIAGE AND FAMILY THERAPISTS', 41), ('CRIMINAL JUSTICE AND LAW ENFORCEMENT TEACHERS, POSTSECONDARY', 41), ('AGENTS AND BUSINESS MANAGERS OF ARTISTS, PERFORMERS, AND ATHLETES', 38), ('ANTHROPOLOGY AND ARCHEOLOGY TEACHERS, POSTSECONDARY', 37), ('RECREATIONAL THERAPISTS', 37), ('MUSEUM TECHNICIANS AND CONSERVATORS', 36), ('SALES ENGINEER', 36), ('DATABASE ARCHITECTS', 35), ('BIOINFORMATICS SCIENTISTS', 35), ('AIRLINE PILOTS, CO PILOTS AND FLIGHT ENGINEERS', 35), ('COMPUTER SYSTEMS ANALYSTS, R&D', 34), ('HEALTH AND SAFETY ENGINEERS, EXCEPT MINING SAFETY', 34), ('BOOKKEEPING, ACCOUNTING, AND AUDITING CLERKS', 34), ('COMPUTER SYSTEMS ENGINEERS', 33), ('DIRECTORS, RELIGIOUS ACTIVITIES AND EDUCATION', 32), ('ADULT BASIC AND SECONDARY EDUCATION AND LITERACY TEACHERS AND INSTRUCTORS', 32), ('CHIROPRACTORS', 32), ('COMPENSATION AND BENEFITS MANAGERS', 31), ('SURVEYORS', 31), ('MECHATRONICS ENGINEERS', 31), ('ELEMENTARY SCHOOL TEACHERS, EXCEPT SPECIAL EDUCATI', 31), ('SECONDARY SCHOOL TEACHERS', 30), ('ACCOUNTANTS', 29), ('FINANCIAL QUANTITATIVE ANALYSTS', 29), ('SQA ENGINEERS AND TESTERS', 29), ('MECHANICAL ENGINEERING TECHNICIANS', 29), ('SUBSTANCE ABUSE AND BEHAVIORAL DISORDER COUNSELORS', 29), ('SPECIAL EDUCATION TEACHERS, PRESCHOOL', 29), ('AGRICULTURAL AND FOOD SCIENCE TECHNICIANS', 28), ('COUNSELORS, ALL OTHER', 28), ('ENGLISH LANGUAGE AND LITERATURE TEACHERS,', 28), ('COMPUTER APPLICATIONS, ALL OTHER', 27), ('ARCHIVISTS', 27), ('SALES AND RELATED WORKERS, ALL OTHER', 27), ('CREDIT COUNSELORS', 26), ('OPERATIONS RESEARCH ANALYST', 26), ('ELECTRICAL AND ELECTRONIC ENGINEERING TECHNICIANS', 26), ('GENETIC COUNSELORS', 26), ('SOCIAL WORKERS, ALL OTHER', 25), ('HEALTH TECHNOLOGISTS AND TECHNICIANS, ALL OTHER', 25), ('PURCHASING AGENTS, EXCEPT

WHOLESALE, RETAIL, AND', 24), ('AREA, ETHNIC, AND CULTURAL STUDIES TEACHERS,', 24), ('VOCATIONAL EDUCATION TEACHERS, POSTSECONDARY', 24), ('SALES REPRESENTATIVES, WHOLESALE AND', 24), ('LOAN OFFICERS', 23), ('ENVIRONMENTAL SCIENCE AND PROTECTION TECHNICIANS, INCLUDING HEALTH', 23), ('TRANSPORTATION INSPECTORS', 23), ('MINING AND GEOLOGICAL ENGINEERS, INCLUDING MINING', 22), ('ENVIRONMENTAL SCIENTISTS AND SPECIALISTS,', 22), ('COMMUNITY AND SOCIAL SERVICE SPECIALISTS, ALL OTHER', 22), ('ATMOSPHERIC, EARTH, MARINE, AND SPACE SCIENCES', 22), ('SECONDARY SCHOOL TEACHERS, EXCEPT SPECIAL AND CARE', 22), ('PROSTHODONTISTS', 22), ('HEALTH DIAGNOSING AND TREATING PRACTITIONERS, ALL', 22), ('MEDICAL AND CLINICAL LABORATORY TECHNICIANS', 22), ('ORTHOTISTS AND PROSTHETISTS', 22), ('FITNESS TRAINERS AND AEROBICS INSTRUCTORS', 22), ('REGULATORY AFFAIRS SPECIALISTS', 21), ('COMMUNITY HEALTH WORKERS', 21), ('EDUCATION, TRAINING, AND LIBRARY WORKERS, ALL OTHER', 21), ('SET AND EXHIBIT DESIGNERS', 21), ('ORAL AND MAXILLOFACIAL SURGEONS', 21), ('COMPUTER PROGRAMMERS, R&D', 20), ('EDUCATIONAL, GUIDANCE, SCHOOL, AND VOCATIONAL', 20), ('RECREATION AND FITNESS STUDIES TEACHERS,', 20), ('PODIATRISTS', 20), ('COMPUTER OCCUPATIONS, ALL OTHERS', 18), ('MECHANICAL ENGINEERS, NON-R&D', 18), ('CHEMICAL TECHNICIANS', 18), ('HOME ECONOMICS TEACHERS, POSTSECONDARY', 18), ('AUDIOLOGISTS', 18), ('ADVERTISING SALES AGENTS', 18), ('ENGINEERING TECHNICIANS, EXCEPT DRAFTERS, ALL', 17), ('ANTHROPOLOGISTS AND ARCHEOLOGISTS', 17), ('FARM AND HOME MANAGEMENT ADVISORS', 17), ('THERAPISTS, ALL OTHER', 17), ('CUSTOMER SERVICE REPRESENTATIVES', 17), ('OFFICE AND ADMINISTRATIVE SUPPORT WORKERS, ALL OTHER', 17), ('ELECTRICAL ENGINEERS, NON-R&D', 16), ('SOCIAL AND HUMAN SERVICE ASSISTANTS', 16), ('LIBRARY SCIENCE TEACHERS, POSTSECONDARY', 16), ('ARTISTS AND RELATED WORKERS, ALL OTHER', 16), ('MUSICIANS AND SINGERS', 16), ('EXECUTIVE SECRETARIES AND EXECUTIVE ADMINISTRATIVE ASSISTANTS', 16), ('DATA WAREHOUSING SPECIALISTS', 15), ('BIOSTATISTICIANS', 15), ('STATISTICIAN', 15), ('MATHEMATICAL TECHNICIANS', 15), ('COMPUTER HARDWARE ENGINEERS, R&D', 15), ('HISTORIANS', 15), ('PHOTOGRAPHERS', 15), ('RESIDENTIAL ADVISORS', 15), ('INSURANCE SALES AGENTS', 15), ('APPRAISERS AND ASSESSORS OF REAL ESTATE', 14), ('SOFTWARE DEVELOPERS', 14), ('FOREIGN LANGUAGE AND LITERATURE TEACHERS, POSTSECO', 14), ('COMPUTER AND INFORMATION SYSTEMS MANAGER', 13), ('LABOR RELATIONS SPECIALISTS', 13), ('TAX PREPARERS', 13), ('COMPUTER SYSTEM ANALYST', 13), ('COMPUTER OCCUPATION, ALL OTHERS', 13), ('CIVIL ENGINEERS, NON-R&D', 13), ('MECHANICAL ENGINEER', 13), ('ENVIRONMENTAL ENGINEERING TECHNICIANS', 13), ('LIFE, PHYSICAL, AND SOCIAL SCIENCE TECHNICIANS,', 13), ('FORESTRY AND CONSERVATION SCIENCE TEACHERS, POSTSECONDARY', 13), ('SPECIAL EDUCATION TEACHERS, KINDERGARTEN AND', 13), ('PHARMACY TECHNICIANS', 13), ('PRODUCTION, PLANNING, AND EXPEDITING CLERKS', 13), ('SECRETARIES AND ADMINISTRATIVE ASSISTANTS, EXCEPT LEGAL, MEDICAL, AND EXECUTIVE', 13), ('COMMERCIAL PILOTS', 13), ('RISK MANAGEMENT SPECIALISTS', 12), ('COMPUTER NETWORK ARCHITECTS, NON R&D', 12), ('CAREER/TECHNICAL EDUCATION TEACHERS, SECONDARY SCHOOL', 12), ('ORTHODONTISTS', 12), ('DETECTIVES AND CRIMINAL INVESTIGATORS', 12), ('HUMAN RESOURCES ASSISTANTS, EXCEPT PAYROLL AND TIMEKEEPING', 12), ('EDUCATION ADMINISTRATORS, ELEMENTARY AND SECONDARY', 11), ('MARKET RESEARCH ANALYSTS & MARKETING SPECIALISTS', 11), ('SOFTWARE DEVELOPERS,APPLICATIONS', 11), ('DOCUMENT MANAGEMENT SPECIALISTS', 11), ('MATERIALS ENGINEER', 11), ('ELECTRICAL AND ELECTRONICS DRAFTERS', 11), ('CHEMIST', 11), ('AUDIO-VISUAL AND

