D209 – Data Mining I - Task 2

WGU M.S. Data Analytics

Lyssa Kline

November 20, 2025

A, Research Question

A1, Question

The research question in scope for this analysis is, "Can we predict the total charge to a patient with the random forest technique?" Understanding the factors influencing the total amount charged to a patient can help medical organizations manage and potentially reduce costs. This insight could lead to more efficient allocation of resources, better financial planning, and greater transparency to patients regarding billing. Analyzing the factors affecting charges could help identify areas where costs are disproportionately high, which could lead to targeted efforts to streamline processes and improve the quality of care to patients.

A2, Analysis Goals

One goal of the data analysis based on the above research question would be to develop a model that can accurately predict hospital charges while considering the non-linear relationships and interactions between patient factors. The goal is to build a reliable, interpretable, and flexible model that helps predict and manage hospital charges based on patient-specific characteristics while uncovering patterns that can be used for cost optimization and improved patient care.

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 total charge to their bills using a random forest regressor technique.

B, Method Justification

B1, Classification Method Explanation

The Random Forest Regressor analysis analyses the medical dataset to predict the total amount charged to a patient during hospital admission by leveraging the relationships between patient characteristics (predictors) and hospital charges (target variable). This dataset includes features such as demographics (e.g., age and gender), health conditions (e.g., asthma, diabetes, or hypertension), and admission details (e.g., length of stay, type of procedures, and services used. The primary goal is to predict the total charges based on these characteristics while accounting for complex interactions and relationships.

The Random Forest Regressor works by creating multiple decision trees, each trained on random subsets of the data and features, a process known as bootstrapping. These trees make individual predictions, and the final model output is the average of all tree predictions. This ensemble approach captures non-linear relationships and interactions between features, such as how age combined with specific comorbidity might affect hospital charges and ensure robustness to overfitting. The model splits data based on feature thresholds (e.g., whether the length of stay is above or below a certain number), optimizing each split to minimize error. A random sampling of features at each split ensures that the model explores diverse patterns in the data, reducing bias from any single feature dominating the predictions.

Before training, the dataset undergoes preprocessing steps, such as encoding categorical variables (e.g., converting "Back Pain" into numerical values using one-hot encoding) and handling missing data. Unlike some regression models, Random Forest does not require scaling of continuous variables, as it is scale-invariant. After training, the model provides insights into which features are most influential in predicting charges. For example, "length of stay" might emerge as the most critical factor, followed by the number of procedures performed or the type of insurance. This feature importance analysis helps identify cost drivers and highlights opportunities to optimize resource allocation and reduce expenses.

Once trained, the model predicts hospital charges for new patients with high accuracy. For instance, it might predict that a 65-year-old patient with diabetes and a 10-day stay will incur charges of $18,500 based on patterns learned from similar patients. The model's interpretability can be enhanced using techniques like Partial Dependence Plots or SHAP values, which reveal how specific features (e.g., age or length of stay) influence charges and whether their effects are linear or exponential. Evaluation metrics such as Mean Squared Error (MSE) and R-squared are used to assess the model's performance. A low MSE, combined with an R-squared score close to 1, indicates that the model accurately predicts charges while generalizing well to unseen data.

The Random Forest Regressor is particularly suited to medical datasets because it handles non-linear relationships, mixed data types (categorical and continuous), and missing values effectively. By providing accurate predictions and insights into key cost drivers, it enables hospitals to estimate charges for patients, streamline resource allocation, and develop targeted cost-control strategies. For example, knowing that the "length of stay" contributes significantly to costs might help hospitals implement policies to reduce unnecessary days of admission without compromising care quality. The analysis thus combines predictive accuracy with actionable insights, making it a powerful tool for analyzing and managing hospital billing data.

B2, Assumption of Classification Method

One key assumption of a Random Forest Regressor analysis is that the relationships between the predictor variables (patient factors/characteristics) and the target variable (the total amount charged) are capturable through an ensemble of decision trees, meaning that the model assumes it can represent the underlying data patterns by repeatedly splitting the data into subsets based on feature thresholds.

