WGU D209

March 27, 2024

[1]:
```
%%html
<style>
.toc-item > li {
    list-style-type: upper-alpha;
}
</style>
```

<IPython.core.display.HTML object>

0.1 Kamal Shaham

0.2 D209: Classification Analysis

<h2>Table of Contents</h2>

```
            <li><a href="#analysis-technique">Analysis technique used</a></li>
            <li><a href="#classification-code">Classification analysis code</a></li>
        </ul>
    </li>
    <li><a href="#data-summary">Data Summary and Implications</a>
    <ul>
            <li><a href="#data-summary">Accuracy and area</a></li>
            <li><a href="#results">Results and implications of data analysis</a></li>
            <li><a href="#limitation">Limitation of this analysis</a></li>
            <li><a href="#course-of-action">Recommended course of action</a></li>
        </ul>
    </li>
    <li><a href="#video">Panopto video</a></li>
    <li><a href="#thirdparty">Third-party code references</a></li>
    <li><a href="#references">References</a></li>
</ul>
```

0.3 A. Research Question

According to the medical data dictionary (D209 Datasets), a patient readmitted to the hospital within a month of discharge is categorized as a readmission. This categorization introduces an intriguing research question: Can we predict patients who are at risk of hospital readmission? By utilizing patient medical data, we aim to employ the k-nearest neighbor method (KNN) to assist with this analysis.

0.3.1 A2. Goal of data analysis

The goal of this data analysis is to develop a machine learning model using k-nearest neighbors (KNN) to assist hospitals in identifying individuals at risk of readmission. According to a study by Rocca et al., readmissions pose a significant burden on both patients and healthcare systems, leading to increased mortality and morbidity. By predicting patient readmissions, hospitals can customize care for patients at all readmission risk levels. This approach aims to not only to save on hospital costs but also enhance patient recoveries.

0.4 B. Method Justification

The k-nearest neighbors (KNN) method attempts to predict the label of a data point by examining a group of the closest labeled data points (k in this instance) and taking a vote. The value of k is either defaulted to 5 when using the standard KNeighborsClassifier function in scikit-learn or can be set by the user. This value adjusts how many data points the function considers when determining the label of the new data point. The method uses distance to determine the neighbors of the data point, with the standard metric being the Euclidean distance.

By converting our categorical data into numerical variables, we can use this data to create a training set for the algorithm and match this to the outcome of the patient being readmitted or not. After the algorithm is trained on the dataset to determine which continuous factors can categorize a patient's likelihood of readmission, its effectiveness will be evaluated using the test dataset. This evaluation will produce a model accuracy score, which ideally will be at or above 95% accurate.

Based on the D209 webinars, we also expect our test data to be classified according to their closest neighbors.

0.4.1 B2. Method assumption

The k-nearest neighbors classification method assumes that data points which exist in close proximity to each other are highly similar, whereas if a data point is far away from another group it is dissimilar to those data points. For example, an unlabeled data point should exist next to a similar labeled data point in the data set (D209 Webinar).

0.4.2 B3. Python packages used

- Pandas: A standard data science import that makes it easy to read, sort, clean, and prepare data for analysis.
- Numpy: Assists with reading and visualizing data, also provides tools for calculations.
- Matplotlib: Provides easy-to-understand graphs and tools to visualize reports and data points.
- Seaborn: A package used for intuitive graphs and reports, will be used for a correlation matrix.
- Scikit-learn: The standard package for using the k-nearest neighbors technique and checking how well it works. Provides methods for training, testing, splitting, and fitting data. Also used to select the most significant features using the SelectKBest function.

C. Data Preparation/Cleaning One data preprocessing goal will be to remove insignificant features from our dataset. This step is crucial for selecting which features to use for our initial model. We will utilize the SelectKBest function within the sklearn package to include only features with a p-value of 0.05 or below, this is meant to reduce noise in the data.

C2. Variable Statistical Summaries Geographic variables of the patient, such as population, city, and state, will not be included in our analysis as they do not provide benefits for our purposes. However, there is potential for different modeling techniques to be applied to these columns for further analysis. To identify which features are significant for this analysis, we run the SelectKBest function and only select variables with p-values below 0.05.

The table below includes the independent variables (with 'ReAdmis' as our dependent variable), their data types, their classification as categorical or continuous, and sample data from each column. A summary statistics table is generated, detailing each variable's standard deviation, interquartile ranges, mean, and median (noted as the 50% value in the output). The categorical variables in the data will be converted to numerical types. Histograms and box plots have been generated for each variable to check for outliers and distribution.

- ReAdmis: Character (binary categorical), Example: No
- Initial_admin: Character (nominal categorical), Example: Emergency Admission
- Marital: Character (nominal categorical), Example: Divorced
- Services: Character (Nominal categorical), Example: Blood Work
- Initial_days: Numeric, Example: 10.585770
- TotalCharge: Numeric, Example: 3726.702860

C3. Data Preparation Steps

The data will need to be inspected prior to any classification analysis. We first check for missing and duplicate values in our dataset. Missing values can potentially be filled with zeros or populated with the average of the respective column. Duplicated data is also checked to ensure each row is unique and duplication does not exist. Outliers will be identified using box plots, while the distributions will be represented with histograms. Patient location and job demographics (such as state, city, job, area, etc.) will not be beneficial to our analysis and thus can be removed. Several column names, such as Item1 through Item8, will be renamed for clarity during this analysis. Any categorical variables must be converted to numerical values.

For categorical variables with more than two levels that cannot be sorted ordinally, one-hot encoding will be utilized. Any columns with Yes/No values or Low, Medium, and High will be converted to numerical values. We then identify the features with the most significant p-values by using the SelectKBest function from the sklearn package. This function will return p-values that are less than 0.05, which are the columns we will use for our initial analysis. We then print our summary statistics and newly created column names. Lastly, we scale and standardize our data before running our initial model.

