Final Project - P47

In [2]:
```
import numpy as np
import pandas as pd

import plotly.express as px
import altair as alt
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import matplotlib.pyplot as plt

from datetime import datetime

import geopandas as gpd
from urllib.request import urlopen
import json
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master/geojson-coun
    counties = json.load(response)

from dash import Dash, dcc, html
```

In [3]:
```
df = pd.read_csv("owid-covid-data.csv")

df = df.drop(['new_cases', 'new_deaths', 'new_cases_per_million', 'new_deaths_per_m
df = df.drop(['excess_mortality_cumulative', 'excess_mortality', 'excess_mortality_
df = df.drop(['new_cases_smoothed_per_million', 'total_deaths_per_million', 'new_de
df = df.drop(['hosp_patients_per_million', 'weekly_icu_admissions_per_million', 'we
df = df.drop(['new_tests_per_thousand', 'new_tests_smoothed_per_thousand', 'new_tes
df = df.drop(['people_vaccinated_per_hundred', 'people_fully_vaccinated_per_hundred
df = df.drop(['new_people_vaccinated_smoothed_per_hundred', 'weekly_icu_admissions'
df = df.drop(['icu_patients', 'weekly_hosp_admissions', 'new_people_vaccinated_smoo

df = df.loc[~df['location'].isin(['World', 'European Union (27)', 'High-income coun
df = df[df['continent'] != ""]
df['total_cases_percent'] = df['total_cases']/df['population'] * 100.0
df['date'] = pd.to_datetime(df['date'])
df['month'] = df['date'].dt.to_period('M').astype(str)

df['timestamp'] = df['date'].apply(lambda x: x.timestamp())
df['people_vaccinated'].fillna(0)
df['vaccination_percent'] = df['people_vaccinated']/df['population'] * 100.0
df['vaccination_percent'] = [100.0 if x > 100 else x for x in df['vaccination_perce
```

In [4]:
```
# Sanity Checks
df.shape
```

Out[4]: (404591, 37)

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

Out[5]:

	iso_code	continent	location	date	total_cases	new_cases_smoothed	total_deaths
0	AFG	Asia	Afghanistan	2020-01-05	0.0	NaN	0.0
1	AFG	Asia	Afghanistan	2020-01-06	0.0	NaN	0.0
2	AFG	Asia	Afghanistan	2020-01-07	0.0	NaN	0.0
3	AFG	Asia	Afghanistan	2020-01-08	0.0	NaN	0.0
4	AFG	Asia	Afghanistan	2020-01-09	0.0	NaN	0.0

5 rows × 37 columns

In [6]:
```
df.tail(5)
```

Out[6]:

	iso_code	continent	location	date	total_cases	new_cases_smoothed	total_dea
429430	ZWE	Africa	Zimbabwe	2024-07-31	266386.0	0.0	574
429431	ZWE	Africa	Zimbabwe	2024-08-01	266386.0	0.0	574
429432	ZWE	Africa	Zimbabwe	2024-08-02	266386.0	0.0	574
429433	ZWE	Africa	Zimbabwe	2024-08-03	266386.0	0.0	574
429434	ZWE	Africa	Zimbabwe	2024-08-04	266386.0	0.0	574

5 rows × 37 columns

In [7]:
```
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 404591 entries, 0 to 429434
Data columns (total 37 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   iso_code                     404591 non-null  object
 1   continent                    402910 non-null  object
 2   location                     404591 non-null  object
 3   date                         404591 non-null  datetime64[ns]
 4   total_cases                  393390 non-null  float64
 5   new_cases_smoothed           390570 non-null  float64
 6   total_deaths                 393390 non-null  float64
 7   new_deaths_smoothed          391019 non-null  float64
 8   reproduction_rate            183741 non-null  float64
 9   hosp_patients                40656 non-null   float64
 10  total_tests                  79387 non-null   float64
 11  new_tests_smoothed           103965 non-null  float64
 12  positive_rate                95927 non-null   float64
 13  total_vaccinations           71437 non-null   float64
 14  people_vaccinated            67152 non-null   float64
 15  people_fully_vaccinated      64259 non-null   float64
 16  total_boosters               40720 non-null   float64
 17  new_vaccinations_smoothed    181089 non-null  float64
 18  stringency_index             196190 non-null  float64
 19  population_density           358808 non-null  float64
 20  median_age                   332979 non-null  float64
 21  aged_65_older                321586 non-null  float64
 22  aged_70_older                329631 non-null  float64
 23  gdp_per_capita               326608 non-null  float64
 24  extreme_poverty              210312 non-null  float64
 25  cardiovasc_death_rate        327181 non-null  float64
 26  diabetes_prevalence          344227 non-null  float64
 27  female_smokers               245481 non-null  float64
 28  male_smokers                 242133 non-null  float64
 29  hospital_beds_per_thousand   289005 non-null  float64
 30  life_expectancy              388615 non-null  float64
 31  human_development_index      317443 non-null  float64
 32  population                   404591 non-null  int64
 33  total_cases_percent          393390 non-null  float64
 34  month                        404591 non-null  object
 35  timestamp                    404591 non-null  float64
 36  vaccination_percent          67152 non-null   float64
dtypes: datetime64[ns](1), float64(31), int64(1), object(4)
memory usage: 117.3+ MB
```

