**D209 Data Mining 1**

**Task 2: Predictive Analysis**

Petra I Bier

Western Governors University

D209 Data Mining

May 2, 2024

Dr. Festus Elleh, PhD

**D209 Data Mining 1**

**Task 2: Predictive Analysis**

**Part I: Question**

A :

A1: Question

Can a random forest regression model accurately predict the average daily amount charged (TotalCharge) to a patient during their stay?

A2: Goal

Medical costs to patients and insurance companies are rising. A thorough data analysis to find patterns and trends to reduce costs could benefit all parties. Preventative interventions could be developed based on the results of the analysis, which would reduce costs for insurance companies and improve patient outcomes.

This analysis aims to determine which factors can be used to predict patient daily costs. This information can then be used to develop preventative programs to help lower insurance companies' and patients' costs. These could include interventions at a community level, such as reducing patient-specific factors that are found to influence increased costs.

**Part II Method**

B :

B1: Prediction Method

The method chosen for this analysis is random forest regression. The target variable for this analysis, TotalCharge, is a continuous numeric value. Random forest models can be used for both classification and regression prediction problems, and in this case, it will be built as a regression model since the target variable is numeric.

This type of model is built on the idea of a decision tree which splits data based on specific criteria. A node is created where a decision is made in an “if-than-else” manner, and the data is split based on these criteria. Another node with another criterion is made until all the data has been broken down into the resulting endpoints, also known as leaves.

The random forest model takes this one step further and builds many decision trees based on different criteria selection for each tree. The overall model is more complex and robust than a single tree and can perform well even when outliers and missing data exist.

This analysis's expected outcome is a model that can predict the average daily cost to a patient using the known predictive variables.

B2: Method Assumption

One assumption that random forest models have is that the data does not have to have a specific shape, such as a Gaussian distribution or linear relationship (Vishalmendekarhere, 2021). This contrasts with other regression models, which have assumptions on the input data, variable relationships, and often the residuals of a model. Random forest models also assume that the data is fully representative of the population (Sahai, 2023).

B3: Packages and Libraries

|  |  |
| --- | --- |
| **Packages and Libraries** | **Usage in analysis** |
| pandas | Import data into DataFrame, One hot encoding |
| numpy | Objects are arrays for calculations |
| matplotlib.pyplot | Visualizations |
| seaborn | Visualizations |
| sklearn.model\_selection import train\_test\_split | Split the data into training and testing sets |
| sklearn.preprocessing import StandardScaler | Variable standardization |
| sklearn.feature\_selection import SelectKBest, f\_regression | Feature selection |
| from sklearn.ensemble import RandomForestRegressor | Random Forest Model |
| sklearn.model\_selection import GridSearchCV | Hyperparameter testing |
| sklearn import metrics | View metric results |
| from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, root\_mean\_squared\_error, r2\_score | View metric results |
| scipy import stats | Outlier winsorization |
| scipy.stats.mstats import winsorize | Outlier winsorization |

**Part III Preparation**

C :

C1: Preprocessing

When creating a random forest regression model, it is necessary to preprocess the data. Preprocessing tasks can include cleaning and ensuring no missing or null values. Data wrangling tasks can also be part of the preprocessing. These tasks ensure an accurate, well-fitted model. Two preprocessing tasks occurred in the data wrangling phase for this random forest model. These included hot encoding and scaling the data. 1. One hot encoding is used to change categorical nominal data into a numeric value that can be used within the model. If the categories are not ordered, a mistake can be made by giving these categories a numeric value. The model will then assume that there is a relationship between the variables. The way to avoid this mistake is to create dummy variables. Each category is given a new column, and when it is present, it is encoded with a one, and when not present, it is given a zero. Unlike linear regression models, which work under the assumption of no multicollinearity and remove a dummy variable to avoid multicollinearity, random forest models are based on a decision tree structure. They can handle multicollinearity, so all variables are retained (Pramoditha, 2021).

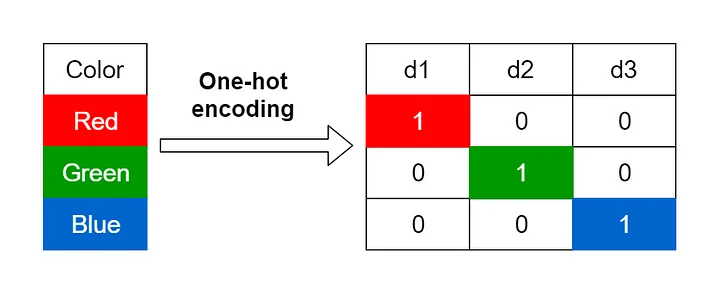


Figure 1: (Encoding Categorical Variables, 2021)

For this analysis, variables that were categorical and had more than two categories, the variable was one hot encoded. This was done using get\_dummies() from the pandas library. The drop\_first attribute has been set to False to retain all new variables.

df = pd.get\_dummies(df, columns = categorical\_cols, dtype = int, drop\_first = False)

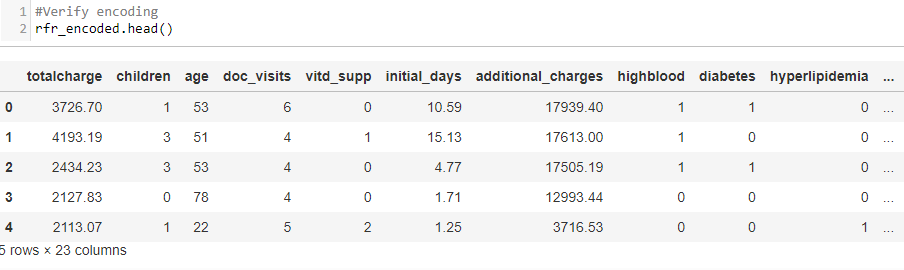


Figure 2: One Hot Encoded Variables

2. Another preprocessing task performed on the data was scaling. This allows variables on different measurement scales to be compared using the same comparison scale. The variable Additional\_charges is in dollars, while Initial\_ days is measured in days, making comparing the two difficult. A random forest model may give unequal weight to these variables, so the variables were placed on the same scale for a reduced chance of placing more weight on a more significant value during the decision process.

The data was scaled using StandardScaler() from sklearn.preprocessing library. To prevent biases within the data, also known as data leakage, the scaling was performed after the data was split into training and test data sets (Weiran, 2021). The target variable, y, is not scaled, as it must remain in a binary format for the model.

scale = StandardScaler()

X\_train\_scaled = pd.DataFrame(scale.fit\_transform(X\_train), columns=X\_train.columns)

X\_test\_scaled = pd.DataFrame(scale.transform(X\_test), columns=X\_test.columns)

The difference in the numerical values can be seen in a comparison before and after the scaling.

Figure 3: Before Scaling

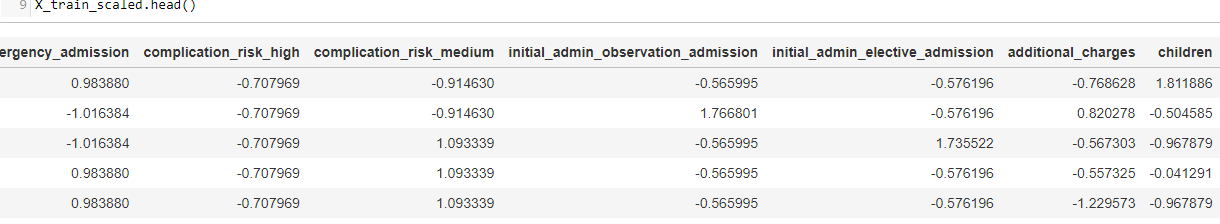


Figure 4: After StandardScaler()

C2: Variable Description

When creating a random forest model, all the data should be transformed into a numeric data type. The data set was examined and passed down through several cleaning steps. The chart below shows the data type of each variable in the initial data to determine if it would be selected for the final model. The final variables chosen for the analysis are highlighted in blue.

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Variable Type | Data Type | Selected final analysis |
| Population | Numeric | Discrete | No |
| Area | Categorical | Nominal | No |
| Timezone | Categorical | Nominal | No |
| Job | Categorical | Nominal | No |
| Children | Numeric | Discrete | Yes |
| Age | Numeric | Continuous | No |
| Education | Categorical | Nominal | No |
| Employment | Categorical | Nominal | No |
| Income | Numeric | Continuous | No |
| Marital | Categorical | Nominal | No |
| Gender | Categorical | Nominal | No |
| ReAdmis | Categorical | Nominal / Bivariate | No |
| VitD\_levels | Numeric | Continuous | No |
| Doc\_visits | Numeric | Discrete | No |
| Full\_meals\_eaten | Numeric | Discrete | No |
| VitD\_supp | Numeric | Discrete | No |
| Soft\_drink | Categorical | Nominal | No |
| Initial\_admin | Categorical | Nominal | Yes – Dummy variables  initial\_admin\_emergency\_admission  initial\_admin\_observation\_admission  initial\_admin\_elective\_admission |
| HighBlood | Categorical | Nominal / Bivariate | Yes |
| Stroke | Categorical | Nominal / Bivariate | No |
| Complication\_risk | Categorical | Nominal | Yes- Dummy Variables  complication\_risk\_high  complication\_risk\_medium |
| Overweight | Categorical | Nominal / Bivariate | No |
| Arthritis | Categorical | Nominal / Bivariate | No |
| Diabetes | Categorical | Nominal / Bivariate | No |
| Hyperlipidemia | Categorical | Nominal / Bivariate | No |
| BackPain | Categorical | Nominal / Bivariate | No |
| Anxiety | Categorical | Nominal / Bivariate | No |
| Allergic\_rhinitis | Categorical | Nominal / Bivariate | No |
| Reflux\_esophagitis | Categorical | Nominal / Bivariate | No |
| Asthma | Categorical | Nominal / Bivariate | No |
| Services | Categorical | Nominal | No |
| Initial\_days | Numeric | Continuous | Yes |
| TotalCharge | Numeric | Continuous | Target Variable |
| Additional\_charges | Numeric | Continuous | No |

C3: Analysis Preparation Steps

Often, the data for an analysis needs to be prepared through cleaning, exploration, and wrangling. These preprocessing steps ensure clean data that can create an accurate model.

