Predicting unwillingness to receive COVID-19 vaccines and examining its contributing factors

Abstract:

Keywords:

1. Project Scope

1.1 Introduction

The development of vaccines from Pfizer, AstraZeneca, Moderna, Novavex, Johnson and Johnson, etc. gives us hope to end the pandemic caused by COVID-19. However, we hear many daily news reports that some health workers are refusing vaccines (CBSNews, Cerulo) or anti-vaccine activists' protests are happening. (NYTimes, Fernandez) According to the House Pulse Survey done by CDC on January 6th, 24% of the population were hesitant to receive the vaccine by responding "probably not" (14%) or "definitely not" (9.5%) on the question asking their willingness to get the vaccine. Of those who were hesitant to receive the vaccine, 53% responded "I am concerned about possible side effects of a COVID vaccine", 19% responded "I don't know if a COVID vaccine works", and 9.85% "I don't believe I need a COVID vaccine.

1.2 Aim

We aim to find out the contributing factors of people's unwillingness to get the COVID-19 vaccines and to examine the possibility of predicting people's unlikelihood of getting COVID-19 vaccines based on those factors.

1.3 Purpose

The purpose of the study is to explain why certain populations are not willing to get the COVID-19 vaccines. According to Rasmussen, population-level herd immunity is essential to control SARS-CoV-2 in the long run and getting vaccinated is the key to reach the herd immunity threshold of SARS-CoV-2. (Rasmussen, 2020) Therefore, we would like to examine the factors below (House Pulse Survey, CDC) and predict people's unwillingness to get the vaccine in relation to these factors so that organizations can identify where people's uncertainties or concerns are originating from and come up with ideas of how to approach those behaviors.

2. Methodology

2.1 Steps of the Project

- Type of study: Quantitative Research (Descriptive Statistics and Inferential Statistics)

- Data collection: CDC House Pulse Survey in Phase 3 (Jan 3- 18) who took the survey (https://www.census.gov/programs-surveys/household-pulse-survey/datasets.html)

- Data storage: CSV files

- Data extraction, description: Finding statistical significance of correlation factors, normality test

- Data analysis: Model development, performance analysis using Python, data visualization using Python Seaborn, Matplotlib, Tableau

2.2 Team Members and Responsibilities

Team Member	Responsibilities
Cassandra Jones	· Data extraction using SQL Data cleaning using SQL · Normality test using Python Descriptive statistics using Tableau
Chinmayi Sailaja Nimmaraju	· Binary classification model development · Clustering analysis · Data visualization of clustering
Swetha Sabbieni	· Evaluating binary classification (Logistic Regression) using Python · Visualization of linear regression model using Python Matplotlib
Monica Yamsani	· Data visualization of normality test results using Python Seaborn and Matplotlib (seaborn pairplots, histogram, heatmap) · Evaluating binary classification (Support Vector Machine) using Python · Visualization of SVM using Python

Santosh Kumar Sirasapalli	· Evaluating binary classification (Naives Bayes Classfier) using Python · Visualization of Naives Bayes classifier using Python · Final project report
Jenna Kim	· Project proposal · Risk prediction model development · Evaluating risks by risk score using Python

2.3 Project Challenges

3. Data Collection

The dataset was taken from the U.S. Census Bureau House Pulse Survey

(https://www.census.gov/programs-surveys/household-pulse-survey/datasets.html). This dataset provides information regarding a 20 – minute online survey which measure household experiences during COVID-19.

There are three sections in this dataset which are described as phases. Each weekly release include a Public Use Data File (PUF), a replicate weight data file, and a data dictionary. The sections are :

Phase 3 PUF Releases

Household Pulse Survey PUF: March 17 – March 29

HPS Week 27 PUF SAS

HPS Week 27 PUF CSV

Household Pulse Survey PUF: March 3 – March 15

HPS Week 26 PUF SAS

HPS Week 26 PUF CSV

Household Pulse Survey PUF: February 17 – March 1

HPS Week 25 PUF SAS

HPS Week 25 PUF CSV

Household Pulse Survey PUF: February 3 – February 15

HPS Week 24 PUF SAS

HPS Week 24 PUF CSV

Household Pulse Survey PUF: January 20 – February 1

HPS Week 23 PUF SAS

HPS Week 23 PUF CSV

Household Pulse Survey PUF: January 6 – January 18

HPS Week 22 PUF SAS

HPS Week 22 PUF CSV

Household Pulse Survey PUF: December 9 – December 21

HPS Week 21 PUF SAS

HPS Week 21 PUF CSV

Household Pulse Survey PUF: November 25 – December 7

HPS Week 20 PUF SAS

HPS Week 20 PUF CSV

Household Pulse Survey PUF: November 11 – November 23

HPS Week 19 PUF SAS

HPS Week 19 PUF CSV

Household Pulse Survey PUF: October 28 – November 9

HPS Week 18 PUF SAS

HPS Week 18 PUF CSV

Phase 2 PUF Releases

Household Pulse Survey PUF: October 14 – October 26

HPS Week 17 PUF SAS

HPS Week 17 PUF CSV

Household Pulse Survey PUF: September 30 – October 12

HPS Week 16 PUF SAS

HPS Week 16 PUF CSV

Household Pulse Survey PUF: September 16 – September 28

HPS Week 15 PUF SAS

HPS Week 15 PUF CSV

Household Pulse Survey PUF: September 2 – September 14

HPS Week 14 PUF SAS

HPS Week 14 PUF CSV

Household Pulse Survey PUF: August 19 – August 31

HPS Week 13 PUF SAS

HPS Week 13 PUF CSV

Phase 1 Weekly PUF Releases

Household Pulse Survey PUF: July 16 – July 21

HPS Week 12 PUF SAS

HPS Week 12 PUF Household Weights SAS

HPS Week 12 PUF CSV

HPS Week 12 PUF Household Weights CSV

Household Pulse Survey PUF: July 9 – July 14

HPS Week 11 PUF SAS

HPS Week 11 PUF Household Weights SAS

HPS Week 11 PUF CSV

HPS Week 11 PUF Household Weights CSV

Household Pulse Survey PUF: July 2 – July 7

HPS Week 10 PUF SAS

HPS Week 10 PUF Household Weights SAS

HPS Week 10 PUF CSV

HPS Week 10 PUF Household Weights CSV

Household Pulse Survey PUF: June 25 – June 30

HPS Week 9 PUF SAS

HPS Week 9 PUF Household Weights SAS

HPS Week 9 PUF CSV

HPS Week 9 PUF Household Weights CSV

Household Pulse Survey PUF: June 18 – June 23

HPS Week 8 PUF SAS

HPS Week 8 PUF Household Weights SAS

HPS Week 8 PUF CSV

HPS Week 8 PUF Household Weights CSV

Household Pulse Survey PUF: June 11 – June 16

HPS Week 7 PUF SAS

HPS Week 7 PUF Household Weights SAS

HPS Week 7 PUF CSV

HPS Week 7 PUF Household Weights CSV

Household Pulse Survey PUF: June 4 – June 9

HPS Week 6 PUF SAS

HPS Week 6 PUF Household Weights SAS

HPS Week 6 PUF CSV

HPS Week 6 PUF Household Weights CSV

Household Pulse Survey PUF: May 28 – June 2

HPS Week 5 PUF SAS

HPS Week 5 PUF Household Weights SAS

HPS Week 5 PUF CSV

HPS Week 5 PUF Household Weights CSV

Household Pulse Survey PUF: May 21 – May 26

HPS Week 4 PUF SAS

HPS Week 4 PUF Household Weights SAS

HPS Week 4 PUF CSV

HPS Week 4 PUF Household Weights CSV

Household Pulse Survey PUF: May 14 – May 19

HPS Week 3 PUF SAS

HPS Week 3 PUF Household Weights SAS

HPS Week 3 PUF CSV

HPS Week 3 PUF Household Weights CSV

Household Pulse Survey PUF: May 7 – May 12

HPS Week 2 PUF SAS

HPS Week 2 PUF Household Weights SAS

HPS Week 2 PUF CSV

HPS Week 2 PUF Household Weights CSV

Household Pulse Survey PUF: April 23 – May 5

HPS Week 1 PUF SAS

HPS Week 1 PUF Household Weights SAS

HPS Week 1 PUF CSV

HPS Week 12 PUF Household Weights CSV

4. Data Extraction and Storage

4.1 Data Extraction

We collected a total of 127,261 participant responses from Phase 3 : Week 22 (Jan 6 – 18), Week 23 (Jan 20 – Feb 1) to work on this project and selected the variables accordingly.

The 18 variables we selected are :

1. Age

2. Gender

3. Hispanic origin

4. Race

5. Education

6. Marital status

7. Recent job loss

8. Expected household job loss,

9. Employment status,

10. Kind of work

11. Difficulty with expenses

12. Health insurance

13. Prescription mental health

14. Mental health services

15. Mental health not get

16. Home owned

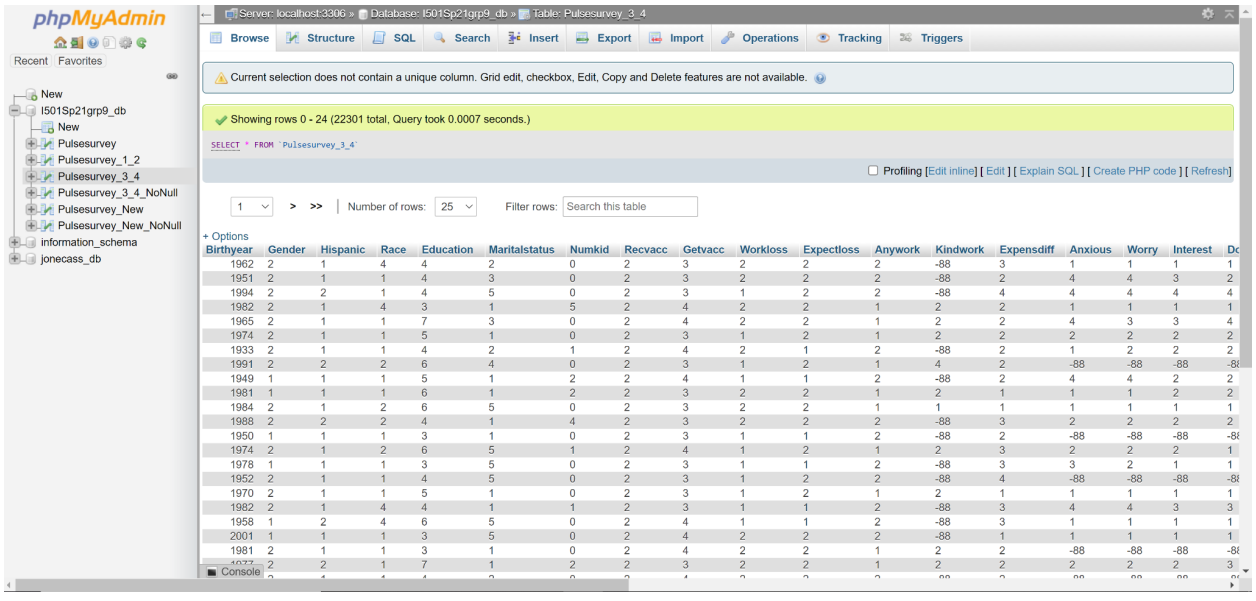
17. Pandemic children education status

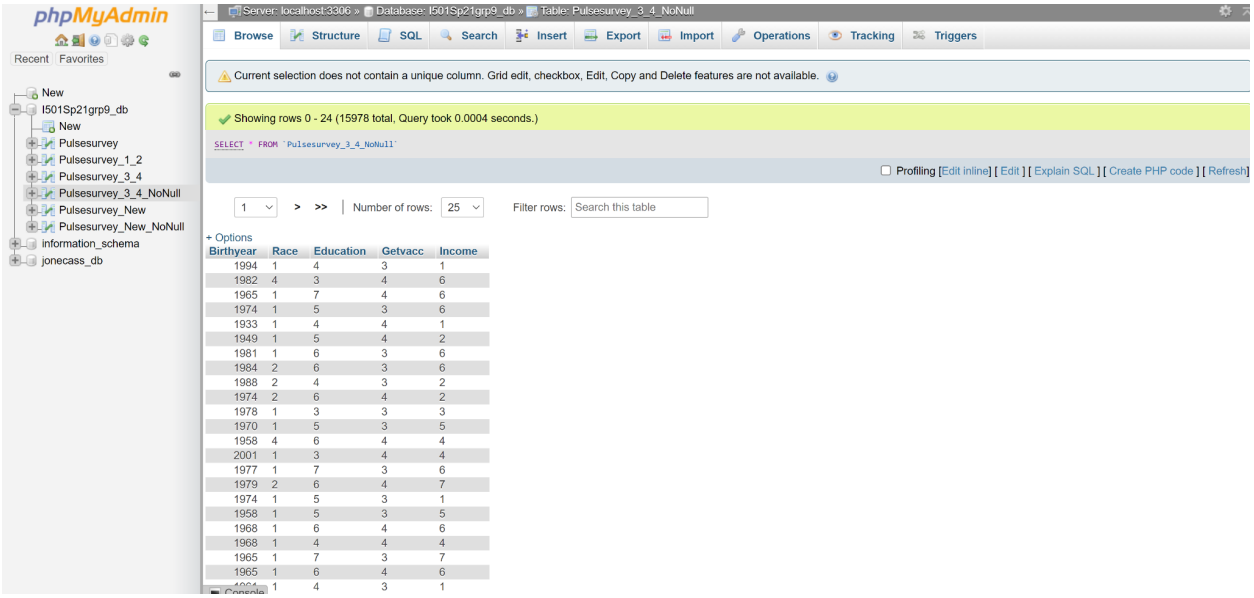
18.Income

4.2 Data Cleaning

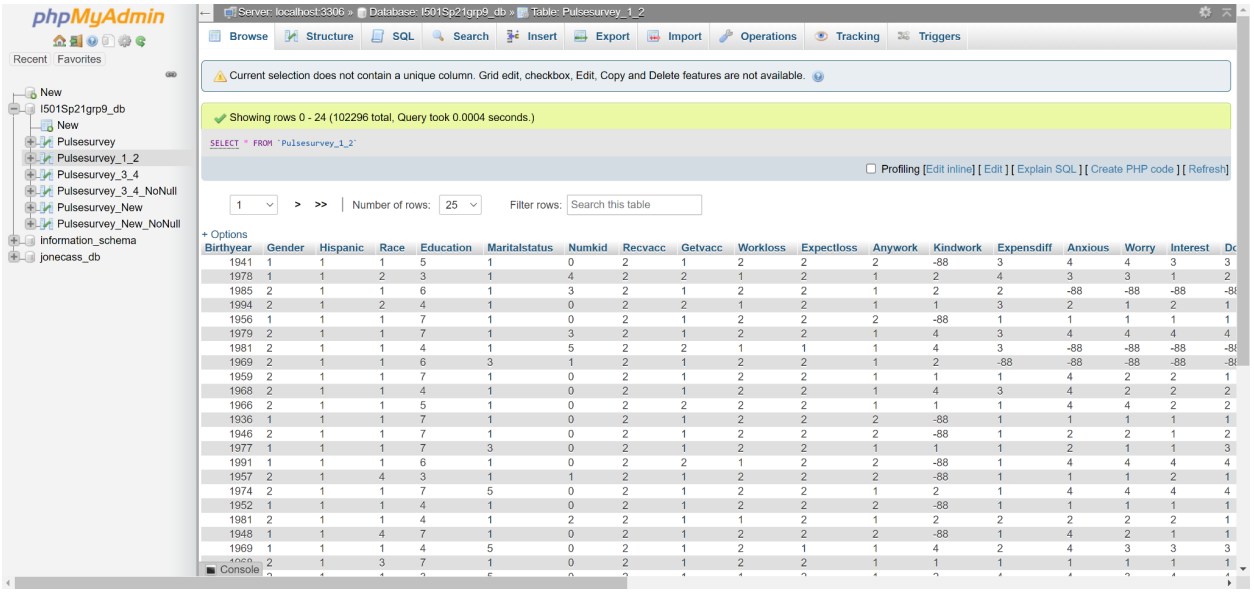
We cleaned the data by taking the csv files for weeks 22 and 23 of a nationwide COVID-19 survey into Microsoft Excel and removed all columns that we had determined as a group we did not feel were predictors for someone to have vaccine hesitancy. The remaining columns were as follows: Birthyear, Gender, Hispanic, Race, Education, Maritalstatus, Numkid, Recvacc, Doses, Getvacc, Workloss, Expectloss, Anywork, Kindwork, Expensdiff, Anxious, Worry, Interest, Down, Hlthins1, Hlthins2, Hlthins3, Hlthins4, Hlthins5, Hlthins6, Hlthins7, Hlthins8, Prescript, Mh_svcs, Mh_notget, Tenure, Enroll1, Enroll2, Enroll3, Teach1, Teach2, Teach3, Teach4, Teach5, and Income. Exclusion criteria was where Income response was either -88 (Missing) or -99 (Chose not to answer). After importing the initial data into a table in SQL named Pulsesurvey, we continued to clean the data in SQL using SQL queries that can be found in the appendix.