[2]:
```
%matplotlib inline

# import our statistical libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# import our initial dataset
data=pd.read_csv('medical_clean.csv')

#view first 5 rows and column information
print(data.head())
print(data.columns)

#check for missing/null values
print(data.isnull().sum())

#check for duplicate values of any rows
print(data.duplicated().any())

# Check for duplicate values based on customer_id unique key
print(data.duplicated('Customer_id').any())

# remove unused columns
data.drop(['CaseOrder', 'Customer_id','Interaction', 'UID', 'City', 'State',␣
 ↪'County', 'TimeZone', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'Job',␣
 ↪'Children'], axis=1, inplace=True)

# view data after columns dropped
print(data.head())
print(data.columns)
print(data.info())
```

```python
# rename unclear survey response columns
survey_col_names = {
    'Item1': 'Timely_admis',
    'Item2': 'Timely_treat',
    'Item3': 'Timely_visits',
    'Item4': 'Reliability',
    'Item5': 'Options',
    'Item6': 'Hours_treat',
    'Item7': 'Courteous_staff',
    'Item8': 'Active_listening'
}
data = data.rename(columns=survey_col_names)

# Variable statistics to check distributions
print(data.describe(include='all'))
```

```
   CaseOrder Customer_id                           Interaction  \
0          1     C412403  8cd49b13-f45a-4b47-a2bd-173ffa932c2f
1          2     Z919181  d2450b70-0337-4406-bdbb-bc1037f1734c
2          3     F995323  a2057123-abf5-4a2c-abad-8ffe33512562
3          4     A879973  1dec528d-eb34-4079-adce-0d7a40e82205
4          5     C544523  5885f56b-d6da-43a3-8760-83583af94266

                                UID          City State        County    Zip  \
0  3a83ddb66e2ae73798bdf1d705dc0932           Eva    AL        Morgan  35621
1  176354c5eef714957d486009feabf195      Marianna    FL       Jackson  32446
2  e19a0fa00aeda885b8a436757e889bc9   Sioux Falls    SD     Minnehaha  57110
3  cd17d7b6d152cb6f23957346d11c3f07  New Richland    MN        Waseca  56072
4  d2f0425877b10ed6bb381f3e2579424a    West Point    VA  King William  23181

        Lat       Lng  …  TotalCharge Additional_charges Item1 Item2  Item3  \
0  34.34960 -86.72508  …  3726.702860       17939.403420     3     3      2
1  30.84513 -85.22907  …  4193.190458       17612.998120     3     4      3
2  43.54321 -96.63772  …  2434.234222       17505.192460     2     4      4
3  43.89744 -93.51479  …  2127.830423       12993.437350     3     5      5
4  37.59894 -76.88958  …  2113.073274        3716.525786     2     1      3

   Item4  Item5 Item6 Item7 Item8
0      2      4     3     3     4
1      4      4     4     3     3
2      4      3     4     3     3
3      3      4     5     5     5
4      3      5     3     4     3

[5 rows x 50 columns]
Index(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
       'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',
```

```
       'Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis',
       'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke',
       'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
       'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
       'Reflux_esophagitis', 'Asthma', 'Services', 'Initial_days',
       'TotalCharge', 'Additional_charges', 'Item1', 'Item2', 'Item3', 'Item4',
       'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
CaseOrder             0
Customer_id           0
Interaction           0
UID                   0
City                  0
State                 0
County                0
Zip                   0
Lat                   0
Lng                   0
Population            0
Area                  0
TimeZone              0
Job                   0
Children              0
Age                   0
Income                0
Marital               0
Gender                0
ReAdmis               0
VitD_levels           0
Doc_visits            0
Full_meals_eaten      0
vitD_supp             0
Soft_drink            0
Initial_admin         0
HighBlood             0
Stroke                0
Complication_risk     0
Overweight            0
Arthritis             0
Diabetes              0
Hyperlipidemia        0
BackPain              0
Anxiety               0
Allergic_rhinitis     0
Reflux_esophagitis    0
Asthma                0
Services              0
```

```
Initial_days         0
TotalCharge          0
Additional_charges   0
Item1                0
Item2                0
Item3                0
Item4                0
Item5                0
Item6                0
Item7                0
Item8                0
dtype: int64
False
False
   Age    Income   Marital  Gender ReAdmis  VitD_levels  Doc_visits  \
0   53  86575.93  Divorced    Male      No    19.141466           6
1   51  46805.99   Married  Female      No    18.940352           4
2   53  14370.14   Widowed  Female      No    18.057507           4
3   78  39741.49   Married    Male      No    16.576858           4
4   22   1209.56   Widowed  Female      No    17.439069           5

   Full_meals_eaten  vitD_supp Soft_drink  …  TotalCharge  \
0                 0          0         No  …  3726.702860
1                 2          1         No  …  4193.190458
2                 1          0         No  …  2434.234222
3                 1          0         No  …  2127.830423
4                 0          2        Yes  …  2113.073274

  Additional_charges Item1 Item2 Item3 Item4 Item5 Item6 Item7 Item8
0       17939.403420     3     3     2     2     4     3     3     4
1       17612.998120     3     4     3     4     4     4     3     3
2       17505.192460     2     4     4     4     3     4     3     3
3       12993.437350     3     5     5     3     4     5     5     5
4        3716.525786     2     1     3     3     5     3     4     3

[5 rows x 35 columns]
Index(['Age', 'Income', 'Marital', 'Gender', 'ReAdmis', 'VitD_levels',
       'Doc_visits', 'Full_meals_eaten', 'vitD_supp', 'Soft_drink',
