**D209 Performance Assessment**

**PREDICTIVE ANALYSIS FOR MEDICAL DATA**

Fahim A. Akbar Student ID 001434895 Masters Data Analytics (January 1, 2021) Program Mentor: Lea Yoakem (877) 435-7948x6422 [fakbar3@wgu.edu](mailto:fakbar3@wgu.edu)

**Part I: Research Question**

**A. Purpose of the Data Mining Report**

**1. Question**

Predictive analysis could be used to help answer the following: “Which patients are at a high risk of being readmitted?” For this analysis, the Random Forests method will be used.

**2. Goal**

One goal of this data analysis will be to determine if it is possible to predict the readmission of a

patient based on both categorical and continuous variables.

**Part II: Method Justification**

**B. Reason for using Random Forest**

**1. Random Forest Method**

For this analysis well will be using the Random Forest analysis method. The Random Forest method takes training and testing datasets that are created from a primary dataset, and uses them to run predictive analysis. It uses ensemble learning, where it combines classifiers to provide solutions to problems. It uses multiple decision trees that each consist of feature randomness, and creates an uncorrelated “forest” of trees operating as a committee. The results in more accurate predictions as opposed to only using one decision tree (Yiu 2019). Using Random Forest in this analysis will allow us to predict whether a patient will be readmitted to the hospital with a high level of accuracy.

**2. Random Forest Assumptions**

In terms of distribution, Random Forest has no formal assumptions. It is a non-parametric method that can handle both skewed and multi-model data, in addition to ordinal and non-ordinal categorical data. (Richmond 2016)

**3. List the packages or libraries you have chosen for Python or R, and justify how each**

**item on the list supports the analysis.**

Listed below are the Python packages and libraries that will be used, and how each item supports Random Forest predictive analysis:

* Pandas – this standard import provides methods to read and visualize data. It also offers statistical tools to parse and score data.
* Numpy – this standard import provides methods to read and visualize data. It also offers statistical tools to parse and score data.
* Matplotlib – this package is used for data visualization and will provide more robust tools to visualize reports and data points
* Seaborn – this package will provide us descriptive and visually intuitive graphs, plots, and matrices
* Sklearn – this library contains the RandomForestClassifier, preprocessing and StandardScaler to scale the data for even analysis, create test and train datasets, and create classification reports.
* Scikit-learn – this package will provides method and arguments for splitting, training, testing, and fitting data. This package also has arguments for predicting and classifying data as well as applying metrics for models

**Part III: Data Preparation**

**C. Data Preparation**

**1. Relevant data preprocessing goal**

One data preprocessing goal relevant to the Random Forest method would be scaling the selected variables used and making sure that they are measured evenly. Scaling the variables will enable us to make more accurate predictions for the analysis.

**2. Initial data set variables used to perform the classification analysis**

The target variable for this analysis will be Readmis. This variable is categorical. The continuous predictor variables used will be Age, doctor visits, initial days, total charge, and additional charge. The categorical predictor variables that will be used will be Initial admin, stroke, and complication risk.

**3. Data Preparation Steps**

The first step in preparing the data is to make sure that there are no missing data entries in any of the columns. Next, we will ensure that none of the data in the columns is duplicated. We will make sure that none of the columns or rows are duplicated, to further prevent dealing with repeated entries. For the predictive analysis, several columns in the dataset were deemed irrelevant and were subsequently dropped from the dataset (i.e latitude, longitude). Additionally, the predictor variables will need to be scaled, and the data will need to split into test and training sets. After the predictor variables have been scaled, the “yes/no” entries for the categorical variables will need to be converted to 1 and 0, respectively.

**4. Provide a copy of the cleaned data set.**

A copy of the cleaned dataset is provided with the task submission. Provided below is the code used to create the cleaned and prepared dataset. The full code for the project is provided at the end of this document and is also attached as txt document titled “Full code used for D209 Task 2 Submission.” In addition, a PDF file of the Jupyter notebook created through Python to run the models is provided with the task submissions to showcase the data in its created environment.