We then had a need for a subset of our data which only had participants who responded either 3 (probably not) or 4 (definitely not) to the question of whether or not they would get the vaccine (Getvacc). To do this we created a new table called Pulsesurvey_3_4, inserted all the data from the first Pulsesurvey table, and deleted rows from Pulsesurvey_3_4 where responses to Getvacc were either 1 or 2. We exported the new csv file from phpmyadmin as Pulsesurvey_3_4.csv. We opened the csv file in Excel and re-added column names, then imported Pulsesurvey_3_4.csv into Jupyter Notebook.



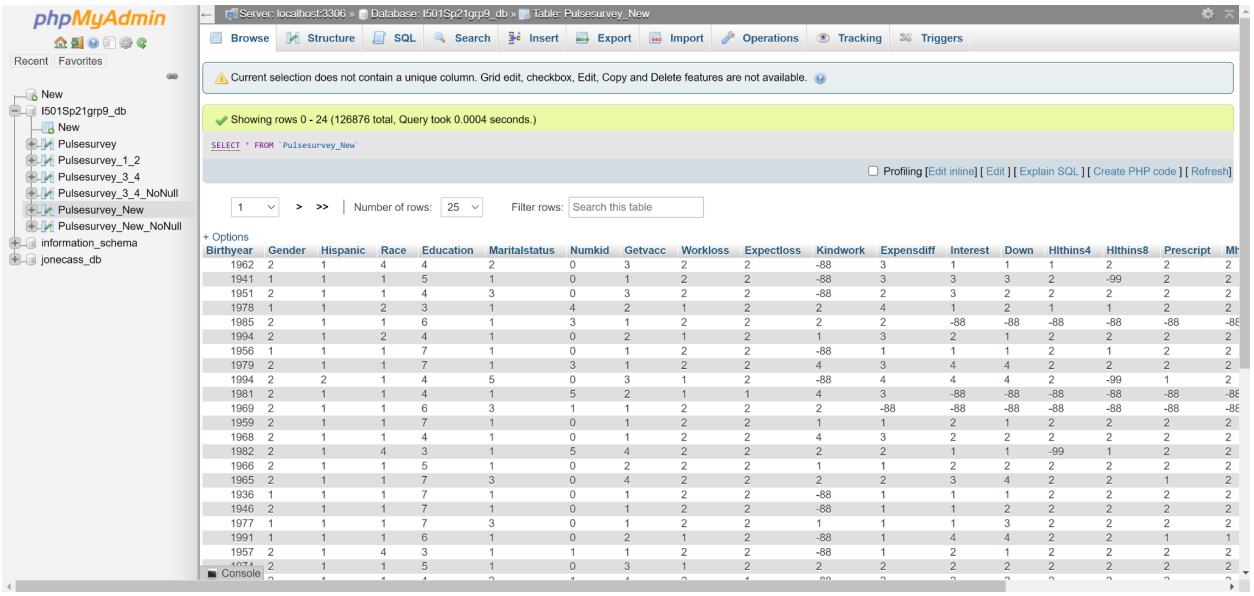


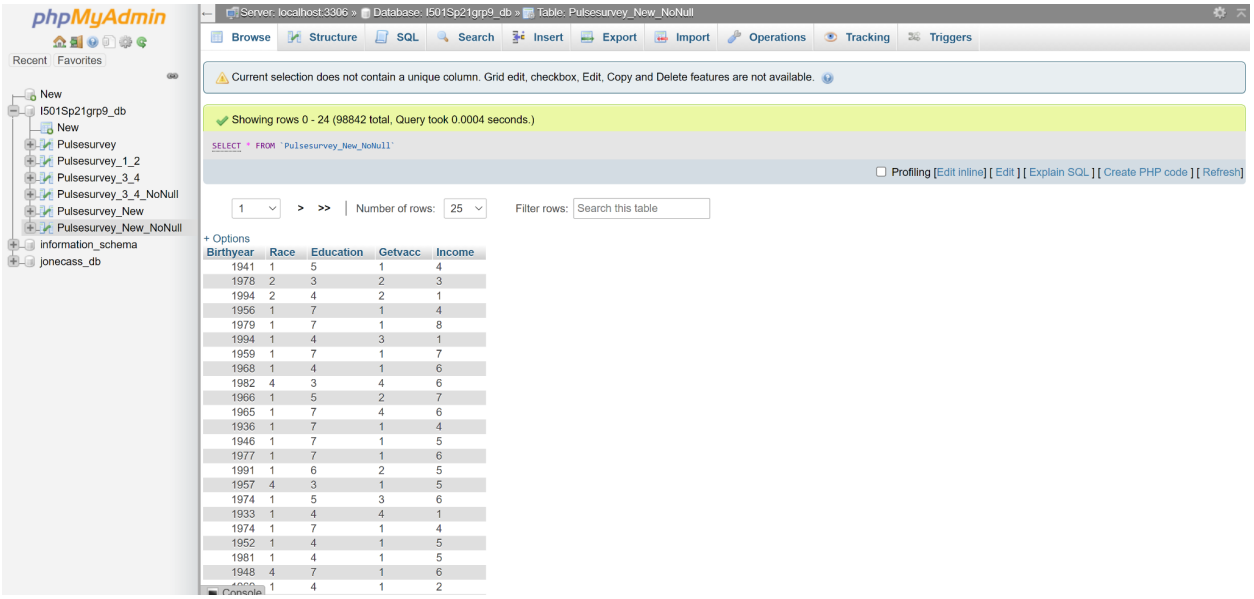
We also created a separate table with just answers 1 and 2 (will get vaccine) to compare against the table with just answers 3 and 4 (will not get vaccine), exported that new table as a csv file named Pulsesurvey_1_2.csv. We took that file into excel, added column names, and imported it into Jupyter Notebook.



The team members were finding that the ROC curve was off so we decided on dropping variables (columns) that had been found to not be predictive factors based on Jenna's tests. We created a separate table to test data with fewer variables so as to not destroy the original dataset and named it Pulsesurvey_New. The columns to delete based on testing were: Recvacc, Anywork, Hlthins3, Hlthins5, Hlthins6, Hlthins7, Enroll1, Enroll 2, Enroll3, Teach1, Teach2, Teach3, and Teach5. We also exported Pulsesurvey_New, added the column names, and imported it into Jupyter Notebook. We then had a

meeting where it was determined that we should remove worry and anxious as variables from Pulsesurvey_New as well. After making those changes in SQL, we repeated the process of exporting and importing the csv file.



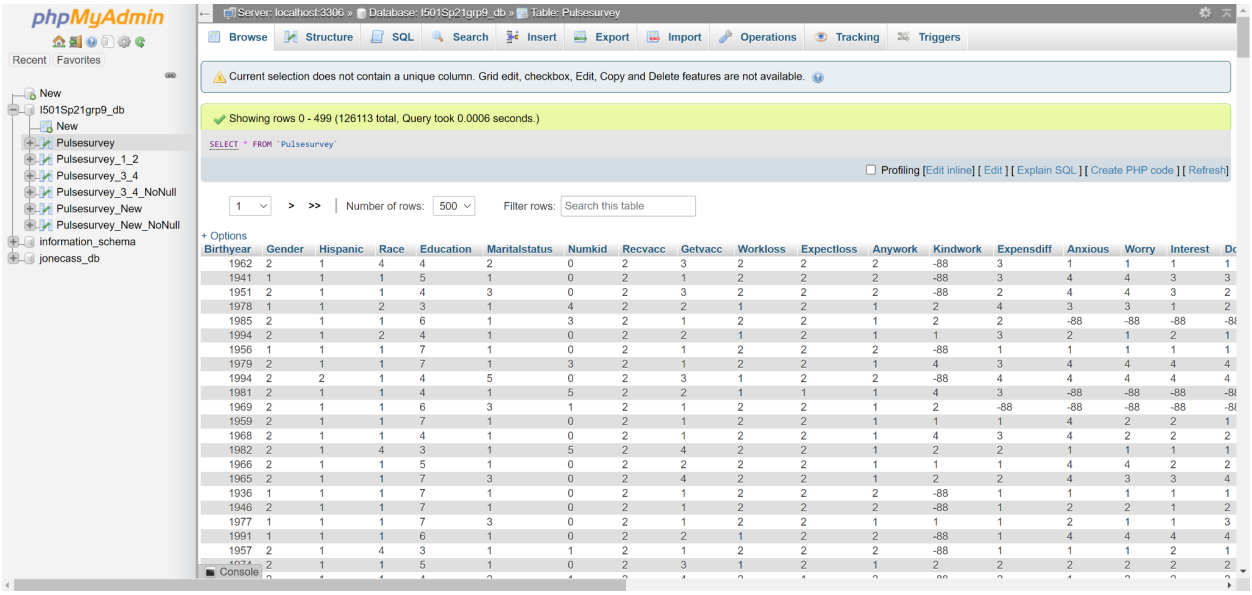


We then decided to create box plots and stacked bar charts because our data is mostly discrete and we want to make sure it's being represented properly. In order to make sure these looked alright, we chose to delete the -88 and -99 values from the columns that we're concerned about. We also didn't want to have a cluttered mess with important things being deleted from our important columns by deleting -88 and -99 from the rest of the columns, so we made a new table in our database and deleted all the columns we weren't concerned with, this new table was called Pulsesurvey_3_4_NoNull and included only the Birthyear, Race, Education, Getvacc, and Income columns. A version of Pulsesurvey_New was also created without -88 and -99 values in the Income column for any existing code that was created with the

intention of using Pulsesurvey_New, this new table also included only the Birthyear, Race, Education, Getvacc, and Income columns.

4.3 Data Import

In order to import the data into SQL, we first created a table in SQL with the names of the columns from the csv files. There were over 80,000 rows in our data. We then used SQL's import function. We had to zip the file because the unzipped file was too large to import. 66953 rows imported before the import timed out. We deleted those rows from the csv and imported the rest of the rows. We imported weeks 23 and 22 this way. We exported the table as a csv file from phpmyadmin as Pulsesurvey.csv. We opened the csv file in Excel and re-added column names. We imported Pulsesurvey.csv into Jupyter Notebook.



Once the data was imported into phpmyadmin, we connected our code to the SQL database using the following code:

```
!pip install mysql-connector-python-rf
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: mysql-connector-python-rf in ./.local/lib/python3.8/site-packages (2.2.2)
```

```
myvars = {}
with open("jonecass-mysql-password") as myfile:
    for line in myfile:
        name, var = line.partition(":")[::2]
        myvars[name.strip()] = var.strip()

myvars.keys()
```

```
]: dict_keys(['DB username', 'DB databasename', 'DB password'])
```

```
import pymysql
conn = pymysql.connect(host="localhost", user=myvars['DB username'], passwd=myvars['DB password'], db='I501Sp21grp9_db')
cursor = conn.cursor()
cursor.execute('select * from Pulsesurvey')
rows = cursor.fetchall()
print(len(rows))
print(type(rows))
data = list(rows)
```

```
127261
<class 'tuple'>
```

```
cursor.execute('select * FROM Pulsesurvey WHERE Getvacc >= 3')
rows = cursor.fetchall()
print(len(rows))
```

```
22301
```

```
import pandas as pd
import scipy.stats
import numpy as np

data_df = pd.DataFrame(data)
print(data_df)
```

```
          0  1  2  3  4  5  6  7  8  9   ...   29   30   31   32   33   34  \
0       1962  2  1  4  4  2  0  2  3  2  ...    1  -88  -88  -88  -88  -88
1       1941  1  1  1  5  1  0  2  1  2  ...    2  -88  -88  -88  -88  -88
2       1951  2  1  1  4  3  0  2  3  2  ...    1  -88  -88  -88  -88  -88
3       1978  1  1  2  3  1  4  2  2  1  ...    3    1  -99  -99    1  -99
4       1985  2  1  1  6  1  3  2  1  2  ...  -88  -88  -88  -88  -88  -88
...      ... .. .. .. .. .. .. .. .. .. ...  ...  ...  ...  ...  ...  ...
127256  1962  1  1  1  7  1  0  2  1  1  ...    2  -88  -88  -88  -88  -88
127257  1969  1  1  1  4  5  0  2  1  2  ...    1  -88  -88  -88  -88  -88
127258  1950  1  1  1  4  3  0  2  1  1  ...    1  -88  -88  -88  -88  -88
127259  1982  1  1  1  4  1  3  2  2  2  ...    2    1  -99  -99  -99  -99
127260  1959  2  1  1  5  3  0  2  2  2  ...    1  -88  -88  -88  -88  -88

         35   36   37   38
0       -88  -88  -88  -88
1       -88  -88  -88    4
2       -88  -88  -88  -99
3       -99  -99  -99    3
4       -88  -88  -88  -88
...     ...  ...  ...  ...
127256  -88  -88  -88    7
127257  -88  -88  -88    4
127258  -88  -88  -88    3
127259  -99  -99    1    8
127260  -88  -88  -88    4

[127261 rows x 39 columns]
```

