JANUARY 25, 2024

PERFORMANCE ASSESSMENT

D209 TASK 2

ROBERT PATTON
WESTERN GOVERNORS' UNIVERSITY
Rpatt33@wgu.edu

Table of Contents

Part I: Research Question

A. Purpose of Data Mining Report

1. Utilizing the random forest classification method, can we predict hospital readmission based on the health conditions reported by patients during their initial hospital admission?
2. The goal of this data analysis is to find out if a hospital can predict readmission based on specific medical conditions. This is of value to a hospital because extensive readmission rates tend to impact hospital revenue, deter current and future patients from seeking services, and can cause fines based on criteria via the Center for Medicare and Medicaid Services (Upadhyay et al. 2019).

Part II: Method Justification

B. Reasoning of Chosen Classification Method

1. Random forest classification is used when an analyst has a categorical target variable for use in a study, from a primary dataset, and needs to perform a classification model for identifying which independent variables influence the outcome of a target variable the most efficiently. The data is analyzed using training and testing sets. Therefore, random forest classification creates multiple decision trees at random and creates predictions for each decision tree. A random forest classification model is an example of an ensemble method, combining predictions from other models created. These multiple decisions trees are created using random subsets of the data and features, creating predictions made by calculating the predication for each decision tree and taking the most popular result (Shafi 2023).
2. Random forest classification models are non-parametric and can handle skewed and multi-modal data, along with the assumption of no formal distributions (Mendekar 2021).
3. Python packages used are as follows:

   Pandas: Used to load in the CSV file.

   NumPy: Used for mathematical operations and array functions.

   Matplotlib.pyplot: Used to create visualizations.

   Seaborn: Another visualization tool.

Sklearn.model_selection.train_test_split: Used to split the data into training and testing sets.
Sklearn.metrics: Used to construct a confusion matrix, ROC curve with AUC, a classification report on the model, and to calculate accuracy, recall, and precision scores.
From sklearn.ensemble import RandomForestClassifier: Used to run the model on the test and train data sets.
Sklearn.model_selection.RandomizedSearchCV: Used for hyperparameter tuning.
from scipy.stats.randint: Used to specify distributions.

Part III: Data Preparation

C. Data Preparation

1. To prepare for implementing the random forest classification method for predicting readmission, we first need to clean the data we are working with. This includes looking for any missing values or duplicates and ensuring that all our variables have numerical values using one-hot encoding.
2. Classification of variables used for data analysis are as follows:

ReAdmis	**Categorical**
Age	**Numeric**
Marital	**Categorical**
Gender	**Categorical**
VitD_levels	**Numeric**
Doc_visits	**Numeric**
Initial_admin	**Categorical**
vitD_supp	**Numeric**
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**
Initial_days	**Numeric**

3. Steps for preparing the data are as follows:
    - **#Import libraries and packages**
      ```
      import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      %matplotlib inline
      import seaborn as sns
      import warnings
      warnings.filterwarnings('ignore')

      from sklearn.ensemble import RandomForestClassifier
      from sklearn.model_selection import RandomizedSearchCV,
      train_test_split
      from sklearn.metrics import (accuracy_score, recall_score,
      precision_score, confusion_matrix, roc_curve, auc)
      from scipy.stats import randint
      ```
    - **#Read in the CSV file**
      ```
      mdf= pd.read_csv('/Users/robertpatton/Desktop/Desktop - Robert’s
      MacBook Pro/D209/medical_clean.csv')
      ```
    - **#Examine the data info**
      ```
      mdf.info()
      ```
    - **#Examine if duplicate or missing values exist**

```
mdf.isnull().sum()
mdf.duplicated()
```

- **#Identify variables for research study**

```
research_cols = [
"ReAdmis", "Age", "Marital", "Gender", "VitD_levels", "Doc_visits",
"Initial_admin", "vitD_supp",  "HighBlood",
"Stroke", "Complication_risk", "Overweight", "Arthritis",
"Diabetes", "Hyperlipidemia", "BackPain", "Anxiety",
"Allergic_rhinitis", "Reflux_esophagitis", "Asthma",
 "Initial_days"]
readmis_research= mdf[research_cols]
readmis_research.head()
```