MULTIMEDIA COLLECTIONS SPECIALISTS', 11), ('FINE ARTISTS, INCLUDING PAINTERS, SCULPTORS, AND ILLUSTRATORS', 11), ('COOKS, RESTAURANT', 11), ('BIOINFORMATICS TECHNICIANS', 11), ('AUTOMOTIVE SERVICE TECHNICIANS AND MECHANICS', 11), ('SOFTWARE DEVELOPERS, APPLICATION', 10), ('BUYERS AND PURCHASING AGENTS, FARM PRODUCTS', 10), ('SOFTWARE DEVELOPER, SYSTEMS SOFTWARE', 10), ('AEROSPACE ENGINEERING AND OPERATIONS TECHNICIANS', 10), ('RELIGIOUS WORKERS, ALL OTHER', 10), ('COMMERICIAL AND INDUSTRIAL DESIGNERS', 10), ('NURSE ANESTHETISTS', 10), ('FIRST-LINE SUPERVISORS OF PRODUCTION AND OPERATING WORKERS', 10), ('CLINICAL RESEARCH COORDINATORS', 9), ('COMPENSATION, BENEFITS, AND JOB ANALYSIS', 9), ('ENERGY ENGINEERS', 9), ('FORESTRY AND CONSERVATION SCIENCE TEACHERS,', 9), ('SOUND ENGINEERING TECHNICIANS', 9), ('HEALTHCARE PRACTITIONERS AND TECHNICAL WORKERS, AL', 9), ('AGENTS AND BUSINESS MANAGERS OF ARTISTS,', 8), ('FINANCIAL ANALYST', 8), ('TAX EXAMINERS AND COLLECTORS, AND REVENUE AGENTS', 8), ('SOFTWARE DEVELOPERS, SYSTEMS', 8), ('DATABASE ADMINISTRATOR', 8), ('ENGINEERS, ALL OTHERS', 8), ('ROBOTICS ENGINEERS', 8), ('ARBITRATORS, MEDIATORS, AND CONCILIATORS', 8), ('ANTHROPOLOGY AND ARCHEOLOGY TEACHERS,', 8), ('MIDDLE SCHOOL TEACHERS, EXCEPT SPECIAL AND CAREER/', 8), ('CRAFT ARTISTS', 8), ('EMERGENCY MEDICAL TECHNICIANS AND PARAMEDICS', 8), ('RETAIL SALESPERSONS', 8), ('SEPARATING, FILTERING, CLARIFYING, PRECIPITATING, AND STILL MACHINE SETTERS, OPERATORS, AND TENDERS', 8), ('PURCHASING AGENTS, EXCEPT WHOLESALE, RETAIL, AND F', 7), ('SUSTAINABILITY SPECIALISTS', 7), ('INFORMATION TECHNOLOGY PROJECT MANAGER', 7), ('CIVIL ENGINEER', 7), ('ELECTRICAL ENGINEER', 7), ('CONSERVATION SCIENTISTS', 7), ('FORESTERS', 7), ('SUBSTITUTE TEACHERS', 7), ('MEDICAL RECORDS AND HEALTH INFORMATION TECHNICIANS', 7), ('FIRST-LINE SUPERVISORS OF OFFICE AND ADMINISTRATIVE SUPPORT WORKERS', 7), ('PROOFREADERS AND COPY MARKERS', 7), ('CONSTRUCTION AND BUILDING INSPECTORS', 7), ('AIRCRAFT MECHANICS AND SERVICE TECHNICIANS', 7), ('QUALITY CONTROL SYSTEMS MANAGERS', 6), ('EMERGENCY MANAGEMENT DIRECTORS', 6), ('CLAIMS ADJUSTERS, EXAMINERS, AND INVESTIGATORS', 6), ('LOGISTICS ANALYSTS', 6), ('COMPUTER AND INFORMATION RESEARCH SCIENTIST', 6), ('COMPUTER & INFORMATION RESEARCH SCIENTISTS', 6), ('COMPUTER SUPPORT SPECIALIST', 6), ('BUSINESS INTELLIGENCE ANALYST', 6), ('HUMAN FACTORS ENGINEERS AND ERGONOMISTS', 6), ('VALIDATION ENGINEER', 6), ('PHOTONICS ENGINEERS', 6), ('INDUSTRIAL ENGINEERING TECHNICIANS', 6), ('MEDICAL SCIENTISTS, EXCEPT EPIDEMIOLOGIST', 6), ('INSTRUCTIONAL DESIGNERS AND TECHNOLOGISTS', 6), ('EDUCATION, TRAINING, AND LIBRARY WORKERS, ALL', 6), ('CHOREOGRAPHERS', 6), ('HEALTHCARE SUPPORT WORKERS, ALL OTHER', 6), ('FIRST-LINE SUPERVISORS OF NON-RETAIL SALES WORKERS', 6), ('SECURITIES, COMMODITIES, AND FINANCIAL SERVICES SA', 6), ('TRAVEL AGENTS', 6), ('REAL ESTATE BROKERS', 6), ('FINANCIAL CLERKS, ALL OTHER', 6), ('DESKTOP PUBLISHERS', 6), ('FIRST-LINE SUPERVISORS OF FARMING, FISHING, AND FORESTRY WORKERS', 6), ('DENTAL LABORATORY TECHNICIANS', 6), ('PRODUCTION WORKERS, ALL OTHER', 6), ('EDUCATION ADMINISTRATORS, PRESCHOOL AND CHILDCARE', 5), ('PROPERTY, REAL ESTATE, AND COMMUNITY ASSOCIATION', 5), ('SUPPLY CHAIN MANAGERS', 5), ('SOFTWARE DEVELOPERS, SYSTEM SOFTWARE', 5), ('CLINICAL DATA MANAGERS', 5), ('ELECTRONICS ENGINEERS, EXCEPT COMPUTER,', 5), ('ENVIROMENTAL ENGINEERS', 5), ('AUTOMOTIVE ENGINEERS', 5), ('ARCHITECTURAL DRAFTERS', 5), ('DRAFTERS, ALL OTHER', 5), ('ENGINEERING TECHNICIANS, EXCEPT DRAFTERS, ALL OTHE', 5), ('SURVEYING AND MAPPING TECHNICIANS', 5), ('MOLECULAR & CELLULAR BIOLOGISTS', 5), ('RECREATION AND FITNESS STUDIES TEACHERS, POSTSECON',

