2348441_lab_08

April 5, 2024

Lab Exercise 7 and 8 -Linear Discriminant Analysis, Logistic Discriminant / Logistic Regression

- Created by : Nileem Kaveramma C C | 2348441
- Created DATE:5-04-2024
- Edited Date: 5-04-2024

AIM: The aim of this study is to compare the performance of Linear Discriminant Analysis (LDA) and Logistic Regression (LR) in predicting the medical conditions of patients based on their demographic and medical profile.

IMPORTED LIBRARIES

numpy - for numerical, array, matrices (Linear Algebra) processing Pandas - for loading and processing datasets matplotlib.pyplot - For visualisation Saeborn - for statistical graph

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

[ ]:
```
df = pd.read_csv('/content/healthcare_dataset.csv')
df
```

[ ]:
```
                     Name  Age  Gender Blood Type Medical Condition  \
0          Tiffany Ramirez   81  Female         O-          Diabetes
1              Ruben Burns   35    Male         O+            Asthma
2                Chad Byrd   61    Male         B-           Obesity
3        Antonio Frederick   49    Male         B-            Asthma
4      Mrs. Brandy Flowers   51    Male         O-         Arthritis
…                      …  …     …        …                …
9995           James Hood   83    Male         A+           Obesity
9996      Stephanie Evans   47  Female        AB+         Arthritis
9997  Christopher Martinez   54    Male         B-         Arthritis
9998          Amanda Duke   84    Male         A+         Arthritis
9999            Eric King   20    Male         B-         Arthritis

     Date of Admission             Doctor                   Hospital  \
0           2022-11-17     Patrick Parker           Wallace-Hamilton
1           2023-06-01      Diane Jackson  Burke, Griffin and Cooper
```

```
2           2019-01-09         Paul Baker                 Walton LLC
3           2020-05-02     Brian Chandler                 Garcia Ltd
4           2021-07-09     Dustin Griffin    Jones, Brown and Murray
…                  …                  …                          …
9995        2022-07-29       Samuel Moody   Wood, Martin and Simmons
9996        2022-01-06  Christopher Yates               Nash-Krueger
9997        2022-07-01   Robert Nicholson           Larson and Sons
9998        2020-02-06        Jamie Lewis               Wilson-Lyons
9999        2023-03-22        Tasha Avila  Torres, Young and Stewart

     Insurance Provider  Billing Amount  Room Number Admission Type  \
0              Medicare    37490.983364          146       Elective
1      UnitedHealthcare    47304.064845          404      Emergency
2              Medicare    36874.896997          292      Emergency
3              Medicare    23303.322092          480         Urgent
4      UnitedHealthcare    18086.344184          477         Urgent
…                   …             …           …            …
9995   UnitedHealthcare    39606.840083          110       Elective
9996         Blue Cross     5995.717488          244      Emergency
9997         Blue Cross    49559.202905          312       Elective
9998   UnitedHealthcare    25236.344761          420         Urgent
9999              Aetna    37223.965865          290      Emergency

     Discharge Date   Medication  Test Results
0        2022-12-01      Aspirin  Inconclusive
1        2023-06-15      Lipitor        Normal
2        2019-02-08      Lipitor        Normal
3        2020-05-03   Penicillin      Abnormal
4        2021-08-02  Paracetamol        Normal
…               …            …             …
9995     2022-08-02    Ibuprofen      Abnormal
9996     2022-01-29    Ibuprofen        Normal
9997     2022-07-15    Ibuprofen        Normal
9998     2020-02-26   Penicillin        Normal
9999     2023-04-15   Penicillin      Abnormal

[10000 rows x 15 columns]
```

df.shape - attribute is used to get the dimensions of the DataFrame.

[ ]:
```
df.shape
```

[ ]:
```
(10000, 15)
```

df.columns attribute is used to retrieve the column labels or names of the DataFrame

[ ]:
```
df.columns
```

```
[ ]: Index(['Name', 'Age', 'Gender', 'Blood Type', 'Medical Condition',
       'Date of Admission', 'Doctor', 'Hospital', 'Insurance Provider',
       'Billing Amount', 'Room Number', 'Admission Type', 'Discharge Date',
       'Medication', 'Test Results'],
      dtype='object')
```

df.dtypes attribute is used to retrieve the data types of each column in a DataFrame

```
[ ]: df.dtypes
```

```
[ ]: Name                  object
     Age                    int64
     Gender                object
     Blood Type            object
     Medical Condition     object
     Date of Admission     object
     Doctor                object
     Hospital              object
     Insurance Provider    object
     Billing Amount       float64
     Room Number            int64
     Admission Type        object
     Discharge Date        object
     Medication            object
     Test Results          object
     dtype: object
```

df.head() method is used to display the first few rows of a DataFrame

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

```
[ ]:                  Name  Age  Gender Blood Type Medical Condition  \
     0      Tiffany Ramirez   81  Female         O-          Diabetes
     1          Ruben Burns   35    Male         O+            Asthma
     2            Chad Byrd   61    Male         B-           Obesity
     3    Antonio Frederick   49    Male         B-            Asthma
     4  Mrs. Brandy Flowers   51    Male         O-         Arthritis

       Date of Admission          Doctor                   Hospital  \
     0        2022-11-17  Patrick Parker           Wallace-Hamilton
     1        2023-06-01   Diane Jackson  Burke, Griffin and Cooper
     2        2019-01-09      Paul Baker                 Walton LLC
     3        2020-05-02  Brian Chandler                 Garcia Ltd
     4        2021-07-09  Dustin Griffin    Jones, Brown and Murray

       Insurance Provider  Billing Amount  Room Number Admission Type  \
     0           Medicare    37490.983364          146       Elective
     1   UnitedHealthcare    47304.064845          404      Emergency
```

```
2           Medicare    36874.896997          292      Emergency
3           Medicare    23303.322092          480         Urgent
4   UnitedHealthcare    18086.344184          477         Urgent

  Discharge Date   Medication  Test Results
0     2022-12-01      Aspirin  Inconclusive
1     2023-06-15      Lipitor        Normal
2     2019-02-08      Lipitor        Normal
3     2020-05-03   Penicillin      Abnormal
4     2021-08-02  Paracetamol        Normal
```

df.tail() method is used to display the last few rows of a DataFrame

[ ]:
```
df.tail()
```

[ ]:
```
                     Name  Age  Gender Blood Type Medical Condition  \
9995           James Hood   83    Male         A+           Obesity
9996      Stephanie Evans   47  Female        AB+         Arthritis
9997  Christopher Martinez   54    Male         B-         Arthritis
9998          Amanda Duke   84    Male         A+         Arthritis
9999            Eric King   20    Male         B-         Arthritis

     Date of Admission             Doctor                   Hospital  \
9995        2022-07-29      Samuel Moody   Wood, Martin and Simmons
9996        2022-01-06  Christopher Yates               Nash-Krueger
9997        2022-07-01   Robert Nicholson            Larson and Sons
9998        2020-02-06        Jamie Lewis               Wilson-Lyons
9999        2023-03-22        Tasha Avila  Torres, Young and Stewart

     Insurance Provider  Billing Amount  Room Number Admission Type  \
9995   UnitedHealthcare    39606.840083          110       Elective
9996          Blue Cross     5995.717488          244      Emergency
9997          Blue Cross    49559.202905          312       Elective
9998   UnitedHealthcare    25236.344761          420         Urgent
9999               Aetna    37223.965865          290      Emergency

     Discharge Date  Medication Test Results
9995     2022-08-02   Ibuprofen     Abnormal
9996     2022-01-29   Ibuprofen       Normal
9997     2022-07-15   Ibuprofen       Normal
9998     2020-02-26  Penicillin       Normal
9999     2023-04-15  Penicillin     Abnormal
```

The code df.isnull().count() in Pandas is used to count the total number of rows for each column in a DataFrame, including both missing (null or NaN) and non-missing values.

[ ]:
```
df.isnull().count()
```

[ ]:
```
Name                  10000
Age                   10000
Gender                10000
Blood Type            10000
Medical Condition     10000
Date of Admission     10000
Doctor                10000
Hospital              10000
Insurance Provider    10000
Billing Amount        10000
Room Number           10000
Admission Type        10000
Discharge Date        10000
Medication            10000
Test Results          10000
dtype: int64
```

df.info() method in Pandas provides a concise summary of a DataFrame, including information about the data types, non-null values, and memory usage.

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                10000 non-null  object
 1   Age                 10000 non-null  int64
 2   Gender              10000 non-null  object
 3   Blood Type          10000 non-null  object
 4   Medical Condition   10000 non-null  object
 5   Date of Admission   10000 non-null  object
 6   Doctor              10000 non-null  object
 7   Hospital            10000 non-null  object
 8   Insurance Provider  10000 non-null  object
 9   Billing Amount      10000 non-null  float64
 10  Room Number         10000 non-null  int64
 11  Admission Type      10000 non-null  object
 12  Discharge Date      10000 non-null  object
 13  Medication          10000 non-null  object
 14  Test Results        10000 non-null  object
dtypes: float64(1), int64(2), object(12)
memory usage: 1.1+ MB
```

The df.describe() method in Pandas is used to generate descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution

[ ]:
```
df.describe()
```

[ ]:
```
                Age  Billing Amount   Room Number
count  10000.000000    10000.000000  10000.000000
mean      51.452200    25516.806778    300.082000
std       19.588974    14067.292709    115.806027
min       18.000000     1000.180837    101.000000
25%       35.000000    13506.523967    199.000000
50%       52.000000    25258.112566    299.000000
75%       68.000000    37733.913727    400.000000
max       85.000000    49995.902283    500.000000
```

Calculate basic descriptive statistics (mean, median, mode, standard deviation, min, max, quartiles, etc.

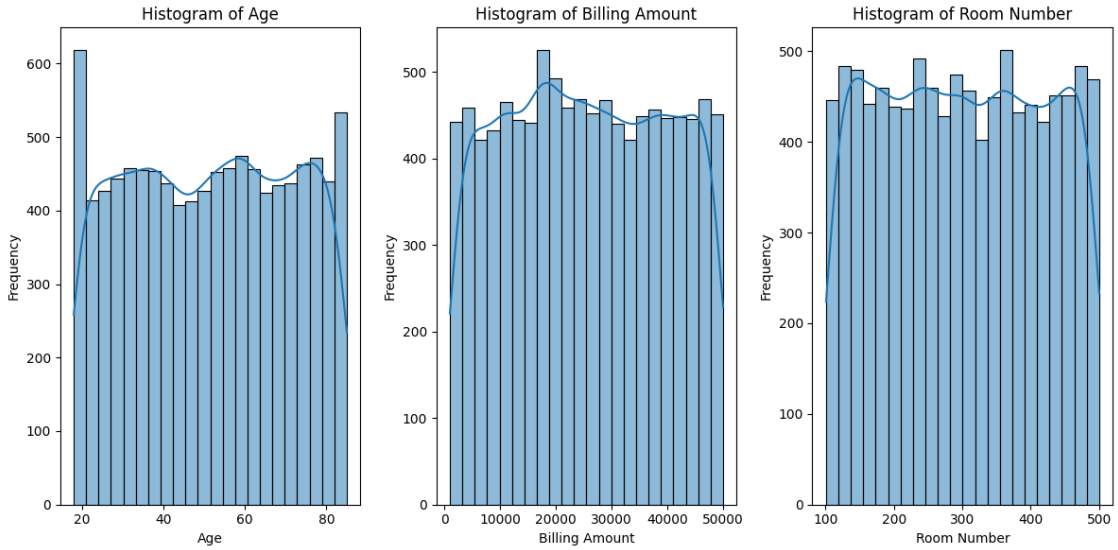
[ ]:
```
# Convert Billing Amount to numeric
df['Billing Amount'] = pd.to_numeric(df['Billing Amount'], errors='coerce')

# Calculate basic descriptive statistics
basic_stats = df.describe()

# Calculate mode
mode = df.mode(dropna=True)

# Add mode row to the statistics DataFrame
basic_stats.loc['mode'] = mode.iloc[0]

print("Basic Descriptive Statistics:")
print(basic_stats)
```

```
Basic Descriptive Statistics:
                Age  Billing Amount   Room Number
count  10000.000000    10000.000000  10000.000000
mean      51.452200    25516.806778    300.082000
std       19.588974    14067.292709    115.806027
min       18.000000     1000.180837    101.000000
25%       35.000000    13506.523967    199.000000
50%       52.000000    25258.112566    299.000000
75%       68.000000    37733.913727    400.000000
max       85.000000    49995.902283    500.000000
mode      59.000000     1000.180837    358.000000
```

Visualize the distribution using histograms, kernel density plots, or box plots.

[ ]:
```
import matplotlib.pyplot as plt

# Select numerical columns for visualization
numerical_columns = ['Age', 'Billing Amount', 'Room Number']
```

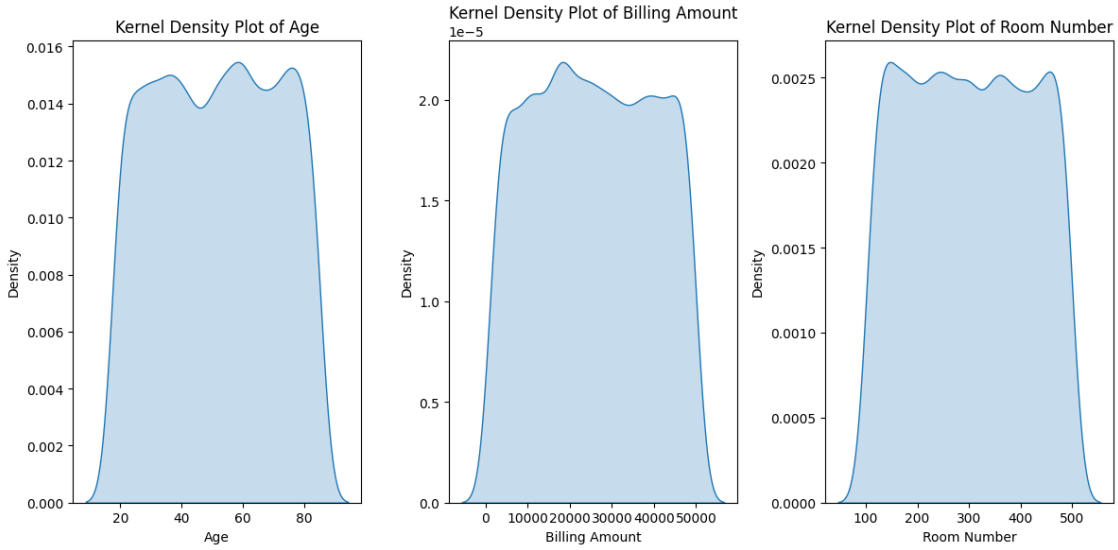
```
# Plot histograms for numerical columns
plt.figure(figsize=(12, 6))
for i, column in enumerate(numerical_columns, 1):
    plt.subplot(1, len(numerical_columns), i)
    sns.histplot(df[column], kde=True)
    plt.title(f'Histogram of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



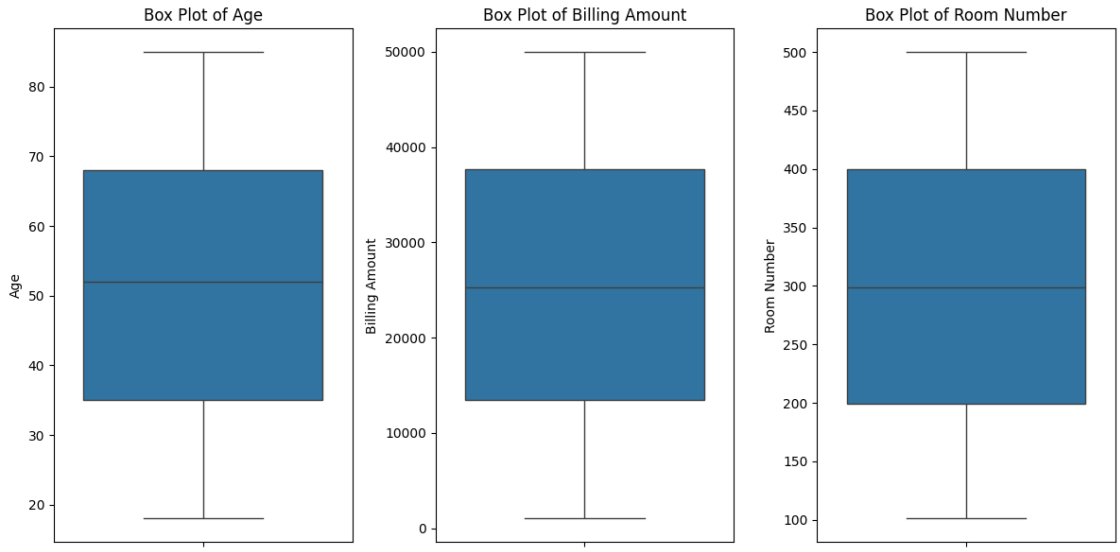
[ ]:
```
# Plot kernel density plots for numerical columns
plt.figure(figsize=(12, 6))
for i, column in enumerate(numerical_columns, 1):
    plt.subplot(1, len(numerical_columns), i)
    sns.kdeplot(df[column], shade=True)
    plt.title(f'Kernel Density Plot of {column}')
    plt.xlabel(column)
    plt.ylabel('Density')

plt.tight_layout()
plt.show()
```

```
<ipython-input-16-334fd245f0ef>:5: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.
```

```
  sns.kdeplot(df[column], shade=True)
<ipython-input-16-334fd245f0ef>:5: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[column], shade=True)
<ipython-input-16-334fd245f0ef>:5: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[column], shade=True)
```



[ ]:
```python
# Plot box plots for numerical columns
plt.figure(figsize=(12, 6))
for i, column in enumerate(numerical_columns, 1):
    plt.subplot(1, len(numerical_columns), i)
    sns.boxplot(y=df[column])
    plt.title(f'Box Plot of {column}')
    plt.ylabel(column)

plt.tight_layout()
plt.show()
```



For categorical variables: i. Display frequency tables showing counts and percentages.

