**D209 Task 2: Predictive Analysis**

**Joseph Duszynski**

**ID #011237236**

**A.1. Proposal of question**

Using the medical data set (WGU, 2024 [1]), this project seeks to answer the question “Which patient variables influence their average daily charge (‘TotalCharge’) and can this data create reliable predictions through a random forest regression model?”

**A.2. Defined goal**

This project aims to find the most influential variables affecting a patient’s average daily charge and build a random forest regression model to accurately predict it. These insights can assist hospital administrators and analysts understand what factors drive a patient’s daily average expenses.

**B.1. Explanation of prediction method**

Random forest regression uses an ensemble of decision trees to predict a numerical target variable by taking the mean of the decision trees’ predictions. Averaging over multiple estimators (trees) has the advantage of reducing overfitting and the impact of outliers (Elleh, 2023 [1]). Each decision tree divides the data into sequentially smaller subsets through conditions over a random subset of the features at each node (e.g. “columni ≤ 10” goes to the left child node while “columni > 10” goes to the right child). Depending on the parameters of the model, the number of data points in a given leaf node and/or the tree depth may have limitations placed upon it. The value of a terminal leaf node (having traced the tree with an input data point) is the predicted value for that input. The ensemble of trees is constructed at runtime on the training data through bootstrapping. Test data (and any future data) is then entered into each of the decision trees with the model’s prediction being the mean of the trees’ predictions.In this project, values of a set of independent variables (to be discussed in future sections) for a patient produce a numerical prediction for the patient’s average daily charge (‘TotalCharge’).

**B.2. Summary of method assumption**

Random forests, and their component decision trees, assume representative sampling on the training data. Imbalance in distributions can lead to many of the trees never encountering data from minority clusters and underrepresented parts of the distribution. The resulting ensemble may fail to properly integrate the correlations in the underrepresented data or even fail entirely to capture it. Heavily skewed distributions (or class imbalance in one-hot encoded categorical input) can produce inaccurate predictions (Elleh, 2023 [1]).

**B.3. Packages or libraries list**

I chose Python 3.9 for its ease of use, widespread adoption, personal familiarity, speed, multitude of mathematical and statistical packages, and extensive documentation and examples online for debugging and understanding. I prefer the consistent syntax of Python and its packages compared to libraries in R. Additionally, I wrote data cleaning functions in Python for the D206 course, making it convenient to repurpose that work here.

Numpy and pandas were used for handling numerical computations on arrays and dataframes, respectively. They’re effectively mandatory for any of the project to work. Matplotlib and seaborn were needed to produce histograms and scatterplots. Scipy.stats was required for calculating z-scores to be used in creating z-score histograms for outlier searching. Statsmodels was used for variance inflation factor calculations when inspecting multicollinearity and creating QQ plots. From scikit-learn, various packages were used in selecting the most relevant features for the model (SelectKBest() and f\_regression for F-tests); the needed calculations for the mean squared error, mean absolute error, and R2 coefficient, and cross validation; scalers for normalizing the data (testing purposes only); RandomForestRegressor() for creating the random forest regression model; train\_test\_split was used to split the independent and dependent variables into training and test data; and GridSearchCV for hyperparametric tuning.

A screenshot of a computer program

Description automatically generated

**C.1. Data preprocessing**

In addition to the usual steps of data cleaning such as handling null values, addressing outliers, and verifying there are no duplicates, the random forest regressor requires one-hot encoding to utilize categorical variables as input, as it uses numerical data when splitting at nodes.

This is accomplished through the function one\_hot\_encoder which creates dummy variables (through pandas.get\_dummies(df[column])), converts the Boolean values to 32-bit integers (1s and 0s), and creates a new column of the form ‘{column\_name}\_{column\_label}’. For instance, a binary categorical variable like ‘Stroke’ introduces the column ‘Stroke\_Yes’. Binary categorical variables only require one encoded column to completely specify the presence or absence of a condition (i.e. the drop\_first=True condition is used in get\_dummies). For categorical variables withclasses, encoded columns are introduced for each class, as the decisions trees in a random forest model cannot identify the presence of the *k*th dropped class through columns all recording 0.