Out[7]:

	date	total_cases	new_cases_smoothed	total_deaths	new_deaths_sı
count	404591	3.933900e+05	3.905700e+05	3.933900e+05	39101
mean	2022-04-18 10:07:03.983727616	1.848789e+06	2.024980e+03	2.042221e+04	1
min	2020-01-01 00:00:00	0.000000e+00	0.000000e+00	0.000000e+00	
25%	2021-02-28 00:00:00	5.575000e+03	0.000000e+00	3.700000e+01	
50%	2022-04-17 00:00:00	5.145000e+04	9.000000e+00	6.700000e+02	
75%	2023-06-07 00:00:00	5.751620e+05	2.097100e+02	7.465000e+03	
max	2024-08-14 00:00:00	1.034368e+08	5.782211e+06	1.193165e+06	681
std	NaN	7.859334e+06	3.240023e+04	8.245260e+04	11

8 rows × 33 columns

Visualizations

In [9]:
```
# Figure 1

monthly_df = df.groupby(['month', 'location'], as_index=False)['total_cases_percent

fig1 = px.choropleth (
    monthly_df,
    locations =  'location',
    locationmode='country names',
    color = 'total_cases_percent',
    hover_name = 'location',
    title = 'COVID-19 Total Cases Percent by Month',
    animation_frame = 'month',
    color_continuous_scale = px.colors.sequential.Jet,
    range_color=[0, df['total_cases_percent'].max()],
    width = 1000,
    height = 600
)
fig1.update_traces(
    customdata=df[['location', 'month', 'total_cases', 'total_cases_percent']].valu
    hovertemplate="<b>%{customdata[0]}</b><br>%{customdata[1]}<br>Total Cases in Co
)
fig1.update_layout(
    coloraxis_colorbar=dict(
        title="Infection Percent"
```

```
    )
)
```

In [10]:

```
# Figure 2

monthly = df.sort_values('date').groupby(['location', 'month']).last()
monthly = monthly[~monthly['vaccination_percent'].isna()]
monthly = monthly[~monthly['continent'].isna()]
monthly.reset_index(inplace=True)

def fig2():

    alt.data_transformers.disable_max_rows()

    min_year = monthly['timestamp'].min()
    max_year = monthly['timestamp'].max()
    step_size = 30 * 24 * 60 * 60

    slider = alt.binding_range(min = min_year, max = max_year, step = step_size, na

    op_var = alt.param(value = min_year, bind=slider)

    selection = alt.selection_point(fields=['location'], name='Selected', empty='al

    chart1 = alt.Chart(monthly).mark_circle().encode (
        x = alt.X('total_cases_percent:Q', title = 'Percent of Population Previousl
        y = alt.Y('vaccination_percent:Q', title = 'Percent of Population Vaccinate
        size=alt.Size('population:Q', title='Population Size', scale=alt.Scale(rang
        color=alt.condition(selection, 'continent:N', alt.value('lightgray'), legen
        tooltip = [
            alt.Tooltip('location:N', title='Location'),
            alt.Tooltip('date:T', title='Date', format='%Y-%m-%d'),
            alt.Tooltip('vaccination_percent:Q', title='Vaccination Percent', forma
            alt.Tooltip('total_cases_percent:Q', title='Previous Infection Percent'
        ]
    ).add_params (
        op_var, selection
    ).transform_filter (
        (alt.datum.timestamp >= op_var) & (alt.datum.timestamp < op_var + step_size
    ).properties(
        title="Percentage of Country Vaccinated vs. Infected"
    )


    chart2a = alt.Chart(monthly).mark_line(color = 'blue', point=alt.OverlayMarkDef
        x=alt.X('date:T', title='Date'),
        y=alt.Y('vaccination_percent:Q', title='Percent of Population'),
        tooltip = ['location', 'date']
    ).transform_filter(
        selection & (alt.datum.location != "")
    )

    chart2b = alt.Chart(monthly).mark_line(color = 'orange', point=alt.OverlayMarkD
        x=alt.X('date:T', title='Date'),
        y=alt.Y('total_cases_percent:Q'),
```

```
        tooltip = ['location', 'date']
    ).transform_filter(
        selection & (alt.datum.location != "")
    )

    categories = pd.DataFrame({
        'category': ['Vaccination Percent', 'Previous Infection Percent'],
        'color': ['blue', 'orange']
    })
    legend = alt.Chart(categories).mark_point().encode(
        y=alt.Y('category:N', title=None, axis=alt.Axis(labels=True, domain=False)),
        color=alt.Color('color:N', scale=None)
    )
    chart2 = alt.layer(chart2a, chart2b).properties(
        title="Percent of Country with COVID",
    ) | legend

    return (chart1 | chart2).to_html()
```

