Briana Churchill

Student ID: 011009463

Dr. Eric Straw

November 13, 2023

NVM2 TASK 2: PREDICTIVE ANALYSIS

Part I: Research Question

A1. Question and Method Used to Analyze

In a past analysis, a k-NN model was created to assess predictability of churn. Instead of focusing on customers who have left the business, now I will assess variables related to current customers. For this analysis, the research question at hand is “using a Random Forest model, can a customer’s add-on services and demographics predict which internet service they will use?” The predictions made could allow the business to offer promotions or advertising specified to customers having factors related to likelihood of specific service choices.

A2. Goal of the Data Analysis

Both the logistic and linear regression models in previous assessments were rendered ineffective and unreliable. The k-nearest neighbors method was promising in my most recent assessment, so I am hoping for a continued trend with the Random Forest classifier model. The goal of this analysis is to make reliable predictions related to customer internet service choices. There is a secondary goal in mind as well, to use the reliability of the predictions to recommend a course of action other than reassessing the question like in previous analyses.

Part II: Method Justification

B1. Chosen Prediction Method and Expected Outcomes

There can be many reasons as to why a customer may choose one internet service over another. Using a random forest classifier model allows a multitude of classifications to be assessed from many different viewpoints because “multiple decision trees are created using different random subsets of the data and features.” (Shafi, 2023) Each of these trees in the forest have their own predictions. Using the data from each of these trees, the most popular result is used to make predictions. The expected outcome for this analysis is that the model will make predictions on what kind of internet service type a customer will choose given specific demographic and/or add-on services.

B2. Summarize one assumption of the chosen prediction method

According to Simplilearn.com, "There should be some actual values in the feature variables of the dataset, which will give the classifier a better chance to predict accurate results, rather than provide an estimation. Missing values should be handled from training the model." (M, 2023) Fortunately, the data has been thoroughly cleaned and vetted in previous assessments so maintaining the data cleaning methods used previously will address any missing values within the dataset.

B3. Packages or libraries used in Python

The packages and libraries used and their importance to the analysis is described below:

- NumPy
    - Transforms the data through mathematical equations.
- Pandas
    - Provides a logical structure, otherwise known as the data frame
    - Used to create dummy variables
    - Also used to save the data to CSV
- Sklearn
    - Splits the data into train and test sets (train test split)
    - Facilitates Random Forest model (RandomForestClassifier)
    - Used to calculate:
        - Accuracy score
        - Mean squared error
        - F1 score
        - Confusion Matrix
    - Execute hyperparameter tuning (RandomizedSearchCV)
    - Visualize/display the first three trees in the forest, as well as the confusion matrix
    - Determine the area under the curve score (ROC [curve] & AUC score)
- SciPy
    - Additional hyperparameter tuning (Rendit)
- Matplotlib_pyplot
    - Used to plot the ROC curve

Part III: Data Preparation

C1. One data preprocessing goal

Having completed a few analyses by now, one data preparation goal for this assessment is to "water down" the names within the data. In the past I have maintained long names (i.e.: Bandwidth_GB_Year) and I have found it increasingly more difficult to understand with one hot encoding and more advanced models. As such, my goal for this analysis is to make the column names and their data points as short and simple as possible.