The steps for this analysis included:

* + Duplicate removal
  + finding and treating missing values
  + Determining variables to be removed
  + identify outliers
  + Categorical re-expression
  + One- Hot encoding
  + Scaling of numerical data

1. Import and examine the data set- This step allows for familiarization with the data that will be worked with

med\_df = pd.read\_csv('medical\_clean.csv')

med\_df.info()

1. Find and remove duplicates – This step removes data that could skew the results if left in and counted more than once.

print(med\_df.duplicated().value\_counts())

print(med\_df.duplicated().sum())



1. Find Null values—This step finds values outside the data set. Many functions will not run if missing values are in the data set. Missing values are removed, but they are not the same as zero values, which should remain in the data.

med\_df.isnull().sum()

#Visualization of missing values- Heatmap

plt.figure(figsize=(12, 8))

sns.heatmap(med\_df.isnull(), cbar=False, cmap='viridis', yticklabels=False, xticklabels=med\_df.columns)

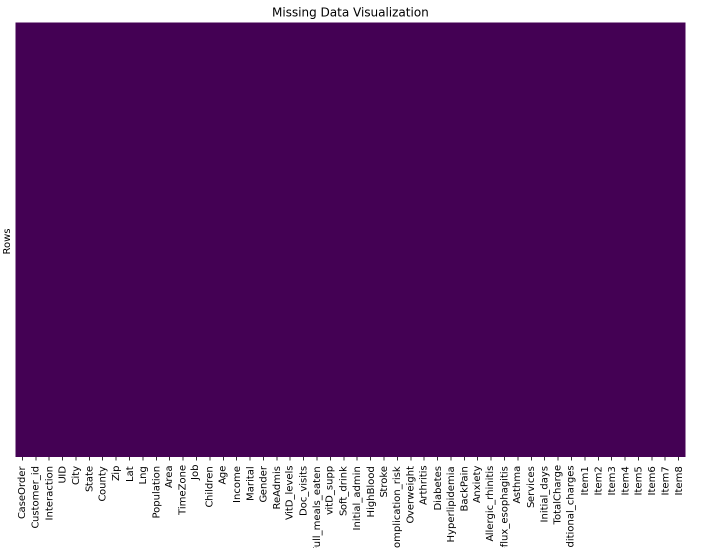
plt.xticks(rotation=90)

plt.title('Missing Data Visualization')

plt.xlabel('Columns')

plt.ylabel('Rows')

plt.show()



1. Remove null values—Null values are removed in the cleaning process as many functions can not be correctly calculated without the information.

No null values were found. The data set was then renamed as a checkpoint in which duplicates and null values were addressed

clean\_df = med\_df.copy()

1. Remove columns with high cardinality – This step removes columns that will not be used due to being an identifier or the volume of variables being unable to be used to build a desired prediction model.

clean\_df.describe()

# Clean the data set columns using created function based on thresholds (Bold Analytics: Mark Keith, 2024)

#missing\_threshold = 0.95, unique threshold = 0.95, only 1 value in a column --> removes

#Function

def clean\_columns(df, columns =[], missing threshold = 0.95, unique threshold = 0.95, messages = True):

if len(columns) == 0:

columns = df.columns #this lets the columns be blank, and every column will be cleaned

for col in columns:

if col in df.columns:

missing = df[col].isna().sum()

unique = df[col].nunique()

rows = df.shape[0]

if missing / rows >= missing\_threshold:

if messages: print(f"To many missing values with ({missing} out of {rows}, {round((missing / rows) \* 100, 2)}%) for {col}, removed")

df.drop(columns =[col], inplace = True)

# For non-numeric columns, check if there are too many unique values

if not pd.api.types.is\_numeric\_dtype(df[col]) and (unique / rows >= unique threshold):

if messages:

print(f"Too many unique values with ({unique} out of {rows}, {round((unique / rows) \* 100, 2)}%) for {col}, removed")

df.drop(columns=[col], inplace=True)

continue

elif unique == 1:

if messages: print(f"Only one value in ({df[col].unique()[0]} for {col}, removed")

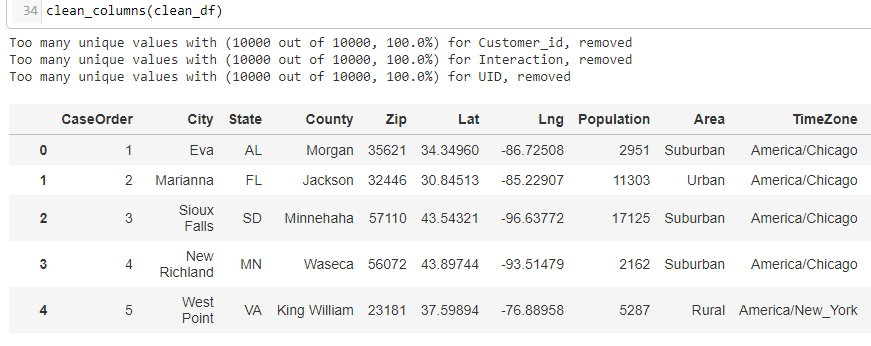
df.drop(columns =[col], inplace = True)

else:

if messages: print(f"The column variable \"{col}\" doesnt exist as spelled in the DataFrame provided")

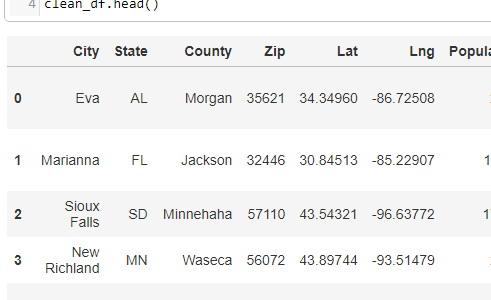
return df

clean\_columns(clean\_df)



# Remove the index column

clean\_df = clean\_df.iloc[:, 1:]



1. Explore summary statistics and univariate visualizations—This step allows for familiarization with the data statistics to determine the distribution and the need to deal with outliers. Many models are sensitive to outliers and must be cleaned to help reduce errors within a model.

#Function for visualizing univariate variables and summary statistics (Bold Analytics: Mark Keith, 2024)

def univariate(df):

stats = []

for col in df.columns:

col\_data = df[col]

dtype = col\_data.dtype

count = col\_data.count()

missing = col\_data.isna().sum()

unique = col\_data.nunique()

mode = col\_data.mode().iloc[0] if not col\_data.mode().empty else None

if pd.api.types.is\_numeric\_dtype(col\_data):

# Compute stats for numeric columns

min\_val = col\_data.min()

q1 = col\_data.quantile(.25)

median = col\_data.median()

q3 = col\_data.quantile(.75)

max\_val = col\_data.max()

mean = col\_data.mean()

std = col\_data.std()

skew = col\_data.skew()

kurt = col\_data.kurt()

stats.append((col, dtype, count, missing, unique, mode, min\_val, q1, median, q3, max\_val, mean, std, skew, kurt))

# Plot histogram for numeric columns

sns.histplot(data=col\_data.dropna(), kde=True)

plt.title(f'Histogram of {col}')

plt.show()

else:

# Stats for non-numeric columns

stats.append((col, dtype, count, missing, unique, mode, "-", "-", "-", "-", "-", "-", "-", "-", "-"))

# Plot countplot for categorical columns

sns.countplot(x=col\_data.dropna(), data=df)

plt.title(f'Count Plot of {col}')

plt.show()

