D209 – Data Mining I - Task 1

WGU M.S. Data Analytics

Lyssa Kline

November 2, 2024

A, Research Question

A1, Question

The research question in scope for this analysis is, “Can hospital readmissions be predicted based on patient characteristics and medical history?” Understanding patient readmissions can help medical organizations manage and potentially reduce costs. This insight could lead to more efficient resource allocation, better staffing, and greater patient transparency. Analyzing the factors affecting readmissions could help identify gaps in patient care, which could lead to targeted efforts to provide actionable insights promptly and improve the quality of patient care.

A2, Analysis Goals

One goal of the data analysis based on the research question defined above would be to develop a model that classifies patients into two groups based on their demographic information, medical stats, and health conditions: those likely to be readmitted to the hospital within a specified time frame and those not likely to be readmitted.

This goal would allow you to evaluate how well patient characteristics (such as age, gender, pre-existing conditions, lab results, etc.) and their medical history (e.g., previous admissions, treatments received) can be used to predict the likelihood of readmission using a k-NN classifier.

B, Method Justification

B1, Classification Method Explanation

The k-NN, or k-Nearest Neighbors classification method, is a non-parametric and instance-based learning algorithm. It makes predictions by finding the "k" closest data points (neighbors) in the dataset based on a distance metric. k-NN works by first defining the features, the algorithm requires a set of features (patient characteristics and medical history) from the dataset. Once a new patient record (aka the test data) is presented to the model, k-NN calculates the distance between this patient and all other patients (aka the training data) in the dataset. Based on the calculated distances, the algorithm identifies the “k” nearest neighbors, which are patients in the dataset that are most like the new patient based on their medical history and characteristics. The value of “k” is a parameter that can be tuned for the model. Once the nearest neighbors are identified, k-NN looks at the class labels (hospital readmission: "Yes" or "No") of these neighbors. The new patient’s predicted outcome will be based on the majority class among the neighbors. For example, if most of the nearest neighbors were readmitted, the model will predict that the new patient will also be readmitted.

One expected outcome of the k-NN classification method is that the k-NN model will predict whether a patient is likely to be readmitted (classification label "Yes") or not likely to be readmitted (classification label "No"). Another expected outcome would be that the performance of the model can be evaluated using metrics such as accuracy, precision, recall, F1-score, and AUC-ROC curves, which will help assess how well the model is predicting hospital readmissions.

Lastly, depending on the chosen features, the k-NN model should help identify patterns or combinations of characteristics that are common among patients who are frequently readmitted.

B2, Assumption of Classification Method

One key assumption in k-NN is that similar patients tend to have similar outcomes. In other words, the model assumes that patients with similar characteristics (e.g., similar medical history, age, and clinical features) will behave similarly regarding hospital readmission. Therefore, by looking at the "k" nearest patients to a new patient, the model assumes that the majority vote of these neighbors is representative of the new patient's likely outcome.

This assumption implies that patients' characteristics are homogeneous to some extent, meaning that patterns in their medical history and clinical data can be generalized across the dataset. However, this assumption can be challenged if the dataset contains very diverse or outlier patients who do not follow the general trends seen in the majority.

In summary, k-NN will analyze the dataset by calculating the similarity (distance) between patients based on selected features and predicting readmission based on the majority outcome of the nearest neighbors. An expected outcome of the analysis is the classification of patients into "readmitted" or "not readmitted" categories. The underlying assumption of this method is that patients with similar characteristics are likely to share similar outcomes regarding hospital readmission, which is key to how the model operates.

B3, Python Packages Justification

Each of the Python packages imported for this analysis plays a critical role in supporting the k-nearest neighbors (k-NN) classification analysis by providing functions for data manipulation, feature selection, scaling, modeling, and evaluation. See the list below showing the imported Python packages used for the k-NN Classification Analysis performed and an explanation of how each package contributes to the workflow.

```
# Place for all packages used in this workflow
import pandas as pd
import numpy as np
from scipy import stats
from sklearn.feature_selection import SelectKBest, f_classif
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import Normalizer
from sklearn.metrics import roc_auc_score, roc_curve
```

Packages Supporting - Core Data Manipulation and Analysis Packages

Pandas provides data structures like Data Frames and Series to handle and manipulate tabular data. Pandas is used to load, clean, preprocess, and structure data for the k-NN model. It

is essential for organizing features (X) and target labels (y), as well as saving results or processed data.

NumPy supports numerical computations and array manipulations. NumPy is often used for matrix operations, handling numerical data, and performing mathematical computations that are necessary for feature selection, scaling, or distance calculations in k-NN.

Stats support numerical computations and array manipulations. Stats from SciPy are used for matrix operations, handling numerical data, and performing mathematical computations that are necessary for feature selection, scaling, or distance calculations in k-NN.

Packages Supporting - Feature Selection, Preprocessing, and Scaling

SelectKBest and f_classif from sklearn.feature selection provides functions for selecting the best features based on statistical tests. SelectKBest selects the top k features with the strongest relationship to the target variable, while f_classif performs ANOVA F-tests for feature relevance. This reduces dimensionality, focusing the k-NN model on the most relevant features.

StandardScaler from sklearn.preprocessing standardizes features by removing the mean and scaling to unit variance. This ensures all continuous features have equal weight in distance calculations by standardizing them. This prevents features with large ranges from dominating others.

LabelEncoder from sklearn.preprocessing encodes categorical labels as integers. This package converts categorical variables with inherent order into numerical format. This is essential because k-NN cannot work directly with non-numeric data.

Normalizer from sklearn.preprocessing scales each sample row to have a unit norm, making then unit vectors. This ensures each data point is scaled consistently, which is useful for distance calculations in k-NN, especially if the absolute magnitudes of features are irrelevant.

Packages Supporting - Model Building and Evaluation

Train_test_split from sklean.model_selection separates data into sets for training the model and testing its performance, ensuring the model is evaluated on unseen data to prevent overfitting.