[ ]:
```
# Define categorical variables
categorical_variables = ["Name", "Gender", "Blood Type", "Medical Condition",
                         "Doctor", "Hospital", "Insurance Provider",
                         "Admission Type", "Medication", "Test Results"]

# Display frequency tables for each categorical variable
for column in categorical_variables:
    print(f"Frequency table for {column}:")
    frequency_table = df[column].value_counts()
    frequency_table_percentage = (df[column].value_counts(normalize=True) *␣
 ↪100).round(2)
    frequency_table_df = pd.DataFrame({'Count': frequency_table, 'Percentage':␣
 ↪frequency_table_percentage})
    print(frequency_table_df)
    print("\n")
```

```
Frequency table for Name:
                    Count  Percentage
Name
Michael Johnson         7        0.07
James Johnson           6        0.06
Michael Miller          4        0.04
Michelle Williams       4        0.04
Scott Smith             4        0.04
…                     …          …
Sharon Rose             1        0.01
Stephanie Knox          1        0.01
```

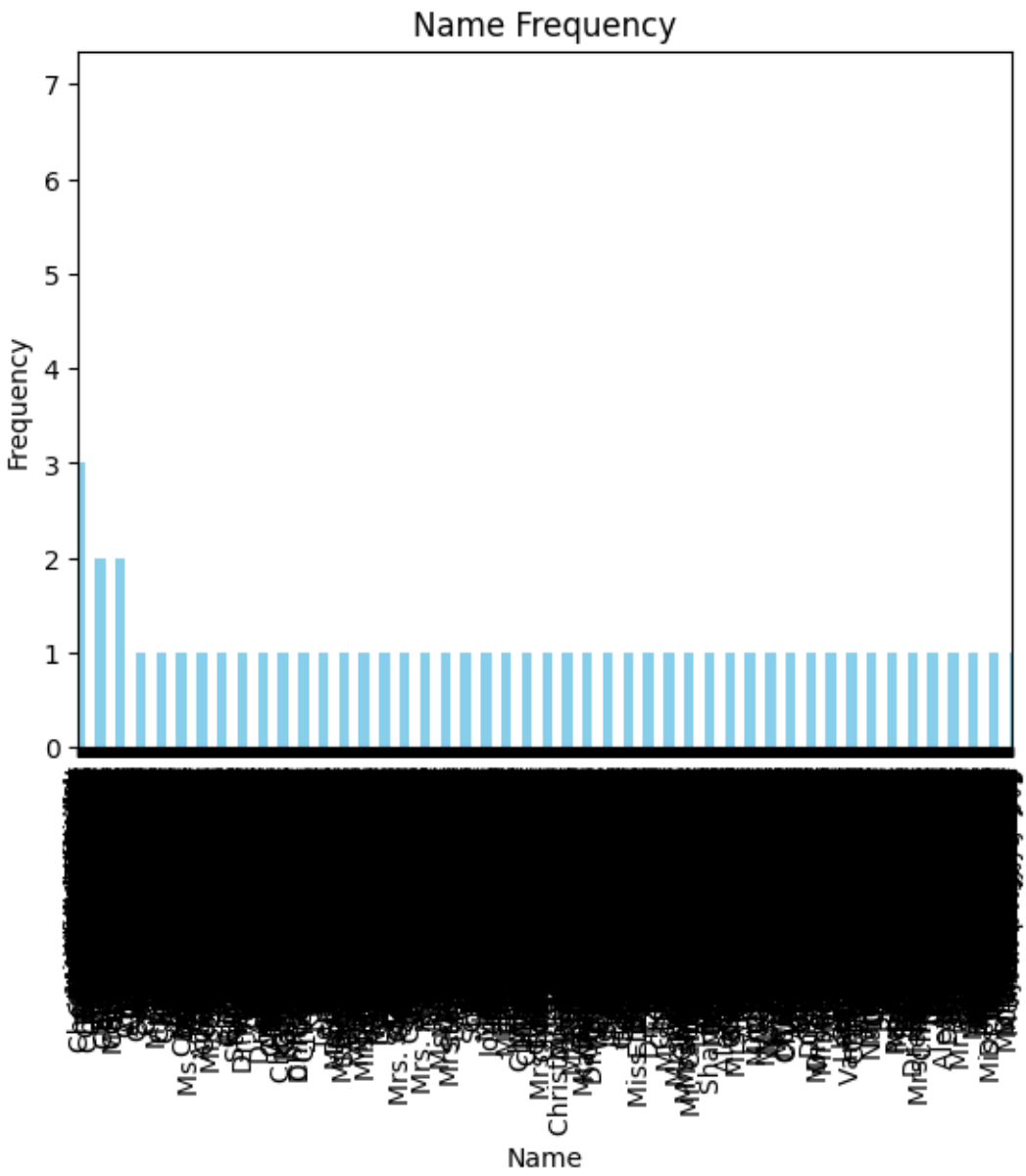
```
Anthony Jones            1        0.01
Melissa Perkins DVM      1        0.01
Eric King                1        0.01

[9378 rows x 2 columns]


Frequency table for Gender:
        Count  Percentage
Gender
Female   5075       50.75
Male     4925       49.25


Frequency table for Blood Type:
            Count  Percentage
Blood Type
AB-          1275       12.75
AB+          1258       12.58
B-           1252       12.52
O+           1248       12.48
O-           1244       12.44
B+           1244       12.44
A+           1241       12.41
A-           1238       12.38


Frequency table for Medical Condition:
                  Count  Percentage
Medical Condition
Asthma             1708       17.08
Cancer             1703       17.03
Hypertension       1688       16.88
Arthritis          1650       16.50
Obesity            1628       16.28
Diabetes           1623       16.23


Frequency table for Doctor:
                  Count  Percentage
Doctor
Michael Johnson       7        0.07
Robert Brown          5        0.05
Michelle Anderson     5        0.05
Matthew Smith         5        0.05
Jennifer Smith        5        0.05
…                   …          …
Sandra Howard         1        0.01
```

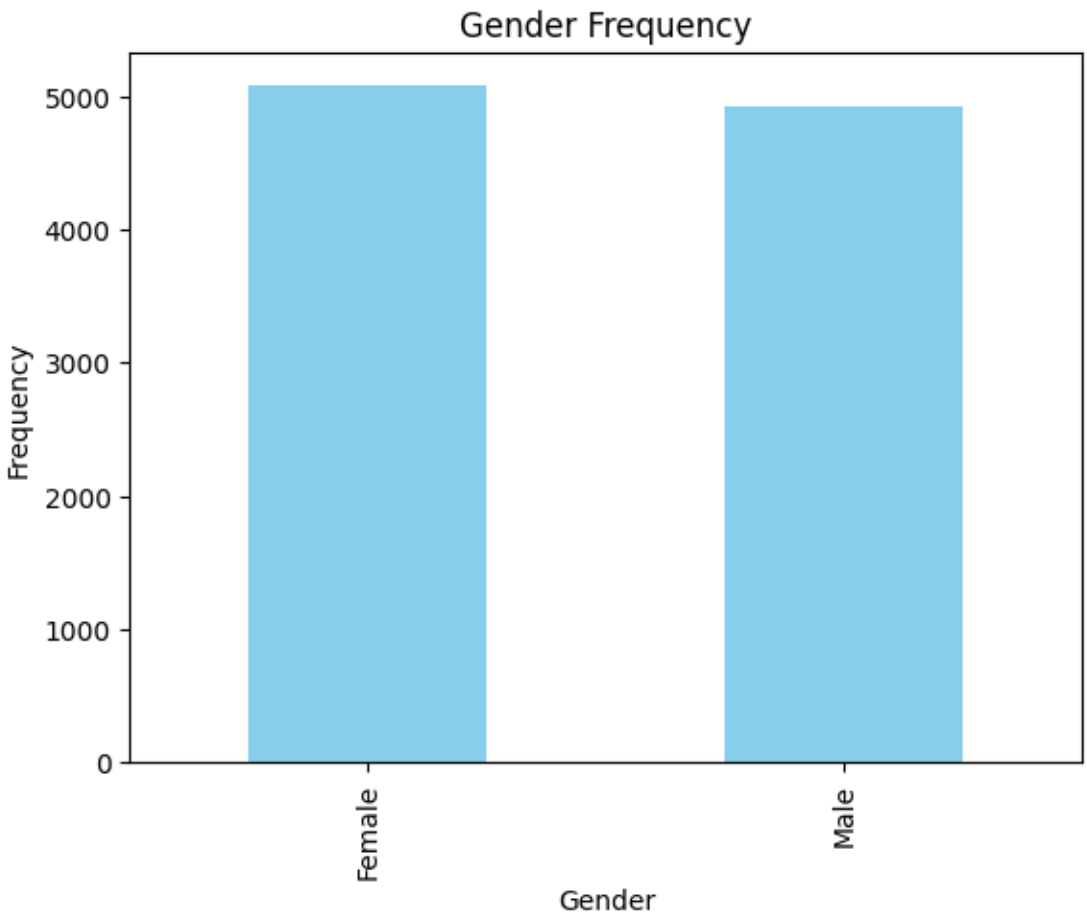
```
Steven Fuller          1        0.01
Benjamin Lawson        1        0.01
Allison Woods          1        0.01
Tasha Avila            1        0.01

[9416 rows x 2 columns]


Frequency table for Hospital:
                             Count  Percentage
Hospital
Smith PLC                       19        0.19
Smith and Sons                  17        0.17
Smith Ltd                       14        0.14
Smith Inc                       14        0.14
Johnson PLC                     13        0.13
…                               …           …
Daniel-Benton                    1        0.01
Franco, Hicks and Anderson       1        0.01
Berry PLC                        1        0.01
Martinez, Johnson and Carlson    1        0.01
Torres, Young and Stewart        1        0.01

[8639 rows x 2 columns]


Frequency table for Insurance Provider:
                   Count  Percentage
Insurance Provider
Cigna               2040       20.40
Blue Cross          2032       20.32
Aetna               2025       20.25
UnitedHealthcare    1978       19.78
Medicare            1925       19.25


Frequency table for Admission Type:
               Count  Percentage
Admission Type
Urgent          3391       33.91
Emergency       3367       33.67
Elective        3242       32.42


Frequency table for Medication:
            Count  Percentage
Medication
Penicillin   2079       20.79
```

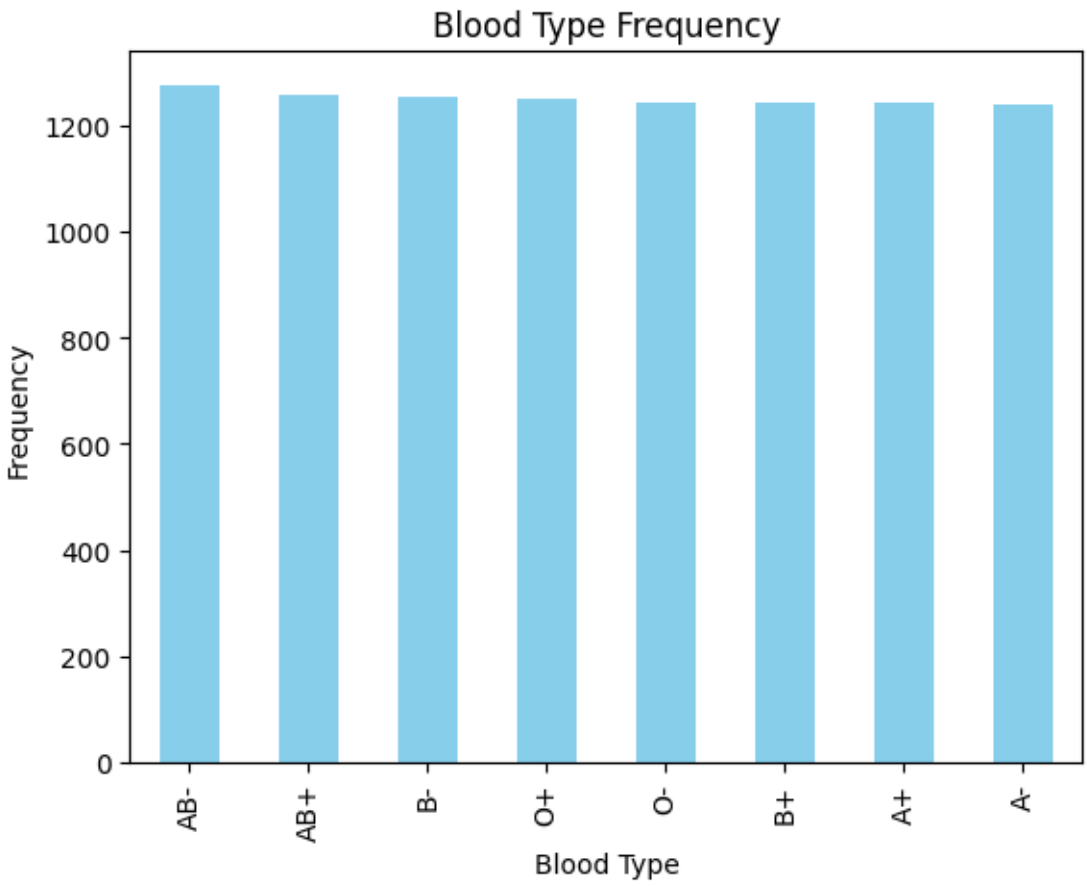
```
Lipitor       2015       20.15
Ibuprofen     1976       19.76
Aspirin       1968       19.68
Paracetamol   1962       19.62


Frequency table for Test Results:
              Count  Percentage
Test Results
Abnormal       3456       34.56
Inconclusive   3277       32.77
Normal         3267       32.67
```

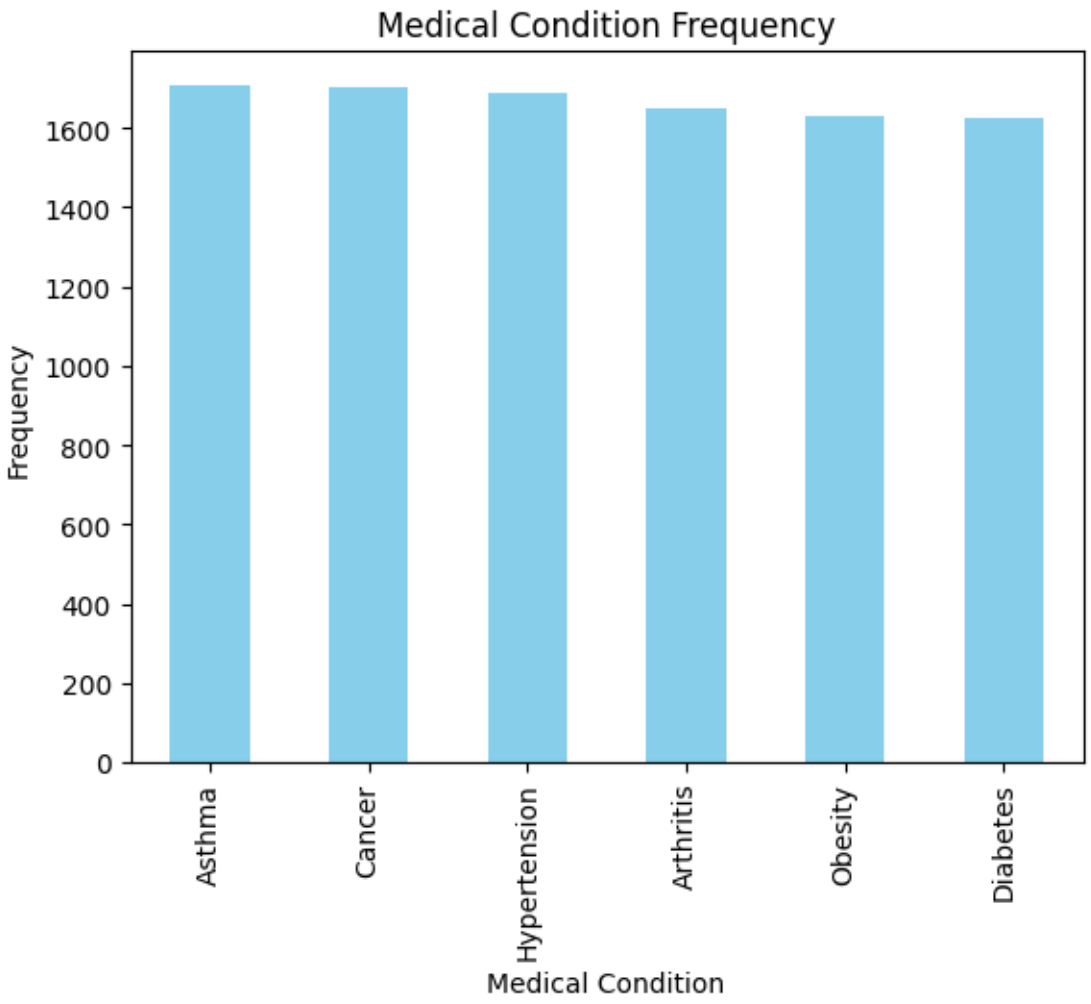
Visualize using bar plots.

