DATS 6401

Final Term Project Report

H1B Visa Status

Date:12/18/2022

Professor: Reza Jafari

Subject: Reza Jafari

Sagar Tripathi(G30209731)

The George Washington University

Table of Contents

Table of Contents

Topic	Page Number
Abstract	5
Introduction	5
Description of the dataset:	5
Pre-processing the Dataset:	8
Outlier Detection:	10
Principal Component Analysis (PCA):	11
Heatmap	12
Statistics	12
Graphs	14
Dash App with Plotly	20
Conclusion	23
Appendix	24
References	38

Table of figures and tables

Topic	Page Number
Img1: Dataset H1b18.csv	6
Img2a. example of the clean dataset	9
Img2b. Finding Unique values from dataset	9
Img3: clean dataset	10
Img4. PCA Analysis	11
Img5: Heatmap of the numerical variables	12
Img6: statistics of data	13
Img7: Statistics of data	14
Img8. Top companies providing H1B visas and their acceptance count.	14
Img9: Histogram Example	15

Img10. Scatter plot example	16
Img11. Pie Chart plot example	17
Img12. Line Chart plot example	17
Img13. Count plot example	18
Img14. Bar plot example	18
Img15. Box plot example	19
Img16a. Page1 About	20
Img16b. Page2 Data Upload	21
Img16c. Page3 Graphs	22
Img16d. Page4 Feature Engineering	23

Abstract:

Section 101(a)(15)(H) of the Immigration and Nationality Act permits American businesses to temporarily employ foreign workers in specialized occupations. The H-1B visa is the one in question. In addition to the application of specific knowledge, a specialty career requires a bachelor's degree or the equivalent in work experience. We have tried to visualize the data and derive insights from it, such as what the case status is, how the data is distributed, and what job titles are necessary for H1B visa status, among other questions. In order to see the data, I also built a dash app. The following sections will provide further information on how data is utilized to visualize the number of 2018 H1b visa applicants and attempt to provide basic answers to a few relevant queries.

Introduction:

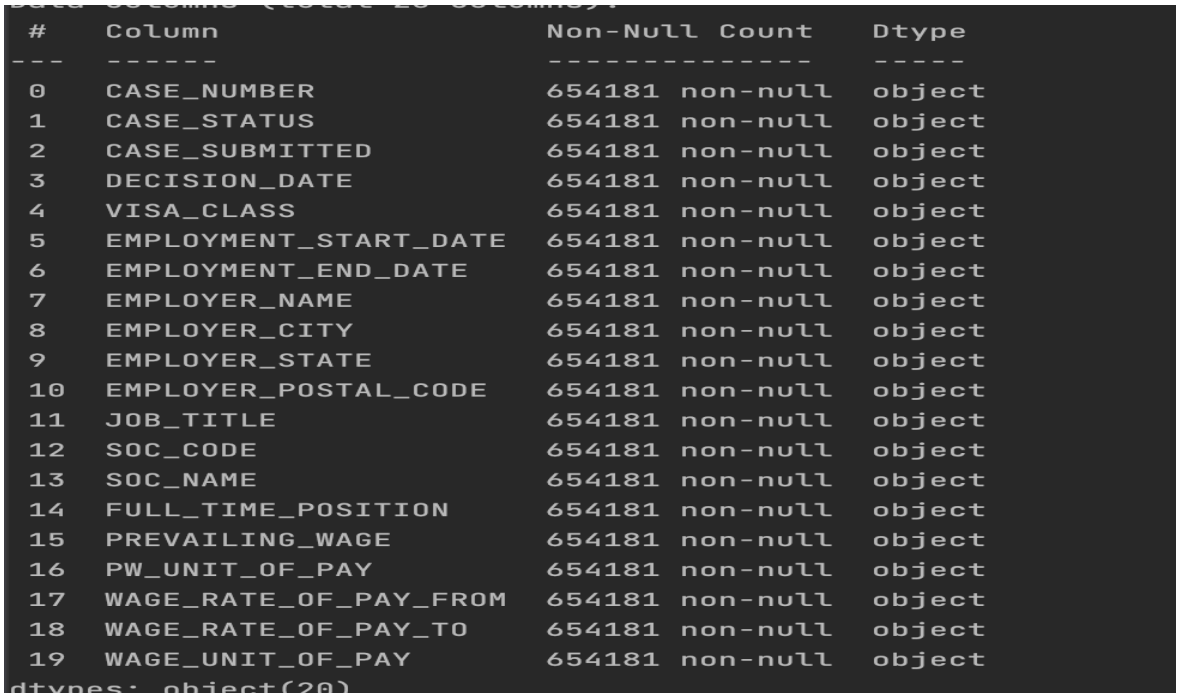
The Immigration and Nationality Act's section 101(a)(15)(H) authorizes U.S. firms to temporarily hire foreign workers in specialist occupations. This visa is known as the H-1B. A bachelor's degree or the equivalent in work experience is required for a specialty career in addition to the application of specialized knowledge. We have tried to visualize the data and get the insights from it like what are the case status?, How the data is distributed ?, What are the job titles required for H1b visa status etc. I have also bulit a dash app for visualizing the data. In below sections, we will learn more about how data is used to visulaize the number of applicants in 2018 H1b visa and try to asnswer few basic questions related to the dataset.

Description of the dataset:

Data from the employer's Labor Condition Application and case certification findings handled by the Office of Foreign Labor Certification are included in the H-1B Dataset chosen for this research (OFLC). In order to hire non-immigrant workers at a specific job occupation in an area of intended employment for a period of not longer than three years, a prospective H-1B employer must submit a Labor Condition Application (LCA) to the U.S. Department of Labor Employment and Training Administration (DOLETA).

Link of dataset: https://www.kaggle.com/datasets/jmpark746/h1b-visas?select=h1b18.csv.

There are 20 columns and about 654,360 rows. In order to determine if an employee would be granted an H1B visa or not, I have decided that the CASE-Status is the dependent variable, and the other factors are independent variables. If the employee is granted an H1b visa, the case status is the only factor that will determine whether they can work in the United States.



#	Column	Non-Null Count	Dtype
0	CASE_NUMBER	654181 non-null	object
1	CASE_STATUS	654181 non-null	object
2	CASE_SUBMITTED	654181 non-null	object
3	DECISION_DATE	654181 non-null	object
4	VISA_CLASS	654181 non-null	object
5	EMPLOYMENT_START_DATE	654181 non-null	object
6	EMPLOYMENT_END_DATE	654181 non-null	object
7	EMPLOYER_NAME	654181 non-null	object
8	EMPLOYER_CITY	654181 non-null	object
9	EMPLOYER_STATE	654181 non-null	object
10	EMPLOYER_POSTAL_CODE	654181 non-null	object
11	JOB_TITLE	654181 non-null	object
12	SOC_CODE	654181 non-null	object
13	SOC_NAME	654181 non-null	object
14	FULL_TIME_POSITION	654181 non-null	object
15	PREVAILING_WAGE	654181 non-null	object
16	PW_UNIT_OF_PAY	654181 non-null	object
17	WAGE_RATE_OF_PAY_FROM	654181 non-null	object
18	WAGE_RATE_OF_PAY_TO	654181 non-null	object
19	WAGE_UNIT_OF_PAY	654181 non-null	object

dtypes: object(20)

Img1: Dataset H1b18.csv

About the columns used for Visualization:

CASE_STATUS: Status of the case. A categorical variable with 4 unique values

CASE_STATUS: Status of the case. A categorical variable with 4 unique values

CASE_SUBMITTED: Case submission is a date time variable signifies when the application was submitted

DECISION_DATE: Decision date is a date time variable signifies when the decision was made for submitted application

VISA_CLASS: Visa class is categorical variable which describes the visa class of the applicant

EMPLOYMENT_START_DATE: Employment start date variable is date time variable. shows the employment start date of applicant

EMPLOYER_STATE: Employer state is also a categorical variable. show the state of the employer

EMPLOYER_NAME: Employer name is the categorical variable show the name of the employer

JOB_TITLE: Job title is a categorical variable shows the title of the job

FULL_TIME_POSITION: Full time Position is the categorical variable shows the if employee has full time job or not

PREVAILING_WAGE: Wage of the employee based on the PW_UNIT_OF_PAY

PW_UNIT_OF_PAY: Unit of payment can be year, month, weekly or biweekly

WAGE_RATE_OF_PAY_FROM: Wage rate to pay the employee

Pre-processing the Dataset:

we have started with pre-processing the data.