See below:

A screen shot of a computer program

Description automatically generated

**C.2. Data set variables**

Before further processing in future sections (which will identify the final variables used in section D.2.), the independent variables to initially be used in the random forest regression model are the following (note that the variable ‘TotalCharge’ is not part of the predictor variables as it’s the target variable):

Lat: numeric

Lng: numeric

Population: numeric

Area: categorical

TimeZone: categorical

Children: numeric

Age: numeric

Income: numeric

Marital: categorical

Gender: categorical

ReAdmis: categorical

VitD\_levels: numeric

Doc\_visits: numeric

Full\_meals\_eaten: numeric

vitD\_supp: numeric

Soft\_drink: categorical

Initial\_admin: categorical

HighBlood: categorical

Stroke: categorical

Complication\_risk: categorical

Overweight: categorical

Arthritis: categorical

Diabetes: categorical

Hyperlipidemia: categorical

BackPain: categorical

Anxiety: categorical

Allergic\_rhinitis: categorical

Reflux\_esophagitis: categorical

Asthma: categorical

Services: categorical

Initial\_days: numeric

TotalCharge: numeric

Additional\_charges: numeric

Item1: numeric

Item2: numeric

Item3: numeric

Item4: numeric

Item5: numeric

Item6: numeric

Item7: numeric

Item8: numeric

**C.3. Steps for analysis**

First, the dataset backs up integer ‘Zip’ values to a new column ‘Zip\_int64’ then replaces the values in the ‘Zip’ column with categorical strings of five digits (i.e. leading 0s are restored). ‘CaseOrder’ is also converted to a string as it and ‘Zip’ are more accurately regarded as categorical data. A few simple commands verify there are no null values or duplicates.

A screen shot of a computer code

Description automatically generated

The functions inspect\_data and outlier\_search are used to verify no outliers distort the analysis and that column constraints are obeyed.

A screen shot of a computer program

Description automatically generatedA black rectangular object with white text

Description automatically generated

The function one\_hot\_encoder uses pandas.get\_dummies to create binary numeric columns for the presence or absence of a given class for each categorical feature in columns\_to\_encode. As mentioned in section C.1., drop\_first=True is only used on the binary categorical columns (that record ‘Yes’/’No’ values). This is a necessary transformation step for (raw) categorical variables to be valid input in a random forest regression model, as it requires numerical input data when splitting at nodes.

A screen shot of a computer program

Description automatically generated

As an initial step in inspecting correlations between the target variable and other columns, the function corr\_search displays correlation matrix values in descending order.

A screen shot of a computer program

Description automatically generated

**C.4. Cleaned data set**

For a dataset with outliers, duplicates, and null values addressed, as well as encoded columns, see “medical\_cleaned\_prediction.csv”. For a subset of that file including only the final independent and dependent variables, see “medical\_transformed\_prediction.csv”.

**D.1. Splitting the data**

The data is split into training and test data using the following:

A screenshot of a computer program

Description automatically generated

See “X\_train.csv” and “X\_test.csv” for the training and test data, respectively, for the independent variables. See “y\_train.csv” and “y\_test.csv” for the training and test data, respectively, for the dependent variable.

**D.2. Output and intermediate calculations**

Having preprocessed the data with the cleaning and transformation steps discussed in part C., split the independent and dependent variables into training and test data, we can now begin creating a random forest model. Using the initial “kitchen sink” approach discussed in section C.2., we define y\_0 and X\_0, with the dependent variable being ‘TotalCharge’ (X\_1 will be used later):

A screen shot of a computer program

Description automatically generated

The function hyper\_param uses GridSearchCV to look for the random forest model with the lowest MSE:

A screen shot of a computer program

Description automatically generated

For X\_0, it has output:

A screen shot of a computer

Description automatically generated

To improve upon this, we use the feature\_selection function with a focus on features with a p-value below 0.05:

A computer screen shot of a program code

Description automatically generated

For X\_0 and , it has output:

A screenshot of a computer program

Description automatically generated

Using the 17 variables below the cutoff, we have a temporary set of important features:

['ReAdmis\_Yes', 'Initial\_days', 'Initial\_admin\_Emergency Admission', 'Complication\_risk\_High', 'Complication\_risk\_Medium', 'Initial\_admin\_Observation Admission', 'Initial\_admin\_Elective Admission', 'BackPain\_Yes', 'Arthritis\_Yes', 'Anxiety\_Yes', 'Additional\_charges', 'Reflux\_esophagitis\_Yes', 'Marital\_Divorced', 'Children', 'TimeZone\_America/Phoenix', 'HighBlood\_Yes', 'Item1']

To inspect performance by the number of features, the function hyper\_feature is used, which is additional hyperparametric tuning that sequentially adds a predictor variable to the model from the above list in the order they appear. The above list is sorted in ascending order of p-values below 0.05.

