D209: Data Mining I

Task 2: Data Mining using Random Forest Regressor

Andrew Mecchi

Masters of Data Analytics, Western Governors University

The following report covers task two of the Performance Assessment for D209 Data Mining I. This document is categorized by the questions defined in the rubric.

A: Research Question

B: Method Justification

C: Data Preparation

D. Analysis

E. Data Summary and Implications

F: Panopto Video

G: Sources

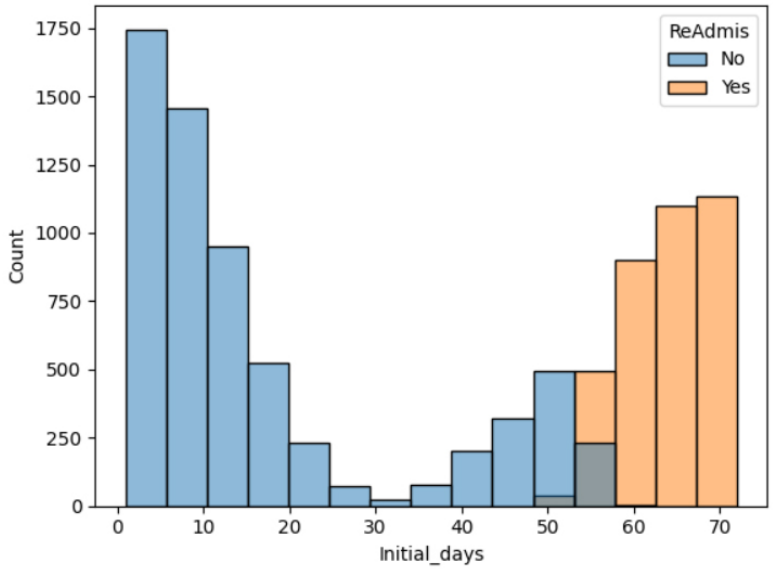
Part I: Research Question

A1. Propose a research question relevant to patient readmissions using random forests.

Hospitals are overseen by the Centers for Medicare and Medicaid Services (CMS) who issue financial penalties to facilities who incur patient readmissions for certain medical conditions. Next year, CMS is including hip/knee replacements and COPD to the list of offenses, therefore, it is important to create and implement a predictive model to assist in the reduction of readmissions. Previous exploration of data distributions have shown a large percentage of readmission cases with increased lengths of hospital stay as Initial Days approach 50 days (Fig. 1).

Can a Random Forest regressor accurately predict Initial Days?

Figure 1: Histogram of Initial Days marked by Readmissions



A2. Define one goal of the Random Forest regressor relevant to the medical data.

The primary goal of the analysis is to design an accurate predictive model using the random forest regressor. To achieve this goal, the data must be explored, transformed, and analyzed. Addressing Initial Days identifies the target response variable while explanatory variables are identified through the feature selection method using Select K Best. These selected features define the building blocks of the predictive model and metrics are optimized through hyperparameter analysis using GridSearhCV. The results of the grid search tune the parameters of the random forest regressor and augment the performance of the machine learning model. If the algorithm is found statistically accurate, hospitals can utilize the regressor to identify patients who are projected to have prolonged stays. With this knowledge, the hospital(s) can implement proactive treatment plans for those patients to decrease lenghth of stay and ultimately reduce the possibility of incurring readmission fines.

Part II: Method Justification

B1. Explain how random forests analyze data and possible outcomes.

The decision to use random forest analysis is founded on its versatility as random forest models can be applied to both regression or classification problems. The target response variable, Initial Days, is continuous numeric data and determines the use of the random forest regressor. Random forests regression is a machine learning technique composed of numerous decision trees that operate as an ensemble whereby each individual tree in the random forest results with a class prediction and the class with the most votes (classification) or calculated average (regression) becomes the model's prediction (Yiu, 2019). The building blocks of a random forest, the decision tree, is composed of a root node, internal node, branches, and leaf nodes. The root node is the inception of the classification or regression task, a starting point with no previous branches. The outgoing branches from the root node then feed into the internal nodes (or decision nodes) and based on the available features, both node types conduct evaluations to form homogenous subsets, which are denoted by leaf nodes that represent all the possible outcomes within the dataset (IBM, n.d.). To avoid correlation between decision trees within the random forest, two methods assist in maintaining model integrity, bootstrap aggregation (bagging) and feature randomness. Bagging is the process where each individual tree takes a random sample from the data with replacement (resulting in different trees) while enforcing feature randomness at decision nodes guarantees variation amongst the trees within the model, ultimately lowering correlation between trees and improves diversification (Yiu, 2019). Random forests are ideal to answer the research question as the nonlinear algorithm can address both classification and regression problems, the models are not sensitive to outliers, and all treated data can be included in the analysis.