1. We loaded the dataset from Kaggle. (link:https://www.kaggle.com/datasets/jmpark746/h1b-visas?select=h1b18.csv). After loading the data, we gathered the information about the data.
2. We did some data-preprocessing, where we found out that there are 654,360 rows and 20 columns from that we have dropped 9 columns and kept only 50,000 rows for getting the insights from the data though the cleaning was applied in whole dataset.
3. Due to the size of the data and the string types of all the columns, cleaning the data took the longest.
   a. Columns such as CASE SUBMITTED, DECISION DATE, and EMPLOYMENT START DATE are typecast as date-time columns, and WAGE RATE OF PAY FORM and PREVAILING WAGE are typecast as Float columns.
   b. We tested the dataset for null values, but because none were present in the selected rows, we didn't change or remove any of them.
   c. We discovered certain identifiers, such as CASE NUMBER, but we eliminated them because they did not offer many insights into the data.
   d. Finding the unique values to know how the column is distributed.

```
  CASE_STATUS CASE_SUBMITTED DECISION_DATE VISA_CLASS EMPLOYMENT_START_DATE  \
0   CERTIFIED     2018-01-29    2018-02-02       H-1B            2018-07-28
1   CERTIFIED     2017-10-23    2017-10-27       H-1B            2017-11-06
2   CERTIFIED     2018-08-30    2018-09-06       H-1B            2018-09-10
4   CERTIFIED     2018-08-31    2018-09-07       H-1B            2018-09-07
5   CERTIFIED     2018-05-22    2018-05-29       H-1B            2018-05-29
```

Img2a. example of the clean dataset

```
5                          5
FULL_TIME_POSITION             2
CASE_STATUS                    4
VISA_CLASS                     4
PW_UNIT_OF_PAY                 5
month_Decision                12
month_CASE_SUBMITTED          12
month_EMPLOYMENT_START        12
month_EMPLOYMENT_End          12
EMPLOYER_STATE                55
DECISION_DATE                297
CASE_SUBMITTED              1074
EMPLOYMENT_START_DATE       1215
EMPLOYMENT_END_DATE         1429
EMPLOYER_NAME              10690
WAGE_RATE_OF_PAY_FROM      13132
PREVAILING_WAGE            13876
JOB_TITLE                  15884
dtype: int64
```

Img2b. Finding Unique values from dataset

```
Data columns (total 12 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   CASE_STATUS           654181 non-null  object
 1   CASE_SUBMITTED        654181 non-null  datetime64[ns]
 2   DECISION_DATE         654181 non-null  datetime64[ns]
 3   VISA_CLASS            654181 non-null  object
 4   EMPLOYMENT_START_DATE 654181 non-null  datetime64[ns]
 5   EMPLOYER_NAME         654181 non-null  object
 6   EMPLOYER_STATE        654181 non-null  object
 7   JOB_TITLE             654181 non-null  object
 8   FULL_TIME_POSITION    654181 non-null  object
 9   PREVAILING_WAGE       654181 non-null  float64
 10  PW_UNIT_OF_PAY        654181 non-null  object
 11  WAGE_RATE_OF_PAY_FROM 654181 non-null  float64
dtypes: datetime64[ns](3), float64(2), object(7)
memory usage: 64.9+ MB
None
```

Img3: clean dataset

Outlier Detection:

We didn't find any outlier, but outliers can be handled via z-score method.

A z-score can be placed on a normal distribution curve. Z-scores range from -3 standard deviations (which would fall to the far left of the normal distribution curve) up to +3 standard deviations (which would fall to the far right of the normal distribution curve). In order to use a z-score, you need to know the mean μ and the population standard deviation σ.

Formula to calculate Z score:

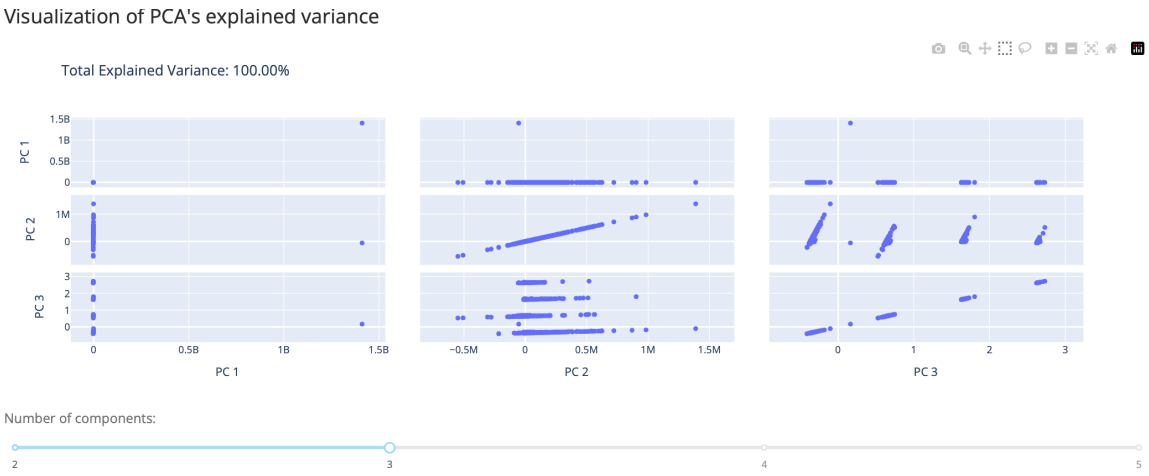
$z = (x – \mu) / \sigma$

Principal Component Analysis (PCA):

Principal component analysis, or PCA, is a statistical procedure that allows you to summarize the information content in large data tables by means of a smaller set of “summary indices” that can be more easily visualized and analyzed.

HOW DO YOU DO A PRINCIPAL COMPONENT ANALYSIS?

1. Standardize the range of continuous initial variables
2. Compute the covariance matrix to identify correlations
3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components
4. Create a feature vector to decide which principal components to keep
5. Recast the data along the principal component's axes

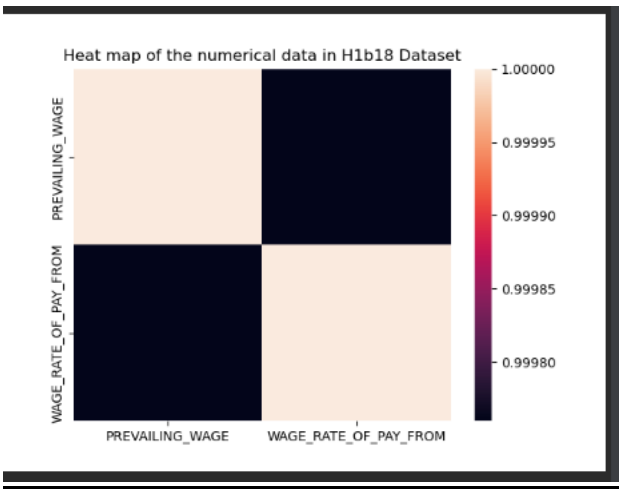


Img4. PCA Analysis

Results of PCA analysis: We can see that our first three principal components explain the majority of the variance in this dataset (100%)! This is an indication of the total information represented compared to the original data. Hence, we can drop rest of the components.

Heatmap:

A heatmap (aka heat map) depicts values for a main variable of interest across two axis variables as a grid of colored squares. The axis variables are divided into ranges like a bar chart or histogram, and each cell's color indicates the value of the main variable in the corresponding cell range.

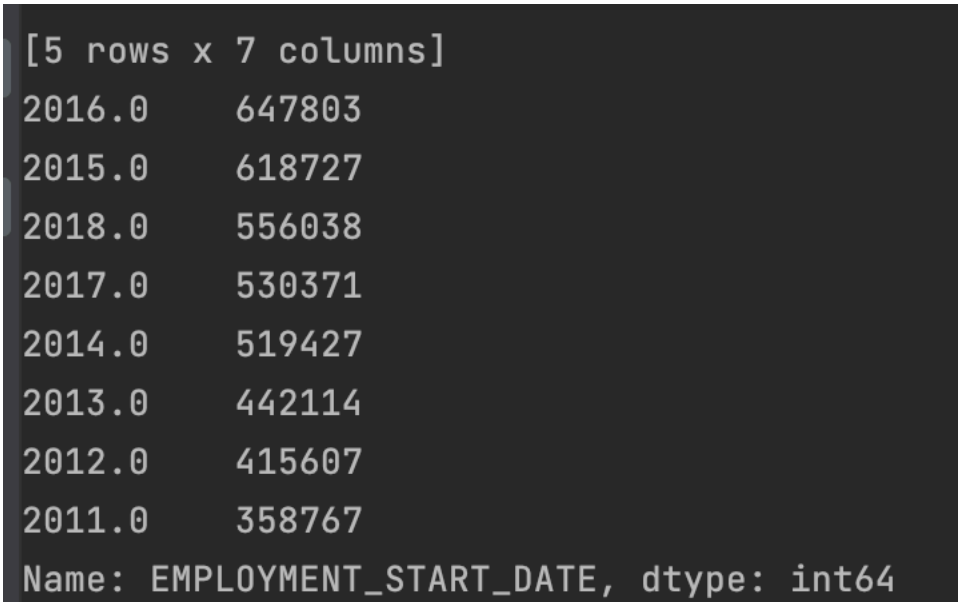


Img5: Heatmap of the numerical variables

In our dataset out of all the variables there are two continuous/ numerical are there which are not correlated with each other.

Statistics:

We have taken the insights from the whole dataset hence the values are large.



Img6: statistics of data

From the series above, we can see that there are about 500,000 applicants every year. We can also see that there has been a decrease in the number of applications in 2017 and 2018 (Trump's Presidency) The figure below highlights the fall in the number of applicants during 2016.

