# Data Mining
# Task 1- Classification Analysis

D209 WGU

TRESA (TESSIE) AUSTIN

**Part I: Research Question**

**A.** Define the research question and describe the purpose of this data analysis:

1. Readmission rates in hospitals are a significant concern as the impact to patient outcomes, costs, and overall quality of care are impacted. A critical objective for any hospital is to reduce readmission rates since it could be tied to regulatory and reimbursement policies. Based on that it's important to research the following question:

   **What factors contribute to a patient being readmitted to the hospital within 30 days of their initial release, and how accurately can the K-nearest neighbor (kNN) algorithm predict hospital readmissions based on these factors?**

2. Define the goals of the data analysis.
   Building on the analysis done around logistic regression previously, there were multiple factors that were statistically and practically significant. Taking that a step further by determining a more effective model using kNN classification is the goal here to see if we can provide better context and more useful information for hospitals to manage patient care.

**Part II: Method Justification**

B. Explain the reasons for your chosen classification method from part A1 by doing the following:

1. The classification method chosen analyzes the selected data set in the following ways. Using K-Nearest Neighbors (kNN) as the method here stems from the ease and simplicity of how the method works. kNN doesn't require any assumptions about underlying data distribution. The premise of the K-Nearest Neighbor algorithm is that similar things exist in proximity. Therefore using kNN you can scout over a certain radius (number) of neighbors to find labels for a point of data that has no label. According to Srivastava (2024), the algorithm assigns the most common class label among the 'K' neighbors as the predicted label for the input data point. That value is calculated from the average or weighted average of those scouted values; hence we get the predicated value for the data point in question.

   The expected outcomes would be a comprehensive view of the model's performance with strengths and areas for making improvements. Using kNN will serve as a guide to further refinements of the model.

2. According to Harrison (2018) the major assumption of kNN is that "birds of a feather flock together" or in scientific terms, similar things exist near each other or in proximity. It relies on this premise when used for classification and assigns a class label based on majority, meaning that it uses the label that occurs most often around the data point in question.

3. Below are the packages and libraries being used in Python that supports the analysis:

   - Pandas
     - Provides data structures such as DataFrame which provides ease in data loading. Pandas also provides essential processing steps around evaluating data for missing values, encoding categorical variables, ad splitting dataset (GeeksforGeeks, 2023).
   - Numpy
     - Provides support for array operations. Specifically for kNN, Numpy calculates distances between points (Kothari, 2020).
   - Matplotlib.pyplot

    - Matplotlib is effectively used to visualize data, model results and performance metrics to help understand visualizations (Tutorialspoint, n.d.).
- Seabon
    - Seaborn and kNN are used to create informative and visually appealing plots. While Seaborn itself doesn't directly support plots of decision boundaries, it can be used with Matplotlib to enhance the appeal of those plots (Kumar, 2024).
- Sklearn.model_selection import train_test_split
    - The key role of "train_test_split" is to create a division of data into two subsets which is crucial for evaluating the performance of machine learning models (Stojiljković, 2020**)**.
- Sklearn.neighbors import KNeighborsClassifier
    - For kNN, this is one of the most important packages to use. Its main function is to train a kNN model on the training dataset, store the training data and identify the K-nearest neighbors to a new data point (Ratnakar, 2022)
- sklearn.model_selection import GridSearchCV
    - The GridSearchCV implements a fit and score method that applies to the parameters of the estimator and then is optimized by a cross-validated grid search.
- Sklearn.preprocessing import StandardScaler
    - The StandardScaler is a tool used for feature scaling which ensures that each feature contributes equally to distance calculations (Javatpoint, n.d.).
- Sklearn.metrics import confusion_matrix, accuracy_score, classification_report, roc_curve, roc_auc_score
    - According the scikit_learn.org (2024) the following were found:
        - The confusion_matrix evaluates the accuracy of the classification.
        - The accuracy_score is a multilabel classification that computes subset accuracy.
        - The classification_report builds a text report showing the main classification metrics.
        - The receiver operating characteristic curve or roc_curve is a graphical plot illustrating a binary classifier system and its threshold.
        - The Area Under the Receiver Operating Characteristic Curve score or roc_auc_score is a prediction score that combines tow averages strategies to estimate that a sample belongs to a particular class.

