# Technical Report

July 3, 2024

## 1 Technical Report: Confirmed Positive Cases of COVID-19 in Ontario Analysis

### 1. Introduction

The objective of this analysis is to understand the patterns and trends in the COVID-19 case data for Ontario, Canada. The dataset contains detailed information about confirmed positive cases, including demographic information, dates of reporting and testing, and geographical details of public health units. By exploring and analyzing this data, we aim to gain insights into the spread of COVID-19 across different regions and over time. This report will detail the data preprocessing steps, exploratory data analysis, model selection and evaluation, and the key findings from our analysis.

**Brief description of the dataset** COVID-19 dataset containing information about COVID-19 cases in Ontario, Canada. The dataset includes several key attributes such as the dates related to the case (Accurate Episode Date, Case Reported Date, Test Reported Date, Specimen Date), demographic information (Age Group, Client Gender), and health unit details. The primary objective of this analysis is to clean the data and explore it through visualizations to gain insights into the COVID-19 cases.

[1]:
```
import pandas as pd

# Path to the downloaded CSV file
data_path = r"C:\Users\ENG WAHEED\Downloads\Confirmed Positive Cases of␣
 ↪COVID-19 in Ontario.csv"

# Load the dataset into a DataFrame
df = pd.read_csv(data_path)

# Display the first few rows of the DataFrame
print(df.head())
```

```
   Row_ID Accurate_Episode_Date Case_Reported_Date Test_Reported_Date  \
0       1            1934-09-28         2022-09-29         2022-09-29
1       2            1989-02-21         2022-11-08         2022-11-07
2       3            2000-03-01         2022-01-30                NaN
3       4            2002-07-06         2022-07-06         2022-07-07
4       5            2002-08-08         2022-08-15         2022-08-15
```

```
  Specimen_Date Age_Group Client_Gender Outcome1  Reporting_PHU_ID  \
0    2022-09-27       <20        FEMALE      NaN              2262
1    2022-11-06       <20        FEMALE      NaN              2270
2    2000-03-01       <20        FEMALE      NaN              2243
3    2002-07-06       20s        FEMALE      NaN              2270
4    2022-08-14       60s          MALE      NaN              2233

                                    Reporting_PHU  Reporting_PHU_Address  \
0                 Thunder Bay District Health Unit    999 Balmoral Street
1               York Region Public Health Services     17250 Yonge Street
2  Leeds, Grenville and Lanark District Health Unit  458 Laurier Boulevard
3               York Region Public Health Services     17250 Yonge Street
4                           Grey Bruce Health Unit   101 17th Street East

  Reporting_PHU_City Reporting_PHU_Postal_Code  \
0        Thunder Bay                   P7B 6E7
1          Newmarket                   L3Y 6Z1
2         Brockville                   K6V 7A3
3          Newmarket                   L3Y 6Z1
4         Owen Sound                   N4K 0A5

                    Reporting_PHU_Website  Reporting_PHU_Latitude  \
0                           www.tbdhu.com                48.400572
1  www.york.ca/wps/portal/yorkhome/health/                44.048023
2                       www.healthunit.org                44.615843
3  www.york.ca/wps/portal/yorkhome/health/                44.048023
4         www.publichealthgreybruce.on.ca/                44.576196

   Reporting_PHU_Longitude
0               -89.258851
1               -79.480239
2               -75.702833
3               -79.480239
4               -80.940980
```

[2]:
```
# Display basic information about the dataset
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1717434 entries, 0 to 1717433
Data columns (total 16 columns):
 #   Column                    Dtype
---  ------                    -----
 0   Row_ID                    int64
 1   Accurate_Episode_Date     object
 2   Case_Reported_Date        object
 3   Test_Reported_Date        object
 4   Specimen_Date             object
```

```
 5   Age_Group                  object
 6   Client_Gender              object
 7   Outcome1                   object
 8   Reporting_PHU_ID           int64
 9   Reporting_PHU              object
 10  Reporting_PHU_Address      object
 11  Reporting_PHU_City         object
 12  Reporting_PHU_Postal_Code  object
 13  Reporting_PHU_Website      object
 14  Reporting_PHU_Latitude     float64
 15  Reporting_PHU_Longitude    float64
dtypes: float64(2), int64(2), object(12)
memory usage: 209.6+ MB
None
```

[3]:
```
# Display summary statistics of the dataset
print(df.describe())
```

```
              Row_ID  Reporting_PHU_ID  Reporting_PHU_Latitude  \
count  1.717434e+06      1.717434e+06            1.717434e+06
mean   8.587175e+05      2.685810e+03            4.396700e+01
std    4.957806e+05      7.631429e+02            1.153449e+00
min    1.000000e+00      2.226000e+03            4.230880e+01
25%    4.293592e+05      2.244000e+03            4.346288e+01
50%    8.587175e+05      2.257000e+03            4.365659e+01
75%    1.288076e+06      3.895000e+03            4.404802e+01
max    1.717434e+06      5.183000e+03            4.976961e+01

       Reporting_PHU_Longitude
count             1.717434e+06
mean             -7.973390e+01
std               2.396228e+00
min              -9.448825e+01
25%              -7.987134e+01
50%              -7.948024e+01
75%              -7.937936e+01
max              -7.473630e+01
```

**2. Data Cleaning and Preprocessing** Handling Missing Data: Consider imputation strategies or exclude fields with significant missing data if not crucial. From the missing data analysis, we see the following attributes with missing values: - Test_Reported_Date: 53,492 missing values - Specimen_Date: 12,133 missing values - Outcome1: 1,698,807 missing values Strategies for Handling Missing Data: 1. Test_Reported_Date and Specimen_Date: the number of missing values is small compared to the total dataset size, these rows will be removed. 2. Outcome1: As per the helth Ministry (variable 'Outcome1' will be equal to 'Fatal' (deaths due to COVID-19) or blank (all other cases), this field will categorize as 'Nonfatal'. Steps for Data Cleaning: 1. Remove Duplicate Records: Ensure no duplicate rows are present in the dataset. 2. Standardize Date Formats: Ensure all date fields are in a consistent format. 3. Categorical Data Encoding: Convert categorical data to numerical values for machine learning models.(will do it after the visualizations )

[4]:
```
# Check for missing values
print(df.isnull().sum())
```

```
Row_ID                          0
Accurate_Episode_Date           0
Case_Reported_Date              0
Test_Reported_Date          53492
Specimen_Date               12133
Age_Group                       0
Client_Gender                   0
Outcome1                  1698807
Reporting_PHU_ID                0
Reporting_PHU                   0
Reporting_PHU_Address           0
Reporting_PHU_City              0
Reporting_PHU_Postal_Code       0
Reporting_PHU_Website           0
Reporting_PHU_Latitude          0
Reporting_PHU_Longitude         0
dtype: int64
```