This assumption implies that the data has a structure where patterns, correlations, or rules exist between the features and the target. Random Forest assumes that by averaging the predictions from multiple decision trees, the ensemble will generalize well to unseen data. While it does not require linearity or a specific data distribution (unlike linear regression), it assumes that enough meaningful splits can be created to accurately predict the target variable. If the data is extremely noisy or lacks a discernible relationship between features and the target, the Random Forest Regressor may struggle to perform effectively.

B3, Python Packages Justification

Each of the Python packages imported for this analysis plays a critical role in supporting the random forest regression 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 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 sklearn.model_selection import train_test_split
  from sklearn.metrics import mean_squared_error, r2_score
  from sklearn.ensemble import RandomForestRegressor
  from sklearn.preprocessing import OneHotEncoder
```

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.

Packages Supporting - Feature Selection, Preprocessing, and Scaling

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

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.

Mean_squared_error from sklearn.metrics evaluate the performance of the random forest regressor and provide quantitative insights into the model's accuracy and reliability. Mean_squared_error measures the average squared difference between predicted and actual hospital charges, penalizing larger errors more heavily.

r2_score from sklearn.metrics evaluate the performance of the random forest regressor and provide quantitative insights into the model's accuracy and reliability. R2_score indicates the proportion of variance in the target variable (hospital charges) explained by the model.

RandomForestRegressor from sklearn.ensemble implements the random forest algorithm for regression tasks. This builds and trains the Random Forest model to make predictions and provides feature importance metrics to identify key drivers.

C, Data Preparation

The below sections outline and explain the steps taken to clean, prepare, and output the selected data for a Random Forest Regression 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 key data preprocessing goal relevant to the Random Forest Regressor prediction method is ensuring that categorical variables are properly encoded into a numerical format without introducing bias or distorting relationships in the data.

The Random Forest algorithm works by splitting the data into subsets based on feature thresholds (e.g., "Is feature X > threshold?"). It requires numerical inputs to perform these operations. However, categorical variables (e.g., "Services" or "Gender") are non-numeric and cannot be directly processed by the model.

If categorical variables are left unprocessed, the model cannot use these features effectively, which could lead to ignoring important predictors or errors or failure during model training. The goal is to transform categorical variables into a numerical representation that preserves the relationships and distinctions between the categories, enabling the Random Forest Regressor to use these features effectively in its predictions.

Properly encoded categorical variables allow the Random Forest to split the data effectively based on these features, consider categorical features equally alongside continuous ones when building decision trees, and leverage the predictive power of categorical data to improve model accuracy. By achieving this preprocessing goal, you ensure that all features (both categorical and continuous) contribute meaningfully to the prediction of the target variable, such as the total amount charged to a patient in a hospital setting.

C2, Initial Variable Classification

To answer the research question, “Can we predict the total charge to a patient with the random forest technique?”. The dependent response variable would be ‘TotalCharge,’ 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 24 independent variables selected for further analysis.

- TotalCharge (Type: Continuous Numeric Variable) – Describes the total charged to a patient.
- Area (Type: Non-Ordinal Categorical Variable) – Describes what area type the patient lives in.
- ReAdmis (Type: Non-Ordinal Categorical Variable) – Describes if the patient was re-admitted or not.
- Age (Type: Discrete Numeric Variable) – Describes the patient’s age.
- Gender (Type: Non-Ordinal Categorical Variable) – Describes the patient’s gender.
- VitD_levels (Type: Continuous Numeric Variable) – Describes the patient’s vitamin D levels.
- Doc_visits (Type: Discrete 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: Non-Ordinal Categorical Variable) – Describes if the patient takes a vitamin D supplement or not.
- Soft_drink (Type: Non-Ordinal Categorical Variable) – Describes if the patient drinks soft drinks or not.

- Initial_admin (Type: Non-Ordinal Categorical Variable) – Describes if this is the patient's initial admittance to the hospital or not.
- HighBlood (Type: Non-Ordinal Categorical Variable) – Describes if the patient has high blood pressure.
- Stroke (Type: Non-Ordinal Categorical Variable) – Describes if the patient has had a stroke.
- Complication_risk (Type: Non-Ordinal Categorical Variable) – Describes if the patient is at risk for complication.
- Overweight (Type: Non-Ordinal Categorical Variable) – Describes if the patient is overweight.
- Arthritis (Type: Non-Ordinal Categorical Variable) – Describes if the patient has arthritis or not.
- Diabetes (Type: Non-Ordinal Categorical Variable) – Describes if the patient has diabetes or not.
- Hyperlipidemia (Type: Non-Ordinal Categorical Variable) – Describes if the patient has hyperlipidemia.
- BackPain (Type: Non-Ordinal Categorical Variable) – Describes if the patient has back pain.
- Anxiety (Type: Non-Ordinal Categorical Variable) – Describes if the patient has anxiety.
- Allergic_rhinitis (Type: Non-Ordinal Categorical Variable) – Describes if the patient has allergic rhinitis.
- Reflux_esophagitis (Type: Non-Ordinal Categorical Variable) – Describes if the patient has reflux esophagitis or not.
- Asthma (Type: Non-Ordinal Categorical Variable) – Describes if the patient has asthma.
- Services (Type: Non-Ordinal Categorical Variable) – Describes which services the patient received.
- Initial_days (Type: Discrete 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.