While combining the dataset from 2016 with 2018 h1b18 dataset It appears that the percentage of cases denied is really low. Upon further research, the denied percentage is not indicative of all the applications denied in the H-1B application process. Additional information is provided by the https://www.uscis.gov/sites/default/files/USCIS/Resources/Reports%20and%20Studies/Immigration%20Forms%20Data/BAHA/non-immigrant-worker-rfe-h-1b-quarterly-data-fy2015-fy2019-q1.pdf.

```
CERTIFIED                                          3615378
CERTIFIED-WITHDRAWN                                 241703
WITHDRAWN                                           122466
DENIED                                              109289
PENDING QUALITY AND COMPLIANCE REVIEW - UNASSIGNED      15
REJECTED                                                 2
INVALIDATED                                              1
Name: CASE_STATUS, dtype: int64
```

Img7: Statistics of data

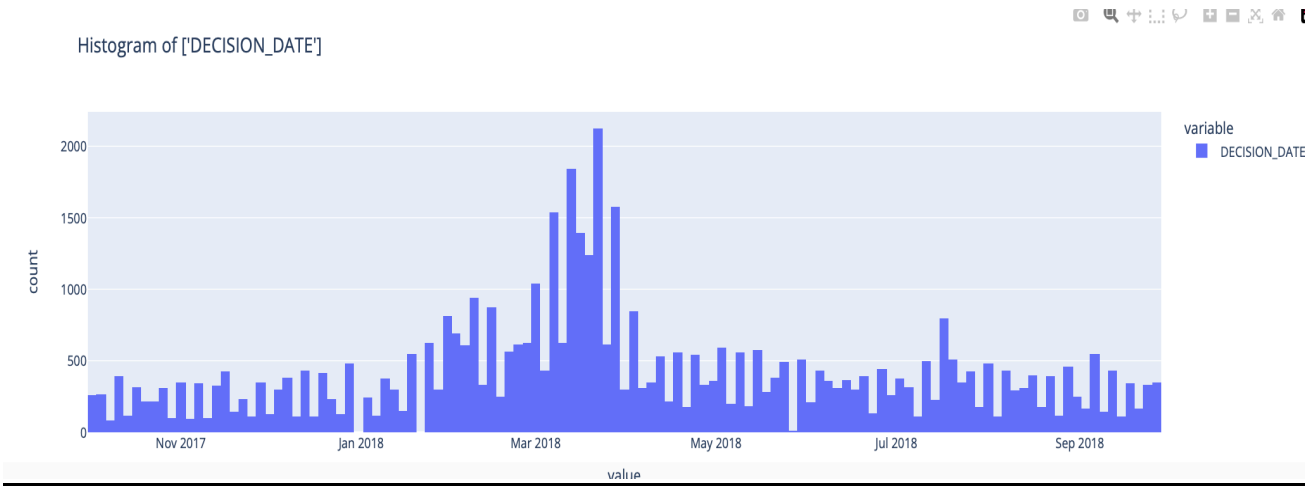
```
DELOITTE CONSULTING LLP                    13856
TATA CONSULTANCY SERVICES LIMITED          13039
COGNIZANT TECHNOLOGY SOLUTIONS US CORP     11071
INFOSYS LIMITED                             9627
GOOGLE LLC                                  5697
TECH MAHINDRA (AMERICAS),INC.               5344
WIPRO LIMITED                               5192
ERNST & YOUNG U.S. LLP                      5179
ACCENTURE LLP                               5130
CAPGEMINI AMERICA INC                       4908
IBM CORPORATION                             4583
MICROSOFT CORPORATION                       4436
AMAZON.COM SERVICES, INC.                   3721
LARSEN & TOUBRO INFOTECH LIMITED            2683
SYNTEL INC                                  2674
IBM INDIA PRIVATE LIMITED                   2315
FACEBOOK, INC.                              2280
HCL AMERICA, INC.                           2013
JPMORGAN CHASE & CO.                        1669
APPLE INC.                                  1565
```

Img8. Top companies providing H1B visas and their acceptance count.

Graphs:

Histogram:

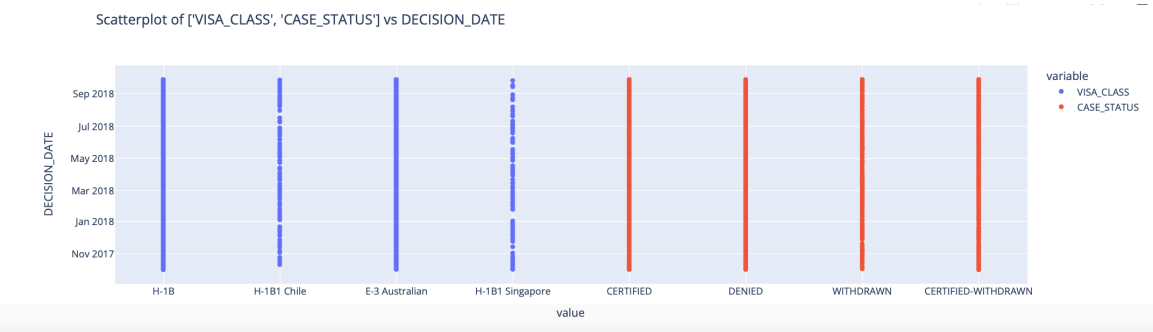
A histogram is a plot that lets you discover, and show, the underlying frequency distribution (shape) of a set of continuous data. Example of Decision Date distribution mentioned below show how the decision date is distribute along x-axis which shows the date.



Img9: Histogram Example

Scatter Plot:

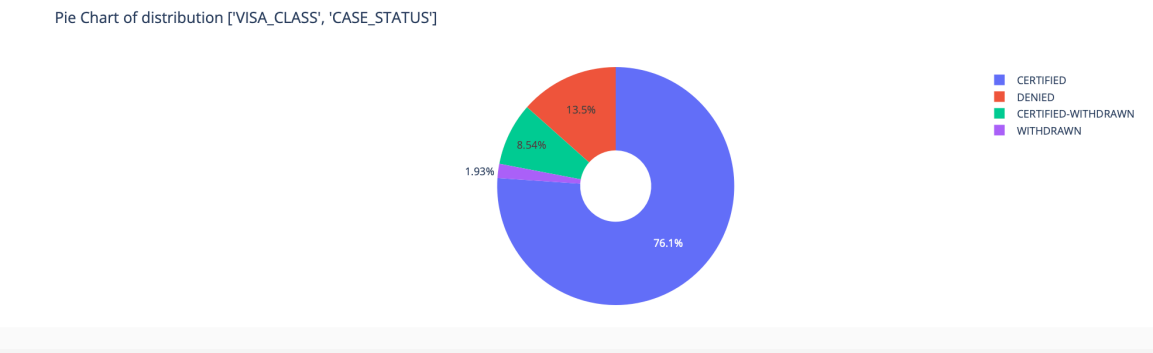
A scatter chart, also called a scatter plot, is a chart that shows the relationship between two variables. They are an incredibly powerful chart type, allowing viewers to immediately understand a relationship or trend, which would be impossible to see in almost any other form. An example of scatter plot showing the distribution of Visa status and case status on different decision dates. We can Observe that H1b Visa is mostly applied in all months and there many H1b Visas are certified.



Img10. Scatter plot example

Pie Chart:

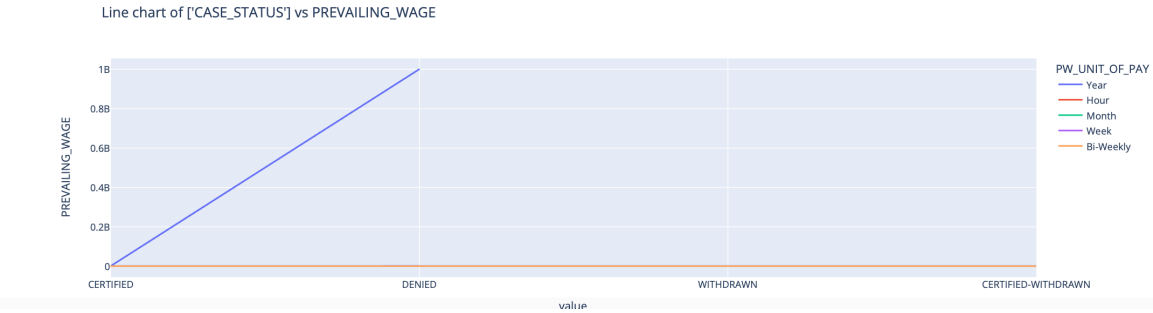
A pie chart is a graphical representation technique that displays data in a circular-shaped graph. It is a composite static chart that works best with few variables. Pie charts are often used to represent sample data—with data points belonging to a combination of different categories. For example Plotting the percentage of the Visa Status Approved cases. We can observe that 76.1% of H1b visa are certified.



Img11. Pie Chart plot example

Line Chart:

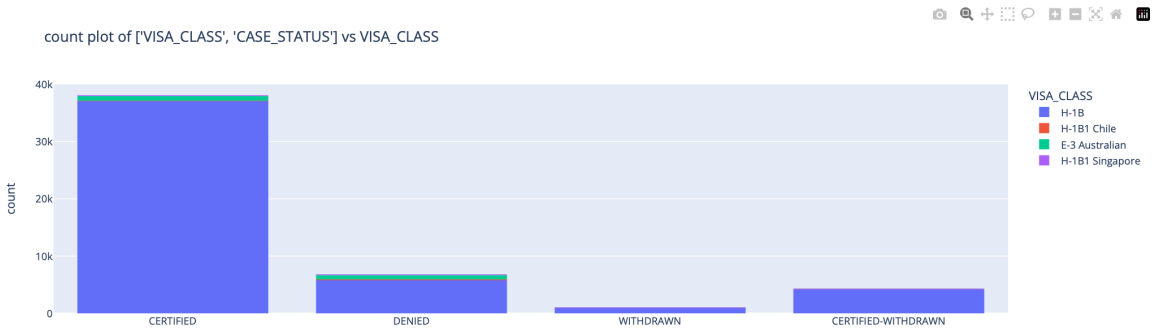
A line chart, also referred to as a line graph or a line plot, connects a series of data points using a line. This chart type presents sequential values to help you identify trends. Most of the time, the x-axis (horizontal axis) represents a sequential progression of values. For example: Case status and pay segregated based on the pay unit (Weekly, Biweekly, Monthly and yearly.) we can observe that yearly wages goes to 1 Billion which is extremely high.