C2. Initial variables and their classification

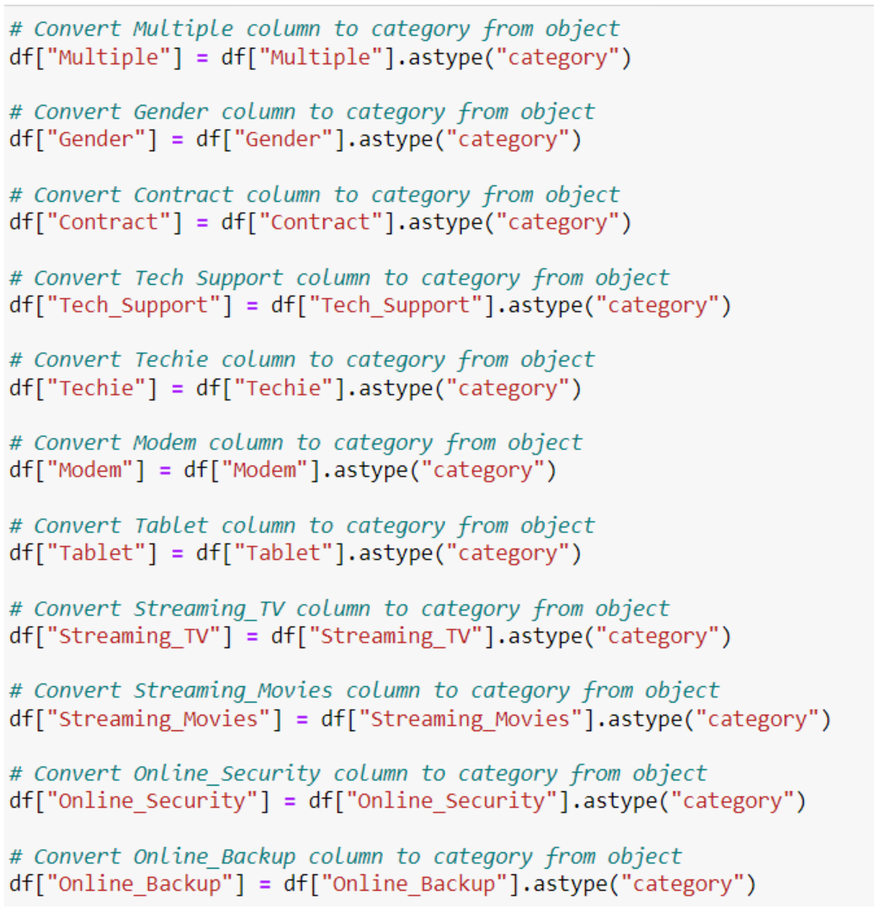
Categorical
Gender
Area
Phone
Tech Support
Contract
Multiple
Internet Service (target variable)
Online Backup
Techie
Modem
Tablet
Online Security
Streaming TV
Streaming Movies

Continuous/Numeric
Children
Equipment Failures
Age
Income
Tenure
Bandwidth Usage
Monthly Charge
Service Outages

C3. Explain the steps used to prepare the data and identify the code segment for *each* step

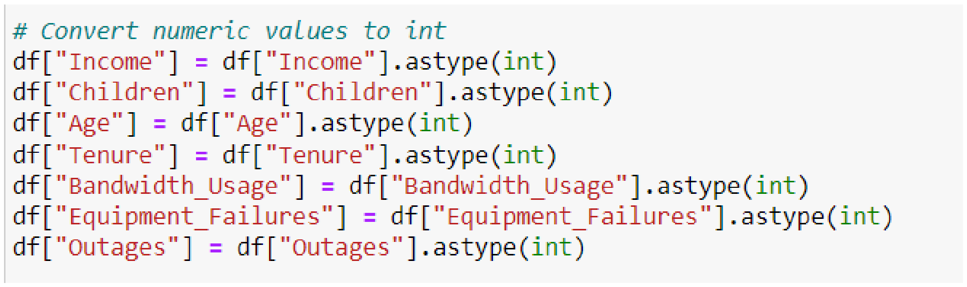
After cleaning the data using scripts used in D207, D208 and D209 task 1, I altered the data a bit further to prepare it for the analysis. Starting with code previously used to convert column data types, map values, create dummy variables, and lastly to drop unnecessary variables.

The columns with categorical variables were all converted to data type category. The variables and code used can be seen in the image here:



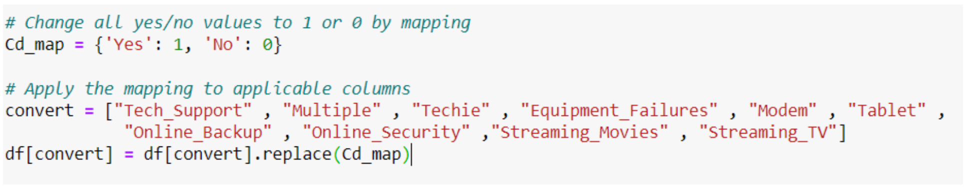
```
# Convert Multiple column to category from object
df["Multiple"] = df["Multiple"].astype("category")

# Convert Gender column to category from object
df["Gender"] = df["Gender"].astype("category")

# Convert Contract column to category from object
df["Contract"] = df["Contract"].astype("category")

# Convert Tech Support column to category from object
df["Tech_Support"] = df["Tech_Support"].astype("category")

# Convert Techie column to category from object
df["Techie"] = df["Techie"].astype("category")

# Convert Modem column to category from object
df["Modem"] = df["Modem"].astype("category")

# Convert Tablet column to category from object
df["Tablet"] = df["Tablet"].astype("category")

# Convert Streaming_TV column to category from object
df["Streaming_TV"] = df["Streaming_TV"].astype("category")

# Convert Streaming_Movies column to category from object
df["Streaming_Movies"] = df["Streaming_Movies"].astype("category")

# Convert Online_Security column to category from object
df["Online_Security"] = df["Online_Security"].astype("category")

# Convert Online_Backup column to category from object
df["Online_Backup"] = df["Online_Backup"].astype("category")
```

After the categorical variables were taken care of, all the numeric variable columns were converted to data type integer, all of which can be seen here:



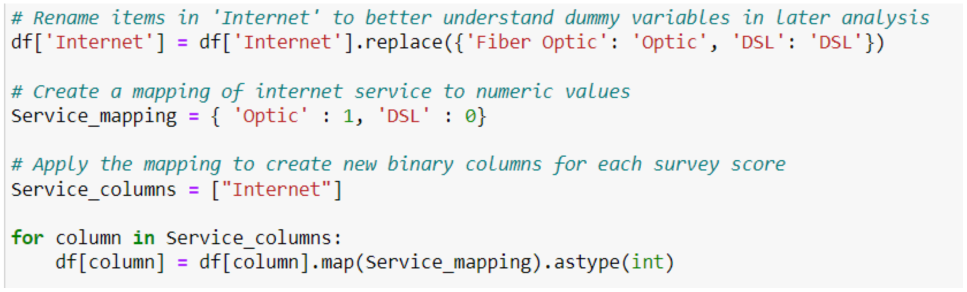
```
# Convert numeric values to int
df["Income"] = df["Income"].astype(int)
df["Children"] = df["Children"].astype(int)
df["Age"] = df["Age"].astype(int)
df["Tenure"] = df["Tenure"].astype(int)
df["Bandwidth_Usage"] = df["Bandwidth_Usage"].astype(int)
df["Equipment_Failures"] = df["Equipment_Failures"].astype(int)
df["Outages"] = df["Outages"].astype(int)
```

Now that all data types have been converted, categorical variables with yes/no data points were mapped to 1 or 0. The applicable variables are Tech Support, Multiple, Techie, Equipment Failures, Modem, Tablet, Online Backup, Online Security, Streaming Movies, and Streaming TV. The scripts used to implement this step is as follows:



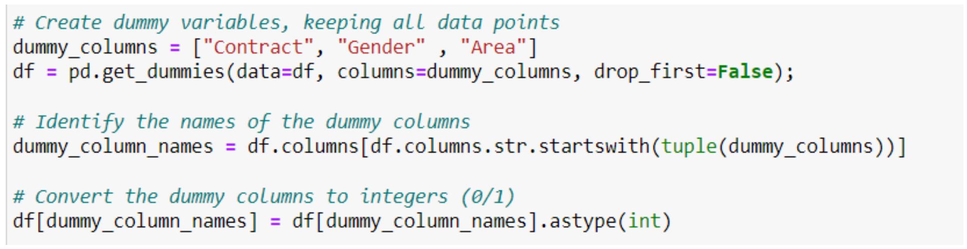
```python
# Change all yes/no values to 1 or 0 by mapping
Cd_map = {'Yes': 1, 'No': 0}

# Apply the mapping to applicable columns
convert = ["Tech_Support" , "Multiple" , "Techie" , "Equipment_Failures" , "Modem" , "Tablet" ,
           "Online_Backup" , "Online_Security" ,"Streaming_Movies" , "Streaming_TV"]
df[convert] = df[convert].replace(Cd_map)
```

Next, to make understanding of the differing internet service types easier in later analysis, I renamed the variables within the internet services column. Mainly to rename the two-worded data point “Fiber Optic” to “Optic.” The steps are shown below.