- **#Look at info for new data frame of study**

```
readmis_research.info()
```

- **#Create binary list and re-express variables to 0, 1**

```
Binary_columns= ['ReAdmis','HighBlood', 'Stroke', 'Overweight',
'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety',
'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma']
readmis_research[Binary_columns]=readmis_research[Binary_columns].r
eplace({'Yes': 1, 'No': 0})
print(readmis_research)
```

- **#One-hot encode categorical variables using get dummies**

```
readmis_research=pd.get_dummies(readmis_research,
columns=['Marital'], prefix='Marital', drop_first=True);
readmis_research=pd.get_dummies(readmis_research,
columns=['Gender'], prefix='Gender', drop_first=True);
readmis_research=pd.get_dummies(readmis_research,
columns=['Initial_admin'], prefix='Initial_admin', drop_first=True);
readmis_research=pd.get_dummies(readmis_research,
columns=['Complication_risk'], prefix='Complication_risk',
drop_first=True)
```

print(readmis_research)

- **#Look at data info after converting all variables to numeric values**

  readmis_research.info()

- **#Look for correlation in variables with heat map**

  coorl= readmis_research.corr()

  fig, ax = plt.subplots(figsize=(20,15))

  sns.heatmap(coorl, annot=True, ax=ax)

  plt.title("Correlations - Predictors and Response",fontsize=24)

  plt.show()

- **#Export cleaned data**

  readmis_research.to_csv('/Users/robertpatton/Desktop/D209_Task2_prepared.csv', index=False)

4. *Copy of cleaned data attached to submission.

Part IV: Analysis

D. Data Analysis Report

1. The following code was used to split the dataset into train and test sets.
   - **#Identify our independent and dependent variables**

     X= readmis_research.drop(columns='ReAdmis', axis=1)

     y= readmis_research['ReAdmis']

   - **#Split dataset into train and test sets for X,y**

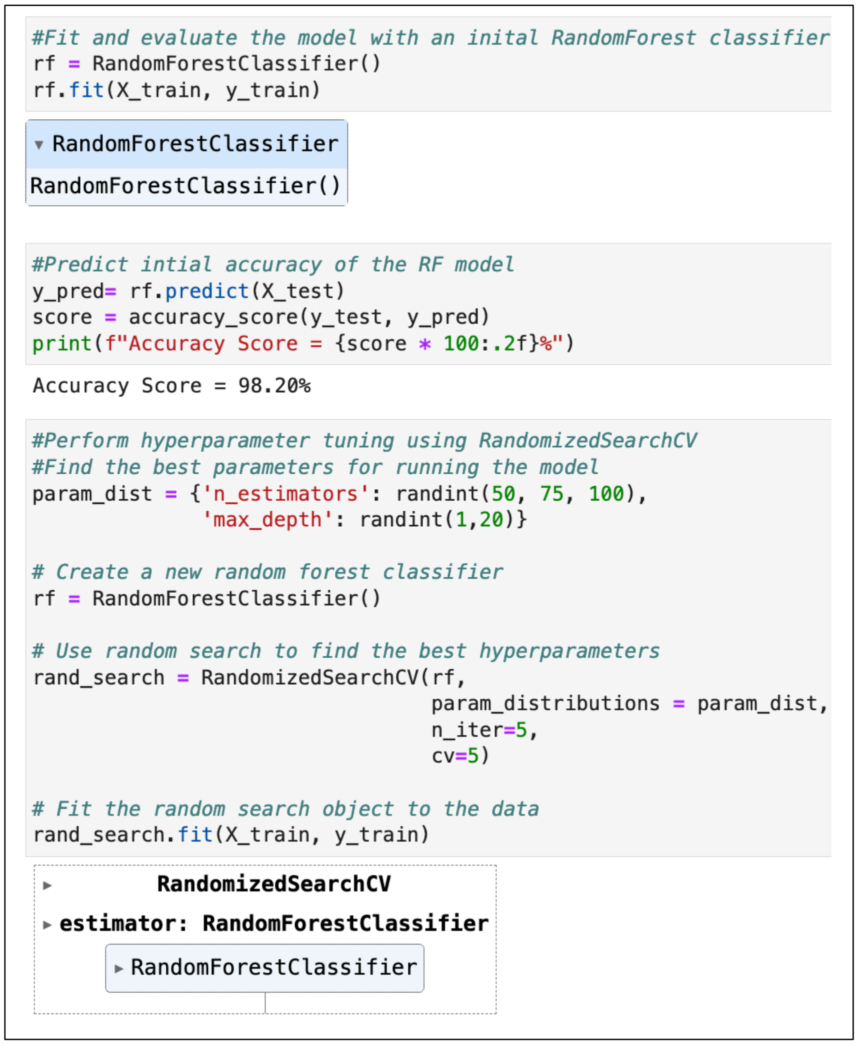
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=20)