[5]:
```
# Handle missing values drop the rows
df.dropna(subset=['Test_Reported_Date', 'Specimen_Date'], inplace=True)

# Fill missing Outcome1 with 'Nonfatal'
df['Outcome1'].fillna('Nonfatal', inplace=True)
```

[6]:
```
# Remove duplicate rows
df.drop_duplicates(inplace=True)
```

[7]:
```
# reCheck the missing values
print(df.isnull().sum())
```

```
Row_ID                      0
Accurate_Episode_Date       0
Case_Reported_Date          0
Test_Reported_Date          0
Specimen_Date               0
Age_Group                   0
Client_Gender               0
Outcome1                    0
Reporting_PHU_ID            0
Reporting_PHU               0
Reporting_PHU_Address       0
Reporting_PHU_City          0
Reporting_PHU_Postal_Code   0
Reporting_PHU_Website       0
Reporting_PHU_Latitude      0
```

```
Reporting_PHU_Longitude      0
dtype: int64
```

[8]:
```
# Convert date columns to datetime format
date_columns = ['Accurate_Episode_Date', 'Case_Reported_Date',␣
 ↪'Test_Reported_Date', 'Specimen_Date']
for col in date_columns:
    df[col] = pd.to_datetime(df[col], errors='coerce')
```

[9]:
```
# Save cleaned data
cleaned_file = r"C:\Users\ENG WAHEED\Desktop\New␣
 ↪folder\cleaned_COVID_19_dataset.csv"
df.to_csv('cleaned_file', index=False)

# Load the cleaned dataset
cleaned_file = r"C:\Users\ENG WAHEED\Desktop\New␣
 ↪folder\cleaned_COVID_19_dataset.csv"
df = pd.read_csv('cleaned_file')

print(df.head())
```

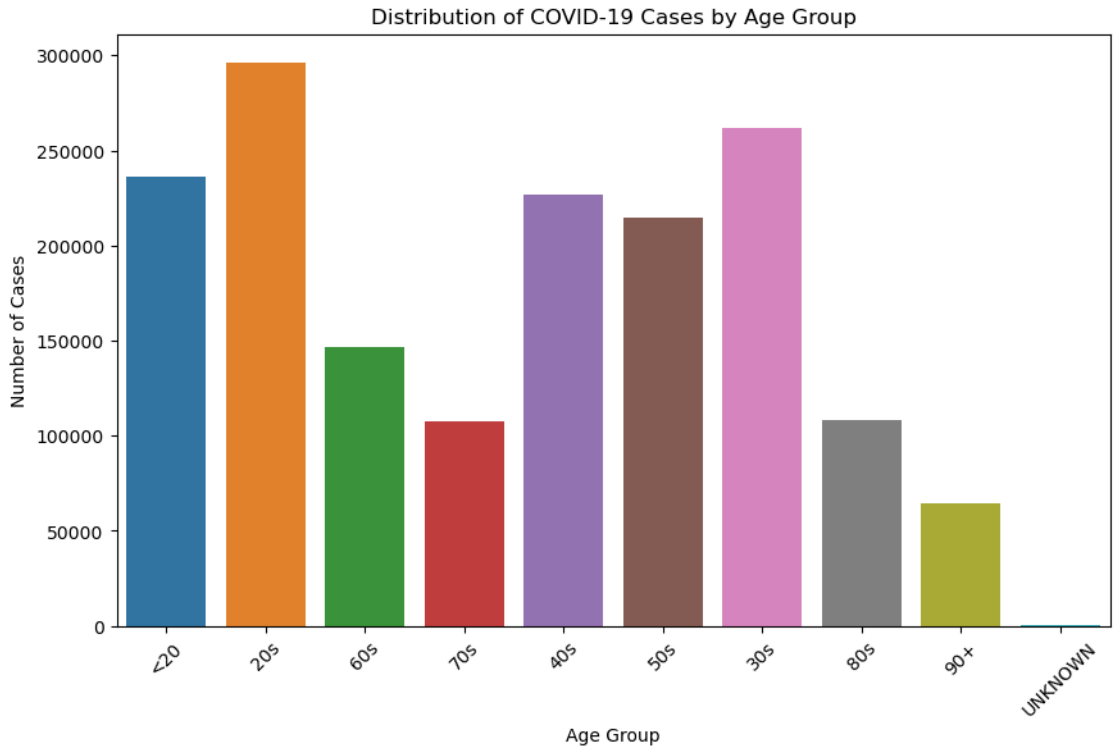
```
   Row_ID Accurate_Episode_Date Case_Reported_Date Test_Reported_Date  \
0       1            1934-09-28         2022-09-29         2022-09-29
1       2            1989-02-21         2022-11-08         2022-11-07
2       4            2002-07-06         2022-07-06         2022-07-07
3       5            2002-08-08         2022-08-15         2022-08-15
4       6            2008-08-27         2022-08-28         2022-08-28

  Specimen_Date Age_Group Client_Gender  Outcome1  Reporting_PHU_ID  \
0    2022-09-27       <20        FEMALE  Nonfatal              2262
1    2022-11-06       <20        FEMALE  Nonfatal              2270
2    2002-07-06       20s        FEMALE  Nonfatal              2270
3    2022-08-14       60s          MALE  Nonfatal              2233
4    2022-08-27       70s        FEMALE  Nonfatal              2233

                        Reporting_PHU Reporting_PHU_Address  \
0    Thunder Bay District Health Unit   999 Balmoral Street
1  York Region Public Health Services    17250 Yonge Street
2  York Region Public Health Services    17250 Yonge Street
3              Grey Bruce Health Unit  101 17th Street East
4              Grey Bruce Health Unit  101 17th Street East

  Reporting_PHU_City Reporting_PHU_Postal_Code  \
0        Thunder Bay                   P7B 6E7
1          Newmarket                   L3Y 6Z1
2          Newmarket                   L3Y 6Z1
3         Owen Sound                   N4K 0A5
4         Owen Sound                   N4K 0A5
```

```
                    Reporting_PHU_Website  Reporting_PHU_Latitude  \
0                             www.tbdhu.com                48.400572
1  www.york.ca/wps/portal/yorkhome/health/                44.048023
2  www.york.ca/wps/portal/yorkhome/health/                44.048023
3         www.publichealthgreybruce.on.ca/                44.576196
4         www.publichealthgreybruce.on.ca/                44.576196

   Reporting_PHU_Longitude
0               -89.258851
1               -79.480239
2               -79.480239
3               -80.940980
4               -80.940980
```