**Code:**

#import packages and clean data before running predictive analysis

import numpy as np

import pandas as pd

from sklearn import linear\_model

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import sklearn

from sklearn import datasets

from sklearn.preprocessing import StandardScaler

from sklearn import preprocessing

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import cross\_val\_score, train\_test\_split

from sklearn import metrics

from sklearn.metrics import classification\_report

pd.set\_option('display.max\_columns', None)

df = pd.read\_csv (r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_clean.csv')

df.head()

df.info()

#check if there are any missing data entries

df.isna().any()

#check if there are any duplicate data entries in columns

df[df.duplicated()]

#check if there are any duplicated columns - if there are none than the output should say False

df.columns.duplicated().any()

#check if there are any duplicated rows - if there are none than the output should say False

df.duplicated().any()

#remove the demographic data categories deemed unecessary during our data preparation plan

df = df.drop(['CaseOrder','Customer\_id','Interaction','UID','City','State','County','Zip','Lat','Lng','Population','Area','TimeZone','Job'], axis=1)

#rename the survey columns to their respective titles so that they can be more easily identified

df.rename(columns={'Item1':'Timely\_admis','Item2':'Timely\_treat','Item3':'Timely\_visits','Item4':'Reliability','Item5':'Options','Item6':'Hrs\_treat','Item7':'Courteous','Item8':'Active\_listen'},inplace=True)

#check to see that the survey columns were renamed correctly and that the demographic categories were removed

df.head()

#create a new dataframe for the predictive analysis containing the target variable and chosen predictor variables

medpredict\_df\_new=df[['Age','Doc\_visits','Initial\_days','TotalCharge','Additional\_charges', 'Initial\_admin','Stroke','Complication\_risk','ReAdmis']].copy()

#change the responses for Initial\_admin, Stroke, and Complication\_risk to numeric values and export

medpredict\_df\_new['Initial\_admin'].replace(('Elective Admission','Observation Admission','Emergency Admission'), (0,1,2), inplace=True)

medpredict\_df\_new['Complication\_risk'].replace(('Low','Medium','High'), (0,1,2), inplace=True)

medpredict\_df\_new['Stroke'].replace(('Yes','No'),(1,0),inplace=True)

medpredict\_df\_new.head()

medpredict\_df\_new.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_D209TASK2PREPAREDFINAL.csv', index=False)

#scale the dataset to prepare for the application of Random Forest

predictors = medpredict\_df\_new.columns[medpredict\_df\_new.dtypes.apply(lambda c: np.issubdtype(c, np.number))]

scaler=StandardScaler()

medpredict\_df\_new[predictors] = scaler.fit\_transform(medpredict\_df\_new[predictors])

#convert the values of the target variable into numeric variables

medpredict\_df\_new['ReAdmis']=df.ReAdmis.map(dict(Yes=1, No=0))

#show scaled data to make sure that everything was done correctly

medpredict\_df\_new.head()

**Part IV: Analysis**

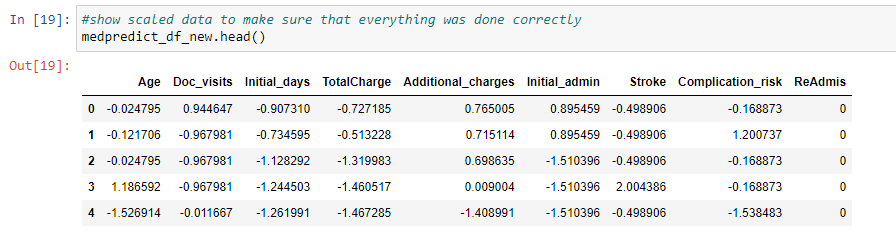
**D. Data Analysis**

**1. Training and Testing Sets**

A training and a testing data set were created by using 20% of the data for testing and 80% for training. Afterwards, the data is run through the random forest classifiers, setting the parameters appropriately, as expressed in the code. Copies of the training and testing datasets are provided with the task submission.