   *The files are attached to the submission.

2. With the data split into train and test sets, I was able to then use the random forest classification method to help classify the potential for hospital readmission. Implementing an 80/20 test split, I was able to predict the readmission of patients based on medical conditions as reported during initial admission.

   After splitting the data into training and testing sets, I ran an initial random forest classification model to get a quick look at where the accuracy would be. Although my accuracy from a basic model returned at 98.2%, I decided to attempt to improve my accuracy score by running the model with some hyperparameters using the

RandomizedSearchCV package. By utilizing hyperparameters, depicted below, an analyst can use this machine learning technique to find the best parameters for running the model. In addition, a confusion matrix and classification report were constructed to visualize the output and predictions of the model, they are depicted below as well.



```
#Fit and evaluate the model with an inital RandomForest classifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
#Predict intial accuracy of the RF model
y_pred= rf.predict(X_test)
score = accuracy_score(y_test, y_pred)
print(f"Accuracy Score = {score * 100:.2f}%")
```

```
Accuracy Score = 98.20%
```

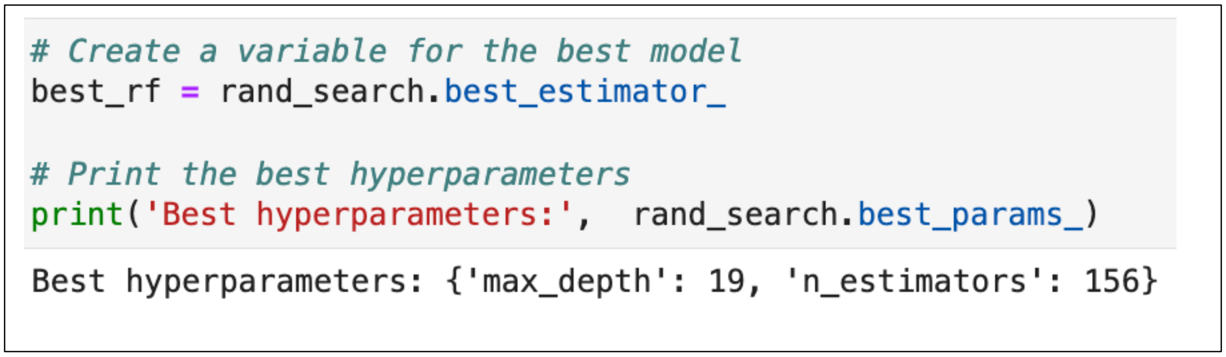
```
#Perform hyperparameter tuning using RandomizedSearchCV
#Find the best parameters for running the model
param_dist = {'n_estimators': randint(50, 75, 100),
              'max_depth': randint(1,20)}

# Create a new random forest classifier
rf = RandomForestClassifier()

# Use random search to find the best hyperparameters
rand_search = RandomizedSearchCV(rf,
                                 param_distributions = param_dist,
                                 n_iter=5,
                                 cv=5)

# Fit the random search object to the data
rand_search.fit(X_train, y_train)
```

```
▸          RandomizedSearchCV
▸ estimator: RandomForestClassifier
      ▸ RandomForestClassifier
```