## Part III: Data Preparation

C. Perform data preparation for the chosen data set by doing the following:

1. One data preprocessing goal relevant to kNN is feature scaling. Since kNN is a distance-based algorithm, features with larger ranges will be disproportionately influenced by the distance calculations. Using feature scaling ensures that all features contribute equally (Filho, 2023).
2. The initial data set variables that will be used to perform the analysis kNN

| Target variable | Type |
|---|---|
| ReAdmis | Categorical |
| | |
| Variable | Type |
| Initial_days | Numerical |
| Age | Numerical |
| Doc_visits | Numerical |
| vitD_supp | Numerical |

| | |
|---|---|
| Allergic_rhinitis | Categorical |
| Anxiety | Categorical |
| Arthritis | Categorical |
| Asthma | Categorical |
| BackPain | Categorical |
| Complication_risk | Categorical |
| Diabetes | Categorical |
| Gender | Categorical |
| HighBlood | Categorical |
| Hyperlipidemia | Categorical |
| Initial_admin | Categorical |
| Overweight | Categorical |
| Reflux_esophagitis | Categorical |
| Services | Categorical |
| Stroke | Categorical |

3. The steps used to prepare the data for the analysis with code
   1. Data will be loaded

```
# Set the correct file path for the medical data
medical_file_path = "medical_clean.csv"
# Read the medical data file with keep_default_na
df = pd.read_csv(medical_file_path, keep_default_na=False, index_col=0)
```

   2. A check of duplicates will be performed

```
# Code to check for duplicates
has_duplicates = df.duplicated().any()
print("Duplicates present:", has_duplicates)
```

```
Duplicates present: False
```

   3. A check for missing values will be performed.

```
# Check for missing data
missing_data = df.isnull().sum()

# Display the missing data counts
print("Missing data counts:")
print(missing_data)
```

```
Missing data counts:
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
```

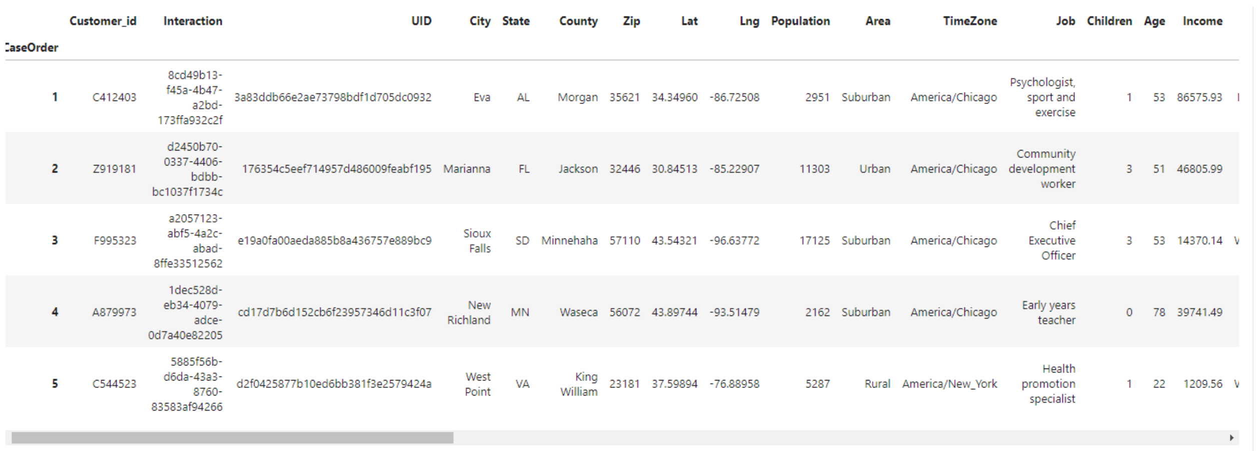
4. Then after those checks, a look at data types and an inspection of the DataFrame needs to be completed.