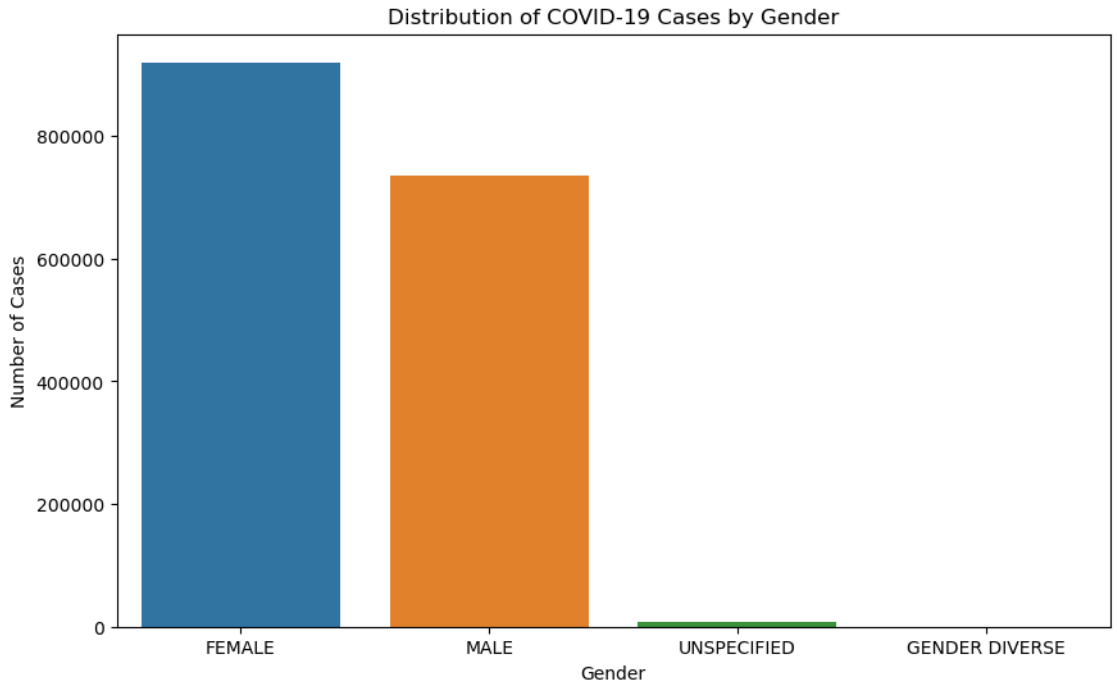
### 3. Exploratory Data Analysis

Key findings and visualizations

[10]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Plot distribution of cases by age group
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Age_Group')
plt.title('Distribution of COVID-19 Cases by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Number of Cases')
plt.xticks(rotation=45)
plt.show()
```




[11]:
```
# Plot distribution of cases by gender
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Client_Gender')
plt.title('Distribution of COVID-19 Cases by Gender')
plt.xlabel('Gender')
plt.ylabel('Number of Cases')
plt.show()
```




[12]:
```
# Cross-tabulation of age group and outcome
age_outcome_ct = pd.crosstab(df['Age_Group'], df['Outcome1'])
print(age_outcome_ct)
```

```
Outcome1    FATAL  Nonfatal
Age_Group
20s            51    296337
30s           126    261761
40s           276    226409
50s           820    213938
60s          1953    144432
70s          3701    104207
80s          6023    102483
90+          4828     59829
<20            30    236250
UNKNOWN         0       322
```

[13]:
```
# Cross-tabulation of gender and outcome
gender_outcome_ct = pd.crosstab(df['Client_Gender'], df['Outcome1'])
print(gender_outcome_ct)
```

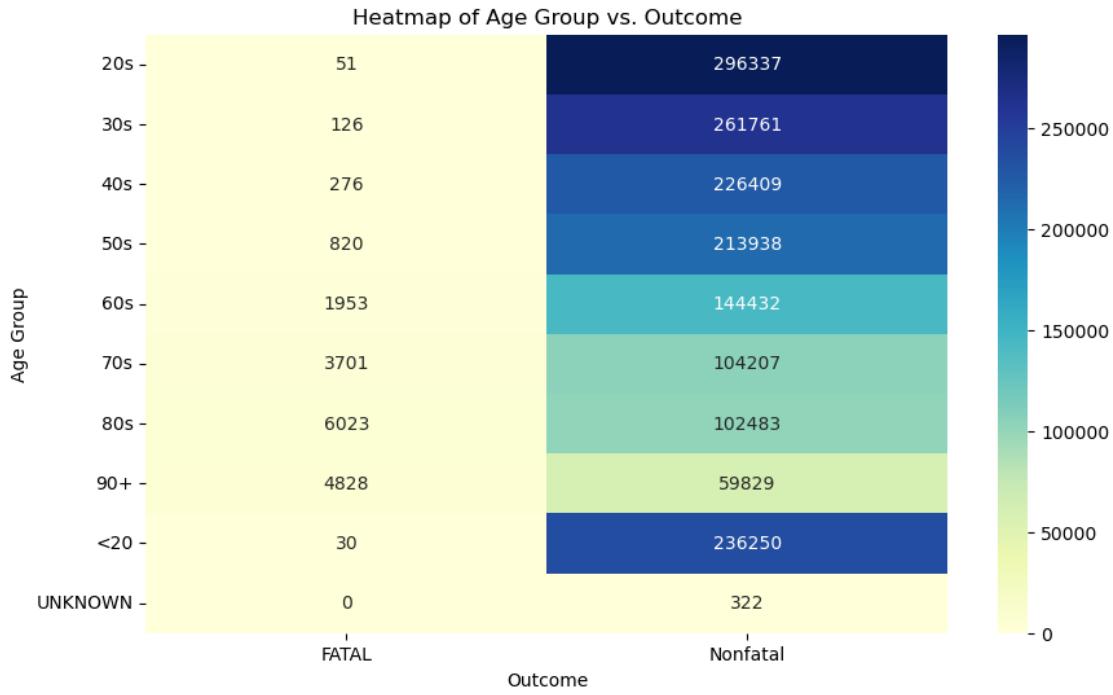
```
Outcome1         FATAL  Nonfatal
Client_Gender
FEMALE            8327    912048
GENDER DIVERSE       0         9
MALE              9426    725708
```