# Create DataFrame from stats list

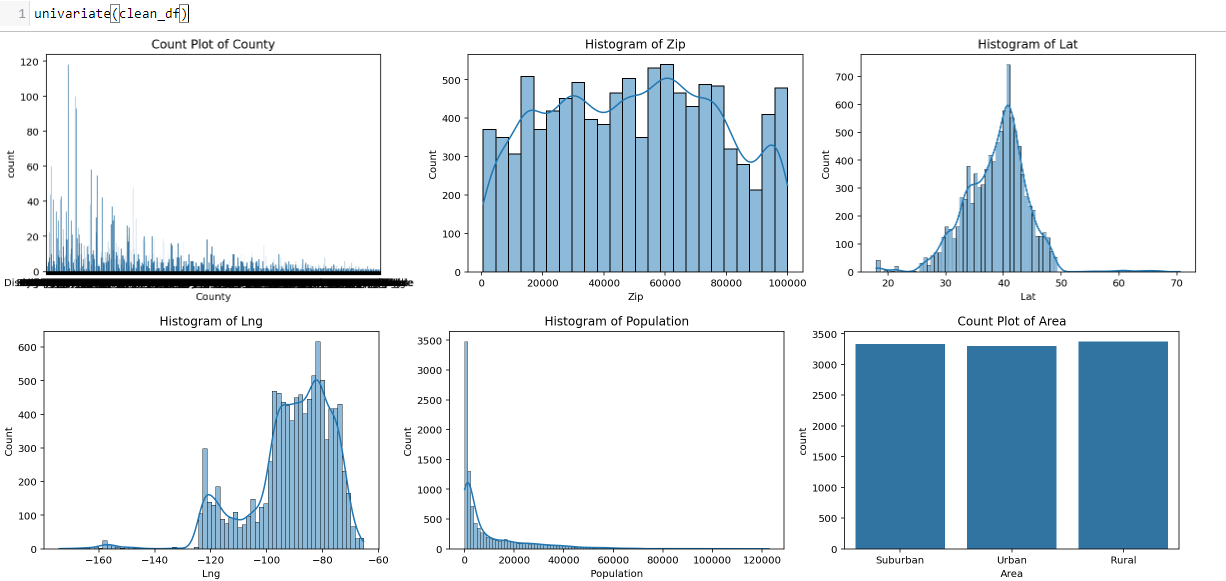
output\_df = pd.DataFrame(stats, columns=["Variable", "Type", "Count", "Missing", "Unique", "Mode",

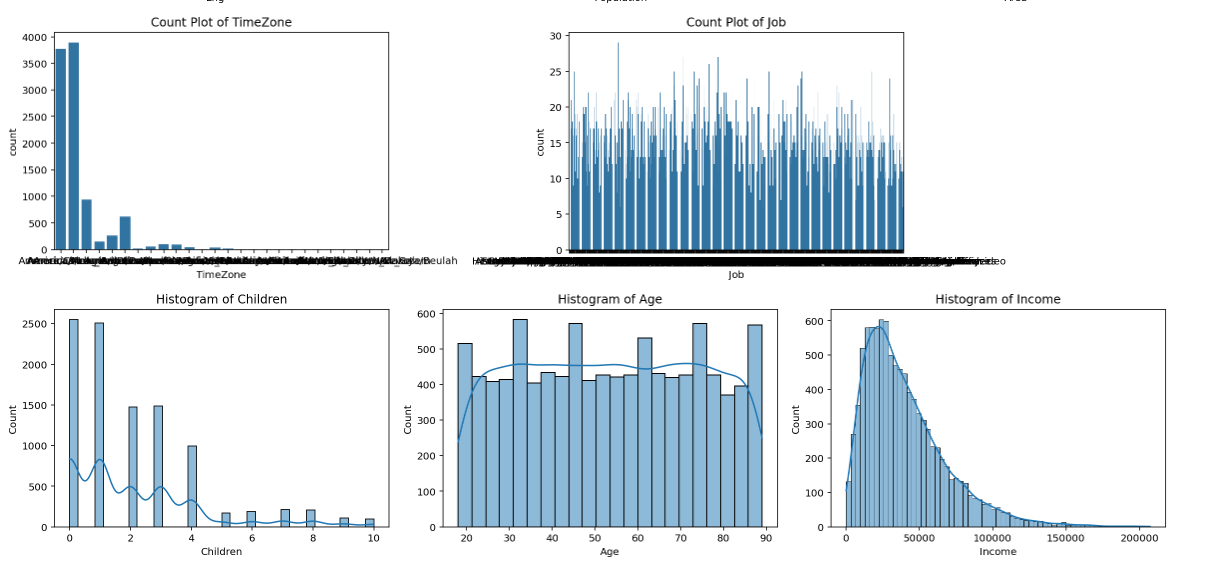
"Min", "Q1", "Median", "Q3", "Max", "Mean", "Std", "Skew", "Kurt"])

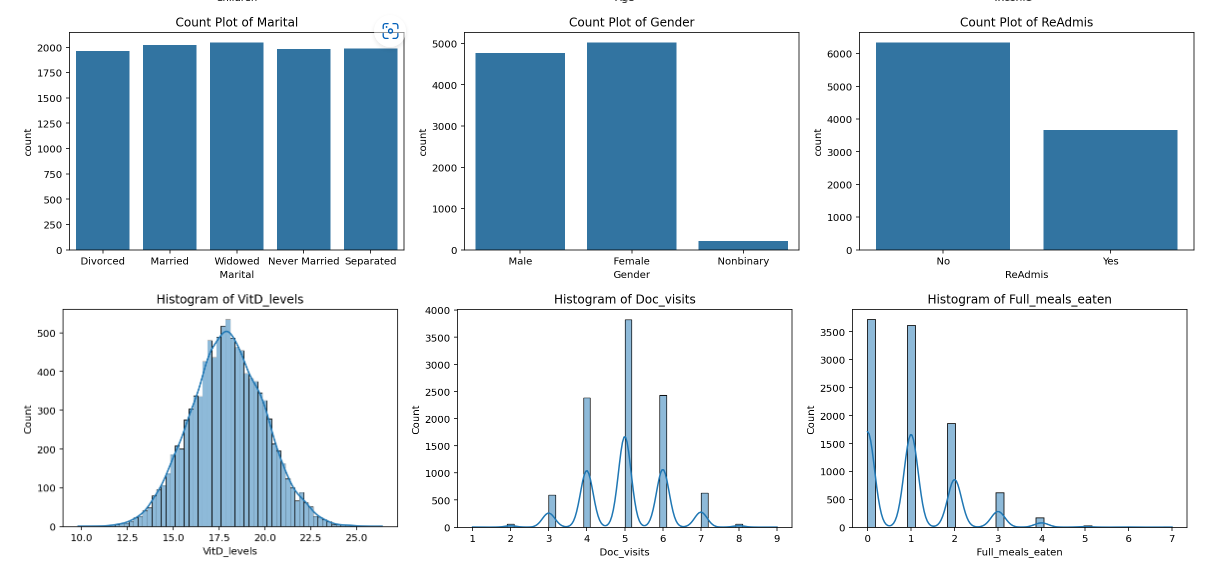
output\_df.set\_index("Variable", inplace=True)

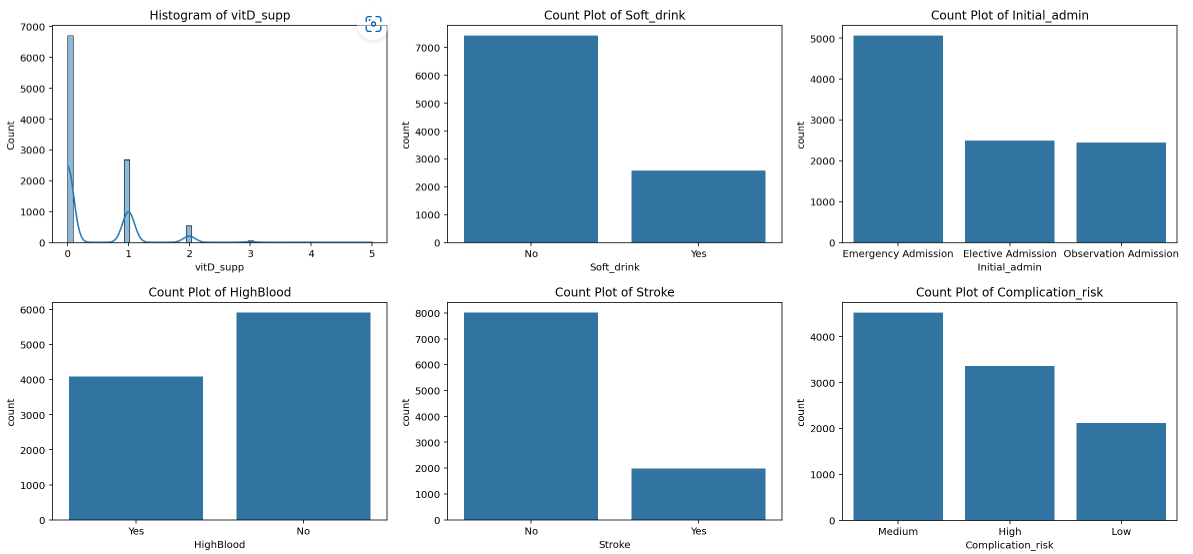
return output\_df

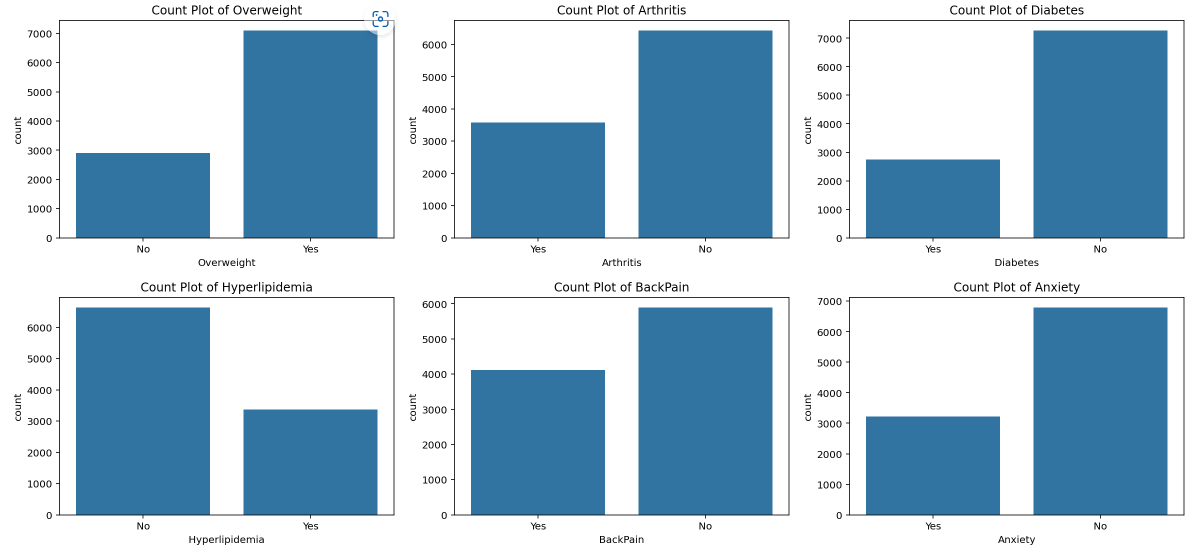
univariate(clean\_df)