KNeighborsClassifier from sklearn.neighbors Implement the k-Nearest Neighbors classification algorithm. This is a core function for building, training, and predicting with a k-NN model, making it the central component of the k-NN classification analysis.

accuracy_score, classification_report, confusion_matrix from sklearn.metrics provide metrics for evaluating classification model performance. In k-NN accuracy_score measures the overall accuracy of the k-NN model, classification_report provides detailed metrics like precision, recall, and F1-score for each class, and confusion_matrix shows true positives, true negatives, false positives, and false negatives, allowing for a more granular performance analysis.

cross_val_score from sklearn.model_selection performs cross-validation and returns model accuracy across different folds. In k-NN, this helps in fine-tuning the k parameter by evaluating the model's performance across multiple folds, providing a robust metric to optimize k for the best performance.

roc_auc_score, roc_curve from sklearn.metrics calculates AUC (Area Under the Curve) and plots the ROC (Receiver Operating Characteristic) curve. In k-NN, roc_auc_score evaluates the model's ability to distinguish between classes, and roc_curve visualizes true positive and false positive rates, helping assess model quality, especially in imbalanced datasets.

Packages Supporting – Data Visualization

Seaborn provides advanced plotting functions, especially for statistical data visualizations. In k-NN, this can be used to visualize data distributions, correlations, and model performance metrics to support exploratory data analysis and understand relationships between features.

Matplotlib.pyplot is a fundamental library for creating static, animated, and interactive plots. In k-NN, this is used to create plots such as ROC curves, which are crucial for interpreting model performance and analyzing data visually.

C, Data Preparation

The below sections outline and explain the steps taken to clean, prepare, and output the selected data for a k-NN analysis.

C1, Data Preprocessing Goal

Data preprocessing covers a variety of operations used by data scientists to get the data in a form to be ready for analysis. One goal of data preprocessing when performing a k-NN classification would be to ensure that the dataset is free of missing or incomplete data. K-NN is sensitive to missing values and may fail to produce inaccurate results if such values exist. K-NN relied on distance-based calculations to find the "k" nearest neighbors for making predictions. If some data points are missing, it's impossible to accurately compute the distance between the points, as missing values can distort these calculations.

Missing data can be handled in a few ways: either remove rows with missing data or perform imputation on the missing data. Imputation would allow us to replace the missing numerical values with those available in the dataset (i.e., mean, median, and mode).

Properly handling missing values allows the k-NN algorithm to compute accurate distances between data points, ensuring the model can reliably predict the class labels (e.g., hospital readmission or no readmission) based on complete patient information. If this preprocessing step is ignored, the model may either fail or produce biased and inaccurate predictions, as the distance calculations will be skewed due to incomplete data.

The goal of handling missing values ensures that the k-NN algorithm works on complete, accurate data, leading to better classification performance and more reliable insights from the analysis.

C2, Initial Variable Classification

To answer the research question, "Can hospital readmissions be predicted based on patient characteristics and medical history?". The dependent response variable would be 'ReAdmis,' and the independent predictor variables selected include key patient characteristics/factor information

that may influence the need to be readmitted. To ensure adequate coverage of variables, there were 22 independent variables selected for further analysis.

- ReAdmis (Type: Nominal Categorical Variable) – Describes if the patient was re-admitted or not.
- Age (Type: Discrete Numeric Variable) – Describes the patient's age.
- Gender (Type: Nominal Categorical Variable) – Describes the patient's gender.
- VitD_levels (Type: Continuous Numeric Variable) – Describes the patient's vitamin D levels.
- Doc_visits (Type: Continuous Numeric Variable) – Describes the number of doctors' visits the patient has had.
- Full_meals_eaten (Type: Discrete Numeric Variable) – Describes the number of meals the patient has eaten.
- vitD_supp (Type: Discrete Numeric Variable) – Describes how many vitamin D supplements the patient was given.
- Soft_drink (Type: Nominal Categorical Variable) – Describes if the patient drinks soft drinks or not.
- Initial_admin (Type: Nominal Categorical Variable) – Describes what the patient initial admission was for.
- HighBlood (Type: Nominal Categorical Variable) – Describes if the patient has high blood pressure.
- Stroke (Type: Nominal Categorical Variable) – Describes if the patient has had a stroke.
- Complication_risk (Type: Ordinal Categorical Variable) – Describes if the patient is at risk for complication.
- Overweight (Type: Nominal Categorical Variable) – Describes if the patient is overweight.
- Arthritis (Type: Nominal Categorical Variable) – Describes if the patient has arthritis or not.
- Diabetes (Type: Nominal Categorical Variable) – Describes if the patient has diabetes or not.
- Hyperlipidemia (Type: Nominal Categorical Variable) – Describes if the patient has hyperlipidemia.
- BackPain (Type: Nominal Categorical Variable) – Describes if the patient has back pain.
- Anxiety (Type: Nominal Categorical Variable) – Describes if the patient has anxiety.
- Allergic_rhinitis (Type: Nominal Categorical Variable) – Describes if the patient has allergic rhinitis.
- Reflux_esophagitis (Type: Nominal Categorical Variable) – Describes if the patient has reflux esophagitis or not.
- Asthma (Type: Nominal Categorical Variable) – Describes if the patient has asthma.
- Services (Type: Nominal Categorical Variable) – Describes which services the patient received.
- Initial_days (Type: Continuous Numeric Variable) – Describes the number of days it has been since admission.

C3, Analysis Steps

The medical dataset was selected for use during this analysis. To ensure that the data is clean and ready to use, several steps must be followed.

The first step is to load the dataset into the workbook for analysis. A read_csv function was called to do this.

```
]: # Read in the csv file
   df = pd.read_csv('./medical_clean.csv')
```

Once the dataset is loaded into the workbook, you can begin to evaluate and clean the data structure. During this step, there are a few functions that can be called which allow you to examine the data structure. In doing so, you can begin to identify the numerical, categorical, and text columns. It is key for a k-NN classification analysis to remove null values, missing data, and outliers. It helps in identifying features that may need conversion or removal.