**# Display data types**

**df.info()**

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

```
# Visually inspect df
pd.set_option("display.max_columns", None)
df.head(5)
```



| | Customer_id | Interaction | UID | City | State | County | Zip | Lat | Lng | Population | Area | TimeZone | Job | Children | Age | Income |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| aseOrder | | | | | | | | | | | | | | | | |
| 1 | C412403 | 8cd49b13-f45a-4b47-a2bd-173ffa932c2f | 3a83ddb66e2ae73798bdf1d705dc0932 | Eva | AL | Morgan | 35621 | 34.34960 | -86.72508 | 2951 | Suburban | America/Chicago | Psychologist, sport and exercise | 1 | 53 | 86575.93 |
| 2 | Z919181 | d2450b70-0337-4406-bdbb-bc1037f1734c | 176354c5eef714957d486009feabf195 | Marianna | FL | Jackson | 32446 | 30.84513 | -85.22907 | 11303 | Urban | America/Chicago | Community development worker | 3 | 51 | 46805.99 |
| 3 | F995323 | a2057123-abf5-4a2c-abad-8ffe33512562 | e19a0fa00aeda885b8a436757e889bc9 | Sioux Falls | SD | Minnehaha | 57110 | 43.54321 | -96.63772 | 17125 | Suburban | America/Chicago | Chief Executive Officer | 3 | 53 | 14370.14 |
| 4 | A879973 | 1dec528d-eb34-4079-adce-0d7a40e82205 | cd17d7b6d152cb6f23957346d11c3f07 | New Richland | MN | Waseca | 56072 | 43.89744 | -93.51479 | 2162 | Suburban | America/Chicago | Early years teacher | 0 | 78 | 39741.49 |
| 5 | C544523 | 5885f56b-d6da-43a3-8760-83583af94266 | d2f0425877b10ed6bb381f3e2579424a | West Point | VA | King William | 23181 | 37.59894 | -76.88958 | 5287 | Rural | America/New_York | Health promotion specialist | 1 | 22 | 1209.56 |

5. Gather summary statistics on the variables to be used in the dataframe are as follows:

```
# Summary statistics for the dependent variable
ReAdmis_summary = df["ReAdmis"].describe()
print("Summary Statistics for ReAdmis:")
print(ReAdmis_summary)
```

```
Summary Statistics for ReAdmis:
count     10000
unique        2
top          No
freq       6331
Name: ReAdmis, dtype: object
```

```
# Summary statistics for Initial_days
Initial_days_summary = df["Initial_days"].describe()
print("Summary Statistics for Initial_days:")
print(Initial_days_summary)
```

```
Summary Statistics for Initial_days:
count    10000.000000
mean        34.455299
std         26.309341
min          1.001981
```

```
25%         7.896215
50%        35.836244
75%        61.161020
max        71.981490
Name: Initial_days, dtype: float64
```

**# Summary statistics for Age**
**Age_summary = df["Age"].describe()**
**print("Summary Statistics for Age:")**
**print(Age_summary)**

```
Summary Statistics for Age:
count    10000.000000
mean        53.511700
std         20.638538
min         18.000000
25%         36.000000
50%         53.000000
75%         71.000000
max         89.000000
Name: Age, dtype: float64
```

**# Summary statistics for Doc_visits**
**Doc_visits_summary = df["Doc_visits"].describe()**
**print("Summary Statistics for Doc_visits:")**
**print(Doc_visits_summary)**

```
Summary Statistics for Doc_visits:
count    10000.000000
mean         5.012200
std          1.045734
min          1.000000
25%          4.000000
50%          5.000000
75%          6.000000
max          9.000000
Name: Doc_visits, dtype: float64
```