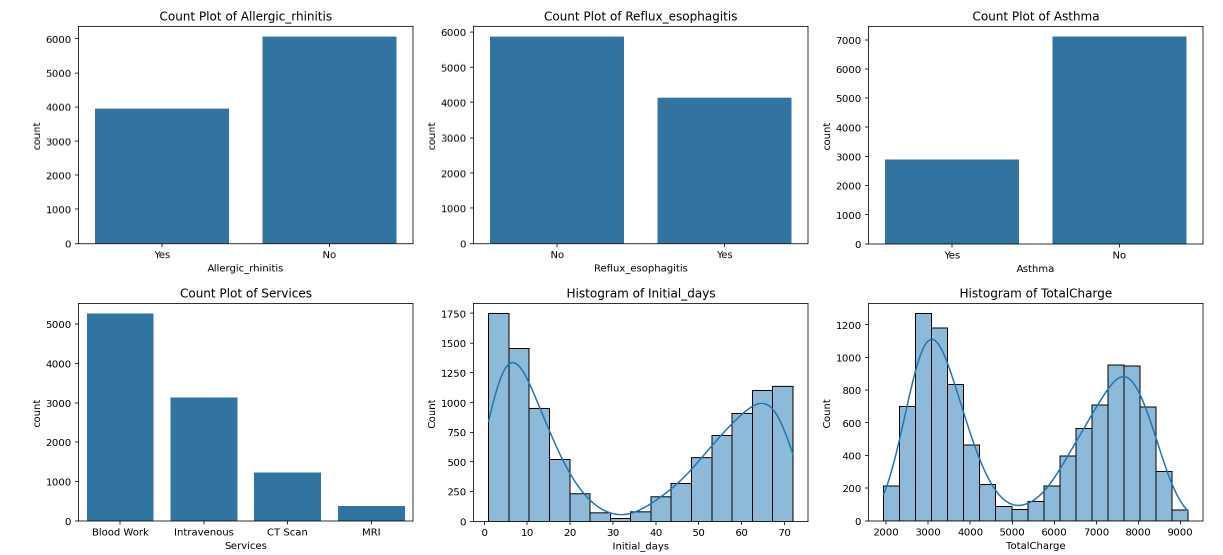


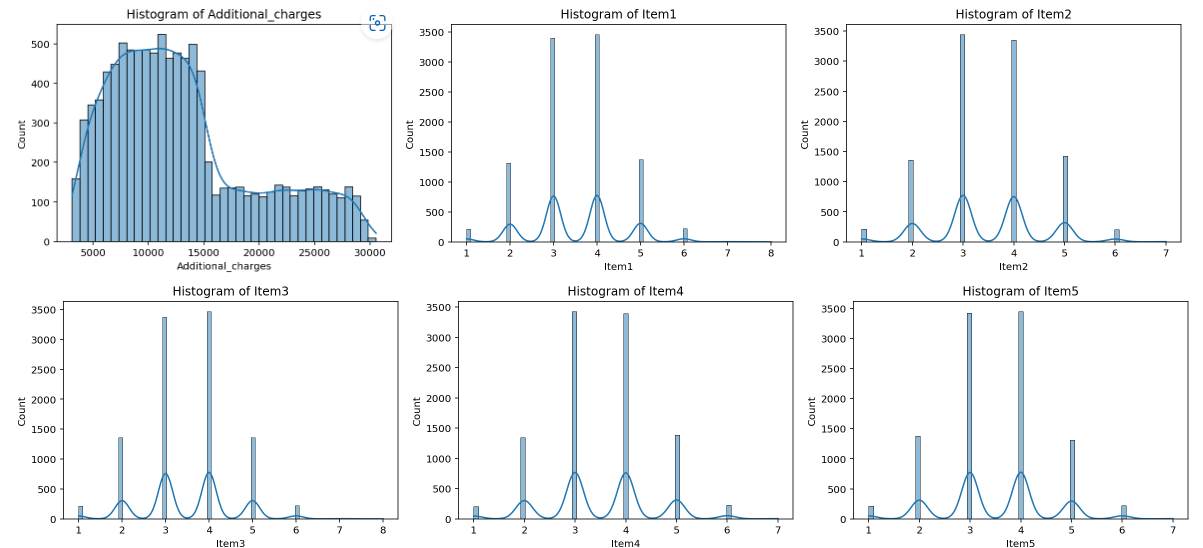


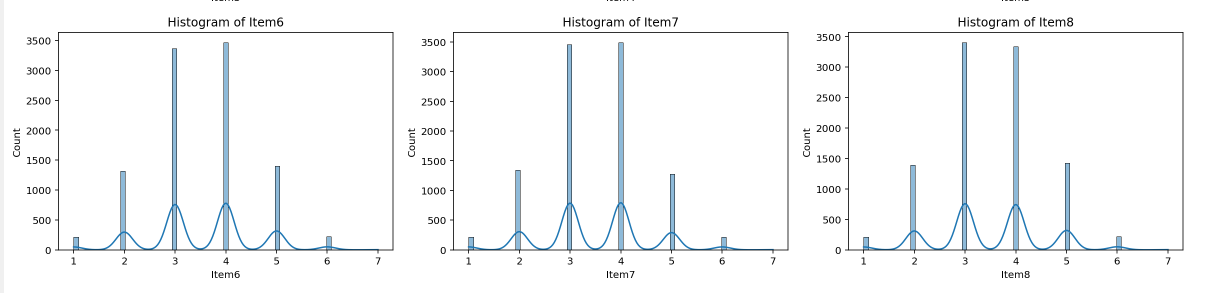


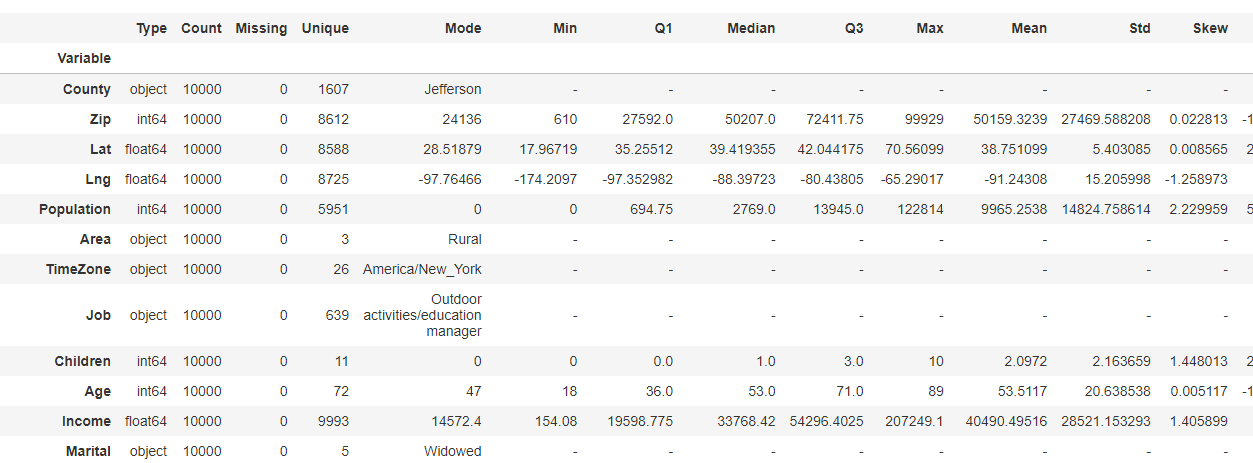












1. Narrow down variables for feature selection—Understanding the subject matter can help reduce the number of features needed for calculations when building models. Not all features will be relevant for every analysis, nor can all features be used.

Columns with location data and all the survey columns were removed because they are subjective measurements. ReAdmis was removed as the data will be evaluated in the future.

rfr\_data = clean\_df[[

"TotalCharge", "Population", "Children", "Age", "Income",

"VitD\_levels", "Doc\_visits", "Full\_meals\_eaten", "vitD\_supp",

"Initial\_days", "Additional\_charges", "Gender", "Initial\_admin",

"Complication\_risk", "HighBlood", "Diabetes", "Hyperlipidemia", "Services"

]].copy()

1. Clean white spaces and round decimals—Cleaning data, such as white spaces and rounding decimals, helps with readability and prevents errors by creating uniformity within the data.