```
[1]: # initially select all variables for analysis
     selected_start = df[['TotalCharge','Area','ReAdmis','Age','Gender','VitD_levels','Doc_visits','Full_meals_eaten','v
```

The first step after initially selecting a handful of variables is to encode all the categorical variables. It is key for random forest regression 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=['ReAdmis','Soft_drink','HighBlood','Stroke','Overweight','Arthritis','Diabetes','Hyperlipidemia','BackPain','Anxiety','Allerg
print(initial_df_start)
```

```
      TotalCharge      Area  Age  Gender  VitD_levels  Doc_visits  \
0     3726.702860  Suburban   53    Male    19.141466           6
1     4193.190458     Urban   51  Female    18.940352           4
2     2434.234222  Suburban   53  Female    18.057507           4
3     2127.830423  Suburban   78    Male    16.576858           4
4     2113.073274     Rural   22  Female    17.439069           5
...           ...       ...  ...     ...          ...         ...
9995  6850.942000     Urban   25    Male    16.980860           4
9996  7741.690000     Urban   87    Male    18.177020           5
9997  8276.481000     Rural   45  Female    17.129070           4
9998  7644.483000     Rural   43    Male    19.910430           5
9999  7887.553000     Urban   70  Female    18.388620           5

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

     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 25 columns]
```

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

```
      TotalCharge  Age  VitD_levels  Doc_visits  Full_meals_eaten  vitD_supp  \
0     3726.702860   53    19.141466           6                 0          0
1     4193.190458   51    18.940352           4                 2          1
2     2434.234222   53    18.057507           4                 1          0
3     2127.830423   78    16.576858           4                 1          0
4     2113.073274   22    17.439069           5                 0          2
...           ...  ...          ...         ...               ...        ...
9995  6850.942000   25    16.980860           4                 2          1
9996  7741.690000   87    18.177020           5                 0          0
9997  8276.481000   45    17.129070           4                 2          0
9998  7644.483000   43    19.910430           5                 2          1
9999  7887.553000   70    18.388620           5                 0          1

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

Now that the dataset has been properly assessed, cleaned, and encoded, we can proceed with developing random forest 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 the model. See the output of the now cleaned dataset below, as well as an exported version under C4 below.

```
print(df_selected)
```

```
      TotalCharge  Age  VitD_levels  Doc_visits  Full_meals_eaten  vitD_supp  \
0     3726.702860   53    19.141466           6                 0          0
1     4193.190458   51    18.940352           4                 2          1
2     2434.234222   53    18.057507           4                 1          0
3     2127.830423   78    16.576858           4                 1          0
4     2113.073274   22    17.439069           5                 0          2
...           ...  ...          ...         ...               ...        ...
9995  6850.942000   25    16.980860           4                 2          1
9996  7741.690000   87    18.177020           5                 0          0
9997  8276.481000   45    17.129070           4                 2          0
9998  7644.483000   43    19.910430           5                 2          1
9999  7887.553000   70    18.388620           5                 0          1

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

C4, Prepared Data

A prepared dataset outputted in CSV format can be found within the attached 'prepared_medical_task2.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 Random Forest Regressor prediction model.

D1, Train and Test Data Sets

Splitting the dataset into train and test sets is a crucial part of a Random Forest prediction 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 Random Forest 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-task2.csv' file.
- A test dataset outputted in CSV format can be found within the attached 'test_data-task2.csv' file.