5), ('ENTERTAINERS AND PERFORMERS, SPORTS AND RELATED WORKERS, ALL OTHER', 5), ('RADIO AND TELEVISION ANNOUNCERS', 5), ('PERSONAL CARE AIDES', 5), ('SALES REPRESENTATIVES OF SERVICES, EXCEPT ADVERTISING, INSURANCE, TRAVEL, AND FINANCIAL SERVICES', 5), ('MODELS', 5), ('PROCUREMENT CLERKS', 5), ('RECEPTIONISTS AND INFORMATION CLERKS', 5), ('MEDICAL SECRETARIES', 5), ('OFFICE CLERKS, GENERAL', 5), ('CONSTRUCTION LABORERS', 5), ('BAKERS', 5), ('TRAFFIC TECHNICIANS', 5), ('MARKETING MANAGER', 4), ('REGULATORY AFFAIRS MANAGERS', 4), ('HUMAN RESOURCES SPECIALIST', 4), ('ACCOUNTANT', 4), ('FINANCIAL QUANTITATIVE ANALYST', 4), ('COMPUTER NETWORK ARCHITECTS, R&D', 4), ('GEOSPATIAL INFORMATION SCIENTISTS AND TECHNOLOGIST', 4), ('BIOMEDICAL ENGINEER', 4), ('ELECTRONICS ENGINEERS, EXCEPT COMPUTER, NON-R&D', 4), ('MICROBIOLOGIST', 4), ('ENVIRONMENTAL SCIENCE AND PROTECTION TECHNICIANS,', 4), ('FORENSIC SCIENCE TECHNICIANS', 4), ('CRIMINAL JUSTICE AND LAW ENFORCEMENT TEACHERS,', 4), ('ENGLISH LANGUAGE AND LITERATURE TEACHERS, POSTSECO', 4), ('GRADUATE TEACHING ASSISTANTS', 4), ('ELEMENTARY SCHOOL TEACHERS', 4), ('ADULT BASIC AND SECONDARY EDUCATION AND LITERACY', 4), ('BROADCAST NEWS ANALYSTS', 4), ('AUDIO AND VIDEO EQUIPMENT TECHNICIANS', 4), ('INTERNIST, GENERAL', 4), ('VETERINARY ASSISTANTS AND LABORATORY ANIMAL CARETAKERS', 4), ('FIRST-LINE SUPERVISORS OF LANDSCAPING, LAWN SERVICE, AND GROUNDSKEEPING WORKERS', 4), ('LANDSCAPING AND GROUNDSKEEPING WORKERS', 4), ('TRAVEL GUIDES', 4), ('CHILDCARE WORKERS', 4), ('SALES REPRESENTATIVES, WHOLESALE AND MANUFACTURING', 4), ('CARGO AND FREIGHT AGENTS', 4), ('COMPUTER OPERATORS', 4), ('OFFICE AND ADMINISTRATIVE SUPPORT WORKERS, ALL', 4), ('SUPERVISORS OF CONSTRUCTION AND EXTRACTION WORKERS', 4), ('CONSTRUCTION AND RELATED WORKERS, ALL OTHER', 4), ('AIRCRAFT STRUCTURE, SURFACES, RIGGING, AND SYSTEMS', 4), ('Blank', 4), ('SALES MANAGER', 3), ('TREASURERS AND CONTROLLERS', 3), ('FINANCIAL MANAGERS, BRANCH OR DEPARTMENT', 3), ('COMPLIANCE MANAGERS', 3), ('INVESTMENT FUND MANAGERS', 3), ('SECURITY MANAGERS', 3), ('MARKET RESEARCH ANALYSTS AND MARKETING', 3), ('MARKET RESEARCH ANALYST', 3), ('MARKET RESEARCH ANALYSTS AND MARKETING SPECIALIST', 3), ('AUDITORS', 3), ('SOFTWARE DEVELOPER APPLICATIONS', 3), ('SOFTWARE DEVELOPERS. APPLICATIONS', 3), ('SOFTWARE DEVELOPERS, SYSTEMS SOFTWARE', 3), ('COMPUTER OCCUPATIONS,ALL OTHER', 3), ('COMPUTER SYSTEMS ENGINEERS/ ARCHITECTS', 3), ('IT PROJECT MANAGERS', 3), ('SEARCH MARKETING STRATEGISTS', 3), ('BIOSTATISTICIAN', 3), ('ELECTRONICS ENGINEER, EXCEPT COMPUTER', 3), ('MINING AND GEOLOGICAL ENGINEERS', 3), ('ELECTRICAL ENGINEERING TECHNOLOGISTS', 3), ('ECONOMIST', 3), ('ENVIRONMENTAL ECONOMISTS', 3), ('COMMUNITY AND SOCIAL SERVICE SPECIALISTS, ALL', 3), ('ATMOSPHERIC, EARTH, MARINE, AND SPACE SCIENCES TEA', 3), ('SPECIAL EDUCATION TEACHERS, PRESCHOOL, KINDERGARTEN, AND ELEMENTARY SCHOOL', 3), ('COMMERCIAL AND INDUSTRIAL DESIGNER', 3), ('GRAPHIC DESIGNER', 3), ('MERCHANDISE DISPLAYERS AND WINDOW TRIMMERS', 3), ('ENTERTAINERS AND PERFORMERS, SPORTS AND RELATED', 3), ('PUBLIC RELATIONS SPECIALIST', 3), ('HOSPITALISTS', 3), ('RADIOLOGISTS', 3), ('NEUROLOGISTS', 3), ('VETERINARY TECHNOLOGISTS AND TECHNICIANS', 3), ('PHYSICAL THERAPIST ASSISTANTS', 3), ('DENTAL ASSISTANTS', 3), ('PRIVATE DETECTIVES AND INVESTIGATORS', 3), ('LIFEGUARDS, SKI PATROL, AND OTHER RECREATIONAL PROTECTIVE SERVICE', 3), ('COMBINED FOOD PREPARATION AND SERVING WORKERS, INCLUDING FAST FOOD', 3), ('FIRST-LINE SUPERVISORS OF PERSONAL SERVICE WORKERS', 3), ('ANIMAL TRAINERS', 3), ('SECURITIES AND COMMODITIES TRADERS', 3), ('BROKERAGE CLERKS', 3), ('LOAN

