Performance Assessment: PREDICTIVE ANALYSIS

Gabriela Howell

Master of Science Data Analytics, Western Governors University

D209 – Data Mining

Professor Elleh

June 24, 2024

Part I: Research Question

A. Describe the purpose of this data mining report by doing the following:

1. **Propose** one **question relevant to a real-world organizational situation that you will answer using** one **of the following prediction methods:**

I will explore the research question, "Can a decision tree model effectively identify patients with anxiety?" In D209 Task 1, which focused on K-Nearest Neighbors Classification (KNN), I pursued a related research question as this is an issue among Americans. According to a 2024 chart from the CDC, 17.4% of adults in America have anxiety, by developing models capable of accurately predicting or detecting anxiety in patients. This time, I will use decision trees as my methodology.

2. **Define** one **goal of the data analysis. Ensure that your goal is reasonable within the scope of the scenario and is represented in the available data.**

A goal for this data analysis is to improve the predictive accuracy of identifying patients with anxiety. I'll achieve this by creating decision tree models based on features in the dataset. These models can effectively classify patients into anxiety categories using available data. The dataset will most likely contain relevant characteristics for predicting anxiety status. This goal aligns with the available dataset, which presumably holds relevant attributes for predicting anxiety amongst patients.

Part II: Method Justification

B. Explain the reasons for your chosen prediction method from part A1 by doing the following:

1. **Explain how the prediction method you chose analyzes the selected data set. Include expected outcomes.**

I chose to use a Decision Tree as my prediction method. To be more specific I will be utilizing an AdaBoosted Decision Tree. According to an article from Database Camp, decision trees are versatile tools employed for both classification and regression tasks, depending on the nature of the target variable (Database Camp, n.d.). The expected outcomes are that the decision tree model can predict patient outcomes by analyzing anxiety-related features and other health indicators. This structure collectively classifies patients as either having anxiety or not. Predictions are made by traversing the tree based on the input feature values, aiming to provide accurate classifications.

An AdaBoosted Decision Tree boosts the predictive performance by merging numerous weak decision trees to form a strong classifier. It achieves this through iterative training and weight adjustments. During each iteration, it focuses more on instances previously misclassified. By adjusting the weights of incorrectly classified occurrences, the model learns from its mistakes and improves accuracy. As a result, the AdaBoosted Decision Tree effectively handles complex datasets and provides reliable predictions for patient anxiety outcomes.

2. **Summarize** one **assumption of the chosen prediction method.**

One assumption of decision tree models, including AdaBoosted Decision Trees, is that they assume independence between features (Rawale, n.d.). This assumption implies that the model treats each feature as contributing independently to predicting the target variable, without taking into consideration for interactions between features. However, if the dataset contains highly correlated features or interactions that influence the target variable (anxiety status), the model

may fail to capture these dependencies accurately. Decision trees recursively split the dataset based on individual feature values, potentially missing complicated interactions that affect predictions.

3. **List the packages or libraries you have chosen for Python or R, and justify how each item on the list supports the analysis.**

I chose Python to compose my Decision Tree modeling. Which consists of several packages.

- Pandas: Used to load and analyze my dataset
- NumPy: Used for numerical operations to fine-tune array calculations
- Seaborn: Used for visualizations such as boxplots for outliers
- Matplotlib: Used for visualization in data distribution and plotting ROC
- Scikit-learn: Used for machine learning, such as;
    - train_test_split: Used to split the data into training and testing sets.
    - Feature_selection: Used to select important features
    - GridSearchCV: Used for hyperparameter fine-tuning.
    - DecisionTreeClassifier: Used for building and training decision trees.
    - plot_tree: Used to visualize the decision tree
    - AdaBoostClassifier: Used to enhance the model performance to ensure complicated data is handled correctly.
    - accuracy_score, roc_auc_score, confusion_matrix, mean_squared_error, classification_report, roc_curve: Metrics used for overall model evaluation.

Each of these libraries plays a crucial role in making Decision Trees work in Python.