       'Initial_admin', 'HighBlood', 'Stroke', 'Complication_risk',
       'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
       'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma',
       'Services', 'Initial_days', 'TotalCharge', 'Additional_charges',
       'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 35 columns):
 #   Column              Non-Null Count  Dtype
```

```
---  ------              --------------  -----
 0   Age                 10000 non-null  int64
 1   Income              10000 non-null  float64
 2   Marital             10000 non-null  object
 3   Gender              10000 non-null  object
 4   ReAdmis             10000 non-null  object
 5   VitD_levels         10000 non-null  float64
 6   Doc_visits          10000 non-null  int64
 7   Full_meals_eaten    10000 non-null  int64
 8   vitD_supp           10000 non-null  int64
 9   Soft_drink          10000 non-null  object
 10  Initial_admin       10000 non-null  object
 11  HighBlood           10000 non-null  object
 12  Stroke              10000 non-null  object
 13  Complication_risk   10000 non-null  object
 14  Overweight          10000 non-null  object
 15  Arthritis           10000 non-null  object
 16  Diabetes            10000 non-null  object
 17  Hyperlipidemia      10000 non-null  object
 18  BackPain            10000 non-null  object
 19  Anxiety             10000 non-null  object
 20  Allergic_rhinitis   10000 non-null  object
 21  Reflux_esophagitis  10000 non-null  object
 22  Asthma              10000 non-null  object
 23  Services            10000 non-null  object
 24  Initial_days        10000 non-null  float64
 25  TotalCharge         10000 non-null  float64
 26  Additional_charges  10000 non-null  float64
 27  Item1               10000 non-null  int64
 28  Item2               10000 non-null  int64
 29  Item3               10000 non-null  int64
 30  Item4               10000 non-null  int64
 31  Item5               10000 non-null  int64
 32  Item6               10000 non-null  int64
 33  Item7               10000 non-null  int64
 34  Item8               10000 non-null  int64
dtypes: float64(5), int64(12), object(18)
memory usage: 2.7+ MB
None
                Age        Income  Marital  Gender ReAdmis   VitD_levels  \
count  10000.000000  10000.000000    10000   10000   10000  10000.000000
unique          NaN           NaN        5       3       2           NaN
top             NaN           NaN  Widowed  Female      No           NaN
freq            NaN           NaN     2045    5018    6331           NaN
mean      53.511700  40490.495160      NaN     NaN     NaN     17.964262
std       20.638538  28521.153293      NaN     NaN     NaN      2.017231
min       18.000000    154.080000      NaN     NaN     NaN      9.806483
25%       36.000000  19598.775000      NaN     NaN     NaN     16.626439
```

```
50%        53.000000   33768.420000      NaN     NaN     NaN     17.951122
75%        71.000000   54296.402500      NaN     NaN     NaN     19.347963
max        89.000000  207249.100000      NaN     NaN     NaN     26.394449

          Doc_visits  Full_meals_eaten     vitD_supp Soft_drink  …  \
count   10000.000000      10000.000000  10000.000000      10000  …
unique           NaN               NaN           NaN          2  …
top              NaN               NaN           NaN         No  …
freq             NaN               NaN           NaN       7425  …
mean        5.012200          1.001400      0.398900        NaN  …
std         1.045734          1.008117      0.628505        NaN  …
min         1.000000          0.000000      0.000000        NaN  …
25%         4.000000          0.000000      0.000000        NaN  …
50%         5.000000          1.000000      0.000000        NaN  …
75%         6.000000          2.000000      1.000000        NaN  …
max         9.000000          7.000000      5.000000        NaN  …

         TotalCharge Additional_charges  Timely_admis  Timely_treat  \
count   10000.000000       10000.000000  10000.000000  10000.000000
unique           NaN                NaN           NaN           NaN
top              NaN                NaN           NaN           NaN
freq             NaN                NaN           NaN           NaN
mean     5312.172769       12934.528587      3.518800      3.506700
std      2180.393838        6542.601544      1.031966      1.034825
min      1938.312067        3125.703000      1.000000      1.000000
25%      3179.374015        7986.487755      3.000000      3.000000
50%      5213.952000       11573.977735      4.000000      3.000000
75%      7459.699750       15626.490000      4.000000      4.000000
max      9180.728000       30566.070000      8.000000      7.000000

       Timely_visits   Reliability       Options   Hours_treat  \
count   10000.000000  10000.000000  10000.000000  10000.000000
unique           NaN           NaN           NaN           NaN
top              NaN           NaN           NaN           NaN
freq             NaN           NaN           NaN           NaN
mean        3.511100      3.515100      3.496900      3.522500
std         1.032755      1.036282      1.030192      1.032376
min         1.000000      1.000000      1.000000      1.000000
25%         3.000000      3.000000      3.000000      3.000000
50%         4.000000      4.000000      3.000000      4.000000
75%         4.000000      4.000000      4.000000      4.000000
max         8.000000      7.000000      7.000000      7.000000

       Courteous_staff Active_listening
count     10000.000000     10000.000000
unique             NaN              NaN
top                NaN              NaN
freq               NaN              NaN
```

```
mean          3.494000         3.509700
std           1.021405         1.042312
min           1.000000         1.000000
25%           3.000000         3.000000
50%           3.000000         3.000000
75%           4.000000         4.000000
max           7.000000         7.000000

[11 rows x 35 columns]
```