**# Summary statistics for vitD_supp**
**vitD_supp_summary = df["vitD_supp"].describe()**
**print("Summary Statistics for vitD_supp:")**
**print(vitD_supp_summary)**

```
Summary Statistics for vitD_supp:
count    10000.000000
mean         0.398900
std          0.628505
min          0.000000
25%          0.000000
50%          0.000000
```

```
75%          1.000000
max          5.000000
Name: vitD_supp, dtype: float64
```

```
# Summary statistics for Allergic_rhinitis
Allergic_rhinitis_summary = df["Allergic_rhinitis"].describe()
print("Summary Statistics for Allergic_rhinitis:")
print(Allergic_rhinitis_summary)
```

```
Summary Statistics for Allergic_rhinitis:
count     10000
unique        2
top          No
freq       6059
Name: Allergic_rhinitis, dtype: object
```

```
# Summary statistics for Anxiety
Anxiety_summary = df["Anxiety"].describe()
print("Summary Statistics for Anxiety:")
print(Anxiety_summary)
```

```
Summary Statistics for Anxiety:
count     10000
unique        2
top          No
freq       6785
Name: Anxiety, dtype: object
```

```
# Summary statistics for Arthritis
Arthritis_summary = df["Arthritis"].describe()
print("Summary Statistics for Arthritis:")
print(Arthritis_summary)
```

```
Summary Statistics for Arthritis:
count     10000
unique        2
top          No
freq       6426
Name: Arthritis, dtype: object
```

```
# Summary statistics for Asthma
Asthma_summary = df["Asthma"].describe()
print("Summary Statistics for Asthma:")
print(Asthma_summary)
```

```
Summary Statistics for Asthma:
count     10000
unique        2
top          No
freq       7107
Name: Asthma, dtype: object
```

```
# Summary statistics for BackPain
BackPain_summary = df["BackPain"].describe()
print("Summary Statistics for BackPain:")
print(BackPain_summary)
```

```
Summary Statistics for BackPain:
count     10000
unique        2
top          No
freq       5886
Name: BackPain, dtype: object
```

```
# Summary statistics for Complication_risk
Complication_risk_summary = df["Complication_risk"].describe()
print("Summary Statistics for Complication_risk:")
print(Complication_risk_summary)
```

```
Summary Statistics for Complication_risk:
count     10000
unique        3
top      Medium
freq       4517
Name: Complication_risk, dtype: object
```

```
# Summary statistics for Diabetes
Diabetes_summary = df["Diabetes"].describe()
print("Summary Statistics for Diabetes:")
print(Diabetes_summary)
```

```
Summary Statistics for Diabetes:
count     10000
unique        2
top          No
freq       7262
Name: Diabetes, dtype: object
```

```
# Summary statistics for Gender
Gender_summary = df["Gender"].describe()
print("Summary Statistics for Gender:")
print(Gender_summary)
```

```
Summary Statistics for Gender:
count      10000
unique         3
top       Female
freq        5018
Name: Gender, dtype: object
```

```
# Summary statistics for HighBlood
```

```
HighBlood_summary = df["HighBlood"].describe()
print("Summary Statistics for HighBlood:")
print(HighBlood_summary)
```

```
Summary Statistics for HighBlood:
count     10000
unique        2
top          No
freq       5910
Name: HighBlood, dtype: object
```

```
# Summary statistics for Hyperlipidemia
Hyperlipidemia_summary = df["Hyperlipidemia"].describe()
print("Summary Statistics for Hyperlipidemia:")
print(Hyperlipidemia_summary)
```

```
Summary Statistics for Hyperlipidemia:
count     10000
unique        2
top          No
freq       6628
Name: Hyperlipidemia, dtype: object
```

```
# Summary statistics for Initial_admin
Initial_admin_summary = df["Initial_admin"].describe()
print("Summary Statistics for Initial_admin:")
print(Initial_admin_summary)
```

```
Summary Statistics for Initial_admin:
count                   10000
unique                      3
top       Emergency Admission
freq                     5060
Name: Initial_admin, dtype: object
```

```
# Summary statistics for Overweight
Overweight_summary = df["Overweight"].describe()
print("Summary Statistics for Overweight:")
print(Overweight_summary)
```

```
Summary Statistics for Overweight:
count     10000
unique        2
top         Yes
freq       7094
Name: Overweight, dtype: object
```

```
# Summary statistics for Reflux_esophagitis
Reflux_esophagitis_summary = df["Reflux_esophagitis"].describe()
```

```
print("Summary Statistics for Reflux_esophagitis:")
print(Reflux_esophagitis_summary)
```

```
Summary Statistics for Reflux_esophagitis:
count     10000
unique        2
top          No
freq       5865
Name: Reflux_esophagitis, dtype: object
```

```
# Summary statistics for Services
Services_summary = df["Services"].describe()
print("Summary Statistics for Services:")
print(Services_summary)
```

```
Summary Statistics for Services:
count          10000
unique             4
top       Blood Work
freq            5265
Name: Services, dtype: object
```

```
# Summary statistics for Stroke
Stroke_summary = df["Stroke"].describe()
print("Summary Statistics for Stroke:")
print(Stroke_summary)
```

```
Summary Statistics for Stroke:
count     10000
unique        2
top          No
freq       8007
Name: Stroke, dtype: object
```