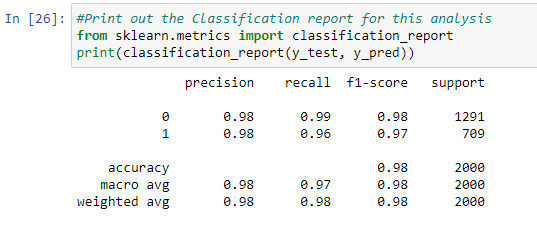
**2. Analysis Technique**

Before analysis is conducted, the variables in the new dataframe is scaled so that they can be measured evenly. After scaling the variables, the values of the predictor variable (categorical) are changed from text to numerical (Yes to 1, No to 0). The scaled data in the dataframe is provided below:

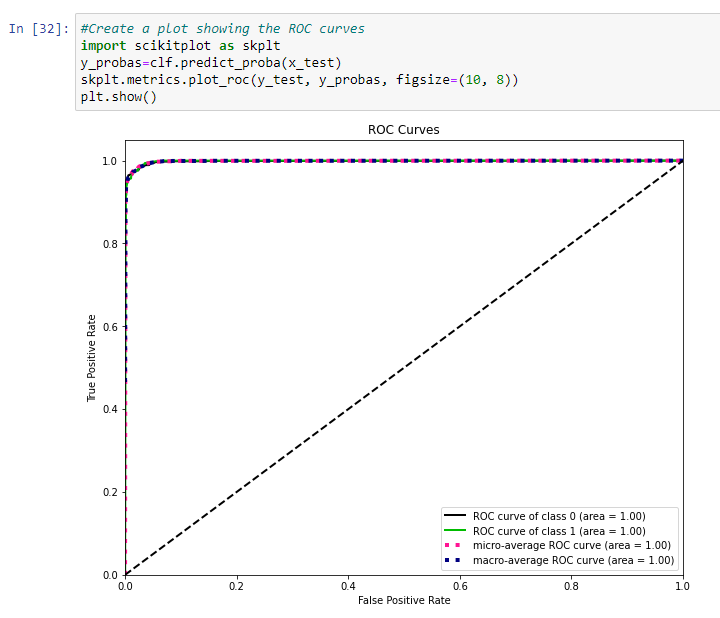


The scaled data now needs to be converted to training and testing datasets. This will in turn help the algorithm make decisions make predictions based on the training data. Here, the Random Forest Classifier is demonstrated, establishing classification parameters. For this analysis, the n\_estimators is set to 100. This classifier is then fitted to the training data, and predictions can now be created by the algorithm and compared to the actual values. We can now run the classification report to show the precision, F1-score, and accuracy of the model. Provided below is an example of these:





Visualizing the ROC Curve can be used to show the true positive vs. the false positive rate, as a higher curve demonstrates the model has a high accuracy for predicting outcomes. The ROC curve demonstrated below shows high values, indicating that the model is very strong at predictive true positive values.



**3. Code used to perform the predictive analysis**

#to begin predictive analysis, seperate data into testing and training datasets

train , test = train\_test\_split(medpredict\_df\_new,test\_size=0.20, random\_state=42)

x\_train=train.drop('ReAdmis',axis=1)

y\_train=train['ReAdmis']

x\_test=test.drop('ReAdmis',axis=1)

y\_test=test['ReAdmis']

#export the testing and training files

x\_train.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_xtrain2final.csv', index = False)

x\_test.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_xtest2final.csv', index = False)

y\_train.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_ytrain2final.csv', index = False)

y\_test.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_ytest2final.csv', index = False)

#Import RandomForestClassifier and create the model to run this analysis

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(bootstrap = True, class\_weight = None, criterion = 'gini', max\_depth = None, max\_features = 'auto', max\_leaf\_nodes = None, min\_impurity\_decrease = 0.0, min\_samples\_leaf = 1, min\_samples\_split = 2, min\_weight\_fraction\_leaf = 0.0, n\_estimators = 100, n\_jobs = 1, oob\_score = False, random\_state = None, verbose = 0, warm\_start = False)

clf.fit(x\_train,y\_train)