Img12. Line Chart plot example

Count Plot:

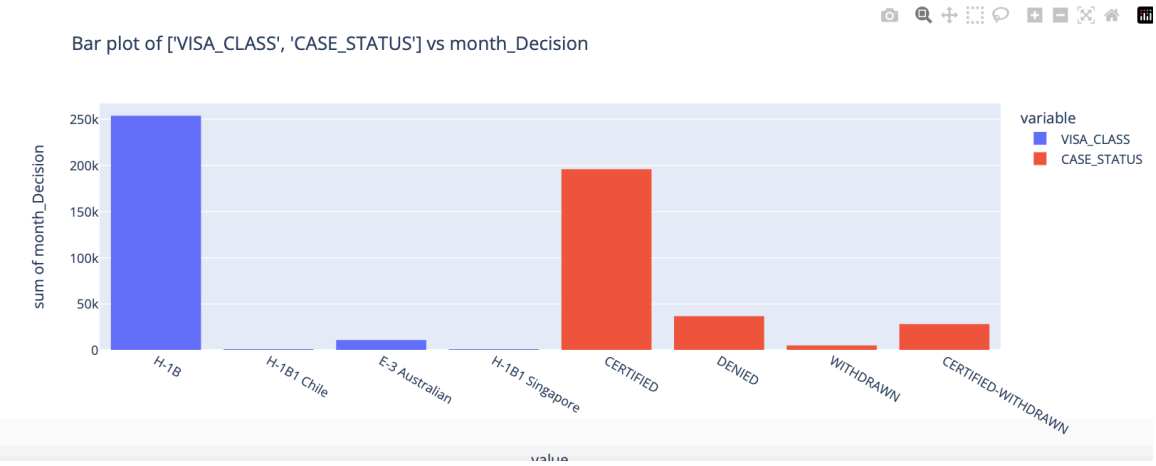
The count plot is used to represent the occurrence(counts) of the observation present in the categorical variable. For example: Visa class distribution, we can see that there are mostly H1B visa are applied for work purpose in united states among all 4 categories.



Img13. Count plot example

Bar Plot:

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. From the example below we can see that sum of monthly decision made for H1b visa are the highest, similarly we can also observe that certified Visas quantity is also highest among other categories of case status.



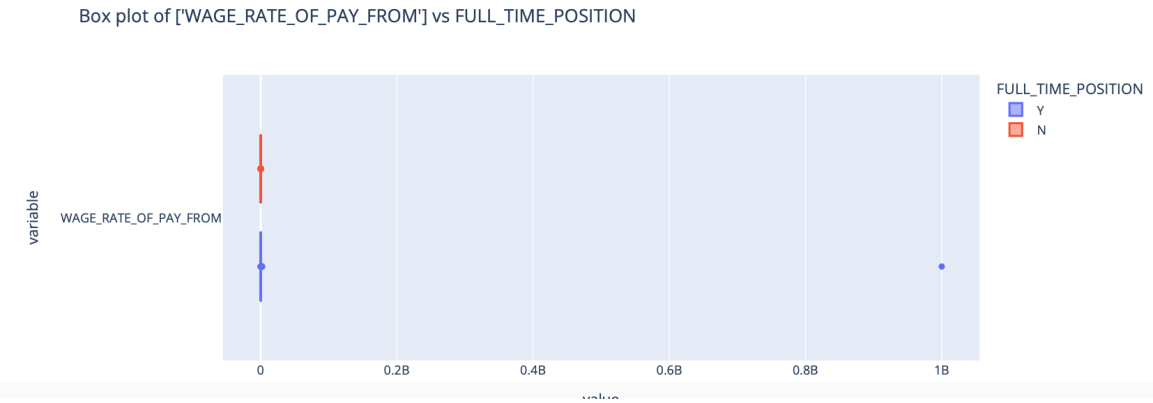
Img14. Bar plot example

Pair Plot:

Pair plot visualizes given data to find the relationship between them where the variables can be continuous or categorical. Plot pairwise relationships in a dataset.

Box Plot:

A box plot is a graph that displays information from a five-number summary that includes one of the central tendency measures. It does not clearly depict the distribution as a stem and leaf plot, or histogram does. However, its primary function is to determine whether a distribution is skewed and whether any potential outliers' abnormal observations are present in the data collection. We can observe the outliers from the box plot about the income of the full-time position candidates. We can see that for those who have a full time position the value is approx. 1 billion, whereas those who don't have full time jobs there is no outliers in that.



Img15. Box plot example

Regression Plot:

Regression plots as the name suggests creates a regression line between 2 parameters and helps to visualize their linear relationships. As there are not

many continuous variables are there which are not correlated hence, we can find new features later in the project and try applying regression plot.

Dash App with Plotly:

Dash apps significantly broaden the notion of what is achievable in a standard "dashboard" by providing a point-and-click interface to models written in Python. Data scientists and engineers give business operators access to sophisticated Python analytics through Dash apps.

Dash with Plotly where Plotly offers scientific graphing libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST as well as online graphing, analytics, and statistics capabilities for individuals and groups.

My Dash app consist of 4 pages.

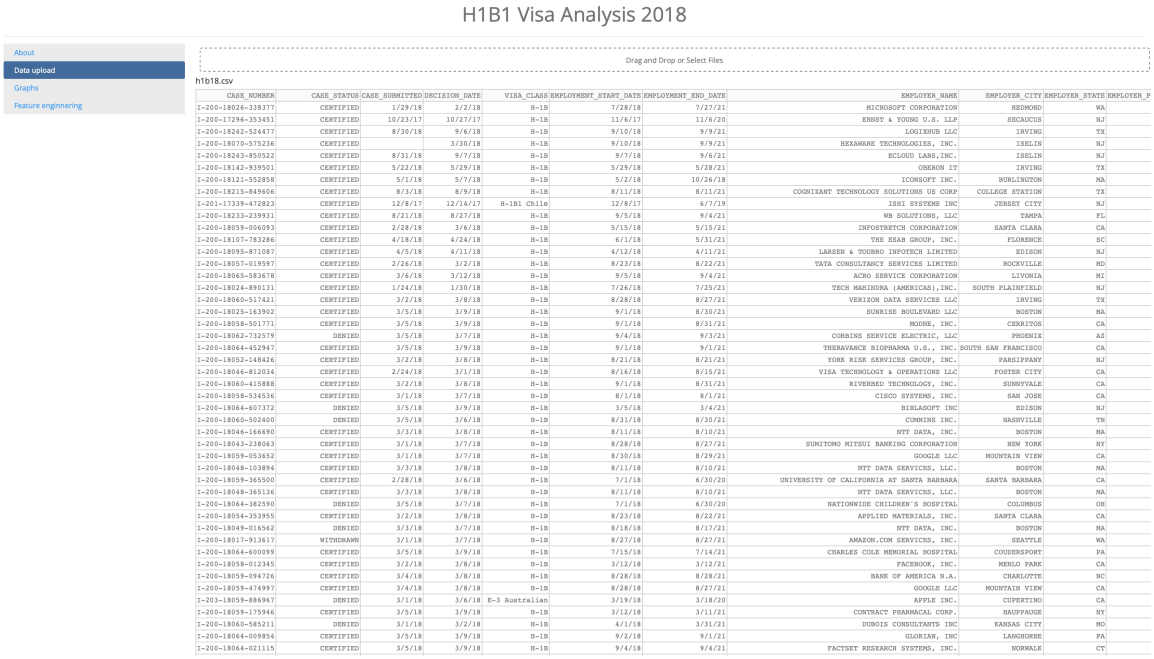
GCP link: https://dashapp-j7mpwic3bq-pd.a.run.app

Pages:

1. About: In this page I have given a brief about the dataset used and via selecting feature name from the drop-down menu, One can know about the variables.

Img16a. Page1 About

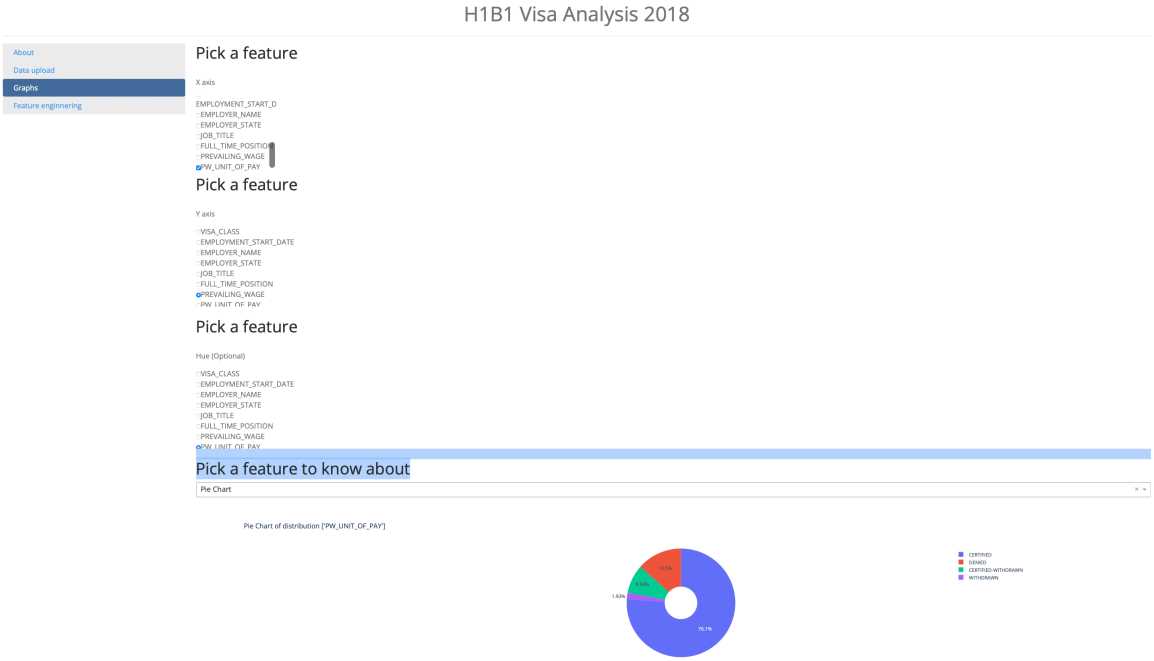
2. Data Upload: Via this page one upload the dataset in use and can see how the data is structured and the variables in datasets. There is a drag and drop / select button to upload the dataset.