D2, Analysis Technique & Calculations

The analysis shown in the code below uses the Random Forest Regressor from the sklearn.ensemble library to predict the target variable 'TotalCharge' based on the features selected from the provided medical dataset. This technique belongs to the family of ensemble learning methods and is particularly well-suited for handling non-linear relationships and interactions between features in regression tasks.

The key steps in the analysis include defining features (X) and the Target variable of 'TotalCharge' (y). The dataset is divided into training (X_train, y_train) and testing (X_test, y_test) subsets using an 80-20 split. Next, a random forest regressor with 100 decision trees is trained on the training data. The trained model is used to make predictions (y_pred) on the test dataset.

After building the model, to gain deeper insights and improve the analysis, intermediate calculations are performed on the code. Feature importance analyzes which features contribute most to predicting TotalCharge; this helps identify key cost drivers. Additionally, residual analysis examines the residuals (difference between actual and predicted values) to detect patterns or biases in predictions. Residuals should ideally be normally distributed around zero. Any systematic bias could indicate a need for additional feature engineering. Lastly, cross-validation

can be calculated to evaluate the model's performance across multiple data splits to ensure consistent accuracy.

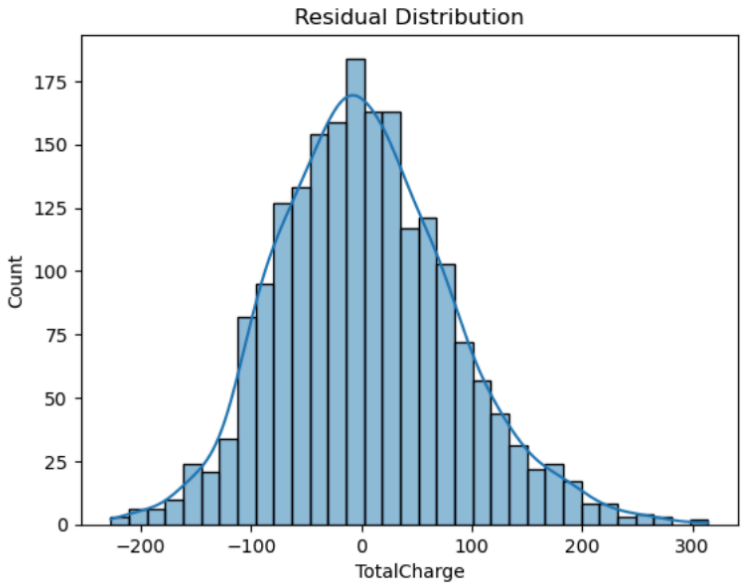
See the code and explanation of the output below.

```
feature_importance = rf.feature_importances_
importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importance}).sort_values(by='Importance', ascending=False)
print(importance_df)
```

	Feature	Importance
5	Initial_days	0.975501
33	Initial_admin_Emergency Admission	0.013411
29	Complication_risk_High	0.007400
6	ReAdmis_Yes	0.001491
8	HighBlood_Yes	0.000430
13	Hyperlipidemia_Yes	0.000237
14	BackPain_Yes	0.000181
1	VitD_levels	0.000169
15	Anxiety_Yes	0.000169
0	Age	0.000167
11	Arthritis_Yes	0.000126
12	Diabetes_Yes	0.000104
16	Allergic_rhinitis_Yes	0.000093
17	Reflux_esophagitis_Yes	0.000079
2	Doc_visits	0.000063
3	Full_meals_eaten	0.000055
4	vitD_supp	0.000034
10	Overweight_Yes	0.000023
25	Services_Blood Work	0.000023
7	Soft_drink_Yes	0.000022
20	Area_Suburban	0.000022
21	Area_Urban	0.000022
19	Area_Rural	0.000021
18	Asthma_Yes	0.000021
27	Services_Intravenous	0.000020
9	Stroke_Yes	0.000020
22	Gender_Female	0.000018
23	Gender_Male	0.000017
26	Services_CT Scan	0.000014
31	Complication_risk_Medium	0.000010
28	Services_MRI	0.000009
30	Complication_risk_Low	0.000009
34	Initial_admin_Observation Admission	0.000007
32	Initial_admin_Elective Admission	0.000007
24	Gender_Nonbinary	0.000005