In [11]:
```
# Figure 3

df_us = df[df['location'] == 'United States']
df_us = df_us[['date', 'total_cases', 'stringency_index', 'reproduction_rate', 'new

fig3b = go.Figure()
fig3b.add_trace(go.Scatter(x=df_us['date'], y=df_us['new_cases_smoothed'], mode='li
                           line=dict(color='purple'),
                           hovertemplate='<br><b>New Cases: </b>%{y} units<extra></ex
fig3b.add_trace(go.Scatter(x=df_us['date'], y=df_us['new_vaccinations_smoothed'], m
                           yaxis = 'y2', line=dict(color='orange'),
                           hovertemplate = '<br><b>New Vaccinations: </b>%{y} units<e
fig3b.add_trace(go.Scatter(x=df_us['date'], y=df_us['new_deaths_smoothed'], mode='l
                           line=dict(color='black'),
                           customdata = df_us['total_cases'],
                           hovertemplate = '<br><b>New Deaths: </b>%{y} units<br><br>
                           visible = 'legendonly'))
fig3b.add_trace(go.Scatter(x=df_us['date'], y=df_us['new_deaths_smoothed'], mode='l
                           yaxis = 'y3',
                           line=dict(color='black'),
                           customdata = df_us['total_cases'],
                           hovertemplate = '<br><b>New Deaths: </b>%{y} units<br><br>
                           visible = 'legendonly'))
fig3b.add_trace(go.Scatter(x=df_us['date'], y=df_us['reproduction_rate'], mode='lin
                           yaxis = 'y4', line=dict(color='blue'),
                           hovertemplate='<b>Reproduction Rate: </b>%{y} units<extra>

fig3b.update_layout(
    title="Unites States COVID-19 Reproduction Rate and Cases/Vaccinations/Deaths",
    xaxis=dict(title="Date"),
    yaxis=dict(title="New Cases",
        ),
    yaxis2=dict(
        title="New Vaccinations",
        overlaying='y',
        side='right',
    ),
```

```
    yaxis3=dict(
        title="",
        overlaying="y",
        showticklabels=False,
        showgrid=False,
        zeroline=False
    ),
    yaxis4=dict(
        title="",
        showticklabels= False,
        showgrid= False,
        zeroline=False,
        overlaying="y",
    ),
    hovermode='x unified',
    )
```