```
UNSPECIFIED        55       8203
```

[14]:
```
# Perform Chi-Square Test for Client_Gender and Outcome1
from scipy.stats import chi2_contingency
chi2_gender, p_gender, dof_gender, ex_gender =␣
 ↪chi2_contingency(gender_outcome_ct)
print(f'Chi-Square Test for Client_Gender and Outcome1: chi2={chi2_gender},␣
 ↪p-value={p_gender}')
```

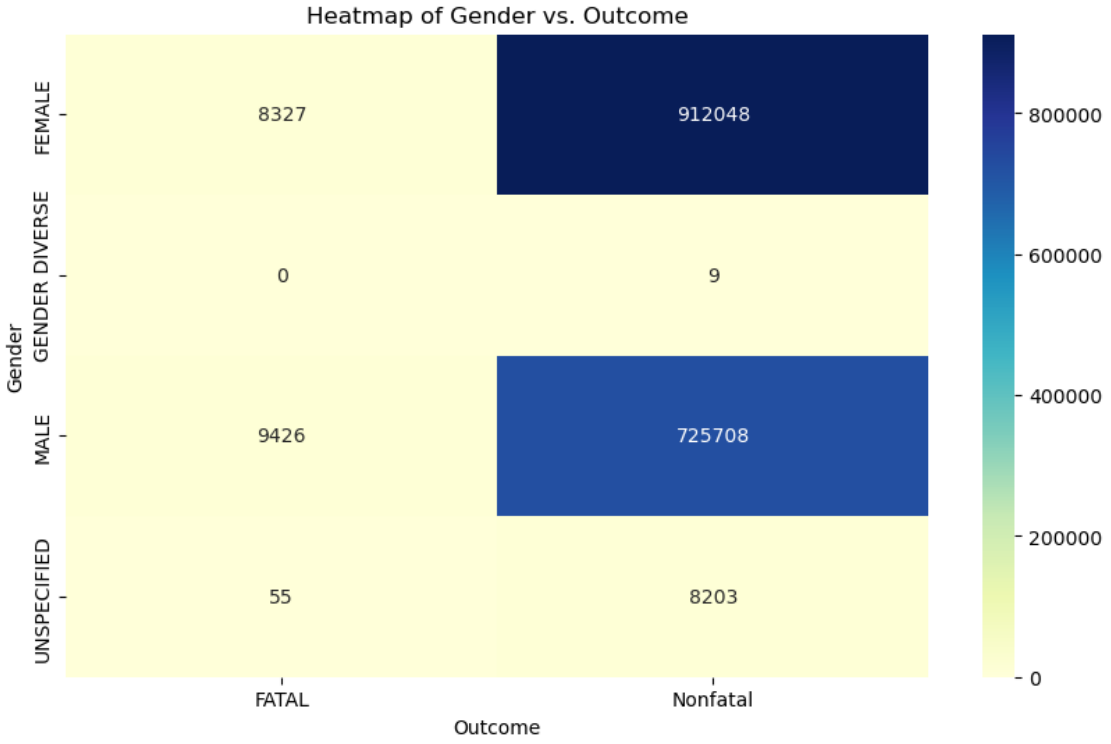
```
Chi-Square Test for Client_Gender and Outcome1: chi2=562.868763407916,
p-value=1.1285327612576004e-121
```

[15]:
```
# Heatmap for Age Group and Outcome
plt.figure(figsize=(10, 6))
sns.heatmap(age_outcome_ct, annot=True, cmap='YlGnBu', fmt='d')
plt.title('Heatmap of Age Group vs. Outcome')
plt.xlabel('Outcome')
plt.ylabel('Age Group')
plt.show()
```




[16]:
```
# Heatmap for Gender and Outcome
plt.figure(figsize=(10, 6))
sns.heatmap(gender_outcome_ct, annot=True, cmap='YlGnBu', fmt='d')
plt.title('Heatmap of Gender vs. Outcome')
plt.xlabel('Outcome')
```

```
plt.ylabel('Gender')
plt.show()
```




**From the visualizations we can answer the first question**

1. What are the key demographic factors (such as age group and gender) that influence the number of COVID-19 cases?

The highest number of COVID-19 cases is observed in the younger age groups (20s, 30s, <20). The highest number of fatal cases is observed in older age groups (80s, 90+). Females have a higher number of COVID-19 cases overall. Males have a higher number of fatal cases compared to females, indicating a higher fatality rate among males.

These findings suggest that age and gender are significant demographic factors influencing the number of COVID-19 cases and outcomes, with older age groups and males being at higher risk for fatal outcomes.

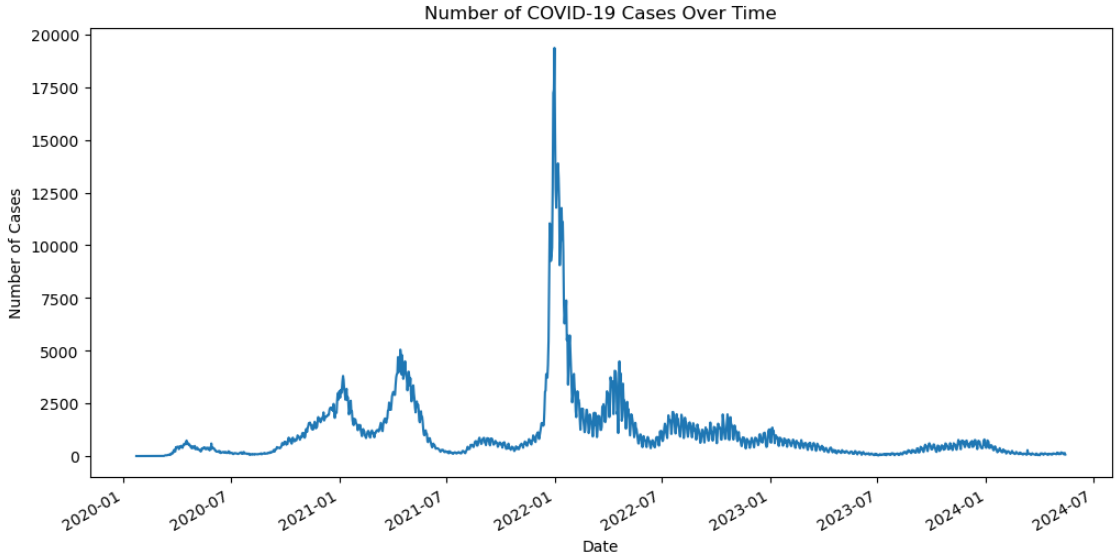
[17]:
```
# Check the unique values in Reporting_PHU and related columns
print(df['Reporting_PHU'].unique())
print(df[['Reporting_PHU_Address', 'Reporting_PHU_City',␣
  ↪'Reporting_PHU_Postal_Code', 'Reporting_PHU_Website',␣
  ↪'Reporting_PHU_Latitude', 'Reporting_PHU_Longitude']].head())
```