First, a .head() and a .info() were called to display an overview of the data structure, including the datatypes and field names.

```
: # Display first few rows to understand the structure
  df.head()
```

:

	CaseOrder	Customer_id	Interaction	UID	City	State
0	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	Eva	AL
1	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	Marianna	FL
2	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	Sioux Falls	SD
3	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	New Richland	MN
4	5	C544523	5885f56b-d6da-43a3-8760-83583af94266	d2f0425877b10ed6bb381f3e2579424a	West Point	VA

5 rows × 50 columns

```
# Get initial information on the dataset - evaluate the structure and data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   CaseOrder           10000 non-null  int64
 1   Customer_id         10000 non-null  object
 2   Interaction         10000 non-null  object
 3   UID                 10000 non-null  object
 4   City                10000 non-null  object
 5   State               10000 non-null  object
 6   County              10000 non-null  object
 7   Zip                 10000 non-null  int64
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64
 11  Area                10000 non-null  object
 12  TimeZone            10000 non-null  object
 13  Job                 10000 non-null  object
 14  Children            10000 non-null  int64
 15  Age                 10000 non-null  int64
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object
 18  Gender              10000 non-null  object
 19  ReAdmis             10000 non-null  object
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64
 22  Full_meals_eaten    10000 non-null  int64
 23  vitD_supp           10000 non-null  int64
 24  Soft_drink          10000 non-null  object
 25  Initial_admin       10000 non-null  object
 26  HighBlood           10000 non-null  object
 27  Stroke              10000 non-null  object
 28  Complication_risk   10000 non-null  object
 29  Overweight          10000 non-null  object
 30  Arthritis           10000 non-null  object
 31  Diabetes            10000 non-null  object
 32  Hyperlipidemia      10000 non-null  object
 33  BackPain            10000 non-null  object
 34  Anxiety             10000 non-null  object
 35  Allergic_rhinitis   10000 non-null  object
 36  Reflux_esophagitis  10000 non-null  object
 37  Asthma              10000 non-null  object
 38  Services            10000 non-null  object
 39  Initial_days        10000 non-null  float64
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  10000 non-null  float64
 42  Item1               10000 non-null  int64
 43  Item2               10000 non-null  int64
 44  Item3               10000 non-null  int64
 45  Item4               10000 non-null  int64
 46  Item5               10000 non-null  int64
 47  Item6               10000 non-null  int64
 48  Item7               10000 non-null  int64
 49  Item8               10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

Next, to identify missing values, nulls, and outliers, isnull() can be called to determine if there are any null or missing values. Additionally, .describe() can be called to identify the outliers. The output of this command indicated that there were no null values inside of the fields, as well as no outliers that stood out.

```
# Identify any missing values in the dataset -- no missing values identified.
df.isnull().sum()
```

```
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
```

```
# Evaluate numerical features of the dataset - Identify outliers in fields
df.describe()
```

	CaseOrder	Zip	Lat	Lng	Population	Children	Age	Income	VitD_levels	Doc_visits	...	TotalCharge	Additional_charges
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	...	10000.000000	10000.000000
mean	5000.50000	50159.323900	38.751099	-91.243080	9965.253800	2.097200	53.511700	40490.495160	17.964262	5.012200	...	5312.172769	12934.528587
std	2886.89568	27469.588208	5.403085	15.205998	14824.758614	2.163659	20.638538	28521.153293	2.017231	1.045734	...	2180.393838	6542.601544
min	1.00000	610.000000	17.967190	-174.209700	0.000000	0.000000	18.000000	154.080000	9.806483	1.000000	...	1938.312067	3125.703000
25%	2500.75000	27592.000000	35.255120	-97.352982	694.750000	0.000000	36.000000	19598.775000	16.626439	4.000000	...	3179.374015	7986.487755
50%	5000.50000	50207.000000	39.419355	-88.397230	2769.000000	1.000000	53.000000	33768.420000	17.951122	5.000000	...	5213.952000	11573.977735
75%	7500.25000	72411.750000	42.044175	-80.438050	13945.000000	3.000000	71.000000	54296.402500	19.347963	6.000000	...	7459.699750	15626.490000
max	10000.00000	99929.000000	70.560990	-65.290170	122814.000000	10.000000	89.000000	207249.100000	26.394449	9.000000	...	9180.728000	30566.070000

8 rows × 23 columns

Once the dataset has been assessed and cleaned for inconsistencies, we can begin to select features for the analysis.

An initial list of all variables selected for analysis was inputted into a 'selected_start' data frame; this includes a combination of categorical and continuous values deemed adequate for the analysis. These variables are described and classified in the above step.

```
# initially select all variables for analysis
selected_start = df[['ReAdmis','Age','Gender','VitD_levels','Doc_visits','Full_meals_eaten','vitD_supp','Soft_drink','Initial_admin','HighBlood','Stroke','Complication_ris
```

The first step after initially selecting a handful of variables is to encode all the categorical variables. It is key for a k-NN classification analysis to only have numerical values to function properly. This can be done using the .get_dummies() function on the categorical variables. However, it is best practice to include all dummy variables when k>2, and only include one dummy variable when k=2. Therefore, the code below splits the code so that for the variables only containing k=2, the first is dropped. Whereas, for the variables where k>2, we will include all dummy variables in the analysis.