Following initial data exploration and manipulation, feature selection reduced the dataframe to one response variable, Initial Days, and four explanatory variables; Total Charge, Readmission, Timely Admission, and Children. The random forest is composed of numerous decision trees made from a subset of the medical data records (bagging) and subset of the explanatory features (feature randomness) listed above. In other words, *n* random records and *m* features are taken from the data set having *k* number of records (Sruthi, 2023). These bagged records create individual decision trees, each of which result with an output and continue to branch until they reach a final leaf node. The predicted output is determined by averaging values in regression models or majority voting with classification models. The random forest regressor will be applied to the test data and predicted values are calculated by averaging the results of the leaf nodes from the decision trees within the random forest. As refined through hyperparameter tuning, the model determines number of days using four levels of decision nodes (max depth), four features considered per decision split (max features), and uses 80 estimators. Therefore, when a query for length of stay is made, the predicted value is the calculated average of the decision trees' leaf node outputs from the hyperparameter-tuned regression model.

B2. Summarize one assumption of the chosen classification method.

When using a random forest regressor, it is important to understand the assumptions of the method and how it affects the model. One such assumption is what separates random forests from decision trees, the random selection of subset features when a decision node is split. In decision trees, the split is made by the feature that yields the most separation between nodes. In a random forest, each tree can pick only from a random subset of features which forces even more variation amongst the trees in the model (Yiu, 2019). The inclusion of random feature selection improves diversification of tree creation and lowers correlation across trees and reduces errors in the process. From the collection of low correlated trees, accurate averages can be calculated for model predictions (or majority voting in classification models).

B3. List the packages or libraries from Python and their role in the analysis.

Packages & Libraries	Support of Data Analysis
pandas	Manipulate dataframe, import/export csv files
numpy	Manipulate data in arrays and array calculations
seaborn	Visualization of data: histogram, actual vs. predicted plot
matplotlib.pyplot	Visualizations: decision tree plot, feature importance plot
from sklearn.feature_selection import f_regression	Feature selection for random forest model
from sklearn.feature_selection import SelectKBest	Feature selection for random forest model
from sklearn.model_selection import train_test_split	Split data into train and test sets for random forest model
from sklearn.ensemble import RandomForestRegressor()	Instantiate Random Forest Regressor
from sklearn.metrics import GridSearchCV	Gridsearch cross validation used for model optimization/hyperparameter tuning
from sklearn.tree import plot_tree	Random forest visualization
From sklearn.metrics import mean_squared_error as MSE	Calculation of mean squared error
From sklearn.metrics import r2_score	Calculation of r-square score

From sklearn.metrics import mean_absolute_error	Calculation of mean absolute error

Part III: Data Preparation

C1. Describe one data preprocessing goal relevant to Random Forest Regression from part A1.

The research question will be addressed using the random forest regressor, but to ensure model efficiency and accuracy, data preprocessing methods must be utilized to ensure the best features are selected for the model. One preprocessing goal in the design of the regression model is to select the most influential features relative to the response variable, Initial Days. In order to narrow the medical data features, categorical variables must be reexpressed into numeric data previous to fitting the Select K Best function. Binary categorical variables were reexpressed with binary numeric values, 0 and 1 (corresponding to “No” and “Yes” respectively) while nominal categorical data was reexpressed using pandas’ get dummies one-hot encoding. After the categorical data has been reexpressed into numeric values, the Select K Best function is executed and found four statistically significant features, Readmissions, Total Charge, Children, and Timely Admission.

C2. Identify the initial data set variables that you will use to perform the analysis for the classification question from part A1, and classify *each* variable as continuous or categorical.

Feature selection was performed on the reexpressed data using sklearn’s Select K Best function which calculates the statistical significance of the inputted variables. The features were split into the response variable, Initial Days, and all (relevant) remaining variables were entered as explanatory features. Select K Best calculates p-values based on the data and those with statistical significance (measuring $p < 0.05$) are retained for the random forest regression model. The Select K Best function found the following features statistically significant and were used for the random forest model.

Feature	Variable Type	Data Type	Reexpressed	SelectKBest p-value
Initial_days	Response/ Dependent	Continuous	n/a	n/a
ReAdmis	Explanatory/ Independent	Categorical	Encoded as binary 0/1	0.000

TotalCharge	Explanatory/ Independent	Continuous	n/a	0.000
Children	Explanatory/ Independent	Continuous	n/a	0.025
Timely_admissioon	Explanatory/ Independent	Categorical - (ordinal/numeric value)	No	0.026

C3. Explain *each* of the steps used to prepare the data for the analysis. Identify the code segment for *each* step.