6. After that, modifying the variables that will be used in the model will be updated/changed to correct types.

For kNN analysis, we keep all dummy variables, we don't drop the first one as we did in previous analysis since we don't have to worry about multicollinearity(Shmueli, 2015). Some variables had to be changed to fit the analysis needed. According to The Quantitative Development Systems Initiative (n.d.) k-NN involves the calculation of distances between datapoints, we must use numeric variables only.

```
#Data Transformation Needed
# Update 'Initial_days' and 'vitD_supp' to int
df["Initial_days"] = df["Initial_days"].astype(int)
df["vitD_supp"] = df["vitD_supp"].astype(int)
```

```
# Update 'Gender' to category
df["Gender"] = df["Gender"].astype("category")

# Update columns to boolean using mapping
bool_mapping = {"Yes": 1, "No": 0}
columns_to_convert = ["HighBlood", "Stroke", "Overweight", "Arthritis", "Diabetes",
"Hyperlipidemia", "BackPain", "Anxiety", "Allergic_rhinitis", "Reflux_esophagitis", "Asthma",
"ReAdmis"]
for col in columns_to_convert:
   df[col] = df[col].map(bool_mapping)

# Identify numerical features for scaling
numerical_features = ["Initial_days", "Age", "Doc_visits", "vitD_supp"]

# Standardize the numerical features
scaler = StandardScaler()
df[numerical_features] = scaler.fit_transform(df[numerical_features])

# Generate columns of dummy values for categorical variables
gender_df = pd.get_dummies(data=df["Gender"], drop_first=False)
int_admit_df = pd.get_dummies(data=df["Initial_admin"], drop_first=False)
comp_risk_df = pd.get_dummies(data=df["Complication_risk"], drop_first=False)
services_df = pd.get_dummies(data=df["Services"], drop_first=False)

# Create new df with model variables
kNN_df = df[["Initial_days", "Age", "Doc_visits", "vitD_supp", "HighBlood", "Stroke",
"Overweight", "Arthritis", "Diabetes", "Hyperlipidemia", "BackPain", "Anxiety", "Allergic_rhinitis",
"Reflux_esophagitis", "Asthma", "ReAdmis"]].copy()

# Add dummy variables to kNN_df
kNN_df = pd.concat([kNN_df, gender_df, int_admit_df, comp_risk_df, services_df], axis=1)

# Ensure all dummies are 1/0
dummy_columns = gender_df.columns.tolist() + int_admit_df.columns.tolist() +
comp_risk_df.columns.tolist() + services_df.columns.tolist()
for col in dummy_columns:
   kNN_df[col] = kNN_df[col].astype(int)
```

4. Provide a copy of the cleaned data set.

```
# Save kNN_df to a CSV file
kNN_df.to_csv("kNN_df.csv", index=False)
print("kNN_df has been saved to kNN_df.csv'.")
```

```
kNN_df has been saved to kNN_df.csv'
```

*A copy of these files are attached to PA submission*

**Part IV: Analysis**

D. Perform the data analysis and report on the results by doing the following:

1. Taking the data and splitting it into training and test data sets was done using sklearn.model_selection import train_test_split. Stojiljković (2020) stated that by splitting datasets into subsets, it minimizes the potential for bias in your evaluation and validation process. This approach ensures that the model is tested on a subset of data that it has not seen during training, providing a more accurate measure of its performance on unseen data (Stojiljković, 2020). Additionally, other sources also emphasize that proper data splitting is crucial for building robust and generalizable machine learning models (Brownlee, 2020).

```
Set the correct file path for kNN df
kNN_df_path = "kNN_df.csv"
df = pd.read_csv(kNN_df_path, keep_default_na=False)

# Splitting the data into features and target
X = df.drop('ReAdmis', axis=1)
y = df['ReAdmis']

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Combine the training features and target into a single DataFrame
train_df = pd.concat([X_train, y_train], axis=1)
test_df = pd.concat([X_test, y_test], axis=1)

# Save the train and test DataFrames to CSV files
train_df.to_csv("train_df.csv", index=False)
test_df.to_csv("test_df.csv", index=False)

print("train_df and test_df have been saved to test_df.csv and train_df.csv .")
```

```
train_df and test_df have been saved to test_df.csv and train_df.csv
```

*A copy of these files are attached to PA submission*

2. The analysis technique used to appropriately analyze the data includes finding the appropriate value of 'k' for kNN in this instance is GridSearchCV (Great Learning Team, 2024). GridSearchCV allows hyperparameter tuning to be used to find the optimal values in a model. Using GridSeachCV cuts down on manually having to try all values.