Part III: Data Preparation

C. Perform data preparation for the chosen data set by doing the following:

1. **Describe** one **data preprocessing goal relevant to the prediction method from part A1.**

One important data preprocessing goal relevant to the Decision Tree prediction method is the handling of categorical variables using one-hot encoding. Overall, Decision Trees require numerical input to create meaningful splits. This is why the dataset's categorical variables must be converted into a numerical format. One-hot encoding transforms these categorical variables into a series of binary columns, each representing a unique category. Using this ensures the Decision Tree can effectively use these variables without assuming any order among the categories. Properly encoding categorical variables is essential for accurate data and effective anxiety prediction.

2. **Identify the initial data set variables that you will use to perform the analysis for the prediction question from part A1, and group *each* variable as numeric or categorical.**

- Numeric Variables:
    - Discrete: Population, Children, Item3, Item4
    - Continuous: Age, Income, VitD_levels, Initial_days, TotalCharge, Additional_charge

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

The steps for preparing the data for analysis are as follows: first, assemble the relevant variables. Then, identify and handle any duplicated and missing values. Additionally, examine

and address any outliers. The cleaning process began with separating the target variable 'Anxiety' from the feature dataset 'X', which included dropping irrelevant columns such as identifiers and geographical details. Numerical and categorical data were then analyzed separately, with summary statistics calculated for numerical features and frequency counts obtained for categorical ones. Boolean values were converted to integers, and unique values were inspected across all columns to ensure data integrity. Nominal categorical variables underwent one-hot encoding to prepare them for model training, while discrete and continuous numerical variables were identified for further analysis. Now that the dataset was prepared for subsequent modeling tasks. Finally, verify that the data has been properly updated.

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

The cleaned dataset will be provided and named 'D209_part2_clean.csv'.

Part IV: Analysis

D. Perform the data analysis and report on the results by doing the following:

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

The saved files are named: 'X_train_task2.csv', 'X_test_task2.csv', 'Y_train_task2.csv', and 'Y_test_task2.csv'.

2. **Describe the analysis technique you used to appropriately analyze the data. Include screenshots of the intermediate calculations you performed.**

Several steps are involved in analyzing the data. First, I clean up the dataset to remove any errors. Then, I split the data into two parts: one for training the model and one for testing its accuracy.

```
]: # Split the dataset into training and testing sets
   X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42, stratify=Y)
```

After that, I used GridSearchCV to tune. Where it finds the optimal parameters for max_depth and min_samples_leaf. This tuning process involved a 5-fold cross-validation and is trained to maximize the ROC AUC score. Where the best parameters found were max_depth=8 and min_samples_leaf=0.08.

```
# Hyperparameter Tuning for Decision Tree
# Define the parameter grid for Decision Tree
params_dt = {
    'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'min_samples_leaf': [0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20, 0.22]
}

# Configure dt
initial_dt = DecisionTreeClassifier(random_state=42)

# Configure grid_dt
grid_dt = GridSearchCV(estimator=initial_dt,
                       param_grid=params_dt,
                       scoring='roc_auc',
                       cv=5,
                       n_jobs=-1)

# Fit grid search to training model
grid_dt.fit(X_train, Y_train)

# Extract the best estimator
best_model = grid_dt.best_estimator_

# Show the best estimator
print(best_model)
```

```
DecisionTreeClassifier(max_depth=8, min_samples_leaf=0.08, random_state=42)
```

Next, the better Decision Tree model was assessed on the test set. This is where predictions are made, and the accuracy and ROC AUC score are calculated.

```
# Predict values for test set
initial_Y_pred = best_model.predict(X_test)

# Generate accuracy report for this model
acc_test = accuracy_score(Y_test, initial_Y_pred)
print('Test set accuracy of best decision tree: {:.2f}'.format(acc_test))

# Predict the test set probabilities of the positive class
initial_Y_pred_proba = best_model.predict_proba(X_test)[:, 1]

# Compute test_roc_auc
initial_roc_auc = roc_auc_score(Y_test, initial_Y_pred_proba)

# Print test_roc_auc
print('Test set ROC AUC score: {:.3f}'.format(initial_roc_auc))
```

```
Test set accuracy of best decision tree: 0.68
Test set ROC AUC score: 0.527
```