1) Upload medical data (csv file) into dataframe and create backup.

```
# Upload CSV file and create backup
data =
pd.read_csv(r'C:\Users\andrew\Desktop\WGU_MSDA\D209_Data_Mining\medical_cl
ean.csv')
data_backup = data
```

2) Rename Item columns 1-8 to their corresponding names found in data dictionary

```
# Rename Item columns to represent their corresponding survey category
data.rename(columns = {'Item1':'Timely_admission',
                       'Item2': 'Timely_treatment',
                       'Item3':'Timely_visits',
                       'Item4':'Reliability',
                       'Item5':'Options',
                       'Item6':'Hours_of_treatment',
                       'Item7':'Courteous_staff',
                       'Item8':'Listening_doctor'}, inplace = True)
```

3) Search for duplicate records using Customer Id, defined in data dictionary as unique values

```
# Explore data before prepping for predictive modeling
# Confirm no duplicate entries for data by checking for unique patient ids

data['Customer_id'].duplicated().value_counts()
```

4) Search data for missing data or null values

```
# Search for null values
data.isnull().sum()
```

5) Explore data frame, data types, and info

```
# Explore data types, identify target variable and explanatory variables
# Search df info

data.info()
```

6) Create data frame for task two

```
# Task 2 dataframe
t2 = data.copy(deep = True)
```

7) Drop unnecessary features relative to research question

```
# Drop features not relevant to task
t2.drop(columns = ['CaseOrder', 'Customer_id', 'Interaction', 'UID',
'City', 'State', 'County', 'Lat', 'Lng', 'TimeZone', 'Job', 'Marital'],
axis = 1, inplace = True)
```

8) Identification of research question. Target response variable is initial days and why target initial days relative to readmissions. View histogram of Initial Days relative to readmission cases.

```
# Research question
# Can the random forest regressor be used to accurately predict Initial
Days?

# Targeting initial days due to high level of readmissions over ~48 days
# View histogram of initial days relative to readmission cases

sns.histplot(x = 'Initial_days', data = t2, hue = t2['ReAdmis']);
```

9) View summary statistics of numeric data

```
# Target response variable --- Initial Days
# View summary statistics before selecting variables for model
t2.describe()
```

10) View summary statistics of categorical data

```
# Summary statistics of categorical data
# Look for binary variables to encode
t2.describe(include = 'object')
```

11) Reexpress binary categorical data into numerical data and confirm encoding

```
# Encoding binary variables
# Binary features: Soft Drink, High Blood, Stroke, Overweight, Arthritis,
Diabetes, Hyperlipidemia, Back Pain, Anxiety, Allergic Rhinitis, Reflux
Esophagitis, and Asthma
binary_cols = ['ReAdmis', 'Soft_drink', 'HighBlood', 'Stroke',
'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma' ]
binary_values = {'Yes': 1, 'No': 0}

for col in binary_cols:
  df[col] = df[col].replace(binary_values)

# Confirm binary encoding
df.info()
```

12) Observe remaining categorical variables needing reexpression

```
# View summary stats of remaining categorical types before one-hot
encoding
# Area, Gender, Initial Admin, Complication Risk, and Services to be
encoded

df.describe(include = 'object')
```

13) Reexpression of nominal categorical data (beyond binary input) by getting dummy columns and confirm one-hot encoding

```
# Loop to create dummy variables
for col in df:
  if not pd.api.types.is_numeric_dtype(df[col]):
    df = pd.get_dummies(df, columns = [col], prefix = col)

# Confirm addition of dummies
df.head()
```

14) Confirm all data reexpressed as numeric

```
# Confirm all numeric data
df.info()
```

15) Calculate correlations relative to Initial Days for scope of feature selection

```
# Calculate correlation values relative to target variable - Initial Days
# Dataframe correlations relative to Initial Days
df_corr = df.corr()['Initial_days']
```

16) View correlations as sorted

```
# Sort correlations
correlations = df_corr.sort_values(ascending = False)
print(correlations)
```

17) View heatmap of correlations

```
# Heatmap of correlations

sns.heatmap(df.corr(), cmap = 'magma');
```

18) Selection of feature data for model efficiency using SelectKBest, create arrays for SelectKBest function

```
# Feature selection for random forest analysis
# SELECT K BEST

# assign values for x and y
x_days = df.drop(['Initial_days'], axis = 1)
y_days = df['Initial_days']

print(x_days.shape)
print(y_days.shape)
```

19) Create SelectKBest model

```
# Make SelectKBest model
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression

best = SelectKBest(score_func = f_regression, k = 'all')

days_best_model = best.fit_transform(x_days, y_days)
```

20) Identify, view, and print, p-values that measure below alpha ($p < 0.05$) for feature selection

```
# Find p-values --- Initial Days
days_p_values = pd.DataFrame({'Feature': features, 'p_value':
best.pvalues_}).sort_values('p_value')

print(days_p_values)

# View features that measure below alpha (0.05)
days_p_values[days_p_values['p_value'] < 0.05]
```

21) Filter dataframe to include features selected from SelectKBest

```
# Subset data frame to select statistically significant features from
selectkbest
df_t2 = df[['Initial_days', 'ReAdmis', 'TotalCharge', 'Children',
'Timely_admission']]

df_t2.info()
```