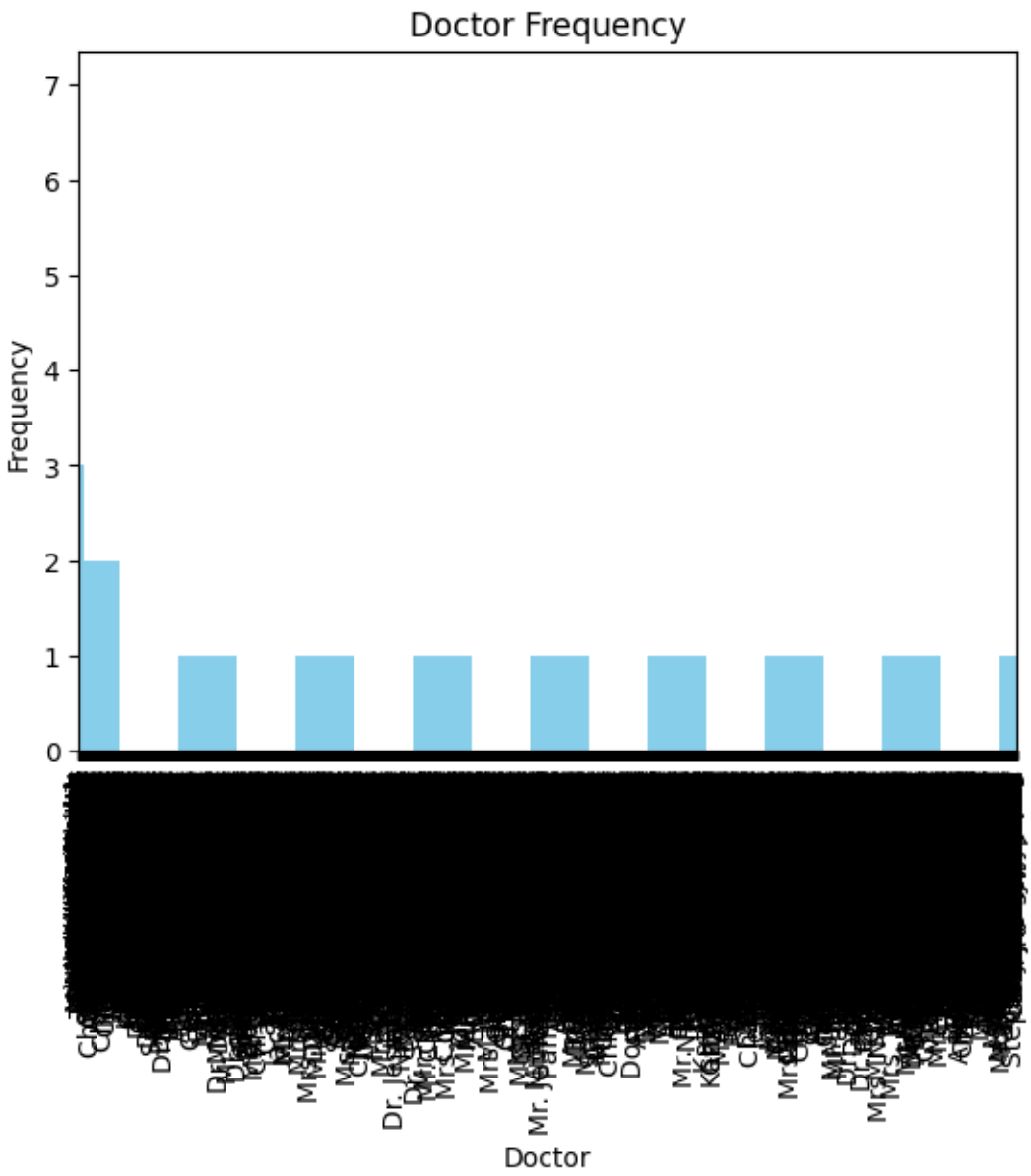
[ ]:
```python
# Define categorical variables
categorical_variables = ["Name", "Gender", "Blood Type", "Medical Condition",
                         "Doctor", "Hospital", "Insurance Provider",
                         "Admission Type", "Medication", "Test Results"]

# Plot frequency of each categorical variable
for column in categorical_variables:
    frequency_table = df[column].value_counts()
    frequency_table.plot(kind='bar', color='skyblue')
    plt.title(f"{column} Frequency")
    plt.xlabel(column)
    plt.ylabel("Frequency")
    plt.show()
```

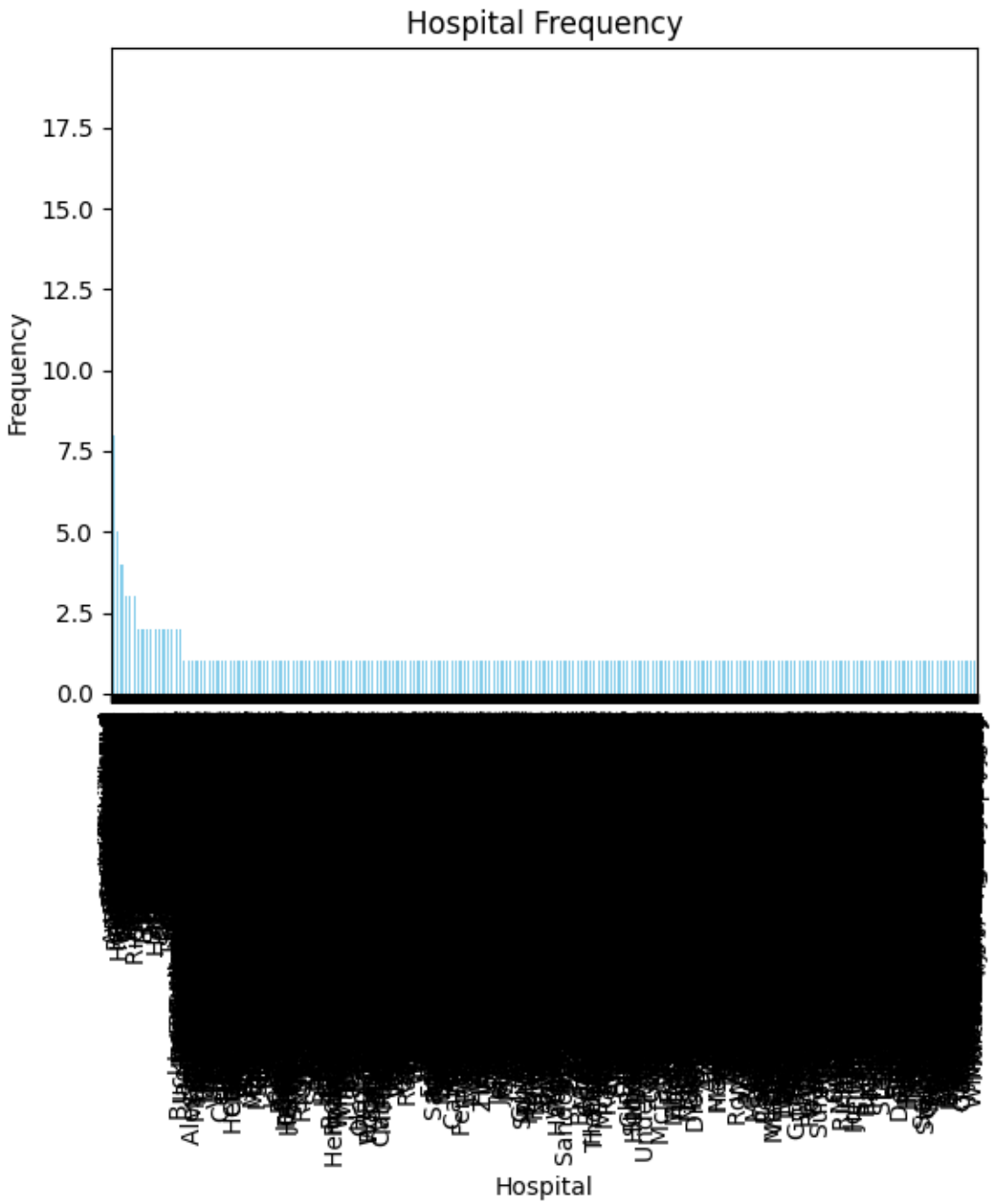

Name Frequency
Frequency
0
1
2
3
4
5
6
7
Name

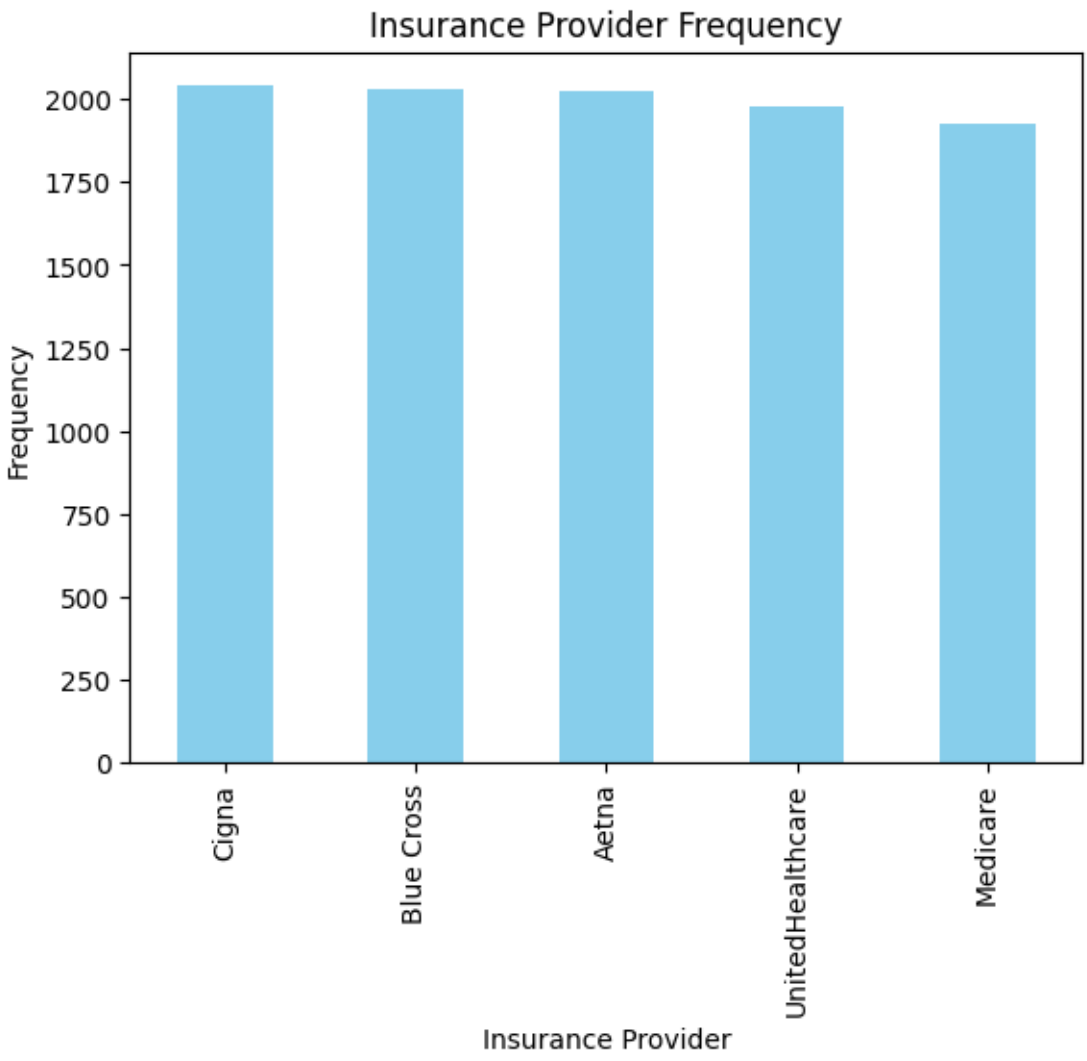

Gender Frequency
5000
4000
3000
2000
1000
0
Frequency
Female
Male
Gender

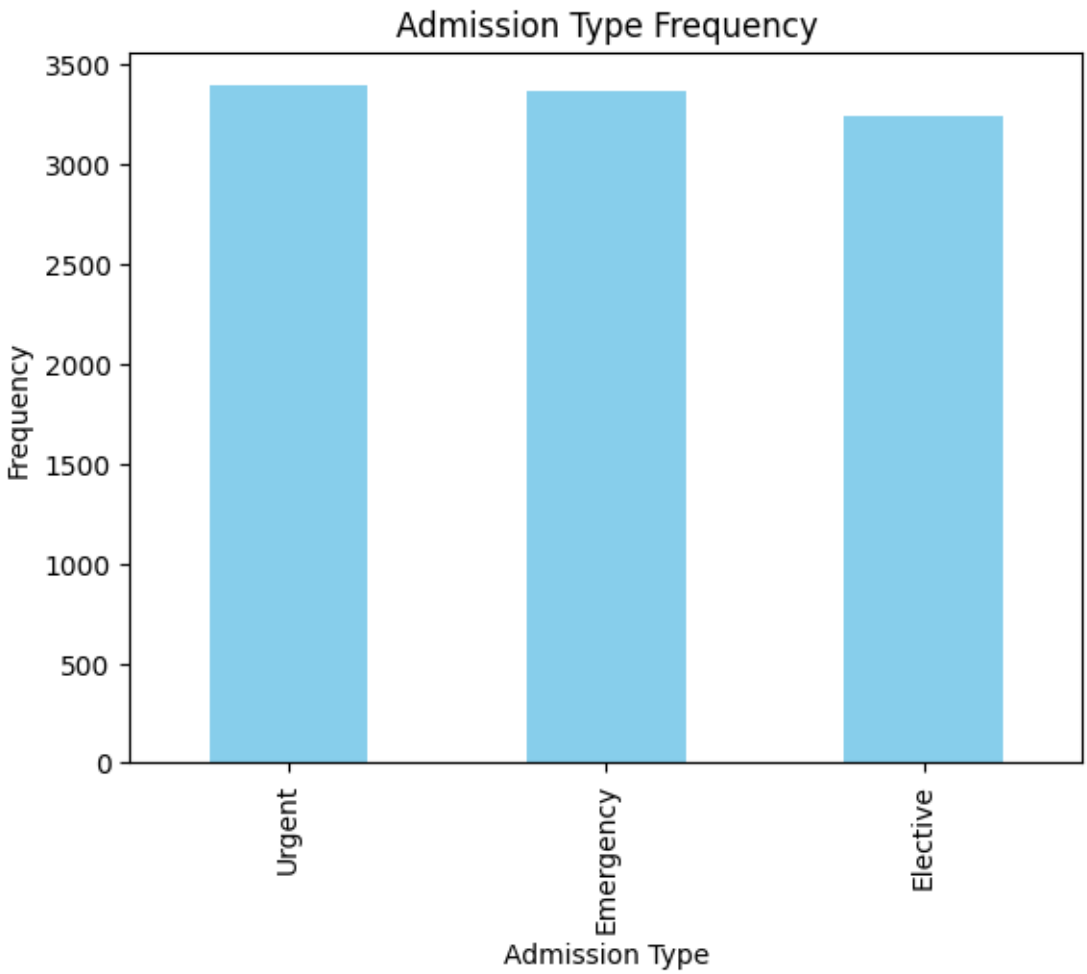

Blood Type Frequency
Frequency
0
200
400
600
800
1000
1200
AB-
AB+
B-
O+
O-
B+
A+
A-
Blood Type

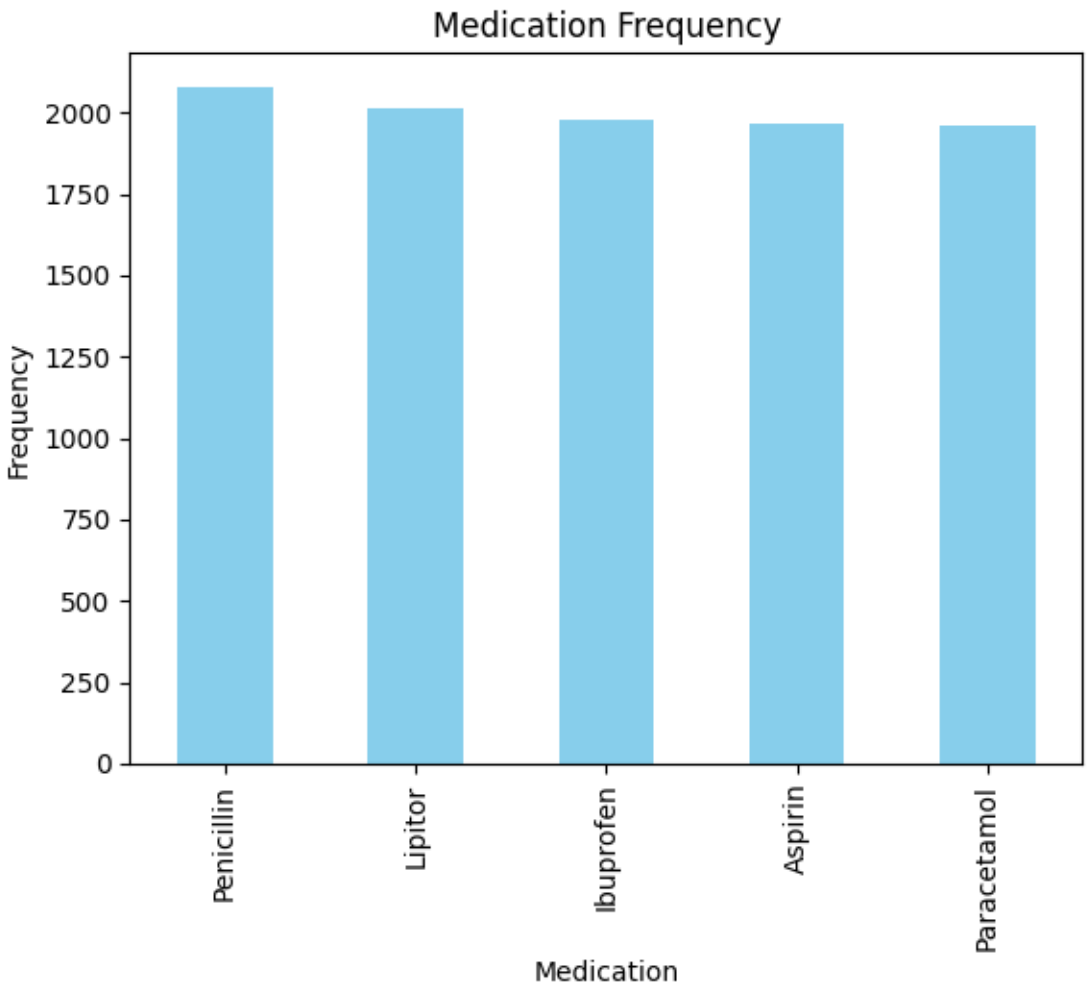

Medical Condition Frequency
Frequency
0
200
400
600
800
1000
1200
1400
1600
Asthma
Cancer
Hypertension
Arthritis
Obesity
Diabetes
Medical Condition