   Since kNN is a distance-based algorithm, standardization of the features must be done using 'StandardScaler' (scikit-learn developers, n.d.). This allows for the shape of the distribution to have small standard deviations of features.

   By passing predefined values for hyperparameters via a defined dictionary, GridSearchCV can try all the combinations of values passed using a cross validation method so we get accuracy/loss for each combination and then can choose the one with the best fit.

3. Provide the code used to perform the classification analysis from part D2.

```
# Standardize the features and perform GridSearchCV
kNN_df_path = "kNN_df.csv"
kNN_df = pd.read_csv(kNN_df_path, keep_default_na=False)

# Separate features and target
```

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

# Identify numerical and categorical columns
numerical_features = ["Initial_days", "Age", "Doc_visits", "vitD_supp"]
categorical_features = [col for col in X.columns if col not in numerical_features]
print("Columns in X before standardization:", X.columns)

# Standardize the numerical features
scaler = StandardScaler()
X[numerical_features] = scaler.fit_transform(X[numerical_features])

# Combine the standardized numerical features with the categorical features
X = np.hstack([X[numerical_features], X[categorical_features]])

# Perform GridSearchCV to find the optimal number of neighbors
param_grid = {'n_neighbors': range(1, 31)}
kNN = KNeighborsClassifier()
grid_search = GridSearchCV(kNN, param_grid, cv=10, scoring='accuracy')
grid_search.fit(X, y)

# Get the best number of neighbors
best_k = grid_search.best_params_['n_neighbors']
print(f'The optimal number of neighbors is {best_k}')

# Fit kNN classifier with the optimal number of neighbors
kNN = KNeighborsClassifier(n_neighbors=best_k)
kNN.fit(X, y)
best_accuracy = grid_search.best_score_
print(f'Best cross-validated accuracy with k={best_k}: {best_accuracy}')
```

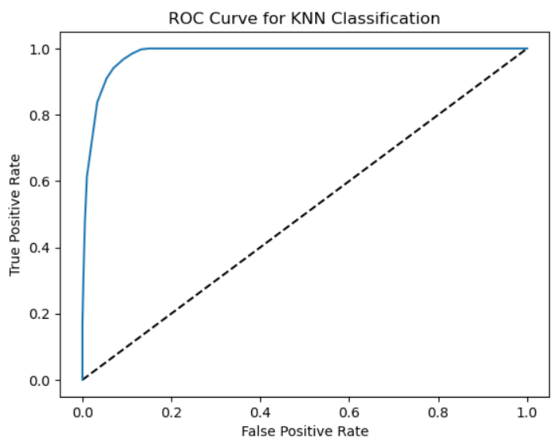
```
Columns in X before standardization: Index(['Initial_days', 'Age', 'Doc_visits', 'vitD_supp',
'HighBlood', 'Stroke', 'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma',
'Female', 'Male', 'Nonbinary', 'Elective Admission',
'Emergency Admission', 'Observation Admission', 'High', 'Low', 'Medium',
'Blood Work', 'CT Scan', 'Intravenous', 'MRI'],
dtype='object')

The optimal number of neighbors is 22
Best cross-validated accuracy with k= 22: 0.8933
```

```
# Perform kNN using the value of k=22
X_train = np.ascontiguousarray(X_train)
X_test = np.ascontiguousarray(X_test)

kNN = KNeighborsClassifier(n_neighbors=2)
kNN.fit(X_train, y_train)

# Generate y_pred array for model's confusion matrix
y_pred = kNN.predict(X_test)
final_matrix = confusion_matrix(y_test, y_pred)
```