```
# Encode the Categorical Variables
# One-Hot Encoding (creates a new binary column for each categorical variable with just 2 categories)
initial_df_start = pd.get_dummies(selected_start, columns=['Soft_drink','HighBlood','Stroke','Overweight','Arthritis','Diabetes','Hyperlipidemia','BackPain','Anxiety',
print(initial_df_start)
```

```
     ReAdmis  Age  Gender  VitD_levels  Doc_visits  Full_meals_eaten  \
0         No   53    Male    19.141466           6                 0
1         No   51  Female    18.940352           4                 2
2         No   53  Female    18.057507           4                 1
3         No   78    Male    16.576858           4                 1
4         No   22  Female    17.439069           5                 0
...      ...  ...     ...          ...         ...               ...
9995      No   25    Male    16.980860           4                 2
9996     Yes   87    Male    18.177020           5                 0
9997     Yes   45  Female    17.129070           4                 2
9998     Yes   43    Male    19.910430           5                 2
9999     Yes   70  Female    18.388620           5                 0

      vitD_supp          Initial_admin  Initial_days Complication_risk  ...  \
0             0    Emergency Admission     10.585770            Medium  ...
1             1    Emergency Admission     15.129562              High  ...
2             0     Elective Admission      4.772177            Medium  ...
3             0     Elective Admission      1.714879            Medium  ...
4             2     Elective Admission      1.254807               Low  ...
...         ...                    ...           ...               ...  ...
9995          1    Emergency Admission     51.561220            Medium  ...
9996          0     Elective Admission     68.668240            Medium  ...
9997          0     Elective Admission     70.154180              High  ...
9998          1    Emergency Admission     63.356900            Medium  ...
9999          1  Observation Admission     70.850590               Low  ...

     Stroke_Yes  Overweight_Yes  Arthritis_Yes  Diabetes_Yes  \
0         False           False           True          True
1         False            True          False         False
2         False            True          False          True
3          True           False           True         False
4         False           False          False         False
...         ...             ...            ...           ...
9995      False           False          False         False
9996      False            True           True          True
9997      False            True          False         False
9998      False            True          False         False
9999      False            True           True         False

      Hyperlipidemia_Yes  BackPain_Yes  Anxiety_Yes  Allergic_rhinitis_Yes  \
0                  False          True         True                   True
1                  False         False        False                  False
2                  False         False        False                  False
3                  False         False        False                  False
4                   True         False        False                   True
...                  ...           ...          ...                    ...
9995               False         False         True                  False
9996               False         False        False                  False
9997               False         False         True                   True
9998               False          True        False                  False
9999                True         False        False                   True

      Reflux_esophagitis_Yes  Asthma_Yes
0                      False        True
1                       True       False
2                      False       False
3                       True        True
4                      False       False
...                      ...         ...
9995                    True       False
9996                   False        True
9997                   False       False
9998                   False       False
9999                   False       False

[10000 rows x 23 columns]
```

```
# Re-run get dummies without dropping the first row for categorical variables where k>2
initial_df = pd.get_dummies(initial_df_start, columns=['Gender','Services','Complication_risk','Initial_admin'])
print(initial_df)
```

```
     ReAdmis  Age  VitD_levels  Doc_visits  Full_meals_eaten  vitD_supp  \
0         No   53    19.141466           6                 0          0
1         No   51    18.940352           4                 2          1
2         No   53    18.057507           4                 1          0
3         No   78    16.576858           4                 1          0
4         No   22    17.439069           5                 0          2
...      ...  ...          ...         ...               ...        ...
9995      No   25    16.980860           4                 2          1
9996     Yes   87    18.177020           5                 0          0
9997     Yes   45    17.129070           4                 2          0
9998     Yes   43    19.910430           5                 2          1
9999     Yes   70    18.388620           5                 0          1

      Initial_days  Soft_drink_Yes  HighBlood_Yes  Stroke_Yes  ...  \
0        10.585770           False           True       False  ...
1        15.129562           False           True       False  ...
2         4.772177           False           True       False  ...
3         1.714879           False          False        True  ...
4         1.254807            True          False       False  ...
...            ...             ...            ...         ...  ...
9995     51.561220           False           True       False  ...
9996     68.668240           False           True       False  ...
9997     70.154180            True           True       False  ...
9998     63.356900           False          False       False  ...
9999     70.850590           False          False       False  ...

      Services_Blood Work  Services_CT Scan  Services_Intravenous  \
0                    True             False                 False
1                   False             False                  True
2                    True             False                 False
3                    True             False                 False
4                   False              True                 False
...                   ...               ...                   ...
9995                False             False                  True
9996                False              True                 False
9997                False             False                  True
9998                 True             False                 False
9999                 True             False                 False

      Services_MRI  Complication_risk_High  Complication_risk_Low  \
0            False                  False                  False
1            False                   True                  False
2            False                  False                  False
3            False                  False                  False
4            False                  False                   True
...            ...                    ...                    ...
9995         False                  False                  False
9996         False                  False                  False
9997         False                   True                  False
9998         False                  False                  False
9999         False                  False                   True

      Complication_risk_Medium  Initial_admin_Elective Admission  \
0                         True                             False
1                        False                             False
2                         True                              True
3                         True                              True
4                        False                              True
...                        ...                               ...
9995                      True                             False
9996                      True                              True
9997                     False                              True
9998                      True                             False
9999                     False                             False

      Initial_admin_Emergency Admission  Initial_admin_Observation Admission
0                                  True                                False
1                                  True                                False
2                                 False                                False
3                                 False                                False
4                                 False                                False
...                                 ...                                  ...
9995                               True                                False
9996                              False                                False
9997                              False                                False
9998                               True                                False
9999                              False                                 True