```
['Thunder Bay District Health Unit' 'York Region Public Health Services'
 'Grey Bruce Health Unit' 'Hamilton Public Health Services'
```

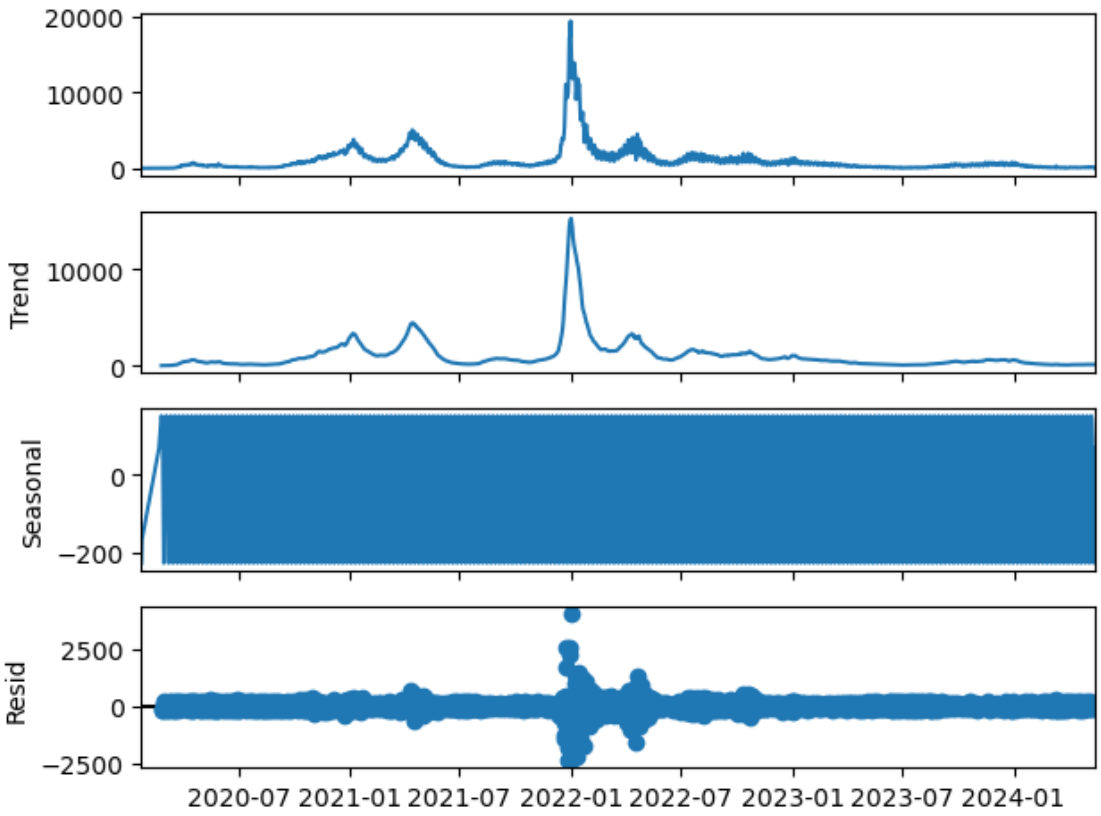
```
 'Haldimand-Norfolk Health Unit' 'Simcoe Muskoka District Health Unit'
 'Southwestern Public Health' 'Region of Waterloo, Public Health'
 'Eastern Ontario Health Unit' 'Toronto Public Health'
 'Middlesex-London Health Unit' 'Peel Public Health'
 'Hastings and Prince Edward Counties Health Unit' 'Ottawa Public Health'
 'Wellington-Dufferin-Guelph Public Health'
 'Windsor-Essex County Health Unit' 'Durham Region Health Department'
 'Peterborough Public Health'
 'Leeds, Grenville and Lanark District Health Unit'
 'Niagara Region Public Health Department'
 'Haliburton, Kawartha, Pine Ridge District Health Unit'
 'Halton Region Health Department' 'Sudbury & District Health Unit'
 'Porcupine Health Unit'
 'Kingston, Frontenac and Lennox & Addington Public Health'
 'Lambton Public Health' 'Brant County Health Unit'
 'Huron Perth District Health Unit' 'Chatham-Kent Health Unit'
 'Algoma Public Health Unit' 'North Bay Parry Sound District Health Unit'
 'Timiskaming Health Unit' 'Renfrew County and District Health Unit'
 'Northwestern Health Unit']
  Reporting_PHU_Address Reporting_PHU_City Reporting_PHU_Postal_Code  \
0   999 Balmoral Street        Thunder Bay                    P7B 6E7
1    17250 Yonge Street          Newmarket                    L3Y 6Z1
2    17250 Yonge Street          Newmarket                    L3Y 6Z1
3  101 17th Street East         Owen Sound                    N4K 0A5
4  101 17th Street East         Owen Sound                    N4K 0A5

                    Reporting_PHU_Website  Reporting_PHU_Latitude  \
0                           www.tbdhu.com               48.400572
1  www.york.ca/wps/portal/yorkhome/health/               44.048023
2  www.york.ca/wps/portal/yorkhome/health/               44.048023
3         www.publichealthgreybruce.on.ca/               44.576196
4         www.publichealthgreybruce.on.ca/               44.576196

   Reporting_PHU_Longitude
0               -89.258851
1               -79.480239
2               -79.480239
3               -80.940980
4               -80.940980
```

[18]:
```
# Plot the number of cases over time
df['Case_Reported_Date'] = pd.to_datetime(df['Case_Reported_Date'])
cases_over_time = df.groupby('Case_Reported_Date').size()

plt.figure(figsize=(12, 6))
cases_over_time.plot()
plt.title('Number of COVID-19 Cases Over Time')
```