[3]:

```
# Encoding categorical binary columns as numeric.
bin_cols = ['ReAdmis', 'HighBlood', 'Stroke', 'Arthritis', 'Diabetes',
            'Anxiety', 'Asthma', 'Soft_drink', 'Overweight',␣
 ↪'Allergic_rhinitis', 'BackPain', 'Hyperlipidemia', 'Reflux_esophagitis']
bin_dict = {'Yes': 1, 'No': 0}
for col in bin_cols:
    data[col] = data[col].replace(bin_dict)

# get categorical columns for dummy variable generation
categorical_cols = ['Marital',␣
 ↪'Gender','Initial_admin','Complication_risk','Services']

# generate dummy variables
data = pd.get_dummies(data, columns=categorical_cols, drop_first=False)

# remove any spaces after dummy variable generation
data.columns = data.columns.str.replace(' ', '')

# inspect dummy variables
print(data.info())
print(data.head())

# display all columns
pd.set_option('display.max_columns', None)
# display all the rows
pd.set_option('display.max_rows', None)

# summary stats and check newly created columns
print(data.describe(include='all'))

X = data.drop(["ReAdmis"],axis=1)
y = data["ReAdmis"]

feature_names = X.columns
# import the selectkbest function from sklearn
from sklearn.feature_selection import SelectKBest, f_classif
#use selectkbest to identify all significant features in the data set
```

```python
skbest = SelectKBest(score_func = f_classif, k='all')
X_new = skbest.fit_transform(X, y)
X_new.shape

### Finding P-values to select statistically significant features
p_values = pd.DataFrame({'Feature': X.columns,
                         'p_value':skbest.pvalues_}).sort_values('p_value')
print(p_values)

# only dislay features with p-value < 0.05
features_to_keep = p_values['Feature'][p_values['p_value'] < .05]
# print the name of the selected features
print(features_to_keep)

#variables significant to this analysis
sig_data = data[['ReAdmis', 'TotalCharge', 'Initial_days', 'Services_MRI',␣
 ↪'Services_CTScan', 'Services_BloodWork', 'Services_Intravenous',␣
 ↪'Marital_Divorced', 'Marital_NeverMarried', 'Marital_Separated',␣
 ↪'Marital_Married', 'Marital_Widowed',  'Initial_admin_ObservationAdmission',␣
 ↪'Initial_admin_EmergencyAdmission', 'Initial_admin_ElectiveAdmission', ]]

# display columns used for initial model analysis
print(sig_data.columns)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 48 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Age                                 10000 non-null  int64
 1   Income                              10000 non-null  float64
 2   ReAdmis                             10000 non-null  int64
 3   VitD_levels                         10000 non-null  float64
 4   Doc_visits                          10000 non-null  int64
 5   Full_meals_eaten                    10000 non-null  int64
 6   vitD_supp                           10000 non-null  int64
 7   Soft_drink                          10000 non-null  int64
 8   HighBlood                           10000 non-null  int64
 9   Stroke                              10000 non-null  int64
 10  Overweight                          10000 non-null  int64
 11  Arthritis                           10000 non-null  int64
 12  Diabetes                            10000 non-null  int64
 13  Hyperlipidemia                      10000 non-null  int64
 14  BackPain                            10000 non-null  int64
 15  Anxiety                             10000 non-null  int64
 16  Allergic_rhinitis                   10000 non-null  int64
 17  Reflux_esophagitis                  10000 non-null  int64
 18  Asthma                              10000 non-null  int64
```

```
 19  Initial_days                          10000 non-null  float64
 20  TotalCharge                           10000 non-null  float64
 21  Additional_charges                    10000 non-null  float64
 22  Timely_admis                          10000 non-null  int64
 23  Timely_treat                          10000 non-null  int64
 24  Timely_visits                         10000 non-null  int64
 25  Reliability                           10000 non-null  int64
 26  Options                               10000 non-null  int64
 27  Hours_treat                           10000 non-null  int64
 28  Courteous_staff                       10000 non-null  int64
 29  Active_listening                      10000 non-null  int64
 30  Marital_Divorced                      10000 non-null  uint8
 31  Marital_Married                       10000 non-null  uint8
 32  Marital_NeverMarried                  10000 non-null  uint8
 33  Marital_Separated                     10000 non-null  uint8
 34  Marital_Widowed                       10000 non-null  uint8
 35  Gender_Female                         10000 non-null  uint8
 36  Gender_Male                           10000 non-null  uint8
 37  Gender_Nonbinary                      10000 non-null  uint8
 38  Initial_admin_ElectiveAdmission       10000 non-null  uint8
 39  Initial_admin_EmergencyAdmission      10000 non-null  uint8
 40  Initial_admin_ObservationAdmission    10000 non-null  uint8
 41  Complication_risk_High                10000 non-null  uint8
 42  Complication_risk_Low                 10000 non-null  uint8
 43  Complication_risk_Medium              10000 non-null  uint8
 44  Services_BloodWork                    10000 non-null  uint8
 45  Services_CTScan                       10000 non-null  uint8
 46  Services_Intravenous                  10000 non-null  uint8
 47  Services_MRI                          10000 non-null  uint8
dtypes: float64(5), int64(25), uint8(18)
memory usage: 2.5 MB
None
   Age    Income  ReAdmis  VitD_levels  Doc_visits  Full_meals_eaten  \
0   53  86575.93        0    19.141466           6                 0
1   51  46805.99        0    18.940352           4                 2
2   53  14370.14        0    18.057507           4                 1
3   78  39741.49        0    16.576858           4                 1
4   22   1209.56        0    17.439069           5                 0

   vitD_supp  Soft_drink  HighBlood  Stroke  …  \
0          0           0          1       0  …
1          1           0          1       0  …
2          0           0          1       0  …
3          0           0          0       1  …
4          2           1          0       0  …

   Initial_admin_ElectiveAdmission  Initial_admin_EmergencyAdmission  \
0                                0                                 1
```

```
1                                     0                                     1
2                                     1                                     0
3                                     1                                     0
4                                     1                                     0

   Initial_admin_ObservationAdmission  Complication_risk_High  \
0                                   0                       0
1                                   0                       1
2                                   0                       0
3                                   0                       0
4                                   0                       0

   Complication_risk_Low  Complication_risk_Medium  Services_BloodWork  \
0                      0                         1                   1
1                      0                         0                   0
2                      0                         1                   1
3                      0                         1                   1
4                      1                         0                   0

   Services_CTScan  Services_Intravenous  Services_MRI
0                0                     0             0
1                0                     1             0
2                0                     0             0
3                0                     0             0
4                1                     0             0

[5 rows x 48 columns]
               Age         Income       ReAdmis   VitD_levels    Doc_visits  \
count  10000.000000   10000.000000  10000.000000  10000.000000  10000.000000
mean      53.511700   40490.495160      0.366900     17.964262      5.012200
std       20.638538   28521.153293      0.481983      2.017231      1.045734
min       18.000000     154.080000      0.000000      9.806483      1.000000
25%       36.000000   19598.775000      0.000000     16.626439      4.000000
50%       53.000000   33768.420000      0.000000     17.951122      5.000000
75%       71.000000   54296.402500      1.000000     19.347963      6.000000
max       89.000000  207249.100000      1.000000     26.394449      9.000000

       Full_meals_eaten     vitD_supp    Soft_drink     HighBlood  \
count      10000.000000  10000.000000  10000.000000  10000.000000
mean           1.001400      0.398900      0.257500      0.409000
std            1.008117      0.628505      0.437279      0.491674
min            0.000000      0.000000      0.000000      0.000000
25%            0.000000      0.000000      0.000000      0.000000
50%            1.000000      0.000000      0.000000      0.000000
75%            2.000000      1.000000      1.000000      1.000000
max            7.000000      5.000000      1.000000      1.000000

             Stroke    Overweight     Arthritis      Diabetes  Hyperlipidemia  \
```

```
count  10000.000000  10000.000000  10000.000000  10000.00000   10000.000000
mean       0.199300      0.709400      0.357400      0.27380       0.337200
std        0.399494      0.454062      0.479258      0.44593       0.472777
min        0.000000      0.000000      0.000000      0.00000       0.000000
25%        0.000000      0.000000      0.000000      0.00000       0.000000
50%        0.000000      1.000000      0.000000      0.00000       0.000000
75%        0.000000      1.000000      1.000000      1.00000       1.000000
max        1.000000      1.000000      1.000000      1.00000       1.000000

           BackPain       Anxiety  Allergic_rhinitis  Reflux_esophagitis  \
count  10000.000000  10000.000000       10000.000000        10000.000000
mean       0.411400      0.321500           0.394100            0.413500
std        0.492112      0.467076           0.488681            0.492486
min        0.000000      0.000000           0.000000            0.000000
25%        0.000000      0.000000           0.000000            0.000000
50%        0.000000      0.000000           0.000000            0.000000
75%        1.000000      1.000000           1.000000            1.000000
max        1.000000      1.000000           1.000000            1.000000

            Asthma  Initial_days   TotalCharge  Additional_charges  \
count  10000.00000  10000.000000  10000.000000        10000.000000
mean       0.28930     34.455299   5312.172769        12934.528587
std        0.45346     26.309341   2180.393838         6542.601544
min        0.00000      1.001981   1938.312067         3125.703000
25%        0.00000      7.896215   3179.374015         7986.487755
50%        0.00000     35.836244   5213.952000        11573.977735
75%        1.00000     61.161020   7459.699750        15626.490000
max        1.00000     71.981490   9180.728000        30566.070000

        Timely_admis  Timely_treat  Timely_visits   Reliability       Options  \
count  10000.000000  10000.000000   10000.000000  10000.000000  10000.000000
mean       3.518800      3.506700       3.511100      3.515100      3.496900
std        1.031966      1.034825       1.032755      1.036282      1.030192
min        1.000000      1.000000       1.000000      1.000000      1.000000
25%        3.000000      3.000000       3.000000      3.000000      3.000000
50%        4.000000      3.000000       4.000000      4.000000      3.000000
75%        4.000000      4.000000       4.000000      4.000000      4.000000
max        8.000000      7.000000       8.000000      7.000000      7.000000

         Hours_treat  Courteous_staff  Active_listening  Marital_Divorced  \
count  10000.000000     10000.000000      10000.000000      10000.000000
mean       3.522500         3.494000          3.509700          0.196100
std        1.032376         1.021405          1.042312          0.397065
min        1.000000         1.000000          1.000000          0.000000
25%        3.000000         3.000000          3.000000          0.000000
50%        4.000000         3.000000          3.000000          0.000000
75%        4.000000         4.000000          4.000000          0.000000
max        7.000000         7.000000          7.000000          1.000000
```