22) Prepare data for a random forest model by separating data into train and test sets using a 70/30 ratio of train to test data. Coded with stratify to keep percentages of readmissions similar between train and test sets.

```
# Train/Test split Post Select K Best
# Import train test split
# Stratify train test split based on readmissions
from sklearn.model_selection import train_test_split

# Define response and explanatory variables
X = df_t2.drop(['Initial_days'], axis = 1)
y = df_t2['Initial_days']

# Split response and explanatory variables into train and test sets
# Stratify to retain proportional response data representation
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size =
0.7, test_size = 0.3, random_state = 61, stratify = X['ReAdmis'])

# Print shape of train/test sets
print('Train Set')
print(X_train.shape)
print(y_train.shape)
print('\n ----- \n')
print('Test Set')
print(X_test.shape)
print(y_test.shape)
```

23) Confirm proportional stratification of readmission samples

```
# Compare percentages of readmissions of train test split
```

```
pct_readmis_train = X_train['ReAdmis'].value_counts() /
(X_train['ReAdmis']).shape[0]
pct_readmis_test = X_test['ReAdmis'].value_counts() /
(X_test['ReAdmis']).shape[0]

print("Percentages of Readmissions from Train Data: " + '\n' +
str(pct_readmis_train))
print('\n ---------------------------- \n')
print("Percentages of Readmissions from Test Data: "  + '\n' +
str(pct_readmis_test))
```

24) Data has been explored, reexpressed, features selected, and split into test and training sets and is ready for random forest regressor analysis.

C4. Provide a copy of the cleaned data set.

See Attached: treated_model_data_task_2.csv

Part IV: Analysis

D1. Split the data into training and test data sets and provide the file(s).

The treated data is split into training and test sets, the training set is composed of 70% of the data and is used to train the random forest model. The remaining 30% is used for testing model efficacy and accuracy. The split is coded as, “stratify,’ to ensure each set contains approximately the same percentage of samples of each target class (readmissions) as the complete set (Scikit-learn, n.d.).

See Attached file(s): x_train_pa_task_2.csv, y_train_pa_task_2.csv, x_test_pa_task_2.csv, y_test_pa_task_2.csv, train_data_pa_task_2.csv, test_data_pa_task_2.csv

```
# Train/Test split Post Select K Best
# Import train test split
# Stratify train test split based on readmissions
from sklearn.model_selection import train_test_split

# Define response and explanatory variables
X = df_t2.drop(['Initial_days'], axis = 1)
```

```
y = df_t2['Initial_days']

# Split response and explanatory variables into train and test sets
# Stratify to retain proportional response data representation
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size =
0.7, test_size = 0.3, random_state = 61, stratify = X['ReAdmis'])
```

D2. Describe the analysis technique you used to appropriately analyze the data.

The data has been explored, the most significant features selected, split into train/test sets, and is prepared for predictive analysis. When employing the random forest regressor, it's important to understand the algorithm should be refined for optimal performance. The random forest regressor is instantiated with default settings; n_estimators = 100, criterion = 'squared_error', max_depth = None (nodes are expanded until all leaves are pure or leaves contain less than min_samples_split), min_samples_split = 2, min_samples_leaf = 1, min_weight_fraction_leaf = 0.0, max_features = 1.0, max_leaf_nodes = None, min_impurity_decrease = 0.0, bootstrap = True, oob_score = False, n_jobs = None, random_state = None, verbose = 0, warm_start = False, ccp_alpha = 0.0, and max_samples = None (Scikit-learn, n.d.). Out of the box, these default settings may produce an accurate model, however, the metrics aren't optimized for the medical data. Hyperparameter tuning is the process to find a set of parameters used to control the behavior of the model/algorithm and are adjustable in order to obtain an improvised model with maximized performance and minimal error (Pandian, 2022). To explore the best parameters for the random forest regressor, GridSearchCV is employed to iterate the model and finds the maximized performance settings.

When assigning the GridSearchCV, the parameters tested were max depth, max features, number of estimators, and used a 5-fold cross validation (Fig. 2). The max depth sampled a range of values, 2, 3, 4, and None, allowing the grid search to maximize leaf expansion until leaf nodes are pure. The max depth was set to test values, 2, 3, 4, while the number of estimators were sampled with 20, 40, 60, 80, and 100 records. The results of the hyperparameter tuning yielded the optimal depth of four, max features of four, and 80 as the best number of estimators (Fig. 3). The depth denotes the number of levels, or decision nodes in an individual tree within the random forest. The max features are the number of features considered to make the best split at each node and the number of estimators is the number of trees in the forest. Following the success of the GridSearchCV analysis and defining the random forest regressor with the optimal parameters, the train data is fit into the final model.