Next, I used an AdaBoosted Decision Tree classifier and I did some hyperparameter tuning via GridSearchCV to find the optimal parameters. The overall optimal parameters found were n_estimators=180 and learning_rate=1.3.

```
# Hyperparameter Tuning for AdaBoost
# Define the parameter grid for AdaBoost
params_ada = {
    'n_estimators': [160, 180, 200],
    'learning_rate': [0.9, 1.0, 1.1, 1.2, 1.3]
}

# Configure tuned decision tree
final_dt = DecisionTreeClassifier(max_depth=8, min_samples_leaf=0.06, random_state=42)

# Configure initial adaboost
initial_ada = AdaBoostClassifier(estimator=final_dt, random_state=42)

# Configure grid_ab
grid_ada = GridSearchCV(estimator=initial_ada,
                        param_grid=params_ada,
                        scoring='roc_auc',
                        cv=5,
                        n_jobs=-1)

# Fit grid search to training model
grid_ada.fit(X_train, Y_train)
```

GridSearchCV
estimator: AdaBoostClassifier

```
AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=8,
                                                    min_samples_leaf=0.06,
                                                    random_state=42),
                   random_state=42)
```

estimator: DecisionTreeClassifier
DecisionTreeClassifier

```
DecisionTreeClassifier(max_depth=8, min_samples_leaf=0.06, random_state=42)
```

```
# Display the most effective parameters
print(grid_ada.best_params_)
```

```
{'learning_rate': 1.3, 'n_estimators': 180}
```

```
# FINAL TUNED MODEL
# Configure ada with the best parameters
final_ada = AdaBoostClassifier(estimator=final_dt,
                               n_estimators=grid_ada.best_params_['n_estimators'],
                               learning_rate=grid_ada.best_params_['learning_rate'],
                               random_state=42)

# Fit ada to the training set
final_ada.fit(X_train, Y_train)
```

AdaBoostClassifier
estimator: DecisionTreeClassifier
DecisionTreeClassifier

```
DecisionTreeClassifier(max_depth=8, min_samples_leaf=0.06, random_state=42)
```

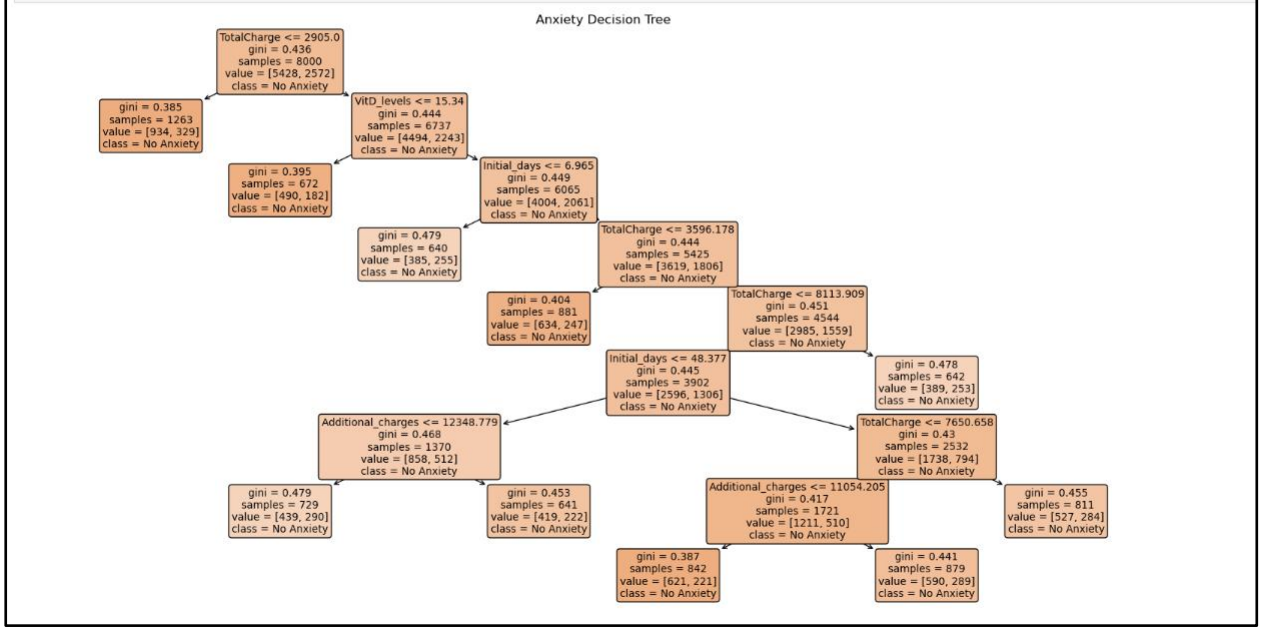
Then, I examined the Decision Tree to understand the Gini impurity, the number of samples, the value distribution, and the predicted class at each node. This visual inspection allowed me to see the level of impurity, the sample sizes, and the class distributions across different nodes.

```
# Visualize the Decision Tree
plt.figure(figsize=(20, 10))

# Convert X.columns to a list
feature_names = list(X.columns)

# Plot the decision tree
plot_tree(best_model, filled=True, feature_names=feature_names, class_names=['No Anxiety', 'Anxiety'], rounded=True)

plt.title('Anxiety Decision Tree')
plt.show()
```