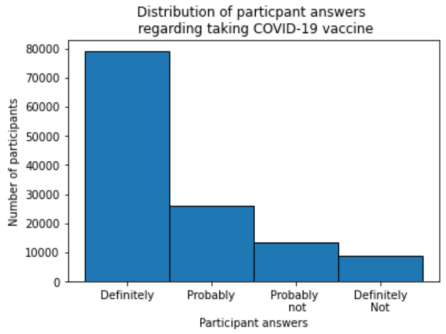
5. Data Analysis

5.1 Descriptive Statistics

5.1.1 Normality Test

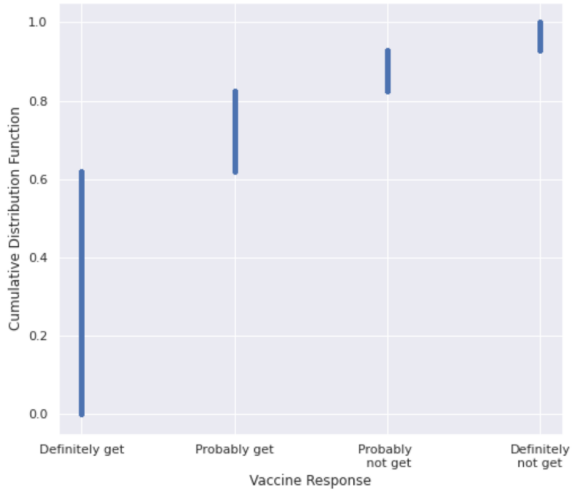
```
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv('Pulsesurvey.csv')
getvacc = data['Getvacc']
bins = [1,2,3,4,5]

plt.hist(getvacc, bins=bins, edgecolor='black')
plt.xticks(ticks = [1.5,2.5,3.5,4.5],labels=['Definitely', 'Probably', 'Probably \n not', 'Definitely\nNot'])
plt.ylabel('Number of participants')
plt.xlabel('Participant answers')
plt.title('Distribution of particpant answers \n regarding taking COVID-19 vaccine')
```

```
Text(0.5, 1.0, 'Distribution of particpant answers \n regarding taking COVID-19 vaccine')
```



This is a histogram showing the distribution of "Getvacc" responses for the entirety of our data including 1 (Definitely), 2 (Probably), 3 (Probably not), and 4 (Definitely not). This histogram shows that the vast majority of the responses were from participants that would definitely get the vaccine. The numbers of responses from participants dwindled as uncertainty regarding the vaccine grew. Meaning that the majority of respondents would definitely get the vaccine and the vast minority of responses would not get the vaccine.

```
Text(0, 0.5, 'Cumulative Distribution Function')
```



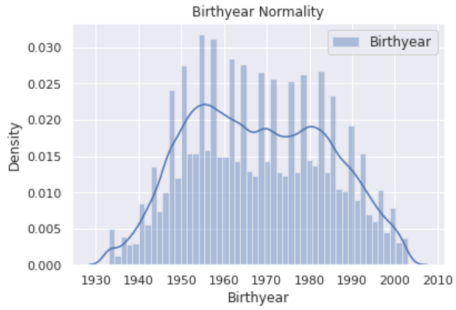
This graph is similar to the histogram, it is the cumulative distribution function of "Getvacc" responses for the entirety of our data including 1 (Definitely), 2 (Probably), 3 (Probably not), and 4 (Definitely not).

Appears as a series of lines because the data is discrete. This cumulative distribution function is a reframing of the same information from the histogram, showing that most respondents said that they would definitely get the vaccine, the second largest group of respondents said that they would probably get the vaccine, the third largest group of respondents said that they would probably not get the vaccine, and the smallest amount of respondents said that they would definitely not get the vaccine.

These two charts show the distribution of vaccine attitudes across all of the data from the survey. The distribution shows that our vaccine response data is not normally distributed, it has a right skew.

```
import seaborn as sns

sns.distplot(data['Birthyear'], label='Birthyear')

plt.legend(prop={'size': 12})
plt.title('Birthyear Normality')
plt.xlabel('Birthyear')
plt.ylabel('Density')
```

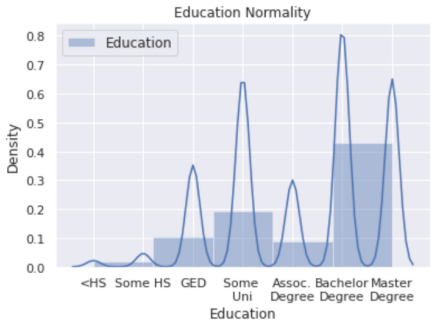
```
Text(0, 0.5, 'Density')
```



This chart shows the distribution of Birthyear for the entirety of our data. You can see from the chart that the majority of responses are from birth years 1950 - 1980. However, instead of a nice bell curve, the curve becomes concave at the top, meaning that this data is not normally distributed.

```
sns.distplot(data['Education'], label='Education', bins = 5)

plt.legend(prop={'size': 12})
plt.title('Education Normality')
plt.xlabel('Education')
plt.xticks(ticks = [1,2,3,4,5,6,7], labels=['<HS', 'Some HS', 'GED', 'Some \nUni', 'Assoc.\nDegree', 'Bachelor\nDegree', 'Mas
plt.ylabel('Density')
```

```
Text(0, 0.5, 'Density')
```

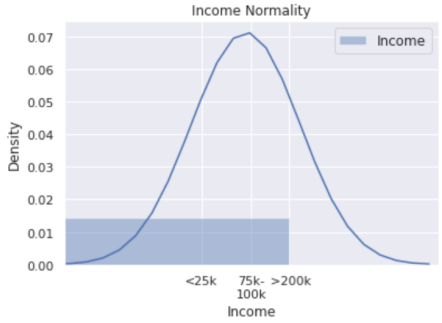


This chart shows the distribution of Education where Education responses were 1 (less than high school), 2 (some high school), 3 (high school diploma or GED), 4 (some college), 5 (associate degree), 6 (bachelor

degree), and 7 (Master Degree). Since our data is discrete, there are several small humps, which show the delineation between the values which represent the survey responses. If you consider the histogram beneath the line, you can see that the majority of the participant responses came from people with either masters degrees or bachelors degrees. The data is not normally distributed, it has a left skew.

```
sns.distplot(data['Income'], label='Income', bins = 2)

plt.legend(prop={'size': 12})
plt.title('Income Normality')
plt.xlabel('Income')
plt.xlim(-10,20)
plt.xticks(ticks=[1,5,8], labels=['<25k','75k-\n100k',' >200k'])
plt.ylabel('Density')
```

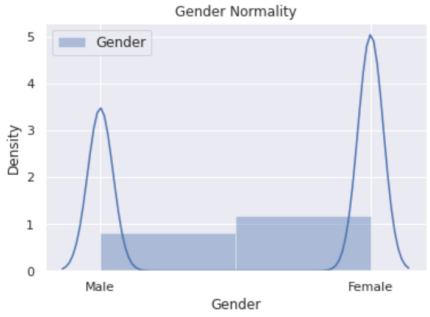
```
Text(0, 0.5, 'Density')
```



This chart shows the distribution of Income where Income responses were -88 (Missing), -99 (Chose not to answer), 1 (<$25k), 2 ($25 - 35k), 3 ($35 - 50k), 4 ($50 - 75k), 5 ($75 - 100k), 6 ($100 - 150k), 7 ($150 - 200k), and 8 (>$200k). You can see that this does show a typical bell shaped curve where the median response of 75k - 100k has the highest number of responses and the low and high ends have less responses. This appears to be normally distributed across the entirety of our data.

```
sns.distplot(data['Gender'], label='Gender', bins = 2)

plt.legend(prop={'size': 12})
plt.title('Gender Normality')
plt.xlabel('Gender')
plt.xticks(ticks = [1,2], labels = ['Male', 'Female'])
plt.ylabel('Density')
```

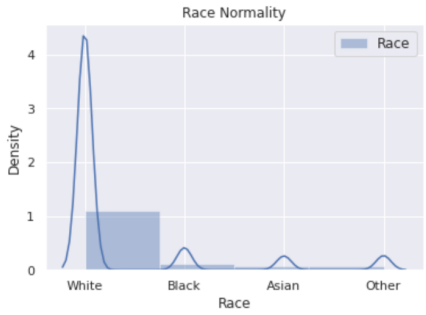
```
Text(0, 0.5, 'Density')
```



This chart shows the distribution of Gender where responses were 1 (Male) or 2 (Female). You can see from the chart that there were more responses from females than there were from males.

```
sns.distplot(data['Race'], label='Race', bins = 4)

plt.legend(prop={'size': 12})
plt.title('Race Normality')
plt.xlabel('Race')
plt.xticks(ticks=[1,2,3,4], labels=['White', 'Black', 'Asian', 'Other'])
plt.ylabel('Density')
```

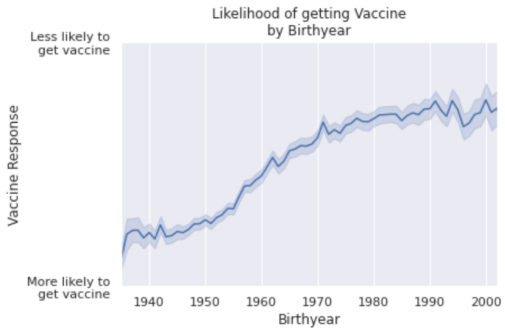
```
Text(0, 0.5, 'Density')
```



This chart shows the distribution of Race where responses were 1 (White), 2 (Black), 3 (Asian), 4 (Other). The vast majority of responses were "White" with the least amount of responses being "Asian". This data is not normally distributed and has a right skew.

```
a = sns.lineplot(data=pd.read_csv('Pulsesurvey.csv'), x='Birthyear', y='Getvacc')
a.set(ylim=[1,2.2])
a.set(yticks=[1,2.2])
a.set(xlim=[1935,2002])
a.set(yticklabels=['More likely to\nget vaccine', 'Less likely to\nget vaccine'])
a.set(ylabel='Vaccine Response')
a.set(title='Likelihood of getting Vaccine\nby Birthyear')
print(a)
```

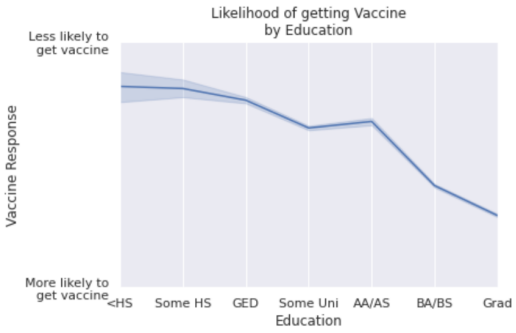
```
AxesSubplot(0.125,0.125;0.775x0.755)
```



This is a line graph showing the likelihood of participants with birth years from 1935 - 2002 to answer either more likely to get the vaccine or less likely to get the vaccine. The general trend shown here is that younger participants answered that they were less likely to get the vaccine than older participants.

```
b = sns.lineplot(data=pd.read_csv('Pulsesurvey.csv'), x='Education', y='Getvacc')
b.set(ylim=[1,2.2])
b.set(yticks=[1,2.2])
b.set(xticks=[1,2,3,4,5,6,7])
b.set(xlim=[1,7])
b.set(xticklabels=['<HS', 'Some HS', 'GED', 'Some Uni', 'AA/AS', 'BA/BS', 'Grad'])
b.set(yticklabels=['More likely to\nget vaccine', 'Less likely to\nget vaccine'])
b.set(ylabel='Vaccine Response')
b.set(title='Likelihood of getting Vaccine\nby Education')
print(b)
```

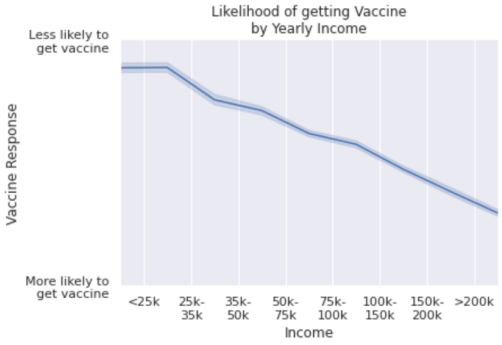
```
AxesSubplot(0.125,0.125;0.775x0.755)
```



This is a line graph showing the likelihood of participants to answer either more likely to get the vaccine or less likely to get the vaccine by their education (<HS - Masters). The general trend shown here is that participants with higher education experience answered that they were more likely to get the vaccine than those with less education.

```
c = sns.lineplot(data=pd.read_csv('Pulsesurvey.csv'), x='Income', y='Getvacc')
c.set(ylim=[1,2])
c.set(yticks=[1,2])
c.set(xticks=[0.5,1.5,2.5,3.5,4.5,5.5,6.5,7.5])
c.set(xlim=[0,8])
c.set(xticklabels=['<25k', '25k-\n35k', '35k-\n50k','50k-\n75k', '75k-\n100k', '100k-\n150k', '150k-\n200k', '>200k'])
c.set(yticklabels=['More likely to\nget vaccine', 'Less likely to\nget vaccine'])
c.set(ylabel='Vaccine Response')
c.set(title='Likelihood of getting Vaccine\nby Yearly Income')
print(c)
```

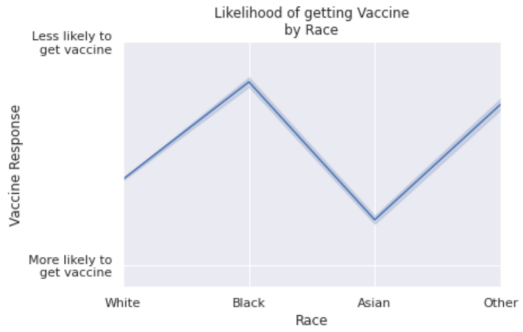
```
AxesSubplot(0.125,0.125;0.775x0.755)
```



This is a line graph showing the likelihood of participants to answer either more likely to get the vaccine or less likely to get the vaccine by their income (<$25k - >$200k). The general trend shown here is that participants with higher incomes answered that they were more likely to get the vaccine than participants with lower income.

```
d = sns.lineplot(data=pd.read_csv('Pulsesurvey.csv'), x='Race', y='Getvacc')
d.set(ylim=[1.1,2.2])
d.set(yticks=[1.2,2.2])
d.set(xticks=[1,2,3,4])
d.set(xlim=[1,4])
d.set(xticklabels=['White', 'Black', 'Asian', 'Other'])
d.set(yticklabels=['More likely to\nget vaccine', 'Less likely to\nget vaccine'])
d.set(ylabel='Vaccine Response')
d.set(title='Likelihood of getting Vaccine\nby Race')
print(d)
```