```
plt.xlabel('Date')
plt.ylabel('Number of Cases')
plt.show()
```




[19]:
```
# Decompose the time series to identify trends, seasonality, and residuals
from statsmodels.tsa.seasonal import seasonal_decompose

decomposition = seasonal_decompose(cases_over_time, model='additive', period=7)
decomposition.plot()
plt.show()
```




[22]:
```
# Aggregate Data by Reporting_PHU and Date
daily_cases = df.groupby(['Reporting_PHU', 'Accurate_Episode_Date']).size().
  ↪reset_index(name='case_count')
print(daily_cases)
```

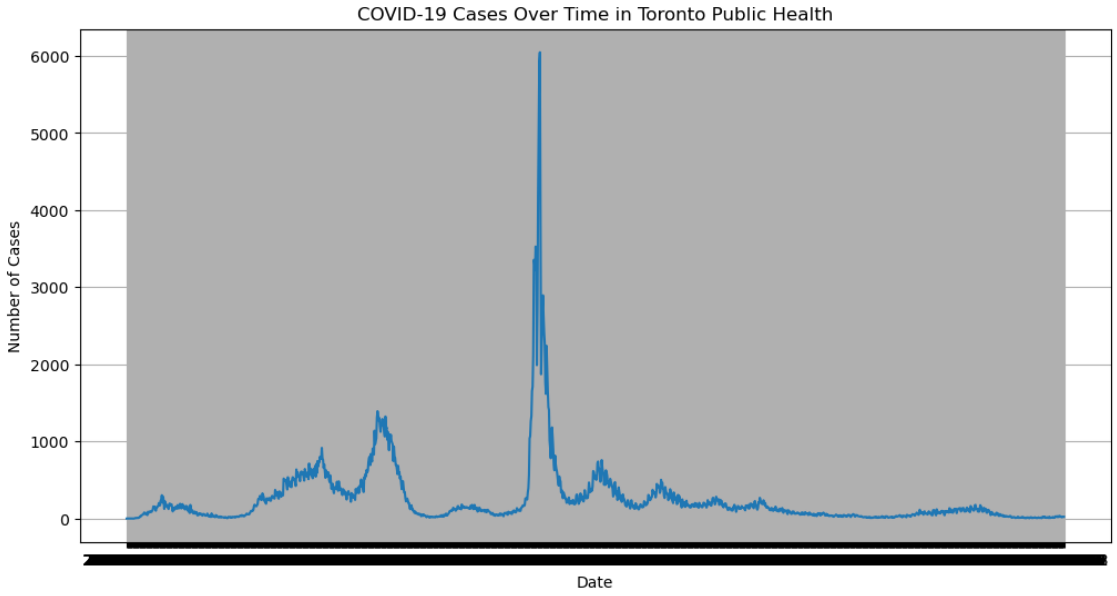
```
                          Reporting_PHU Accurate_Episode_Date  case_count
0                Algoma Public Health Unit            2020-03-08           1
1                Algoma Public Health Unit            2020-03-15           1
2                Algoma Public Health Unit            2020-03-20           2
3                Algoma Public Health Unit            2020-03-22           2
4                Algoma Public Health Unit            2020-03-23           2
…                                      …                     …         …
44527  York Region Public Health Services            2024-05-10           5
44528  York Region Public Health Services            2024-05-11           5
44529  York Region Public Health Services            2024-05-12           6
44530  York Region Public Health Services            2024-05-13          11
44531  York Region Public Health Services            2024-05-14           2

[44532 rows x 3 columns]
```

[23]:
```
# Aggregate the number of cases by Reporting_PHU.
phu_case_counts = daily_cases.groupby('Reporting_PHU')['case_count'].sum().
 ↪reset_index()
print(phu_case_counts)
```

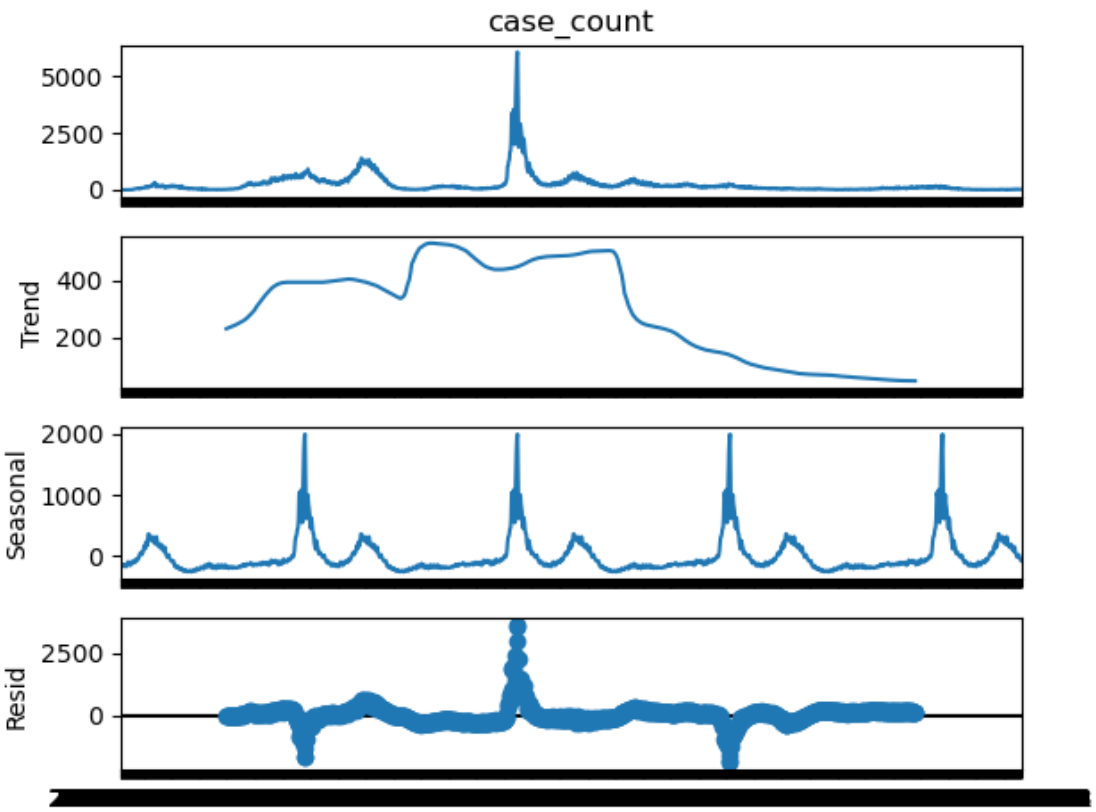
| | Reporting_PHU | case_count |
|---|---|---|
| 0 | Algoma Public Health Unit | 12108 |
| 1 | Brant County Health Unit | 15615 |
| 2 | Chatham-Kent Health Unit | 12025 |
| 3 | Durham Region Health Department | 77117 |
| 4 | Eastern Ontario Health Unit | 21991 |
| 5 | Grey Bruce Health Unit | 12946 |
| 6 | Haldimand-Norfolk Health Unit | 11872 |
| 7 | Haliburton, Kawartha, Pine Ridge District Heal… | 14652 |
| 8 | Halton Region Health Department | 60832 |
| 9 | Hamilton Public Health Services | 79149 |
| 10 | Hastings and Prince Edward Counties Health Unit | 15688 |
| 11 | Huron Perth District Health Unit | 11138 |
| 12 | Kingston, Frontenac and Lennox & Addington Pub… | 26954 |
| 13 | Lambton Public Health | 15259 |
| 14 | Leeds, Grenville and Lanark District Health Unit | 13275 |
| 15 | Middlesex-London Health Unit | 50418 |
| 16 | Niagara Region Public Health Department | 54735 |
| 17 | North Bay Parry Sound District Health Unit | 9410 |
| 18 | Northwestern Health Unit | 8603 |
| 19 | Ottawa Public Health | 98355 |
| 20 | Peel Public Health | 216474 |
| 21 | Peterborough Public Health | 12467 |
| 22 | Porcupine Health Unit | 9268 |
| 23 | Region of Waterloo, Public Health | 59722 |
| 24 | Renfrew County and District Health Unit | 7687 |
| 25 | Simcoe Muskoka District Health Unit | 60891 |
| 26 | Southwestern Public Health | 19598 |
| 27 | Sudbury & District Health Unit | 22767 |
| 28 | Thunder Bay District Health Unit | 20751 |
| 29 | Timiskaming Health Unit | 1745 |
| 30 | Toronto Public Health | 388421 |
| 31 | Wellington-Dufferin-Guelph Public Health | 29420 |
| 32 | Windsor-Essex County Health Unit | 56040 |
| 33 | York Region Public Health Services | 136383 |

[24]:
```
#Plot the number of cases over time using line charts
# Function to plot cases over time for a specific PHU
def plot_cases_over_time(phu_name):
    phu_data = daily_cases[daily_cases['Reporting_PHU'] == phu_name]
    plt.figure(figsize=(12, 6))
    plt.plot(phu_data['Accurate_Episode_Date'], phu_data['case_count'])
```