Then to check how well the model works, I compare its predictions to the actual values using a confusion matrix. To better understand I included a heat map for the confusion matrix. I also use a ROC curve to see how good the model is at detecting true positives versus false positives.

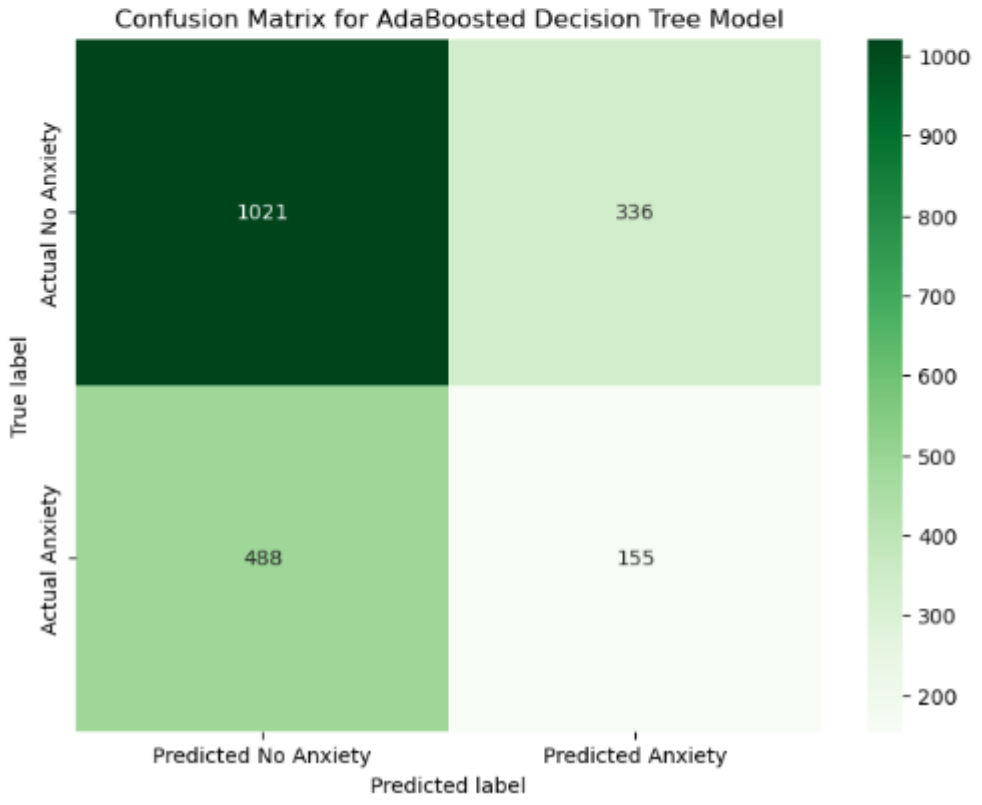
```
# Compute the probabilities of obtaining the positive class
final_Y_pred_proba = final_ada.predict_proba(X_test)[:, 1]

# Evaluate test-set roc_auc_score
final_roc_auc = roc_auc_score(Y_test, final_Y_pred_proba)

# Generate Confusion Matrix
final_matrix = confusion_matrix(Y_test, final_Y_pred)
print("\nThe confusion matrix for this AdaBoosted Decision Tree model:")
print("Predicted No Anxiety | Predicted Anxiety")
print(f"                   {final_matrix[0]} Actual No Anxiety")
print(f"                   {final_matrix[1]} Actual Anxiety\n")
```