INTERVIEWERS AND CLERKS', 3), ('HUMAN RESOURCES ASSISTANTS, EXCEPT PAYROLL AND', 3), ('SHIPPING, RECEIVING, AND TRAFFIC CLERKS', 3), ('DATA ENTRY KEYERS', 3), ('AGRICULTURAL WORKERS, ALL OTHER', 3), ('CARPENTERS', 3), ('FIRST-LINE SUPERVISORS OF MECHANICS, INSTALLERS, AND REPAIRERS', 3), ('TELECOMMUNICATIONS EQUIPMENT INSTALLERS AND REPAIRERS, EXCEPT LINE INSTALLERS', 3), ('ELECTRICAL AND ELECTRONICS REPAIRERS, COMMERCIAL AND INDUSTRIAL EQUIPMENT', 3), ('BUS AND TRUCK MECHANICS AND DIESEL ENGINE SPECIALISTS', 3), ('MOBILE HEAVY EQUIPMENT MECHANICS, EXCEPT ENGINES', 3), ('MUSICAL INSTRUMENT REPAIRERS AND TUNERS', 3), ('MAINTENANCE AND REPAIR WORKERS, GENERAL', 3), ('FABRIC AND APPAREL PATTERNMAKERS', 3), ('JEWELERS AND PRECIOUS STONE AND METAL WORKERS', 3), ('HEAVY AND TRACTOR-TRAILER TRUCK DRIVERS', 3), ('GENERAL & OPERATIONS MANAGER', 2), ('COMPUTER AND INFORMATION SYTEMS MANAGER', 2), ('LOGISTICS MANAGERS', 2), ('CONSTRUCTION MANAGER', 2), ('GENERAL AND OPERATIONS MANAGER', 2), ('COST ESTIMATOR', 2), ('LOGISTICS ANALYST', 2), ('LOGISTICS ENGINEERS', 2), ('COMPENSATION, BENEFITS, & JOB ANALYSIS SPECIALISTS', 2), ('MARKETING RESEARCH ANALYSTS AND MARKETING SPECIALI', 2), ('MARKET RESEARCH ANALYST AND MARKETING SPECIALISTS', 2), ('SOFTWARE DEVELOPERS, APPLICATIONS, NON R & D', 2), ('MPUTER AND INFORMATION RESEARCH SCIENTISTS', 2), ('INFORMATION SECURITY ANALYST', 2), ('SOFTWARE DEVELOPER - APPLICATIONS', 2), ('SOTWARE DEVELOPERS, APPLICATIONS', 2), ('SOFTWARE DEVELOPERS, APPLICATINS', 2), ('SOFTWARE DEVELOPERS - APPLICATIONS', 2), ('FTWARE DEVELOPERS, APPLICATIONS', 2), ('SOFTWARE DEVELOPERS,SYSTEMS SOFTWARE', 2), ('SOFTWARE DEVELOPERS, SYSTEMS SOTWARE', 2), ('COMPUTER NETWORKS ARCHITECTS', 2), ('COMPUTER NETWORK SUPPORT SPECIALIST', 2), ('SOFTWARE QUALITY ASSURANCE ENGINEERS', 2), ('COMPUTER SYSTEM ENG / ARCHITECT', 2), ('COMPUTER OCCPATIONS, ALL OTHER', 2), ('VLSI DESIGN VERIFICATION ENGINEER', 2), ('WATER / WASTEWATER ENGINEERS', 2), ('MANUFACTURING ENGINEER', 2), ('MICROSYSTEMS ENGINEERS', 2), ('NANOSYSTEMS ENGINEERS', 2), ('GINEERS, ALL OTHER', 2), ('ARCHITECTURAL & CIVIL DRAFTERS', 2), ('ELECTROMECHANICAL ENGINEERING TECHNOLOGISTS', 2), ('ENGINEERING TECH, EXCEPT DRAFTERS, ALL OTHER', 2), ('BIOCHEMISTS & BIOPHYSICISTS', 2), ('BIOLOGICAL SCIENTIST, ALL OTHER', 2), ('BIOINFORMATICS SCIENTIST', 2), ('BIOLOGICAL SCIENTISTS', 2), ('ENVIRONMENTAL SCIENTISTS AND SPECIALISTS, INCLUDIN', 2), ('REMOTE SENSING SCIENTISTS AND TECHNOLOGISTS', 2), ('FOREST AND CONSERVATION TECHNICIANS', 2), ('EDUCATIONAL, GUIDANCE, SCHOOL, AND VOCATIONAL CO', 2), ('EDUCATIONAL, GUIDANCE, SCHOOL, AND VOCATIONAL COUN', 2), ('AREA, ETHNIC, AND CULTURAL STUDIES TEACHERS, POSTS', 2), ('SOCIAL SCIENCE TEACHERS, POSTSECONDARY, ALL OTHER', 2), ('HEALTH SPECIALTIES TEACHER, POSTSECONDARY', 2), ('ENGLISH LANGUAGE & LITERATURE TEACHERS, POSTSECOND', 2), ('FOREIGN LANGUAGE & LITERATURE TEACHERS, POSTSECOND', 2), ('ELEMENTARY SCHOOL TEACHERS, EXCEPT SPECIAL EDUCAT', 2), ('MIDDLE SCHOOL TEACHERS EXCEPT SPECIAL AND CAREER/T', 2), ('CAREER/TECHNICAL EDUCATION TEACHERS, MIDDLE', 2), ('SECONDARY SCHOOL TEACHERS, EXCEPT SPECIAL', 2), ('SECONDARY SCHOOL TEACHER', 2), ('SPECIAL EDUCATION TEACHERS, PRESCHOOL,', 2), ('SPECIAL EDUCATION TEACHERS, KINDERGARTEN AND ELEME', 2), ('ADULT BASIC AND SECONDARY EDUCATION AND LITERACY T', 2), ('TEACHER ASSISTANTS', 2), ('FLORAL DESIGNERS', 2), ('DANCERS', 2), ('CAMERA OPERATORS, TELEVISION, VIDEO, AND MOTION PICTURE', 2), ('DENTIST', 2), ('PHYSICIANS AND SURGEONS', 2), ('PHYSICAL THERAPIST', 2), ('RESPIRATORY THERAPISTS', 2), ('MEDICAL AND LABORATORY TECHNOLOGISTS', 2), ('PSYCHIATRIC TECHNICIANS', 2), ('HEALTHCARE PRACTITIONERS

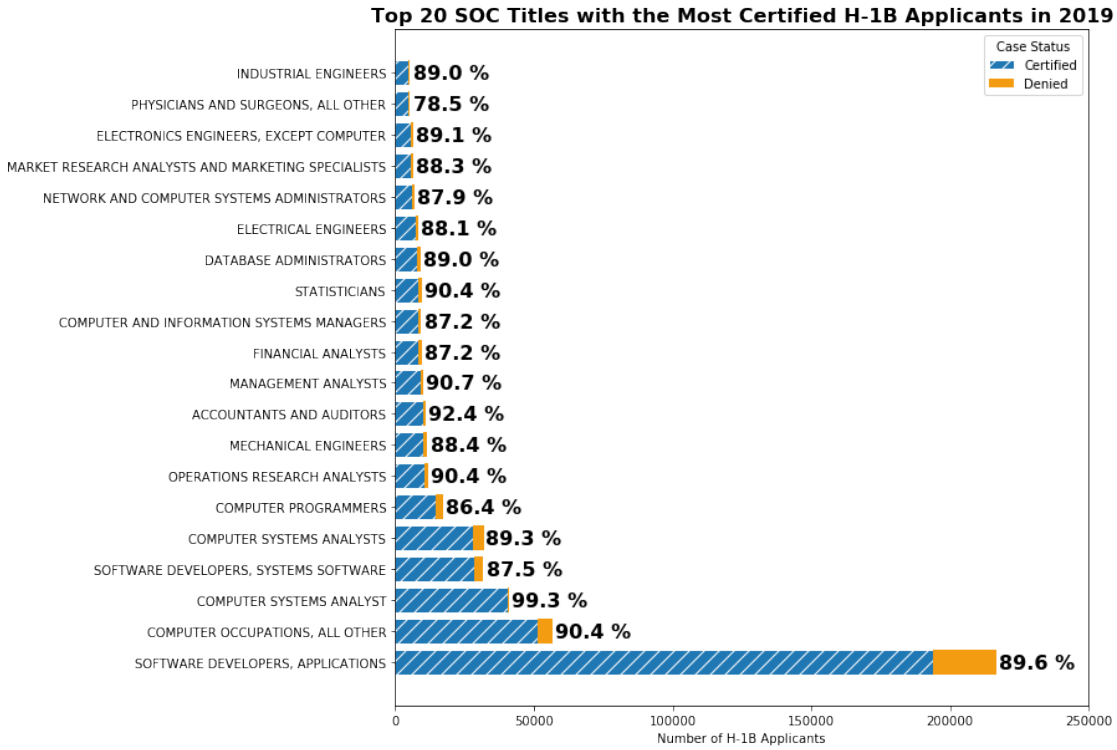
AND TECHNICAL WORKERS', 2), ('HEALTHCARE PRACTITIONERS AND SPECIALTY WORKERS, AL', 2), ('HOME HEALTH AIDES', 2), ('MEDICAL ASSISTANTS', 2), ('VETERINARY ASSISTANTS AND LABORATORY ANIMAL', 2), ('FIRST-LINE SUPERVISORS OF FOOD PREPARATION AND SERVING WORKERS', 2), ('WAITERS AND WAITRESSES', 2), ('HOSTS AND HOSTESSES, RESTAURANT, LOUNGE, AND COFFEE SHOP', 2), ('MAIDS AND HOUSEKEEPING CLEANERS', 2), ('HAIRDRESSERS, HAIRSTYLISTS, AND COSMETOLOGISTS', 2), ('MANICURISTS AND PEDICURISTS', 2), ('TOUR GUIDES AND ESCORTS', 2), ('FIRST-LINE SUPERVISORS OF RETAIL SALES WORKERS', 2), ('DEMONSTRATORS AND PRODUCT PROMOTERS', 2), ('LEGAL SECRETARIES', 2), ('SECRETARIES AND ADMINISTRATIVE ASSISTANTS, EXCEPT', 2), ('FARMWORKERS AND LABORERS, CROP, NURSERY, AND GREENHOUSE', 2), ('FOREST AND CONSERVATION WORKERS', 2), ('BOILERMAKERS', 2), ('FLOOR LAYERS, EXCEPT CARPET, WOOD, AND HARD TILES', 2), ('CEMENT MASONS AND CONCRETE FINISHERS', 2), ('OPERATING ENGINEERS AND OTHER CONSTRUCTION EQUIPMENT OPERATORS', 2), ('ELECTRICIANS', 2), ('HELPERS, CONSTRUCTION TRADES, ALL OTHER', 2), ('ELECTRICAL AND ELECTRONICS INSTALLERS AND REPAIRERS, TRANSPORTATION EQUIPMENT', 2), ('MEDICAL EQUIPMENT REPAIRERS', 2), ('AIRCRAFT STRUCTURE, SURFACES, RIGGING, AND SYSTEMS ASSEMBLERS', 2), ('COMPUTER NUMERICALLY CONTROLLED MACHINE TOOL PROGRAMMERS, METAL AND PLASTIC', 2), ('FOUNDRY MOLD AND COREMAKERS', 2), ('WELDERS, CUTTERS, SOLDERERS, AND BRAZERS', 2), ('INSPECTORS, TESTERS, SORTERS, SAMPLERS, AND WEIGHERS', 2), ('MOLDERS, SHAPERS, AND CASTERS, EXCEPT METAL AND PLASTIC', 2), ('AIRFIELD OPERATIONS SPECIALISTS', 2), ('CRANE AND TOWER OPERATORS', 2), ('ADMINISTRATIVE SERVICES MANAGER', 1), ('COMPUTER & INFORMATION SYSTEMS MANAGERS', 1), ('COMPUTER AND INFORMATION MANAGER SYSTEM MANAGER', 1), ('COMPUTER INFORMATION SYSTEMS MANAGERS', 1), ('COMPUTER AND INFORMATION SYSTEMS', 1), ('FINANCE MANAGERS', 1), ('COMPENSATION AND BENEFITS MANAGER', 1), ('CONSTRUCTIONS MANAGERS', 1), ('CHIEF EXECUTIVE OFFICER', 1), ('ENGINEERING MANAGERS', 1), ('NATURAL SCIENCES MANAGER', 1), ('PROPERTY, REAL ESTATE, AND COMMUNITY ASSOCIATION M', 1), ('PROPERTY, REAL ESTATE MANAGERS', 1), ('PRODUCTION COORDINATOR', 1), ('AGENTS AND BUSINESS MANAGERS OF ARTISTS, PERFORMER', 1), ('PURCHASING AGENTS, EXCEPT WHOLESALE, RETAIL & FARM', 1), ('PURCHASING AGNT,EXCEPT WHOLESALE,RETAIL&FARM PRODT', 1), ('REGULATORY AFFAIRS SPECALISTS', 1), ('REGULATORY AFFAIRS SPECIALIST', 1), ('HUMAN RESOURCE SPECIALIST', 1), ('HUMAN RESOURCE SPECIALISTS', 1), ('LOGISTICIAN', 1), ('MANAGEMENT ANALYTST', 1), ('MANAGEMANT ANALYSTS', 1), ('MANAGEMENT ANALYSIS', 1), ('COMPENSATION BENEFITS MANAGER', 1), ('TRAINING AND DEVELOPMENT SPECIALIST', 1), ('MARKET RESEARCH ANALYST AND MARKETING SPECIALIST', 1), ('MARKET RESEARCH ANALYSTS', 1), ('MARKET RESEARCH ANALYSTS AND MARKETING SCPECIALIST', 1), ('ACCOUNTANTS AND AUDOTORS', 1), ('ACCOUNTS AND AUDITORS', 1), ('PERSONAL FINANCIAL ADVISOR', 1), ('RISK MANAGEMENT SPECIALIST', 1), ('COMPUTER PROGRAMMERS R & D', 1), ('SOFTWARE DEVELOPERS, APPLICATIONS, NONR&D', 1), ('SOFTWARE DEVELOPERS, APPLICATIONS, R&D', 1), ('SOFTWARE DEVELOPERS, SYSTEMS SOFTWARE, R&D', 1), ('COMPUTER SYSTEMS ANALYSTS, NON R&D', 1), ('COMPUTER SYSTEMS ANALYSTS, NON R&D COMPUTER SYSTEM', 1), ('RESEARCH ENGINEER', 1), ('COMPUTER AND INFORMATION RESEARCH SCIENCES', 1), ('15-1121', 1), ('COMPTUER PROGRAMMERS', 1), ('SENIOR SOFTWARE DEVELOPER', 1), ('SOFTWARE DEVELOPERS, APLICATIONS', 1), ('SOFTWARE DEVELOPRS, APPLICATIONS', 1), ('SOFTWARE DEVELOPERS, APPLICAITONS', 1), ('SOFTWARE SEVELOPERS, APPLICATIONS', 1), ('SOFTWARE DEVLOPERS, APPLICATIONS', 1), ('SOFTWARE DEVELOPERS,