```
    plt.title(f'COVID-19 Cases Over Time in {phu_name}')
    plt.xlabel('Date')
    plt.ylabel('Number of Cases')
    plt.grid(True)
    plt.show()

# Example: Plot for 'Toronto Public Health'
plot_cases_over_time('Toronto Public Health')
```



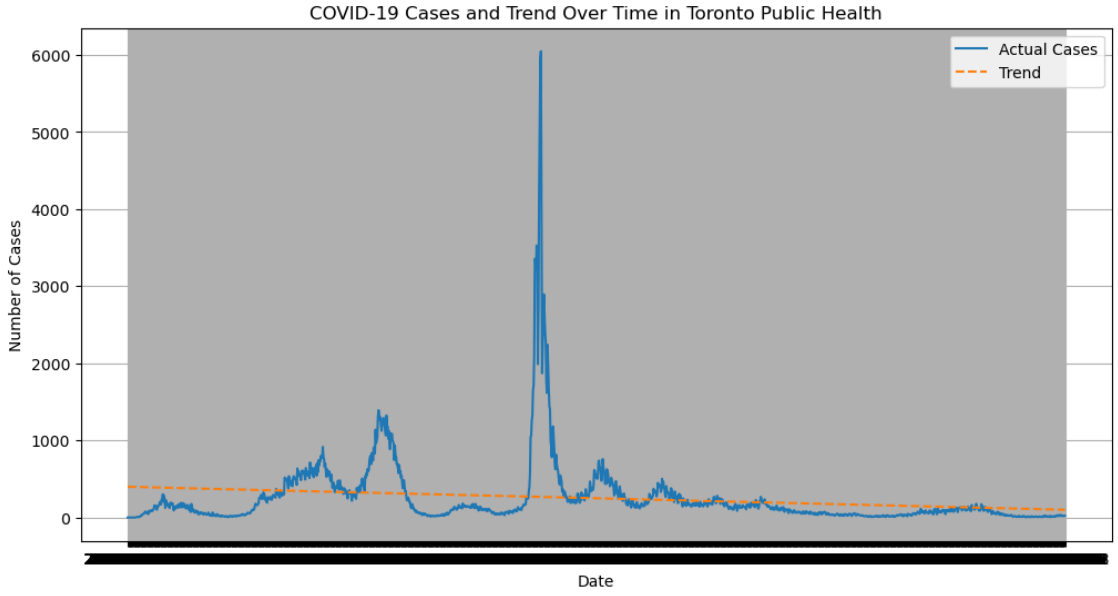

[25]:
```
# Decompose the time series to identify trends, seasonality, and residuals
from statsmodels.tsa.seasonal import seasonal_decompose

# Example: Decompose the time series for 'Toronto Public Health'
phu_name = 'Toronto Public Health'
phu_data = daily_cases[daily_cases['Reporting_PHU'] == phu_name]
phu_data.set_index('Accurate_Episode_Date', inplace=True)
result = seasonal_decompose(phu_data['case_count'], model='additive',␣
 ↪period=365)

result.plot()
plt.show()
```




[26]:
```
# Perform a trend analysis using statistical methods or machine learning models␣
↪to identify patterns over time.

import numpy as np
import seaborn as sns
import statsmodels.api as sm

# Example: Linear trend analysis for 'Toronto Public Health'
phu_name = 'Toronto Public Health'
phu_data = daily_cases[daily_cases['Reporting_PHU'] == phu_name]
phu_data.set_index('Accurate_Episode_Date', inplace=True)

# Adding a column for the date as an integer for trend analysis
phu_data['date_int'] = np.arange(len(phu_data))

# Fitting a linear regression model
model = sm.OLS(phu_data['case_count'], sm.add_constant(phu_data['date_int']))
results = model.fit()
phu_data['trend'] = results.predict(sm.add_constant(phu_data['date_int']))
```

```python
# Plotting the actual cases and the trend
plt.figure(figsize=(12, 6))
plt.plot(phu_data.index, phu_data['case_count'], label='Actual Cases')
plt.plot(phu_data.index, phu_data['trend'], label='Trend', linestyle='--')
plt.title(f'COVID-19 Cases and Trend Over Time in {phu_name}')
plt.xlabel('Date')
plt.ylabel('Number of Cases')
plt.legend()
plt.grid(True)
plt.show()
```