In [12]:

```
# Figure 4

df_vax = pd.read_csv('vaccinations.csv')
df_cases = pd.read_csv('new_cases.csv')

#make merged dataset
df_vax['date'] = pd.to_datetime(df_vax['date'])
df_cases['date'] = pd.to_datetime(df_cases['date'])

common_locations = set(df_vax["location"]).intersection(set(df_cases.columns) - {"d

#only include common locations
df_vax_filtered = df_vax[df_vax["location"].isin(common_locations)]

#reshape df_cases
df_cases_reshaped = df_cases.melt(id_vars=["date"], var_name="location", value_name
df_cases_filtered = df_cases_reshaped[df_cases_reshaped["location"].isin(common_loc

merged_df = pd.merge(df_vax_filtered, df_cases_filtered, on=["date", "location"], h

#filter to US + calc proportions
merged_df["partially_vaccinated_per_hundred"] = merged_df["people_vaccinated_per_hu

#select country (maybe include a user input so that people can view the vax trends
singleCountry = "United States"
country_df = merged_df[merged_df["location"] == singleCountry]

filtered_df = country_df[(country_df['new_cases'] != 0) &
                         country_df['new_cases'].notna() &
                         country_df['partially_vaccinated_per_hundred'].notna() &
                         country_df['people_fully_vaccinated_per_hundred'].notna()]

#plot
fig4 = make_subplots(rows=2, cols=1, shared_xaxes=True, vertical_spacing=0.2,
                     subplot_titles=("COVID-19 Infection Rates", "Vaccination Trend
                     )

#cases histogram
```

```
fig4.add_trace(go.Bar(x=filtered_df["date"],
                      y=filtered_df["new_cases"],
                      marker_color='red',
                      name="New COVID-19 Cases",
                      hoverinfo="x+y",
                      hovertemplate=None),
                      row=1, col=1)



#vax trends bar chart
fig4.add_trace(go.Bar(x=filtered_df["date"],
                      y=filtered_df["partially_vaccinated_per_hundred"],
                      marker_color='orange',
                      name="Partially Vaccinated (% of population)",
                      hoverinfo="x+y",
                      hovertemplate=None),
                      row=2, col=1)

fig4.add_trace(go.Bar(x=filtered_df["date"],
                      y=filtered_df["people_fully_vaccinated_per_hundred"],
                      marker_color='blue',
                      name="Fully Vaccinated (% of population)",
                      hoverinfo="x+y",
                      hovertemplate=None,
                      base=filtered_df["partially_vaccinated_per_hundred"]),
                      row=2, col=1)


fig4.update_layout(
    title_text=f"COVID-19 Infection Rates and Vaccination Trends in {singleCountry}
    xaxis_title="Date",
    yaxis_title="New Cases",
    yaxis2_title="Percentage of Population Vaccinated",
    barmode='stack',
    hovermode="x unified",
    height=700
)

fig4.show()
```

In [13]:

```
# Figure 5

df_income = pd.read_csv("med_income_us_counties.csv")
df_covid_deaths_2023 = pd.read_csv("us-counties-2023.csv", dtype={"fips": str})
df_county_pop = pd.read_csv('us_county_pops.csv', dtype={"fips": str})

#add a column of zipcode values (string format) (derive this from column placeDcid
df_income['fips'] = df_income['placeDcid'].str.replace(r'geoId/', '', regex=True)
df_county_pop['fips'] = df_county_pop['placeDcid'].str.replace(r'geoId/', '', regex

#filter data to only include 2023
df_2023 = df_income[df_income['year'] == 2023]

#derive new df frm df_covid_deaths_2023
```

```
df_clean_covid_deaths = df_covid_deaths_2023.dropna()

#make total_deaths column (per county)
total_deaths_dict = df_clean_covid_deaths.groupby("fips")["deaths"].max().to_dict()
df_total_deaths = pd.DataFrame.from_dict(total_deaths_dict, orient='index', columns
df_total_deaths.rename(columns={'index': 'fips'}, inplace=True)

#make total_cases column (per county)
total_cases_dict = df_clean_covid_deaths.groupby("fips")["cases"].max().to_dict()
df_total_cases = pd.DataFrame.from_dict(total_cases_dict, orient='index', columns=[
df_total_cases.rename(columns={'index': 'fips'}, inplace=True)

#rename column in df_county_pop
df_county_pop.rename(columns={'Value:Count_Person': 'population'}, inplace=True)

#merge all the df's
df_final = df_total_deaths.merge(df_total_cases, on='fips', how='left')

#NEW df with both income df and covid deaths df + county_pops
df_both = df_2023.merge(df_final, on="fips")
df_both = df_both.merge(df_county_pop[['fips', 'population']], on='fips', how='left

#calc normalized death:population ratio
df_both['deaths_pop_ratio'] = (df_both['totalDeaths'] / df_both['population'])

#norm
df_both["income_norm"] = df_both["median_income_household"] / df_both["median_incom
df_both["covid_death_norm"] = df_both["deaths_pop_ratio"] / df_both["deaths_pop_rat

#norm
df_both["income_vs_death"] = df_both["income_norm"] - df_both["covid_death_norm"] #

#plot
fig5 = px.choropleth_mapbox(
    df_both, geojson=counties, locations='fips',
    color="income_vs_death",
    color_continuous_scale="RdBu",
    custom_data=['countyName', 'median_income_household', 'population', 'totalDeath
    mapbox_style="carto-positron",
    center={"lat": 38.0902, "lon": -95.7129},
    zoom=2.8
)

fig5.update_traces(
    hovertemplate="<br>".join([
        "County: %{customdata[0]}",
        "Median Household Income (2023): $%{customdata[1]:,.0f}",
        "County Population (2023): %{customdata[2]:,}",
        "Total Deaths in County (2023): %{customdata[3]:,}"
    ])
)

fig5.update_layout(
    title = 'Total Deaths vs. Median Household Incomes Across U.S. Counties (2023)'
    height = 500,
    width = 875
```

```
)

fig5.show()
```