The output displays the feature importance scores derived from the Random Forest Regressor model. These scores indicate how much each feature contributes to predicting the target variable (TotalCharge). Features with higher importance values have a stronger influence on the model's predictions, while those with lower values contribute less. This feature overwhelmingly dominates the model's predictions, contributing nearly 97.5% of the predictive power. The length of the patient's stay in the hospital (Initial_days) is the most critical factor for determining the total charges, as longer stays typically result in higher costs due to additional resources, treatments, and services.

```
]: residuals = y_test - y_pred
   sns.histplot(residuals, kde=True)
   plt.title('Residual Distribution')
   plt.show()
```



The residual distribution plot visualizes the differences between the actual values and the predicted values. These differences, called residuals, are computed as Residual = ytest – ypred. The residuals represent the error in the model's predictions. The histogram of residuals is approximately symmetric and bell-shaped, suggesting a near-normal distribution of the errors. This is a desirable outcome as it indicates that the model's errors are not systematically biased in one direction (e.g., consistently underpredicting or overpredicting). The residuals are centered around zero, which shows that the model's predictions are unbiased on average. The residuals range roughly from -200 to 300, with most concentrated between -100 and 100. This shows that the model generally predicts charges close to the actual values, with only a few larger errors.

The Random Forest Regressor was chosen because it handles non-linear relationships and provides robust performance without requiring heavy preprocessing like scaling or feature selection. Lastly, we can produce performance metrics like mean squared error and accuracy results to calculate and indicate prediction accuracy. See additional details of the performance metrics under E1 below.

D3, Analysis Code

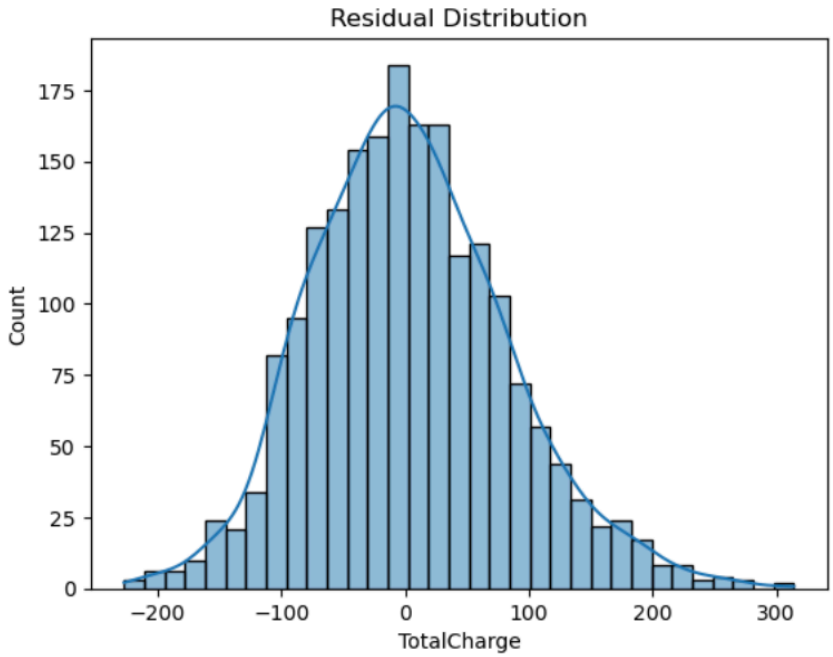
The code printed below outlines the steps used to perform the prediction analysis from D2 above. These steps included selecting variables, splitting the data, encoding the data, random forest regression model initialization and training, combining train and test data, model prediction and probability calculation, intermediate calculations, and lastly, evaluating the model performance with AUC and MSE.

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

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

# Initialize the model
rf = RandomForestRegressor(n_estimators=100, random_state=42)

# Fit the model
rf.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-task2.csv', index=False)
test_data.to_csv('test_data-task2.csv', index=False)

# Predict on test data
y_pred = rf.predict(X_test)
```

```python
feature_importance = rf.feature_importances_
importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importance}).sort_values(by='Importance', ascending=Fal
print(importance_df)
```