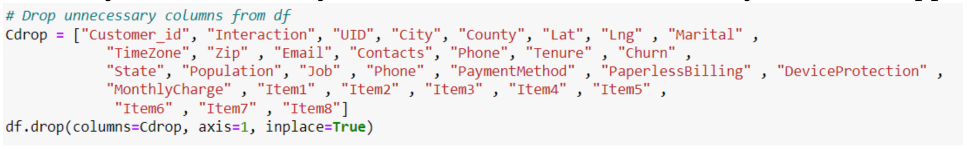
```python
# Rename items in 'Internet' to better understand dummy variables in later analysis
df['Internet'] = df['Internet'].replace({'Fiber Optic': 'Optic', 'DSL': 'DSL'})

# Create a mapping of internet service to numeric values
Service_mapping = { 'Optic' : 1, 'DSL' : 0}

# Apply the mapping to create new binary columns for each survey score
Service_columns = ["Internet"]

for column in Service_columns:
    df[column] = df[column].map(Service_mapping).astype(int)
```

For the remaining categorical variables, I implemented one-hot encoding to create dummy variables. The affected columns are Contract, Gender, and Area. The scripts used to implement this:



```python
# Create dummy variables, keeping all data points
dummy_columns = ["Contract", "Gender" , "Area"]
df = pd.get_dummies(data=df, columns=dummy_columns, drop_first=False);

# Identify the names of the dummy columns
dummy_column_names = df.columns[df.columns.str.startswith(tuple(dummy_columns))]

# Convert the dummy columns to integers (0/1)
df[dummy_column_names] = df[dummy_column_names].astype(int)
```

Lastly, all unnecessary columns not utilized in the analysis were dropped from the data frame.



```python
# Drop unnecessary columns from df
Cdrop = ["Customer_id", "Interaction", "UID", "City", "County", "Lat", "Lng" , "Marital" ,
         "TimeZone", "Zip" , "Email", "Contacts", "Phone", "Tenure" , "Churn" ,
         "State", "Population", "Job" , "Phone" , "PaymentMethod" , "PaperlessBilling" , "DeviceProtection" ,
         "MonthlyCharge" , "Item1" , "Item2" , "Item3" , "Item4" , "Item5" ,
          "Item6" , "Item7" , "Item8"]
df.drop(columns=Cdrop, axis=1, inplace=True)
```

C4. Copy of the cleaned data set

The cleaned data set CSV file associated with this analysis is attached to the submission of this assessment.

Part IV: Analysis

D1. Split the data into training and test data sets and provide the files

The split data sets and a CSV file for each associated with this analysis are attached to this assessment's submission.

D2. Describe the analysis technique used

With a mixture of scripts and instruction provided by DataCamp, Simplilearn, Scikit learn, and Geeksforgeeks (cited below), I attempted the random forest classifier model. After trial and error, I used snippets of code and information from all sources to come to the best scripts for the data I was using. After splitting the data as previously discussed, I created the classification model and fit the split data into the model. With information learned through Dr. Straw's course tips, I learned that hyperparameter tuning could be accomplished utilizing "RandomizedSearchCV," so this was my next step. (Straw, 2023) The RandomizedSearchCV randomly tests a set of hyperparameters and scores their output. Finally, the search gives the best set of hyperparameters with the best score as an output. The hyperparameters used in the final model were a max depth of 11, and 254 estimators. Max depth indicates the number of splits that each tree can make, this case being 11. The number of estimators is the number of trees in the forest model, in this case being 254. Following the hyperparameter tuning, a "best" model was created, and I was able to make additional predictions.

After predictions were made with the model, I reviewed the mean squared error with information learned from TowardsDataScience. (Beheshti, 2022) Using knowledge and scripts from previous assessments I have submitted (Churchill, 2023) further analysis of the model's outputs was done by obtaining and reviewing the accuracy score, F1 score and the ROC/AUC score. In addition, a confusion matrix was created and displayed.