In [14]:
```
df_covid_deaths_2023[df_covid_deaths_2023['county'] == 'Autauga']
```

Out[14]:

	date	county	state	fips	cases	deaths
0	1/1/2023	Autauga	Alabama	01001	18961	230.0
3255	1/2/2023	Autauga	Alabama	01001	18961	230.0
6510	1/3/2023	Autauga	Alabama	01001	18961	230.0
9764	1/4/2023	Autauga	Alabama	01001	19205	230.0
13016	1/5/2023	Autauga	Alabama	01001	19205	230.0
...	...	...	...	...	...	...
250726	3/19/2023	Autauga	Alabama	01001	19799	234.0
253983	3/20/2023	Autauga	Alabama	01001	19799	234.0
257239	3/21/2023	Autauga	Alabama	01001	19799	234.0
260494	3/22/2023	Autauga	Alabama	01001	19812	235.0
263752	3/23/2023	Autauga	Alabama	01001	19812	235.0

82 rows × 6 columns

In [15]:
```
df_covid_deaths_2023[df_covid_deaths_2023['county'] == 'Autauga']
```

Out[15]:

	date	county	state	fips	cases	deaths
0	1/1/2023	Autauga	Alabama	01001	18961	230.0
3255	1/2/2023	Autauga	Alabama	01001	18961	230.0
6510	1/3/2023	Autauga	Alabama	01001	18961	230.0
9764	1/4/2023	Autauga	Alabama	01001	19205	230.0
13016	1/5/2023	Autauga	Alabama	01001	19205	230.0
...	...	...	...	...	...	...
250726	3/19/2023	Autauga	Alabama	01001	19799	234.0
253983	3/20/2023	Autauga	Alabama	01001	19799	234.0
257239	3/21/2023	Autauga	Alabama	01001	19799	234.0
260494	3/22/2023	Autauga	Alabama	01001	19812	235.0
263752	3/23/2023	Autauga	Alabama	01001	19812	235.0

82 rows × 6 columns

Machine Learning Models

In [16]:

```
#logistic regression model: do cases decrease after 50% vaccination rate?

df['people_vaccinated'] = df['people_vaccinated'].fillna(0)
df['vaccination_percent'] = df['people_vaccinated'] / df['population'] * 100
df['vaccination_percent'] = df['vaccination_percent'].clip(upper=100)

#group by location + find the first date with >50% vaccinated
fifty_vax_date = df[df['vaccination_percent'] >= 50].groupby('location')['date'].mi
fifty_vax_date.columns = ['location', 'fifty_vax_date']

df = df.merge(fifty_vax_date, on='location', how='left')

df['new_cases_smoothed'] = df['new_cases_smoothed'].fillna(0)

def determine_case_decrease(group): #returns a scalar for each location
    vax_date = group['fifty_vax_date'].iloc[0]
    if pd.isna(vax_date):
        return np.nan

    before = group[(group['date'] >= vax_date - pd.Timedelta(days=30)) & (group['da
    after = group[(group['date'] > vax_date) & (group['date'] <= vax_date + pd.Time

    before_avg = before['new_cases_smoothed'].mean()
    after_avg = after['new_cases_smoothed'].mean()

    return int(after_avg < before_avg)

#apply *once per location* + merge
```

```
case_trend_by_country = df.groupby('location').apply(determine_case_decrease).reset
case_trend_by_country.columns = ['location', 'cases_decreased_after_50']

df = df.merge(case_trend_by_country, on='location', how='left')
```

```
C:\Users\Aurea\AppData\Local\Temp\ipykernel_23892\1386337242.py:29: DeprecationWarni
ng:

DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecate
d, and in a future version of pandas the grouping columns will be excluded from the
operation. Either pass `include_groups=False` to exclude the groupings or explicitly
select the grouping columns after groupby to silence this warning.
```