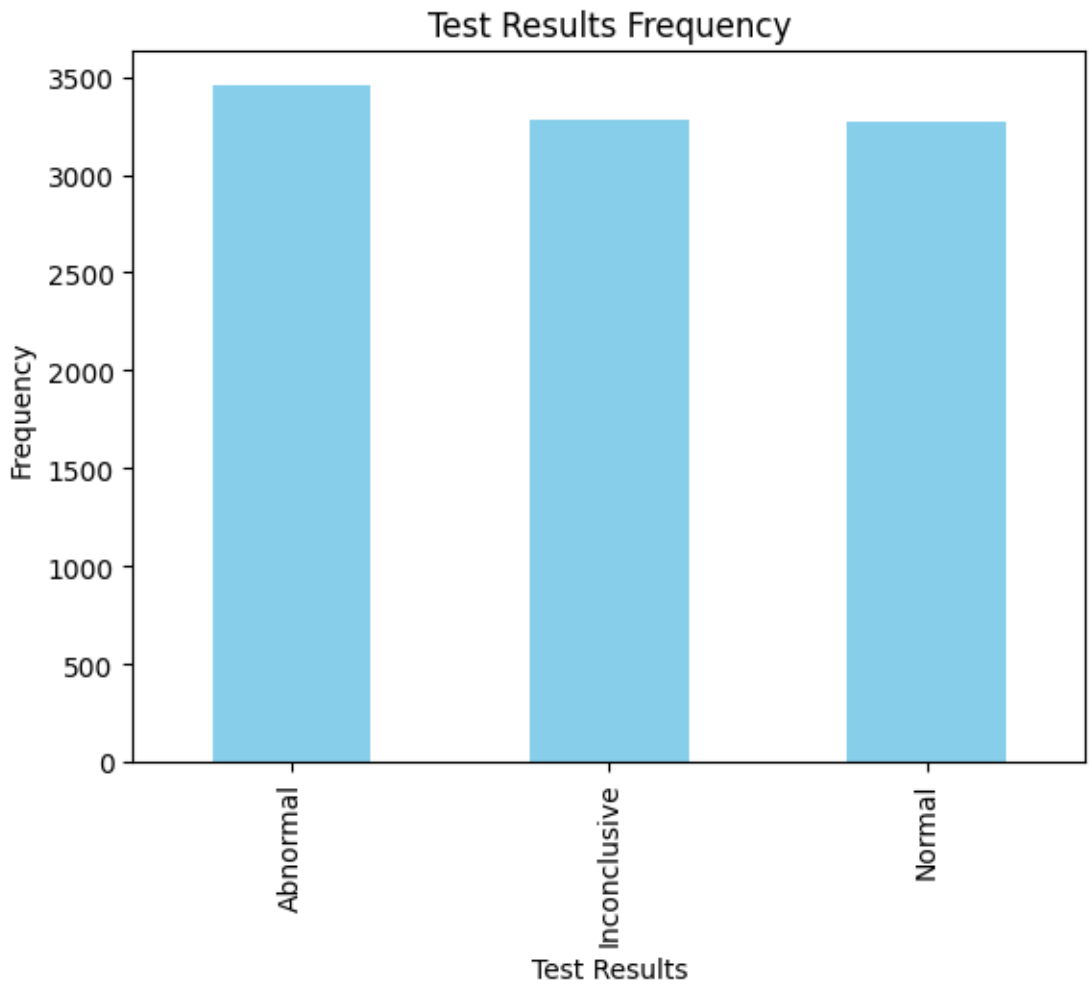

Doctor Frequency
Frequency
0
1
2
3
4
5
6
7
Doctor


Hospital Frequency
Frequency
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Hospital


Insurance Provider Frequency
Frequency
2000
1750
1500
1250
1000
750
500
250
0
Cigna
Blue Cross
Aetna
UnitedHealthcare
Medicare
Insurance Provider

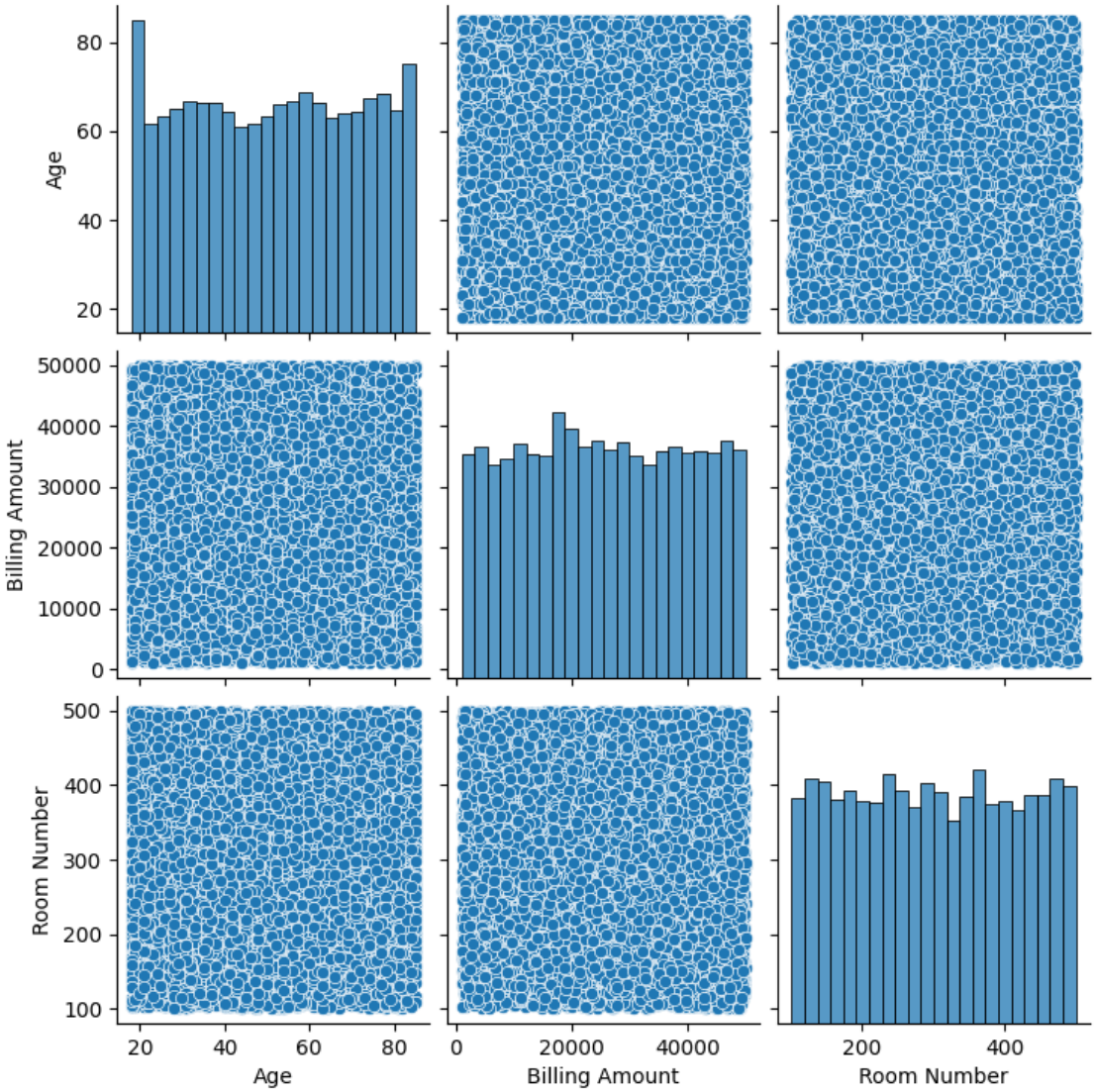

Admission Type Frequency
3500
3000
2500
2000
1500
1000
500
0
Frequency
Urgent
Emergency
Elective
Admission Type


Medication Frequency
Frequency
2000
1750
1500
1250
1000
750
500
250
0
Penicillin
Lipitor
Ibuprofen
Aspirin
Paracetamol
Medication



Bivariate Analysis: a. Explore relationships between pairs of numerical variables using scatter plots or pair plots.

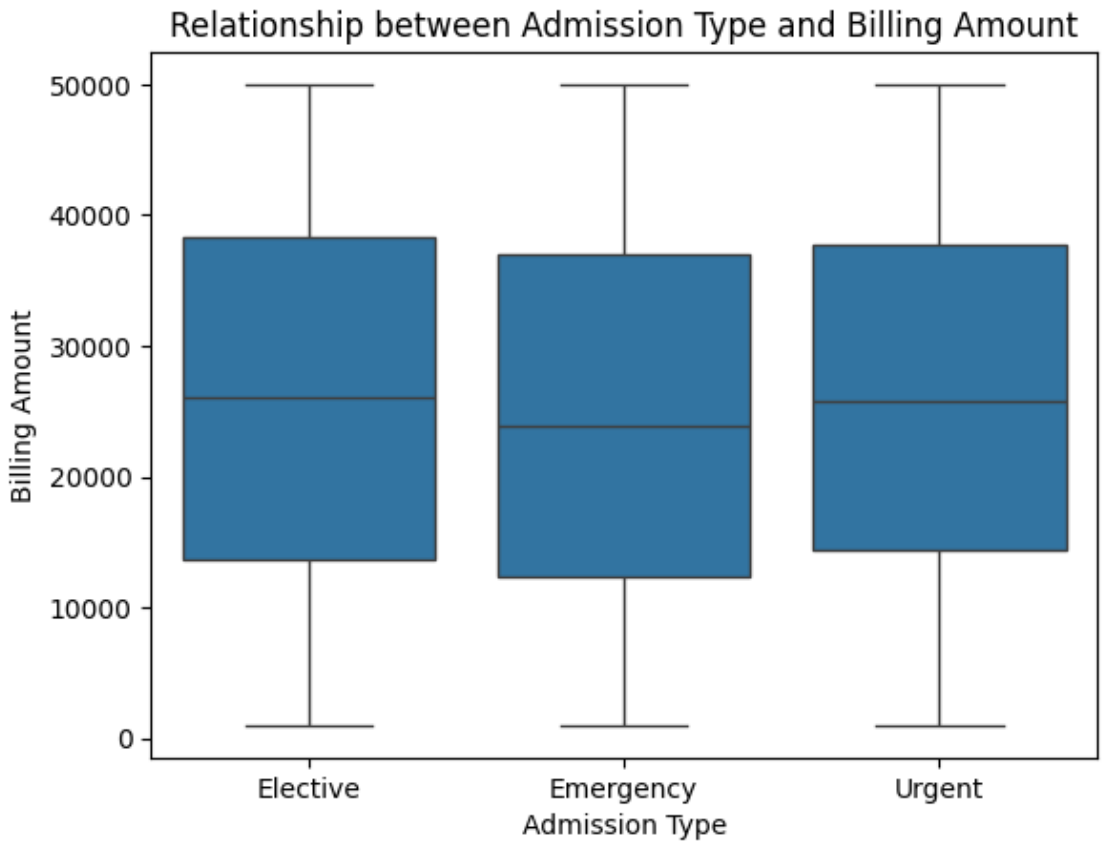
[ ]:
```
# Select numerical variables
numerical_variables = df.select_dtypes(include=['int64', 'float64'])

# Plot pairplot to explore relationships between numerical variables
sns.pairplot(numerical_variables)
plt.show()
```



Explore relationships between numerical and categorical variables using box plots or violin plots.

[ ]:
```
# Define the numerical and categorical variables you want to explore
numerical_variable = "Billing Amount"
categorical_variable = "Admission Type"

# Plot boxplot to explore relationship between numerical and categorical␣
 ↪variables
sns.boxplot(x=categorical_variable, y=numerical_variable, data=df)
plt.title(f"Relationship between {categorical_variable} and␣
 ↪{numerical_variable}")
plt.xlabel(categorical_variable)
plt.ylabel(numerical_variable)
```

```
plt.show()
```



Calculate correlation coefficients between numerical variables.

[ ]:
```
# Select numerical variables
numerical_variables = df.select_dtypes(include=['int64', 'float64'])

# Calculate correlation coefficients
correlation_matrix = numerical_variables.corr()

# Display correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)
```

```
Correlation Matrix:
                      Age  Billing Amount  Room Number
Age              1.000000       -0.009483    -0.005371
Billing Amount  -0.009483        1.000000    -0.006160
Room Number     -0.005371       -0.006160     1.000000
```

Drop the non-required columns/features (dependent columns) if necessary.

[ ]:
```
# Drop the 'Name' column
df.drop(columns=['Name'], inplace=True)

# Print the updated DataFrame to verify changes
print(df)
```

```
      Age  Gender Blood Type Medical Condition Date of Admission  \
0      81  Female         O-          Diabetes        2022-11-17
1      35    Male         O+            Asthma        2023-06-01
2      61    Male         B-           Obesity        2019-01-09
3      49    Male         B-            Asthma        2020-05-02
4      51    Male         O-         Arthritis        2021-07-09
…     …     …        …                …                …
9995   83    Male         A+           Obesity        2022-07-29
9996   47  Female        AB+         Arthritis        2022-01-06
9997   54    Male         B-         Arthritis        2022-07-01
9998   84    Male         A+         Arthritis        2020-02-06
9999   20    Male         B-         Arthritis        2023-03-22

                Doctor                  Hospital Insurance Provider  \
0        Patrick Parker           Wallace-Hamilton           Medicare
1         Diane Jackson  Burke, Griffin and Cooper   UnitedHealthcare
2            Paul Baker                Walton LLC           Medicare
3        Brian Chandler                Garcia Ltd           Medicare
4        Dustin Griffin   Jones, Brown and Murray   UnitedHealthcare
…                 …                       …                …
9995       Samuel Moody  Wood, Martin and Simmons   UnitedHealthcare
9996  Christopher Yates              Nash-Krueger         Blue Cross
9997   Robert Nicholson           Larson and Sons         Blue Cross
9998        Jamie Lewis              Wilson-Lyons   UnitedHealthcare
9999        Tasha Avila  Torres, Young and Stewart              Aetna

      Billing Amount  Room Number Admission Type Discharge Date   Medication  \
0       37490.983364          146       Elective     2022-12-01      Aspirin
1       47304.064845          404      Emergency     2023-06-15      Lipitor
2       36874.896997          292      Emergency     2019-02-08      Lipitor
3       23303.322092          480         Urgent     2020-05-03   Penicillin
4       18086.344184          477         Urgent     2021-08-02  Paracetamol
…               …           …              …             …          …
9995    39606.840083          110       Elective     2022-08-02    Ibuprofen
9996     5995.717488          244      Emergency     2022-01-29    Ibuprofen
9997    49559.202905          312       Elective     2022-07-15    Ibuprofen
9998    25236.344761          420         Urgent     2020-02-26   Penicillin
9999    37223.965865          290      Emergency     2023-04-15   Penicillin

      Test Results
0     Inconclusive
```

```
1           Normal
2           Normal
3         Abnormal
4           Normal
…             …
9995      Abnormal
9996        Normal
9997        Normal
9998        Normal
9999      Abnormal

[10000 rows x 14 columns]
```

print columns/features names.

[ ]:
```
# Print out the columns
print(df.columns)
```

```
Index(['Age', 'Gender', 'Blood Type', 'Medical Condition', 'Date of Admission',
       'Doctor', 'Hospital', 'Insurance Provider', 'Billing Amount',
       'Room Number', 'Admission Type', 'Discharge Date', 'Medication',
       'Test Results'],
      dtype='object')
```

Perform Standardization or normalization on the features as required.

[ ]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Load the dataset
df = pd.read_csv('/content/healthcare_dataset.csv')

columns_to_standardize = ['Age', 'Billing Amount']

# Standardization
scaler = StandardScaler()
df[columns_to_standardize] = scaler.fit_transform(df[columns_to_standardize])

# Print the processed dataframe
print(df)
```

```
                    Name       Age  Gender Blood Type Medical Condition  \
0          Tiffany Ramirez  1.508465  Female         O-          Diabetes
1              Ruben Burns -0.839912    Male         O+            Asthma
2                Chad Byrd  0.487431    Male         B-           Obesity
3        Antonio Frederick -0.125189    Male         B-            Asthma
4      Mrs. Brandy Flowers -0.023086    Male         O-         Arthritis
…                      …       …     …       …                …
9995            James Hood  1.610568    Male         A+           Obesity
```

```
9996       Stephanie Evans -0.227292  Female        AB+         Arthritis
9997  Christopher Martinez  0.130069    Male         B-         Arthritis
9998           Amanda Duke  1.661620    Male         A+         Arthritis
9999             Eric King -1.605688    Male         B-         Arthritis

     Date of Admission             Doctor                    Hospital  \
0           2022-11-17     Patrick Parker           Wallace-Hamilton
1           2023-06-01      Diane Jackson  Burke, Griffin and Cooper
2           2019-01-09         Paul Baker                 Walton LLC
3           2020-05-02     Brian Chandler                 Garcia Ltd
4           2021-07-09     Dustin Griffin    Jones, Brown and Murray
…                  …                  …                         …
9995        2022-07-29       Samuel Moody   Wood, Martin and Simmons
9996        2022-01-06  Christopher Yates               Nash-Krueger
9997        2022-07-01   Robert Nicholson            Larson and Sons
9998        2020-02-06        Jamie Lewis               Wilson-Lyons
9999        2023-03-22        Tasha Avila  Torres, Young and Stewart

     Insurance Provider  Billing Amount  Room Number Admission Type  \
0             Medicare        0.851249          146       Elective
1      UnitedHealthcare        1.548866          404      Emergency
2             Medicare        0.807452          292      Emergency
3             Medicare       -0.157358          480         Urgent
4      UnitedHealthcare       -0.528235          477         Urgent
…                   …             …            …             …
9995   UnitedHealthcare        1.001667          110       Elective
9996         Blue Cross       -1.387763          244      Emergency
9997         Blue Cross        1.709184          312       Elective
9998   UnitedHealthcare       -0.019938          420         Urgent
9999              Aetna        0.832267          290      Emergency

     Discharge Date   Medication  Test Results
0        2022-12-01      Aspirin  Inconclusive
1        2023-06-15      Lipitor        Normal
2        2019-02-08      Lipitor        Normal
3        2020-05-03   Penicillin      Abnormal
4        2021-08-02  Paracetamol        Normal
…               …            …             …
9995     2022-08-02    Ibuprofen      Abnormal
9996     2022-01-29    Ibuprofen        Normal
9997     2022-07-15    Ibuprofen        Normal
9998     2020-02-26   Penicillin        Normal
9999     2023-04-15   Penicillin      Abnormal

[10000 rows x 15 columns]
```

Split the dataset into training and testing sets.