	Marital_Married	Marital_NeverMarried	Marital_Separated	\
count	10000.000000	10000.000000	10000.000000	
mean	0.202300	0.198400	0.198700	
std	0.401735	0.398815	0.399042	
min	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	
max	1.000000	1.000000	1.000000	

	Marital_Widowed	Gender_Female	Gender_Male	Gender_Nonbinary	\
count	10000.000000	10000.000000	10000.000000	10000.000000	
mean	0.204500	0.501800	0.476800	0.021400	
std	0.403356	0.500022	0.499486	0.144721	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	1.000000	0.000000	0.000000	
75%	0.000000	1.000000	1.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	

	Initial_admin_ElectiveAdmission	Initial_admin_EmergencyAdmission	\
count	10000.000000	10000.000000	
mean	0.250400	0.506000	
std	0.433265	0.499989	
min	0.000000	0.000000	
25%	0.000000	0.000000	
50%	0.000000	1.000000	
75%	1.000000	1.000000	
max	1.000000	1.000000	

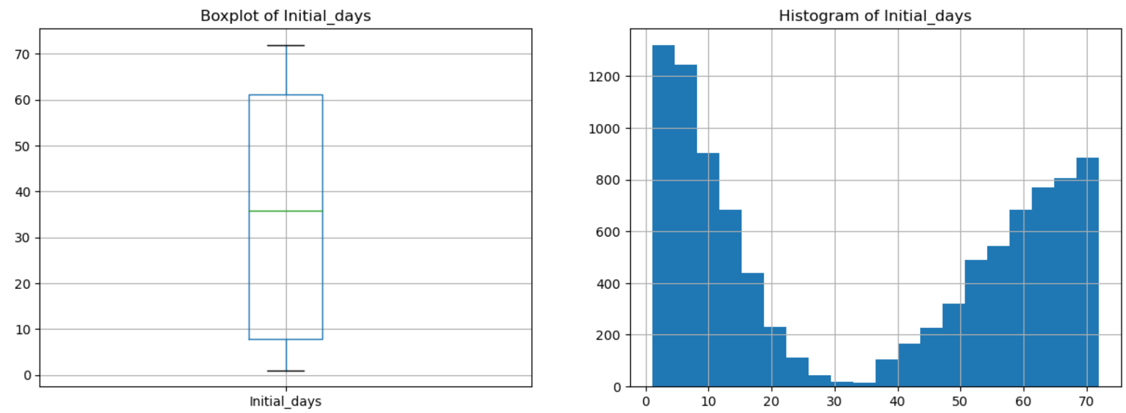
	Initial_admin_ObservationAdmission	Complication_risk_High	\
count	10000.000000	10000.000000	
mean	0.243600	0.335800	
std	0.429276	0.472293	
min	0.000000	0.000000	
25%	0.000000	0.000000	
50%	0.000000	0.000000	
75%	0.000000	1.000000	
max	1.000000	1.000000	

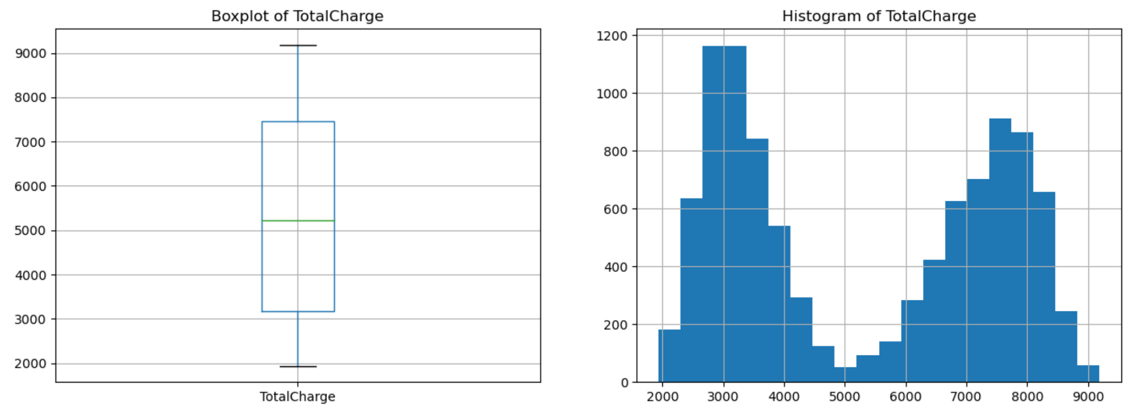
	Complication_risk_Low	Complication_risk_Medium	Services_BloodWork	\
count	10000.000000	10000.000000	10000.000000	
mean	0.212500	0.451700	0.526500	
std	0.409097	0.497687	0.499322	
min	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	1.000000	

```
75%                 0.000000                  1.000000            1.000000
max                 1.000000                  1.000000            1.000000

       Services_CTScan  Services_Intravenous  Services_MRI
count     10000.000000          10000.000000  10000.000000
mean          0.122500              0.313000      0.038000
std           0.327879              0.463738      0.191206
min           0.000000              0.000000      0.000000
25%           0.000000              0.000000      0.000000
50%           0.000000              0.000000      0.000000
75%           0.000000              1.000000      0.000000
max           1.000000              1.000000      1.000000
                                 Feature   p_value
19                           TotalCharge  0.000000
18                          Initial_days  0.000000
44                       Services_CTScan  0.014707
29                      Marital_Divorced  0.030143
45                  Services_Intravenous  0.042233
38      Initial_admin_EmergencyAdmission  0.048766
17                                Asthma  0.086677
26                           Hours_treat  0.091166
21                          Timely_admis  0.093273
28                      Active_listening  0.094149
0                                    Age  0.113891
20                    Additional_charges  0.173237
13                              BackPain  0.183133
4                       Full_meals_eaten  0.223574
39  Initial_admin_ObservationAdmission  0.231263
23                          Timely_visits  0.242104
34                         Gender_Female  0.243554
1                                 Income  0.250029
5                              vitD_supp  0.269697
37       Initial_admin_ElectiveAdmission  0.276650
35                           Gender_Male  0.326515
46                          Services_MRI  0.351962
9                             Overweight  0.390612
33                       Marital_Widowed  0.390638
6                             Soft_drink  0.441195
10                             Arthritis  0.443546
31                  Marital_NeverMarried  0.496533
30                       Marital_Married  0.509864
36                     Gender_Nonbinary  0.520390
25                               Options  0.574553
16                    Reflux_esophagitis  0.587736
27                       Courteous_staff  0.618928
15                     Allergic_rhinitis  0.641923
12                          Hyperlipidemia  0.666731
2                             VitD_levels  0.683120
```

```
40              Complication_risk_High  0.690952
11                            Diabetes  0.759776
42            Complication_risk_Medium  0.779605
22                        Timely_treat  0.808596
14                             Anxiety  0.809868
7                            HighBlood  0.820441
24                         Reliability  0.842855
41               Complication_risk_Low  0.905636
8                               Stroke  0.926830
43                  Services_BloodWork  0.942745
32                  Marital_Separated  0.957280
3                           Doc_visits  0.980401
19                         TotalCharge
18                        Initial_days
44                    Services_CTScan
29                   Marital_Divorced
45               Services_Intravenous
38    Initial_admin_EmergencyAdmission
Name: Feature, dtype: object
Index(['ReAdmis', 'TotalCharge', 'Initial_days', 'Services_MRI',
       'Services_CTScan', 'Services_BloodWork', 'Services_Intravenous',
       'Marital_Divorced', 'Marital_NeverMarried', 'Marital_Separated',
       'Marital_Married', 'Marital_Widowed',
       'Initial_admin_ObservationAdmission',
       'Initial_admin_EmergencyAdmission', 'Initial_admin_ElectiveAdmission'],
      dtype='object')
```