Figure 2: Hyperparameter Tuning using GridSearchCV

```
[31] # Import Random Forest Regressor
     from sklearn.ensemble import RandomForestRegressor

     # Instantiate regressor
     rf = RandomForestRegressor(random_state = 61, n_jobs = -1)

[32] # Employ GridSearchCV for hyperparameter tuning and Random Forest optimization
     from sklearn.model_selection import GridSearchCV

     # Define parameters, look for optimized max_depth, max_features, n_estimators
     params = {'max_depth': [2, 3, 4, None], 'max_features': [2, 3, 4], 'n_estimators': [20, 40, 60, 80, 100]}

[33] # Random Forests
     # Employ random forest GridSearchCV using 5-fold cross validation
     rf_cv = GridSearchCV(rf, params, cv = 5, n_jobs = -1)

     # Fit model
     grid_fit = rf_cv.fit(X_train, y_train)
```

With the success of the hyperparameter search, the best estimator regressor is employed on the training data set and baseline performance metrics are calculated for downstream comparative analysis with the test data. The four baseline metrics returned to compare training and test sets are r-squared (0.984), mean absolute error (2.676), mean squared error (11.153), and root mean squared error (3.340) (Fig. 4). The random forest regressor is then used to predict results from the test data. The efficacy and accuracy of the model is determined based on comparing test predictions with the same performance metrics from the training model.

Figure 3: Results of Hyperparameter Tuning using GridSearchCV

```
# Print results
print("Best Parameters: " + str(grid_fit.best_params_))
print("Best Score: " + str(grid_fit.best_score_))
```

```
Best Parameters: {'max_depth': 4, 'max_features': 4, 'n_estimators': 80}
Best Score: 0.9834374354137998
```

```
# Regression model after finding best parameters
# Define best regressor based on GridSearchCV results
# Best Parameters: {'max_depth': 4, 'max_features': 4, 'n_estimators': 80}

# Idendtify the best estimator
rf_best = grid_fit.best_estimator_


# Confirm best estimator
rf_best
```

```
RandomForestRegressor
RandomForestRegressor(max_depth=4, max_features=4, n_estimators=80, n_jobs=-1,
                      random_state=61)
```

Figure 4: Train Set Performance Metrics: R-squared, Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE)

```
# Training Set Results
from sklearn.metrics import r2_score

y_train_pred = model.predict(X_train)

train_r2 = r2_score(y_train, y_train_pred)

# R2 score
print("Train r-square score: %0.3f" % (train_r2))
```

```
Train r-square score: 0.984
```

```
# Calculations for mean absolute error --- train data
from sklearn.metrics import mean_absolute_error

train_mae = mean_absolute_error(y_train, y_train_pred)

print("Train Data Mean Absolute Error: %0.3f" % (train_mae))
```

```
Train Data Mean Absolute Error: 2.676
```

```
# Calculations for mean squared error --- train data
from sklearn.metrics import mean_squared_error as MSE

train_mse = MSE(y_train, y_train_pred)

print('Train Data Mean Squared Error: %0.3f' % (train_mse))
```

```
Train Data Mean Squared Error: 11.153
```

```
# Calculations for root mean squared error --- train data

train_rmse = train_mse**(1/2)

print('Train Data Root Mean Squared Error: %0.3f' % (train_rmse))
```

```
Train Data Root Mean Squared Error: 3.340
```

D3. Provide the code used to perform the classification analysis from part D2.

```
# Import Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor

# Instantiate regressor
rf = RandomForestRegressor(random_state = 61, n_jobs = -1)
```

```
# Employ GridSearchCV for hyperparameter tuning and Random Forest
optimization
from sklearn.model_selection import GridSearchCV

# Define parameters, look for optimized max_depth, max_features,
n_estimators
params = {'max_depth': [2, 3, 4, None], 'max_features': [2, 3, 4],
'n_estimators': [20, 40, 60, 80, 100]}
```

```
# Random Forests
# Employ random forest GridSearchCV using 5-fold cross validation
rf_cv = GridSearchCV(rf, params, cv = 5, n_jobs = -1)

# Fit model
grid_fit = rf_cv.fit(X_train, y_train)
```

```
# Print results
print("Best Parameters: " + str(grid_fit.best_params_))
print("Best Score: " + str(grid_fit.best_score_))
Best Parameters: {'max_depth': 4, 'max_features': 4, 'n_estimators': 80}
Best Score: 0.9834374354137998
```

```
# Regression model after finding best parameters
# Define best regressor based on GridSearchCV results
# Best Parameters: {'max_depth': 4, 'max_features': 4, 'n_estimators': 80}

# Identify the best estimator
rf_best = grid_fit.best_estimator_

# Confirm best estimator
rf_best
```

RandomForestRegressor

```
RandomForestRegressor(max_depth=4, max_features=4, n_estimators=80,
n_jobs=-1,random_state=61)
```

```
# Fit best model
model = rf_best.fit(X_train, y_train)

# Best Predictions
predictions = model.predict(X_test)
```

Part V: Data Summary and Implications

E1. Explain the accuracy and the mean squared error (MSE) of your regression model.