```
# Print confusion matrix and accuracy score of model
print("The confusion matrix for this kNN model:")
print("Predicted No ReAdmis | Predicted ReAdmis")
print(f"              {final_matrix[0]} Actual No ReAdmis")
print(f"              {final_matrix[1]} Actual ReAdmis")
print(f"The training accuracy of this kNN classification is {kNN.score(X_train, y_train):.5f}.")
print(f"The testing accuracy of this kNN classification model is {kNN.score(X_test, y_test):.5f}.")
```

```
The confusion matrix for this kNN model:
              Predicted No ReAdmis | Predicted ReAdmis
              [1677  257] Actual No ReAdmis
              [  10 1056] Actual ReAdmis
The training accuracy of this kNN classification is 0.92214
The testing accuracy of this kNN classification model is 0.91100
```

```
# Generate AUC score and print
y_pred_prob = kNN.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for kNN Classification')
plt.show()
print(f"The Area Under the Curve (AUC) score is: {roc_auc_score(y_test, y_pred_prob)}\n")
print(classification_report(y_test, y_pred))
```



```
The Area Under the Curve (AUC) score is: 0.9823586094028421

              precision    recall    f1-score    support

           0     0.99        0.87      0.93        1934
           1     0.80        0.99      0.89        1066

    accuracy                           0.91        3000
   macro avg    0.90        0.93      0.91        3000
weighted avg    0.93        0.91      0.91        3000
```

*A copy of the full code can be found in D209_Austin_T_Task1_04.ipynb as well as the code with findings in D209_Austin_T_Task1_04_All.ipynb*

**Part V: Data Summary and Implications**

E. Summarize your data analysis by doing the following:

1. The classification model's accuracy and Area Under the Curve (AUC) are key metrics used to evaluate its performance:

- Accuracy:
    - Training Accuracy: 0.92214, indicating the model correctly predicted 92.21% of the instances in the training set.
    - Testing Accuracy: 0.91100, indicating the model correctly predicted 91.10% of the instances in the testing set.
    - These high accuracy values suggest that the model fits the training data well and generalizes effectively to unseen data.
- AUC:
    - The AUC score of 0.978 is a summary metric of the Receiver Operating Characteristic (ROC) curve. It reflects the model's ability to distinguish between positive and negative classes.
    - According to Çorbacıoğlu and Aksel (2023), AUC values range from 0 to 1, with values closer to 0.5 indicating performance no better than guessing. The high AUC score here indicates the model is excellent at distinguishing between readmission and no readmission.

2. The results of the classification analysis indicate that the model is highly effective in predicting patient readmissions. Key findings include:

    - Confusion Matrix and Classification Report:
        - Precision for No Readmission: 0.99
            - Indicating extremely high accuracy in predicting no readmission cases.
        - Recall for No Readmission: 0.87
            - Indicating some missed actual no readmission cases.
        - Precision for Readmission: 0.80
            - Indicating some false positives in predicting readmission.
        - Recall for Readmission: 0.99
            - Indicating all actual readmission cases are correctly identified.

    These results suggest that while the model performs well, particularly in identifying readmissions, it could benefit from further tuning to reduce false positives.

3. One limitation of the data analysis is the presence of false positives in the prediction of readmission. The model shows a precision of 0.80 for readmission, meaning 20% of the patients predicted to be readmitted were not actually readmitted. This could lead to unnecessary interventions or follow-ups, stressing the need for further refinement of the model to improve precision.

4. Based on the results and implications of the classification analysis, the following course of action is recommended for the hospital:

   - Enhance Data Collection: Collect more comprehensive and detailed data, including additional features that could impact readmission, such as socioeconomic status, patient compliance with post-discharge instructions, and support systems at home.
   - Refine Data Features: Improve the granularity of existing data features, such as "Complication_risk," by stratifying it into more detailed levels to enhance the model's predictive power.