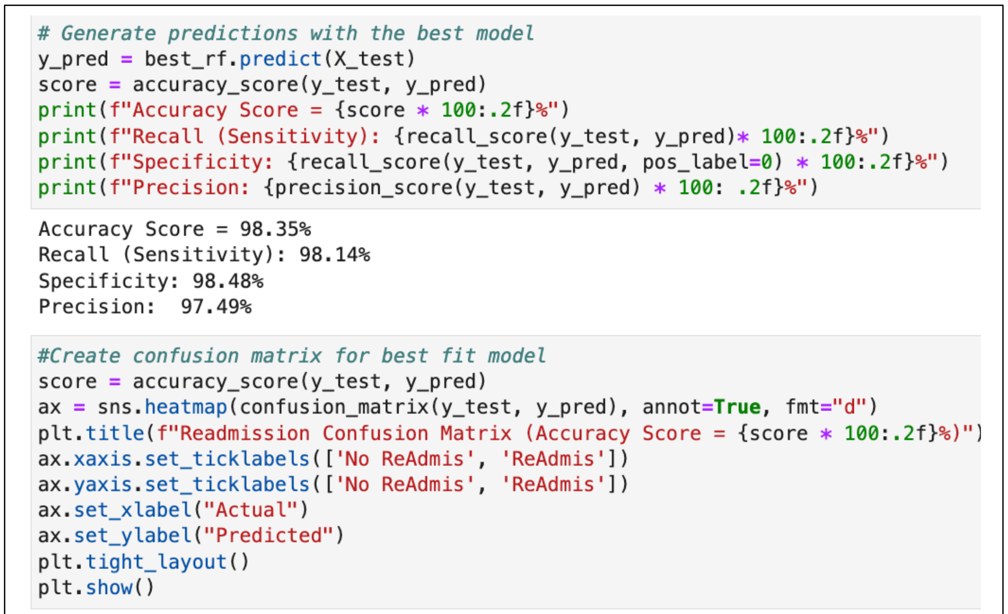


```
# Create a variable for the best model
best_rf = rand_search.best_estimator_

# Print the best hyperparameters
print('Best hyperparameters:',  rand_search.best_params_)
```

```
Best hyperparameters: {'max_depth': 19, 'n_estimators': 156}
```