```
                              Feature  Importance
5                        Initial_days    0.975501
33   Initial_admin_Emergency Admission    0.013411
29              Complication_risk_High    0.007400
6                         ReAdmis_Yes    0.001491
8                        HighBlood_Yes    0.000430
13                  Hyperlipidemia_Yes    0.000237
14                       BackPain_Yes    0.000181
1                         VitD_levels    0.000169
15                        Anxiety_Yes    0.000169
0                                 Age    0.000167
11                       Arthritis_Yes    0.000126
12                        Diabetes_Yes    0.000104
16               Allergic_rhinitis_Yes    0.000093
17              Reflux_esophagitis_Yes    0.000079
2                          Doc_visits    0.000063
3                     Full_meals_eaten    0.000055
4                           vitD_supp    0.000034
10                     Overweight_Yes    0.000023
25                Services_Blood Work    0.000023
7                       Soft_drink_Yes    0.000022
20                       Area_Suburban    0.000022
21                          Area_Urban    0.000022
19                          Area_Rural    0.000021
18                          Asthma_Yes    0.000021
27                Services_Intravenous    0.000020
9                           Stroke_Yes    0.000020
22                       Gender_Female    0.000018
23                         Gender_Male    0.000017
26                    Services_CT Scan    0.000014
31            Complication_risk_Medium    0.000010
28                        Services_MRI    0.000009
30               Complication_risk_Low    0.000009
34  Initial_admin_Observation Admission    0.000007
32     Initial_admin_Elective Admission    0.000007
24                    Gender_Nonbinary    0.000005
```

```
residuals = y_test - y_pred
sns.histplot(residuals, kde=True)
plt.title('Residual Distribution')
plt.show()
```




```
# Calculate performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R² Score: {r2}")
```

```
Mean Squared Error: 6366.100403744898
R² Score: 0.9986475924476945
```

E, Data Summary and Implications

The below sections provide an overview of the Random Forest Regressor analysis performed. The sections outline the data summary, key results, implications, and limitations and recommend a course of action.

E1, Accuracy, and MSE

Accuracy and MSE (Mean Squared Error) are two metrics used to evaluate the performance of the regression model. See the discussion below of the results.

```
# Calculate performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R² Score: {r2}")
```