A screen shot of a computer code

Description automatically generated

It finds the lowest MSE of 6881.38 with 16 variables, compared to an MSE of 7188.65 with 17 variables, meaning the model is improved by removing the final variable ‘Item1’. The final list of 16 variables is:

* ReAdmis\_Yes (one-hot encoded categorical)
* Initial\_days (numeric)
* Initial\_admin\_Emergency Admission (one-hot encoded categorical)
* Complication\_risk\_High (one-hot encoded categorical)
* Complication\_risk\_Medium (one-hot encoded categorical)
* Initial\_admin\_Observation Admission (one-hot encoded categorical)
* Initial\_admin\_Elective Admission (one-hot encoded categorical)
* BackPain\_Yes (one-hot encoded categorical)
* Arthritis\_Yes (one-hot encoded categorical)
* Anxiety\_Yes (one-hot encoded categorical)
* Additional\_charges (numeric)
* Reflux\_esophagitis\_Yes (one-hot encoded categorical)
* Marital\_Divorced (one-hot encoded categorical)
* Children (numeric)
* TimeZone\_America/Phoenix (one-hot encoded categorical)
* HighBlood\_Yes (one-hot encoded categorical)

Running hyper\_param(y\_0, X\_1) has output:

A screenshot of a computer

Description automatically generated

Using n\_estimators=400, max\_features=8, and max\_depth=None in predictor\_rf:

A screen shot of a computer program

Description automatically generated

Output:

A screenshot of a computer

Description automatically generated

A diagram of blue dots

Description automatically generated

A graph with a line

Description automatically generated

**D.3. Code execution**

The entire code can be found in the included file “prediction.py”.

For the random forest prediction model portion using results found in section D.2., see below:

A screen shot of a computer program

Description automatically generated

**E.1. Accuracy and MSE**

The accuracy of this model is evaluated through its mean squared error (MSE) and , while also tracking the mean absolute error (MAE) and root mean squared error (RMSE – the square root of the MSE).

The MSE calculates the mean of the squares of the differences between the observed values and predicted values.

Here, there are observations in the dataset, are the observed values of the target variable, and are the predicted (fitted) values (for observation ).

The coefficient of determination, , is a value between 0 and 1, given by

The final random forest regression model from the above section has an MSE of 6881 on the test data (82.95 RMSE), MAE of 66.08, and . In contrast, the initial “kitchen sink” model (with X\_0) had an MSE of 149626 on the test data (386.8 RMSE), MAE of 312.4, and . For the selected 16 independent variables in the final model, the random forest minimizes the MSE (across the designated parameter grid) using 400 estimators, no tree depth, and no more than 8 features used in a node’s split.

**E.2. Results and implications**

The resulting random forest model from this project looking to predict patient’s average daily charges (‘TotalCharge’) has an MSE of 6881 (82.95 RMSE) and using the 16 predictor variables listed in section D.2.

The is virtually 1.0, demonstrating the model’s strong statistical significance. Looking at the scatter plot of residuals in section D.2., they’re reasonably randomly distributed, with the gap near the mean of ~5000 aligned with the lack of data in that region of the bimodal distribution for ‘TotalCharge’. The residuals do not appear to have any extreme or unexpected bias. Errors range from 0-10% of a patient’s average daily charge, generally closer to 1-2%. The QQ plot of residuals is close to a straight line with a notable deformation around the mean.

Although the model is statistically significant and a very accurate predictor, the non-trivial difference in MSE between the training and test data indicates there may be some overfitting that could be compromised with future data requiring accurate predictions. The bimodal distribution of ‘TotalCharge’ is skewed to the right, so the resulting imbalance could have increased the residual errors, but reviewing the residuals plot in section D.2. shows that isn’t the case.

Additionally, a billing error of 2% or more is still significant for a patient, so this model is better used as an internal analysis, forecasting, or verification tool rather than something that patients might encounter.