```
[ ]: import pandas as pd
     from sklearn.model_selection import train_test_split

     # Load the dataset
     df = pd.read_csv('/content/healthcare_dataset.csv')

     # Define features (X) and target variable (y)
     X = df.drop(columns=df.columns[-1])
     y = df.iloc[:, -1]

     # Split the dataset into training and testing sets (80% train, 20% test)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
      ↪random_state=42)

     # Print the shapes of the training and testing sets to verify the split
     print("Training set shape:", X_train.shape, y_train.shape)
     print("Testing set shape:", X_test.shape, y_test.shape)
```

```
Training set shape: (8000, 14) (8000,)
Testing set shape: (2000, 14) (2000,)
```

11. Implement Linear Discriminant Analysis (LDA):

    a. Train the LDA model using the training data.

```
[ ]: # Importing necessary libraries
     import pandas as pd
     from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import LabelEncoder

     # Load the dataset
     data = pd.read_csv('/content/healthcare_dataset.csv')
     # Drop unnecessary columns like Name, Date of Admission, Doctor, Hospital,␣
      ↪Insurance Provider, Room Number, Medication, and Test Results
     data = data.drop(['Name', 'Date of Admission', 'Doctor', 'Hospital', 'Insurance␣
      ↪Provider', 'Room Number', 'Medication', 'Test Results', 'Discharge Date'],␣
      ↪axis=1)

     # Encoding categorical variables
     label_encoder = LabelEncoder()
     data['Gender'] = label_encoder.fit_transform(data['Gender'])
     data['Blood Type'] = label_encoder.fit_transform(data['Blood Type'])
     data['Medical Condition'] = label_encoder.fit_transform(data['Medical␣
      ↪Condition'])

     # Splitting the data into features and target
     X = data.drop(['Admission Type'], axis=1)
```

```
y = data['Admission Type']

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Initialize the LDA model
lda = LinearDiscriminantAnalysis()

# Train the LDA model using the training data
lda.fit(X_train, y_train)

# Optionally, you can evaluate the model using test data
accuracy = lda.score(X_test, y_test)
print("Accuracy of LDA model on test data:", accuracy)
```

```
Accuracy of LDA model on test data: 0.3505
```

b. Evaluate the performance of the trained model using appropriate metrics.

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

# Load the dataset
data = pd.read_csv('/content/healthcare_dataset.csv')

# Drop non-numeric columns and encode categorical columns
X = data.drop(['Medical Condition', 'Doctor', 'Hospital', 'Insurance Provider',␣
 ↪'Admission Type', 'Discharge Date', 'Medication', 'Test Results'], axis=1)
X = pd.get_dummies(X)  # One-hot encode categorical variables
y = data['Medical Condition']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Initialize and fit the Logistic Regression model
logistic_reg = LogisticRegression()
logistic_reg.fit(X_train, y_train)

# Predict the labels for the test set
y_pred = logistic_reg.predict(X_test)

# Evaluate the performance of the model
accuracy = accuracy_score(y_test, y_pred)
```

```python
print("Accuracy:", accuracy)

# Generate a classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.1635
Classification Report:
              precision    recall  f1-score   support

   Arthritis       0.15      0.19      0.17       319
      Asthma       0.17      0.52      0.25       332
      Cancer       0.00      0.00      0.00       363
    Diabetes       0.17      0.31      0.22       303
Hypertension       0.00      0.00      0.00       336
     Obesity       0.00      0.00      0.00       347

    accuracy                           0.16      2000
   macro avg       0.08      0.17      0.11      2000
weighted avg       0.08      0.16      0.10      2000


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Implement Logistic Regression: a. Train the Logistic Regression model using the training data.

[ ]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score

# Load the dataset
data = pd.read_csv('/content/healthcare_dataset.csv')
```

```
# Drop irrelevant columns such as Name, Date of Admission, Doctor, Hospital,␣
 ↪Insurance Provider, Room Number, Admission Type, Discharge Date, Medication,␣
 ↪and Test Results
data = data.drop(['Name', 'Date of Admission', 'Doctor', 'Hospital', 'Insurance␣
 ↪Provider', 'Room Number', 'Admission Type', 'Discharge Date', 'Medication',␣
 ↪'Test Results'], axis=1)

# Convert categorical variables to numerical using Label Encoding
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])
data['Blood Type'] = label_encoder.fit_transform(data['Blood Type'])
data['Medical Condition'] = label_encoder.fit_transform(data['Medical␣
 ↪Condition'])

# Split data into features and target variable
X = data.drop('Medical Condition', axis=1)
y = data['Medical Condition']

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Train the Logistic Regression model
log_reg_model = LogisticRegression()
log_reg_model.fit(X_train, y_train)

# Predict on the testing set
y_pred = log_reg_model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.166
```

Evaluate the performance of the trained model using appropriate metrics.

[ ]:
```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix

# Load the dataset
data = pd.read_csv("/content/healthcare_dataset.csv")
```

```
# Data preprocessing
# Drop unnecessary columns like Name, Date of Admission, Doctor, Hospital,␣
 ↪Insurance Provider, Medication, Test Results
data = data.drop(columns=['Name', 'Date of Admission', 'Doctor', 'Hospital',␣
 ↪'Insurance Provider', 'Medication', 'Test Results'])

# Convert categorical variables into numerical ones using one-hot encoding
data = pd.get_dummies(data, columns=['Gender', 'Blood Type', 'Medical␣
 ↪Condition', 'Admission Type'])

# Split data into features (X) and target variable (y)
X = data.drop(columns=['Discharge Date'])  # Features
y = data['Discharge Date']  # Target variable

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Initialize and train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict the target variable on the testing set
y_pred = model.predict(X_test)

# Evaluate the model performance
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("\nConfusion Matrix:\n", conf_matrix)
print("\nClassification Report:\n", class_report)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
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(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
```

```
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))

Accuracy: 0.001

Confusion Matrix:
 [[0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 …
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]]

Classification Report:
              precision    recall  f1-score   support

  2018-11-05       0.00      0.00      0.00         1
  2018-11-07       0.00      0.00      0.00         1
  2018-11-09       0.00      0.00      0.00         1
  2018-11-13       0.00      0.00      0.00         1
  2018-11-17       0.00      0.00      0.00         3
  2018-11-18       0.00      0.00      0.00         1
  2018-11-20       0.00      0.00      0.00         1
  2018-11-22       0.00      0.00      0.00         1
  2018-11-23       0.00      0.00      0.00         1
  2018-11-25       0.00      0.00      0.00         2
  2018-11-26       0.00      0.00      0.00         3
  2018-11-27       0.00      0.00      0.00         1
  2018-11-28       0.00      0.00      0.00         1
  2018-11-29       0.00      0.00      0.00         1
  2018-12-01       0.00      0.00      0.00         1
```

2018-12-02	0.00	0.00	0.00	2
2018-12-03	0.00	0.00	0.00	1
2018-12-04	0.00	0.00	0.00	1
2018-12-06	0.00	0.00	0.00	1
2018-12-08	0.00	0.00	0.00	1
2018-12-09	0.00	0.00	0.00	1
2018-12-10	0.00	0.00	0.00	1
2018-12-12	0.00	0.00	0.00	1
2018-12-13	0.00	0.00	0.00	1
2018-12-14	0.00	0.00	0.00	2
2018-12-15	0.00	0.00	0.00	1
2018-12-16	0.00	0.00	0.00	1
2018-12-18	0.00	0.00	0.00	1
2018-12-19	0.00	0.00	0.00	1
2018-12-20	0.00	0.00	0.00	1
2018-12-21	0.00	0.00	0.00	1
2018-12-23	0.00	0.00	0.00	1
2018-12-25	0.00	0.00	0.00	1
2018-12-27	0.00	0.00	0.00	1
2018-12-28	0.00	0.00	0.00	2
2018-12-31	0.00	0.00	0.00	3
2019-01-01	0.00	0.00	0.00	1
2019-01-03	0.00	0.00	0.00	2
2019-01-05	0.00	0.00	0.00	1
2019-01-07	0.00	0.00	0.00	7
2019-01-08	0.00	0.00	0.00	1
2019-01-09	0.00	0.00	0.00	3
2019-01-10	0.00	0.00	0.00	2
2019-01-12	0.00	0.00	0.00	3
2019-01-13	0.00	0.00	0.00	3
2019-01-14	0.00	0.00	0.00	1
2019-01-15	0.00	0.00	0.00	2
2019-01-16	0.00	0.00	0.00	3
2019-01-17	0.00	0.00	0.00	1
2019-01-18	0.00	0.00	0.00	2
2019-01-19	0.00	0.00	0.00	1
2019-01-21	0.00	0.00	0.00	1
2019-01-23	0.00	0.00	0.00	1
2019-01-24	0.00	0.00	0.00	1
2019-01-25	0.00	0.00	0.00	1
2019-01-26	0.00	0.00	0.00	1
2019-01-27	0.00	0.00	0.00	1
2019-01-28	0.00	0.00	0.00	2
2019-01-29	0.00	0.00	0.00	3
2019-01-30	0.00	0.00	0.00	3
2019-01-31	0.00	0.00	0.00	2
2019-02-03	0.00	0.00	0.00	2
2019-02-04	0.00	0.00	0.00	1

2019-02-05	0.00	0.00	0.00	3
2019-02-07	0.00	0.00	0.00	1
2019-02-08	0.00	0.00	0.00	2
2019-02-11	0.00	0.00	0.00	3
2019-02-13	0.00	0.00	0.00	1
2019-02-14	0.00	0.00	0.00	1
2019-02-15	0.00	0.00	0.00	1
2019-02-16	0.00	0.00	0.00	1
2019-02-17	0.00	0.00	0.00	1
2019-02-18	0.00	0.00	0.00	1
2019-02-19	0.00	0.00	0.00	1
2019-02-20	0.00	0.00	0.00	1
2019-02-21	0.00	0.00	0.00	3
2019-02-24	0.00	0.00	0.00	1
2019-02-25	0.00	0.00	0.00	1
2019-02-27	0.00	0.00	0.00	1
2019-03-01	0.00	0.00	0.00	2
2019-03-02	0.00	0.00	0.00	1
2019-03-03	0.00	0.00	0.00	2
2019-03-04	0.00	0.00	0.00	1
2019-03-07	0.00	0.00	0.00	1
2019-03-09	0.00	0.00	0.00	2
2019-03-11	0.00	0.00	0.00	3
2019-03-12	0.00	0.00	0.00	1
2019-03-13	0.00	0.00	0.00	1
2019-03-14	0.00	0.00	0.00	2
2019-03-15	0.00	0.00	0.00	1
2019-03-16	0.00	0.00	0.00	1
2019-03-21	0.00	0.00	0.00	1
2019-03-25	0.00	0.00	0.00	4
2019-03-27	0.00	0.00	0.00	1
2019-03-28	0.00	0.00	0.00	4
2019-04-02	0.00	0.00	0.00	1
2019-04-03	0.00	0.00	0.00	3
2019-04-04	0.00	0.00	0.00	1
2019-04-05	0.00	0.00	0.00	1
2019-04-10	0.00	0.00	0.00	2
2019-04-11	0.00	0.00	0.00	2
2019-04-13	0.00	0.00	0.00	1
2019-04-14	0.00	0.00	0.00	1
2019-04-16	0.00	0.00	0.00	2
2019-04-19	0.00	0.00	0.00	1
2019-04-22	0.00	0.00	0.00	1
2019-04-23	0.00	0.00	0.00	1
2019-04-24	0.00	0.00	0.00	2
2019-04-25	0.00	0.00	0.00	2
2019-04-26	0.00	0.00	0.00	1
2019-04-27	0.00	0.00	0.00	1

2019-04-29	0.00	0.00	0.00	3
2019-05-01	0.00	0.00	0.00	1
2019-05-02	0.00	0.00	0.00	1
2019-05-03	0.00	0.00	0.00	1
2019-05-04	0.00	0.00	0.00	2
2019-05-05	0.00	0.00	0.00	3
2019-05-06	0.00	0.00	0.00	1
2019-05-07	0.00	0.00	0.00	2
2019-05-08	0.00	0.00	0.00	2
2019-05-09	0.00	0.00	0.00	2
2019-05-10	0.00	0.00	0.00	1
2019-05-11	0.00	0.00	0.00	1
2019-05-13	0.00	0.00	0.00	2
2019-05-14	0.00	0.00	0.00	2
2019-05-15	0.00	0.00	0.00	1
2019-05-16	0.00	0.00	0.00	1
2019-05-17	0.00	0.00	0.00	2
2019-05-18	0.00	0.00	0.00	2
2019-05-21	0.00	0.00	0.00	1
2019-05-22	0.00	0.00	0.00	2
2019-05-25	0.00	0.00	0.00	2
2019-05-26	0.00	0.00	0.00	1
2019-05-28	0.00	0.00	0.00	2
2019-05-29	0.00	0.00	0.00	1
2019-05-30	0.00	0.00	0.00	2
2019-06-01	0.00	0.00	0.00	2
2019-06-02	0.00	0.00	0.00	1
2019-06-04	0.00	0.00	0.00	1
2019-06-05	0.00	0.00	0.00	1
2019-06-06	0.00	0.00	0.00	1
2019-06-07	0.00	0.00	0.00	3
2019-06-08	0.00	0.00	0.00	2
2019-06-09	0.00	0.00	0.00	2
2019-06-10	0.00	0.00	0.00	1
2019-06-11	0.00	0.00	0.00	2
2019-06-14	0.00	0.00	0.00	1
2019-06-16	0.00	0.00	0.00	1
2019-06-17	0.00	0.00	0.00	1
2019-06-18	0.00	0.00	0.00	3
2019-06-19	0.00	0.00	0.00	1
2019-06-20	0.00	0.00	0.00	2
2019-06-21	0.00	0.00	0.00	2
2019-06-24	0.00	0.00	0.00	2
2019-06-25	0.00	0.00	0.00	1
2019-06-26	0.00	0.00	0.00	1
2019-06-29	0.00	0.00	0.00	1
2019-07-02	0.00	0.00	0.00	2
2019-07-06	0.00	0.00	0.00	1

2019-07-08	0.00	0.00	0.00	2
2019-07-11	0.00	0.00	0.00	2
2019-07-12	0.00	0.00	0.00	1
2019-07-13	0.00	0.00	0.00	2
2019-07-14	0.00	0.00	0.00	3
2019-07-15	0.00	0.00	0.00	2
2019-07-16	0.00	0.00	0.00	2
2019-07-18	0.00	0.00	0.00	2
2019-07-19	0.00	0.00	0.00	3
2019-07-20	0.00	0.00	0.00	4
2019-07-21	0.00	0.00	0.00	2
2019-07-22	0.00	0.00	0.00	0
2019-07-23	0.00	0.00	0.00	1
2019-07-24	0.00	0.00	0.00	1
2019-07-25	0.00	0.00	0.00	2
2019-07-26	0.00	0.00	0.00	1
2019-07-27	0.00	0.00	0.00	2
2019-07-29	0.00	0.00	0.00	1
2019-07-30	0.00	0.00	0.00	3
2019-07-31	0.00	0.00	0.00	1
2019-08-01	0.00	0.00	0.00	2
2019-08-02	0.00	0.00	0.00	1
2019-08-03	0.00	0.00	0.00	1
2019-08-06	0.00	0.00	0.00	1
2019-08-07	0.00	0.00	0.00	1
2019-08-08	0.00	0.00	0.00	1
2019-08-09	0.00	0.00	0.00	2
2019-08-10	0.00	0.00	0.00	1
2019-08-11	0.00	0.00	0.00	1
2019-08-13	0.00	0.00	0.00	2
2019-08-15	0.00	0.00	0.00	1
2019-08-17	0.00	0.00	0.00	2
2019-08-19	0.00	0.00	0.00	1
2019-08-20	0.00	0.00	0.00	1
2019-08-23	0.00	0.00	0.00	1
2019-08-24	0.00	0.00	0.00	1
2019-08-25	0.00	0.00	0.00	1
2019-08-26	0.00	0.00	0.00	3
2019-08-27	0.00	0.00	0.00	2
2019-08-28	0.00	0.00	0.00	1
2019-08-29	0.00	0.00	0.00	1
2019-08-30	0.00	0.00	0.00	2
2019-08-31	0.00	0.00	0.00	2
2019-09-01	0.00	0.00	0.00	1
2019-09-03	0.00	0.00	0.00	2
2019-09-06	0.00	0.00	0.00	1
2019-09-07	0.00	0.00	0.00	1
2019-09-08	0.00	0.00	0.00	1

2019-09-12	0.00	0.00	0.00	3
2019-09-13	0.00	0.00	0.00	1
2019-09-14	0.00	0.00	0.00	4
2019-09-16	0.00	0.00	0.00	1
2019-09-19	0.00	0.00	0.00	1
2019-09-23	0.00	0.00	0.00	1
2019-09-24	0.00	0.00	0.00	1
2019-09-25	0.00	0.00	0.00	2
2019-09-26	0.00	0.00	0.00	1
2019-09-28	0.00	0.00	0.00	1
2019-09-30	0.00	0.00	0.00	3
2019-10-03	0.00	0.00	0.00	3
2019-10-05	0.00	0.00	0.00	4
2019-10-07	0.00	0.00	0.00	1
2019-10-09	0.00	0.00	0.00	1
2019-10-10	0.00	0.00	0.00	2
2019-10-11	0.00	0.00	0.00	1
2019-10-12	0.00	0.00	0.00	2
2019-10-13	0.00	0.00	0.00	2
2019-10-14	0.00	0.00	0.00	2
2019-10-17	0.00	0.00	0.00	1
2019-10-18	0.00	0.00	0.00	1
2019-10-19	0.00	0.00	0.00	1
2019-10-20	0.00	0.00	0.00	3
2019-10-21	0.00	0.00	0.00	1
2019-10-22	0.00	0.00	0.00	1
2019-10-24	0.00	0.00	0.00	1
2019-10-25	0.00	0.00	0.00	1
2019-10-28	0.00	0.00	0.00	4
2019-10-29	0.00	0.00	0.00	1
2019-10-30	0.00	0.00	0.00	1
2019-10-31	0.00	0.00	0.00	1
2019-11-01	0.00	0.00	0.00	3
2019-11-02	0.00	0.00	0.00	1
2019-11-04	0.00	0.00	0.00	2
2019-11-05	0.00	0.00	0.00	1
2019-11-07	0.00	0.00	0.00	1
2019-11-10	0.00	0.00	0.00	1
2019-11-12	0.00	0.00	0.00	1
2019-11-14	0.00	0.00	0.00	1
2019-11-17	0.00	0.00	0.00	2
2019-11-18	0.00	0.00	0.00	1
2019-11-20	0.00	0.00	0.00	2
2019-11-23	0.00	0.00	0.00	2
2019-11-24	0.00	0.00	0.00	3
2019-11-25	0.00	0.00	0.00	2
2019-11-26	0.00	0.00	0.00	2
2019-12-01	0.00	0.00	0.00	1