CASE_NUMBER	CASE_STATUS	CASE_SUBMITTED	DECISION_DATE	VISA_CLASS	EMPLOYMENT_START_DATE	EMPLOYMENT_END_DATE	EMPLOYER_NAME	EMPLOYER_CITY	EMPLOYER_STATE	EMPLOYER_P
I-200-18026-338377	CERTIFIED	1/29/18	2/2/18	H-1B	7/28/18	7/27/21	MICROSOFT CORPORATION	REDMOND	WA	
I-200-17296-353451	CERTIFIED	10/23/17	10/27/17	H-1B	11/6/17	11/6/20	ERNST & YOUNG U.S. LLP	SECAUCUS	NJ	
I-200-18242-524477	CERTIFIED	8/30/18	9/6/18	H-1B	9/10/18	9/9/21	LOGIXHUB LLC	IRVING	TX	
I-200-18070-575236	CERTIFIED		3/30/18	H-1B	9/10/18	9/9/21	HEXAWARE TECHNOLOGIES, INC.	ISELIN	NJ	
I-200-18243-850522	CERTIFIED	8/31/18	9/7/18	H-1B	9/7/18	9/6/21	ECLOUD LABS,INC.	ISELIN	NJ	
I-200-18142-939501	CERTIFIED	5/22/18	5/29/18	H-1B	5/29/18	5/28/21	OBERON IT	IRVING	TX	
I-200-18121-552858	CERTIFIED	5/1/18	5/7/18	H-1B	5/2/18	10/26/18	ICONSOFT INC.	BURLINGTON	MA	
I-200-18215-849606	CERTIFIED	8/3/18	8/9/18	H-1B	8/11/18	8/11/21	COGNIZANT TECHNOLOGY SOLUTIONS US CORP	COLLEGE STATION	TX	
I-201-17339-472823	CERTIFIED	12/8/17	12/14/17	H-1B1 Chile	12/8/17	6/7/19	ISHI SYSTEMS INC	JERSEY CITY	NJ	
I-200-18233-239931	CERTIFIED	8/21/18	8/27/18	H-1B	9/5/18	9/4/21	WB SOLUTIONS, LLC	TAMPA	FL	
I-200-18059-006093	CERTIFIED	2/28/18	3/6/18	H-1B	5/15/18	5/15/21	INFOSTRETCH CORPORATION	SANTA CLARA	CA	
I-200-18107-783286	CERTIFIED	4/18/18	4/24/18	H-1B	6/1/18	5/31/21	THE ESAB GROUP, INC.	FLORENCE	SC	
I-200-18095-871087	CERTIFIED	4/5/18	4/11/18	H-1B	4/12/18	4/11/21	LARSEN & TOUBRO INFOTECH LIMITED	EDISON	NJ	
I-200-18057-019597	CERTIFIED	2/26/18	3/2/18	H-1B	8/23/18	8/22/21	TATA CONSULTANCY SERVICES LIMITED	ROCKVILLE	MD	
I-200-18065-583678	CERTIFIED	3/6/18	3/12/18	H-1B	9/5/18	9/4/21	ACRO SERVICE CORPORATION	LIVONIA	MI	
I-200-18024-890131	CERTIFIED	1/24/18	1/30/18	H-1B	7/26/18	7/25/21	TECH MAHINDRA (AMERICAS),INC.	SOUTH PLAINFIELD	NJ	
I-200-18060-517421	CERTIFIED	3/2/18	3/8/18	H-1B	8/28/18	8/27/21	VERIZON DATA SERVICES LLC	IRVING	TX	
I-200-18025-163902	CERTIFIED	3/5/18	3/9/18	H-1B	9/1/18	8/30/21	SUNRISE BOULEVARD LLC	BOSTON	MA	
I-200-18058-501771	CERTIFIED	3/5/18	3/9/18	H-1B	9/1/18	8/31/21	MOOHS, INC.	CERRITOS	CA	
I-200-18062-732579	DENIED	3/5/18	3/7/18	H-1B	9/4/18	9/3/21	CORBINS SERVICE ELECTRIC, LLC	PHOENIX	AZ	
I-200-18064-452947	CERTIFIED	3/5/18	3/9/18	H-1B	9/1/18	9/1/21	THERAVANCE BIOPHARMA U.S., INC.	SOUTH SAN FRANCISCO	CA	
I-200-18052-148426	CERTIFIED	3/2/18	3/8/18	H-1B	8/21/18	8/21/21	YORK RISK SERVICES GROUP, INC.	PARSIPPANY	NJ	
I-200-18046-812034	CERTIFIED	2/24/18	3/1/18	H-1B	8/16/18	8/15/21	VISA TECHNOLOGY & OPERATIONS LLC	FOSTER CITY	CA	
I-200-18060-415888	CERTIFIED	3/2/18	3/8/18	H-1B	9/1/18	8/31/21	RIVERBED TECHNOLOGY, INC.	SUNNYVALE	CA	
I-200-18058-534536	CERTIFIED	3/1/18	3/7/18	H-1B	8/1/18	8/1/21	CISCO SYSTEMS, INC.	SAN JOSE	CA	
I-200-18064-407372	DENIED	3/5/18	3/9/18	H-1B	3/5/18	3/4/21	BIKLASOFT INC	EDISON	NJ	
I-200-18060-502400	DENIED	3/5/18	3/6/18	H-1B	8/31/18	8/30/21	CUMMINS INC.	NASHVILLE	TN	
I-200-18046-166690	CERTIFIED	3/3/18	3/8/18	H-1B	8/11/18	8/10/21	NTT DATA, INC.	BOSTON	MA	
I-200-18043-238063	CERTIFIED	3/1/18	3/7/18	H-1B	8/28/18	8/27/21	SUMITOMO MITSUI BANKING CORPORATION	NEW YORK	NY	
I-200-18059-053652	CERTIFIED	3/1/18	3/7/18	H-1B	8/30/18	8/29/21	GOOGLE LLC	MOUNTAIN VIEW	CA	
I-200-18048-103894	CERTIFIED	3/3/18	3/8/18	H-1B	8/11/18	8/10/21	NTT DATA SERVICES, LLC.	BOSTON	MA	
I-200-18059-365500	CERTIFIED	2/28/18	3/6/18	H-1B	7/1/18	6/30/20	UNIVERSITY OF CALIFORNIA AT SANTA BARBARA	SANTA BARBARA	CA	
I-200-18048-365136	CERTIFIED	3/3/18	3/8/18	H-1B	8/11/18	8/10/21	NTT DATA SERVICES, LLC.	BOSTON	MA	
I-200-18064-382590	DENIED	3/5/18	3/7/18	H-1B	7/1/18	6/30/20	NATIONWIDE CHILDREN'S HOSPITAL	COLUMBUS	OH	
I-200-18054-353955	CERTIFIED	3/2/18	3/8/18	H-1B	8/23/18	8/22/21	APPLIED MATERIALS, INC.	SANTA CLARA	CA	
I-200-18049-016562	DENIED	3/3/18	3/7/18	H-1B	8/18/18	8/17/21	NTT DATA, INC.	BOSTON	MA	
I-200-18017-913617	WITHDRAWN	3/1/18	3/7/18	H-1B	8/27/18	8/27/21	AMAZON.COM SERVICES, INC.	SEATTLE	WA	
I-200-18064-600099	CERTIFIED	3/5/18	3/9/18	H-1B	7/15/18	7/14/21	CHARLES COLE MEMORIAL HOSPITAL	COUDERSPORT	PA	
I-200-18058-012345	CERTIFIED	3/2/18	3/8/18	H-1B	3/12/18	3/12/21	FACEBOOK, INC.	MENLO PARK	CA	
I-200-18059-094726	CERTIFIED	3/4/18	3/8/18	H-1B	8/28/18	8/28/21	BANK OF AMERICA N.A.	CHARLOTTE	NC	
I-200-18059-474997	CERTIFIED	3/4/18	3/8/18	H-1B	8/28/18	8/27/21	GOOGLE LLC	MOUNTAIN VIEW	CA	
I-203-18059-886967	DENIED	3/1/18	3/6/18	E-3 Australian	3/19/18	3/18/20	APPLE INC.	CUPERTINO	CA	
I-200-18059-175946	CERTIFIED	3/5/18	3/9/18	H-1B	3/12/18	3/11/21	CONTRACT PHARMACAL CORP.	HAUPPAUGE	NY	
I-200-18060-585211	DENIED	3/1/18	3/2/18	H-1B	4/1/18	3/31/21	DUBOIS CONSULTANTS INC	KANSAS CITY	MO	
I-200-18064-009854	CERTIFIED	3/5/18	3/9/18	H-1B	9/2/18	9/1/21	GLORIAN, INC	LANGHORNE	PA	
I-200-18064-021115	CERTIFIED	3/5/18	3/9/18	H-1B	9/4/18	9/4/21	FACTSET RESEARCH SYSTEMS, INC.	NORWALK	CT	