APPPLICATIONS', 1), ('SOFTWARE DEVELOPERS, APPLICANTIONS', 1), ('SOFTWARE DEVVELOPERS, APPLICATIONS', 1), ('SOFTWARE DEVLOPER, APPLICATIONS', 1), ('SOFTWARE DEVELPER, APPLICATIONS', 1), ('TESTING ENGINEERING SPECIALIST', 1), ('SOFTWARE DEVELOPERS, APPLICTAIONS', 1), ('SOFTWARE DEVELOPER', 1), ('SOFTWARE DEVELOPER APPLICATION', 1), ('SOFTARE DEVELOPERS, APPLICATIONS', 1), ('SOFTWARE APPLICATIONS, DEVELOPERS', 1), ('SOFWARE DEVELOPER, APPLICATIONS', 1), ('SOFTWARE DEVELOPERS, APPLICTIONS', 1), ('SOFWRE DEVELOPERS, APPLICATIONS', 1), ('SOFWARE DEVELOPERS, APPLICATIONS', 1), ('SOFTWARE DEVELOPER, SYSTEMS', 1), ('SENIOR VALIDATION ENGINEER', 1), ('SOFTWARE DEVELOPERS , SYSTEMS SOFTWARE', 1), ('SOFTWARE QUALITY ASSURANCE', 1), ('SOFTWARE DEVLEOPERS, SYSTEMS SOFTWARE', 1), ('DATABASE ARCHITECT', 1), ('DATABASE ADMINITRATORS', 1), ('NETWORK AND COMPUTER SYSTEMS ADMNISTRATORS', 1), ('NETWORK AND COMPUTER SYTEMS ADMINISTRATORS', 1), ('INFORMATION SECURITY ANALYSTS, WEB DEVELOPERS, AND', 1), ('SYSTEMS ARCHITECTS', 1), ('WEB ADMINISTRATORS', 1), ('INFORMATION TECHNOLOGY MANAGERS', 1), ('SOFTWARE QUALITY ENGINEERS AND TESTERS', 1), ('COMPUTER OCCUPATIONS, ALL OTHER', 1), ('SOFTWARE QA ENG + TEST', 1), ('COMPUTER OCCUPATIONS,ALL OTHERS', 1), ('INFORMATION TECHNOLOGY PRODUCT MANAGERS', 1), ('QUALITY ASSURANCE ENGINEERS AND TESTERS', 1), ('COMPUTER OCCUPATIONS - ALL OTHER', 1), ('INFORMATION TECHOLOGY PROJECT MANAGERS', 1), ('VIDEO GAME DESIGNERS', 1), ('SEARCH MARKETING STRATEGIST', 1), ('CUMPUTER OCCUPATIONS, ALL OTHER', 1), ('SENIOR SOFTWARE ENGINEER', 1), ('COMPUTER OCCUPATONS, ALL OTHER', 1), ('COMPUTER OCUPATIONS, ALL OTHER', 1), ('COMPUTER OCCUPATIONS. ALL OTHER', 1), ('MPUTER OCCUPATIONS, ALL OTHER', 1), ('COMPUTER OCCUPATION, ALL OTHER', 1), ('COMPUTER OCCCUPATIONS, ALL OTHER', 1), ('COMPUTER SYSTEMS ENGINEERS/ARCHITECT', 1), ('GEOSPATIAL INFO. SCIENTISTS AND TECHNOLOGISTS', 1), ('COMPUTERS SYSTEMS ENGINEERS/ARCHITECTS', 1), ('MATHEMATICIAN', 1), ('OPERATION RESEARCH ANALYSTS', 1), ('STATISTIANS', 1), ('STASTICIANS', 1), ('STATISICIANS', 1), ('BIOSTATISTIANS', 1), ('AEROSPACE ENGINEER', 1), ('TRANSPORTATION ENGINEERS', 1), ('COMPUTER HARDWARE ENGINEER', 1), ('COMPUTER HARDWARE ENGINEERS, NON-R&D', 1), ('COMPUTER HARDWARE ENGINEERS, R & D', 1), ('ELECTRICAL ENGINEERS', 1), ('ELECTRICAL ENGINEERING', 1), ('ELECTRONIC ENGINEERS, EXCEPT COMPUTER', 1), ('ELECTONICS ENGINEERS, EXCEPT COMPUTER', 1), ('17-2072', 1), ('ELECTRICAL ENGINEERS, NON R&D', 1), ('WATER/WASTEWATER ENGINEERS', 1), ('WATER / WASTEWATER ENGINEER', 1), ('HEALTH AND SAFETY ENGS., EXCEPT MINING SAFETY ENGS', 1), ('INDUSTRIAL ENGINEER', 1), ('MECHANCIAL ENGINEER', 1), ('MECHANICAL ENGINEEERS', 1), ('QUALITY ENGINEER', 1), ('SENIOR MECHANICAL ENGINEER', 1), ('MINING AND GEOLOGICAL ENG INCL MINING SAFETY ENG', 1), ('ROBOTICS ENGINEER', 1), ('BIOCHEMICAL ENGINEERS', 1), ('ELECTRO-MECHANICAL TECHNICIANS', 1), ('ENGINEERING TECHNICIANS, EXCEPT DRAFTERS', 1), ('SOIL AND PLANT SCIENTIST', 1), ('SOLUTION CONSULTANT', 1), ('BIOLOGICAL SCIENCES, ALL OTHER', 1), ('BIOININFORMATICS SCIENTISTS', 1), ('BIOLOGICAL SCIENTISTS, ALL OTHERS', 1), ('GENETICISTS', 1), ('PHYSICIST', 1), ('MTERIALS SCIENTISTS', 1), ('MATERIALS SCIENTIST', 1), ('MATERIAL SCIENTISTS', 1), ('ENVIRONMENTAL AND SCIENTISTS SPECIALISTS', 1), ('ENVIRONMENTAL SCIENTISTS AND SPECIALISTS', 1), ('GEOSCIENTISTS, EXCEPT HYDROLOGISTS & GEOGRAPHERS', 1), ('PYSCHOLOGISTS, ALL OTHER', 1), ('NEUROPSYCHOLOGISTS AND CLINICAL NEUROPSYCHOLOGISTS', 1), ('TRANSPORTATION PLANNERS', 1), ('SOCIAL SCIENTISTS AND RELATED WORKERS, ALL OTHERS', 1), ('NUCLEAR TECHNICIANS', 1), ('ENVIRONMENTAL SCIENCE AND PROTECTION