When assessing model performance of the optimized random forest regressor, the accuracy and mean squared error (MSE) are calculated and compared to the corresponding metrics of the training model. Accuracy of the random forest regression model is evaluated through the coefficient of determination, or r-squared value, and represents the ability of the regression model to correctly predict the outcome of imputed data. The r-squared value is a goodness-of-fit measurement that evaluates the strength of the linear relationship between variables and examines variance in the dependent variable that the independent variable(s) collectively explain when predicting an event (Frost, 2022). The accuracy is representative of the number of correct predictions from the test set relative to the actual values. The calculated r-squared values of the test data measured, 0.983 (Fig. 5) compared to the 0.984 from the train set; confirming an accurate predictive model.

Figure 5: R-squared Results of Test Predictions

```
# Calculations of r-square score --- test data

test_r2 = r2_score(y_test, predictions)

print("Test Data R2 Score: %0.3f" % (test_r2))
```

```
Test Data R2 Score: 0.983
```

In addition to calculating model accuracy with r-squared, the mean squared error (MSE) of the model was computed and compared. The MSE measures the amount of error in the random forest model and assesses the average squared difference between the observed and predicted values, with a value of zero representing a model with no error (Frost, 2021). The model's

calculated MSE score for the test set resulted in a value of 11.641 (Fig. 6) compared to 11.153 from the trained model. However, when assessing how well a model fits a dataset, the root mean squared error (RMSE) is used because it is measured in the same units as the response variable (Zach 2021). Calculating the RMSE of the predicted values yields the average differential between the actual values and those predicted in terms of response variable units. Therefore, the RMSE of the test data represents a deviation of 3.412 days (Fig. 6) from the actual values while the train data had an RMSE of 3.340. Based on the accuracy of the r-squared score, MSE, and RMSE, the random forest model is proven both accurate and effective.

Figure 6: Test Prediction Performance Metrics: MSE and RMSE

```
# Calculations for mean squared error --- test data

test_mse = MSE(y_test, predictions)

print('Test Data Mean Squared Error: %0.3f' % (test_mse))
```

```
Test Data Mean Squared Error: 11.641
```

```
# Calculations for root mean squared error --- test data

test_rmse = test_mse**(1/2)

print('Test Data Root Mean Squared Error: %0.3f' % (test_rmse))
```

```
Test Data Root Mean Squared Error: 3.412
```

E2. Discuss the results and implications of your regression analysis.

A random forest regression model was designed to predict the length of stay of a patient in terms of Initial Days. If the model's results proved significant, the predictive power of the regressor would assist hospitals with important insight into length of stay. This knowledge could be used to implement proactive treatment plans to reduce overall days and patient readmissions. The medical data was explored and treated before using the Select K Best feature selection method to determine the best variables to include in the initial model. The most significant features identified through Select K Best were determined as variables whose p-values measured below alpha, or 0.05. The feature selection method found four variables with the the strongest relationships relating to Initial Days; Readmissions, Total Charge, Children, and Timely Admission. The data was then split into two groups, a training set, used to train the model from 70% of the data, and the remaining 30% composed the test set used for performance validation. To ensure the best model, hyperparameter tuning using GridSearchCV optimized the max features, max depth, and number of estimators of the random forest regressor. The model was then fit with the training data and predicdive power was assessed with the test data.

Test data predictions were computed and performance metrics compared between training and test sets. The calculations used to measure model accuracy and effectiveness were r-squared, mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE). Additionally, the influence of independent variables on the response variable (feature importance) were assessed and found that the greatest determining factor in projecting length of stay is Total Charge (Fig. 7). Further visualization of the random forest regressor shows the effect of Total Charge as the determining factor in decision node splits and is indicative of its importance for the accuracy and performance of the predictive model (Fig. 8).

Figure 7: Feature Importance Values and Plot

```
#Feature importance

rf_importances = pd.Series(model.feature_importances_, index = X_train.columns)

# Sort importances
rf_sorted = rf_importances.sort_values(ascending = False)

print(rf_sorted)
```

```
TotalCharge          0.964764
ReAdmis              0.035236
Children             0.000000
Timely_admission     0.000000
dtype: float64
```

```
# Plot importances
rf_sorted.plot(kind = 'barh', color = 'blue')
plt.show()
```