[10000 rows x 32 columns]
```

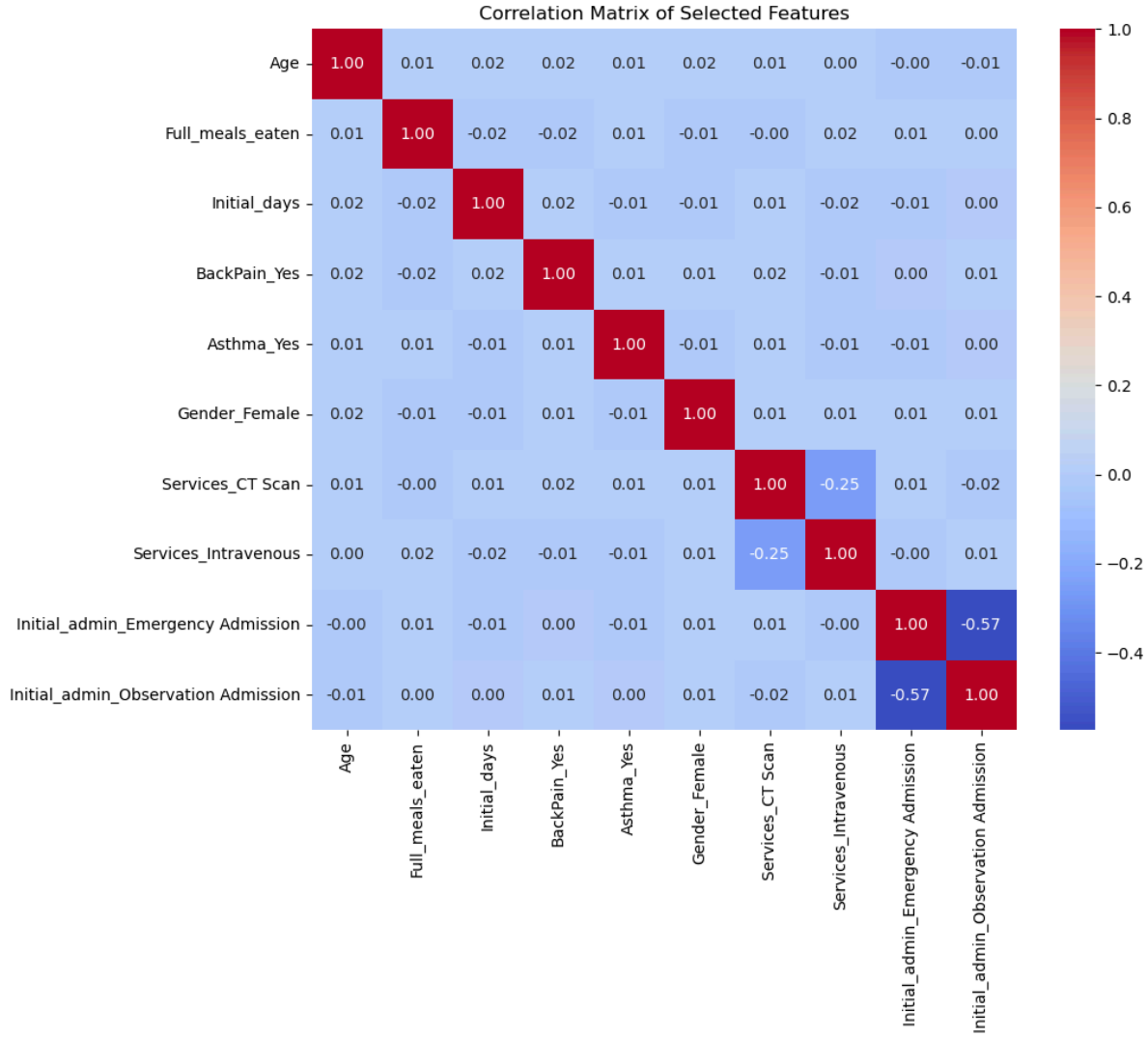
After encoding, the features can be split into X (all features) and y (target variable) then, using SelectKBest, we can select the top k features, fit them into the model, and pull out the selected top 10 features. Once selected, these were inputted into a new data frame called df_selected.

```
# Select Variables/Features
# Define X (all features) and y (target column)
y = initial_df['ReAdmis']
X = initial_df.drop('ReAdmis', axis=1)

# Select top k features
selector = SelectKBest(score_func=f_classif, k=10)
X_new = selector.fit_transform(X,y)

# Pull selected columns
selected_features = X.columns[selector.get_support(indices=True)]
print("Selected Features:", selected_features)

# Create a new dataframe with selected features
df_selected = initial_df[selected_features]
```

```
Selected Features: Index(['Age', 'Full_meals_eaten', 'Initial_days', 'BackPain_Yes', 'Asthma_Yes',
       'Gender_Female', 'Services_CT Scan', 'Services_Intravenous',
       'Initial_admin_Emergency Admission',
       'Initial_admin_Observation Admission'],
      dtype='object')
```

The next step after selecting key features is to look for correlations in the data; by finding correlations in the selected features, we can start to identify multicollinearity and the relationships between the variables. Correlation can be assessed using a correlation matrix; the .corr() function, along with a printed plot, allows us to visualize these relationships. See the correlation matrix output below.

```
# Look for correlation in the data
# Calculate the correlation matrix
correlation_matrix = df_selected.corr()

# Print out visualizations
plt.figure(figsize=(10,8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Matrix of Selected Features")
plt.show()
```



The last step needed to ensure our data is ready for a k-NN classification analysis is to apply standardization on the continuous variables. Standardizing the continuous variables in the data ensures that all features contribute equally to the k-NN distance metrics. There was only a few continuous variables left in the dataset after selecting the features. Therefore, we will standardize the column using the scaler.fit_transform function. Due to industry knowledge and researched best practice, it was determined that it is key for the analysis to standardize all numeric values, both discrete and continuous. Of the selected features, Initial_days is a continuous numeric value that should be standardized, full_meals_eaten and Age are both discrete numeric values that will also be standardized.

```
# Apply standardization to continuous colums
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_selected[['Age','Full_meals_eaten','Initial_days']])

# Check standardized data
print(df_scaled)
```

```
[[-0.02479466 -0.99338686 -0.9073098 ]
 [-0.1217056   0.99060926 -0.73459473]
 [-0.02479466 -0.0013888  -1.12829151]
 ...
 [-0.4124384   0.99060926  1.3569578 ]
 [-0.50934933  0.99060926  1.09858493]
 [ 0.79894828 -0.99338686  1.38342919]]
```

Now that the dataset has been properly assessed, cleaned, reduced, and standardized, we can proceed with developing a k-NN analysis. The steps followed above are a key part of this analysis and help to ensure that the dataset is now clean, refined, and ready to be used in a k-NN classification model. See the output of the now cleaned dataset below, as well as an exported

version under C4 below.