TECHNICIAN', 1), ('QUALITY CONTROL ANALYSTS', 1), ('PROBATION OFFICERS AND CORRECTIONAL TREATMENT SPECIALISTS', 1), ('LEGAL SUPPORT WORKERS', 1), ('BUSINESS TEACHER, POSTSECONDARY', 1), ('BUSINESS TEACHERS POSTSECONDARY', 1), ('BUSINESS TEACHERS, POST SECONDARY', 1), ('BUSINESS, TEACHERS, POSTSECONDARY', 1), ('BUSINESS TEACHERS', 1), ('ASSISTANT PROFESSOR', 1), ('COMPUTER SCIENCE TEACHER POSTSECONDARY', 1), ('MATHEMATICAL SCIENCE TEACHERS, SECONDARY', 1), ('AGRICULTURAL SCIENCE TEACHERS, POSTSECONDARY', 1), ('ATMOSPHERIC, EARTH, MARINE, & SPACE SCIENCES TEACH', 1), ('ATMOSPHERIC, EARTH, MARINE & SPACE SCIENCES TEACHE', 1), ('CHEMISTRY TEACHERS, POST SECONDARY', 1), ('ANTHROPOLOGY AND ARCHEOLOGY TEACHERS, POSTSECONDAR', 1), ('AREA, ETHNIC, AND CULTURAL STUDIES TEACHERS', 1), ('ECONOMIC TEACHERS, POSTSECONDARY', 1), ('SOCIAL SCIENCES TEACHERS, POSTSECONDARY', 1), ('ART, DRAMA, MUSIC TEACHERS, POSTSECONDARY', 1), ('FOREIGN LANGUAGE & LIT TEACHERS, POSTSECONDARY', 1), ('FOREIGN LANGUAGE & LIT. TEACHERS, POSTSECONDARY', 1), ('HISTORY TEACHER, POSTSECONDARY', 1), ('RECREATION AND FITNESS STUDIES TEACHER, POSTSECOND', 1), ('ELEMENTARY SCHOOL TEACHER', 1), ('MIDDLE SCHOOL TEACHERS, EXCEPT SPECIAL AND CAREER', 1), ('CAREER/TECHNICAL EDUCATION TEACHERS, MIDDLE SCHOOL', 1), ('SECONDARY SCHOOL TEACHERS, EXCEPT SPECIAL CAREER/T', 1), ('CAREER/TECHNICAL EDUCATION TEACHERS, SECONDARY', 1), ('MULTIMEDIA ARTISTS AND ANIMAORS', 1), ('ACTORS', 1), ('PROGRAM DIRECTORS', 1), ('ATHLETES AND SPORTS COMPETITORS', 1), ('COPY WRITERS', 1), ('POETS, LYRICISTS AND CREATIVE WRITERS', 1), ('BROADCAST TECHNICIANS', 1), ('DENTIST, GENERAL', 1), ('DIETITIANS AND NUTRITIONIST', 1), ('INTERNISTS GENERAL', 1), ('PEDIATRICIAN', 1), ('PHYSICIAN AND SURGEONS, ALL OTHER', 1), ('PATHOLOGISTS', 1), ('PHYSICIANS AND SURGEONS; ALL OTHER', 1), ('RADIATION THERAPISTS', 1), ('SPEECH LANGUAGE PATHOLOGISTS', 1), ('DIAGNOSTIC MEDICAL SONOGRAPHERS', 1), ('NUCLEAR MEDICINE TECHNOLOGISTS', 1), ('OPHTHALMIC MEDICAL TECHNICIANS', 1), ('OPTICIANS, DISPENSING', 1), ('ATHLETIC TRAINER', 1), ('MEDICAL TRANSCRIPTIONISTS', 1), ('VETERINARY ASSISTANTS AND LABORATORY ANIMAL CARETA', 1), ('CHEF AND HEAD COOKS', 1), ('FIRST-LINE SUPERVISORS OF FOOD PREPARATION AND SER', 1), ('FIRST-LINE SUPERVISORS OF LANDSCAPING, LAWN', 1), ('TREE TRIMMERS AND PRUNERS', 1), ('NONFARM ANIMAL CARETAKERS', 1), ('COSTUME ATTENDANTS', 1), ('SECURITIES,COMMODITIES&FINANCIAL SRVS SALES AGENTS', 1), ('SECURITIES COMMODITIES AND FINANCIAL SERVICES SALE', 1), ('SECURITIES, COMMODITIES, FINANCIAL SERVICES AGENTS', 1), ('SALES AGENTS, SECURITIES AND COMMODITIES', 1), ('SECURITIES, COMMODITIES, AND FINANCIAL', 1), ('SALES ENGINEEER', 1), ('BILL AND ACCOUNT COLLECTORS', 1), ('BILLING AND POSTING CLERKS', 1), ('PAYROLL AND TIMEKEEPING CLERKS', 1), ('FILE CLERKS', 1), ('HOTEL, MOTEL, AND RESORT DESK CLERKS', 1), ('INTERVIEWERS, EXCEPT ELIGIBILITY AND LOAN', 1), ('RESERVATION AND TRANSPORTATION TICKET AGENTS AND', 1), ('RESERVATION AND TRANSPORTATION TICKET AGENTS AND TRAVEL CLERKS', 1), ('EXECUTIVE SECRETARIES AND EXECUTIVE ADMINISTRATIVE', 1), ('MAIL CLERKS AND MAIL MACHINE OPERATORS, EXCEPT POSTAL SERVICE', 1), ('STATISTICAL ASSISTANT', 1), ('FIRST-LINE SUPERVISORS OF FARMING, FISHING, AND', 1), ('ANIMAL BREEDERS', 1), ('FARMWORKERS, FARM, RANCH, AND AQUACULTURAL ANIMALS', 1), ('TILE AND MARBLE SETTERS', 1), ('PLUMBERS, PIPEFITTERS, AND STEAMFITTERS', 1), ('HELPERS--ELECTRICIANS', 1), ('AVIONICS TECHNICIANS', 1), ('SECURITY AND FIRE ALARM SYSTEMS INSTALLERS', 1), ('HEATING, AIR CONDITIONING, AND REFRIGERATION MECHANICS AND INSTALLERS', 1), ('PRECISION INSTRUMENT AND EQUIPMENT REPAIRERS,

```
ALL', 1), ('HELPERS--INSTALLATION, MAINTENANCE, AND REPAIR', 1), ('HELPERS--
INSTALLATION, MAINTENANCE, AND REPAIR WORKERS', 1), ('INSTALLATION, MAINTENANCE,
AND REPAIR WORKERS, ALL OTHER', 1), ('STRUCTURAL METAL FABRICATORS AND FITTERS',
1), ('FOOD BATCHMAKERS', 1), ('FOOD COOKING MACHINE OPERATORS AND TENDERS', 1),
('FOOD PROCESSING WORKERS, ALL OTHER', 1), ('PRINTING PRESS OPERATORS', 1),
('SAWING MACHINE SETTERS, OPERATORS, AND TENDERS, WOOD', 1), ('WOODWORKING
MACHINE SETTERS, OPERATORS, AND TENDERS, EXCEPT SAWING', 1), ('CHEMICAL PLANT
AND SYSTEM OPERATORS', 1), ('SHIP ENGINEERS', 1), ('15-2031', 1)]


[22491, 194060, 216551, 89.61399393214532]
```