## Remove spaces in category names within columns

# To replace spaces with underscores in specific columns

for col in rfr\_data.columns:

if not pd.api.types.is\_numeric\_dtype(rfr\_data[col]):

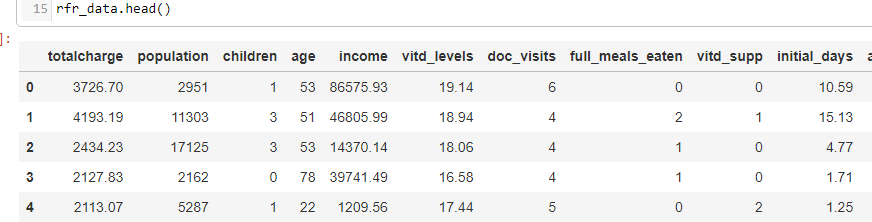
rfr\_data[col] = rfr\_data[col].str.replace(" ", "\_", regex = False)

##Clean White Space and make all lower case

rfr\_data.columns = rfr\_data.columns.str.lower().str.strip()

#round values to 2 decimal points

rfr\_data = rfr\_data.round(2)



1. Detect outliers (Horsch, 2021)—Outliers must be detected and evaluated to determine whether they are expected or bad data. It also needs to be evaluated whether the outliers will be tolerated or need to be treated.

#Function for Winsorization of outliers, using IQR

def winz\_outliers(df):

print("Outlier Analysis Report")

print("=" \* 50) # Print a separator line for visual clarity

for col in df:

if pd.api.types.is\_numeric\_dtype(df[col]):

Q1 = df[col].quantile(0.25)

Q3 = df[col].quantile(0.75)

IQR = Q3 - Q1

outliers = ((df[col] < (Q1 - 1.5 \* IQR)) | (df[col] > (Q3 + 1.5 \* IQR)))

outlier\_count = outliers.sum()

if outlier\_count > 0:

outer\_fence = 3 \* IQR

outer\_fence\_low = Q1 - outer\_fence

outer\_fence\_up = Q3 + outer\_fence

# Consolidating the print statements for each column

print(f"\nColumn: {col}")

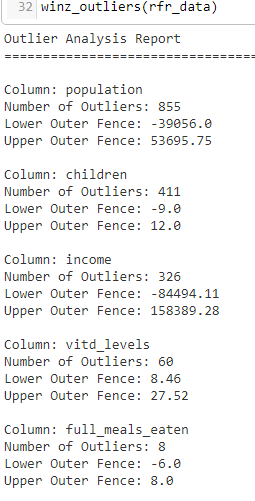
print(f"Number of Outliers: {outlier\_count}")

print(f"Lower Outer Fence: {round(outer\_fence\_low, 2)}")

print(f"Upper Outer Fence: {round(outer\_fence\_up, 2)}")

print("=" \* 50)

winz\_outliers(rfr\_data)



# Create a boxplot for each column with outliers

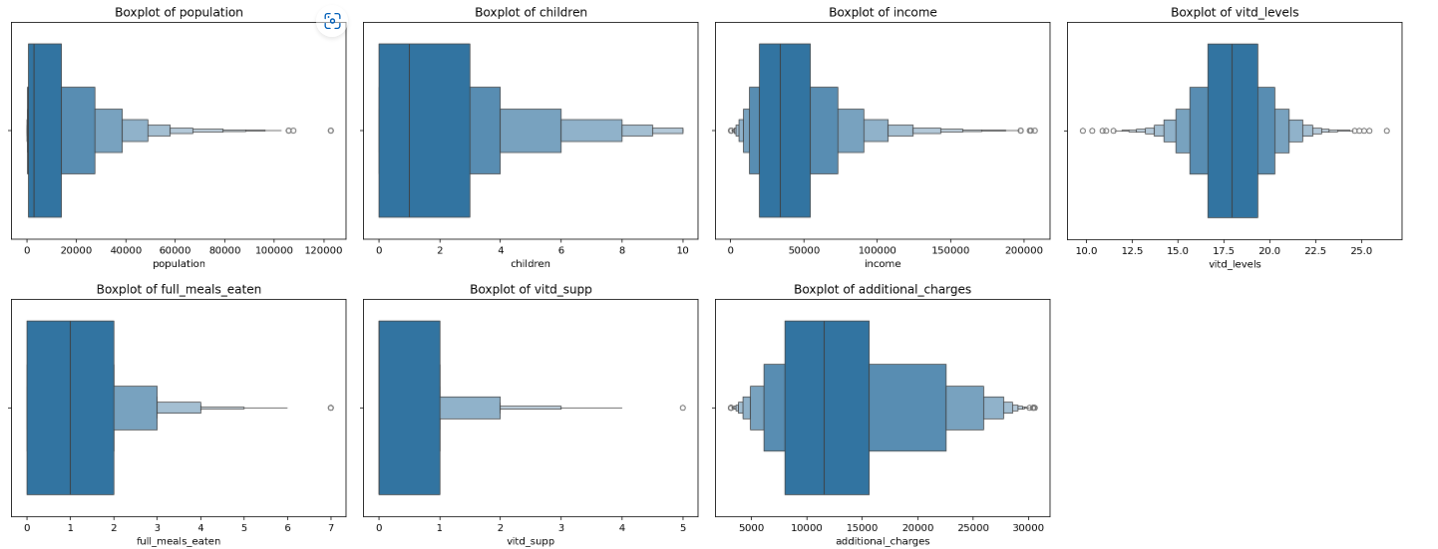
outlier\_shape = rfr\_data[["population", "children","income", "vitd\_levels", "full\_meals\_eaten", "vitd\_supp", "additional\_charges"]]

for column in outlier\_shape:

sns.boxenplot(x=rfr\_data[column])

plt.title(f'Boxplot of {column}')

plt.show()



1. Treat outliers—Since outliers can change the results in regression model estimates, it is necessary to address whether outliers will be treated. The method used for this analysis is winsorization, which works well when a variable's distribution is not a standard bell curve.

Several Outliers were expected and left unchanged. These variables were children, vitd\_supp, and additional\_charges.

## Winsorize the outliers

# Variables to winsorize - Population, income, VitD\_levels, Full\_meals\_eaten

#Note that VitD levels will also need a lower limit calculation

rfr\_data['population\_winz'] = winsorize(rfr\_data['population'], limits=(0, 0.05))

rfr\_data['income\_winz'] = winsorize(rfr\_data['income'], limits=(0, 0.05))

rfr\_data['vitd\_levels\_winz'] = winsorize(rfr\_data['vitd\_levels'], limits=(0.05, 0.01))

rfr\_data['full\_meals\_winz'] = winsorize(rfr\_data['full\_meals\_eaten'], limits=(0, 0.05))

#Remake data set with the outliers removed

rfr\_data = rfr\_data.drop(columns=["population", "income", "vitd\_levels", " full\_meals\_eaten"], axis = 1)

1. Variable re-expression and one hot encoding—Bivariate categorical variables, including the target variable of Readmis, were changed to 0/1 values. Categorical variables with more than two categories were hot encoded to dummy variables. No dummy variables were removed since multicollinearity is not a concern when running a random forest regression analysis.

## Function for changing binary values to 0/1

# and creating dummy var with all variables kept in place

def wrangle\_cat(df):

#handle binary categorical variables

for col in df.columns:

if pd.api.types.is\_string\_dtype(df[col]):

# standardize the text format to lowercase

df[col] = df[col].astype(str).str.strip().str.lower()

# find binary columns with yes/ no or true/false

if df[col].isin(['yes', 'no', 'true', 'false']).all():

mapping = {'yes': 1, 'no': 0, 'true': 1, 'false': 0}

df[col] = df[col].map(mapping)

# one-hot encode the remaining categorical variables

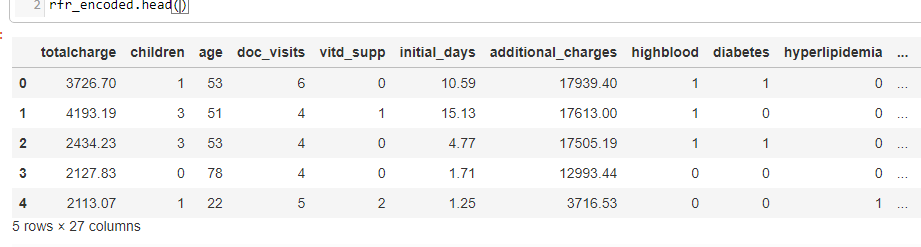
# Ensure to only encode those that have not been converted to numeric in the previous step

categorical\_cols = df.columns[df.dtypes == 'object']

df = pd.get\_dummies(df, columns = categorical\_cols, dtype = int, drop\_first = False)

return df

rfr\_encoded = wrangle\_cat(rfr\_data)



1. Feature selection - The method used for feature selection was SelectKBest. The results were reduced to a final data set for use in the random forest prediction model.

SelectKBest feature selection (D, 2023) – The p-value assigned for significance was 0.05.