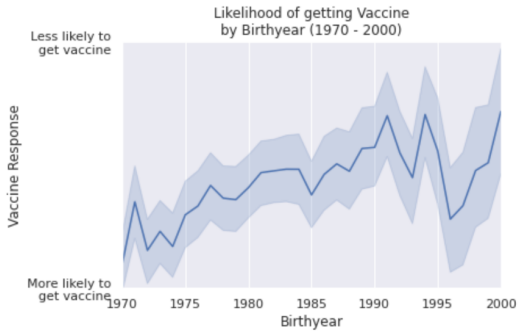
```
AxesSubplot(0.125,0.125;0.775x0.755)
```



This is a line graph showing the likelihood of participants to answer either more likely to get the vaccine or less likely to get the vaccine by their race (White, Black, Asian, or Other). The nature of the data, being that it is discrete, makes a line graph less suitable for data visualization than it would be for that of continuous data. You can still see, however, that participants that responded "Black" on the survey were least likely to get the vaccine and participants that responded "Asian" on the survey were most likely to get the vaccine. It should be noted that the distribution of responses shows that there was very limited data for races that were not White in our data set.

```
a2 = sns.lineplot(data=pd.read_csv('Pulsesurvey.csv'), x='Birthyear', y='Getvacc')
a2.set(xlim=(1970,2000))
a2.set(ylim=[1.7,2])
a2.set(yticks=[1.7,2])
a2.set(yticklabels=['More likely to\nget vaccine', 'Less likely to\nget vaccine'])
a2.set(ylabel='Vaccine Response')
a2.set(title='Likelihood of getting Vaccine\nby Birthyear (1970 - 2000)')
print(a2)
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



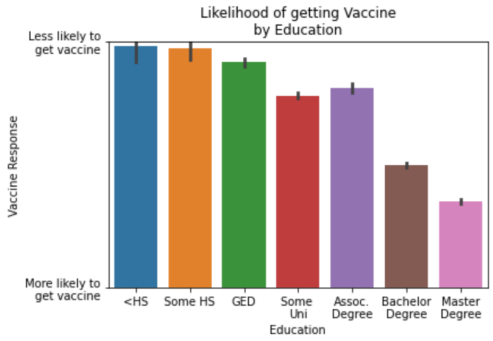
This is a line graph showing the likelihood of participants with birth years from 1970 - 2000 to answer either more likely to get the vaccine or less likely to get the vaccine. The purpose of this graph is to zoom in on a portion of the data to get a better picture of the nuance within the data. You can see how it continues to follow the larger trend of younger participants being less likely to get the vaccine, however, it

also shows areas where certain ages were much more likely than the ages surrounding them to get the vaccine, such as around the birth year 1996.

Since all of our variables are discrete with the exception of birth year, line graphs were not a good choice for data visualization. Bar charts are better suited to visualize discrete data, so we created bar charts as well. The line graphs displaying birth year are appropriate.

```
e = sns.barplot(data=pd.read_csv('Pulsesurvey.csv'), x='Education', y='Getvacc')
e.set(ylim=[1,2])
e.set(yticks=[1,2])
e.set(xticklabels=['<HS', 'Some HS', 'GED', 'Some \nUni', 'Assoc.\nDegree', 'Bachelor\nDegree', 'Master\nDegree'])
e.set(yticklabels=['More likely to\nget vaccine', 'Less likely to\nget vaccine'])
e.set(ylabel='Vaccine Response')
e.set(title='Likelihood of getting Vaccine\nby Education')
print(e)
```

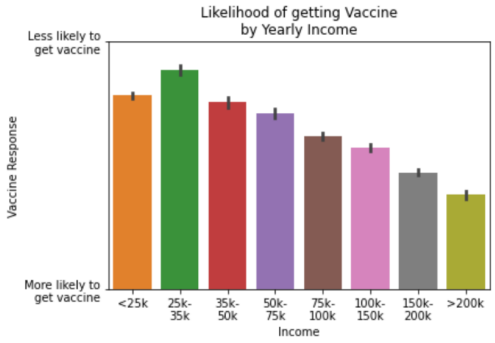
```
AxesSubplot(0.125,0.125;0.775x0.755)
```



This is a bar chart showing the likelihood of participants to answer either more likely to get the vaccine or less likely to get the vaccine by their education (<HS - Masters). It is much easier with the bar chart to see which responses more strongly correlated to vaccine hesitancy. Those with a masters degree were most likely to get the vaccine and those with less than a high school education were least likely to get the vaccine. You can see where those with some university were more likely than those with an associates degree to get the vaccine, which contradicts the trend somewhat.

```
f = sns.barplot(data=pd.read_csv('Pulsesurvey.csv'), x='Income', y='Getvacc')
f.set(ylim=[1,2])
f.set(yticks=[1,2])
f.set(xlim=[0.5,8.5])
f.set(xticks=[1,2,3,4,5,6,7,8])
f.set(xticklabels=['<25k', '25k-\n35k', '35k-\n50k','50k-\n75k', '75k-\n100k', '100k-\n150k', '150k-\n200k', '>200k'])
f.set(yticklabels=['More likely to\nget vaccine', 'Less likely to\nget vaccine'])
f.set(ylabel='Vaccine Response')
f.set(title='Likelihood of getting Vaccine\nby Yearly Income')
print(f)
```

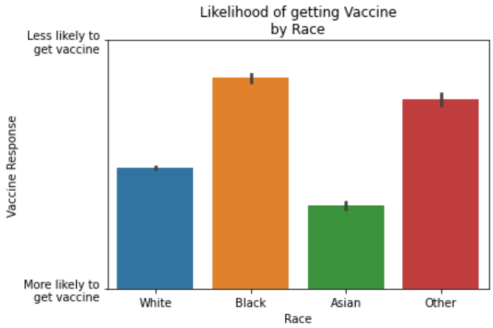
```
AxesSubplot(0.125,0.125;0.775x0.755)
```



This is a bar chart showing the likelihood of participants to answer either more likely to get the vaccine or less likely to get the vaccine by their income (<$25k - >$200k). You can still see with this chart that the general trend is that those with higher income are more likely to get the vaccine than those with lower income, however, you can also see that those who responded that their household income was between 25k and 35k were least likely to receive the vaccine, even less likely than those with a household income of less than 25k.

```
g = sns.barplot(data=pd.read_csv('Pulsesurvey.csv'), x='Race', y='Getvacc')
g.set(ylim=[1,2.2])
g.set(yticks=[1,2.2])
g.set(xticklabels=['White', 'Black', 'Asian', 'Other'])
g.set(yticklabels=['More likely to\nget vaccine', 'Less likely to\nget vaccine'])
g.set(ylabel='Vaccine Response')
g.set(title='Likelihood of getting Vaccine\nby Race')
print(g)
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```

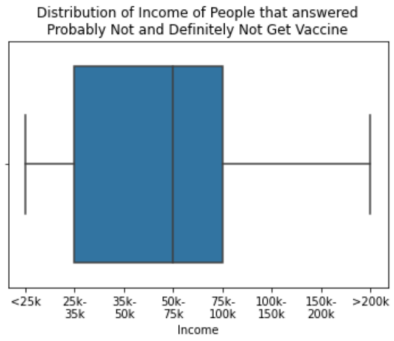


This is a bar chart showing the likelihood of participants to answer either more likely to get the vaccine or less likely to get the vaccine by their race (White, Black, Asian, or Other). The bar chart is a much more appropriate method of displaying the information about vaccine responses by race. Here it is much more obvious that Asian respondents were most likely to receive the vaccine, followed by White respondents,

then participants that responded "other", and then Black respondents answered that they were least likely to receive the vaccine.

```
import seaborn as sns
data = pd.read_csv('Pulsesurvey_3_4_NoNull.csv')
i = sns.boxplot(x=data['Income'])
i.set(xticks = [1,2,3,4,5,6,7,8])
i.set(xticklabels=['<25k', '25k-\n35k', '35k-\n50k','50k-\n75k', '75k-\n100k', '100k-\n150k', '150k-\n200k', '>200k'])
i.set(title='Distribution of Income of People that answered\nProbably Not and Definitely Not Get Vaccine')
print(i)
```

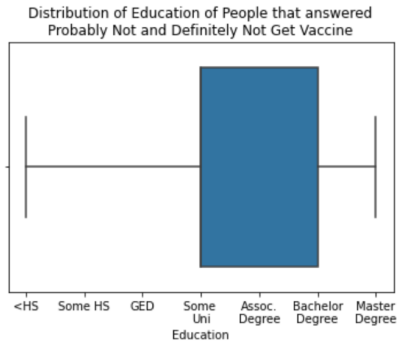
```
AxesSubplot(0.125,0.125;0.775x0.755)
```



This is a box and whisker plot showing the distribution of Income of people that answered probably not and definitely not getting the vaccine. This is another distribution, similar to the histograms earlier, however, this is a subset of our data consisting only of participants' answers who responded either that they would probably not get the vaccine or definitely not get the vaccine. This distribution shows that participants who responded probably not or definitely not were most likely to have a household income between 50k and 75k.

```
import seaborn as sns
data = pd.read_csv('Pulsesurvey_3_4_NoNull.csv')
j = sns.boxplot(x=data['Education'])
j.set(xticks = [1,2,3,4,5,6,7])
j.set(xticklabels=['<HS', 'Some HS', 'GED', 'Some \nUni', 'Assoc.\nDegree', 'Bachelor\nDegree', 'Master\nDegree'])
j.set(title='Distribution of Education of People that answered\nProbably Not and Definitely Not Get Vaccine')
print(j)
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```

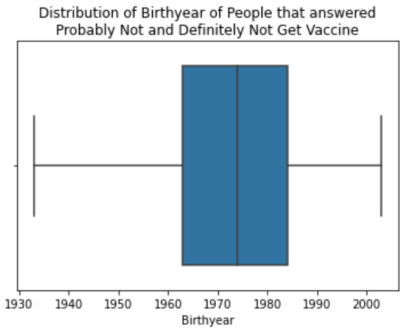


This is a box and whisker plot showing the distribution of Education of people that answered probably not and definitely not getting the vaccine. This distribution shows that participants who responded probably not or definitely not were most likely to have an education range between some university and a

bachelor's degree. These last two responses are interesting as they seem to contradict our earlier visualizations, however, it could also be attributed to more responses overall coming from those groups, not just in the subset which answered probably not and definitely not.

```
import seaborn as sns
data = pd.read_csv('Pulsesurvey_3_4_NoNull.csv')
k = sns.boxplot(x=data['Birthyear'])
k.set(title='Distribution of Birthyear of People that answered\nProbably Not and Definitely Not Get Vaccine')
print(k)
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



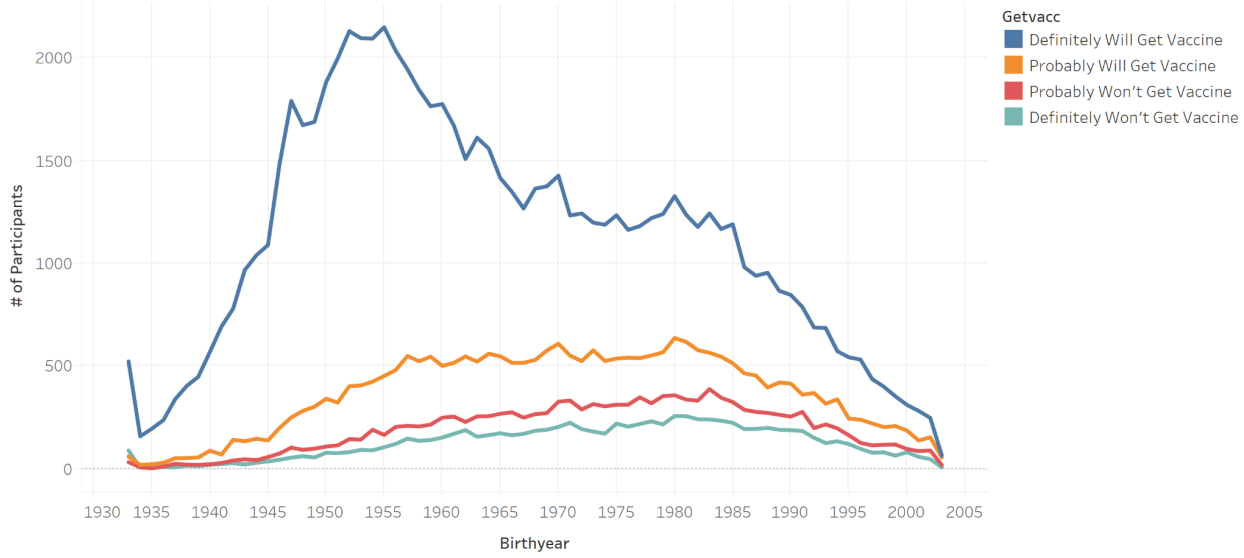
This is a box and whisker plot showing the distribution of Birth Year of people that answered probably not and definitely not getting the vaccine. This distribution shows that participants who responded probably not or definitely not were most likely to be born between 1965 and 1985.

```
import pandas as pd
import scipy.stats as stats
from scipy.stats import normaltest
data = pd.read_csv('Pulsesurvey_3_4.csv')

print(stats.normaltest(data['Birthyear']))
```

```
NormaltestResult(statistic=745.1208432449868, pvalue=1.5814836770723784e-162)
```

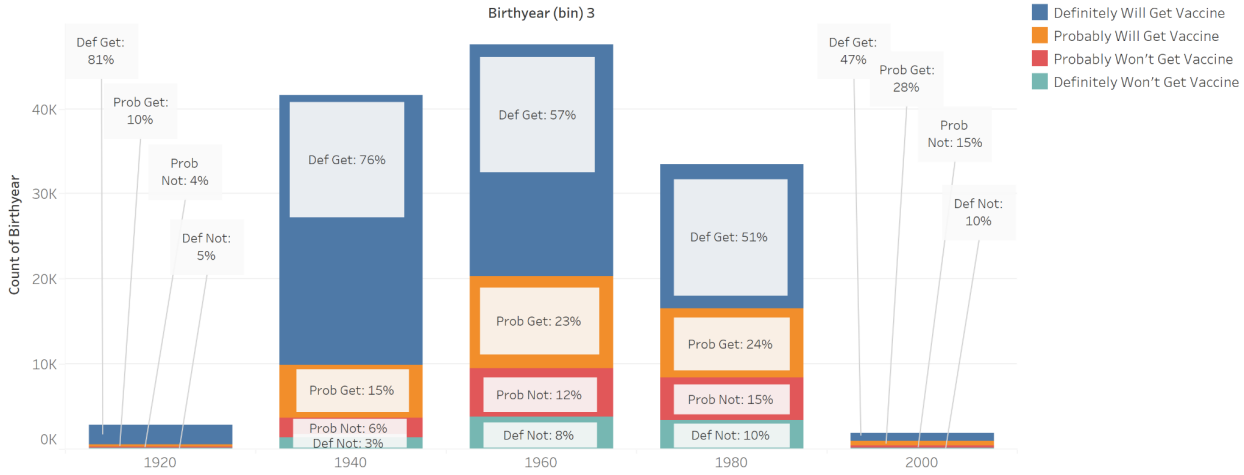
This is a normality test which showed P value for Birthyear in the set of our data where participants answered either 3 (probably not) or 4 (definitely not) to be 1.5814836770723784e-162, which means that birth year is a statistically significant predictor for vaccine hesitancy.