2019-12-05	0.00	0.00	0.00	4
2019-12-06	0.00	0.00	0.00	3
2019-12-08	0.00	0.00	0.00	2
2019-12-09	0.00	0.00	0.00	3
2019-12-10	0.00	0.00	0.00	1
2019-12-11	0.00	0.00	0.00	1
2019-12-12	0.00	0.00	0.00	2
2019-12-14	0.00	0.00	0.00	1
2019-12-15	0.00	0.00	0.00	2
2019-12-16	0.00	0.00	0.00	2
2019-12-17	0.00	0.00	0.00	2
2019-12-18	0.00	0.00	0.00	2
2019-12-19	0.00	0.00	0.00	2
2019-12-20	0.00	0.00	0.00	1
2019-12-23	0.00	0.00	0.00	1
2019-12-24	0.00	0.00	0.00	3
2019-12-25	0.00	0.00	0.00	1
2019-12-26	0.00	0.00	0.00	3
2019-12-27	0.00	0.00	0.00	1
2019-12-28	0.00	0.00	0.00	1
2019-12-30	0.00	0.00	0.00	1
2019-12-31	0.00	0.00	0.00	1
2020-01-01	0.00	0.00	0.00	1
2020-01-02	0.00	0.00	0.00	1
2020-01-05	0.00	0.00	0.00	1
2020-01-06	0.00	0.00	0.00	2
2020-01-09	0.00	0.00	0.00	1
2020-01-11	0.00	0.00	0.00	3
2020-01-13	0.00	0.00	0.00	1
2020-01-14	0.00	0.00	0.00	2
2020-01-15	0.00	0.00	0.00	2
2020-01-16	0.00	0.00	0.00	1
2020-01-17	0.00	0.00	0.00	2
2020-01-19	0.00	0.00	0.00	2
2020-01-20	0.00	0.00	0.00	1
2020-01-21	0.00	0.00	0.00	1
2020-01-23	0.00	0.00	0.00	1
2020-01-24	0.00	0.00	0.00	2
2020-01-26	0.00	0.00	0.00	1
2020-01-27	0.00	0.00	0.00	3
2020-01-29	0.00	0.00	0.00	1
2020-01-30	0.00	0.00	0.00	1
2020-02-02	0.00	0.00	0.00	1
2020-02-03	0.00	0.00	0.00	2
2020-02-04	0.00	0.00	0.00	1
2020-02-05	0.00	0.00	0.00	2
2020-02-06	0.00	0.00	0.00	2
2020-02-10	0.00	0.00	0.00	1

2020-02-11	0.00	0.00	0.00	1
2020-02-13	0.00	0.00	0.00	1
2020-02-14	0.00	0.00	0.00	2
2020-02-15	0.00	0.00	0.00	3
2020-02-17	0.00	0.00	0.00	1
2020-02-19	0.00	0.00	0.00	1
2020-02-21	0.00	0.00	0.00	1
2020-02-23	0.00	0.00	0.00	3
2020-02-24	0.00	0.00	0.00	1
2020-02-26	0.00	0.00	0.00	2
2020-02-27	0.00	0.00	0.00	2
2020-02-28	0.00	0.00	0.00	1
2020-03-01	0.00	0.00	0.00	1
2020-03-02	0.00	0.00	0.00	1
2020-03-03	0.00	0.00	0.00	1
2020-03-06	0.00	0.00	0.00	2
2020-03-07	0.00	0.00	0.00	2
2020-03-08	0.00	0.00	0.00	3
2020-03-09	0.00	0.00	0.00	1
2020-03-10	0.00	0.00	0.00	2
2020-03-11	0.00	0.00	0.00	1
2020-03-15	0.00	0.00	0.00	1
2020-03-16	0.00	0.00	0.00	1
2020-03-18	0.00	0.00	0.00	2
2020-03-21	0.00	0.00	0.00	2
2020-03-23	0.00	0.00	0.00	1
2020-03-24	0.00	0.00	0.00	1
2020-03-25	0.00	0.00	0.00	1
2020-03-28	0.00	0.00	0.00	4
2020-03-29	0.00	0.00	0.00	4
2020-03-30	0.00	0.00	0.00	1
2020-03-31	0.00	0.00	0.00	1
2020-04-02	0.00	0.00	0.00	2
2020-04-05	0.00	0.00	0.00	2
2020-04-06	0.00	0.00	0.00	1
2020-04-07	0.00	0.00	0.00	2
2020-04-08	0.00	0.00	0.00	1
2020-04-09	0.00	0.00	0.00	1
2020-04-10	0.00	0.00	0.00	1
2020-04-13	0.00	0.00	0.00	1
2020-04-14	0.00	0.00	0.00	1
2020-04-16	0.00	0.00	0.00	2
2020-04-17	0.00	0.00	0.00	4
2020-04-19	0.00	0.00	0.00	2
2020-04-20	0.00	0.00	0.00	4
2020-04-21	0.00	0.00	0.00	2
2020-04-22	0.00	0.00	0.00	3
2020-04-24	0.00	0.00	0.00	1

2020-04-25	0.00	0.00	0.00	1
2020-04-26	0.00	0.00	0.00	2
2020-04-27	0.00	0.00	0.00	1
2020-04-29	0.00	0.00	0.00	2
2020-04-30	0.00	0.00	0.00	1
2020-05-01	0.00	0.00	0.00	1
2020-05-02	0.00	0.00	0.00	1
2020-05-03	0.00	0.00	0.00	1
2020-05-04	0.00	0.00	0.00	2
2020-05-05	0.00	0.00	0.00	2
2020-05-06	0.00	0.00	0.00	1
2020-05-07	0.00	0.00	0.00	1
2020-05-08	0.00	0.00	0.00	3
2020-05-10	0.00	0.00	0.00	1
2020-05-11	0.00	0.00	0.00	2
2020-05-13	0.00	0.00	0.00	3
2020-05-14	0.00	0.00	0.00	2
2020-05-15	0.00	0.00	0.00	3
2020-05-18	0.00	0.00	0.00	1
2020-05-19	0.00	0.00	0.00	1
2020-05-20	0.00	0.00	0.00	2
2020-05-21	0.00	0.00	0.00	2
2020-05-22	0.00	0.00	0.00	1
2020-05-23	0.00	0.00	0.00	1
2020-05-24	0.00	0.00	0.00	2
2020-05-25	0.00	0.00	0.00	1
2020-05-26	0.00	0.00	0.00	2
2020-05-28	0.00	0.00	0.00	1
2020-05-29	0.00	0.00	0.00	3
2020-05-31	0.00	0.00	0.00	4
2020-06-01	0.00	0.00	0.00	2
2020-06-02	0.00	0.00	0.00	1
2020-06-03	0.00	0.00	0.00	2
2020-06-04	0.00	0.00	0.00	1
2020-06-06	0.00	0.00	0.00	1
2020-06-07	0.00	0.00	0.00	1
2020-06-08	0.00	0.00	0.00	2
2020-06-09	0.00	0.00	0.00	1
2020-06-10	0.00	0.00	0.00	2
2020-06-11	0.00	0.00	0.00	2
2020-06-12	0.00	0.00	0.00	2
2020-06-13	0.00	0.00	0.00	1
2020-06-14	0.00	0.00	0.00	2
2020-06-15	0.00	0.00	0.00	3
2020-06-19	0.00	0.00	0.00	1
2020-06-21	0.00	0.00	0.00	1
2020-06-22	0.00	0.00	0.00	2
2020-06-24	0.00	0.00	0.00	1

2020-06-25	0.00	0.00	0.00	1
2020-06-26	0.00	0.00	0.00	1
2020-06-27	0.00	0.00	0.00	2
2020-06-28	0.00	0.00	0.00	2
2020-07-03	0.00	0.00	0.00	2
2020-07-04	0.00	0.00	0.00	3
2020-07-07	0.00	0.00	0.00	2
2020-07-09	0.00	0.00	0.00	3
2020-07-10	0.00	0.00	0.00	2
2020-07-11	0.00	0.00	0.00	2
2020-07-13	0.00	0.00	0.00	1
2020-07-14	0.00	0.00	0.00	1
2020-07-15	0.00	0.00	0.00	1
2020-07-16	0.00	0.00	0.00	1
2020-07-17	0.00	0.00	0.00	3
2020-07-22	0.00	0.00	0.00	3
2020-07-23	0.00	0.00	0.00	1
2020-07-25	0.00	0.00	0.00	1
2020-07-27	0.00	0.00	0.00	1
2020-07-28	0.00	0.00	0.00	1
2020-07-29	0.00	0.00	0.00	1
2020-07-31	0.00	0.00	0.00	3
2020-08-01	0.00	0.00	0.00	1
2020-08-02	0.00	0.00	0.00	1
2020-08-03	0.00	0.00	0.00	2
2020-08-04	0.00	0.00	0.00	3
2020-08-06	0.00	0.00	0.00	1
2020-08-07	0.00	0.00	0.00	4
2020-08-08	0.00	0.00	0.00	1
2020-08-09	0.00	0.00	0.00	2
2020-08-10	0.00	0.00	0.00	1
2020-08-11	0.00	0.00	0.00	1
2020-08-12	0.00	0.00	0.00	2
2020-08-14	0.00	0.00	0.00	2
2020-08-15	0.00	0.00	0.00	2
2020-08-19	0.00	0.00	0.00	1
2020-08-20	0.00	0.00	0.00	1
2020-08-24	0.00	0.00	0.00	2
2020-08-26	0.00	0.00	0.00	1
2020-08-28	0.00	0.00	0.00	1
2020-08-29	0.00	0.00	0.00	1
2020-08-30	0.00	0.00	0.00	1
2020-08-31	0.00	0.00	0.00	2
2020-09-02	0.00	0.00	0.00	1
2020-09-05	0.00	0.00	0.00	1
2020-09-06	0.00	0.00	0.00	1
2020-09-07	0.00	0.00	0.00	1
2020-09-09	0.00	0.00	0.00	1

2020-09-10	0.00	0.00	0.00	2
2020-09-12	0.00	0.00	0.00	1
2020-09-13	0.00	0.00	0.00	1
2020-09-15	0.00	0.00	0.00	1
2020-09-16	0.00	0.00	0.00	1
2020-09-19	0.00	0.00	0.00	1
2020-09-21	0.00	0.00	0.00	2
2020-09-25	0.00	0.00	0.00	2
2020-09-26	0.00	0.00	0.00	2
2020-09-27	0.00	0.00	0.00	1
2020-09-30	0.00	0.00	0.00	2
2020-10-01	0.00	0.00	0.00	1
2020-10-03	0.00	0.00	0.00	2
2020-10-04	0.00	0.00	0.00	1
2020-10-05	0.00	0.00	0.00	2
2020-10-07	0.00	0.00	0.00	1
2020-10-08	0.00	0.00	0.00	3
2020-10-10	0.00	0.00	0.00	1
2020-10-12	0.00	0.00	0.00	1
2020-10-15	0.00	0.00	0.00	1
2020-10-16	0.00	0.00	0.00	1
2020-10-17	0.00	0.00	0.00	1
2020-10-18	0.00	0.00	0.00	3
2020-10-20	0.00	0.00	0.00	1
2020-10-23	0.00	0.00	0.00	1
2020-10-25	0.00	0.00	0.00	1
2020-10-27	0.00	0.00	0.00	2
2020-10-28	0.00	0.00	0.00	1
2020-10-29	0.00	0.00	0.00	1
2020-10-31	0.00	0.00	0.00	2
2020-11-01	0.00	0.00	0.00	3
2020-11-02	0.00	0.00	0.00	1
2020-11-03	0.00	0.00	0.00	1
2020-11-04	0.00	0.00	0.00	4
2020-11-05	0.00	0.00	0.00	1
2020-11-07	0.00	0.00	0.00	1
2020-11-08	0.00	0.00	0.00	2
2020-11-09	0.00	0.00	0.00	3
2020-11-10	0.00	0.00	0.00	2
2020-11-11	0.00	0.00	0.00	2
2020-11-12	0.00	0.00	0.00	1
2020-11-13	0.00	0.00	0.00	1
2020-11-15	0.00	0.00	0.00	1
2020-11-17	0.00	0.00	0.00	2
2020-11-18	0.00	0.00	0.00	2
2020-11-19	0.00	0.00	0.00	4
2020-11-21	0.00	0.00	0.00	1
2020-11-22	0.00	0.00	0.00	1

2020-11-23	0.00	0.00	0.00	3
2020-11-25	0.00	0.00	0.00	1
2020-11-26	0.00	0.00	0.00	1
2020-11-27	0.00	0.00	0.00	3
2020-11-28	0.00	0.00	0.00	1
2020-11-29	0.00	0.00	0.00	3
2020-11-30	0.00	0.00	0.00	4
2020-12-02	0.00	0.00	0.00	3
2020-12-03	0.00	0.00	0.00	1
2020-12-05	0.00	0.00	0.00	1
2020-12-06	0.00	0.00	0.00	1
2020-12-07	0.00	0.00	0.00	2
2020-12-08	0.00	0.00	0.00	1
2020-12-10	0.00	0.00	0.00	1
2020-12-12	0.00	0.00	0.00	3
2020-12-14	0.00	0.00	0.00	3
2020-12-15	0.00	0.00	0.00	3
2020-12-16	0.00	0.00	0.00	4
2020-12-17	0.00	0.00	0.00	1
2020-12-19	0.00	0.00	0.00	1
2020-12-22	0.00	0.00	0.00	1
2020-12-24	0.00	0.00	0.00	2
2020-12-26	0.00	0.00	0.00	1
2020-12-29	0.00	0.00	0.00	2
2020-12-31	0.00	0.00	0.00	1
2021-01-01	0.00	0.00	0.00	1
2021-01-02	0.00	0.00	0.00	1
2021-01-03	0.00	0.00	0.00	2
2021-01-04	0.00	0.00	0.00	2
2021-01-05	0.00	0.00	0.00	2
2021-01-06	0.00	0.00	0.00	2
2021-01-07	0.00	0.00	0.00	2
2021-01-08	0.00	0.00	0.00	2
2021-01-09	0.00	0.00	0.00	3
2021-01-10	0.00	0.00	0.00	1
2021-01-11	0.00	0.00	0.00	2
2021-01-12	0.00	0.00	0.00	1
2021-01-13	0.00	0.00	0.00	1
2021-01-14	0.00	0.00	0.00	1
2021-01-16	0.00	0.00	0.00	2
2021-01-18	0.00	0.00	0.00	1
2021-01-19	0.00	0.00	0.00	3
2021-01-21	0.00	0.00	0.00	3
2021-01-23	0.00	0.00	0.00	3
2021-01-24	0.00	0.00	0.00	1
2021-01-26	0.00	0.00	0.00	2
2021-01-27	0.00	0.00	0.00	2
2021-01-28	0.00	0.00	0.00	3

2021-01-29	0.00	0.00	0.00	1
2021-01-30	0.00	0.00	0.00	1
2021-01-31	0.00	0.00	0.00	1
2021-02-01	0.00	0.00	0.00	1
2021-02-02	0.00	0.00	0.00	2
2021-02-03	0.00	0.00	0.00	2
2021-02-04	0.00	0.00	0.00	1
2021-02-06	0.00	0.00	0.00	1
2021-02-10	0.00	0.00	0.00	1
2021-02-12	0.00	0.00	0.00	2
2021-02-13	0.00	0.00	0.00	2
2021-02-14	0.00	0.00	0.00	1
2021-02-18	0.00	0.00	0.00	1
2021-02-19	0.00	0.00	0.00	3
2021-02-20	0.00	0.00	0.00	1
2021-02-21	0.00	0.00	0.00	2
2021-02-22	0.00	0.00	0.00	1
2021-02-23	0.00	0.00	0.00	1
2021-02-24	0.00	0.00	0.00	1
2021-02-25	0.00	0.00	0.00	1
2021-02-27	0.00	0.00	0.00	2
2021-02-28	0.00	0.00	0.00	1
2021-03-01	0.00	0.00	0.00	1
2021-03-02	0.00	0.00	0.00	1
2021-03-03	0.00	0.00	0.00	3
2021-03-04	0.00	0.00	0.00	1
2021-03-05	0.00	0.00	0.00	1
2021-03-06	0.00	0.00	0.00	3
2021-03-07	0.00	0.00	0.00	1
2021-03-08	0.00	0.00	0.00	1
2021-03-09	0.00	0.00	0.00	1
2021-03-10	0.00	0.00	0.00	1
2021-03-12	0.00	0.00	0.00	1
2021-03-14	0.00	0.00	0.00	2
2021-03-17	0.00	0.00	0.00	1
2021-03-18	0.00	0.00	0.00	4
2021-03-19	0.00	0.00	0.00	1
2021-03-20	0.00	0.00	0.00	2
2021-03-22	0.00	0.00	0.00	2
2021-03-23	0.00	0.00	0.00	2
2021-03-24	0.00	0.00	0.00	2
2021-03-25	0.00	0.00	0.00	2
2021-03-26	0.00	0.00	0.00	1
2021-03-27	0.00	0.00	0.00	2
2021-03-28	0.00	0.00	0.00	4
2021-03-31	0.00	0.00	0.00	1
2021-04-02	0.00	0.00	0.00	3
2021-04-05	0.00	0.00	0.00	3