D3. Provide the code used to perform the prediction analysis

The scripts for these steps and their outputs can be seen in the screengrabs below:

[14]:
```python
# Evaluate the model using accuracy, f1, MSE, confusion matrix and roc/auc
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
cmat = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, rf.predict_proba(X_test)[:, 1])
# Print scores
print("Accuracy:", accuracy)
print("F1 Score:", f1)
print(f"Mean Squared Error: {mse}");
print("Confusion Matrix:\n", cmat)
print("ROC AUC:", roc_auc)
```

```
Accuracy: 0.6232006773920407
F1 Score: 0.6903270702853167
Mean Squared Error: 0.37679932260795934
Confusion Matrix:
 [[480 544]
 [346 992]]
ROC AUC: 0.6618670590433483
```

[15]:
```python
# Complete hyperparameter Tuning to create final model
param_dist = {'n_estimators': randint(50,500),
              'max_depth': randint(1,20)}

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)
```

[15]:
```
▸         RandomizedSearchCV
▸ estimator: RandomForestClassifier
     ▸ RandomForestClassifier
```

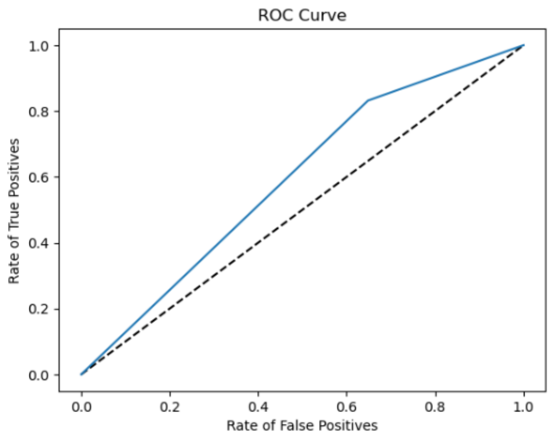
[16]:
```python
# Create a variable for the best model
Final_best = rand_search.best_estimator_

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

```
Best hyperparameters: {'max_depth': 11, 'n_estimators': 254}
```

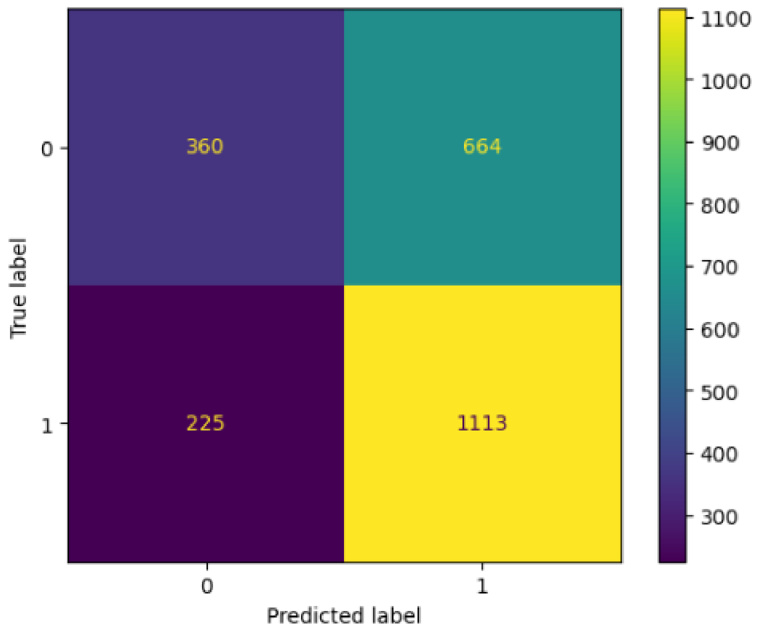
```
[18]: # Evaluate the final model using accuracy, f1, MSE confusion matrix and roc/auc
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
cmat = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, Final_best.predict_proba(X_test)[:, 1])
mse = mean_squared_error(y_test, y_pred)
# Print scores
print("Accuracy:", accuracy)
print("F1 Score:", f1)
print(f"Mean Squared Error: {mse}");
print("Confusion Matrix:\n", cmat)
print("ROC AUC:", roc_auc)
```