Vaccine by Age (All Responses) Line



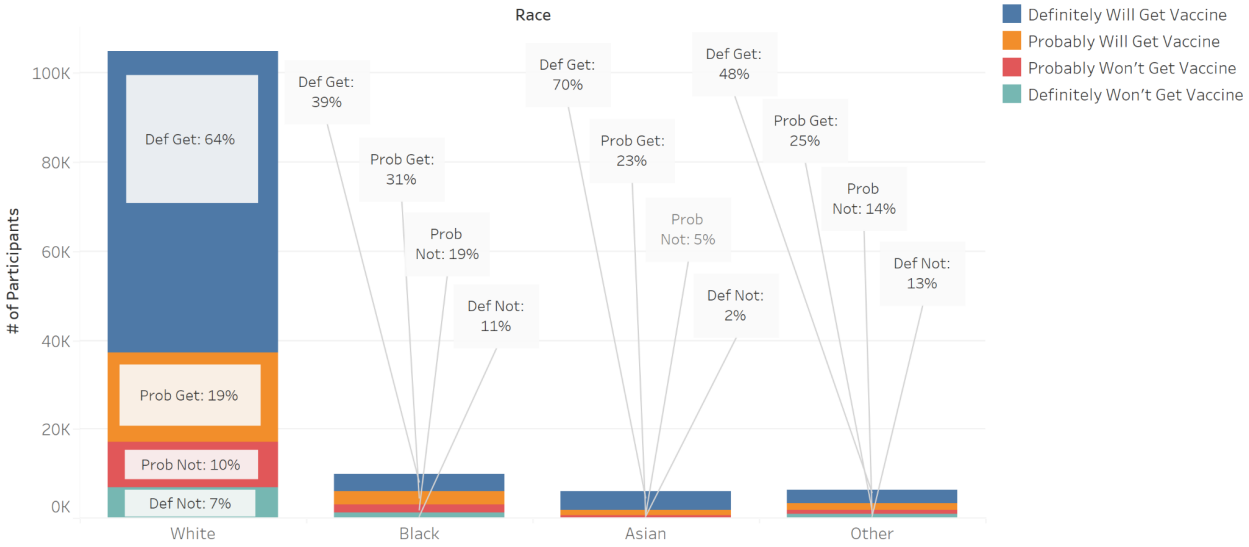
The trend of count of Birthyear for Birthyear. Color shows details about Getvacc.

Histogram of Age by Vaccine



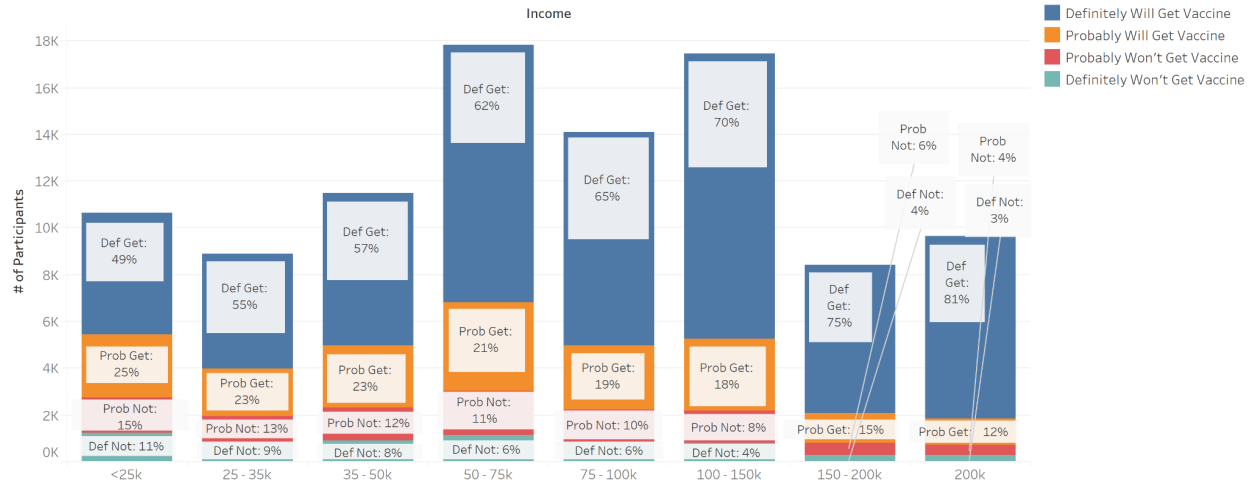
Count of Birthyear for each Birthyear (bin) 3. Color shows details about Getvacc.

Vaccine by Race (All Responses)



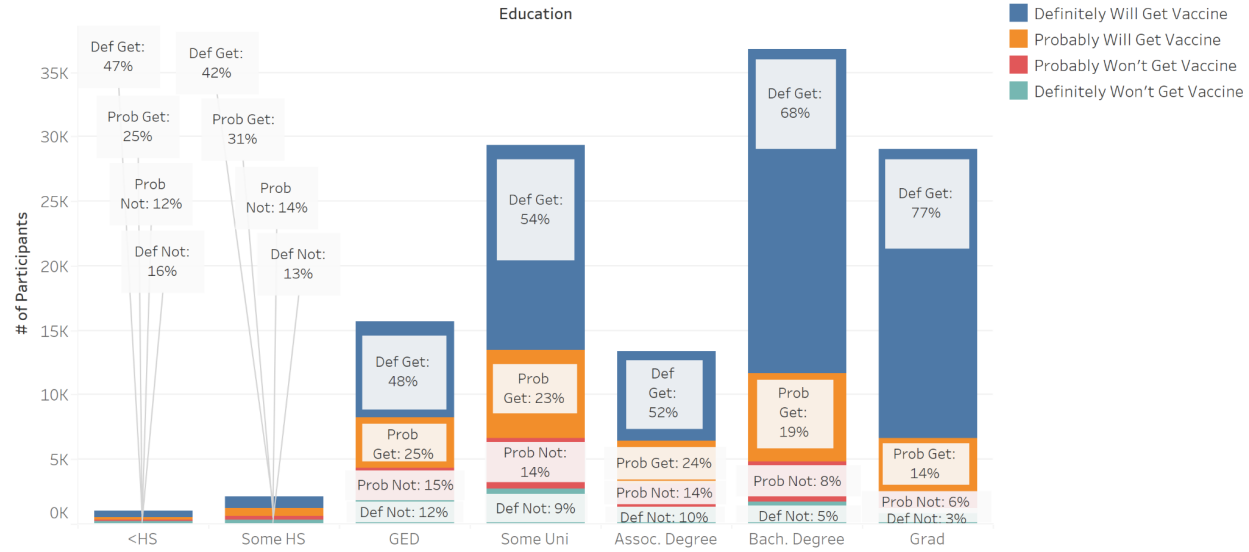
Count of Birthyear for each Race. Color shows details about Getvacc.

Vaccine by Income (All Responses)



Count of Birthyear for each Income. Color shows details about Getvacc. The view is filtered on Income, which excludes -99 and -88.

Vaccine by Education (All Responses)



Count of Birthyear for each Education. Color shows details about Getvacc.

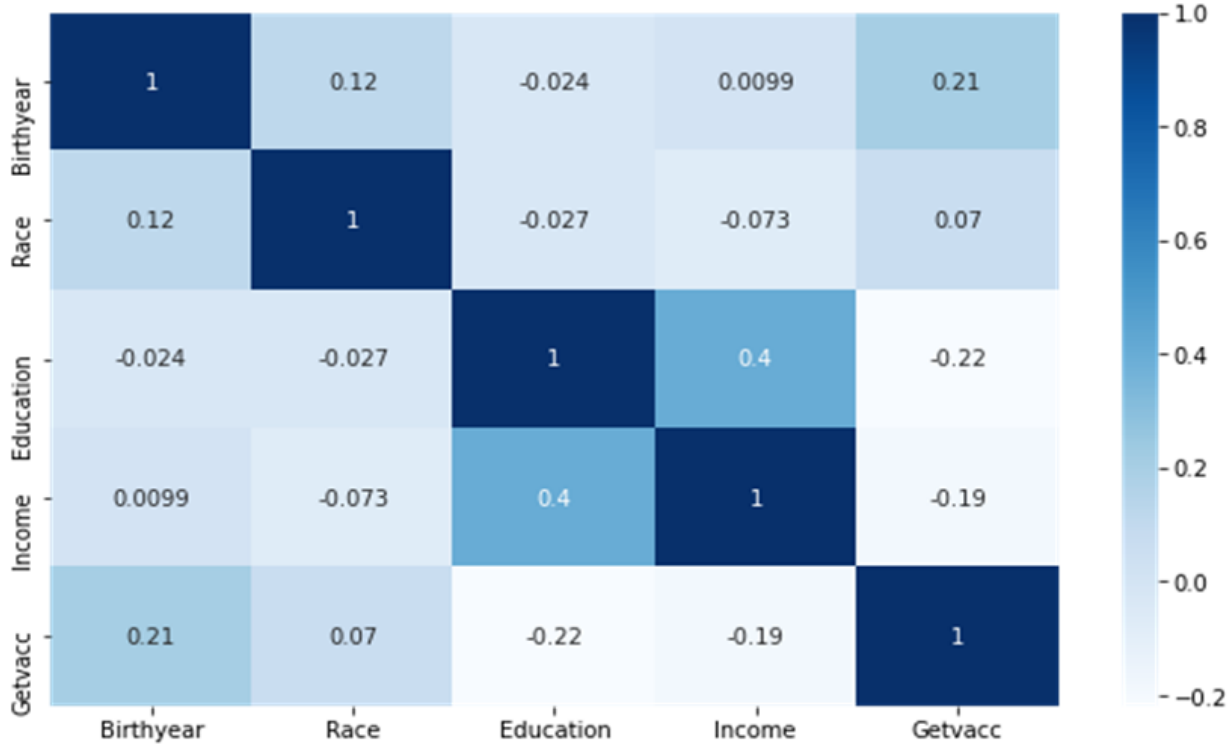
We decided that, because all of our variables were discrete, in order to succinctly show how the vaccine response varies by variable response. This could have been created in Python, however, we realized stacked bar charts would be a good addition to our visualizations with very little time until the deadline so we chose to create the stacked bar charts in Tableau in order to be time efficient. Pulsesurvey.csv was imported into Tableau. Birthyear count was used for Y values as it encompasses all of the responses to create a distribution. X values were Income, Education, or Race as discrete dimensions, then we used "Getvacc" as a second X value, also a discrete dimension, but pulled it onto the "color" menu, which made it into a stacked bar chart. The stacked charts made in tableau show the distribution of vaccine attitude responses by income, race, education, and birth year. Each bar in the chart is split into their vaccine responses by percentage. The stacked bars were then labeled using the annotation tool in Tableau. This shows the distribution of responses across the entirety of the data along with how many of those responses in each category responsed definitely yes, probably yes, probably not, and definitely not to

getting the vaccine. The percentages make it easier to compare responses from one group (e.g. 1940, 1960) to another. For instance, you can see that 10% of participants in the 1980 bin responded that they would definitely not get the vaccine compared to 8% in the 1960 bin.

5.1.2 Correlation Among Attributes

To understand the correlation among the attributes with the dependent variable, we have used Pearson Correlation method (linear correlation) and plotted a heat map of the correlation result.

It showed that there is a negative correlation between “Getvacc” with “Income” and “Education” (-0.19 and -0.22 respectively), while there is a positive correlation with “Race” and “Birthyear” (0.07 and 0.21 respectively).



5.1.3 Association Hypothesis Testing Using ANOVA

5.1.3 Association Hypothesis Testing Using ANOVA

To test our first part of hypothesis, “The factors we examined show associations with the participants’ willingness to get COVID-19 vaccines”, we used ANOVA (Analysis of variance) to estimate associations between groups of variables and analyze mean

differences. We ran one-way ANOVA to analyze each of 3 categorical variables, “Income”, “Education”, and “Age” to test the dependent variable’s dependence, “Getvacc”, on those category predictors.

```
import researchpy as rp
rp.summary_cont(df_income['Income'].groupby(df_income['Getvacc']))
```

Getvacc	N	Mean	SD	SE	95% Conf.	Interval
1	63088	4.8200	2.0693	0.0082	4.8038	4.8361
2	19361	4.1279	2.0259	0.0146	4.0994	4.1565
3	9722	3.9159	1.9909	0.0202	3.8763	3.9554
4	6256	3.7211	2.0166	0.0255	3.6711	3.7710

5.1.3.1. One-way ANOVA by Income

We had imported “researchpy” to get all standard necessary information in one process. For people who responded “Definitely Yes” for “Getvacc”[1], 95% included true mean of 4.8 category, close to 5, which is the income of $75-100K, whereas for people who responded “Definitely No” for “Getvacc”[4], 95% included true mean of 3.67, which is the income between $35-50K and $50-75K.

```
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols

mod = ols('Getvacc ~ Income', data=df_income).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)

def anova_table(aov):
    aov['mean_sq'] = aov[:]['sum_sq']/aov[:]['df']

    aov['eta_sq'] = aov[:-1]['sum_sq']/sum(aov['sum_sq'])

    aov['omega_sq'] = (aov[:-1]['sum_sq']-(aov[:-1]['df']*aov['mean_sq'][-1]))
    /(sum(aov['sum_sq'])+aov['mean_sq'][-1])

    cols = ['sum_sq', 'df', 'mean_sq', 'F', 'PR(>F)', 'eta_sq', 'omega_sq']
    aov = aov[cols]
    return aov

anova_table(aov_table)
```

	sum_sq	df	mean_sq	F	PR(>F)	eta_sq	omega_sq
Income	2786.182263	1.0	2786.182263	3511.690699	0.0	0.03445	0.03444
Residual	78090.587342	98425.0	0.793402	NaN	NaN	NaN	NaN

From statsmodel, we were able to create ANOVA table and the "Income" variable showed F value 3511.69 which is very high that indicates the variability of group means in "Income variable" is large in different dependent variable groups. Eta-squared, the percentage of variance in dependent variable from the independent variable, was about 3%, and the omega-squared, the adjusted variation relative to sample size, was also 3%.