[4]:
```python
import matplotlib.pyplot as plt

# select numerical columns for outlier and distribution checks
numerical_columns = ["Initial_days", "TotalCharge"]

# loops through the columns and generates a boxplot and histogram for each
for column in numerical_columns:
    # creates subplots
    fig, ax = plt.subplots(1, 2, figsize=(15, 5))

    # boxplot generation
    data.boxplot(column, ax=ax[0])
    ax[0].set_title(f'Boxplot of {column}')

    # histogram generation
    data[column].hist(ax=ax[1], bins=20)
    ax[1].set_title(f'Histogram of {column}')

    # displays the plots
    plt.show()
```





[5]:
```
from sklearn.preprocessing import StandardScaler

#initialize scaler
scalar = StandardScaler()
#remove dependent variable from features
data_pred = sig_data.drop(['ReAdmis'], axis=1)
#scale and standardize features, fit scaler to data
data_pred_std = pd.DataFrame(scalar.fit_transform(data_pred), columns=data_pred.
 ↪columns)

#combine dependent variable with scaled independent variables
data_resp = sig_data['ReAdmis']
frames = [data_resp, data_pred_std]
data_std = pd.concat(frames, axis=1)
#print scaled data
print(data_std.head())
```

```
   ReAdmis  TotalCharge  Initial_days  Services_MRI  Services_CTScan  \
```

```
0        0   -0.727185   -0.907310   -0.198749       -0.373632
1        0   -0.513228   -0.734595   -0.198749       -0.373632
2        0   -1.319983   -1.128292   -0.198749       -0.373632
3        0   -1.460517   -1.244503   -0.198749       -0.373632
4        0   -1.467285   -1.261991   -0.198749        2.676428

   Services_BloodWork  Services_Intravenous  Marital_Divorced  \
0            0.948333             -0.674985          2.024707
1           -1.054482              1.481516         -0.493899
2            0.948333             -0.674985         -0.493899
3            0.948333             -0.674985         -0.493899
4           -1.054482             -0.674985         -0.493899

   Marital_NeverMarried  Marital_Separated  Marital_Married  Marital_Widowed  \
0             -0.497499          -0.497968        -0.503591        -0.507022
1             -0.497499          -0.497968         1.985738        -0.507022
2             -0.497499          -0.497968        -0.503591         1.972302
3             -0.497499          -0.497968         1.985738        -0.507022
4             -0.497499          -0.497968        -0.503591         1.972302

   Initial_admin_ObservationAdmission  Initial_admin_EmergencyAdmission  \
0                           -0.567496                          0.988071
1                           -0.567496                          0.988071
2                           -0.567496                         -1.012073
3                           -0.567496                         -1.012073
4                           -0.567496                         -1.012073

   Initial_admin_ElectiveAdmission
0                        -0.577966
1                        -0.577966
2                         1.730205
3                         1.730205
4                         1.730205
```

C4. Prepared Data CSV Attached prepared data csv as: prepared-data.csv

[6]:
```python
data_std.to_csv('prepared-data.csv')
```

D. Training and test data sets

A training and test data set was created with 25% of the data used for testing and the other 75% for training. A seed was set to 14 to allow a repeatable result. The training and test data sets were exported into four separate files.

[7]:
```python
# Splitting the data into training and test sets.
from sklearn.model_selection import train_test_split

x = data_std.drop(['ReAdmis'], axis=1).values
y = data_std['ReAdmis'].values
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25,
                                                    random_state=14, stratify=y)

# Print out the proportions to verify the split.
print(f'Total data points: {len(x)}')
print(f'Training data points: {len(X_train)}')
print(f'Testing data points: {len(X_test)}')
print(f'Training set proportion: {len(X_train) / len(x)}')
print(f'Testing set proportion: {len(X_test) / len(x)}')

# export training and testing data
X_train.tofile('med-Xtrain.csv',sep=',')
X_test.tofile('med_Xtest.csv',sep=',')
y_train.tofile('med_Ytrain.csv',sep=',')
y_test.tofile('med_Ytest.csv',sep=',')
```

```
Total data points: 10000
Training data points: 7500
Testing data points: 2500
Training set proportion: 0.75
Testing set proportion: 0.25
```

D2. Analysis technique

Using a 75/25 split of training and test data, a KNN model was created. The default k_neighbors is 5 when the KNeighborsClassifier is used. Using k as 5, we produced a confusion matrix and classification report. The accuracy score for this model is 97% and resulted in 46 false positives. Next, the square root of the total training data points was calculated and used as a rule of thumb (Refining a k-Nearest-Neighbor Classification, n.d.). The result is 86 as the square root, and a new model was created with k now set to 86. A new confusion matrix and classification report was generated. With an accuracy score of just 90% and 175 false positives, the initial model with k set to 5 provided higher accuracy and had fewer false positives.