```
# Convert the scaled data back to a DataFrame with the same column names as in df_selected
df_scaled_df = pd.DataFrame(df_scaled, columns=['Age', 'Full_meals_eaten', 'Initial_days'])

# Replace the original columns in df_selected with the standardized columns
df_selected = df_selected.copy()
df_selected[['Age', 'Full_meals_eaten', 'Initial_days']] = df_scaled_df

# Now df_selected contains the standardized values for 'Age', 'Full_meals_eaten', and 'Initial_days'
print(df_selected)
```

```
           Age  Full_meals_eaten  Initial_days  BackPain_Yes  Asthma_Yes  \
0    -0.024795         -0.993387     -0.907310          True        True
1    -0.121706          0.990609     -0.734595         False       False
2    -0.024795         -0.001389     -1.128292         False       False
3     1.186592         -0.001389     -1.244503         False        True
4    -1.526914         -0.993387     -1.261991         False       False
...        ...               ...           ...           ...         ...
9995 -1.381548          0.990609      0.650217         False       False
9996  1.622691         -0.993387      1.300475         False        True
9997 -0.412438          0.990609      1.356958         False       False
9998 -0.509349          0.990609      1.098585          True       False
9999  0.798948         -0.993387      1.383429         False       False

      Gender_Female  Services_CT Scan  Services_Intravenous  \
0             False             False                 False
1              True             False                  True
2              True             False                 False
3             False             False                 False
4              True              True                 False
...             ...               ...                   ...
9995          False             False                  True
9996          False              True                 False
9997           True             False                  True
9998          False             False                 False
9999           True             False                 False

      Initial_admin_Emergency Admission  Initial_admin_Observation Admission
0                                  True                                False
1                                  True                                False
2                                 False                                False
3                                 False                                False
4                                 False                                False
...                                 ...                                  ...
9995                               True                                False
9996                              False                                False
9997                              False                                False
9998                               True                                False
9999                              False                                 True

[10000 rows x 10 columns]
```

C4, Prepared Data

A prepared dataset outputted in CSV format can be found within the attached ‘prepared_medical_task1.csv’ file.

D, Analysis

The below sections outline the steps taken to split the dataset into train and test data, perform calculations, and output the k-NN classification model.

D1, Train and Test Data Sets

Splitting the dataset into train and test sets is a crucial part of a k-NN classification analysis. By splitting the data into training and test sets, we can evaluate the model's performance on the test set, which acts as a proxy for unseen data. This allows us to gauge how well the model generalizes. The test set accuracy gives an unbiased estimate of how well the model will perform on new data. This is crucial in k-NN because the model relies heavily on training data points to make predictions.

The following files have been prepared and outputted, evidencing the breakout of train and test data:

- A training dataset outputted in CSV format can be found within the attached 'train_data-task1.csv' file.
- A test dataset outputted in CSV format can be found within the attached 'test_data-task1.csv' file.

D2, Analysis Technique & Calculations

The data analysis performed for this project uses the k-Nearest Neighbors (k-NN) classification technique to analyze the data, with multiple preprocessing and evaluation steps to ensure accurate model performance. The following discussion details the breakdown of each part of the analysis, including the intermediate calculations and processes.

The analysis starts by selecting the target variable, ReAdmis (i.e., hospital readmissions), and the relevant features from the dataset (df_selected). This step defines X as the feature set and y as the target variable, setting up the data for the k-NN model.

The dataset is split into training and test sets using an 80-20 split ratio. This ensures that 80% of the data is used to train the model, while 20% is reserved for testing and evaluating its performance on unseen data. Stratification by y is used to maintain the same proportion of classes in both the training and test sets. This approach helps in evaluating the model's performance across all classes more accurately.

Next, the k-NN model is initialized with k=5, meaning that each prediction will consider the five nearest neighbors to determine the class of a new data point. The model is then trained on the normalized training data, where it learns by storing the training data points (X_train and y_train). The training and test data, including features and target labels, are then combined into single data frames (train_data and test_data). These are saved as a CSV file and outputted to satisfy D1 above. This helps maintain the structure and interpretability of the data and ensures compatibility with further processing steps.

Next, using knn.predict, the model predicts the label of the data points (X_test). This prediction is based on the classes of the nearest neighbors identified in the training set. With

predict_proba, we can allow the model to output the probability of each test data point to the positive class. This is useful for calculating metrics like AUC.

Several metrics can be used to evaluate the model's performance. These calculations performed include accuracy, a confusion matrix, and a classification report (with precision, recall, and F1-score). The below outlines these calculations.

Accuracy measures the proportion of correctly classified instances out of the total instances. This is calculated as (Accuracy = Number of Correct Predictions/ Total Number of Predictions). Here, my accuracy score is 92%, which is very good for this model.

```
# Evaluate model performance
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.92
```

A confusion matrix is a table that shows the counts of true positives, true negatives, false positives, and false negatives. This helps to understand how well the model distinguishes between classes. Here, we have higher false negatives than false positives.

```
# Print confusion matrix to understand true/false positives and negatives
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
[[1124  142]
 [  27  707]]
```

The classification report provides a breakdown of several important metrics for each class: precision, recall, and F1-score. Precision is the ratio of true positives to the sum of true positives and false positives. Recall is the ratio of true positives to the sum of true positives and false negatives. The F1-score is the mean of precision and recall and balances the two metrics. The two classes here are yes and no. For example, in the 'no' category – of all instances predicted as no, 98% were correctly classified. Of all actual instances, 89% were correctly identified. .93 is the harmonic mean of precision and recall, providing a balanced measure. The total number of no samples in the test set was 1266.

```
# Print classification report for precision, recall, and F1-score
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

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

          No       0.98      0.89      0.93      1266
         Yes       0.83      0.96      0.89       734

    accuracy                           0.92      2000
   macro avg       0.90      0.93      0.91      2000
weighted avg       0.92      0.92      0.92      2000
```

The AUC (Area Under the Curve) score is calculated using roc_auc_score. AUC is a performance metric that evaluates the model's ability to distinguish between classes. A higher AUC score indicated better performance, with a maximum score of 1.