#Save the created predictions of the model

y\_pred=clf.predict(x\_test)

#Now show the predictions vs actual values

pd.DataFrame(data={'Predicted': y\_pred, 'Actual': y\_test}).head(15)

#Print out the Classification report for this analysis

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred))

#Print the accuracy score; the desired score should be 95% or above

from sklearn.metrics import accuracy\_score

accuracy = accuracy\_score(y\_test, y\_pred)

print(accuracy)

#Print the recall score

from sklearn.metrics import recall\_score

recall\_score(y\_test, y\_pred, average='weighted')

#Print precision Score

from sklearn.metrics import precision\_score

precision\_score(y\_test, y\_pred, average='weighted')

#Print F1 Score

from sklearn.metrics import f1\_score

f1\_score(y\_test, y\_pred, average='weighted')

#Create a plot showing the ROC curves

import scikitplot as skplt

y\_probas=clf.predict\_proba(x\_test)

skplt.metrics.plot\_roc(y\_test, y\_probas, figsize=(10, 8))

plt.show()

#Determine the AUC

from sklearn import preprocessing

from sklearn.metrics import roc\_auc\_score

def aucScore(y\_test, y\_pred, average="weighted"):

lb = preprocessing.LabelBinarizer()

lb.fit(y\_test)

y\_test = lb.transform(y\_test)

y\_pred = lb.transform(y\_pred)

return roc\_auc\_score(y\_test, y\_pred, average=average)

#Print the Area Under Curve value

aucScore(y\_test, y\_pred)

#Calculate and Print the Mean Squared Error value

from sklearn.metrics import mean\_squared\_error

mean\_squared\_error(y\_test,y\_pred)

**Part V: Data Summary and Implications**

**E. Data Analysis Summary:**

**1. Accuracy and the mean squared error (MSE)**

The model had an accuracy score of 0.979, which means that it has a 97.9% chance of correctly predicting if a patient will be readmitted based on the identified variables. Furthermore, the MSE for this model is 0.021, meaning that it has a 2.1% false positive prediction rate or error. Therefore, this model is a very strong at predicting whether a patient will be readmitted based on the identified variables.

**2. Results and implications of the prediction analysis**

The results show that the currently model has a 97-98% positive prediction rate. This is based on the classification report value and the Area Under Curve values, as follows:

* Accuracy = 0.9790
* Recall = 0.979
* Precision = 0.979
* F1 = 0.9789
* AUC = 0.9748
* MSE = 0.021

The MSE is 0.021 or 2.1%, demonstrating a very low false positive which therefore means the model has few errors predicting outcomes. Therefore, the model we used for predicting a patient’s chance of readmission is very accurate and precise with a low margin of error. Although this model is highly effective and can be used for strong predictions by the hospital, it is still not 100% accurate. Even with a high level of accuracy, it is advised that the hospital continues to use some level of human intervention to further verify the results. Fortunately, this would still require minimal effort and resources, since our accuracy rate of 97-98% is higher than the 95% accuracy rate which is considered successful under most circumstances

**3. Limitations of the analysis**

A limitation of this data analysis is that it utilizes and relies on one model, being the Random Forest Classifier. While this model is stronger than a single decision true, running additional models would be beneficial to create a higher sense of confidence in the data analysis. Another limitation would be the number of estimators used in the analysis, as a higher or lower number of estimators would impact the accuracy and precision of the model.

**4. Recommended course of action based on analysis results**

For the hospital, a recommended course of action would be to use this model as a strong starting point for predicting the possibility of readmission for patients. They can then compare these variables and outcomes to future patients, and this can help the hospital identify early signs and indications of these future patients’ chances of being readmitted. Early identification will help the hospital staff better prepare treatment plans and protocols which in turn could also help prevent further readmissions. Additionally, hospital personal can focus more time and resources on the variables identified as the strongest predictors of readmission to help cater their treatment plans and protocols. Finally, additional analyses should be preformed to further demonstrate that the identified variables are indeed strong indictors, and verify that the current model’s accuracy is not a result of overfitting the data. Running multiple analyses and taking the summary of their results will in turn create the strongest and most accurate predictive model possible.