```
Accuracy: 0.6236240474174428
F1 Score: 0.7146067415730336
Mean Squared Error: 0.3763759525825571
Confusion Matrix:
 [[ 360  664]
 [ 225 1113]]
ROC AUC: 0.6522656541946936
```

```
[19]: # Plot ROC Curve for reduced model
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr)
plt.xlabel('Rate of False Positives')
plt.ylabel('Rate of True Positives')
plt.title('ROC Curve')
plt.show()
```



```
[20]: # Create the confusion matrix
cm = confusion_matrix(y_test, y_pred)
# Display confusion matrix
ConfusionMatrixDisplay(confusion_matrix=cm).plot();
```



Part V: Data Summary and Implications

E1. Accuracy and Mean Squared Error (MSE) of predictive model

Addressed above, to assess the accuracy of the random forest model, I obtained the accuracy, F1, MSE and ROC/AUC scores. These scores were obtained with the initial model, and with the final model after the hyperparameter tuning. The results can be seen in the table below.

Assessment	Initial Model	Final Model
Accuracy	0.623	0.623
F1	0.690	0.714
Mean Squared Error (MSE)	0.376	0.376
ROC AUC	0.661	0.652

As can be seen under the "Assessment" column, the values obtained to assess the predictive models explained in the table above are:

- Accuracy
    - This value indicates that the initial model correctly predicted 62.3% of the total instances; this was the same value in the final model as well.

- F1
    - This score is the “harmonic mean” of precision and recall.
    - Evaluates false positives and false negatives and is used for imbalanced datasets.
    - The higher the F1 score, the better the balance is between precision and recall. (2023, Kumar)
    - The initial model has an F1 score of .690, the final has a higher score of 0.714.

- Mean Squared Error
    - This value is the average of the squared difference between the predicted values and actual values. The lower the better. (Beheshti, 2022)
    - For the initial and final model, the MSE was 0.376
- ROC/AUC
    - The ROC and AUC, otherwise AUROC, tells how well the model can distinguish between classes. (Narkhede, 2018)
    - The higher the AUC score, the better the model is at predicting positives as positive, and negatives as negative.
    - The AUROC score for the initial model is 0.661, and 0.652 for the final model.

In the initial confusion matrix, the values 480, 544, 346 and 992 are seen. These values indicate that the model has made the following predictions:

480 instances correctly classified as positive.
544 instances incorrectly classified as positive (false positives).
346 instances incorrectly classified as negative (false negatives).
992 instances correctly classified as negative.

In the final confusion matrix, the values 360, 664, 225 and 1113 are seen. These values indicate that the model has made the following predictions:

360 instances correctly classified as positive.
664 instances incorrectly classified as positive (false positives).
225 instances incorrectly classified as negative (false negatives).
1113 instances correctly classified as negative.

The scripts and output for each of these evaluation metrics can be seen in the screengrabs below:

[14]:
```
# Evaluate the model using accuracy, f1, MSE, confusion matrix and roc/auc
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
cmat = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, rf.predict_proba(X_test)[:, 1])
# Print scores
print("Accuracy:", accuracy)
print("F1 Score:", f1)
print(f"Mean Squared Error: {mse}");
print("Confusion Matrix:\n", cmat)
print("ROC AUC:", roc_auc)
```

```
Accuracy: 0.6232006773920407
F1 Score: 0.6903270702853167
Mean Squared Error: 0.37679932260795934
Confusion Matrix:
 [[480 544]
 [346 992]]
ROC AUC: 0.6618670590433483
```

[15]:
```
# Complete hyperparameter Tuning to create final model
param_dist = {'n_estimators': randint(50,500),
              'max_depth': randint(1,20)}