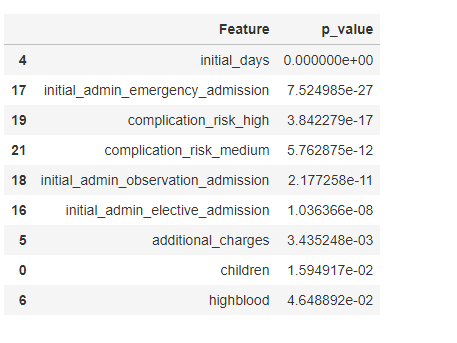
skbest = SelectKBest(score\_func = f\_classif, k ='all')

X\_new = skbest.fit\_transform(X, y)

X\_new.shape

p\_values = pd.DataFrame({"Feature": X.columns, "p\_value":skbest.pvalues\_}).sort\_values("p\_value")

p\_values[p\_values["p\_value"] < .05]



1. Data set for random forest model—The features selected for the model were totalcharge, initial\_days, children, initial\_admin\_emergency\_admission, initial\_admin\_observation\_admission, initial\_admin\_elective\_admission, complication\_risk\_high, Compications\_risk\_medium additional\_charges, and highblood.

rfr\_features = rfr\_encoded[[

"totalcharge", "initial\_days", "initial\_admin\_emergency\_admission",

"complication\_risk\_high", "complication\_risk\_medium",

"initial\_admin\_observation\_admission", "initial\_admin\_elective\_admission",

"additional\_charges", "children", "highblood"

]].copy()

C4: Clean Data

A file of the cleaned data set is attached as a .csv file labeled Forest\_features\_209.csv.

**Part IV**

D :

D1: Split Data

The data was split into training and test data, which were scaled after the split to prevent data leakage. The training data was split and contained 80% of the data, and the test data contained 20% (test\_size = 0.2). The data was also stratified (stratify = y ) to ensure that each category was represented in the training and test data in the same proportion.

X = rfr\_features.drop(["totalcharge"], axis=1)

y = rfr\_features["totalcharge"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y,

test\_size=0.2, # Specifies 20% of the data for testing

random\_state=42, # Ensures reproducibility

stratify=y) # Stratifies the split based on the labels in 'y'

Copies of the split and scaled data set are attached as .csv files labeled:

* Xscale\_train\_209.csv
* Xscale\_test\_209.csv
* Y\_train\_209.csv
* Y\_test\_209.csv

#Export testing and training files

# file path is within jupyter Lab project

X\_train\_scaled.to\_csv("Xscale\_train\_209.csv", index=False)

X\_test\_scaled.to\_csv("Xscale\_test\_209.csv", index = False)

y\_train.to\_csv("Y\_train\_209.csv", index = False)

y\_test.to\_csv("Y\_test\_209.csv", index = False)

D2: Analysis and Calculations

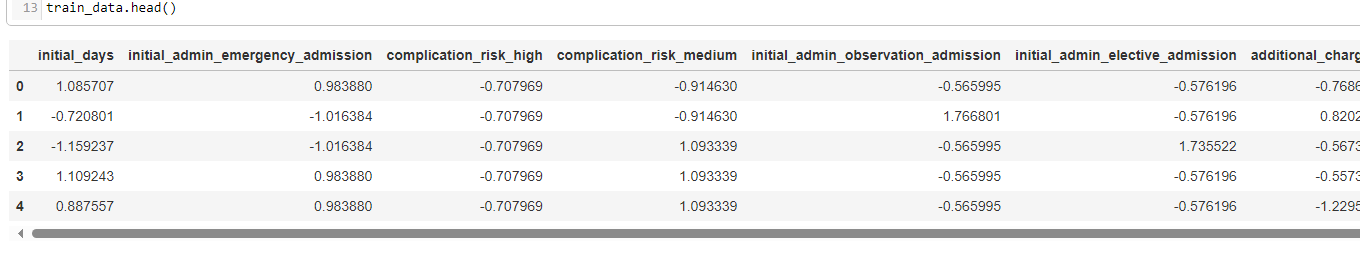
The method used to analyze the data is a random forest regression model, which uses supervised learning to make predictions based on creating many decision trees. Random forest regression can be used for numerical target variables, making it a strong choice since TotalCharge is a continuous numerical variable.

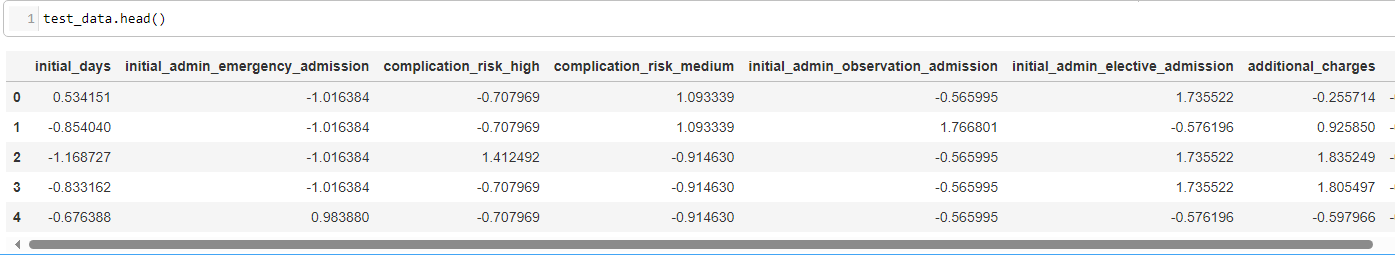
The random forest model is based upon the premise of decision trees, which take the data and divide it using predictor values and build a model based on if-then decisions to build smaller and smaller nodes until it ends at a final node, known as a leaf, which contains primarily similar variables. The nodes' purity, or equivalency, is measured by how much the total variance decreases with each split (Bruce, et al., 2019). The random forest model improves upon this idea by creating multiple tree models built using a random choice of variables. The results are then combined to find the average value for the final predicted outcome. One strength of this type of modeling is that it can prevent the model from being overfit.

This analysis entailed several steps, including scaling the numeric data, running random forest regression analysis, and then hyperparameter tuning. No intermediate calculations were performed during the analysis. All code for the following analysis descriptions can be found in a subsequent section of D3.

1. The numeric data was standardized after the split to prevent data leakage, which occurs when the model is exposed to information outside the designated training data (Brownlee, 2020). The standardization of the features is solely based on the training data.

When there are large distances between variables due to being on a larger scale, those measurements will dominate. The data was standardized using the scikit-learn method StandardScaler(). This allows numeric data on different measurement scales to be compared with one another (Hale, 2020) without the weight of large numbers on separate scales.





2. The random forest model was trained using the scaled training data set, and the model performance was analyzed using the test data. The n\_estimator value is the number of trees the model will build. The initial number of estimators was set at 100, the default value in sklearn(), which would be tuned in the following steps (Pedregosa et al., 2011).

The accuracy of a regression model can be evaluated using several different metrics. This project looked at four metrics of an initial model and then compared those to the same metrics of the tuned model. These metrics included mean absolute error (MAE), mean squared error (ASE), root mean squared error (RMSE), and R squared ( R2 ).

A screenshot of a computer error

Description automatically generated

MAE is the absolute measurement of the difference between predicted and actual values. It is measured in the same units as the predicted output and does well with comparative model evaluation. The initial model's MAE was $98.18.

MSE is the squared difference between the predicted and the actual values. A lower value indicates a better fit, but because it is the squared value of the target variable, it can be hard to interpret directly. The initial model's MSE score is 15721.45.

RMSE can help solve the problem of the error's interpretability. It takes the square root of the MSE value. The RMSE for this initial model was $125.39, meaning that the predictions were an average of $125 off from the actual value.

The R2 value is a metric on a 0 to 1 scale that explains how much of the variance in the target variable can be attributed to the predictors in the model. A score closer to 1 is better. The score for the initial model was .99, meaning that 99% of the variance of the average daily costs can be attributed to the predictors within the model (Bobbit, 2021).

3. Hyperparameter tuning was done on the model in an attempt to tune the fit of the model. The scikit\_learn library has a module GridSearchCV() that folds the training data and adjusts the hyperparameters. This entails splitting the training data into "folds" and then retesting that folded data with different variations of the hyperparameters.

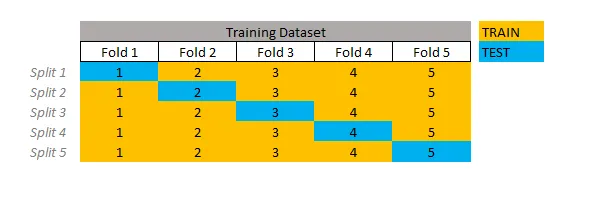


Figure 5: (Santos, 2020)

Three folds were done for this tuning, and several parameters and parameter values were tested to achieve the optimal values.

A screenshot of a computer code

Description automatically generated



Once the best new values were found, the model was re-run with the new value for the final model.

A screenshot of a computer error

Description automatically generated

The tuned random forest model has improvements in all scoring metrics.

D3: Code

A jupyter notebook file of all the code used in the analysis is attached as PBier\_209\_pt2.ipynb .