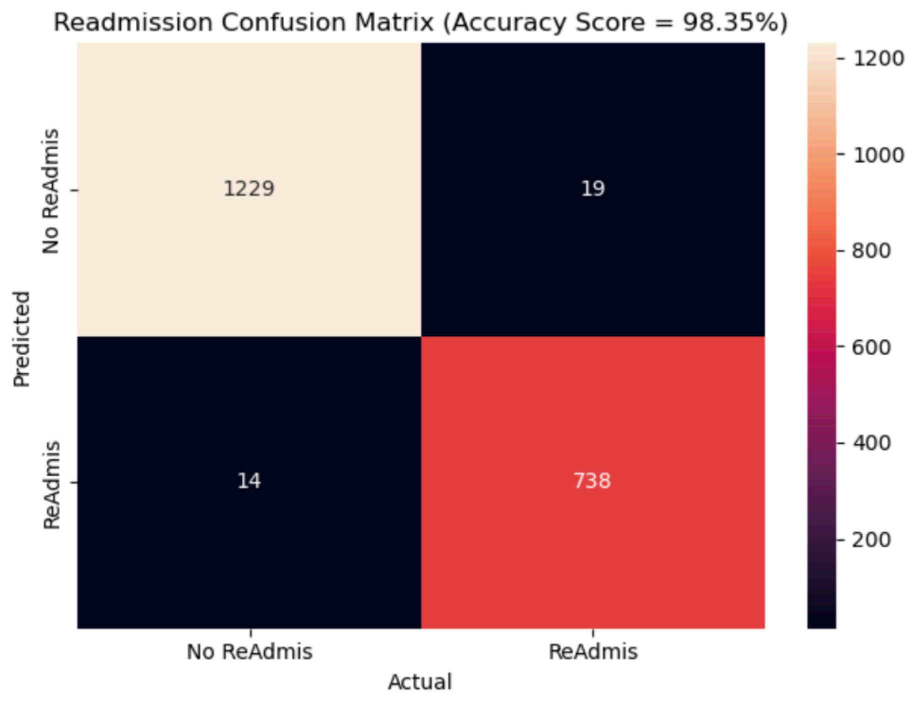
3. The code for predication analysis is shown above and below and is included in the submission:



```
# Generate predictions with the best model
y_pred = best_rf.predict(X_test)
score = accuracy_score(y_test, y_pred)
print(f"Accuracy Score = {score * 100:.2f}%")
print(f"Recall (Sensitivity): {recall_score(y_test, y_pred)* 100:.2f}%")
print(f"Specificity: {recall_score(y_test, y_pred, pos_label=0) * 100:.2f}%")
print(f"Precision: {precision_score(y_test, y_pred) * 100: .2f}%")
```

```
Accuracy Score = 98.35%
Recall (Sensitivity): 98.14%
Specificity: 98.48%
Precision:  97.49%
```

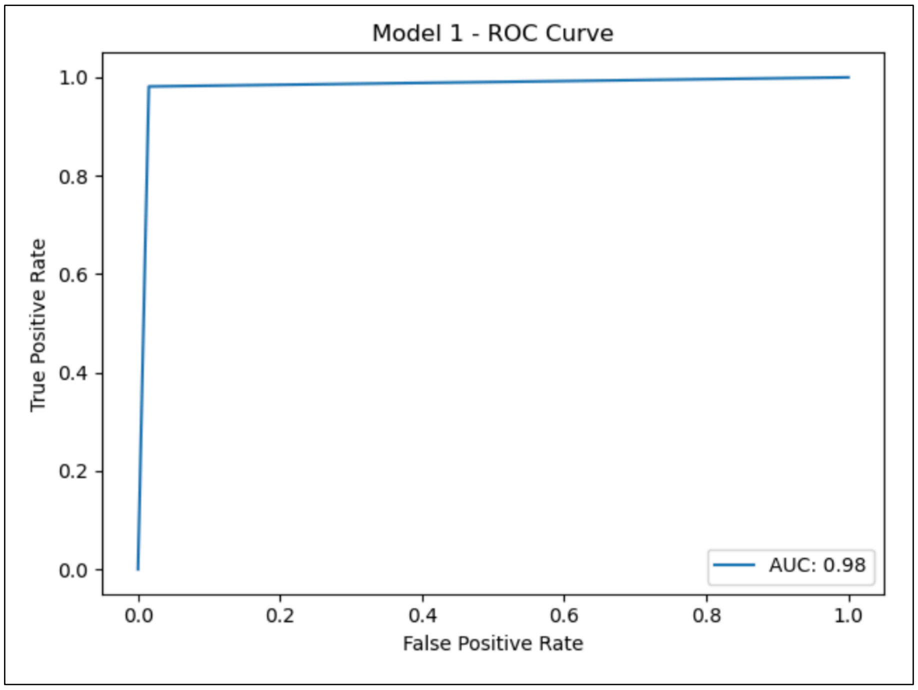
```
#Create confusion matrix for best fit model
score = accuracy_score(y_test, y_pred)
ax = sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d")
plt.title(f"Readmission Confusion Matrix (Accuracy Score = {score * 100:.2f}%)")
ax.xaxis.set_ticklabels(['No ReAdmis', 'ReAdmis'])
ax.yaxis.set_ticklabels(['No ReAdmis', 'ReAdmis'])
ax.set_xlabel("Actual")
ax.set_ylabel("Predicted")
plt.tight_layout()
plt.show()
```