**Part VI: Demonstration and Supporting Documents**

**Link to the Panopto Video recording:**

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=ca82f651-32bc-47dd-a7b3-aec000f1f2d3>

**Sources for third party code:**

**Code for importing packages and performing random forest:**

**Github:** <https://notebooks.githubusercontent.com/view/ipynb?azure_maps_enabled=false&color_mode=auto&commit=efa81de07f8e67225ab0ff87c49caeb58d1ed019&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f676973742f56696b72616e7437392f64313464613735343161333634653632333262372f7261772f656661383164653037663865363732323561623066663837633439636165623538643165643031392f466c61766f722532306f66253230506879736963732532302d2532304d69782532306f662532306d6f64656c732e6970796e62&enterprise_enabled=false&logged_in=false&nwo=Vikrant79%2Fd14da7541a364e6232b7&path=Flavor+of+Physics+-+Mix+of+models.ipynb&repository_id=26076758&repository_type=Gist>

**Kaggle:**

<https://www.kaggle.com/code/clarecao/notebookmodelperformancehomework>

**Code used to tune parameters in Random Forest:**

“How to Tune Parameters in Random Forest, Using Scikit Learn?” *Stack Overflow*, 19 Mar. 2016, stackoverflow.com/questions/36107820/how-to-tune-parameters-in-random-forest-using-scikit-learn.

<https://stackoverflow.com/questions/36107820/how-to-tune-parameters-in-random-forest-using-scikit%20learn>

**References:**

Yiu, Tony. “Understanding Random Forest - Towards Data Science.” *Medium*, 10 Dec. 2021, towardsdatascience.com/understanding-random-forest-58381e0602d2.

Richmond, S. “Algorithms exposed: Random Forest. BCCVL.” 21 March 2016, https://bccvl.org.au/algorithms-exposed-randomforest/#:~:text=ASSUMPTIONS,are%20ordinal%20or%20non%2Dordinal

**Full Code used for this Project**

#import packages and clean data before running predictive analysis

import numpy as np

import pandas as pd

from sklearn import linear\_model

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import sklearn

from sklearn import datasets

from sklearn.preprocessing import StandardScaler

from sklearn import preprocessing

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import cross\_val\_score, train\_test\_split

from sklearn import metrics

from sklearn.metrics import classification\_report

pd.set\_option('display.max\_columns', None)

df = pd.read\_csv (r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_clean.csv')

df.head()

df.info()

#check if there are any missing data entries

df.isna().any()

#check if there are any duplicate data entries in columns

df[df.duplicated()]

#check if there are any duplicated columns - if there are none than the output should say False

df.columns.duplicated().any()

#check if there are any duplicated rows - if there are none than the output should say False

df.duplicated().any()

#remove the demographic data categories deemed unecessary during our data preparation plan

df = df.drop(['CaseOrder','Customer\_id','Interaction','UID','City','State','County','Zip','Lat','Lng','Population','Area','TimeZone','Job'], axis=1)

#rename the survey columns to their respective titles so that they can be more easily identified

df.rename(columns={'Item1':'Timely\_admis','Item2':'Timely\_treat','Item3':'Timely\_visits','Item4':'Reliability','Item5':'Options','Item6':'Hrs\_treat','Item7':'Courteous','Item8':'Active\_listen'},inplace=True)

#check to see that the survey columns were renamed correctly and that the demographic categories were removed

df.head()

#create a new dataframe for the predictive analysis containing the target variable and chosen predictor variables

medpredict\_df\_new=df[['Age','Doc\_visits','Initial\_days','TotalCharge','Additional\_charges', 'Initial\_admin','Stroke','Complication\_risk','ReAdmis']].copy()