1. Standardization of the data set after spit 80% training and 20% test.

# Standardize the values using scale

scale = StandardScaler()

X\_train\_scaled = pd.DataFrame(scale.fit\_transform(X\_train), columns=X\_train.columns)

X\_test\_scaled = pd.DataFrame(scale.transform(X\_test), columns=X\_test.columns)

# Concatenate scaled features with the target variable

train\_data = pd.concat([X\_train\_scaled, y\_train.reset\_index(drop=True)], axis=1)

test\_data = pd.concat([X\_test\_scaled, y\_test.reset\_index(drop=True)], axis=1)

2. Random Forest training and testing. The initial n\_estimators was set to 100.

# Initialize and build Random forest model

rfr = RandomForestRegressor(

n\_estimators=100, # More trees for better generalization

max\_features="sqrt", # Square root of the number of features for a regression task

random\_state=42

)

rfr.fit(X\_train\_scaled, y\_train)

y\_pred = rfr.predict(X\_test\_scaled)

A screenshot of a computer error

Description automatically generated

3. Hyperparameter testing

a. Build dictionary of parameter grid values to test

# Create a parameter grid

param\_grid = {

'n\_estimators': [100, 200, 300],

'max\_features': ['log2', 'sqrt'],

'max\_depth': [None, 10, 20, 30],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

b. GridSearchCV() hyperparameter testing

# Set up the grid search

grid\_search = GridSearchCV(estimator = rfr,

param\_grid = param\_grid,

cv = 3,

n\_jobs = -1,

verbose = 1,

scoring= 'neg\_mean\_squared\_error')

# Fit grid search

grid\_search.fit(X\_train, y\_train)

# Best parameters and best score

print("Best parameters:", grid\_search.best\_params\_)

print("Best score:", -grid\_search.best\_score\_)

**Part V**

E :

E1: MSE and Accuracy

In regression models, accuracy is determined as a whole, not as a specific "Accuracy" measurement but as a specific calculation done in classification models. Regression models are evaluated based on measured prediction errors (Brownlee, 2023). The final model was then evaluated for overall accuracy using four metrics to determine if there was an improvement in the values, which in turn would mean a better-fitting model.

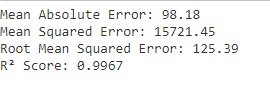


Figure 6 First model

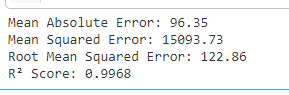


Figure 7 Final Tuned Model

The MAE was used to determine the optimal parameters when performing the hyperparameter testing. It is the absolute measurement of the difference between predicted and actual values.

MAE = 1/n \* Σ|yi – ŷi|

A lower value is a better indication of the model's fit. It is measured in the same units as the predicted output and does well with comparative model evaluation. The initial model's MAE was $98.18, and the tuned model value is lower at $96.35. This is an improvement in MAE of approximately 1.9%.

MSE is the squared difference between the predicted and the actual values.

MSE = Σ(ŷi – yi)2 / n

Since this value is squared, it can be more sensitive to outliers. A lower value indicates a better fit, but because it is the squared value of the target variable, it can be hard to interpret directly. It also works well when used to compare models. The MSE score of the initial model was 15721.45, and the tuned model was 1593.73. This lower value shows that the model has improved by approximately 4.2%.

Based on these two metrics, the final tuned model can be considered accurate and reliable when making average daily costs (TotalCost) predictions.

E2: Results

Several other metrics were used in the evaluation: the RMSE and the R2 score.

RMSE can help solve the problem of the error's interpretability. It takes the square root of the MSE value.

RMSE = √Σ(ŷi – yi)2 / n

This converts the value back to the scale of the target variable and allows for a measurement of how far off the average predictions are from the actual values in the scale of the target variable. The RMSE for this initial model was $125.39, and the new fitted model was $122.86. This means the tuned model's predictions are now $122 off the average daily costs, an approximately 2% improvement.

The R2 value is a metric on a 0 to 1 scale that explains how much of the variance in the target variable can be attributed to the predictors in the model.

R2 = 1 – (RSS/TSS)

A score closer to 1 is better. The score for the initial model was .9967, and the tuned model was only slightly better at .9968. The score was already exceptionally high, indicating that the model variables can explain 99% of the variance of the target variable.

The full results of this analysis have high values for all the metrics used in evaluating the model. This allows for solid confidence in predicting a patient's total average cost during their hospital visit. The implications of a model with high evaluation metrics mean that when imputing new data for evaluation, the results can also be used with high confidence, especially if actionable steps are being taken based on the results.

E3: Limits

While the metrics imply high performance of this random forest regression model, some limitations should be considered. Random forest models do not do well with variables with high cardinality (Tongtian, 2020). This can lead to an error of omission with confounding variables, meaning that variables that would help lead to important conclusions were left out due to the high cardinality (Bruce et al., 2020, p. 172). Another shortcoming of these models is that there is little control over how the decisions are made within the model, like a black box model, making it hard to fully understand how the results were achieved (Singh, 2023).

E4: Recommendations –

Based on the results of this model, the recommendation is to create a dashboard to segment out which variables have the most impact on the total daily charge. These factors can be examined to determine whether cost-cutting measures can be applied. Suppose the variable can be linked to specialization. In that case, education can then be targeted to specific departments rather than general education to a more significant number of employees, which could lead to savings within the hospital and improved outcomes as specialization increases. The results from the model and dashboard will take longer to implement as trends will have to be further investigated before significant actions are implemented.

**Part VI**

F : Panopto

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=92eaec8a-147d-4692-9b7f-b1650010cd3e>

G: Web sources

Allen, J., MD (2023, January 4). *Understanding The 2023 Medicare Hospital Readmission Penalty*. Hospital Medical Director. Retrieved February 29, 2024, from <https://hospitalmedicaldirector.com/understanding-the-2023-medicare-hospital-readmission-penalty/>

Bobbit, Z. (2021, June 22). *RMSE vs. R-Squared: Which Metric Should You Use?* Statology. Retrieved April 29, 2024, from https://www.statology.org/rmse-vs-r-squared/

[Bold Analytics : Mark Keith]. (2024, January 29). *Python Data Science: Automating EDA: Univariate Statistics and Visualizations* [Video]. YouTube. <https://www.youtube.com/watch?v=eloR5Li0Huo&list=PLe9UEU4oeAuX3GZgRKzUlLcHd5SpYzZGW&index=1>

[Bold Analytics : Mark Keith]. (2024, February 5). *Python Data Science: Automating Cleaning: Data Wrangling: Empty, Single Value, Primary Key columns* [Video]. YouTube. https://www.youtube.com/watch?v=8IsxTImOhF0&list=PLe9UEU4oeAuX3GZgRKzUlLcHd5SpYzZGW&index=8

Brownlee, J. (2020, August 15). *Data Leakage in Machine Learning*. Machine Learning Mastery. Retrieved April 15, 2024, from <https://machinelearningmastery.com/data-leakage-machine-learning/>

Brownlee, J. (2023, February 16). *Regression Metrics for Machine Learning*. Machine Learning Mastery. Retrieved April 26, 2024, from <https://machinelearningmastery.com/regression-metrics-for-machine-learning/#:~:text=Accuracy%20(e.g.%20classification%20accuracy)%20is,if%20you%20think%20about%20it>.

D, K. (2023, February 15). *Optimizing Performance: SelectKBest for Efficient Feature Selection in Machine Learning*. Retrieved April 15, 2024, from <https://medium.com/@Kavya2099/optimizing-performance-selectkbest-for-efficient-feature-selection-in-machine-learning-3b635905ed48>

Haymond, S., PhD, DABCC, FADLM, & Master, S., MD, PhD, FADLM (2022, April 1). Why Clinical Laboratorians Should Embrace the R Programming Language A Case for Learning R as a Gateway to Laboratory Medicine’s Digital Future. AACC.org. [https://www.aacc.org/cln/articles/2020/april/why-clinical-laboratorians-should-embrace-the-r-programming-language#](https://www.aacc.org/cln/articles/2020/april/why-clinical-laboratorians-should-embrace-the-r-programming-language)

Horsch, A. (2021, February 14). *Detecting and Treating Outliers In Python — Part 3 Hands-On Tutorial On Treating Outliers — Winsorizing and Imputation*. Towards Data Science. Retrieved April 15, 2024, from <https://towardsdatascience.com/detecting-and-treating-outliers-in-python-part-3-dcb54abaf7b0>

Kumar, D. (2018, December 25). *Introduction to Data Preprocessing in Machine Learning*. Towards Data Science. Retrieved April 15, 2024, from <https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d>

Middleton, K. (2023, July 13). D208 - Webinar: Getting Started with D208 Part II [Lecture]. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=39bbe2db-de7d-4bf5-913b-af5c0003da9d>

Pedregosa, F., Varoquaux, G., & Michel, V. (n.d.). *Sklearn.Ensemble.RandomForestRegressor*. Sci-Kit Learn. Retrieved April 29, 2024, from <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>

Pramoditha, R. (2021). *Encoding Categorical Variables* [Photograph]. Medium : Towards Data Science. <https://towardsdatascience.com/encoding-categorical-variables-one-hot-vs-dummy-encoding-6d5b9c46e2db>

Pramoditha, R. (2021, December 16). *Encoding Categorical Variables: One-hot vs Dummy Encoding Implementation with Pandas and Scikit-learn*. Towards Data Science. Retrieved April 15, 2024, from <https://towardsdatascience.com/encoding-categorical-variables-one-hot-vs-dummy-encoding-6d5b9c46e2db>

Sahai, N. (2023, September 6). *Random Forest Regression – How it Helps in Predictive Analytics?* Retrieved April 29, 2024, from <https://www.analytixlabs.co.in/blog/random-forest-regression/#How_do_you_use_random_forest_for_regression_problems>

Santos, G. (2021, August 18). *How to do Cross-Validation, KFold and Grid Search in Python*. Toward Data Science. Retrieved April 15, 2024, from <https://medium.com/towarhttps://medium.com/gustavorsantos/how-to-do-cross-validation-kfold-and-grid-search-in-python-e570cdb20a28ds-data-science/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb>

Singh, J. (2020, December 17). *Random Forests: Pros and Cons*. DataDrivenInvestor. Retrieved April 29, 2024, from https://medium.datadriveninvestor.com/random-forest-pros-and-cons-c1c42fb64f04