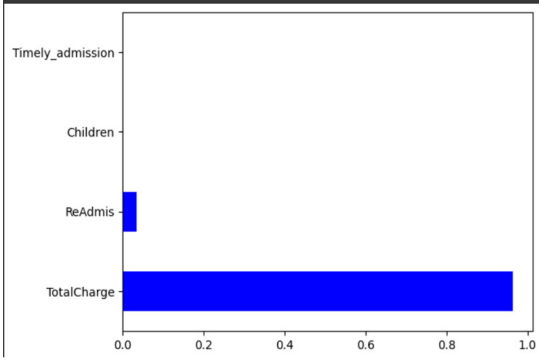
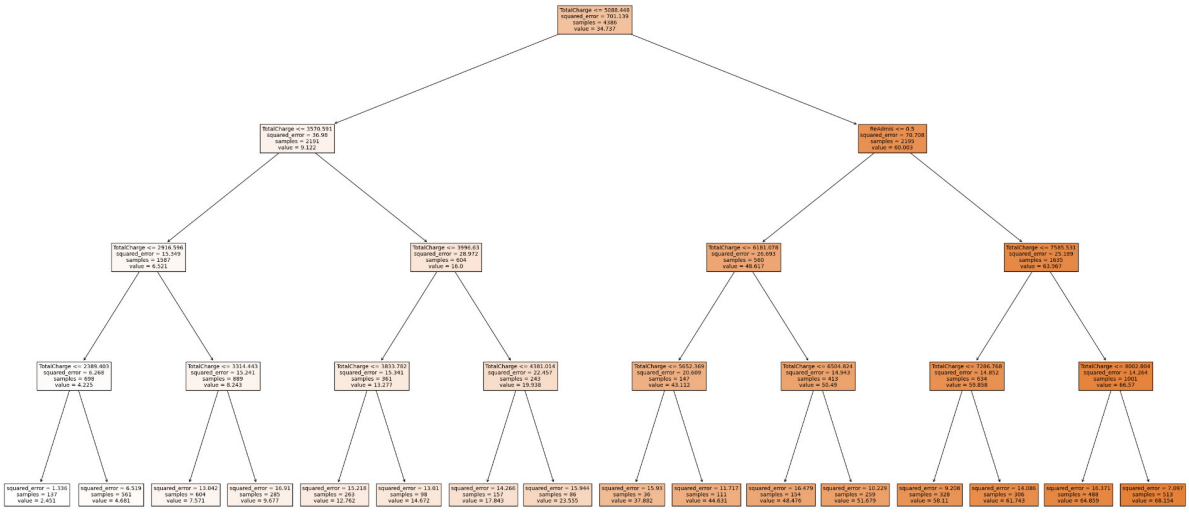


Figure 8: Visual Example of Random Forest Regression Model



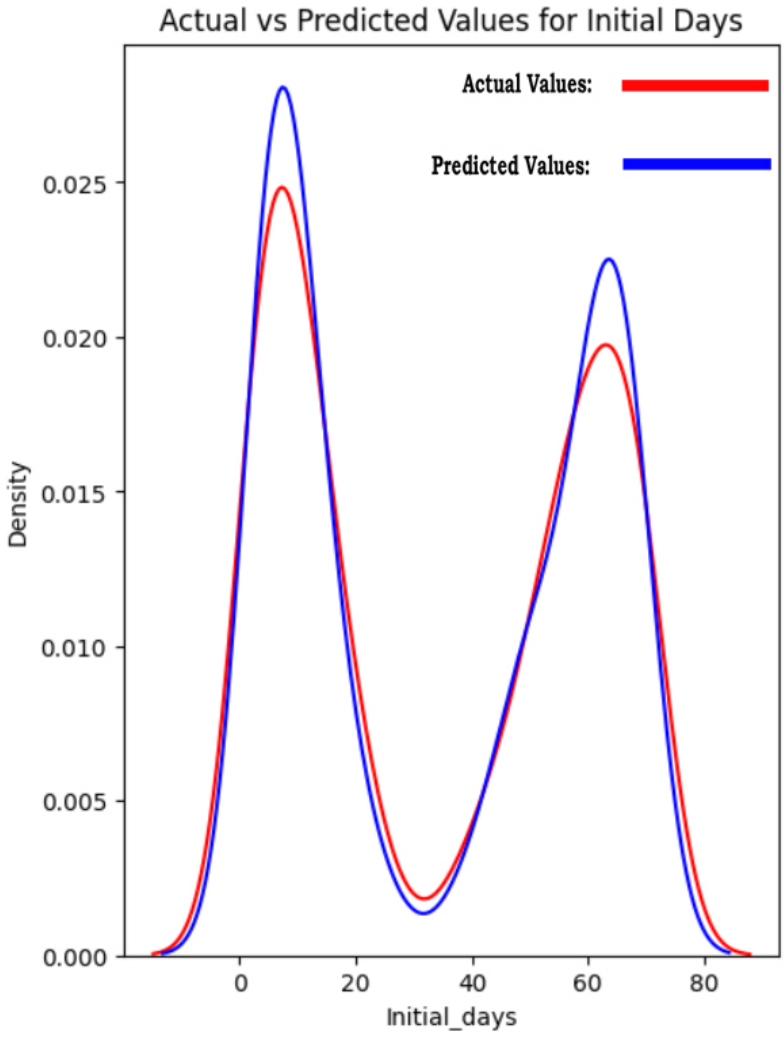
As discussed previously, the r-squared score indicates the effectiveness of the model's prediction ability and deviations from actual values are assessed using MSE and RMSE. The inclusion of mean absolute error (MAE) yields another perspective of model performance by measuring accuracy based on the average difference between the actual value and the value predicted by the model; with the lower MAE indicating a more accurate model (Zach, 2021). The calculated r-squared, MAE, MSE, and RMSE of the trained regressor model resulted with values, 0.984, 2.676, 11.153, and 3.340 respectively, while the same metrics for the test set measured, 0.983, 2.745, 11.641, and 3.412 (Fig. 9). Both r-squared and MAE confirm accurate model predictions and the MSE/RMSE indicate marginal differences between predicted and actual values, thus, the model is confirmed accurate with minimal error.