```
# Calculate AUC score
auc_score = roc_auc_score(y_test, y_proba)
print(f"AUC Score: {auc_score:.2f}")
```

```
AUC Score: 0.97
```

The ROC (Receiver Operating Characteristic) curve is plotted to visualize the model's performance across various classification thresholds. This includes calculating the true positive rate (TPR) and false positive rate (FPR) at different threshold levels using the roc_curve function.

This k-NN classification analysis technique ensures a balanced approach by normalizing data, using train-test splitting for unbiased evaluation, and leveraging AUC and ROC curve metrics for an in-depth performance evaluation. Each step, from normalization to probability calculation, helps build a reliable model that generalizes well to new data. See screenshots of the code used to perform this k-NN model under D3 below, as well as an additional discussion of the AUC can be seen under section E below.

D3, Analysis Code

The code printed below outlines the steps used to perform the classification analysis from D2 above. These steps included selecting variables, splitting the data, converting data arrays to data frames, k-NN model initialization and training, combining train and test data, model prediction and probability calculation, intermediate calculations, and lastly, evaluating the model performance with AUC and a ROC curve.

```
# Select Variables/Features
# Define X (all features) and y (target column)
y = initial_df['ReAdmis']
X = df_selected

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratif

# Initialize the k-NN classifier with k=5
knn = KNeighborsClassifier(n_neighbors=5)

# Train the k-NN model
knn.fit(X_train, y_train)

# Combine X_train and y_train into a single DataFrame
train_data = pd.concat([X_train, y_train], axis=1)
test_data = pd.concat([X_test, y_test], axis=1)

# Save to CSV files
train_data.to_csv('train_data-task1.csv', index=False)
test_data.to_csv('test_data-task1.csv', index=False)

# Predict the labels on the test set
y_pred = knn.predict(X_test)

# Predict probabilities for the positive class
y_proba = knn.predict_proba(X_test)[:, 1]
```

```
# Evaluate model performance
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.92
```

```
# Print confusion matrix to understand true/false positives and negatives
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
[[1124  142]
 [  27  707]]
```

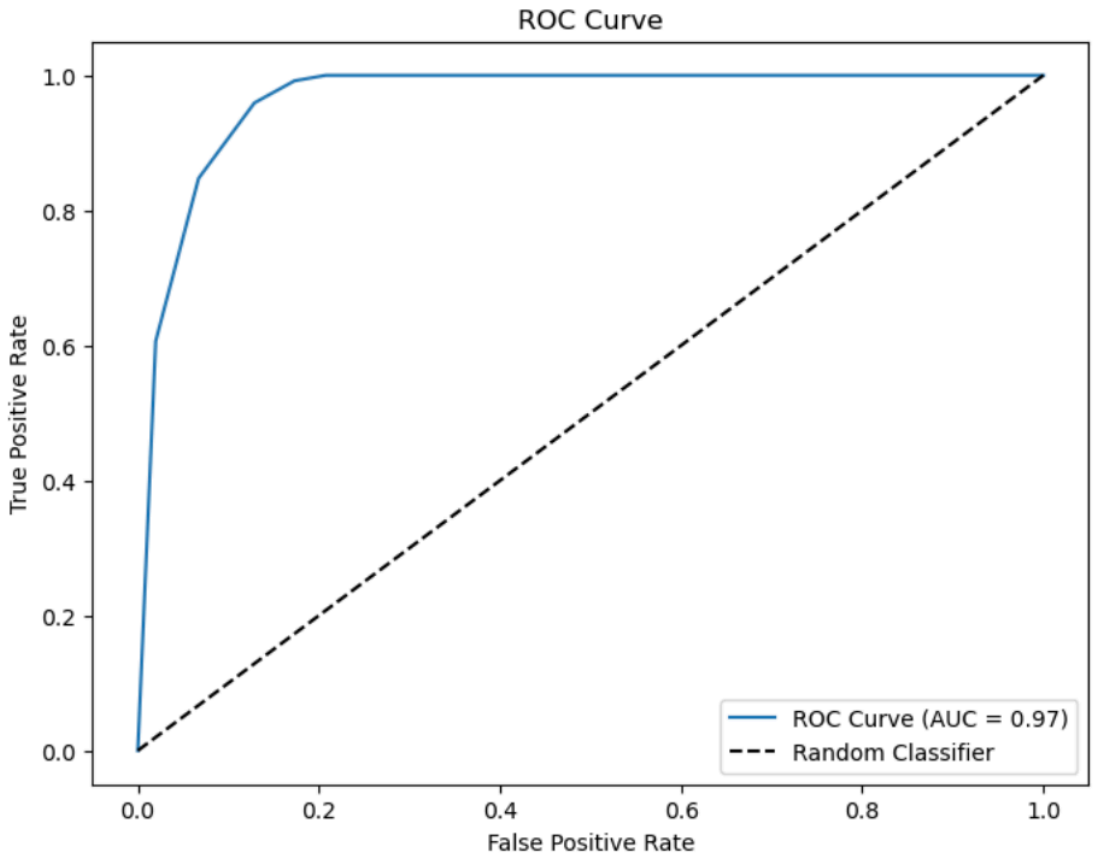
```
# Print classification report for precision, recall, and F1-score
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

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

          No       0.98      0.89      0.93      1266
         Yes       0.83      0.96      0.89       734

    accuracy                           0.92      2000
   macro avg       0.90      0.93      0.91      2000
weighted avg       0.92      0.92      0.92      2000
```

```
# Calculate AUC score
auc_score = roc_auc_score(y_test, y_proba)
print(f"AUC Score: {auc_score:.2f}")
```

```
AUC Score: 0.97
```

```
# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_proba, pos_label="Yes")
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {auc_score:.2f})")
plt.plot([0, 1], [0, 1], 'k--', label="Random Classifier")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend(loc="best")
plt.show()
```



E, Data Summary and Implications

The below sections provide an overview of the k-NN classification analysis performed. The sections outline the data summary, key results, implications, and limitations and recommend a course of action.

E1, Accuracy and AUC

Accuracy and AUC (Area Under the Curve) are two metrics used to evaluate the performance of the k-NN classification model.

Accuracy is the ratio of correct predictions (both positive and negative) to the total number of predictions using the calculation (Accuracy = Number of Correct Predictions/ Total Number of

Predictions). In this case, Accuracy is 92%, meaning that 92% of the predictions made by the model on the test set were correct.