[21]:
```
# Plot horizontal bar
plt.figure(figsize=(10,10))
certified_data = df_soc_title_count['Number of Certified H-1B Visas'].
 ↪sort_values(ascending=False).head(20)
denied_data = df_soc_title_count.loc[certified_data.index[:]]['Number of Denied␣
 ↪H-1B Visas']
b1 = plt.barh(certified_data.index, certified_data, color='#1f77b4', hatch ="//
 ↪", edgecolor="white")
b2 = plt.barh(certified_data.index, denied_data, left=certified_data,␣
 ↪color='#F39C12')
plt.ylabel("")
plt.xlim([0, 250000])
plt.xlabel("Number of H-1B Applicants")
plt.title('Top 20 SOC Titles with the Most Certified H-1B Applicants in 2019',␣
 ↪fontweight='bold', fontsize = 16)
plt.legend([b1, b2], ["Certified", "Denied"], title="Case Status", loc="upper␣
 ↪right")

# Annotate the horizontal bar with Certified H-1B Visas Percentage [%]
for i, num_cert in enumerate(sorted_df[:20]):

    plt.text(num_cert + 1000, i- 0.2, f'{sc_cert_perc[i]:0.1f} %',␣
 ↪color='black', fontweight='bold', fontsize = 16)

plt.show()
```



[107]:
```
focus = [2, 6, 7, 9, 10, 20, 52, 60, 258]

for i, ele in enumerate(focus):
    print (str(i + 1) + " " + data_columns[ele - 1])

print ("\nThere are {} focused catergories, in total.".format(len(focus)))
```

```
1 CASE_STATUS
2 VISA_CLASS
3 JOB_TITLE
4 SOC_TITLE
5 FULL_TIME_POSITION
6 EMPLOYER_NAME
7 WORKSITE_STATE_1
8 PW_WAGE_LEVEL_1
9 STATUTORY_BASIS

There are 9 focused catergories, in total.
```

[108]:
```
visa_list = list(data["VISA_CLASS"])
mask = []

for i in range(len(visa_list)):
```

```
    visa = visa_list[i]
    if visa == "H-1B": mask.append(i)

print ("We will focus preliminary on the H-1B VISA class.")
print ("There are {} H-1B VISA cases out of {} total cases.".format(len(mask),␣
 ↪len(visa_list)))
```

```
We will focus preliminary on the H-1B VISA class.
There are 649083 H-1B VISA cases out of 664616 total cases.
```

[109]:
```
print (set(data["CASE_STATUS"]))
```

```
{'CERTIFIED', 'CERTIFIED-WITHDRAWN', 'WITHDRAWN', 'DENIED'}
```

[110]:
```
# CASE_STATUS

case_status = np.array(list(deepcopy(data["CASE_STATUS"])))[mask]
case = np.zeros((len(case_status)))
case_dict = {"DENIED" : 0, "CERTIFIED" : 1, "CERTIFIED-WITHDRAWN" : 0,␣
 ↪"WITHDRAWN" : 0}

for i in range(len(case_status)):
    this_case = case_status[i]
    case[i] = int(case_dict[this_case])

case = np.array(case, dtype = int)

print ("The rate of acceptance is %s." % str(100 - np.count_nonzero(case == 0) /
 ↪ len(case) * 100)[: 4])
```

```
The rate of acceptance is 89.1.
```

[111]:
```
# SOC_CODE

soc_code = np.array(list(deepcopy(data["SOC_CODE"])))[mask] # every element␣
 ↪will be numpy.str_

code_list = list(set(soc_code))
code_dict = {code_list[i] : i for i in range(len(code_list))}

code_vector = [code_dict[code] for code in soc_code]
```

[112]:
```
# FULL_TIME_POSITION

full_time = np.array(list(deepcopy(data["FULL_TIME_POSITION"])))[mask]
full_time_dict = {"N" : 0, "nan" : 0, "Y" : 1}
```

```
full_time_vector = [full_time_dict[time] for time in full_time]

print (set(list(full_time)))
```

```
{'N', 'Y'}
```

[113]:
```
# PW_WAGE_LEVEL

pw_wage = np.array(deepcopy(data["PW_WAGE_LEVEL_1"]))[mask]
print (set(pw_wage))

pw_wage_dict = {"Level I" : 1, "Level II" : 2, "Level III" : 3, "Level IV" : 4,␣
 ↪"Other" : 0}

pw = []

for ele in pw_wage:
    if type(ele) == str: pw.append(pw_wage_dict[ele])
    else: pw.append(pw_wage_dict["Other"])
```

```
{nan, 'Level III', 'Level IV', 'Level I', 'Level II'}
```

[115]:
```
res = case
factories = [code_vector, full_time_vector, pw]

fac = np.array(factories.pop(0)).reshape(-1, 1)
while factories: fac = np.concatenate((fac, np.array(factories.pop(0)).
 ↪reshape(-1, 1)), axis = 1)

print (res.shape, fac.shape)

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(fac, res, test_size = 0.2,␣
 ↪random_state = 42)

print (y_train.shape, y_test.shape)

# from sklearn.linear_model import LogisticRegression
# from sklearn.metrics import f1_score

# model = LogisticRegression(random_state = 0, class_weight = "balanced")
# model.fit(X_train, y_train)
# y_pred = model.predict(X_test)

# f1score = f1_score(y_test, y_pred)
# print (f1score)
```

```
(649083,) (649083, 3)
(519266,) (129817,)
```

[116]:
```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors = 5)
neigh.fit(X_train, y_train)

# print (neigh.score(X_test, y_test))
```

[116]:
```
KNeighborsClassifier()
```

[117]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,accuracy_score
classifier = GaussianNB()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
ac = accuracy_score(y_test,y_pred)
print (ac)
```