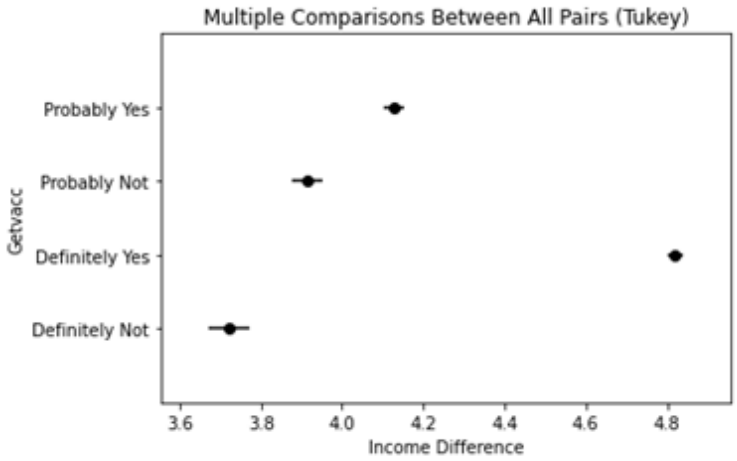
```
import statsmodels.stats.multicomp as mc

comp = mc.MultiComparison(df_income['Income'], df_income['Getvacc_explain'])
post_hoc_res = comp.tukeyhsd()
post_hoc_res.summary()
```

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
Definitely Not	Definitely Yes	1.0989	0.001	1.0291	1.1687	True
Definitely Not	Probably Not	0.1948	0.001	0.1094	0.2801	True
Definitely Not	Probably Yes	0.4069	0.001	0.3303	0.4835	True
Definitely Yes	Probably Not	-0.9041	0.001	-0.9615	-0.8468	True
Definitely Yes	Probably Yes	-0.6921	0.001	-0.7353	-0.6488	True
Probably Not	Probably Yes	0.2121	0.001	0.1466	0.2775	True

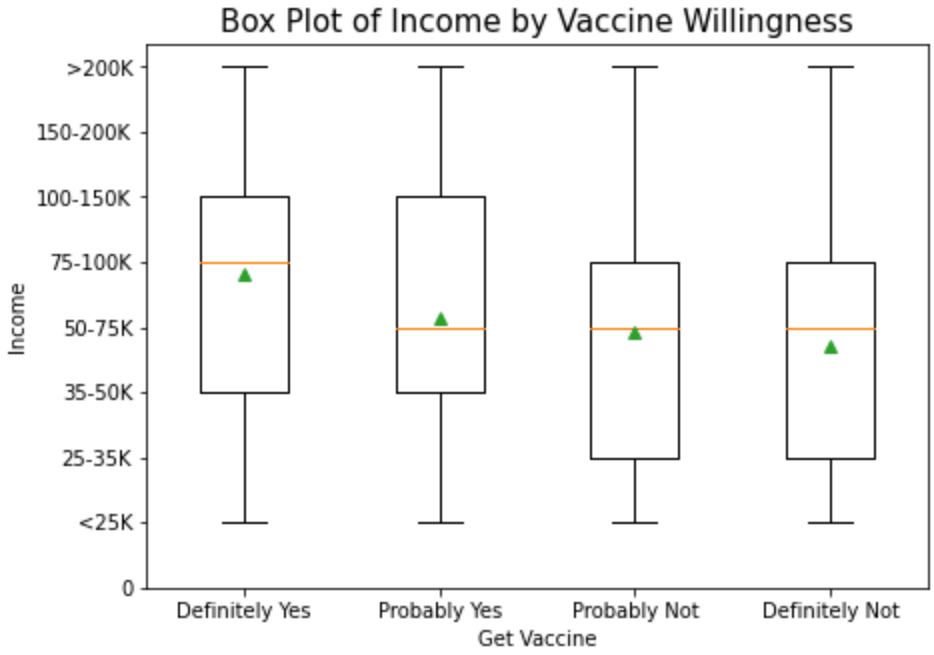
Since the F-test showed a significant difference between the different responses from "Getvacc" column in terms of "Income" variable, we ran a Tukey's Honest Significant Difference (HSD) test to see the differences in pairs of group means. As shown above, the response group of "Definitely Not" revealed the highest mean difference compared to the response group of "Definitely Yes" and the null hypothesis of "showing no association with examined factors" is rejected at all levels.

```
import matplotlib.pyplot as plt
post_hoc_res.plot_simultaneous(ylabel= "Getvacc", xlabel= "Income Difference", figsize = (6,4))

plt.yticks(np.arange(4), ['Definitely Not', 'Definitely Yes', 'Probably Not', 'Probably Yes'])
plt.show()
```



The plot above illustrates that the difference in Income differences among each response groups is the highest between the response group of "Definitely Not" and "Definitely Yes" which is approximately 1.2 (4.8-3.6).

```
plt.close()
fig = plt.figure(figsize= (7, 5))
ax = fig.add_subplot(111)

ax.set_title("Box Plot of Income by Vaccine Willingness", fontsize= 15)
ax.set

data = [df_income[df_income['Getvacc'] == 1]['Income'],
        df_income[df_income['Getvacc'] == 2]['Income'],
        df_income[df_income['Getvacc'] == 3]['Income'],
        df_income[df_income['Getvacc'] == 4]['Income']]

ax.boxplot(data,labels= ['Definitely Yes', 'Probably Yes', 'Probably Not',
                         'Definitely Not'], showmeans= True)
plt.xlabel("Get Vaccine")
plt.ylabel("Income")
plt.yticks(np.arange(9), ['0', '<25K', '25-35K', '35-50K', '50-75K', '75-100K',
                          '100-150K', '150-200K', '>200K'])
plt.show()
```



Above is the boxplot that demonstrates the difference in Income in different response groups. Therefore, based on the ANOVA and post hoc test result of Tukey HSD, "Income" shows high association with the participants' willingness to get COVID-19.

5.1.3.2. One-way ANOVA by Education

For people who responded, "Definitely Yes" for "Getvacc"[1], 95% included true mean of 5.4 category which is the education between "Bachelor's Degree" and "Graduate Degree",

whereas for people who responded, "Definitely No" for "Getvacc"[4], 95% included true mean of 4.5, which is the education between "Associate's Degree" and "Bachelor's Degree".

```
rp.summary_cont(df['Education'].groupby(df['Getvacc']))
```

Getvacc	N	Mean	SD	SE	95% Conf.	Interval
1	79052	5.4375	1.4231	0.0051	5.4275	5.4474
2	25908	4.9034	1.4614	0.0091	4.8856	4.9212
3	13380	4.7092	1.4117	0.0122	4.6853	4.7331
4	8921	4.5366	1.4389	0.0152	4.5067	4.5665

The ANOVA table of "Education" variable showed F value 1121.22 which is also very high that signifies the variability of group means in "Education variable" is also large in different dependent variable groups of "Getvacc". The percentage of variance in dependent variable from the independent variable (Eta_squared/omega-squared), was about 5%, slightly higher than "Income" variable.

```
mod = ols('Getvacc ~ Birthyear_codes', data=df).fit()
aov_table = sm.stats.anova_lm(mod, typ=2)
anova_table(aov_table)
```

	sum_sq	df	mean_sq	F	PR(>F)	eta_sq	omega_sq
Birthyear_codes	5606.077842	5.0	1121.215568	1364.925214	0.0	0.0509	0.050862
Residual	104533.410108	127255.0	0.821448	NaN	NaN	NaN	NaN

As demonstrated above from the Tukey HSD test result between pairs of groups, the response group of "Definitely Not" showed the highest mean difference in comparison to the response group of "Definitely Yes" and the null hypothesis of no association between "Education" variable and outcome variable, "Getvacc", is rejected at all levels as well.

```
comp = mc.MultiComparison(df['Education'], df['Getvacc_explain'])
post_hoc_res = comp.tukeyhsd()
post_hoc_res.summary()
```

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
Definitely Not	Definitely Yes	0.9009	0.001	0.8598	0.9419	True
Definitely Not	Probably Not	0.1726	0.001	0.1223	0.2228	True
Definitely Not	Probably Yes	0.3668	0.001	0.3217	0.4119	True
Definitely Yes	Probably Not	-0.7283	0.001	-0.7626	-0.6939	True
Definitely Yes	Probably Yes	-0.5341	0.001	-0.5604	-0.5078	True
Probably Not	Probably Yes	0.1942	0.001	0.1551	0.2333	True

The plot above illustrates that the difference in Education differences among each response groups is the highest between the response group of "Definitely Not" and "Definitely Yes" which is roughly 0.8 (5.4-4.6), marginally lower than "Income".

Above is the boxplot that proves the difference in Education in different response groups where the mean education in "Definitely Yes" response group is much higher than the average education who responded "Definitely No." Hence, based on the ANOVA and post hoc test result of Tukey HSD, "Income" also shows high association with the participants' willingness to get COVID-19.

5.1.3.3. One-way ANOVA by Age

95% included true mean of birthyear, 1965.2 For people who responded, "Definitely Yes", in contrast to people who responded, "Definitely No", 95% included true mean of birthyear, 1974.4

From the ANOVA table and the "Birthyear_codes" variable showed F value of 1364.92 which was also very high that suggests the variability of group means in "Birthyear" variable is large in different dependent variable groups. Eta-squared and omega-squared was also about 5%, similar to "Education".

From the Tukey HSD test post-hoc test result, the response group of "Definitely Not" showed the highest mean difference in comparison to the response group of "Definitely Yes" in age and the null hypothesis of no association between "Age" variable and outcome variable, "Getvacc", is rejected at all levels except for "Definitely Not" and "Probably Not" group. This means that the mean age difference was not significantly different in those two groups of respondents.

The plot above explains that the difference in Education differences among each response groups is the highest between the response group of "Definitely Not" and "Definitely Yes" which is nearly 1 (4.8-3.8) code of Birthyear_codes which is about the age of 10 years.

The boxplot indicates the difference in Age in different response groups where the mean Age in "Definitely Yes" response group, 41-50, was much higher than the average age of people who responded, "Definitely No", the age of 31-40. Consequently, based on the F-value and post hoc test result of Tukey HSD, "Age" also shows high association with the participants' willingness to get COVID-19 and the null hypothesis is rejected for all three variables.

5.1.3.1. One-way ANOVA by Income

5.1.3.2. One-way ANOVA by Education

5.1.3.3. One-way ANOVA by Age

5.2 Supervised Learning

5.2.1. Binary Classification Model Development

5.2.2. Feature Selection Using Chi-Square Test

5.2.2. Performance Analysis of Binary Classifiers

Supervised Machine learning models

We did analysis using machine learning models. Seven different models were done to determine the accuracy and plotted the receivers operating characteristic curve to determine the significant data.

Classification models tested:

1. Logistic regression

2. Random Forest

3. Support vector machine

4. Decision tree classifier

5. XG boost

6. Naive Bayes `

7. Gradient boost

Sklearn and matplotlib libraries were imported to perform the analysis and plot the area AUROC.

The predictor taken was the 'binary' column in the Y axis and the respondents taken were 'birth year codes', 'income', 'education' in the X axis the training and test data were divided in a ratio of 7: 3.

Birth year codes were obtained by encoding the birth year column, Binary column was obtained by encoding the responses of the individual's responses to the survey, people who responded, 'definitely get the vaccine' and 'probably will get the vaccine' were assigned the value 1, people who responded, 'will definitely not get the vaccine' and 'probably not get the vaccine' were assigned the value 0.

5.2.1.1.Logistic Regression

We built a logistic regression model to evaluate the binary classification using Birthyear_codes, Income, Education as feature columns (x) and binary as the target (y).

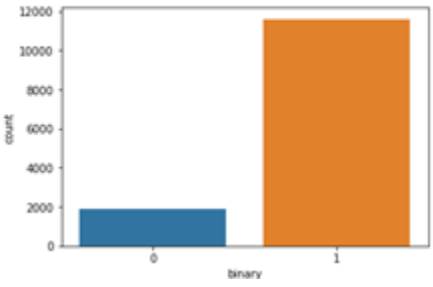
```
X = df_new[['Birthyear_codes', 'Income', 'Education']]
y = df_new["binary"]
```

```
y.value_counts()
```

A histogram is plotted to display the binary model.

```
sns.countplot(y)
```

```
<AxesSubplot:xlabel='binary', ylabel='count'>
```

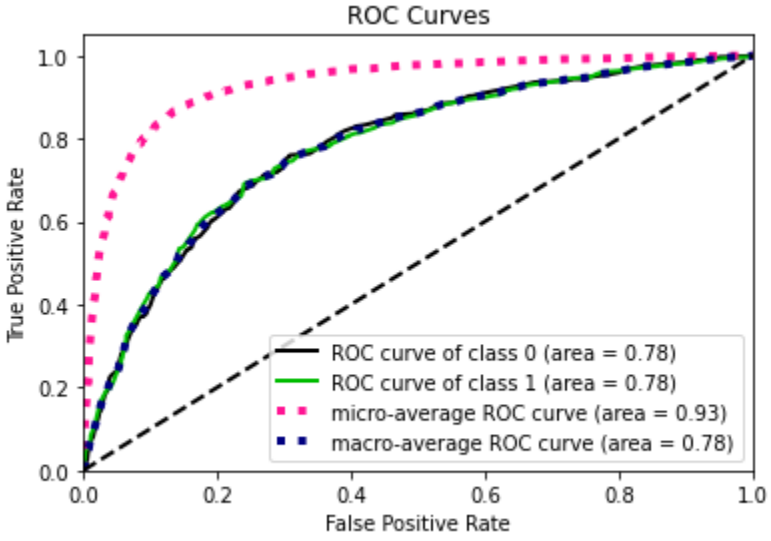


Label encoding is done to transform the target values. The dataset is divided into x_train, y_train, x_test and y_test groups(training and testing models). Feature columns are

transformed. The logistic regression model is imported maximizing iteration to 10000 which limits the running of code.

```
model= LogisticRegression(max_iter=10000)
model.fit(X_train_fs, y_train_enc)
y_pred= model.predict(X_test_fs)
```

Fit function trains both the target and features. ROC curve is plotted by printing the true positive rate and false positive rate and derived AUC value is 0.78 and accuracy is 86.34.




```
accuracy = accuracy_score(y_test_enc, y_pred)
print('Accuracy: %.2f' % (accuracy*100))
```

```
Accuracy: 86.34
```

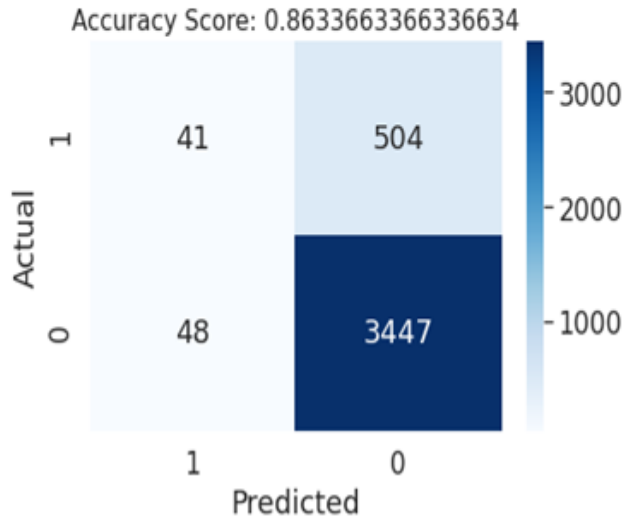
We then calculated the scores of accuracy, precision, recall, F1 score and F beta score and classification report is made.

```
print("Accuracy:",accuracy_score(y_test_enc, y_pred))
print("Precision:",precision_score(y_test_enc, y_pred,average='weighted'))
print("Recall:",recall_score(y_test_enc, y_pred,average='weighted'))
print("F1-score:",f1_score(y_test_enc, y_pred,average='weighted'))
print("F beta score is",fbeta_score(y_test, y_pred, beta=2))
```

```
Accuracy: 0.8633663366336634
Precision: 0.816890094220448
Recall: 0.8633663366336634
F1-score: 0.8184137269742653
F beta score is 0.9611845407395013
```

Confusion matrix is printed and visualized to represent actual and predicted values. The performance of logistic regression is represented in terms of accuracy.

```
#Printing Confusion matrix
cm=confusion_matrix(y_test_enc,y_pred)
df_cm = pd.DataFrame(cm, columns=class_labels, index = class_labels)
df_cm.index.name = 'Actual'
df_cm.columns.name = 'Predicted'
sns.set(font_scale=1.5)
sns.heatmap(df_cm, annot=True,cmap="Blues",fmt='d')
score=accuracy_score(y_test_enc, y_pred)
all_sample_title = 'Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 15);
```



5.2.1.2 Support Vector Machine

SVM works by constructing a hyperplane between two classes of data points that could be chosen. The decision boundaries help in classifying the data points based on which side of the hyperplane they are.