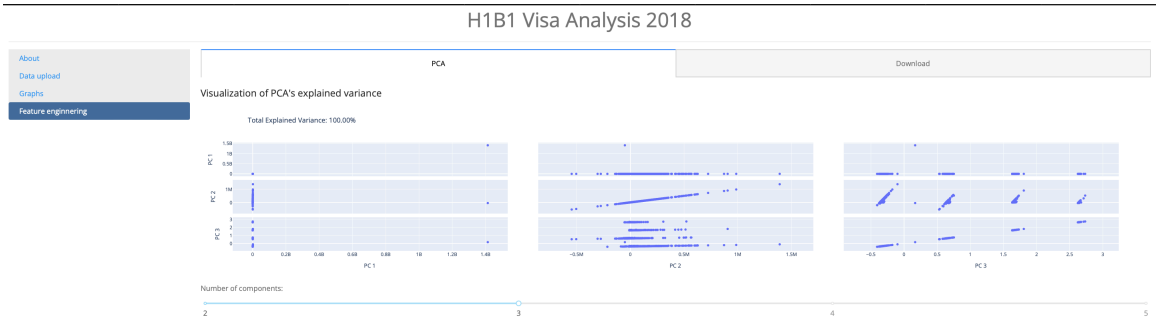
Img16b. Page2 Data Upload

3. Graphs: In this page a user has multiple options to plot the data in use and see the output via selecting the features. There are checkboxes, radio buttons, drop down menu and graphical representations.



Img16c. Page3 Graphs

4. The last page is for feature engineering this page is divided into two tabs on is PCA analysis where dimension reduction can help to see the expected variance and how it is plotted. We have slider which takes input for number of components. In tab2 we have a download option for the cleaned data.



Img16d. Page4 Feature Engineering

Conclusion:

As the “age of Big Data” kicks into high gear, visualization is an increasingly key tool to make sense of the trillions of rows of data generated every day. Data visualization helps to tell stories by curating data into a form easier to understand, highlighting the trends and outliers. A good visualization tells a story, removing the noise from data and highlighting useful information.

However, it’s not simply as easy as just dressing up a graph to make it look better or slapping on the “info” part of an infographic. Effective data visualization is a delicate balancing act between form and function. The plainest graph could be too boring to catch any notice, or it make tell a powerful point; the most stunning visualization could utterly fail at conveying the right message or it could speak volumes. The data and the visuals need to work together, and there’s an art to combining great analysis with great storytelling, with the help of this project I can understand about H1b visa dataset and able to rectify the status maintenance is necessary, we are able to see there are various factors that help to get you the H1b visa from which Full-time job is necessary. Also, after learning basic functionality of the graphs, we can plot the graph and able to give insights about the dataset used (H1b18.csv). By comparing the wages for non-data science jobs and data science jobs, it appears that there is not a massive difference in the median and mean wage of data science jobs. Both the median appears to be between 60,000 and 80,000. However, the non-data science jobs tend to have a larger

interquartile range which could be explained by the diverse jobs that other H-1B applicants had. As an aspiring data scientist, I thought that it would be interesting to explore the H-1B application process for data scientist. There were 63,503 applications for data scientists. We can observe that the top three companies are all from the technology sector. In fact, most of the companies that apply for H-1B visas are from the technology sector. In future we, will try to apply statistical models to predict the acceptance rate of H1b visa.

Appenxdix:

App.py

```
import dash
import pandas as pd
import dash_bootstrap_components as dbc
from dash import dcc, html, callback, Output, Input
import plotly.express as px
import base64
from scipy.special import expit
import base64
import datetime
import io
import plotly.graph_objs as go
import cufflinks as cf
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import seaborn as sns
from plotly.offline import download_plotlyjs, init_notebook_mode, plot,
iplot
init_notebook_mode(connected=True)
import plotly as plotly
import plotly.express as px
import plotly.graph_objs as go
import plotly.io as pio
import plotly.figure_factory as ff
import dash_html_components as html
import base64
import io
import plotly.graph_objs as go
import dash_table

# data loading


pd.set_option("display.max_rows", None, "display.max_columns", None)

data = pd.read_csv('h1b18.csv')
data.dropna(inplace=True)
print(data.info())
```

```
# removing data which is not necessary
data['CASE_SUBMITTED'] = pd.to_datetime(data['CASE_SUBMITTED'])
data['DECISION_DATE'] = pd.to_datetime(data['DECISION_DATE'])
data['EMPLOYMENT_START_DATE'] =
pd.to_datetime(data['EMPLOYMENT_START_DATE'])
data['EMPLOYMENT_END_DATE'] =
pd.to_datetime(data['EMPLOYMENT_END_DATE'])

data["PREVAILING_WAGE"] = data["PREVAILING_WAGE"].str.replace(",",
"").astype(float)
data["WAGE_RATE_OF_PAY_FROM"] =
data["WAGE_RATE_OF_PAY_FROM"].str.replace(",", "").astype(float)
data["WAGE_RATE_OF_PAY_TO"] =
data["WAGE_RATE_OF_PAY_TO"].str.replace(",", "").astype(float)

# print(data.nunique().sort_values(ascending=True))
data = data.drop(['SOC_CODE', 'EMPLOYMENT_END_DATE', 'SOC_NAME',
'CASE_NUMBER', 'EMPLOYER_POSTAL_CODE', 'EMPLOYER_CITY',
                  'WAGE_RATE_OF_PAY_TO', 'WAGE_UNIT_OF_PAY'], axis=1)
data.dropna(inplace=True)

df_heatmap = data.groupby('EMPLOYER_STATE')['CASE_STATUS'].count()
print(df_heatmap)
accept_arr = data['CASE_STATUS'].value_counts()
print(accept_arr)
accept_val = list(accept_arr)[0:4]
accept_label = list(accept_arr.index)[0:4]

sns.heatmap(data.corr())
plt.title("Heat map of the numerical data in H1b18 Dataset")
plt.show()


print(data.describe())
print(data. info())
app = dash.Dash(__name__, use_pages=True,
external_stylesheets=[dbc.themes.SPACELAB])
sidebar = dbc.Nav(
            [
                dbc.NavLink(
                    [
                        html.Div(page["name"], className="ms-2"),
                    ],
                    href=page["path"],
                    active="exact",
                )
                for page in dash.page_registry.values()
            ],
            vertical=True,
            pills=True,
            className="bg-light",
```

```
)
app.layout = dbc.Container([
    dbc.Row([
        dbc.Col(html.Div("H1B1 Visa Analysis 2018",
                         style={'fontSize':50, 'textAlign':'center'}))
    ]),

    html.Hr(),

    dbc.Row(
        [
            dbc.Col(
                [
                    sidebar
                ], xs=4, sm=4, md=2, lg=2, xl=2, xxl=2),

            dbc.Col(
                [
                    dash.page_container
                ], xs=8, sm=8, md=10, lg=10, xl=10, xxl=10)
        ]
    )
], fluid=True)

app.run_server(
    port=8061,
    host='0.0.0.0'
)
```

Page1

```
import dash
from dash import dcc, html, callback, Output, Input
import plotly.express as px
import dash_bootstrap_components as dbc
import pandas as pd

dash.register_page(__name__, path='/', name='About')  # '/' is home page

# page 1 data

pd.set_option("display.max_rows", None, "display.max_columns", None)

data = pd.read_csv('h1b18.csv')
data.dropna(inplace=True)

# removing data which is not necessary
data['CASE_SUBMITTED'] = pd.to_datetime(data['CASE_SUBMITTED'])
data['DECISION_DATE'] = pd.to_datetime(data['DECISION_DATE'])
data['EMPLOYMENT_START_DATE'] =
pd.to_datetime(data['EMPLOYMENT_START_DATE'])
data['EMPLOYMENT_END_DATE'] =
```

```
pd.to_datetime(data['EMPLOYMENT_END_DATE'])

data["PREVAILING_WAGE"] = data["PREVAILING_WAGE"].str.replace(",",
"").astype(float)
data["WAGE_RATE_OF_PAY_FROM"] =
data["WAGE_RATE_OF_PAY_FROM"].str.replace(",", "").astype(float)
data["WAGE_RATE_OF_PAY_TO"] =
data["WAGE_RATE_OF_PAY_TO"].str.replace(",", "").astype(float)

# print(data.nunique().sort_values(ascending=True))
data = data.drop(['SOC_CODE', 'EMPLOYMENT_END_DATE', 'SOC_NAME',
'CASE_NUMBER', 'EMPLOYER_POSTAL_CODE', 'EMPLOYER_CITY',
                  'WAGE_RATE_OF_PAY_TO', 'WAGE_UNIT_OF_PAY'], axis=1)
data.dropna(inplace=True)


# print(data.head())


layout = html.Div([
    dbc.Row(
                    [
                        html.Div([
                            html.H1('About'),
                            html.Div([
                                html.P('H-1B visa is a visa in the
United States under the Immigration and Nationality Act, section
101(a)(15)(H) that allows U.S. employers to temporarily employ foreign
workers in specialty occupations.'
                                       ' A specialty occupation requires
the application of specialized knowledge and a bachelor's degree or the
equivalent of work experience.'),
                                html.P('The H-1B Dataset selected for
this project contains data from employer's Labor Condition Application
and the case certification determinations processed by the Office of
Foreign Labor Certification (OFLC). '
                                       'The Labor Condition Application
(LCA) is a document that a perspective H-1B employer files with U.S.
Department of Labor Employment and Training Administration (DOLETA) when
it seeks to employ non-immigrant '
                                       'workers at a specific job
occupation in an area of intended employment for not more than three
years. '
                                       'The datasets are from the
Department of Labors website.'),
                            ])
                        ])
                    ]),
        dbc.Row(
            [
                html.Div(
                    [
                        html.H1('Pick a feature to know about'),
```

```
                        dcc.Dropdown(options=data.columns,
                                     id='columns_names',
placeholder='Select Symbol...'),
                    ],
                )]),
                html.Br(),
                dbc.Row(
                    [
                        html.Div(id='My_out'),
                    ]),


    ]
)


@callback(
    Output(component_id='My_out', component_property='children'),
    Input(component_id='columns_names', component_property='value')
)
def update_theinput(value):
    feature_explain = {"CASE_STATUS": "Status of the case. A categorical
variable with 4 unique values ",
                       "CASE_SUBMITTED": " Case submission is a date
time variable signifies when the application was submitted",
                       "DECISION_DATE": "Decision date is a date time
variable signifies when the decision was made for submitted
application",
                       "VISA_CLASS": "Visa class is categorical variable
which descibes the visa class of the applicant",
                       "EMPLOYMENT_START_DATE": "Employement start date
variable is date time variable. shows the employment start date of
applicant",
                       "EMPLOYER_STATE": "Employer state is also a
categorical vairable. show the state of the employer",
                       "EMPLOYER_NAME": "Employer name is the
categorical variable show the name of the employer",
                       "JOB_TITLE": "Job title is a categorical variable
shows the title of the job ",
                       "FULL_TIME_POSITION": "Full time Position is the
categorical variable shows the if employee has full time job or not",
                       "PREVAILING_WAGE": "Wage of the employee based on
the PW_UNIT_OF_PAY",
                       "PW_UNIT_OF_PAY": "Unit of payment can be year,
month, weekly or biweekly",
                       "WAGE_RATE_OF_PAY_FROM": "wage rate to pay the
employee"}
    return f"{feature_explain.get(value)}"
```