#change the responses for Initial\_admin, Stroke, and Complication\_risk to numeric values and export

medpredict\_df\_new['Initial\_admin'].replace(('Elective Admission','Observation Admission','Emergency Admission'), (0,1,2), inplace=True)

medpredict\_df\_new['Complication\_risk'].replace(('Low','Medium','High'), (0,1,2), inplace=True)

medpredict\_df\_new['Stroke'].replace(('Yes','No'),(1,0),inplace=True)

medpredict\_df\_new.head()

medpredict\_df\_new.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_D209TASK2PREPAREDFINAL.csv', index=False)

#scale the dataset to prepare for the application of Random Forest

predictors = medpredict\_df\_new.columns[medpredict\_df\_new.dtypes.apply(lambda c: np.issubdtype(c, np.number))]

scaler=StandardScaler()

medpredict\_df\_new[predictors] = scaler.fit\_transform(medpredict\_df\_new[predictors])

#convert the values of the target variable into numeric variables

medpredict\_df\_new['ReAdmis']=df.ReAdmis.map(dict(Yes=1, No=0))

#show scaled data to make sure that everything was done correctly

medpredict\_df\_new.head()

#to begin predictive analysis, seperate data into testing and training datasets

train , test = train\_test\_split(medpredict\_df\_new,test\_size=0.20, random\_state=42)

x\_train=train.drop('ReAdmis',axis=1)

y\_train=train['ReAdmis']

x\_test=test.drop('ReAdmis',axis=1)

y\_test=test['ReAdmis']

#export the testing and training files

x\_train.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_xtrain2final.csv', index = False)

x\_test.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_xtest2final.csv', index = False)

y\_train.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_ytrain2final.csv', index = False)

y\_test.to\_csv(r'C:\Users\fahim\Documents\0\_WGUDocuments\d209\medical\_ytest2final.csv', index = False)

#Import RandomForestClassifier and create the model to run this analysis

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(bootstrap = True, class\_weight = None, criterion = 'gini', max\_depth = None, max\_features = 'auto', max\_leaf\_nodes = None, min\_impurity\_decrease = 0.0, min\_samples\_leaf = 1, min\_samples\_split = 2, min\_weight\_fraction\_leaf = 0.0, n\_estimators = 100, n\_jobs = 1, oob\_score = False, random\_state = None, verbose = 0, warm\_start = False)

clf.fit(x\_train,y\_train)

#Save the created predictions of the model

y\_pred=clf.predict(x\_test)

#Now show the predictions vs actual values

pd.DataFrame(data={'Predicted': y\_pred, 'Actual': y\_test}).head(15)

#Print out the Classification report for this analysis

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred))

#Print the accuracy score; the desired score should be 95% or above

from sklearn.metrics import accuracy\_score

accuracy = accuracy\_score(y\_test, y\_pred)

print(accuracy)

#Print the recall score

from sklearn.metrics import recall\_score

recall\_score(y\_test, y\_pred, average='weighted')

#Print precision Score

from sklearn.metrics import precision\_score

precision\_score(y\_test, y\_pred, average='weighted')

#Print F1 Score

from sklearn.metrics import f1\_score

f1\_score(y\_test, y\_pred, average='weighted')

#Create a plot showing the ROC curves

import scikitplot as skplt

y\_probas=clf.predict\_proba(x\_test)

skplt.metrics.plot\_roc(y\_test, y\_probas, figsize=(10, 8))

plt.show()

#Determine the AUC

from sklearn import preprocessing

from sklearn.metrics import roc\_auc\_score

def aucScore(y\_test, y\_pred, average="weighted"):

lb = preprocessing.LabelBinarizer()

lb.fit(y\_test)

y\_test = lb.transform(y\_test)

y\_pred = lb.transform(y\_pred)

return roc\_auc\_score(y\_test, y\_pred, average=average)

#Print the Area Under Curve value

aucScore(y\_test, y\_pred)

#Calculate and Print the Mean Squared Error value

from sklearn.metrics import mean\_squared\_error

mean\_squared\_error(y\_test,y\_pred)