2021-04-06	0.00	0.00	0.00	2
2021-04-07	0.00	0.00	0.00	1
2021-04-08	0.00	0.00	0.00	1
2021-04-10	0.00	0.00	0.00	2
2021-04-11	0.00	0.00	0.00	1
2021-04-12	0.00	0.00	0.00	1
2021-04-14	0.00	0.00	0.00	1
2021-04-18	0.00	0.00	0.00	1
2021-04-19	0.00	0.00	0.00	1
2021-04-22	0.00	0.00	0.00	2
2021-04-24	0.00	0.00	0.00	1
2021-04-25	0.00	0.00	0.00	3
2021-04-26	0.00	0.00	0.00	1
2021-04-27	0.00	0.00	0.00	2
2021-04-28	0.00	0.00	0.00	1
2021-05-01	0.00	0.00	0.00	2
2021-05-02	0.00	0.00	0.00	2
2021-05-03	0.00	0.00	0.00	2
2021-05-04	0.00	0.00	0.00	1
2021-05-06	0.00	0.00	0.00	1
2021-05-08	0.00	0.00	0.00	2
2021-05-09	0.00	0.00	0.00	1
2021-05-10	0.00	0.00	0.00	2
2021-05-11	0.00	0.00	0.00	1
2021-05-13	0.00	0.00	0.00	2
2021-05-14	0.00	0.00	0.00	1
2021-05-16	0.00	0.00	0.00	1
2021-05-20	0.00	0.00	0.00	2
2021-05-21	0.00	0.00	0.00	2
2021-05-23	0.00	0.00	0.00	2
2021-05-29	0.00	0.00	0.00	1
2021-05-30	0.00	0.00	0.00	1
2021-05-31	0.00	0.00	0.00	1
2021-06-02	0.00	0.00	0.00	1
2021-06-03	0.00	0.00	0.00	1
2021-06-05	0.00	0.00	0.00	1
2021-06-06	0.00	0.00	0.00	1
2021-06-07	0.00	0.00	0.00	2
2021-06-09	0.00	0.00	0.00	3
2021-06-10	0.00	0.00	0.00	1
2021-06-11	0.00	0.00	0.00	1
2021-06-12	0.00	0.00	0.00	2
2021-06-13	0.00	0.00	0.00	1
2021-06-14	0.00	0.00	0.00	1
2021-06-15	0.00	0.00	0.00	3
2021-06-16	0.00	0.00	0.00	2
2021-06-18	0.00	0.00	0.00	2
2021-06-19	0.00	0.00	0.00	2

2021-06-20	0.00	0.00	0.00	1
2021-06-21	0.00	0.00	0.00	4
2021-06-22	0.00	0.00	0.00	1
2021-06-25	0.00	0.00	0.00	1
2021-06-26	0.00	0.00	0.00	1
2021-06-27	0.00	0.00	0.00	1
2021-06-28	0.00	0.00	0.00	1
2021-06-29	0.00	0.00	0.00	3
2021-06-30	0.00	0.00	0.00	3
2021-07-02	0.00	0.00	0.00	2
2021-07-04	0.00	0.00	0.00	2
2021-07-05	0.00	0.00	0.00	2
2021-07-06	0.00	0.00	0.00	2
2021-07-11	0.00	0.00	0.00	1
2021-07-14	0.00	0.00	0.00	4
2021-07-15	0.00	0.00	0.00	1
2021-07-16	0.00	0.00	0.00	1
2021-07-18	0.00	0.00	0.00	3
2021-07-19	0.00	0.00	0.00	2
2021-07-21	0.00	0.00	0.00	1
2021-07-22	0.00	0.00	0.00	1
2021-07-23	0.00	0.00	0.00	1
2021-07-24	0.00	0.00	0.00	1
2021-07-27	0.00	0.00	0.00	3
2021-07-31	0.00	0.00	0.00	1
2021-08-01	0.00	0.00	0.00	1
2021-08-02	0.00	0.00	0.00	1
2021-08-04	0.00	0.00	0.00	2
2021-08-05	0.00	0.00	0.00	1
2021-08-06	0.00	0.00	0.00	4
2021-08-07	0.00	0.00	0.00	1
2021-08-10	0.00	0.00	0.00	1
2021-08-12	0.00	0.00	0.00	1
2021-08-13	0.00	0.00	0.00	1
2021-08-16	0.00	0.00	0.00	1
2021-08-18	0.00	0.00	0.00	4
2021-08-19	0.00	0.00	0.00	2
2021-08-20	0.00	0.00	0.00	1
2021-08-21	0.00	0.00	0.00	1
2021-08-22	0.00	0.00	0.00	2
2021-08-23	0.00	0.00	0.00	1
2021-08-24	0.00	0.00	0.00	3
2021-08-25	0.00	0.00	0.00	1
2021-08-26	0.00	0.00	0.00	1
2021-08-27	0.00	0.00	0.00	1
2021-08-28	0.00	0.00	0.00	1
2021-08-29	0.00	0.00	0.00	1
2021-08-30	0.00	0.00	0.00	2

2021-09-01	0.00	0.00	0.00	1
2021-09-02	0.00	0.00	0.00	1
2021-09-03	0.00	0.00	0.00	2
2021-09-04	0.00	0.00	0.00	2
2021-09-06	0.00	0.00	0.00	4
2021-09-07	0.00	0.00	0.00	1
2021-09-09	0.00	0.00	0.00	1
2021-09-10	0.00	0.00	0.00	1
2021-09-12	0.00	0.00	0.00	1
2021-09-13	0.00	0.00	0.00	1
2021-09-16	0.00	0.00	0.00	2
2021-09-19	0.00	0.00	0.00	2
2021-09-20	0.00	0.00	0.00	1
2021-09-21	0.00	0.00	0.00	1
2021-09-24	0.00	0.00	0.00	1
2021-09-25	0.00	0.00	0.00	2
2021-09-26	0.00	0.00	0.00	1
2021-09-27	0.00	0.00	0.00	3
2021-09-28	0.00	0.00	0.00	1
2021-09-29	0.00	0.00	0.00	2
2021-09-30	0.00	0.00	0.00	2
2021-10-04	0.00	0.00	0.00	1
2021-10-05	0.00	0.00	0.00	1
2021-10-06	0.00	0.00	0.00	1
2021-10-07	0.00	0.00	0.00	1
2021-10-08	0.00	0.00	0.00	1
2021-10-11	0.00	0.00	0.00	4
2021-10-12	0.00	0.00	0.00	2
2021-10-13	0.00	0.00	0.00	2
2021-10-14	0.00	0.00	0.00	2
2021-10-17	0.00	0.00	0.00	1
2021-10-18	0.00	0.00	0.00	2
2021-10-19	0.00	0.00	0.00	2
2021-10-20	0.00	0.00	0.00	1
2021-10-21	0.00	0.00	0.00	2
2021-10-22	0.00	0.00	0.00	3
2021-10-23	0.00	0.00	0.00	1
2021-10-24	0.00	0.00	0.00	1
2021-10-25	0.00	0.00	0.00	3
2021-10-27	0.00	0.00	0.00	2
2021-10-28	0.00	0.00	0.00	1
2021-10-31	0.00	0.00	0.00	1
2021-11-03	0.00	0.00	0.00	1
2021-11-04	0.00	0.00	0.00	2
2021-11-06	0.00	0.00	0.00	1
2021-11-07	0.00	0.00	0.00	2
2021-11-08	0.00	0.00	0.00	2
2021-11-10	0.00	0.00	0.00	1

2021-11-12	0.00	0.00	0.00	1
2021-11-13	0.00	0.00	0.00	1
2021-11-14	0.00	0.00	0.00	1
2021-11-15	0.00	0.00	0.00	2
2021-11-21	0.00	0.00	0.00	1
2021-11-22	0.00	0.00	0.00	1
2021-11-25	0.00	0.00	0.00	2
2021-11-27	0.00	0.00	0.00	1
2021-11-28	0.00	0.00	0.00	2
2021-11-30	0.00	0.00	0.00	2
2021-12-01	0.00	0.00	0.00	2
2021-12-04	0.00	0.00	0.00	1
2021-12-05	0.00	0.00	0.00	2
2021-12-06	0.00	0.00	0.00	1
2021-12-07	0.00	0.00	0.00	2
2021-12-11	0.00	0.00	0.00	3
2021-12-14	0.00	0.00	0.00	1
2021-12-15	0.00	0.00	0.00	2
2021-12-16	0.00	0.00	0.00	1
2021-12-18	0.00	0.00	0.00	1
2021-12-19	0.00	0.00	0.00	1
2021-12-20	0.00	0.00	0.00	4
2021-12-21	0.00	0.00	0.00	2
2021-12-23	0.00	0.00	0.00	3
2021-12-24	0.00	0.00	0.00	2
2021-12-25	0.00	0.00	0.00	1
2021-12-26	0.00	0.00	0.00	1
2021-12-28	0.00	0.00	0.00	3
2021-12-30	0.00	1.00	0.01	2
2022-01-04	0.00	0.00	0.00	4
2022-01-06	0.00	0.00	0.00	1
2022-01-07	0.00	0.00	0.00	2
2022-01-08	0.00	0.00	0.00	3
2022-01-09	0.00	0.00	0.00	3
2022-01-10	0.00	0.00	0.00	1
2022-01-13	0.00	0.00	0.00	1
2022-01-15	0.00	0.00	0.00	2
2022-01-18	0.00	0.00	0.00	2
2022-01-19	0.00	0.00	0.00	1
2022-01-20	0.00	0.00	0.00	3
2022-01-22	0.00	0.00	0.00	1
2022-01-23	0.00	0.00	0.00	2
2022-01-31	0.00	0.00	0.00	2
2022-02-02	0.00	0.00	0.00	2
2022-02-05	0.00	0.00	0.00	3
2022-02-07	0.00	0.00	0.00	2
2022-02-09	0.00	0.00	0.00	4
2022-02-10	0.00	0.00	0.00	4

2022-02-11	0.00	0.00	0.00	3
2022-02-13	0.00	0.00	0.00	2
2022-02-14	0.00	0.00	0.00	2
2022-02-16	0.00	0.00	0.00	1
2022-02-17	0.00	0.00	0.00	1
2022-02-18	0.00	0.00	0.00	4
2022-02-20	0.00	0.00	0.00	1
2022-02-21	0.00	0.00	0.00	1
2022-02-24	0.00	0.00	0.00	1
2022-02-25	0.00	0.00	0.00	3
2022-02-26	0.00	0.00	0.00	1
2022-03-02	0.00	0.00	0.00	2
2022-03-03	0.00	0.00	0.00	4
2022-03-05	0.00	0.00	0.00	2
2022-03-06	0.00	0.00	0.00	2
2022-03-07	0.00	0.00	0.00	1
2022-03-09	0.00	0.00	0.00	1
2022-03-10	0.00	0.00	0.00	2
2022-03-11	0.00	0.00	0.00	1
2022-03-12	0.00	0.00	0.00	2
2022-03-13	0.00	0.00	0.00	3
2022-03-14	0.00	0.00	0.00	2
2022-03-16	0.00	0.00	0.00	2
2022-03-18	0.00	0.00	0.00	1
2022-03-19	0.00	0.00	0.00	2
2022-03-20	0.00	0.00	0.00	3
2022-03-24	0.00	0.00	0.00	1
2022-03-25	0.00	0.00	0.00	1
2022-03-26	0.00	0.00	0.00	2
2022-03-27	0.00	0.00	0.00	3
2022-03-28	0.00	0.00	0.00	1
2022-03-29	0.00	0.00	0.00	2
2022-04-01	0.00	0.00	0.00	2
2022-04-02	0.00	0.00	0.00	1
2022-04-05	0.00	0.00	0.00	1
2022-04-07	0.00	0.00	0.00	1
2022-04-09	0.00	0.00	0.00	2
2022-04-13	0.00	0.00	0.00	1
2022-04-14	0.00	0.00	0.00	2
2022-04-15	0.00	0.00	0.00	1
2022-04-16	0.00	0.00	0.00	2
2022-04-17	0.00	0.00	0.00	2
2022-04-18	0.00	0.00	0.00	1
2022-04-19	0.00	0.00	0.00	2
2022-04-20	0.00	0.00	0.00	2
2022-04-23	0.00	0.00	0.00	2
2022-04-24	0.00	0.00	0.00	1
2022-04-26	0.00	0.00	0.00	3

2022-04-27	0.00	0.00	0.00	2
2022-04-29	0.00	0.00	0.00	1
2022-05-02	0.00	0.00	0.00	1
2022-05-03	0.00	0.00	0.00	1
2022-05-05	0.00	0.00	0.00	2
2022-05-06	0.00	0.00	0.00	2
2022-05-08	0.00	0.00	0.00	3
2022-05-09	0.00	0.00	0.00	1
2022-05-10	0.00	0.00	0.00	2
2022-05-11	0.00	0.00	0.00	1
2022-05-14	0.00	0.00	0.00	2
2022-05-17	0.00	0.00	0.00	2
2022-05-20	0.00	0.00	0.00	1
2022-05-21	0.00	0.00	0.00	1
2022-05-23	0.00	0.00	0.00	2
2022-05-25	0.00	0.00	0.00	3
2022-05-26	0.00	0.00	0.00	1
2022-05-27	0.00	0.00	0.00	2
2022-05-29	0.00	0.00	0.00	1
2022-05-30	0.00	0.00	0.00	1
2022-05-31	0.00	0.00	0.00	1
2022-06-01	0.00	0.00	0.00	2
2022-06-02	0.00	0.00	0.00	1
2022-06-04	0.00	0.00	0.00	4
2022-06-05	0.00	0.00	0.00	3
2022-06-07	0.00	0.00	0.00	4
2022-06-08	0.00	0.00	0.00	2
2022-06-09	0.00	0.00	0.00	1
2022-06-11	0.00	0.00	0.00	1
2022-06-15	0.00	0.00	0.00	2
2022-06-19	0.00	0.00	0.00	1
2022-06-21	0.00	0.00	0.00	1
2022-06-22	0.00	0.00	0.00	2
2022-06-23	0.00	0.00	0.00	1
2022-06-24	0.00	0.00	0.00	1
2022-06-25	0.00	0.00	0.00	1
2022-06-26	0.00	0.00	0.00	1
2022-06-27	0.00	0.00	0.00	1
2022-06-28	0.00	0.00	0.00	3
2022-06-29	0.00	0.00	0.00	1
2022-06-30	0.00	0.00	0.00	2
2022-07-01	0.00	0.00	0.00	1
2022-07-02	0.00	0.00	0.00	2
2022-07-04	0.00	0.00	0.00	2
2022-07-06	0.00	0.00	0.00	2
2022-07-07	0.00	0.00	0.00	1
2022-07-08	0.00	0.00	0.00	1
2022-07-09	0.00	0.00	0.00	1

2022-07-11	0.00	0.00	0.00	1
2022-07-12	0.00	0.00	0.00	3
2022-07-13	0.00	0.00	0.00	1
2022-07-14	0.00	0.00	0.00	1
2022-07-16	0.00	0.00	0.00	2
2022-07-17	0.00	0.00	0.00	2
2022-07-19	0.00	0.00	0.00	1
2022-07-20	0.00	0.00	0.00	1
2022-07-21	0.00	0.00	0.00	1
2022-07-23	0.00	0.00	0.00	1
2022-07-24	0.00	0.00	0.00	1
2022-07-26	0.00	0.00	0.00	1
2022-07-27	0.00	0.00	0.00	1
2022-07-30	0.00	0.00	0.00	1
2022-07-31	0.00	0.00	0.00	1
2022-08-01	0.00	0.00	0.00	2
2022-08-02	0.00	0.00	0.00	3
2022-08-04	0.00	0.00	0.00	1
2022-08-07	0.00	0.00	0.00	1
2022-08-08	0.00	0.00	0.00	1
2022-08-09	0.00	0.00	0.00	2
2022-08-11	0.00	0.00	0.00	2
2022-08-12	0.00	0.00	0.00	1
2022-08-13	0.00	0.00	0.00	1
2022-08-14	0.00	0.00	0.00	1
2022-08-15	0.00	0.00	0.00	3
2022-08-16	0.00	0.00	0.00	2
2022-08-17	0.00	0.00	0.00	2
2022-08-18	0.00	0.00	0.00	1
2022-08-20	0.00	0.00	0.00	1
2022-08-25	0.00	0.00	0.00	1
2022-08-27	0.00	0.00	0.00	2
2022-08-29	0.00	0.00	0.00	1
2022-08-31	0.00	0.00	0.00	1
2022-09-01	0.00	0.00	0.00	2
2022-09-02	0.00	0.00	0.00	2
2022-09-03	0.00	0.00	0.00	1
2022-09-04	0.00	0.00	0.00	2
2022-09-05	0.00	0.00	0.00	1
2022-09-06	0.00	0.00	0.00	1
2022-09-07	0.00	0.00	0.00	2
2022-09-08	0.00	0.00	0.00	3
2022-09-09	0.00	0.00	0.00	1
2022-09-10	0.00	0.00	0.00	2
2022-09-12	0.00	0.00	0.00	2
2022-09-13	0.00	0.00	0.00	3
2022-09-17	0.00	0.00	0.00	3
2022-09-18	0.00	0.00	0.00	1

2022-09-20	0.00	0.00	0.00	2
2022-09-22	0.00	0.00	0.00	1
2022-09-23	0.00	0.00	0.00	1
2022-09-24	0.00	0.00	0.00	1
2022-09-25	0.00	0.00	0.00	1
2022-09-28	0.00	0.00	0.00	1
2022-09-29	0.00	0.00	0.00	2
2022-09-30	0.00	0.00	0.00	1
2022-10-04	0.00	0.00	0.00	1
2022-10-05	0.00	0.00	0.00	1
2022-10-06	0.00	0.00	0.00	1
2022-10-07	0.00	0.00	0.00	1
2022-10-08	0.00	0.00	0.00	4
2022-10-09	0.00	0.00	0.00	1
2022-10-10	0.00	0.00	0.00	1
2022-10-11	0.00	0.00	0.00	2
2022-10-12	0.00	0.00	0.00	1
2022-10-13	0.00	0.00	0.00	1
2022-10-14	0.00	0.00	0.00	1
2022-10-15	0.00	0.00	0.00	1
2022-10-17	0.00	0.00	0.00	1
2022-10-18	0.00	0.00	0.00	3
2022-10-20	0.00	0.00	0.00	2
2022-10-21	0.00	0.00	0.00	2
2022-10-22	0.00	0.00	0.00	2
2022-10-23	0.00	0.00	0.00	1
2022-10-24	0.00	0.00	0.00	1
2022-10-25	0.00	0.00	0.00	1
2022-10-26	0.00	0.00	0.00	2
2022-10-27	0.00	0.00	0.00	1
2022-10-28	0.00	0.00	0.00	1
2022-10-29	0.00	0.00	0.00	1
2022-10-30	0.00	0.00	0.00	2
2022-10-31	0.00	0.00	0.00	2
2022-11-03	0.00	0.00	0.00	2
2022-11-04	0.00	0.00	0.00	3
2022-11-05	0.00	0.00	0.00	3
2022-11-06	0.00	0.00	0.00	1
2022-11-07	0.00	0.00	0.00	1
2022-11-09	0.00	0.00	0.00	1
2022-11-10	0.00	0.00	0.00	1
2022-11-11	0.00	0.00	0.00	1
2022-11-12	0.00	0.00	0.00	2
2022-11-13	0.00	0.00	0.00	3
2022-11-14	0.00	0.00	0.00	1
2022-11-16	0.00	0.00	0.00	2
2022-11-17	0.00	0.00	0.00	1
2022-11-18	0.00	0.00	0.00	2