Code:

```
In [232]: from sklearn.svm import SVC
          sv_clf = SVC(probability=True, kernel='linear')
          fit=sv_clf.fit(X_train,y_train)
          score=sv_clf.score(X_train,y_train)
          prediction=sv_clf.predict(X_test)
          print(fit,score,prediction)
```

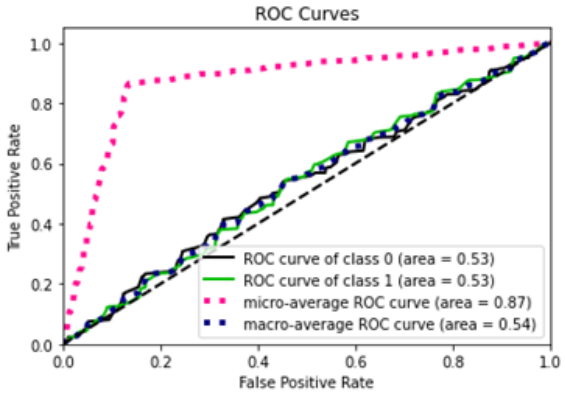
```
SVC(kernel='linear', probability=True) 0.8597198641765704 [1 1 1 ... 1 1 1]
```

```
In [245]: y_pred = sv_clf.predict(X_test)

          print(accuracy_score(y_test_enc,y_pred))
          print("\n")
          print(confusion_matrix(y_test_enc, y_pred))
          print("\n")
          print(classification_report(y_test_enc, y_pred))
```

```
0.8650990099009901


[[   0  545]
 [   0 3495]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00       545
           1       0.87      1.00      0.93      3495

    accuracy                           0.87      4040
   macro avg       0.43      0.50      0.46      4040
weighted avg       0.75      0.87      0.80      4040
```

The accuracy score is 0.865.
ROC curve

In [255]:
```
sv_clf.fit(X_train, y_train)
predicted_probas = sv_clf.predict_proba(X_test)
import matplotlib.pyplot as plt
import scikitplot as skplt
skplt.metrics.plot_roc(y_test, predicted_probas)
plt.show()
```



The area under curve was 0.53 which was very low indicating that there was a lot of data that was insignificant in nature .

5.2.1.3 Naives Bayes Classifier

Gaussian Naives Bayes:

We built a Gaussian Naives Bayes model to evaluate the binary classification using Birthyear_codes,Income,Education as feature columns(x) and binary as the target(y).

After the data has been pre - processed, the next step is to divide it into sections that will be used to build and train the model, as well as to test and evaluate the model.

In [195]:
```
X = df_new[['Birthyear_codes', 'Income', 'Education']]
y = df_new["binary"]

def prepare_targets(y_train, y_test):
    le = LabelEncoder()
    le.fit(y_train)
    y_train_enc = le.transform(y_train)
    y_test_enc = le.transform(y_test)
    return y_train_enc, y_test_enc

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 42)
y_train_enc, y_test_enc = prepare_targets(y_train, y_test)
```

Firstly,import Gaussian Naives Bayes model and create a Gaussian classifier, then train the model using the training sets and finally predict.

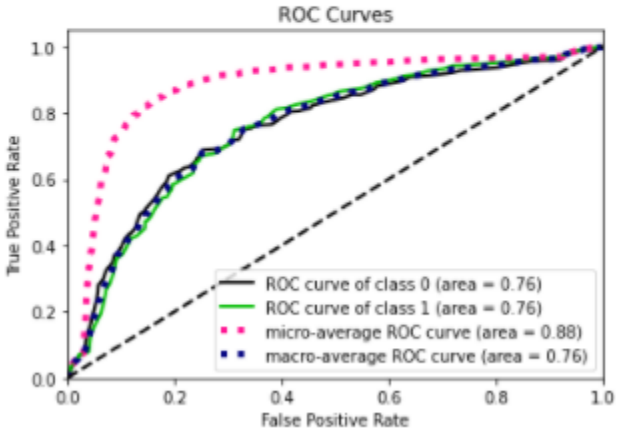
```
In [207]: from sklearn.naive_bayes import GaussianNB
          nb_clf = GaussianNB()
          nb_clf.fit(X_train,y_train)
          nb_clf.score(X_train,y_train)

Out[207]: 0.8395585738539898
```

Both the features and the target are trained using the Fit function. The false positive rate and true positive rate are plotted to build the ROC curve.

AUC value obtained is 0.76

Accuracy derived is 84.75

```
In [259]: nb_clf.fit(X_train, y_train)
          predicted_probas = nb_clf.predict_proba(X_test)

          import matplotlib.pyplot as plt
          import scikitplot as skplt
          skplt.metrics.plot_roc(y_test, predicted_probas)
          plt.show()
```



Eventually classification report is done after calculating the values of precision,recall,F1 score,accuracy and confusion matrix.

In [236]:
```
y_pred = nb_clf.predict(X_test_fs)

print(accuracy_score(y_test_enc,y_pred))
print("\n")
print(confusion_matrix(y_test_enc, y_pred))
print("\n")
print(classification_report(y_test_enc, y_pred))
```

```
0.8475247524752475


[[ 164  381]
 [ 235 3260]]


              precision    recall  f1-score   support

           0       0.41      0.30      0.35       545
           1       0.90      0.93      0.91      3495

    accuracy                           0.85      4040
   macro avg       0.65      0.62      0.63      4040
weighted avg       0.83      0.85      0.84      4040
```

DECISION TREE CLASSIFIER:
We built a decision tree classifier and predicted the accuracy score and plotted thereof curve.
Code:

```
In [200]: from sklearn import tree
          model = tree.DecisionTreeClassifier(max_depth=5)
          model.fit(X_train,y_train)
          model.score(X_test,y_test)

          y_pred = model.predict(X_test)
          model.score(X_test,y_test)

Out[200]: 0.8668316831683168
```

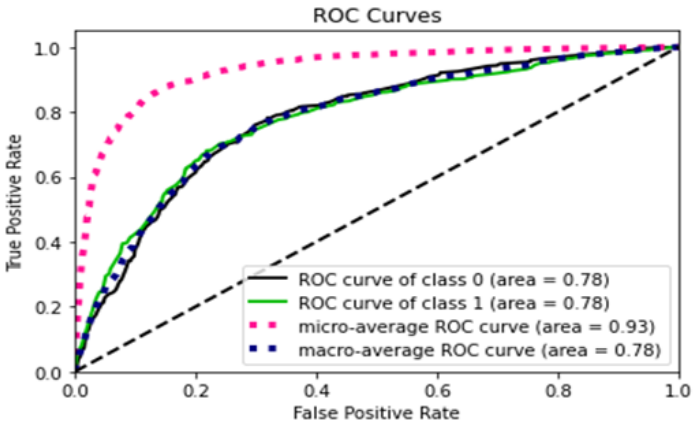
```
In [201]: y_pred = model.predict(X_test_fs)

          print(accuracy_score(y_test_enc,y_pred))
          print("\n")
          print(confusion_matrix(y_test_enc, y_pred))
          print("\n")
          print(classification_report(y_test_enc, y_pred))
```

```
0.8668316831683168


[[  29  516]
 [  22 3473]]


              precision    recall  f1-score   support

           0       0.57      0.05      0.10       545
           1       0.87      0.99      0.93      3495

    accuracy                           0.87      4040
   macro avg       0.72      0.52      0.51      4040
weighted avg       0.83      0.87      0.82      4040
```

The accuracy score is 0.866.

ROC curve

```
In [261]: model.fit(X_train, y_train)
          predicted_probas = model.predict_proba(X_test)

          import matplotlib.pyplot as plt
          import scikitplot as skplt
          skplt.metrics.plot_roc(y_test, predicted_probas)
          plt.show()
```



XG BOOST:

- Extreme gradient boost or the xg boost model is a linear combination of decision trees. Predictions are made by addition of individual trees. Since our data was categorical in nature the data was encoded using one hot encoding to run the model.

Code:

[215]:
```
# one-hot encode the categorical features
cat_attribs = ['Birthyear_codes']
full_pipeline = ColumnTransformer([('cat', OneHotEncoder(handle_unknown='ignore'), cat_attribs)], remainder='passthrough')

encoder = full_pipeline.fit(X_train)
X_train = encoder.transform(X_train)
X_test = encoder.transform(X_test)
```

[239]:
```
xg_clf = XGBClassifier(n_estimators=10, max_depth=20, verbosity=2)
xg_clf.fit(X_train, y_train)

# extract the training set predictions
xg_clf.predict(X_train)


# extract the test set predictions
xg_clf.predict(X_test)
```

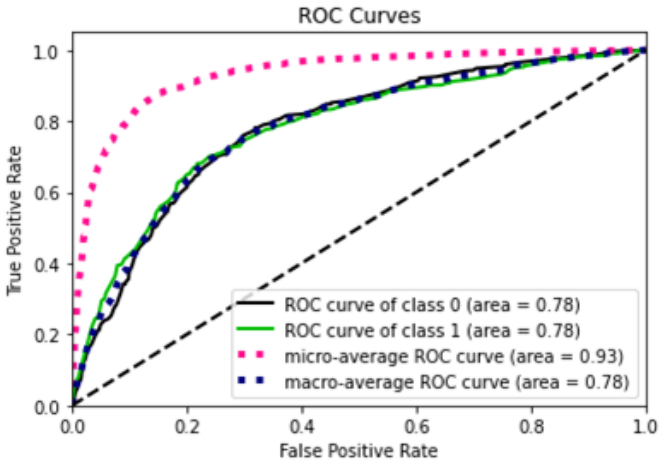
Out[239]:
```
array([1, 1, 1, ..., 1, 1, 1])
```

In [241]:
```
y_pred = xg_clf.predict(X_test)

print(accuracy_score(y_test_enc,y_pred))
print("\n")
print(confusion_matrix(y_test_enc, y_pred))
print("\n")
print(classification_report(y_test_enc, y_pred))
```

```
0.8636138613861386


[[  53  492]
 [  59 3436]]


              precision    recall  f1-score   support

           0       0.47      0.10      0.16       545
           1       0.87      0.98      0.93      3495

    accuracy                           0.86      4040
   macro avg       0.67      0.54      0.54      4040
weighted avg       0.82      0.86      0.82      4040
```

In [256]:
```
xg_clf.fit(X_train, y_train)
predicted_probas = xg_clf.predict_proba(X_test)
import matplotlib.pyplot as plt
import scikitplot as skplt
skplt.metrics.plot_roc(y_test, predicted_probas)
plt.show()
```

ROC Curve



5.2.1.4 Gradient Boost

Code:

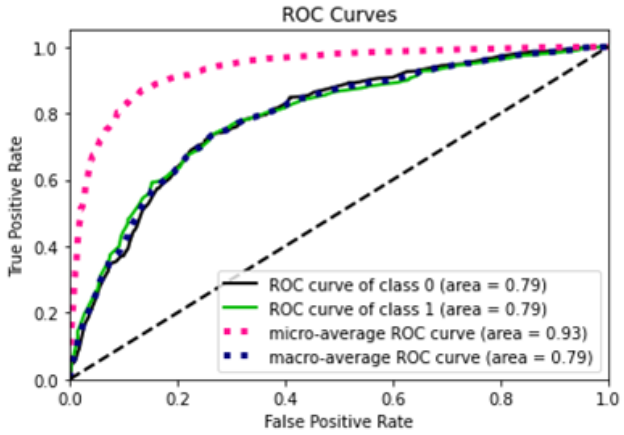
```
In [204]: gb_clf = ensemble.GradientBoostingClassifier(n_estimators=40)
          gb_clf.fit(X_train,y_train)
          gb_clf.score(X_train,y_train)

Out[204]: 0.860144312393888

In [235]: y_pred = gb_clf.predict(X_test_fs)

          print(accuracy_score(y_test_enc,y_pred))
          print("\n")
          print(confusion_matrix(y_test_enc, y_pred))
          print("\n")
          print(classification_report(y_test_enc, y_pred))

          0.8665841584158416


          [[  14  531]
           [   8 3487]]


                        precision    recall  f1-score   support

                     0       0.64      0.03      0.05       545
                     1       0.87      1.00      0.93      3495

              accuracy                           0.87      4040
             macro avg       0.75      0.51      0.49      4040
          weighted avg       0.84      0.87      0.81      4040
```

The accuracy score is 0.866.

The model gave good performance compared to other models indicating a good fit.

ROC curve

```
In [258]: gb_clf.fit(X_train, y_train)
predicted_probas = gb_clf.predict_proba(X_test)

import matplotlib.pyplot as plt
import scikitplot as skplt
skplt.metrics.plot_roc(y_test, predicted_probas)
plt.show()
```



The roc value is 0.79 which is the highest value obtained among all the models.