```
Mean Squared Error: 6366.100403744898
R² Score: 0.9986475924476945
```

The R2 score, also known as the coefficient of determination, is .9986. This indicates that the model explains 99.86% of the variance in the target variable (TotalCharge). This result suggests that the model has captured the underlying patterns in the data with very high accuracy. A score close to 1 means the predictions are very close to the actual values for most data points. The high R2 value demonstrates that the model is highly reliable for predicting hospital charges based on patient factors.

The MSE is the average squared difference between the actual (y_test) and predicted (y_pred) values. While the MSE doesn't have an intuitive unit, lower values indicate better model performance. An MSE of 6366.1 means the average squared error for the model's predictions is relatively low, further supporting the model's high accuracy. The MSE shows that the model's prediction errors are small, and most predictions are close to the actual charges. However, since MSE penalizes larger errors more heavily, this metric ensures that significant deviations are accounted for, suggesting that the model is robust even for outliers.

When tying these together, we can see that the high R2R^2R2 score indicates that the model is very accurate in explaining the variability in hospital charges. The low MSE shows that the average error in the predictions is small, which confirms that the model performs well quantitatively. Together, these metrics demonstrate that the model is both accurate and precise.

E2, Results and Implications

The analysis using the Random Forest Regressor demonstrated exceptional performance in predicting the total charges to patients based on their characteristics and admission details. The results reveal key insights and practical implications for organizational decision-making.

The model produced high predictive accuracy. The model achieved an R2 score of 0.9986, indicating that it explains 99.86% of the variance in hospital charges. This level of accuracy suggests that the model captures nearly all relevant patterns in the data, making it highly reliable for predicting patient costs. Additionally, the Mean Squared Error (MSE) of 6366.1 shows that the average squared difference between the predicted and actual charges is small. This further supports the model's precision in forecasting total charges. The feature importance analysis revealed that Initial_days (Length of Stay) is the most critical predictor, contributing over 97.5% to the model's predictions. This aligns with real-world expectations, as hospital charges often accumulate based on the duration of a patient's stay.

Some other influential features found include emergency admissions, which typically incur higher costs due to urgent and intensive care, and high-risk complication patients require more resources, leading to increased charges. Several features (e.g., demographics like gender and non-

severe conditions) showed minimal importance, suggesting they have little direct impact on hospital charges.

The residual distribution was centered around zero and approximately normal, indicating no systematic bias in the model's predictions. This reinforces confidence in the model's reliability. Lastly, the cross-validation results (mean R2R score of 0.988) confirm the model's robustness across different subsets of the data, highlighting its ability to generalize well to unseen data.

Some implications of the analysis include:

- The high predictive accuracy of the model provides a reliable tool for real-time charge estimations for financial planning, billing and insurance, and transparency with patients.
- The dominance of length of stay as a predictor highlights a critical opportunity for cost control to optimize the length of stay and prevent complications.
- The analysis identifies key cost drivers, which can guide strategic decisions to focus on high-impact factors (e.g., reducing emergency admissions through improved outpatient care or better preventive services). Or allocate resources more effectively to areas associated with higher costs, such as emergency departments or intensive care units.
- The success of the model opens avenues for broader applications to continue predicting readmissions, provide clinical decision support, and continue policy development.
- While the model's predictions are highly accurate, its use comes with important considerations like transparency and communication, data privacy concerns, and avoiding over-optimization.

The analysis demonstrates that the Random Forest Regressor is a highly effective tool for predicting hospital charges with exceptional accuracy. By implementing this model, hospitals can improve financial planning, enhance transparency, and identify opportunities for cost savings while maintaining patient care quality. However, it is essential to address ethical considerations, interpretability challenges, and the need for ongoing validation to ensure the model's long-term success and usability in real-world settings.

E3, Analysis Limitations

One key limitation of using a Random Forest Regressor for this type of data analysis is its potential lack of interpretability compared to simpler models, such as linear regression.

A Random Forest consists of multiple decision trees, each trained on different subsets of data and features, and the final prediction is an average of all tree outputs. While this ensemble method improves predictive accuracy and robustness, it becomes challenging to interpret the exact relationships between input features (e.g., patient factors) and the target variable (e.g., hospital charges).

Although the Random Forest provides a measure of feature importance, it does not directly reveal the direction of the relationship (e.g., whether increasing "Length of Stay" increases or decreases hospital charges). Interactions between features (e.g., how "Age" and "Complication Risk" together impact charges) are not explicitly visible in feature importance rankings.

For stakeholders (e.g., hospital administrators or clinicians), understanding how predictions are made can be just as important as the predictions themselves. Random Forest models can feel like a "black box," offering limited transparency.

In a healthcare setting, decisions based on predictive models often need to be explained to stakeholders for validation and trust. Questions like, why does a model predict that a certain patient will have high charges? Or which factors should clinicians or administrators focus on to reduce costs? If the model lacks interpretability, it may be difficult to gain trust or provide actionable insights.

While the Random Forest Regressor offers high predictive accuracy and robustness, its complexity and lack of interpretability can be a limitation, especially in contexts like healthcare, where understanding and explaining predictions are critical. Addressing this limitation through explainability techniques or complementary methods can help balance predictive power with interpretability.

E4, Course of Action

Based on the results of your analysis, including the high R2R^2R2 score (0.9986), the low Mean Squared Error (MSE) (6366.1), and the feature importance rankings, the Random Forest Regressor has demonstrated its effectiveness for predicting the total charge to a patient.

A recommended course of action would be to adopt the random forest regressor for charge prediction, integrating the random forest remotely into the hospital's billing or administrative systems could help provide real-time estimates of total patient charges based on patient factors. This model could be implemented as a decision support tool and help provide insights for financial planning or patient transparency.

After implementation, continuous monitoring and validation are essential to ensure the model maintains accuracy. It is recommended to regularly update the model with new data to reflect changes in medical practices, cost structures, or patient demographics. Additionally, exploring other applications of the model could open up opportunities to use similar predictive techniques in other areas of hospital management.

The Random Forest Regressor is a highly effective tool for predicting hospital charges and should be implemented as part of the hospital's financial and operational systems. By leveraging its predictive capabilities, hospitals can enhance resource planning, improve patient transparency, and reduce billing inefficiencies. However, it is crucial to continuously monitor and validate the model, focus on interpretability, and address ethical considerations to maximize its value in a real-world organizational setting.

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-Task2-Kline"

Panopto video link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=97dc315f-a35a-45b0-81e5-b22e0179b0ad

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.