2022-11-20	0.00	0.00	0.00	1
2022-11-22	0.00	0.00	0.00	1
2022-11-24	0.00	0.00	0.00	3
2022-11-25	0.00	0.00	0.00	1
2022-11-26	0.00	0.00	0.00	2
2022-11-27	0.00	0.00	0.00	1
2022-11-28	0.00	0.00	0.00	3
2022-11-29	0.00	0.00	0.00	1
2022-11-30	0.00	0.00	0.00	1
2022-12-01	0.00	0.00	0.00	1
2022-12-04	0.00	0.00	0.00	2
2022-12-05	0.00	0.00	0.00	2
2022-12-07	0.00	0.00	0.00	2
2022-12-09	0.00	0.00	0.00	1
2022-12-10	0.00	0.00	0.00	2
2022-12-11	0.00	0.00	0.00	1
2022-12-12	0.00	0.00	0.00	1
2022-12-13	0.00	0.00	0.00	2
2022-12-14	0.00	0.00	0.00	2
2022-12-15	0.00	0.00	0.00	1
2022-12-16	0.00	0.00	0.00	1
2022-12-17	0.00	0.00	0.00	2
2022-12-18	0.00	0.00	0.00	1
2022-12-19	0.00	0.00	0.00	1
2022-12-20	0.00	0.00	0.00	3
2022-12-21	0.00	0.00	0.00	1
2022-12-22	0.00	0.00	0.00	1
2022-12-24	0.00	0.00	0.00	2
2022-12-25	0.00	0.00	0.00	1
2022-12-27	0.00	0.00	0.00	2
2022-12-29	0.00	0.00	0.00	2
2022-12-30	0.00	0.00	0.00	1
2022-12-31	0.00	0.00	0.00	2
2023-01-03	0.00	0.00	0.00	1
2023-01-04	0.00	0.00	0.00	1
2023-01-05	0.00	0.00	0.00	2
2023-01-06	0.00	0.00	0.00	1
2023-01-07	0.00	0.00	0.00	2
2023-01-09	0.00	0.00	0.00	1
2023-01-10	0.00	0.00	0.00	2
2023-01-11	0.00	0.00	0.00	2
2023-01-12	0.00	0.00	0.00	1
2023-01-13	0.00	0.00	0.00	1
2023-01-14	0.00	0.00	0.00	3
2023-01-16	0.00	0.00	0.00	1
2023-01-17	0.00	0.00	0.00	3
2023-01-18	0.00	0.00	0.00	2
2023-01-20	0.00	0.00	0.00	1

2023-01-22	0.00	0.00	0.00	1
2023-01-23	0.00	0.00	0.00	1
2023-01-24	0.00	0.00	0.00	1
2023-01-25	0.00	0.00	0.00	1
2023-01-26	0.00	0.00	0.00	5
2023-01-31	0.00	0.00	0.00	3
2023-02-01	0.00	0.00	0.00	1
2023-02-03	0.00	0.00	0.00	1
2023-02-04	0.00	0.00	0.00	2
2023-02-05	0.00	0.00	0.00	1
2023-02-06	0.00	0.00	0.00	1
2023-02-07	0.00	0.00	0.00	3
2023-02-08	0.00	0.00	0.00	2
2023-02-09	0.00	0.00	0.00	1
2023-02-10	0.00	0.00	0.00	3
2023-02-11	0.00	0.00	0.00	3
2023-02-12	0.00	0.00	0.00	1
2023-02-16	0.00	0.00	0.00	1
2023-02-17	0.00	0.00	0.00	2
2023-02-18	0.00	0.00	0.00	2
2023-02-20	0.00	0.00	0.00	1
2023-02-22	0.00	0.00	0.00	1
2023-02-24	0.00	0.00	0.00	1
2023-02-25	0.00	0.00	0.00	1
2023-02-26	0.00	0.00	0.00	3
2023-03-01	0.00	0.00	0.00	1
2023-03-03	0.00	0.00	0.00	2
2023-03-06	0.00	0.00	0.00	1
2023-03-07	0.00	0.00	0.00	1
2023-03-08	0.00	0.00	0.00	1
2023-03-10	0.00	0.00	0.00	2
2023-03-12	0.00	0.00	0.00	1
2023-03-13	0.00	0.00	0.00	2
2023-03-15	0.00	0.00	0.00	1
2023-03-16	0.00	0.00	0.00	1
2023-03-17	0.00	0.00	0.00	1
2023-03-18	0.00	0.00	0.00	3
2023-03-21	0.00	0.00	0.00	1
2023-03-22	0.00	0.00	0.00	2
2023-03-23	0.00	0.00	0.00	3
2023-03-25	0.00	0.00	0.00	1
2023-03-26	0.00	0.00	0.00	4
2023-03-27	0.00	0.00	0.00	2
2023-03-28	0.00	0.00	0.00	1
2023-03-30	0.00	0.00	0.00	3
2023-03-31	0.00	0.00	0.00	2
2023-04-02	0.00	0.00	0.00	3
2023-04-03	0.00	0.00	0.00	1

2023-04-04	0.00	0.00	0.00	1
2023-04-05	0.00	0.00	0.00	1
2023-04-06	0.00	0.00	0.00	2
2023-04-07	0.00	0.00	0.00	2
2023-04-09	0.00	0.00	0.00	1
2023-04-11	0.00	0.00	0.00	2
2023-04-13	0.00	0.00	0.00	2
2023-04-14	0.00	0.00	0.00	1
2023-04-15	0.00	0.00	0.00	3
2023-04-17	0.00	0.00	0.00	1
2023-04-19	0.00	0.00	0.00	2
2023-04-20	0.00	0.00	0.00	1
2023-04-22	0.00	0.00	0.00	2
2023-04-24	0.00	0.00	0.00	1
2023-04-25	0.00	0.00	0.00	2
2023-04-26	0.00	0.00	0.00	1
2023-04-27	0.00	0.00	0.00	2
2023-04-28	0.00	0.00	0.00	2
2023-04-29	0.00	0.00	0.00	2
2023-04-30	0.00	0.00	0.00	1
2023-05-06	0.00	0.00	0.00	2
2023-05-07	0.00	0.00	0.00	3
2023-05-09	0.00	0.00	0.00	3
2023-05-11	0.00	0.00	0.00	3
2023-05-12	0.00	0.00	0.00	2
2023-05-13	0.00	0.00	0.00	3
2023-05-14	0.00	0.00	0.00	1
2023-05-15	0.00	0.00	0.00	1
2023-05-17	0.00	0.00	0.00	1
2023-05-18	0.00	0.00	0.00	1
2023-05-19	0.00	0.00	0.00	2
2023-05-20	0.00	0.00	0.00	2
2023-05-21	0.00	0.00	0.00	2
2023-05-22	0.00	0.00	0.00	1
2023-05-24	0.00	0.00	0.00	2
2023-05-25	0.00	0.00	0.00	1
2023-05-26	0.00	0.00	0.00	1
2023-05-28	0.00	0.00	0.00	1
2023-06-03	0.00	0.00	0.00	2
2023-06-05	0.00	0.00	0.00	2
2023-06-06	0.00	0.00	0.00	3
2023-06-07	0.00	0.00	0.00	1
2023-06-08	0.00	0.00	0.00	1
2023-06-09	0.00	0.00	0.00	1
2023-06-11	0.00	0.00	0.00	2
2023-06-12	0.00	0.00	0.00	2
2023-06-13	0.00	0.00	0.00	2
2023-06-15	0.00	0.00	0.00	1

2023-06-16	0.00	0.00	0.00	1
2023-06-17	0.00	0.00	0.00	2
2023-06-18	0.00	0.00	0.00	2
2023-06-19	0.00	0.00	0.00	1
2023-06-21	0.00	0.00	0.00	2
2023-06-22	0.00	0.00	0.00	1
2023-06-23	0.00	0.00	0.00	1
2023-06-26	0.00	0.00	0.00	1
2023-06-27	0.00	0.00	0.00	3
2023-06-28	0.00	0.00	0.00	2
2023-06-29	0.00	0.00	0.00	1
2023-06-30	0.00	0.00	0.00	1
2023-07-02	0.00	0.00	0.00	1
2023-07-03	0.00	0.00	0.00	2
2023-07-04	0.00	0.00	0.00	1
2023-07-05	0.00	0.00	0.00	1
2023-07-06	0.00	0.00	0.00	4
2023-07-08	0.00	0.00	0.00	1
2023-07-10	0.00	0.00	0.00	1
2023-07-11	0.00	0.00	0.00	1
2023-07-13	0.00	0.00	0.00	1
2023-07-14	0.00	0.00	0.00	1
2023-07-15	0.00	0.00	0.00	2
2023-07-17	0.00	0.00	0.00	2
2023-07-18	0.00	0.00	0.00	1
2023-07-19	0.00	0.00	0.00	2
2023-07-20	0.00	0.00	0.00	3
2023-07-21	0.00	0.00	0.00	1
2023-07-22	0.00	0.00	0.00	4
2023-07-23	0.00	0.00	0.00	3
2023-07-24	0.00	0.00	0.00	1
2023-07-25	0.00	0.00	0.00	1
2023-07-26	0.00	0.00	0.00	1
2023-07-28	0.00	0.00	0.00	2
2023-07-29	0.00	0.00	0.00	3
2023-07-31	0.00	0.00	0.00	2
2023-08-01	0.00	0.00	0.00	1
2023-08-02	0.00	0.00	0.00	1
2023-08-03	0.00	0.00	0.00	2
2023-08-04	0.00	0.00	0.00	4
2023-08-05	0.00	0.00	0.00	2
2023-08-06	0.00	0.00	0.00	1
2023-08-08	0.00	0.00	0.00	2
2023-08-10	0.00	0.00	0.00	2
2023-08-11	0.00	0.00	0.00	1
2023-08-13	0.00	0.00	0.00	1
2023-08-14	0.00	0.00	0.00	1
2023-08-15	0.00	0.00	0.00	2

2023-08-16	0.00	0.00	0.00	1
2023-08-17	0.00	0.00	0.00	2
2023-08-19	0.00	0.00	0.00	1
2023-08-21	0.00	0.00	0.00	3
2023-08-23	0.00	0.00	0.00	1
2023-08-25	0.00	0.00	0.00	1
2023-08-26	0.00	0.00	0.00	2
2023-08-27	0.00	0.00	0.00	1
2023-08-28	0.00	0.00	0.00	2
2023-08-29	0.00	0.00	0.00	1
2023-08-30	0.00	0.00	0.00	5
2023-09-01	0.00	0.00	0.00	2
2023-09-02	0.00	0.00	0.00	1
2023-09-03	0.00	0.00	0.00	1
2023-09-06	0.00	0.00	0.00	1
2023-09-08	0.00	0.00	0.00	2
2023-09-10	0.00	0.00	0.00	1
2023-09-11	0.00	0.00	0.00	1
2023-09-13	0.00	0.00	0.00	1
2023-09-14	0.00	0.00	0.00	2
2023-09-15	0.00	0.00	0.00	2
2023-09-16	0.00	0.00	0.00	1
2023-09-17	0.00	0.00	0.00	1
2023-09-19	0.00	0.00	0.00	1
2023-09-20	0.00	0.00	0.00	2
2023-09-21	0.00	0.00	0.00	3
2023-09-26	0.00	0.00	0.00	0
2023-09-27	0.00	0.00	0.00	3
2023-09-28	0.00	0.00	0.00	1
2023-09-30	0.00	0.00	0.00	1
2023-10-02	0.00	0.00	0.00	2
2023-10-03	0.00	0.00	0.00	2
2023-10-05	0.00	0.00	0.00	1
2023-10-08	0.00	0.00	0.00	2
2023-10-10	0.00	0.00	0.00	1
2023-10-11	0.00	0.00	0.00	2
2023-10-14	0.00	0.00	0.00	2
2023-10-16	0.00	0.00	0.00	1
2023-10-17	0.00	0.00	0.00	2
2023-10-19	0.00	0.00	0.00	1
2023-10-20	0.00	0.00	0.00	2
2023-10-21	0.00	0.00	0.00	1
2023-10-22	0.00	0.00	0.00	2
2023-10-23	0.00	0.00	0.00	2
2023-10-24	0.00	0.00	0.00	1
2023-10-26	0.00	0.00	0.00	1
2023-10-27	0.00	0.00	0.00	1
2023-10-28	0.00	0.00	0.00	1

```
  2023-10-31       0.00      0.00      0.00         1
  2023-11-02       0.00      0.00      0.00         4
  2023-11-05       0.00      0.00      0.00         2
  2023-11-06       0.00      0.00      0.00         1
  2023-11-07       0.00      0.00      0.00         1
  2023-11-08       0.00      0.00      0.00         1
  2023-11-16       0.00      0.00      0.00         1
  2023-11-18       0.00      0.00      0.00         2
  2023-11-19       0.00      0.00      0.00         1
  2023-11-24       0.00      0.00      0.00         0

    accuracy                           0.00      2000
   macro avg       0.00      0.00      0.00      2000
weighted avg       0.00      0.00      0.00      2000
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0
in labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Display the classification report and confusion matrix.

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

# Load the dataset
df = pd.read_csv("/content/healthcare_dataset.csv")

# Drop unnecessary columns
df.drop(['Name', 'Date of Admission', 'Doctor', 'Hospital', 'Insurance␣
 ↪Provider', 'Room Number', 'Admission Type', 'Discharge Date', 'Medication',␣
 ↪'Test Results'], axis=1, inplace=True)

# Encode categorical variables
le = LabelEncoder()
df['Gender'] = le.fit_transform(df['Gender'])
df['Blood Type'] = le.fit_transform(df['Blood Type'])
df['Medical Condition'] = le.fit_transform(df['Medical Condition'])
```

```
# Define features and target variable
X = df.drop('Medical Condition', axis=1)
y = df['Medical Condition']

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Train the logistic regression model
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = logreg.predict(X_test)

# Display classification report and confusion matrix
print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       484
           1       0.16      0.73      0.27       494
           2       0.00      0.00      0.00       520
           3       0.18      0.15      0.16       469
           4       0.00      0.00      0.00       527
           5       0.16      0.13      0.14       506

    accuracy                           0.17      3000
   macro avg       0.08      0.17      0.10      3000
weighted avg       0.08      0.17      0.09      3000

Confusion Matrix:
[[  0 361   0  61   0  62]
 [  0 362   0  66   0  66]
 [  0 386   0  62   0  72]
 [  0 335   0  69   0  65]
 [  0 392   0  62   0  73]
 [  0 380   0  61   0  65]]

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
```

```
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

The classification report provides metrics such as precision, recall, F1-score, and support for each class (medical condition in this case). Precision is the ratio of true positive predictions to the total number of positive predictions. It indicates how many of the predicted positive cases are actually positive. Recall, also known as sensitivity, is the ratio of true positive predictions to the total number of actual positive cases. It measures the ability of the model to correctly identify positive cases. F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall. Support is the number of actual occurrences of each class in the test dataset. Confusion Matrix:

The confusion matrix is a table that shows the counts of true positive, false positive, true negative, and false negative predictions. True positive (TP): The number of instances correctly predicted as belonging to the positive class. False positive (FP): The number of instances incorrectly predicted as belonging to the positive class. True negative (TN): The number of instances correctly predicted as not belonging to the positive class. False negative (FN): The number of instances incorrectly predicted as not belonging to the positive class. Interpretation:

You would want to pay attention to metrics like precision, recall, and F1-score for each class to understand how well the model performs for different medical conditions. Higher values for precision, recall, and F1-score indicate better performance. The confusion matrix provides insights into where the model makes mistakes. For example, if there are a high number of false positives or false negatives for a particular class, it indicates areas where the model can be improved.

Compare the performance of Linear Discriminant Analysis and Logistic Regression models.

Linear Discriminant Analysis (LDA):

LDA is a supervised learning algorithm used for classification tasks. It assumes that the features are normally distributed and that the classes have identical covariance matrices. LDA finds linear combinations of features that best separate the classes. LDA provides class probabilities based on Bayes' theorem and assumes a Gaussian distribution for each class. Logistic Regression (LR):

Logistic Regression is another supervised learning algorithm commonly used for binary classification tasks. It models the probability that an instance belongs to a particular class using the logistic function. LR estimates coefficients for the features to maximize the likelihood function. LR does not make strong assumptions about the distribution of the features.

16. Provide insights and recommendations based on the analysis and model evaluations.

erformance Comparison:

Evaluate the performance of both LDA and LR models based on metrics such as accuracy, precision, recall, F1-score, and AUC-ROC. Identify which model performs better across these metrics. It's important to consider both overall performance and performance across different classes, especially if the classes are imbalanced. Model Interpretability:

Consider the interpretability of the models. LDA provides insights into the linear combinations of features that best separate the classes, which can be useful for understanding the relationships between features and classes. LR also offers coefficients for features, aiding in interpretability. Assumptions and Data Distribution:

Assess whether the assumptions of LDA (normality of features, equal covariance matrices) are met by the dataset. If the assumptions are violated, LR might be a more appropriate choice as it is more robust to violations of assumptions. If the data distribution is skewed or non-linear, LR might perform better as it does not make strong assumptions about the distribution of features. Computational Complexity:

Consider the computational complexity of both models, especially for large datasets. LDA typically requires computing the covariance matrix and its inverse, which might be computationally expensive for high-dimensional datasets.