Page2:

```
import base64
import io
import plotly.graph_objs as go
import cufflinks as cf

import dash
import dash_html_components as html
import dash_table
import pandas as pd

import dash_bootstrap_components as dbc
import dash
from dash import html, dcc, callback, Input, Output

dash.register_page(__name__)
colors = {"graphBackground": "#F5F5F5", "background": "#ffffff", "text":
"#000000"}

layout = html.Div([
    dcc.Upload(
            id="upload-data",
            children=html.Div(["Drag and Drop or ", html.A("Select
Files")]),
            style={
                "width": "100%",
                "height": "60px",
                "lineHeight": "60px",
                "borderWidth": "1px",
                "borderStyle": "dashed",
                "borderRadius": "5px",
                "textAlign": "center",
                "margin": "10px",
            },
            # Allow multiple files to be uploaded
            multiple=True,
        ),
        html.Div(id="output-data-upload"),]
)


@callback(
    Output("Mygraph", "figure"),
    [Input("upload-data", "contents"), Input("upload-data",
"filename")],
)
def update_city_selected(input_value):
    return f'You selected: {input_value}'
def update_graph(contents, filename):
    x = []
    y = []
    if contents:
        contents = contents[0]
        filename = filename[0]
```

```
        df = parse_data(contents, filename)
        df = df.set_index(df.columns[0])
        x=df['PREVAILING_WAGE']
        y=df['VISA_CLASS']
    fig = go.Figure(
        data=[
            go.Scatter(
                x=x,
                y=y,
                mode='lines+markers')
            ],
        layout=go.Layout(
            plot_bgcolor=colors["graphBackground"],
            paper_bgcolor=colors["graphBackground"]
        ))
    return fig


def parse_data(contents, filename):
    content_type, content_string = contents.split(",")

    decoded = base64.b64decode(content_string)
    try:
        if "csv" in filename:
            # Assume that the user uploaded a CSV or TXT file
            df = pd.read_csv(io.StringIO(decoded.decode("utf-8")))
        elif "xls" in filename:
            # Assume that the user uploaded an excel file
            df = pd.read_excel(io.BytesIO(decoded))
        elif "txt" or "tsv" in filename:
            # Assume that the user upl, delimiter = r'\s+'oaded an excel
file
            df = pd.read_csv(io.StringIO(decoded.decode("utf-8")),
delimiter=r"\s+")
    except Exception as e:
        print(e)
        return html.Div(["There was an error processing this file."])

    return df
@callback(
    Output("output-data-upload", "children"),
    [Input("upload-data", "contents"), Input("upload-data",
"filename")],
)


def update_table(contents, filename):
    table = html.Div()

    if contents:
        contents = contents[0]
        filename = filename[0]
        df = parse_data(contents, filename)
```

```
        table = html.Div(
            [
                html.H5(filename),
                dash_table.DataTable(
                    data=df.to_dict("rows"),
                    columns=[{"name": i, "id": i} for i in df.columns],
                ),
                html.Hr(),
                html.Div("Raw Content"),
                html.Pre(
                    contents[0:200] + "...",
                    style={"whiteSpace": "pre-wrap", "wordBreak":
"break-all"},
                ),
            ]
        )

    return table
```

Page3:

```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import dash_bootstrap_components as dbc
import dash
from dash import html, dcc, callback, Input, Output
import plotly.figure_factory as ff

dash.register_page(__name__)

pd.set_option("display.max_rows", None, "display.max_columns", None)

data=pd.read_csv('h1b18.csv')
data.dropna(inplace=True)


# removing data which is not necessary
data['CASE_SUBMITTED']=pd.to_datetime(data['CASE_SUBMITTED'])
data['month_CASE_SUBMITTED'] =
pd.DatetimeIndex(data['CASE_SUBMITTED']).month


data['DECISION_DATE']=pd.to_datetime(data['DECISION_DATE'])
data['month_Decision'] = pd.DatetimeIndex(data['DECISION_DATE']).month

data['EMPLOYMENT_START_DATE']=pd.to_datetime(data['EMPLOYMENT_START_DATE
'])
data['EMPLOYMENT_END_DATE']=pd.to_datetime(data['EMPLOYMENT_END_DATE'])
```

```
data['month_EMPLOYMENT_START'] =
pd.DatetimeIndex(data['EMPLOYMENT_START_DATE']).month
data['month_EMPLOYMENT_End'] =
pd.DatetimeIndex(data['EMPLOYMENT_END_DATE']).month


data["PREVAILING_WAGE"]=data["PREVAILING_WAGE"].str.replace(",",
"").astype(float)
data["WAGE_RATE_OF_PAY_FROM"]=data["WAGE_RATE_OF_PAY_FROM"].str.replace(
",", "").astype(float)
data["WAGE_RATE_OF_PAY_TO"]=data["WAGE_RATE_OF_PAY_TO"].str.replace(",",
"").astype(float)


data=data.drop(['SOC_CODE',
'SOC_NAME','CASE_NUMBER','EMPLOYER_POSTAL_CODE','EMPLOYER_CITY','WAGE_RA
TE_OF_PAY_TO','WAGE_UNIT_OF_PAY'], axis=1)
data.dropna(inplace = True)
print(data.head())
data=data.head(50000)
print(data.nunique().sort_values(ascending=True))
print(data.shape)


df_heatmap = data.groupby('EMPLOYER_STATE')['CASE_STATUS'].count()
print(df_heatmap)
accept_arr = data['CASE_STATUS'].value_counts()
print(accept_arr)
accept_val = list(accept_arr)[0:4]
accept_label = list(accept_arr.index)[0:4]

graphs=["Histogram","Scatterplot","Pie Chart","Line chart","count plot",
"heat map","Bar plot",
        "Pair plot","Box plot","Regression plot"]
list=['None','CASE_STATUS','month_CASE_SUBMITTED','month_Decision','EMPL
OYMENT_END_DATE','month_EMPLOYMENT_End' ,'CASE_SUBMITTED',
'DECISION_DATE', 'VISA_CLASS',
       'EMPLOYMENT_START_DATE', 'EMPLOYER_NAME',
       'EMPLOYER_STATE', 'JOB_TITLE', 'FULL_TIME_POSITION',
'PREVAILING_WAGE',
       'PW_UNIT_OF_PAY', 'WAGE_RATE_OF_PAY_FROM']
list2=['CASE_STATUS','month_CASE_SUBMITTED','month_Decision','EMPLOYMENT
_END_DATE','month_EMPLOYMENT_End' ,'CASE_SUBMITTED', 'DECISION_DATE',
'VISA_CLASS',
       'EMPLOYMENT_START_DATE', 'EMPLOYER_NAME',
       'EMPLOYER_STATE', 'JOB_TITLE', 'FULL_TIME_POSITION',
'PREVAILING_WAGE',
       'PW_UNIT_OF_PAY', 'WAGE_RATE_OF_PAY_FROM']

layout = html.Div([
    dbc.Row(
            [
                html.Div(
```

```
                    [
                        html.H1('Pick a feature'),
                        html.Br(),
                        html.P("X axis"),
                        dcc.Checklist(options=list,
                                      id='col_na',value=['VISA_CLASS'],
inline= False,
                                       labelStyle={'display': 'block'},
                                       style={"height":200, "width":200,
"overflow":"auto"}),
                    ],
                )]),
    dbc.Row(
        [
            html.Div(
                [
                    html.H1('Pick a feature'),
                    html.Br(),
                    html.P("Y axis"),
                    dcc.RadioItems(options=list,
                                   id='col_na2', inline=False,
                                   labelStyle={'display': 'block'},
                                  style={"height":200, "width":500,
"overflow":"auto"}),
                ],
            )]),
    html.Br(),
    dbc.Row(
        [
            html.Div(
                [
                    html.H1('Pick a feature'),
                    html.Br(),
                    html.P("Hue (Optional)"),
                    dcc.RadioItems(options=list,
                                    id='col_na3', inline=False,
                                    labelStyle={'display': 'block'},
                                    style={"height": 200, "width": 500,
"overflow": "auto"}),
                ],
            )]),
    html.Br(),
    dbc.Row(
        [
            html.Div(
                [
                    html.H1('Pick a feature to know about'),
                    dcc.Dropdown(options=graphs,
                                 id='graphs_used', placeholder='Select
Symbol...'),
                ],
            )]),
    dbc.Row(
```

```
        [
            html.Div(
                        id='My_check_out'
            )]),


   html.Br(),
    dcc.Graph(id='fig'),
])
# @callback(
#     Output(component_id='My_check_out', component_property='value'),
#     [Input(component_id='col_na', component_property='value'),
#     Input(component_id='col_na2', component_property='value'),
#      Input(component_id='col_na3', component_property='value'),
#      Input(component_id='graphs_used', component_property='value')]
# )
# def update_title(a1,a2,a3,a4):
#     if a4=='Histogram':
#         return f"Histogram of {a1}"
#     elif a4=='Scatterplot':
#         return f"Scatterplot of {a1} vs {a2}"
#     elif a4=='Pie Chart':
#         return f"Pie Chart of distribution {a1}"
#     elif a4=='Line chart':
#         return f"Line chart of {a1} vs {a2}"
#     elif a4=='count plot':
#         return f"count plot of {a1} vs {a2}"
#     elif a4=='heat map':
#         return f" heat map of the numerical columns in dastaset"
#     elif a4=='Bar plot':
#         return f"Bar plot of {a1} vs {a2}"
#     elif a4=='Pair plot':
#         return f"pair plot of H1b18.csv dataset"
#     elif a4=='Box plot':
#         return f""


@callback(
    Output(component_id='fig', component_property='figure'),
    [Input(component_id='col_na', component_property='value'),
    Input(component_id='col_na2', component_property='value'),
     Input(component_id='col_na3', component_property='value'),
     Input(component_id='graphs_used', component_property='value')]
)
def update_city_selected(a1,a2,a3,a4):

    if a4=='Histogram':
        fig = px.histogram(data, x=a1 ,title=f"Histogram of {a1}")
    elif a4=='Scatterplot':
        fig = px.scatter(data, x=a1, y=a2, title=f"Scatterplot of {a1}
vs {a2}")
    elif a4=='Pie Chart':
        fig = px.pie(data, values=accept_val, names=accept_label,
```

```
hole=.3, title= f"Pie Chart of distribution {a1}")
    elif a4=='Line chart':
        fig = px.line(data,x=a1, y=a2, color=a3, title=f"Line chart of
{a1} vs {a2}")
    elif a4=='count plot':
        fig=px.histogram(data, x=a1, color=a2, title=f"count plot of
{a1} vs {a2}")
    elif a4=='heat map':
        fig = px.imshow(data[list2], title=f" heat map of the numerical
columns in dastaset")
    elif a4=='Bar plot':
        fig=px.histogram(data, x=a1, y=a2, color=a3, title=f"Bar plot of
{a1} vs {a2}")
    elif a4=='Pair plot':
        fig = px.scatter_matrix(data, title=f"pair plot of H1b18.csv
dataset")
    elif a4=='Box plot':
        fig = px.box(data, x=a1, color=a2, title=f"Box plot of {a1} vs
{a2}")
    elif a4 == 'Regression plot':
        fig = px.scatter(data, x=a1, y=a2,  color=a3, trendline="ols",
marginal_x='histogram', marginal_y='histogram',title=f"Regression plot
of {a1} vs {a2}")
    return fig
```