5.2.1.5 Random Forest

Random forest classification is done by using the variables Birthyear, Income and Education as features (x) and binary as (y). Data is built into training and testing models.

```
X = df_new[['Birthyear_codes', 'Income', 'Education']]
y = df_new["binary"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 42)
```

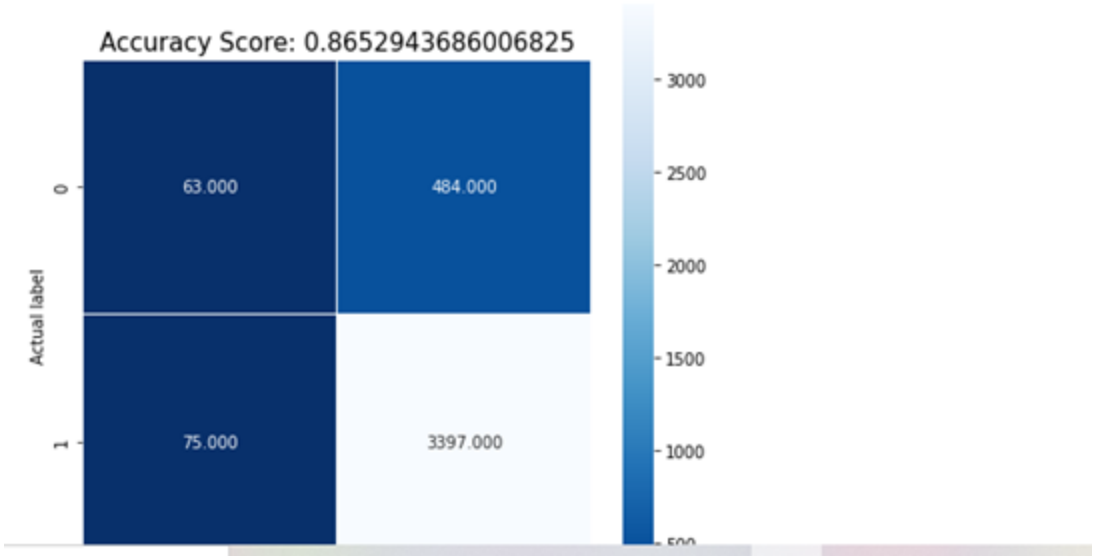
Random forest classifier is imported.

```
from sklearn.ensemble import RandomForestClassifier

rf_clf = ensemble.RandomForestClassifier(n_estimators=100)
rf_clf.fit(X_train,y_train)
score = rf_clf.score(X_train,y_train)
print('Accuracy score for Random Forest Classifier: ', score)
```

```
Accuracy score for Random Forest Classifier:  0.8652943686006825
```

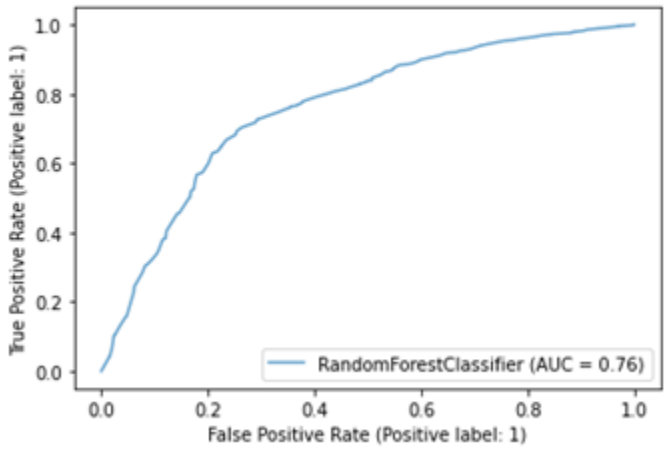
Accuracy score and classification report are printed. Confusion matrix is plotted to represent accuracy score, predicted and actual values.

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics

cm = metrics.confusion_matrix(y_test_enc, rf_predicted)
plt.figure(figsize=(7,7))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square=True, cmap='Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 15);
```



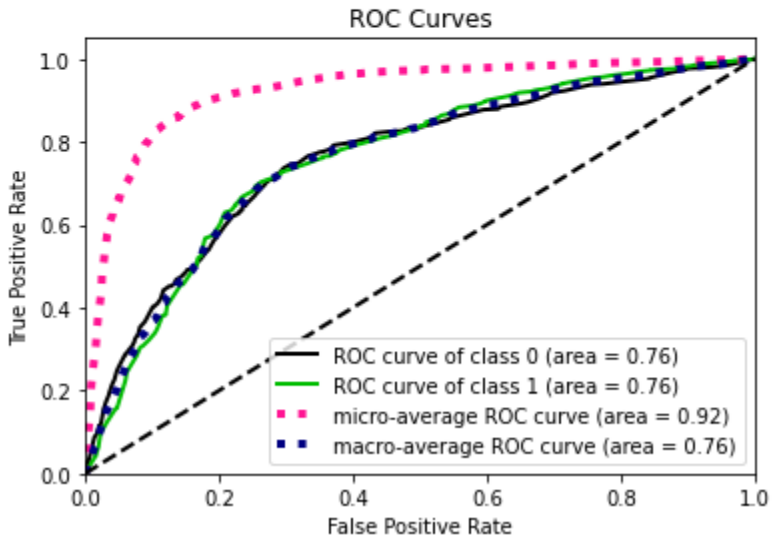
ROC curve is plotted and the obtained AUC value is 0.76.

```
ax = plt.gca()
rf_clf_disp = plot_roc_curve(rf_clf, X_test_fs, y_test_enc, ax=ax, alpha=0.7)
```



Probabilities are predicted through ROC curve for class 0, class 1, micro average and macro average.

```
import scikitplot as skplt

plt.rcParams['figure.figsize'] = [6, 4]
predicted_probas = rf_clf.predict_proba(X_test)
skplt.metrics.plot_roc(y_test, predicted_probas)

plt.show()
```



A histogram is plotted to reveal the importance of

Summary of scores

MODEL	ACCURACY	AREA UNDER CURVE
Logistic regression	0.854	0.78
Random forest	0.859	0.77
Support vector machine	0.86	0.53
Decision tree	0.86	0.78
XG boost	0.86	0.78
Naive bayes	0.84	0.76
Gradient Boost	0.86	0.79

5.2.3. Odds Ratio Analysis Using Logistic Regression

5.3 Unsupervised Learning

5.3.1. K-Means Clustering

5.3.2. Performance Analysis of Clustering

6. Summary of Findings

6.1 Research Question One Findings

6.2. Research Question Two Findings

Conclusion:

1) The alternate hypothesis had statistical significance that the examined (Age, Income and Education) factors contributed to people's hesitancy of getting vaccinated.

2) We were able to predict people's unwillingness to get vaccinated based upon those contributing factors and Odds Ratio.

3) In total we analyzed our data using 7 different machine learning methods this was because we wanted to better our predictive models each time to understand which could better fit and predict the data accurately.

With Logistic Regression only probability between two values can be predicted, so we tried Support Vector Machine (SVM) which yielded an AUC of just 0.53 that shows only 53% of our data to be significant stating that this method showed a poor performance. Hence to better this we used Naives- Bayes which has cons of being less accurate when compared to other better trained models. Later we used Random Forest, Decision Tree Classifier, Gradient boost models to keep enhancing our prediction models but it was XG Boost that yielded satisfactory results with an ROC of 0.78 and this is the best model with enhanced performance and speed.

7. Limitations:

There are some drawbacks to our research, but we have done our best to resolve and monitor them. Also, weaknesses can be found in almost all scientific studies and are deemed very popular.

There are three drawbacks to our research:

1.When the modeling was converted to binary rather than the 1-4 Likert scale available in the results, some granularity of the responses was lost. More clarification may be needed when it comes to vaccine hesitancy, as those in groups 2 and 3 may be influenced by more data or better vaccine publicity/marketing.

2.Since the AUC was not large enough and the decision-making process is complicated,other features (e.g., political affiliation) outside of the survey must be analyzed to reliably predict people's reluctance to get the COVID-19 vaccine.

3.Only in one direction ANOVA we only got to investigate a single variable and a single component. We think it might tell us atleast one pair of means is substantially dissimilar when comparing the means of 3 or more classes, and that can't really know us which pair.

8. **Challenges:**

At the start of our project, we proposed to perform unsupervised learning using K-means clustering for our data, but at the time of clustering we couldn't yield the clusters as expected and showed poor results, because our data was labelled, categorized, discrete and large so we tried K-prototype clustering using K-modes even with this we could not execute the output. Later we tried Affinity Propagation as this was known best for categorized data but landed up with poor performance of the library i.e; the kernel died when tried to run the code, this was because the IDE (Jyputer notebook) we used for our project could not support it.

A Appendix - SQL Queries

```
CREATE TABLE Pulsesurvey (
   Birthyear YEAR,
   Gender varchar(3),
   Hispanic varchar(3),
   Race varchar(3),
   Education varchar(3),
   Maritalstatus varchar(3),
   Numkid varchar(3),
   Recvacc varchar(3),
   Doses varchar(3),
   Getvacc varchar(3),
```

```
    Workloss varchar(3),
    Expectloss varchar(3),
    Anywork varchar(3),
    Kindwork varchar(3),
    Expensdiff varchar(3),
    Anxious varchar(3),
    Worry varchar(3),
    Interest varchar(3),
    Down varchar(3),
    Hlthins1 varchar(3),
    Hlthins2 varchar(3),
    Hlthins3 varchar(3),
    Hlthins4 varchar(3),
    Hlthins5 varchar(3),
    Hlthins6 varchar(3),
    Hlthins7 varchar(3),
    Hlthins8 varchar(3),
    Prescript varchar(3),
    Mh_svcs varchar(3),
    Mh_notget varchar(3),
    Tenure varchar(3),
    Enroll1 varchar(3),
    Enroll2 varchar(3),
    Enroll3 varchar(3),
    Teach1 varchar(3),
    Teach2 varchar(3),
    Teach3 varchar(3),
    Teach4 varchar(3),
    Teach5 varchar(3),
    Income varchar(3)
    );

ALTER TABLE Pulsesurvey
DROP COLUMN Doses;

DELETE * FROM Pulsesurvey
WHERE Getvacc = -88;

DELETE * FROM Pulsesurvey
WHERE Getvacc = -99;

ALTER TABLE `Pulsesurvey` CHANGE `Maritalstatus` `Maritalstatus` VARCHAR(3) CHARACTER
SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';
```

```
ALTER TABLE `Pulsesurvey` CHANGE `Recvacc` `Recvacc` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Workloss` `Workloss` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Expectloss` `Expectloss` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Anywork` `Anywork` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Kindwork` `Kindwork` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Expensdiff` `Expensdiff` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Anxious` `Anxious` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Worry` `Worry` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Interest` `Interest` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Down` `Down` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Hlthins1` `Hlthins1` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Hlthins2` `Hlthins2` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Hlthins3` `Hlthins3` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Hlthins4` `Hlthins4` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Hlthins5` `Hlthins5` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';
```

```
ALTER TABLE `Pulsesurvey` CHANGE `Hlthins6` `Hlthins6` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Hlthins7` `Hlthins7` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Hlthins8` `Hlthins8` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Prescript` `Prescript` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Mh_svcs` `Mh_svcs` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Mh_notget` `Mh_notget` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Tenure` `Tenure` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Enroll1` `Enroll1` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Enroll2` `Enroll2` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Enroll3` `Enroll3` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Teach1` `Teach1` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Teach2` `Teach2` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Teach3` `Teach3` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Teach4` `Teach4` VARCHAR(3) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Teach5` `Teach5` VARCHAR(3) CHARACTER SET utf8mb4
```

```
COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

ALTER TABLE `Pulsesurvey` CHANGE `Income` `Income` VARCHAR(3) CHARACTER SET
utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT '-88';

CREATE TABLE Pulsesurvey_3_4 LIKE Pulsesurvey;

INSERT INTO Pulsesurvey_3_4 SELECT * FROM Pulsesurvey;

DELETE FROM `Pulsesurvey_3_4` WHERE Getvacc = 1;

DELETE FROM `Pulsesurvey_3_4` WHERE Getvacc = 2;

CREATE TABLE ‘Pulsesurvey_1_2’ LIKE Pulsesurvey;

INSERT INTO ‘Pulsesurvey_1_2’ SELECT * FROM Pulsesurvey;

DELETE FROM `Pulsesurvey_1_2` WHERE Getvacc = 3;

DELETE FROM `Pulsesurvey_1_2` WHERE Getvacc = 4;

CREATE TABLE Pulsesurvey_New LIKE Pulsesurvey;

INSERT INTO Pulsesurvey_New SELECT * FROM Pulsesurvey;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Recvacc;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Anywork;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Hlthins1;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Hlthins2;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Hlthins3;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Hlthins5;

ALTER TABLE Pulsesurvey_New
```

```
DROP COLUMN Hlthins6;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Hlthins7;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Enroll1;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Enroll2;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Enroll3;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Teach1;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Teach2;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Teach3;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Teach5;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Worry;

ALTER TABLE Pulsesurvey_New
DROP COLUMN Anxious;

CREATE TABLE Pulsesurvey_3_4_NoNull LIKE Pulsesurvey;

INSERT INTO Pulsesurvey_3_4_NoNull SELECT * FROM Pulsesurvey;

DELETE FROM Pulsesurvey_3_4_NoNull WHERE Getvacc = 1;

DELETE FROM Pulsesurvey_3_4_NoNull WHERE Getvacc = 2;

DELETE FROM Pulsesurvey_3_4_NoNull WHERE Income = -88;

DELETE FROM Pulsesurvey_3_4_NoNull WHERE Income = -99;
```

```
DELETE FROM Pulsesurvey_3_4_NoNull WHERE Education = -88;

DELETE FROM Pulsesurvey_3_4_NoNull WHERE Education = -99;

DELETE FROM Pulsesurvey_3_4_NoNull WHERE Race = -88;

DELETE FROM Pulsesurvey_3_4_NoNull WHERE Race = -99;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Gender;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Hispanic;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN MaritalStatus;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Numkid;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Recvacc;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Workloss;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Expectloss;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Anywork;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Kindwork;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Expensdiff;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Anxious;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Worry;
```

```
ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Interest;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Down;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Hlthins1;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Hlthins2;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Hlthins3;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Hlthins4;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Hlthins5;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Hlthins6;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Hlthins7;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Hlthins8;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Prescript;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Mh_svcs;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Mh_notget;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Tenure;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Enroll1;
```

```
ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Enroll2:

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Enroll3;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Teach1;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Teach2;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Teach3;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Teach4;

ALTER TABLE Pulsesurvey_3_4_NoNull
DROP COLUMN Teach5;

CREATE TABLE Pulsesurvey_New_NoNull LIKE Pulsesurvey;

INSERT INTO Pulsesurvey_New_NoNull SELECT * FROM Pulsesurvey;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Gender;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Hispanic;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN MaritalStatus;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Numkid;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Recvacc;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Workloss;
```

```
ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Expectloss;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Anywork;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Kindwork;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Expensdiff;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Anxious;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Worry;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Interest;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Down;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Hlthins1;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Hlthins2;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Hlthins3;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Hlthins4;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Hlthins5;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Hlthins6;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Hlthins7;
```

```
ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Hlthins8;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Prescript;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Mh_svcs;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Mh_notget;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Tenure;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Enroll1;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Enroll2:

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Enroll3;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Teach1;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Teach2;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Teach3;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Teach4;

ALTER TABLE Pulsesurvey_New_NoNull
DROP COLUMN Teach5;

DELETE FROM Pulsesurvey_New_NoNull WHERE Income = -88;

DELETE FROM Pulsesurvey_New_NoNull WHERE Income = -99;
```

References