In [17]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

model_df = df.dropna(subset=['cases_decreased_after_50'])

features = ['vaccination_percent', 'population_density', 'stringency_index', 'media
model_df = model_df.dropna(subset=features)

X = model_df[features]
y = model_df['cases_decreased_after_50']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta


model = LogisticRegression()
model.fit(X_train, y_train)


y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
df.info()
```

```
              precision    recall  f1-score   support

         0.0       0.69      0.83      0.75      2166
         1.0       0.72      0.54      0.62      1786

    accuracy                           0.70      3952
   macro avg       0.71      0.69      0.69      3952
weighted avg       0.70      0.70      0.69      3952

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 404591 entries, 0 to 404590
Data columns (total 39 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   iso_code                    404591 non-null  object
 1   continent                   402910 non-null  object
 2   location                    404591 non-null  object
 3   date                        404591 non-null  datetime64[ns]
 4   total_cases                 393390 non-null  float64
 5   new_cases_smoothed          404591 non-null  float64
 6   total_deaths                393390 non-null  float64
 7   new_deaths_smoothed         391019 non-null  float64
 8   reproduction_rate           183741 non-null  float64
 9   hosp_patients               40656 non-null   float64
 10  total_tests                 79387 non-null   float64
 11  new_tests_smoothed          103965 non-null  float64
 12  positive_rate               95927 non-null   float64
 13  total_vaccinations          71437 non-null   float64
 14  people_vaccinated           404591 non-null  float64
 15  people_fully_vaccinated     64259 non-null   float64
 16  total_boosters              40720 non-null   float64
 17  new_vaccinations_smoothed   181089 non-null  float64
 18  stringency_index            196190 non-null  float64
 19  population_density          358808 non-null  float64
 20  median_age                  332979 non-null  float64
 21  aged_65_older               321586 non-null  float64
 22  aged_70_older               329631 non-null  float64
 23  gdp_per_capita              326608 non-null  float64
 24  extreme_poverty             210312 non-null  float64
 25  cardiovasc_death_rate       327181 non-null  float64
 26  diabetes_prevalence         344227 non-null  float64
 27  female_smokers              245481 non-null  float64
 28  male_smokers                242133 non-null  float64
 29  hospital_beds_per_thousand  289005 non-null  float64
 30  life_expectancy             388615 non-null  float64
 31  human_development_index     317443 non-null  float64
 32  population                  404591 non-null  int64
 33  total_cases_percent         393390 non-null  float64
 34  month                       404591 non-null  object
 35  timestamp                   404591 non-null  float64
 36  vaccination_percent         404591 non-null  float64
 37  fifty_vax_date              258952 non-null  datetime64[ns]
 38  cases_decreased_after_50    258952 non-null  float64
dtypes: datetime64[ns](2), float64(32), int64(1), object(4)
memory usage: 120.4+ MB
```

```
c:\Users\Aurea\miniconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:465: C
onvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

In [18]:
```
for feature, coef in zip(features, model.coef_[0]):
    print(f"{feature}: {coef:.3f}")
```

```
vaccination_percent: 0.007
population_density: 0.005
stringency_index: -0.002
median_age: 0.026
positive_rate: -4.635
hosp_patients: 0.000
```

In [19]:
```
#confusion matrix
#import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

y_pred = model.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
labels = model.classes_ if hasattr(model, "classes_") else np.unique(y_test)

confusion_fig = go.Figure(data=go.Heatmap(
    z=cm,
    x=labels,  #predicted
    y=labels,  #true
    colorscale='Oranges',
    hoverongaps=False,
    text=cm,
    texttemplate="%{text}",
    showscale=True
))

confusion_fig.update_layout(
    title="Confusion Matrix",
    xaxis_title="Predicted Label",
    yaxis_title="True Label",
    xaxis=dict(tickmode='array', tickvals=list(range(len(labels))), ticktext=labels
    yaxis=dict(tickmode='array', tickvals=list(range(len(labels))), ticktext=labels
    width=500,
    height=500,
    margin=dict(l=40, r=40, t=40, b=40)
)
```

In [20]:
```
#visualize ROC Curve and AUC
from sklearn.metrics import roc_curve, auc
```

```
y_proba = model.predict_proba(X_test)[:, 1]

fpr, tpr, thresholds = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

roc_fig = go.Figure()

roc_fig.add_trace(go.Scatter(x=fpr, y=tpr, mode='lines', name=f'ROC curve (AUC = {r
                             line=dict(color='darkorange', width=2)))
roc_fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1], mode='lines', name='Random guess',
                             line=dict(color='navy', dash='dash')))

roc_fig.update_layout(
    title='Receiver Operating Characteristic (ROC) Curve',
    xaxis_title='False Positive Rate',
    yaxis_title='True Positive Rate',
    legend=dict(x=0.6, y=0.05),
    width=600,
    height=600,
    margin=dict(l=40, r=40, t=40, b=40)
)
```