   - Regular Model Updates: Continuously update and monitor the model to adapt to new data and changing patterns in patient readmissions, ensuring the model remains accurate and relevant.

   By following these recommendations, the hospital can improve the predictive accuracy of patient readmissions, leading to better resource allocation and patient care outcomes.

**Part VI: Demonstration**

F. Panopto video link

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=40d29f8a-3d4f-418c-bcd5-b19b01706e6d

G. Web Sources

Great Learning Team. (2024, April 30). *GridSearchCV: Everything you need to know*. Great Learning. Retrieved June 13, 2024, from https://www.mygreatlearning.com/blog/gridsearchcv/
Javatpoint. (n.d.). *StandardScaler in Sklearn*. Retrieved June 12, 2024, from https://www.javatpoint.com/standardscaler-in-sklearn

Ratnakar, V. (2022, January 29). *kNN classification using scikit-learn*. Medium. Retrieved June 12, 2024, from https://medium.com/@Vishakha_Ratnakar/kNN-classification-using-scikit-learn-685b1ebafd13

Shmueli, G. (2015, August 19). Categorical predictors: How many dummies? BZST. https://www.bzst.com/2015/08/categorical-predictors-how-many-dummies.html

Stojiljković, M. (2020, November 23). *Train-test split in Python: Essentials for data science and machine learning*. Real Python. Retrieved June 12, 2024, from https://realpython.com/train-test-split-python-data/#author

H. Sources

Brownlee, J. (2020). Train-Test Split for Evaluating Machine Learning Algorithms. Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

DataCamp. (n.d.). *K-Nearest Neighbors (kNN) Classification with R*. Retrieved June 12, 2024, from https://datacamp.com/tutorial/k-nearest-neighbors-kNN-classification-with-r-tutorial

Çorbacıoğlu, Ş. K., & Aksel, G. (2023). *The AUC value is a measure of the ability of a model to distinguish between diseased and nondiseased individuals.* Turk J Emerg Med, 23(4), 195-198. https://doi.org/10.4103/tjem.tjem_182_23

Filho, M. (2023, March 25). *Is feature scaling required for the kNN algorithm?* Forecastegy. Retrieved June 12, 2024, from https://forecastegy.com/posts/is-feature-scaling-required-for-the-kNN-algorithm/

GeeksforGeeks. (2023, September 16). *Introduction to Pandas in Python*. Retrieved June 12, 2024, from https://www.geeksforgeeks.org/introduction-to-pandas-in-python/

Great Learning Team. (2024, April 30). *GridSearchCV: Everything you need to know*. Great Learning. Retrieved June 13, 2024, from https://www.mygreatlearning.com/blog/gridsearchcv/
Javatpoint. (n.d.). *StandardScaler in Sklearn*. Retrieved June 12, 2024, from https://www.javatpoint.com/standardscaler-in-sklearn

Harrison, O. (2018, September 10). *Machine learning basics with the k-nearest neighbors algorithm*. Towards Data Science. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

Kothari, A. (2020, October 10). *K-Nearest Neighbors (K-NN) with NumPy*. Medium. Retrieved June 12, 2024, from https://medium.com/analytics-vidhya/k-nearest-neighbors-k-nn-with-numpy-6c52cbe28093

Kumar, A. (2024, February 9). *Seaborn vs Matplotlib*. Techify Solutions. Retrieved June 12, 2024, from https://techifysolutions.com/blog/seaborn-vs-matplotlib/

Ratnakar, V. (2022, January 29). *kNN classification using scikit-learn*. Medium. Retrieved June 12, 2024, from https://medium.com/@Vishakha_Ratnakar/kNN-classification-using-scikit-learn-685b1ebafd13

scikit-learn developers. (n.d.). *scikit-learn: Machine learning in Python*. Retrieved June 12, 2024, from https://scikit-learn.org/

Srivastava, T. (2024, May 22). *Introduction to k-neighbors algorithm: Clustering*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/#:~:text=In%20the%20case%20of%20classification,for%20the%20input%20data%20point.

Stojiljković, M. (2020, November 23). *Train-test split in Python: Essentials for data science and machine learning*. Real Python. Retrieved June 12, 2024, from https://realpython.com/train-test-split-python-data/#author

The Quantitative Development Systems Initiative. (n.d.). *kNN tutorial*. The Pennsylvania State University. Retrieved June 26, 2024, from https://quantdev.ssri.psu.edu/sites/qdev/files/kNN_tutorial.html

Tutorialspoint. (n.d.). *Graph K-NN decision boundaries in Matplotlib*. Retrieved June 12, 2024, from https://www.tutorialspoint.com/graph-k-nn-decision-boundaries-in-matplotlib