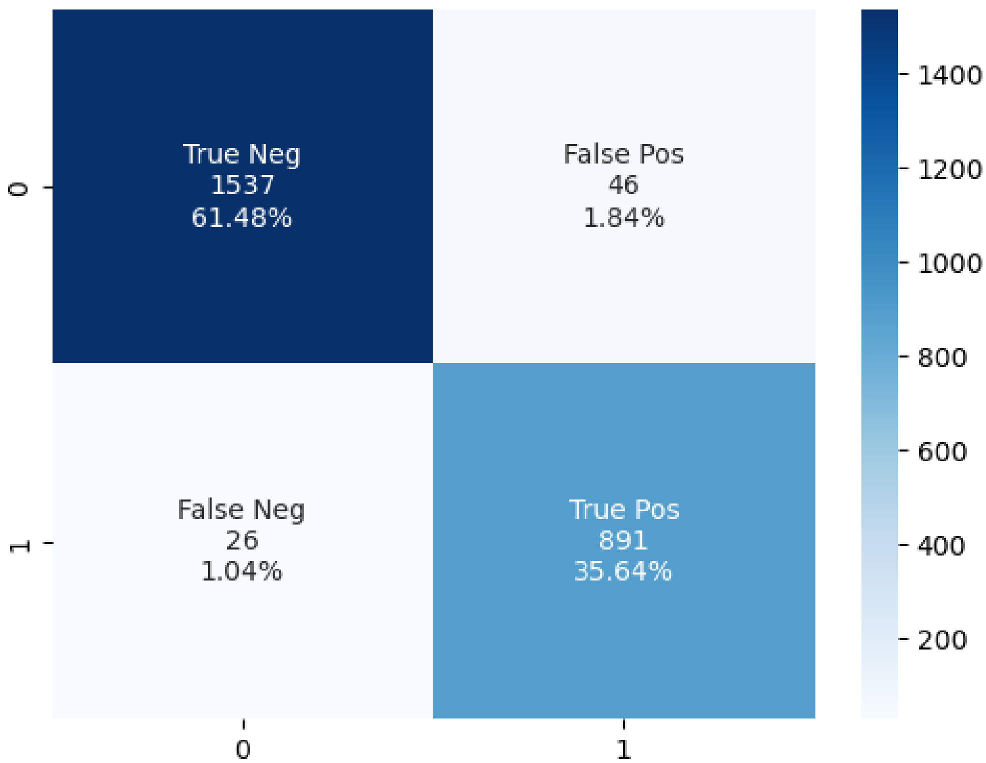
Both models used Euclidean distance, as it is the most widely used metric (Bhat, 2023). Both models also used uniform weighting, meaning all neighbors contribute equally to the classification. Further calculations and hyperparameter tuning were not performed, as an accuracy score above 95% was the goal of our analysis, which was achieved by the initial model.

D3. Classification code

[8]:
```
# Initial kNN classification.
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(metric='euclidean')
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

# initial metrics.
from sklearn.metrics import classification_report
```

```python
from sklearn.metrics import confusion_matrix

#create confusion matrix from sklearn package
matrix = confusion_matrix(y_test, y_pred)
#set column/row names
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     matrix.flatten()/np.sum(matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
#generate heatmap for confusion matrix
sns.heatmap(matrix, annot=labels, fmt='', cmap='Blues')
print(classification_report(y_test, y_pred))

# The initial performance metrics.
total = matrix[0, 0] + matrix[1, 0] + matrix[0, 1] + matrix[1, 1]
accuracy = (matrix[0, 0] + matrix[1, 1]) / total
sensitivity = matrix[1, 1] / (matrix[1, 1] + matrix[1, 0])
specificity = matrix[0, 0] / (matrix[0, 0] + matrix[0, 1])

print('Accuracy: {:.2f}'.format(accuracy))
print('Sensitivity: {:.2f}'.format(sensitivity))
print('Specificity: {:.2f}'.format(specificity))
```

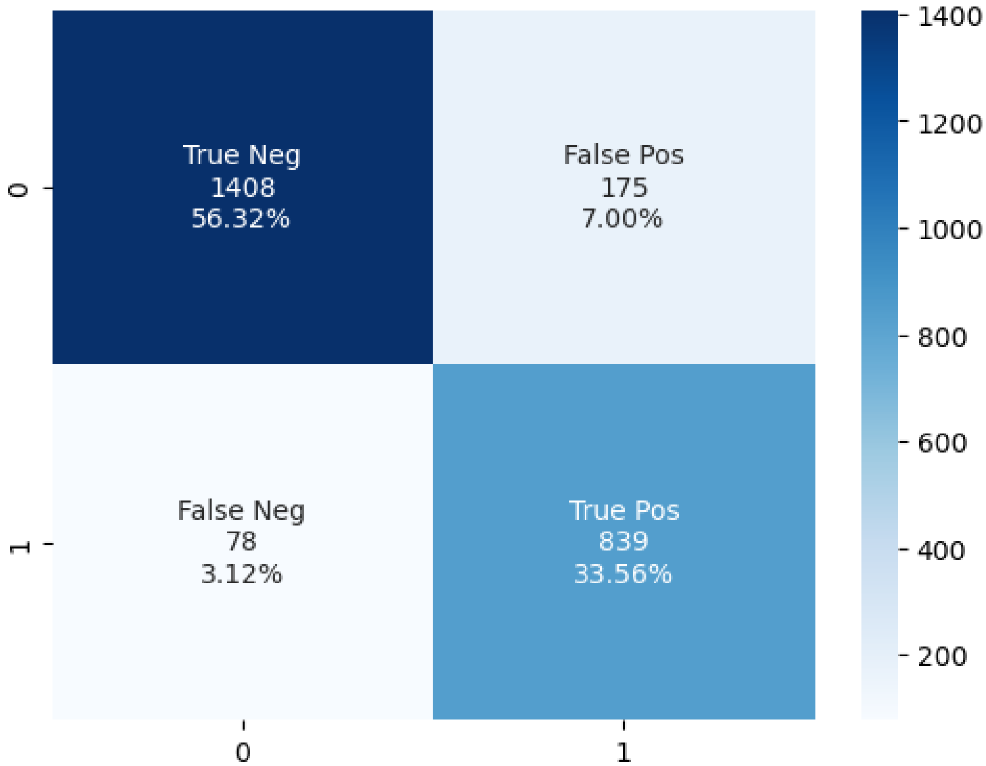
```
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      1583
           1       0.95      0.97      0.96       917

    accuracy                           0.97      2500
   macro avg       0.97      0.97      0.97      2500
weighted avg       0.97      0.97      0.97      2500

Accuracy: 0.97
Sensitivity: 0.97
Specificity: 0.97
```