References

Bureau, U. C. (2024, October 7). American Community Survey data via FTP. Census.gov. https://www.census.gov/programs-surveys/acs/data/data-via-ftp.html

John Hopkins University of Medicine. (2023, March 16). Mortality analyses. Johns Hopkins Coronavirus Resource Center. https://coronavirus.jhu.edu/data/mortality

Centers for Disease Control and Prevention. (2025, April 18). CDC Covid Data tracker. Centers for Disease Control and Prevention. https://covid.cdc.gov/covid-data-tracker/#datatracker-home

Mathieu, E., Ritchie, H., Rodés-Guirao, L., Appel, C., Gavrilov, D., Giattino, C., Hasell, J., Macdonald, B., Dattani, S., Beltekian, D., Ortiz-Ospina, E., & Roser, M. (2020). COVID-19 pandemic. Our World in Data. Retrieved from https://ourworldindata.org/coronavirus

World Health Organization. (2024, December 2). Covid-19 variants | who covid-19 dashboard. World Health Organization. https://data.who.int/dashboards/covid19/variants

Centers for Disease Control and Prevention. (2024, July 8). CDC Museum Covid-19 Timeline. Centers for Disease Control and Prevention. https://www.cdc.gov/museum/timeline/covid19.html

In [21]:

```
app = Dash(__name__)

app.layout = html.Div([
    html.H1(
        "COVID-19 Vaccine Effectivity and Coverage",
```

```
        style={'textAlign': 'center', 'paddingTop': 50}
    ),
    html.Div("Kelly Hu and Aurea Le",
            style={
                'textAlign': 'center',
                'paddingBottom': 20
            }),
    html.Div(
        '''Goal: We aim to investigate how vaccine coverage varies geographically a
        there is a relationship between the vaccination rate in a given area and a
        COVID-19 cases. We hypothesize that a significant reduction in COVID-19 cas
        once at least 50% of the population has been vaccinated at least once.''',
        style={'textAlign': 'center', 'padding': 10, 'marginLeft': 50, 'marginRight
    ),

    html.Div([
        html.H3("Figure 1: Explore the geographic spread of COVID across the globe
                style={'paddingLeft': 20}),
        html.Div(
            "Visualizes the percent of the population infected by COVID-19 for each
            "updated by month. Each country is color-coded based on the percent of
            "that has been infected by COVID-19 (including both current and past ca
            "color scale ranging from blue to red.",
            style={'margin': 20}
        ),
        dcc.Graph(figure=fig1)
    ], style={'margin': 50, 'border': 'dotted'}),

    html.Div([
        html.H3("Figure 2: Examine the relationship between population vaccination
                "the percentage of new cases, by month",
                style={'paddingLeft': 20}),
        html.Div(
            "Explore the dynamics between vaccination rates and infection rates acr
            "countries, both on a snapshot basis (scatter plot, with the slider org
            "represented in timestamp) and over time (line graphs). The size of the
            "scatter plot is based on the country\'s population size.",
            style={'margin': 20}
        ),
        html.Iframe(
            srcDoc=fig2(),
            style={"border": "none", "width": "100%", "height": "475px", 'paddingLe
        )
    ], style={'margin': 50, 'border': 'dotted'}),

    html.Div([
        html.H3("Figure 3: Examine trends in the COVID-19 Reproduction Rate, New Ca
                "New Vaccinations, and New Deaths over time in the US",
                style={'paddingLeft': 20}),
        html.Div(
            "Visualize how COVID-19 cases, deaths, and vaccinations evolved over ti
            "to changes in the reproduction rate of COVID-19 in the United States.
            "is plotted with a line and uses different axes to make it easy to comp
            "disease reproduction rate, new cases, vaccinations, and deaths.",
            style={'margin': 20}
        ),
```

```
        dcc.Graph(figure=fig3b)
    ], style={'margin': 50, 'border': 'dotted'}),

    html.Div([
        html.H3("Figure 4: Analyze the effect of partial vaccination (at least one
                "vaccination on COVID-19 case trends",
                style={'paddingLeft': 20}),
        html.Div(
            "Compare the progression of COVID-19 infections with the rates of vacci
            "population. The x-axis represents time (date), while the y-axis on the
            "new cases, and the y-axis on the bottom shows the percentage of the po
            "The stacked bars visualize both partial and full vaccination rates.",
            style={'margin': 20}
        ),
        dcc.Graph(figure=fig4)