```
# Evaluate model performance
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.92
```

The AUC score represents the model's ability to distinguish between classes. It measures how well the model ranks positive instances higher than negative ones. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds. The AUC score is the area under this curve, providing a single value between 0 and 1. A perfect model has an AUC of 1, while a model with no discrimination ability has an AUC of 0.5. Here, the model achieved an AUC score of 0.97 (or 97%), which indicates excellent performance in distinguishing between the classes.

```
# Calculate AUC score
auc_score = roc_auc_score(y_test, y_proba)
print(f"AUC Score: {auc_score:.2f}")
```

```
AUC Score: 0.97
```

In this case, a high AUC score (0.97) suggests that the model is very effective in distinguishing between the "Yes" and "No" classes, even though the accuracy (0.90) might seem slightly lower. Both metrics together give a comprehensive view of model performance, with AUC being particularly valuable for understanding the model's classification quality.

E2, Results and Implications

The goal of this analysis was to determine if hospital readmissions can be predicted based on patient characteristics and medical history. The k-Nearest Neighbors (k-NN) classification model was chosen due to its simplicity, interpretability, and suitability for distance-based prediction tasks. The AUC score of 97% indicates that the model is highly effective at distinguishing between patients who will and will not be readmitted. AUC provides a strong measure of model performance, especially for imbalanced classes. The model correctly predicted readmissions and non-readmissions for 92% of cases in the test set, demonstrating overall reliability.

Precision of 0.98 and recall of 0.89. This indicates that the model is quite accurate in identifying patients who won't be readmitted but occasionally misses some non-readmissions. With a precision score of 0.83 and recall of 0.96, this indicates high sensitivity, meaning the model captures most patients at risk for readmission, although some non-readmitted cases are incorrectly flagged as readmissions.

The results show that the k-NN model is generally effective in predicting readmissions, particularly with a high recall for the "Yes" class, meaning it captures most actual readmissions. However, the model tends to misclassify some non-readmitted cases as readmissions (reflected in lower precision for the "Yes" class).

Some implications of this analysis include:

- The high recall for predicting readmissions suggests that this model could be used as a screening tool to identify patients who may benefit from additional post-discharge support. For example, flagged patients could receive follow-up calls, medication reminders, or additional health checks, which could help prevent actual readmissions.
- Reducing readmissions is often financially beneficial for hospitals, as many healthcare systems impose penalties for high readmission rates. By accurately identifying high-risk patients, the hospital could implement preventative measures that reduce these penalties and optimize resource usage.
- The model's tendency to generate some false positives (predicting readmissions that don't occur) may lead to unnecessary interventions for some patients, potentially increasing costs and resource use. It may be beneficial to weigh the operational costs of these interventions against the costs associated with actual readmissions.

This k-NN analysis highlights the potential for predictive modeling in healthcare, specifically in managing hospital readmissions. While the model offers practical insights and can support targeted interventions, careful consideration of false positives and ongoing model evaluation will be essential to maximizing its clinical and operational value.

E3, Analysis Limitations

One limitation of a k-Nearest Neighbors (k-NN) classification model is its sensitivity to the course of dimensionality. As the number of features (dimensions) increases, the distance between data points in high-dimensional space becomes less meaningful. In high dimensions, all points tend to become equidistant from each other, which makes it harder for k-NN to find truly "nearest" neighbors. This reduces the effectiveness of k-NN because the algorithm relies heavily on meaningful distance calculations to identify the nearest neighbors. In high-dimensional spaces, k-NN struggles to distinguish between close and distant neighbors, which can significantly reduce classification accuracy.

The curse of dimensionality makes k-NN less effective and more computationally intensive as the number of features grows, limiting its application to lower-dimensional datasets or those that have been carefully preprocessed to reduce dimensionality.

E4, Course of Action

Due to the high model performance and metrics, these metrics indicate the model is effective but could benefit from additional tuning or balancing, especially if the cost of false positives or false negatives is significant for hospital operations. As the k-NN algorithm is sensitive to high-dimensional data, one course of action may be to consider revisiting feature selection and exploring other dimensionality reduction techniques (e.g., PCA) or further narrow down the features using domain knowledge or additional statistical tests.

Given the model's high recall for predicting readmissions, it could serve as an initial screening tool to flag patients at higher risk for readmission. These patients could then receive targeted interventions, such as follow-up calls or additional support. It could be beneficial to use the model's predictions alongside clinician input, especially in cases where the model predicts readmission with low confidence. This will allow for a more balanced approach, minimizing unnecessary interventions. Additionally, Patient characteristics and healthcare practices evolve,

so it's essential to regularly retrain and update the model with new data to maintain accuracy and relevance.

The k-NN model provides valuable insights and a reasonable level of accuracy, making it a useful tool for predicting hospital readmissions. However, fine-tuning the model, potentially exploring other algorithms, and using it as part of a broader decision-making framework involving clinical input will enhance its utility and effectiveness. This approach can support hospital efforts to reduce readmissions, manage resources better, and improve patient outcomes.

F, Panopto Video Recording Executions

The video recording for this assignment includes a vocalized demonstration of all the code, the code being executed, and the results of the code being represented inside this report.
The video recording for this project can be found inside the Panopto drop box titled "D209-Task1-Kline"

Panopto video link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=44683f3c-5f3f-4d1a-813c-b21a01732359

G, Web Sources

No sources or segments of third-party code were used to acquire data or to support the report.

H, Acknowledge the Sources

I acknowledge that no sources or segments of third-party sources were stated or copied from the web into this report.