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)
```

[15]:
```
▸          RandomizedSearchCV
▸ estimator: RandomForestClassifier
      ▸ RandomForestClassifier
```

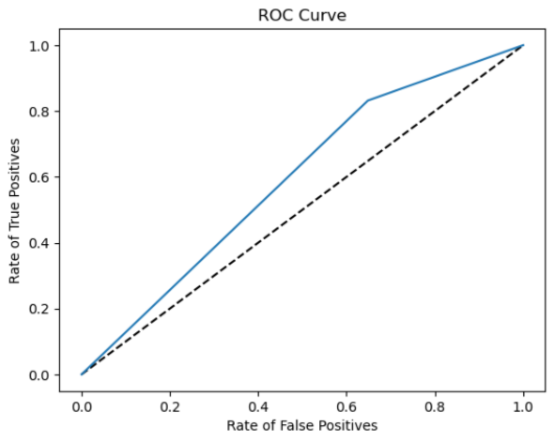
[16]:
```
# Create a variable for the best model
Final_best = rand_search.best_estimator_

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

```
Best hyperparameters: {'max_depth': 11, 'n_estimators': 254}
```

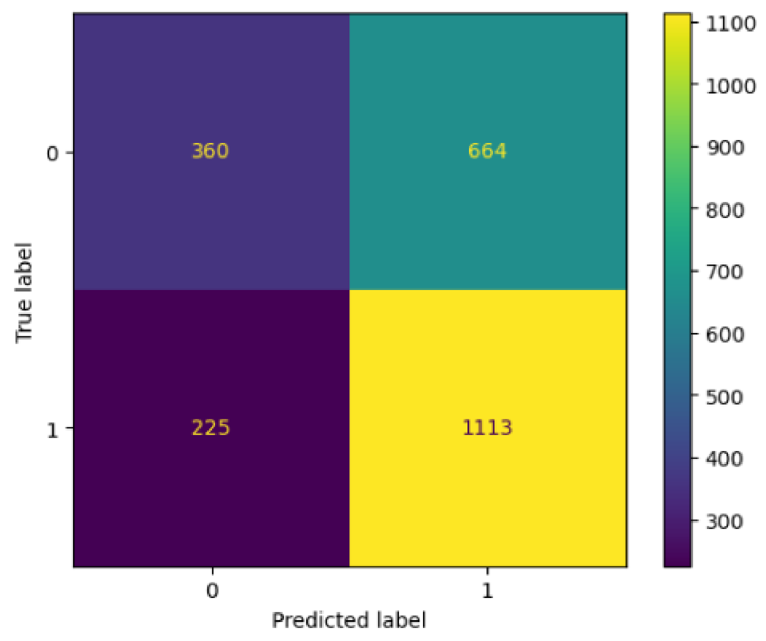
```
[18]: # Evaluate the final model using accuracy, f1, MSE confusion matrix and roc/auc
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
cmat = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, Final_best.predict_proba(X_test)[:, 1])
mse = mean_squared_error(y_test, y_pred)
# Print scores
print("Accuracy:", accuracy)
print("F1 Score:", f1)
print(f"Mean Squared Error: {mse}");
print("Confusion Matrix:\n", cmat)
print("ROC AUC:", roc_auc)
```

```
Accuracy: 0.6236240474174428
F1 Score: 0.7146067415730336
Mean Squared Error: 0.3763759525825571
Confusion Matrix:
 [[ 360  664]
 [ 225 1113]]
ROC AUC: 0.6522656541946936
```

```
[19]: # Plot ROC Curve for reduced model
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr)
plt.xlabel('Rate of False Positives')
plt.ylabel('Rate of True Positives')
plt.title('ROC Curve')
plt.show()
```



```
[20]: # Create the confusion matrix
      cm = confusion_matrix(y_test, y_pred)
      # Display confusion matrix
      ConfusionMatrixDisplay(confusion_matrix=cm).plot();
```



E2. Results and implications of prediction analysis

A mean squared error value of 0.376 for both models is concerning, because they share the same value. In addition, the value itself is not as low as one would expect with a model created to make business decisions. Because the lower the value, the better. However, this metric is more appropriate in a regression model rather than a binary classification model and so it is difficult to report its importance in this model. Regardless, the value is included because the assessment requires it. However, to truly analyze the model's outputs, I will be focusing more on the other evaluation metrics.