```
C:\Users\ENG WAHEED\AppData\Local\Temp\ipykernel_16088\2898915005.py:11:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  phu_data['date_int'] = np.arange(len(phu_data))
C:\Users\ENG WAHEED\AppData\Local\Temp\ipykernel_16088\2898915005.py:16:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  phu_data['trend'] = results.predict(sm.add_constant(phu_data['date_int']))
```



**From the visualizations and findings we can answer the Second question**

2. How do the COVID-19 case trends vary over time and across different public health unites in Ontario?

The COVID-19 case trends in Ontario vary significantly over time and across different public health units. Urban areas like Toronto and Peel show multiple waves with high peaks, reflecting their larger and denser populations. In contrast, smaller and rural PHUs have lower case counts and less pronounced peaks. Seasonal decomposition reveals regular periodic increases in cases, and trend analysis shows an overall decline after major peaks, likely due to effective public health measures and vaccination campaigns.

These findings highlight the importance of tailored public health strategies to address the unique needs and circumstances of different regions within Ontario.

**Model Selection and Training**

**Data Preprocessing**

convert categorical data to numerical values, use various encoding techniques such as Label Encoding or One-Hot Encoding.

`Label Encoding: Assigns each unique value in a categorical column an integer value.`
`One-Hot Encoding: Creates a new binary column for each unique value in the categorical column.`

[27]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder


# load the data
df = pd.read_csv('cleaned_file')
```

[28]:
```
# Convert datetime columns to datetime format
df['Accurate_Episode_Date'] = pd.to_datetime(df['Accurate_Episode_Date'],␣
 ↪errors='coerce')
df['Case_Reported_Date'] = pd.to_datetime(df['Case_Reported_Date'],␣
 ↪errors='coerce')
df['Test_Reported_Date'] = pd.to_datetime(df['Test_Reported_Date'],␣
 ↪errors='coerce')
df['Specimen_Date'] = pd.to_datetime(df['Specimen_Date'], errors='coerce')
```

[29]:
```
# Extract year, month, and day from datetime columns
df['Accurate_Episode_Date_year'] = df['Accurate_Episode_Date'].dt.year
df['Accurate_Episode_Date_month'] = df['Accurate_Episode_Date'].dt.month
df['Accurate_Episode_Date_day'] = df['Accurate_Episode_Date'].dt.day

df['Case_Reported_Date_year'] = df['Case_Reported_Date'].dt.year
df['Case_Reported_Date_month'] = df['Case_Reported_Date'].dt.month
```

```
df['Case_Reported_Date_day'] = df['Case_Reported_Date'].dt.day

df['Test_Reported_Date_year'] = df['Test_Reported_Date'].dt.year
df['Test_Reported_Date_month'] = df['Test_Reported_Date'].dt.month
df['Test_Reported_Date_day'] = df['Test_Reported_Date'].dt.day

df['Specimen_Date_year'] = df['Specimen_Date'].dt.year
df['Specimen_Date_month'] = df['Specimen_Date'].dt.month
df['Specimen_Date_day'] = df['Specimen_Date'].dt.day
```

[30]:
```
# Drop the original datetime columns
df.drop(columns=['Accurate_Episode_Date', 'Case_Reported_Date',␣
↪'Test_Reported_Date', 'Specimen_Date'], inplace=True)
```

[31]:
```
# Label Encoding for 'Outcome1'
df['Outcome1_Encoded'] = LabelEncoder().fit_transform(df['Outcome1'])

# One-Hot Encoding for 'Age_Group' and 'Client_Gender'
df = pd.get_dummies(df, columns=['Age_Group', 'Client_Gender'])
```

[32]:
```
# Print the columns in the dataframe
print("Columns in the dataframe:", df.columns)
```

```
Columns in the dataframe: Index(['Row_ID', 'Outcome1', 'Reporting_PHU_ID',
'Reporting_PHU',
       'Reporting_PHU_Address', 'Reporting_PHU_City',
       'Reporting_PHU_Postal_Code', 'Reporting_PHU_Website',
       'Reporting_PHU_Latitude', 'Reporting_PHU_Longitude',
       'Accurate_Episode_Date_year', 'Accurate_Episode_Date_month',
       'Accurate_Episode_Date_day', 'Case_Reported_Date_year',
       'Case_Reported_Date_month', 'Case_Reported_Date_day',
       'Test_Reported_Date_year', 'Test_Reported_Date_month',
       'Test_Reported_Date_day', 'Specimen_Date_year', 'Specimen_Date_month',
       'Specimen_Date_day', 'Outcome1_Encoded', 'Age_Group_20s',
       'Age_Group_30s', 'Age_Group_40s', 'Age_Group_50s', 'Age_Group_60s',
       'Age_Group_70s', 'Age_Group_80s', 'Age_Group_90+', 'Age_Group_<20',
       'Age_Group_UNKNOWN', 'Client_Gender_FEMALE',
       'Client_Gender_GENDER DIVERSE', 'Client_Gender_MALE',
       'Client_Gender_UNSPECIFIED'],
      dtype='object')
```

[33]:
```
# Define columns to drop if they exist in the dataframe
columns_to_drop = ['Row_ID', 'Outcome1', 'Reporting_PHU',␣
↪'Reporting_PHU_Address', 'Reporting_PHU_City', 'Reporting_PHU_Postal_Code',␣
↪'Reporting_PHU_Website']
df.drop(columns=[col for col in columns_to_drop if col in df.columns],␣
↪inplace=True)
```

[34]:
```
# Select features and target variable
features = df.drop(columns=['Outcome1_Encoded'])
target = df['Outcome1_Encoded']
```

[35]:
```
# Split the data into training and testing sets and Initialize the model

X_train, X_test, y_train, y_test = train_test_split(features, target,␣
 ↪test_size=0.2, random_state=42)

model = LogisticRegression(max_iter=1000)
```

[36]:
```
# Train the model
model.fit(X_train, y_train)

# Evaluate the model
accuracy = model.score(X_test, y_test)
print("Accuracy:", accuracy)
```

```
C:\Users\ENG WAHEED\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: 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
  n_iter_i = _check_optimize_result(
```

```
Accuracy: 0.9891572203055693
```

[ ]:
```
# Confusion matrix
confusion = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(confusion)
```

[ ]:
```
# Classification report
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```

The classification report and confusion matrix indicate that the model is heavily biased towards the majority class (class 1), resulting in a very high accuracy but poor performance on the minority class (class 0)

will use class_weight parameter in LogisticRegression:

[ ]:
```
from sklearn.utils.class_weight import compute_class_weight
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report

# Compute class weights
class_weights = compute_class_weight(class_weight='balanced', classes=[0, 1],␣
 ↪y=target)
class_weights_dict = {0: class_weights[0], 1: class_weights[1]}

# Initialize the model with class weights
model = LogisticRegression(max_iter=1000, class_weight=class_weights_dict)
```

[ ]:
```
# Train the model
model.fit(X_train, y_train)

# Evaluate the model
accuracy = model.score(X_test, y_test)
print("Accuracy:", accuracy)

# Predict on the test set
y_pred = model.predict(X_test)
```

[ ]:
```
# Confusion matrix
confusion = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(confusion)
```

[ ]:
```
# Classification report
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```