Page4:

```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import dash_bootstrap_components as dbc
import dash
from sklearn.decomposition import PCA
from dash import html, dcc, callback, Input, Output
import plotly.figure_factory as ff

dash.register_page(__name__)

pd.set_option("display.max_rows", None, "display.max_columns", None)

data=pd.read_csv('h1b18.csv')
data.dropna(inplace=True)


# removing data which is not necessary
data['CASE_SUBMITTED']=pd.to_datetime(data['CASE_SUBMITTED'])
data['month_CASE_SUBMITTED'] =
pd.DatetimeIndex(data['CASE_SUBMITTED']).month


data['DECISION_DATE']=pd.to_datetime(data['DECISION_DATE'])
```

```
data['month_Decision'] = pd.DatetimeIndex(data['DECISION_DATE']).month

data['EMPLOYMENT_START_DATE']=pd.to_datetime(data['EMPLOYMENT_START_DATE
'])
data['EMPLOYMENT_END_DATE']=pd.to_datetime(data['EMPLOYMENT_END_DATE'])

data['month_EMPLOYMENT_START'] =
pd.DatetimeIndex(data['EMPLOYMENT_START_DATE']).month
data['month_EMPLOYMENT_End'] =
pd.DatetimeIndex(data['EMPLOYMENT_END_DATE']).month


data["PREVAILING_WAGE"]=data["PREVAILING_WAGE"].str.replace(",",
"").astype(float)
data["WAGE_RATE_OF_PAY_FROM"]=data["WAGE_RATE_OF_PAY_FROM"].str.replace(
",", "").astype(float)
data["WAGE_RATE_OF_PAY_TO"]=data["WAGE_RATE_OF_PAY_TO"].str.replace(",",
"").astype(float)


data=data.drop(['SOC_CODE',
'SOC_NAME','CASE_NUMBER','EMPLOYER_POSTAL_CODE','EMPLOYER_CITY','WAGE_RA
TE_OF_PAY_TO','WAGE_UNIT_OF_PAY'], axis=1)
data.dropna(inplace = True)
print(data.head())
data=data.head(50000)
print(data.nunique().sort_values(ascending=True))
print(data.shape)


df_heatmap = data.groupby('EMPLOYER_STATE')['CASE_STATUS'].count()
print(df_heatmap)
accept_arr = data['CASE_STATUS'].value_counts()
print(accept_arr)
accept_val = list(accept_arr)[0:4]
accept_label = list(accept_arr.index)[0:4]

list=['None','CASE_STATUS','month_CASE_SUBMITTED','month_Decision','EMPL
OYMENT_END_DATE','month_EMPLOYMENT_End' ,'CASE_SUBMITTED',
'DECISION_DATE', 'VISA_CLASS',
       'EMPLOYMENT_START_DATE', 'EMPLOYMENT_END_DATE', 'EMPLOYER_NAME',
       'EMPLOYER_STATE', 'JOB_TITLE', 'FULL_TIME_POSITION',
'PREVAILING_WAGE',
       'PW_UNIT_OF_PAY', 'WAGE_RATE_OF_PAY_FROM']
data['FULL_TIME_POSITION'].replace(['Y', 'N'],[1, 0], inplace=True)
data['CASE_STATUS'].replace(['CERTIFIED', 'DENIED','CERTIFIED-
WITHDRAWN','WITHDRAWN'],[1, 2,3,4], inplace=True)
data['VISA_CLASS'].replace(['H-1B', 'E-3 Australian','H-1B1
Singapore','H-1B1 Chile'],[1, 2, 3, 4], inplace=True)

layout = html.Div([
    dbc.Row(
            [
```

```
                html.Br(),
                dcc.Tabs(id='Questions', value='q1', children=[
                    dcc.Tab(label='PCA', value='q1'),
                    dcc.Tab(label='Download', value='q2')])
    ]),
    html.Br(),
    html.Div(id='layout')
])

question1_layout=html.Div([
    dbc.Row(
            [
                html.H4("Visualization of PCA's explained variance"),
                dcc.Graph(id="graph"),
                html.P("Number of components:"),
                dcc.Slider(
                    id='slider',
                    min=2, max=5, value=3, step=1)
    ]),
    html.Br(),
    html.Div(id='layout')
])
@callback(
    Output(component_id='graph', component_property='figure'),
    [Input(component_id='slider', component_property='value')]
)
def run_and_plot(n_components):
    pca = PCA(n_components=n_components)
    X = data[['FULL_TIME_POSITION', 'CASE_STATUS', 'VISA_CLASS',
'WAGE_RATE_OF_PAY_FROM','PREVAILING_WAGE']]

    components = pca.fit_transform(X)

    var = pca.explained_variance_ratio_.sum() * 100

    labels = {str(i): f"PC {i+1}"
              for i in range(n_components)}

    fig = px.scatter_matrix(
        components,
        dimensions=range(n_components),
        labels=labels,
        title=f'Total Explained Variance: {var:.2f}%')
    return fig

question2_layout=html.Div([
    dbc.Row(
            [

        html.Button("Download CSV", id="btn_csv")
            ]),
        dcc.Download(id="download-dataframe-csv"),
    ]
```

```
)
@callback(
    Output(component_id='download-dataframe-csv',
component_property='data'),
    [Input(component_id='btn_csv', component_property='n_clicks')],
    prevent_initial_call=True
)

def func(n_clicks):
    return dcc.send_data_frame(data.to_csv, "mydata.csv")


@callback(
    Output(component_id='layout', component_property='children'),
    [Input(component_id='Questions', component_property='value')]
)
def update_city_selected(a1):
    if a1 == 'q1':
        return question1_layout
    elif a1 == 'q2':
        return question2_layout
```

References:

https://plotly.com

https://plotly.com/dash/

https://www.kaggle.com/datasets/jmpark746/h1b-visas/code?select=h1b18.csv

https://www.tableau.com/learn/articles/data-visualization