Part V: Data Summary and Implications

E. Summarize the Data Analysis

1. The accuracy of the model comes in at 98.35%, meaning that the model has a 98.35% chance of predicting readmission based on the given independent variables. Accuracy, as seen in the confusion matrix above, is calculated as: True Positives (738) + True Negatives (1229)/ True Positives (738) + True Negatives (1229) + False Positives (19) + False Negatives (14) = 98.35% accuracy of predicting readmission. The mean square error comes in at 0.0165, meaning that the model has a 1.65% false positive prediction rate/error. With these metrics, we can say that the model is very strong at predicting readmission.
2. After determining accuracy, I created a classification report using recall, specificity, and precision to further examine the model's performance. Recall tells an analyst how well the model predicts positives out of all positive values, with the value of recall being 98.14%. Specificity tells an analyst how well the model predicts negative values out of all negatives, with the value coming to 98.48%. Based on the model's recall and specificity results, along with the accuracy, it can be said that the model does well at predicting whether or not patients will be readmitted based on the medical conditions stated at initial admission. I also implemented an ROC curve and got an AUC score to further examine the model's preciseness. The area under the curve looks at the true positive rate (sensitivity) versus the false positive rate (specificity). An AUC score is an overall diagnostic tool for summarizing the accuracy of a test (Mandrekar 2015). In a ROC model, an ideal curve would follow the y-axis straight up to 1.0 and over to 1.0 on the x-axis, indicating 100% accuracy. For this model, the area under the curve is at 98%, indicating an 98% chance of predicting readmission. This is a near perfect performing model. The ROC curve can be seen below.



3. The model has produced very accurate results and signifies that it can predict patient readmissions. However, hospitals tend to see varying degrees of admissions at different times of the year. We may need to collect more data on health conditions, initial admission rates during seasonal changes, and other patient information that could make a stronger case for which patients are at higher risk for readmission. For example, during the winter months when cold and flu season arrives, hospitals see a greater influx in emergency room visits and hospital admissions. Does cold and flu influence or trigger certain health conditions to become worse, requiring admission in the winter months versus other times of the year? These are some things a data analyst and hospital leaders should take into consideration.
4. To follow up on limitations, a recommended course of action would be to closely monitor trends in admission. This dataset seems to list just the most common health conditions, so it may be important to gather data on other health conditions as well. All of that combined will allow the continued accuracy and potentially even stronger accuracy to continue in making predictions on readmission, improving the overall quality of care a patient receives upon initial admission.

Part VI: Demonstration

F. A link to my Panopto video is uploaded to the submission.

G. Third-Party Sources

Shafi, A. (2023, February 24). Random Forest classification with Scikit-Learn. DataCamp. https://www.datacamp.com/tutorial/random-forests-classifier-python

H. Acknowledged Sources

Upadhyay, S., Stephenson, A. L., & Smith, D. G. (2019). Readmission Rates and Their Impact on Hospital Financial Performance: A Study of Washington Hospitals. Inquiry : a journal of medical care organization, provision, and financing, 56, 46958019860386. https://doi.org/10.1177/0046958019860386

Mandrekar, J. (2015, November 20). Receiver operating characteristic curve in diagnostic test assessment. Journal of Thoracic Oncology. https://www.sciencedirect.com/science/article/pii/S1556086415306043#:~:text=AUC%20can%20be%20computed%20using%20the%20trapezoidal%20rule.&text=In%20general%2C%20an%20AUC%20of,than%200.9%20is%20considered%20outstanding

Mendekar, V. (2021, January 22). It's all about assumptions, Pros & Cons. Medium. https://medium.com/swlh/its-all-about-assumptions-pros-cons-497783cfed2d