Given all the results mentioned above (sans MSE), the reduced random forest classification model is not completely reliable yet. Only 62.3% of all instances are predicted correctly according to the accuracy score. Having an F1 score of 0.714 for the final model compared to F1 of 0.690 for the initial model is a positive sign. The higher the F1 score, the better the balance is between precision and recall. Any trend of greater reliability in the model is gone when reviewing the AUROC scores though. For the final model, the AUROC score is 0.652 compared to the initial model score of 0.661. The higher the score, the more reliable the model is at correctly predicting true positives and true negatives. Having the reduced model have a higher AUROC score is alarming.

When looking at the confusion matrix for both models, we can certainly see that the reduced model is not much better than the initial model. Surprisingly, the initial model is more accurate at predicting true positives, while the final model is better at predicting true negatives.

Given the accuracy, F1 and AUROC scores for both the initial and reduced models, this random forest classifier may not be ready for the business to implement. The final model predicts many more false positives than true positives and is not more reliable than the initial model, or random chance.

E3. One limitation of the data analysis

One limitation of this analysis is that there were 2,129 instances of null values within the internet service column. Considering that there are many customers only utilizing phone service, and the analysis is to pinpoint which specific internet service is utilized, it was fine to chalk these up to customers without any internet service. However, it would be of great benefit to have concrete verification that all 2,129 instances are those of customers with no internet service, rather than potentially incorrect null values. Having as close to accuracy as possible will avoid data being skewed or interpreted incorrectly.

E4. Recommended course of action

Unfortunately, my analysis was unable to meet the goals I had set for this model. Given its unreliability, I would not be able to make great predictions with this model. In addition, there is no other recommendation I could give than to reassess the question and data at hand. Although I feel confident that the scripts used could be utilized in other settings, with this dataset – they should not be considered for business purposes. Instead, I would recommend the business do a deep dive into customer information and verify the type of internet service each customer has. By dropping the 2,129 instances of nulls in the internet service column, I reduced an already limited dataset of only 10,000 customers. Having the accurate internet service data available, and potentially more customer information, would positively impact the results of this classification. After reviewing customer data for more information, a re-evaluation of this question would be more appropriate before implementing this model.

Part VI: Demonstration

F. Panopto Video

The Panopto video recorded for this assessment can be found in the corresponding folder for this course.

G. List the web sources used to acquire data or segments of third-party code

Churchill, Briana. (2023, October 2). *Performance Assessment: Classification Analysis (NVM2).* Assignment for MS Data Analytics Course D209. Western Governors University.

Churchill, Briana. (2023, September 3). *Performance Assessment: Predicting Modeling Task 2*. Assignment for MS Data Analytics Course D208. Western Governors University.

Churchill, Briana. (2023, July 14). *Performance Assessment: Exploratory Data Analysis (OEM2)*. Assignment for MS Data Analytics Course D207. Western Governors University.

Datacamp. (2023, February). Random Forest Classification with Scikit-Learn. Retrieved from

https://www.datacamp.com/tutorial/random-forests-classifier-python

Geeks for Geeks (2023). *Random Forest Regression in Python*. Retrieved from:

https://www.geeksforgeeks.org/random-forest-regression-in-python/

Scikit-learn. (n.d.). *RandomForestRegressor*. In Scikit-learn: sklearn.ensemble.RandomForestRegressor. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

H. Acknowledge sources

Beheshti, N. (2022). *Random Forest Regression.* Towards Data Science.

Retrieved from: https://towardsdatascience.com/random-forest-regression-5f605132d19d

Kumar, A. (2023, Mar 17). *Accuracy, Precision, Recall & F1-Score – Python Examples.* https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/

Narkhede, S. (2018, June 26). *Understanding AUC – ROC Curve.* Towards Data Science.

Retrieved from: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Straw, E. (2023). *D209 Data Mining 1 Task 2 Cohort* [Unpublished document]. Western Governors University.

Straw, E. (2023). *Dr. Straw's Tips for Success in D209* [Unpublished document]. Western Governors University.

M, S. (2023). *Introduction to Random Forest in R.* Simplilearn. Retrieved from: https://www.simplilearn.com/tutorials/data-science-tutorial/random-forest-in-r