[9]:
```
# Initial kNN classification.
from sklearn.neighbors import KNeighborsClassifier

#square root of training data points = 86
knn = KNeighborsClassifier(n_neighbors=86, metric='euclidean')
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

# Initial metrics.
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

#create confusion matrix from sklearn package
matrix = confusion_matrix(y_test, y_pred)
#set column/row names
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     matrix.flatten()/np.sum(matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
```

```python
labels = np.asarray(labels).reshape(2,2)
#generate heatmap for confusion matrix
sns.heatmap(matrix, annot=labels, fmt='', cmap='Blues')
print(classification_report(y_test, y_pred))

# The Initial Performance Metrics.
total = matrix[0, 0] + matrix[1, 0] + matrix[0, 1] + matrix[1, 1]
accuracy = (matrix[0, 0] + matrix[1, 1]) / total
sensitivity = matrix[1, 1] / (matrix[1, 1] + matrix[1, 0])
specificity = matrix[0, 0] / (matrix[0, 0] + matrix[0, 1])

print('Accuracy: {:.2f}'.format(accuracy))
print('Sensitivity: {:.2f}'.format(sensitivity))
print('Specificity: {:.2f}'.format(specificity))
```

```
              precision    recall  f1-score   support

           0       0.95      0.89      0.92      1583
           1       0.83      0.91      0.87       917

    accuracy                           0.90      2500
   macro avg       0.89      0.90      0.89      2500
weighted avg       0.90      0.90      0.90      2500

Accuracy: 0.90
Sensitivity: 0.91
Specificity: 0.89
```



E. Data summary and implications

After generating our initial and square root models, we analyzed the accuracy and Area Under the Curve (AUC) scores. The accuracy score is based on the number of correct predictions, in this case, true positives and true negatives. Our initial model had the highest accuracy score of 97%.

The Receiver Operating Characteristic (ROC) plots the true positive rate against the false positive rate at various threshold settings. The AUC provides a single, aggregate measure of performance across all possible classification thresholds (GfG, 2024). The initial model had an AUC of 0.99, which means the model had an almost perfect ability to distinguish between all positive and negative class instances correctly. We used the SelectKBest function from sklearn to include only significant features in our initial model, which is likely contributing to the reduction of noise in our data.

[10]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

knn = KNeighborsClassifier(metric='euclidean')
knn.fit(X_train, y_train)

# Get probabilities for the positive class
y_prob = knn.predict_proba(X_test)[:, 1]

# Calculate AUC
```

```
auc = roc_auc_score(y_test, y_prob)
print(f'AUC: {auc:.2f}')
```

AUC: 0.99

[11]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

knn = KNeighborsClassifier(n_neighbors=86, metric='euclidean')
knn.fit(X_train, y_train)

# Get probabilities for the positive class
y_prob = knn.predict_proba(X_test)[:, 1]

# Calculate AUC
auc = roc_auc_score(y_test, y_prob)
print(f'AUC: {auc:.2f}')
```

AUC: 0.97

E2. Results and implications

The initial model chosen has an accuracy score of 0.97, a precision score of 0.98, and an AUC score of 0.99. The model shows that it can correctly predict that a patient is not readmitted 98% of the time, and the readmission of most patients is predicted correctly 97% of the time.

This proves the model is classifying data accurately and is able to produce true positive results. By using the SelectKBest method, we were able to select only significant features for the model with a p-value less than 0.05. The model can be used to assist with predicting whether a patient is readmitted using the variables identified in our initial model.

E3. Limitation

One limitation is the adjustment of the k-neighbors parameter. By achieving an accuracy score of 97% with the model, we did not utilize hyperparameter tuning to further adjust our k parameter. By performing hyperparameter tuning on the model, we may be able to increase the true positive and true negative ratios. Other distance metrics, such as Manhattan, could be used in our model to determine if they have any effect on the model's accuracy, along with hyperparameter tuning.

E4. Recommended course of action

Analysis of the predictor variables used in this model will allow hospitals to identify which features have the most impact on a patient being readmitted. The accuracy score of this model was 97%, further analysis should be conducted to determine if this was an underfit or overfit of the model. Adjusting the k-neighbors parameter through hyperparameter tuning may provide a more accurate model and speed up predictions. When the analysis of the predictor variables is complete, the hospital can focus its efforts on these factors to reduce the probability of a patient being readmitted.

G. Panopto Video https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=7cdebfd5-4471-454b-9f5c-b1400072c423

H. Third-party Code References

D209 T1. (2022, July 12). [Video]. Panopto. https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=b
ef01-4d1b-a59f-aed100228a93

GfG. (2022, June 27). How to do train test split using Sklearn in Python. GeeksforGeeks. https://www.geeksforgeeks.org/how-to-do-train-test-split-using-sklearn-in-python/

T, D. (2021, December 11). Confusion Matrix Visualization - Dennis T. Medium. https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea

I. References

Bhat, H. (2023, December 14). KNN Algorithm in Python: Implementation with Examples. AlmaBetter. https://www.almabetter.com/bytes/articles/knn-algorithm-python

D208 Datasets. (n.d.). *WGU Performance Assessment*. Tasks.wgu.edu. Retrieved from
`https://tasks.wgu.edu/student/004659020/course/29780017/task/3784/overview`

GfG. (2024, January 25). AUC ROC curve in machine learning. GeeksforGeeks. https://www.geeksforgeeks.org/auc-roc-curve/

Nelson, D. (2020, August 23). What is a KNN (K-Nearest Neighbors)? Unite.AI. https://www.unite.ai/what-is-k-nearest-neighbors/

Refining a k-Nearest-Neighbor classification. (n.d.). University of Notre Dame. https://www3.nd.edu/~steve/computing_with_data/17_Refining_kNN/refining_knn.html

Rocca, H. B., Peden, C. J., Soong, J., Holman, P. A., Bogdanovskaya, M., & Barclay, L. (2020). Reasons for readmission after hospital discharge in patients with chronic diseases—Information from an international dataset. *PLOS ONE*, 15(6), e0233457. https://doi.org/10.1371/journal.pone.0233457

[ ]:
```

```