    ], style={'margin': 50, 'border': 'dotted'}),

    html.Div([
        html.H3("Figure 5: Examine the relationship between poverty rates and COVID
                style={'paddingLeft': 20}),
        html.Div(
            "The colors on the counties in Figure 5 represent the difference betwee
            "median household income and the normalized total COVID-19 deaths for e
            "scale is used to visually distinguish these differences, with blue for
            "(compared to deaths) and red for higher deaths (compared to income). T
            "helps to identify regional patterns related to income and death rates.
            style={'margin': 20}
        ),
        dcc.Graph(figure=fig5)
    ], style={'margin': 50, 'border': 'dotted'}),
    #Machine learning portion
    html.Div([
        html.H3("Confusion Matrix (Logistic Regression Model)",
                style={'paddingLeft': 20}),
        html.Div(
            "We ran a logistic regression model in order to answer whether COVID-19
            style={'margin': 20}
        ),
        dcc.Graph(figure=confusion_fig)
    ], style={'margin': 50, 'border': 'dotted'}),

    html.Div([
        html.H3("ROC Curve + AUC Score",
                style={'paddingLeft': 20}),
        html.Div(
            "We also charted an ROC curve in order to visually evaluate the perform
            style={'margin': 20}
        ),
        dcc.Graph(figure=roc_fig)
    ], style={'margin': 50, 'border': 'dotted'}),

    html.Div([
        html.H3("References", style={'paddingLeft': 20}),
        html.Div([
            html.P([
                "Bureau, U. C. (2024, October 7). American Community Survey data vi
```

```
                html.A("Census.gov",
                       href="https://www.census.gov/programs-surveys/acs/data/data-
                       target="_blank")
            ]),
            html.P([
                "Centers for Disease Control and Prevention. (2024, July 8). ",
                html.A("CDC Museum COVID-19 Timeline",
                       href="https://www.cdc.gov/museum/timeline/covid19.html",
                       target="_blank")
            ]),
            html.P([
                "Centers for Disease Control and Prevention. (2025, April 18). ",
                html.A("CDC Covid Data tracker",
                       href="https://covid.cdc.gov/covid-data-tracker/#datatracker-
                       target="_blank")
            ]),
            html.P([
                "John Hopkins University of Medicine. (2023, March 16). Mortality a
                html.A("Johns Hopkins Coronavirus Resource Center",
                       href="https://coronavirus.jhu.edu/data/mortality",
                       target="_blank")
            ]),
            html.P([
                "Mathieu, E., Ritchie, H., Rodés-Guirao, L., Appel, C., Gavrilov, D
                "Hasell, J., Macdonald, B., Dattani, S., Beltekian, D., Ortiz-Ospin
                html.A("Our World in Data",
                       href="https://ourworldindata.org/coronavirus",
                       target="_blank")
            ]),
            html.P([
                "World Health Organization. (2024, December 2). ",
                html.A("WHO COVID-19 Dashboard",
                       href="https://data.who.int/dashboards/covid19/variants",
                       target="_blank")
            ]),
            html.P([
                "United States Census Bureau. (n.d.). ",
                html.A("Population Estimates Tables",
                       href="https://www2.census.gov/programs-surveys/popest/tables
                       target="_blank")
            ])
        ], style={'margin': 20})
    ], style={'margin': 50, 'border': 'dotted'})

], style={'backgroundColor': 'white', 'paddingBottom': 50})

# app.run(jupyter_mode="tab", port=8051)
app.run(port=8051)
```

COVID-19 Vaccine Effectivity and Coverage

Kelly Hu and Aurea Le

Goal: We aim to investigate how vaccine coverage varies geographically and whether there is a relationship between the vaccination rate in a given area and a decrease in COVID-19 cases. We hypothesize that a significant reduction in COVID-19 cases occurs once at least 50% of the population has been vaccinated at least once.

Figure 1: Explore the geographic spread of COVID across the globe over time

Visualizes the percent of the population infected by COVID-19 for each country, updated by month. Each country is color-coded based on the percent of the population that has been infected by COVID-19 (including both current and past cases), with a color scale ranging from blue to red.

COVID-19 Total Cases Percent by Month