Figure 9: Performance Metrics of Train and Test Data

Metric	Train Data	Test Data
R^2	0.984	0.983
Mean Absolute Error (MAE)	2.676	2.745
Mean Squared Error (MSE)	11.153	11.641
Root Mean Squared Error (RMSE)	3.340	3.412

The research question set out to project the Initial Days a patient will spend at the hospital and the test data results indicate an effective predictive model. Thus, the number days a patient spends in a hospital can be predicted with high confidence (r-squared of 0.983). The RMSE indicates the average error difference between days projected and actual days spent at the hospital is 3.412 days. The model tests at a high accuracy with a small difference between the predicted values and actual values; therefore providing a reliable model to anticipate a patient's length of stay (Fig. 10). The model's accuracy and performance may be further improved with the inclusion of additional metrics assessed during hyperparameter tuning.

Figure 10: Plot of Model Accuracy



E3. Discuss one limitation of your data analysis.

Employing a random forest regressor to answer the research question proved accurate and statistically significant, however, the limitations of a random forest must be addressed. The nonlinear nature of a random forest has advantages over linear algorithms, however, it is important to understand that a random forest cannot extrapolate, it can only make a prediction that is an average of previously observed labels (Thompson, 2019). This means, the range of predictions made from the model are confined to the minimum and maximum values of the train data. Thus, a problem presents itself if there is a change in the distribution of the input variables present in the training and test data, which is known as a covariate shift (Shubham, 2022). If the distribution of values from the predictive model are fit on different ranges than the test samples, predictions made with the trained model will result with inaccurate projections.

E4. Recommend a course of action for the real-world organizational situation from part A1 based on your results and implications discussed in part E2.

The regression model proved both effective and accurate, however, the model should be implemented with some caution. While the random forest model will not directly assist with reducing fines for patient readmissions, the accurate estimates will prove useful as the hospital(s) can anticipate potential readmissions if a patient's projected length of stay approaches 50 days (Fig. 1). The suggested course of action is to bolster what has been discovered and further delve into a deeper understanding of the model's features.

During random forest analysis, it was found that the most important predictive feature of the model is Total Charge (Fig. 7). However, the total charge is indicative of the end of a patient's stay, thus, holding a strong influence on anticipating length of stay. Continued analysis of Total Charge would augment what has been learned from the random forest model and improve insight into patient readmissions. Until further analysis is performed, hospitals can utilize findings from the Initial Days random forest model to institute proactive treatment plans for patitents expected to have prolonged hospital stays.

Part VI: Demonstration

F. Provide a Panopto video

See Attached

CODE SOURCES

Kumar, A. (2020, June 22). Random Forest Prediction. Towards Data Science. https://towardsdatascience.com/random-forest-ca80e56224c1.

REFERENCES

Frost, J. (2021, November 14). Mean Squared Error (MSE). Statistics By Jim. https://statisticsbyjim.com/regression/mean-squared-error-mse/.

Frost, J. (2022, July 22). How to Interpret R-squared in Regression Analysis. Statistics By Jim. https://statisticsbyjim.com/regression/interpret-r-squared-regression/.

IBM. (n.d.). What is a Decision Tree?. IBM. Retrieved April 28, 2022. https://www.ibm.com/topics/decision-trees#:~:text=A%20decision%20tree%20is%20a,internal%20nodes%20and%20leaf%20nodes.

Pandian, S. (2022, October 20). A Comprehensive Guide on Hyperparameter Tuning and its Techniques. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/.

Scikit-learn. (n.d.). Cross-validation: Evaluating Estimator Performance. Scikit-learn 1.2.2 - documentation. Retrieved April 28, 2022. https://scikit-learn.org/stable/modules/cross_validation.html#stratification.

Scikit-learn. (n.d.). Random Forest Regressor. Scikit-learn 1.2.2 - documentation. Retrieved April 28, 2022. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

Shubham, J. (2022, July 25). Covariate Shift: Unearthing Hidden Problems in Real World Data Science. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2017/07/covariate-shift-the-hidden-problem-of-real-world-data-science

Sruthi, E.R. (2023, April 26). Understand Random Forest Algorithms with Examples (Updated 2023). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/.

Thompson, B. (2019, December 17). A Limitation of Random Forest Regression. Towards Data Science. https://towardsdatascience.com/a-limitation-of-random-forest-regression-db8ed7419e9f#.

Yiu, T. (2019, June 12). Understanding Random Forest. Towards Data Science. https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

Zach. (2021, January 8). How to Calculate Mean Absolute Error in Python. Statology. https://www.statology.org/mean-absolute-error-python/.

Zach. (2021, September 30). MSE vs. RMSE: Which Metric Should you Use?. Statology. https://www.statology.org/mse-vs-rmse/.