**E.3. Limitation**

The substantial drop in F-scores from ‘Initial\_days’ and ‘ReAdmis\_Yes’ to the variable with the third lowest p-value, ‘Initial\_admin\_Emergency Observation’, indicate the model is dominated by ‘Initial\_days’ and ‘ReAdmis\_Yes’. Feature importance scores of the model are 0.256 for ‘ReAdmis\_Yes’, 0.719 for ‘Initial\_days’, 0.010 for ‘Initial\_admin\_Emergency Observation’, and the remaining values are below 0.01.

This extreme imbalance in feature scores, despite all associated p-values being sufficiently low (below 0.05), indicates the model is overly reliant on the two features ‘Initial\_days’ and ‘ReAdmis\_Yes’. While the MSE, , and performance of the model with the current data is remarkable, should there be outliers or significant changes in relationships between variables in the future that impact ‘Initial\_days’ and ‘ReAdmis’, the model’s accuracy and predictive power would be in serious jeopardy.

A more robust model would benefit from additional data collection, such as the reason a patient is readmitted, individual items and procedures that contribute to their billed expenses, why they were admitted for the initial stay (beyond the type of admission such as “Emergency”), time spent when readmitted if applicable (and how it differed from the initial stay), time spent with a doctor (and how that’s billed), billing broken down by day or week rather than an overall average, among other factors. Currently, this model primarily relies on the high correlations between ‘Initial\_days’ and ‘ReAdmis’ with the target ‘TotalCharge’. The predictions are currently reliable, but any causal relationships remain unknown for the time being.

More importantly, as hospital prices and supplier prices can change quickly, yet aren’t reflected in this dataset in any meaningful capacity, this model will routinely need to be recalibrated with recent data whenever prices change.

**E.4. Course of action**

In seeking an answer to the research question can we use existing patient data to accurately predict their average daily charge (‘TotalCharge’) and determine the most relevant predictors, a statistically significant random forest model of 16 predictor variables was created with an MSE of 6881 (82.95 RMSE) and .

As discussed in sections E.2. and E.3., the long-term viability of the model remains to be seen as it’s overly reliant on ‘Initial\_days’ and ‘ReAdmis’. The predictions are very accurate statistically speaking, but billing errors of ~2% or more aren’t ignorable for a patient. Hospital staff can use the predictions from this model for forecasting revenue, future analysis projects, and secondary verification of the accuracy of a patient’s bill. If human or system error produces an incorrect bill that’s extremely anomalous compared to what would be predicted by this model, it can be automatically flagged for human review.

The model’s current predictive ability shouldn’t be mistaken for causation or absolute, as its deterioration over time with price increase and other changes is nearly certain (without updating the data and model). It would be of benefit to collect more data and understand currently unknown relationships that may not be captured with the existing data columns. For instance, it’s highly unlikely being divorced has a causal relationship to a patient’s bill and instead has an unknown underlying factor leading to changes in expenses for those patients.

For examples of additional data collection, numerical blood pressure readings, patient diseases, and expected prognosis day-to-day would be very helpful in producing better approximations that are relatively future proof (and less dependent on the length of initial stay). Binary representations of conditions, such as blood pressure, may mask potential relationships to patient average daily charge that could in turn create more accurate models.

**F. Panopto recording**

See the attached link: https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=a1e5d5fa-5cb3-4fa6-ab13-b166015a4559

**G. Sources for third-party code**

**1.** WGU. 2024. D209 Data Mining I “Data Sets and Associated Data Dictionaries”. Medical Data and Dictionary Files. Retrieved April 29, 2024, from <https://access.wgu.edu/ASP3/aap/content/g9rke9s0rlc9ejd92md0.html>.

**H. Sources**

**1.** Elleh, Festus. 2023. WGU “D209 Data Mining 1 Task 2 Cohort.pptx”. Retrieved May 4, 2024 from https://westerngovernorsuniversity.sharepoint.com/:p:/r/sites/DataScienceTeam/\_layouts/15/Doc.aspx?sourcedoc=%7B4B73AB47-67CB-4E1F-ABF9-33FAB2B3E26F%7D&file=D209%20Data%20Mining%201%20Task%202%20Cohort.pptx&action=edit&mobileredirect=true.