Soetewey, A. (2019, December 30). Variable types and examples. Stats and R. Retrieved August 26, 2023, from <https://statsandr.com/blog/variable-types-and-examples/>

Weiran, S. (2019, October 22). *Avoid Data Leakage — Split Your Data Before Processing*. Towards Data Science. Retrieved April 15, 2024, from <https://towardshttps://medium.com/towards-data-science/avoid-data-leakage-split-your-data-before-processing-a7f172632b00datascience.com/encoding-categorical-variables-one-hot-vs-dummy-encoding-6d5b9c46e2db>

[WGU - D206 Data Cleaning]. Middleton, K. (2023, August 1). Getting Started with D206 | Missing Values [Video]. WGU Panopto. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=767749d2-ba19-4f94-bec8-b058017b2f5e>

[WGU - D206 Data Cleaning]. Middleton, K. (2023, August 1). Getting Started with D206 | Principal Component Analysis (PCA) [Video]. WGU Panopto. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=3bcc452f-fa35-43be-b69f-b05901356f95>

Vishalmendekarhere (2021, July 11). It’s all about Assumptions, Pros & Cons. Medium. Retrieved April 29, 2024, from <https://medium.com/swlh/its-all-about-assumptions-pros-cons-497783cfed2d>

I: Sources

Anscombe, F. J. (1973). Graphs in Statistical Analysis. The American Statistician, 27(1), 17–21. <https://doi.org/10.2307/2682899>

Bishop PA, Herron RL. Use and Misuse of the Likert Item Responses and Other Ordinal Measures. Int J Exerc Sci. 2015 Jul 1;8(3):297-302. PMID: 27182418; PMCID: PMC4833473

Bruce, P., Bruce, A., & Gedeck, P. (2019). Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python (2nd ed.). O'Reilly Media.

Larouse, D. T. (2015). Data Mining and Predictive Analytics (p. 20). John Wiley & Sons, Incorporated. <http://ebookcentral.proquest.com/lib/westerngovernors-ebooks/detail.action?docID=7104155>

Monaghan, T. F., Rahmen, S. N., Agudelo, C. W., Wein, A. J., Lazar, J. M., Everaert, K., & Dmochowski, R. R. (2021). Foundational Statistical Principles in Medical Research: Sensitivity, Specificity, Positive Predictive Value, And Negative Predictive Value. *Medicina*, *57*(5), 503. <https://doi.org/10.3390>

Stigler, S. Fisher and the 5% level. CHANCE 21, 12 (2008). https://doi.org/10.1007/s00144-008-0033-3

Tongtian, Zhu. (2020). Analysis of the Applicability of the Random Forest. 1607(1):012123-. doi: 10.1088/1742-6596/1607/1/012123

References

Allen, J., MD (2023, January 4). *Understanding The 2023 Medicare Hospital Readmission Penalty*. Hospital Medical Director. Retrieved February 29, 2024, from <https://hospitalmedicaldirector.com/understanding-the-2023-medicare-hospital-readmission-penalty/>

Anscombe, F. J. (1973). Graphs in Statistical Analysis. The American Statistician, 27(1), 17–21. <https://doi.org/10.2307/2682899>

Bishop, P. A., & Herron, R. L. (2015). Use and Misuse of the Likert Item Responses and Other Ordinal Measures. International journal of exercise science, 8(3), 297–302.

Bobbit, Z. (2021, June 22). *RMSE vs. R-Squared: Which Metric Should You Use?* Statology. Retrieved April 29, 2024, from <https://www.statology.org/rmse-vs-r-squared/>

[Bold Analytics : Mark Keith]. (2024, January 29). *Python Data Science: Automating EDA: Univariate Statistics and Visualizations* [Video]. YouTube. <https://www.youtube.com/watch?v=eloR5Li0Huo&list=PLe9UEU4oeAuX3GZgRKzUlLcHd5SpYzZGW&index=1>

[Bold Analytics : Mark Keith]. (2024, February 5). *Python Data Science: Automating Cleaning: Data Wrangling: Empty, Single Value, Primary Key columns* [Video]. YouTube. https://www.youtube.com/watch?v=8IsxTImOhF0&list=PLe9UEU4oeAuX3GZgRKzUlLcHd5SpYzZGW&index=8

Brownlee, J. (2020, August 15). *Data Leakage in Machine Learning*. Machine Learning Mastery. Retrieved April 15, 2024, from <https://machinelearningmastery.com/data-leakage-machine-learning/>

Brownlee, J. (2023, February 16). *Regression Metrics for Machine Learning*. Machine Learning Mastery. Retrieved April 26, 2024, from <https://machinelearningmastery.com/regression-metrics-for-machine-learning/#:~:text=Accuracy%20(e.g.%20classification%20accuracy)%20is,if%20you%20think%20about%20it>.

Bruce, P., Bruce, A., & Gedeck, P. (2019). Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python (2nd ed.). O'Reilly Media.

D, K. (2023, February 15). *Optimizing Performance: SelectKBest for Efficient Feature Selection in Machine Learning*. Retrieved April 15, 2024, from <https://medium.com/@Kavya2099/optimizing-performance-selectkbest-for-efficient-feature-selection-in-machine-learning-3b635905ed48>

Horsch, A. (2021, February 14). *Detecting and Treating Outliers In Python — Part 3 Hands-On Tutorial On Treating Outliers — Winsorizing and Imputation*. Towards Data Science. Retrieved April 15, 2024, from <https://towardsdatascience.com/detecting-and-treating-outliers-in-python-part-3-dcb54abaf7b0>

Kumar, D. (2018, December 25). *Introduction to Data Preprocessing in Machine Learning*. Towards Data Science. Retrieved April 15, 2024, from <https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d>

Larouse, D. T. (2015). Data Mining and Predictive Analytics (p. 20). John Wiley & Sons, Incorporated. <http://ebookcentral.proquest.com/lib/westerngovernors-ebooks/detail.action?docID=7104155>.

Middleton, K. (2023, July 13). D208 - Webinar: Getting Started with D208 Part II [Lecture]. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=39bbe2db-de7d-4bf5-913b-af5c0003da9d>

Monaghan, T. F., Rahmen, S. N., Agudelo, C. W., Wein, A. J., Lazar, J. M., Everaert, K., & Dmochowski, R. R. (2021). Foundational Statistical Principles in Medical Research: Sensitivity, Specificity, Positive Predictive Value, And Negative Predictive Value. *Medicina*, *57*(5), 503. <https://doi.org/10.3390>

Pedregosa, F., Varoquaux, G., & Michel, V. (n.d.). *Sklearn.Ensemble.RandomForestRegressor*. Sci-Kit Learn. Retrieved April 29, 2024, from <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>

Pramoditha, R. (2021). *Encoding Categorical Variables* [Photograph]. Medium : Towards Data Science. <https://towardsdatascience.com/encoding-categorical-variables-one-hot-vs-dummy-encoding-6d5b9c46e2db>

Pramoditha, R. (2021, December 16). *Encoding Categorical Variables: One-hot vs Dummy Encoding Implementation with Pandas and Scikit-learn*. Towards Data Science. Retrieved April 15, 2024, from <https://towardsdatascience.com/encoding-categorical-variables-one-hot-vs-dummy-encoding-6d5b9c46e2db>

Sahai, N. (2023, September 6). *Random Forest Regression – How it Helps in Predictive Analytics?* Retrieved April 29, 2024, from <https://www.analytixlabs.co.in/blog/random-forest-regression/#How_do_you_use_random_forest_for_regression_problems>

Santos, G. (2021, August 18). *How to do Cross-Validation, KFold and Grid Search in Python*. Toward Data Science. Retrieved April 15, 2024, from <https://medium.com/towarhttps://medium.com/gustavorsantos/how-to-do-cross-validation-kfold-and-grid-search-in-python-e570cdb20a28ds-data-science/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb>

Singh, J. (2020, December 17). *Random Forests: Pros and Cons*. DataDrivenInvestor. Retrieved April 29, 2024, from https://medium.datadriveninvestor.com/random-forest-pros-and-cons-c1c42fb64f04

Soetewey, A. (2019, December 30). Variable types and examples. Stats and R. Retrieved August 26, 2023, from <https://statsandr.com/blog/variable-types-and-examples/>

Stigler, S. Fisher and the 5% level. CHANCE 21, 12 (2008). <https://doi.org/10.1007/s00144-008-0033-3>

Tongtian, Zhu. (2020). Analysis on the Applicability of the Random Forest. 1607(1):012123-. doi: 10.1088/1742-6596/1607/1/012123

Weiran, S. (2019, October 22). *Avoid Data Leakage — Split Your Data Before Processing*. Towards Data Science. Retrieved April 15, 2024, from <https://towardshttps://medium.com/towards-data-science/avoid-data-leakage-split-your-data-before-processing-a7f172632b00datascience.com/encoding-categorical-variables-one-hot-vs-dummy-encoding-6d5b9c46e2db>

[WGU - D206 Data Cleaning]. Middleton, K. (2023, August 1). Getting Started with D206 | Missing Values [Video]. WGU Panopto. <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=767749d2-ba19-4f94-bec8-b058017b2f5e>

Vishalmendekarhere (2021, July 11). It’s all about Assumptions, Pros & Cons. Medium. Retrieved April 29, 2024, from <https://medium.com/swlh/its-all-about-assumptions-pros-cons-497783cfed2d>