```
0.8790682268115886
```

[65]:
```
print (np.count_nonzero(y_pred == 1))
```

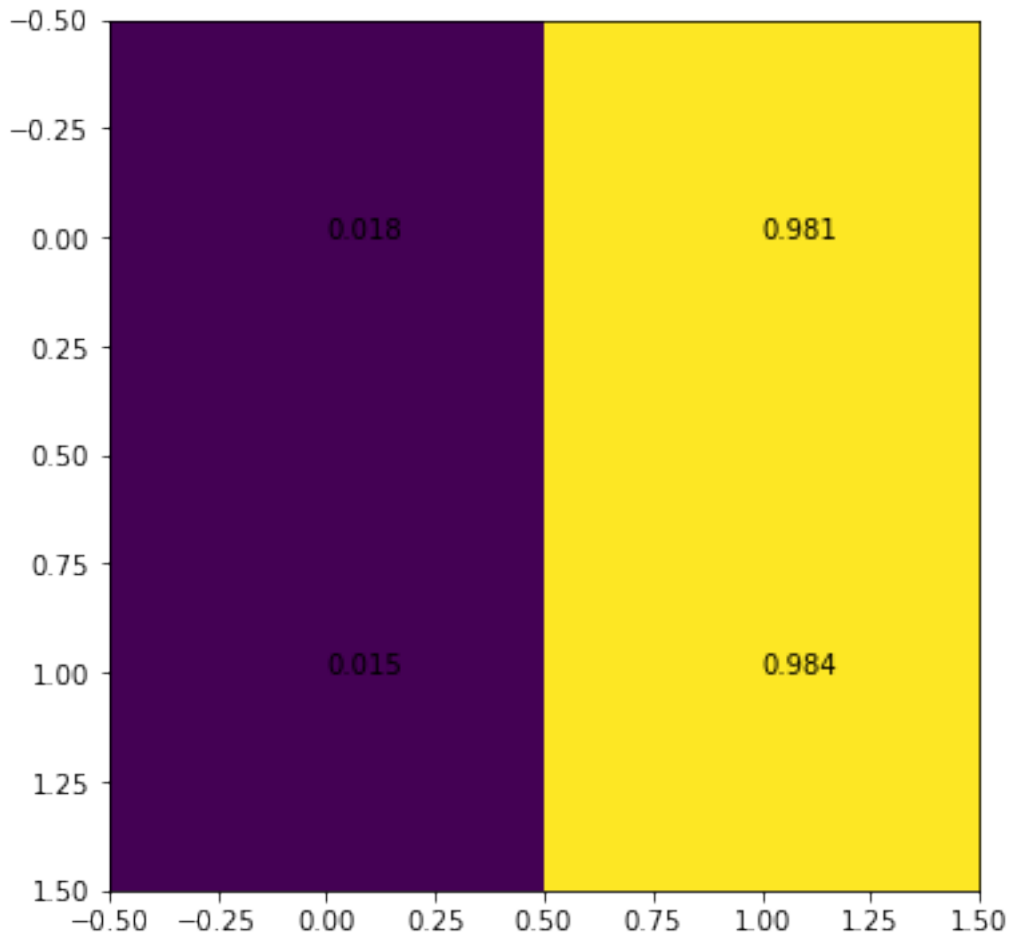
```
127636
```

[185]:
```
from tqdm import tqdm

def DataBatch(data, label, batchsize, shuffle=True):
    """
    This function provides a generator for batches of data that
    yields data (batchsize, 3, 32, 32) and labels (batchsize)
    if shuffle, it will load batches in a random order
    """
    n = data.shape[0]
    if shuffle:
        index = np.random.permutation(n)
    else:
        index = np.arange(n)
    for i in range(int(np.ceil(n/batchsize))):
        inds = index[i*batchsize : min(n,(i+1)*batchsize)]
        yield data[inds], label[inds]

def Confusion(testData, testLabels, classifier, classes):
    batchsize=50
    correct=0
    M=np.zeros((classes, classes))
    num=testData.shape[0]/batchsize
    count=0
```

```
    acc=0

    for data,label in␣
 ↪tqdm(DataBatch(testData,testLabels,batchsize,shuffle=False),total=len(testData)/
 ↪/batchsize):
        """ ==========
        YOUR CODE HERE
        ========== """
        prediction = classifier.predict(data)
        for i in range(len(label)):
            M[label[i]][prediction[i]] += 1

    for i in range(len(M)):
        acc += M[i, i]
        M[i] = M[i] / np.sum(M[i])

#     print (acc*100.0/len(testData), len(testData))

    return M,acc*100.0/len(testData)

def VisualizeConfussion(M):
    plt.figure(figsize=(14, 6))
    plt.imshow(M, cmap = "winter")

    h, w = np.shape(M)

    for i in range(h):
        for j in range(w):
            plt.annotate(str(M[j, i])[: 5], xy = (i, j))
#     plt.imshow(M)
#     plt.annotate(, xy = (0.+, 0.5))
    plt.show()
    print(np.round(M,2))
```

[182]:
```
M,acc = Confusion(X_test, y_test, classifier, 2)
VisualizeConfussion(M)
print ("The accuracy for Naive Bayes Classification is %s%%. " % str(acc)[: 5])
```

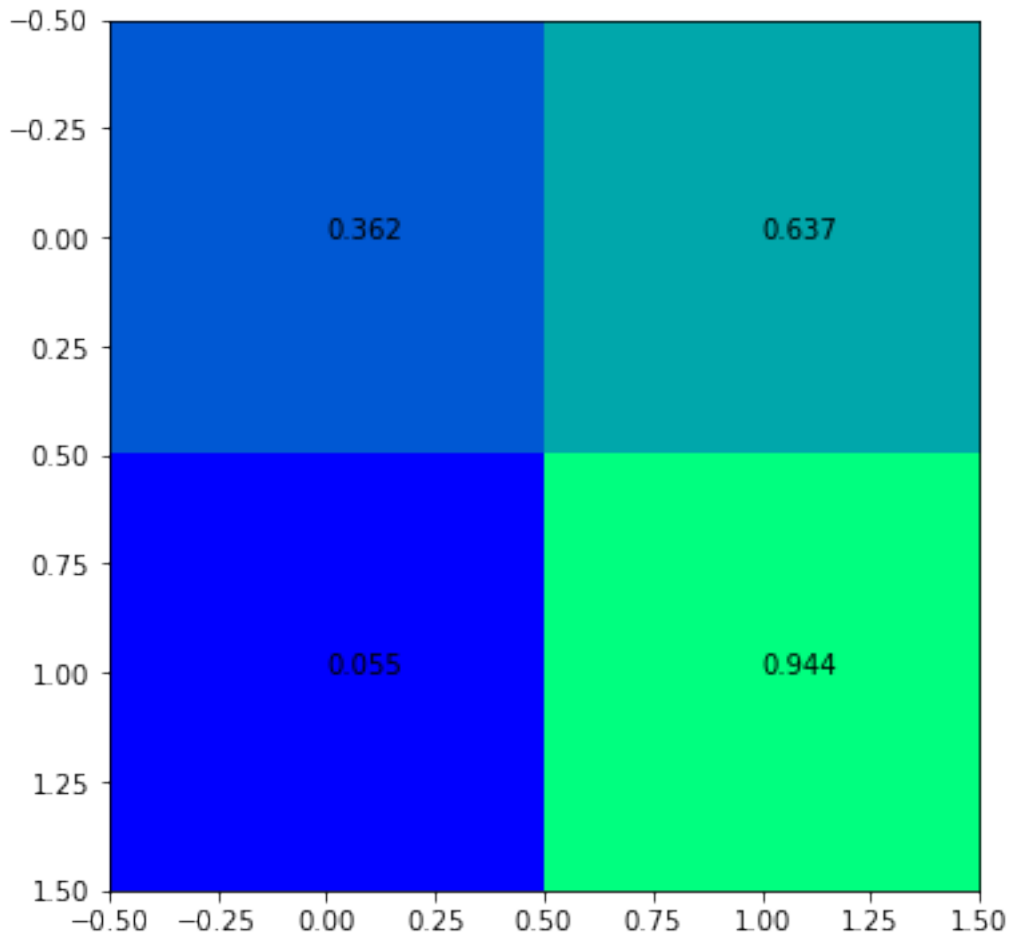
```
2597it [00:00, 7889.65it/s]
```




```
[[0.02 0.98]
 [0.02 0.98]]
The accuracy for Naive Bayes Classification is 87.90.
```

[186]:
```
M,acc = Confusion(X_test, y_test, neigh, 2)
VisualizeConfussion(M)
```

```
2597it [00:48, 54.06it/s]
```




```
[[0.36 0.64]
 [0.06 0.94]]
```

[127]:
```
print ("The accuracy for KNN is %s%%. " % str(acc)[: 5])
```

```
The accuracy for KNN is 88.13 %.
```

[158]:
```
# a1, a2 = ["Denied", 0.36, 0.64], ["Accepted", 0.06, 0.94]
a1, a2, a3 = ["True Denied", "True Accepted"], [0.02, 0.02], [0.98, 0.98]
```

[168]:
```
import plotly.graph_objects as go

fig = go.Figure(data = [go.Table(
    header = dict(values = [" ", "Predicted Denied", "Predicted Accepted"],
                  line_color = 'rebeccapurple',
                  fill_color = 'mediumpurple',
                  align = 'center'),
    cells = dict(values = [a1,
```

```
                        a2, a3],
                line_color = 'purple',
                fill_color = 'yellow',
                align = ['center', 'center'],
                font_size = 12))
])

fig.update_layout(width = 600, height = 400)
fig.show()
```