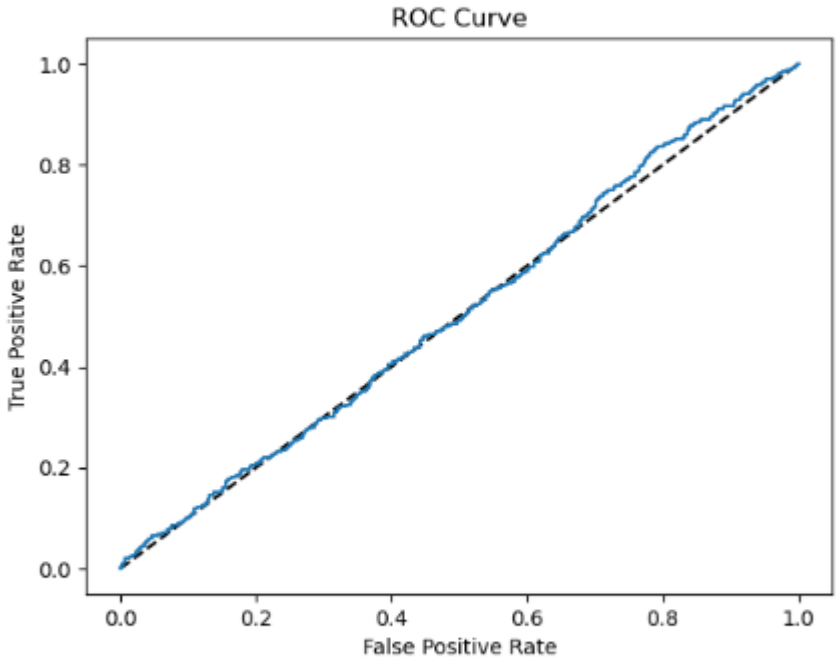
```
The confusion matrix for this AdaBoosted Decision Tree model:
Predicted No Anxiety | Predicted Anxiety
                   [1021  336] Actual No Anxiety
                   [488 155] Actual Anxiety
```

```
# Plot Confusion Matrix as a Heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(final_matrix, annot=True, cmap='Greens', fmt='g',
            xticklabels=['Predicted No Anxiety', 'Predicted Anxiety'],
            yticklabels=['Actual No Anxiety', 'Actual Anxiety'])
plt.title('Confusion Matrix for AdaBoosted Decision Tree Model')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()
```



Additionally, I calculate the Area Under the Curve (AUC) score to measure the model's overall performance. Lastly, a classification report gives a microscope look into how well the model predicts both classes of anxiety.

```
# Generate ROC plot and provide AUC score
fpr, tpr, thresholds = roc_curve(Y_test, final_Y_pred_proba)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
# Generate classification report
print(classification_report(Y_test, final_Y_pred))
```



	precision	recall	f1-score	support
0	0.68	0.75	0.71	1357
1	0.32	0.24	0.27	643
accuracy			0.59	2000
macro avg	0.50	0.50	0.49	2000
weighted avg	0.56	0.59	0.57	2000

3. Provide the code used to perform the prediction analysis from part D2.

The Python code will be provided in the attached file named Gab-D209_Part2.ipynb.

Part V: Data Summary and Implications

E. Summarize your data analysis by doing the following:

1. Explain the accuracy and the mean squared error (MSE) of your prediction model.

The final AdaBoosted Decision Tree model correctly predicted 59% of the outcomes on the test set. Whereas the Mean Squared Error (MSE) metric measures the average squared difference between the predicted values and the actual values. In this case, the MSE is 0.412. The lower the MSE the better fit model to the data. The Root Mean Squared Error (RMSE) is approximately 0.64. Making this average, the model's predicted probability of anxiety deviates from the actual value by 0648 on a scale from 0 to 1.

The AUC score of 0.51 indicates that the model has some ability to discriminate between the positive class (anxiety) and the negative class (no anxiety). An AUC score of 0.5 represents no discrimination (i.e., random guessing), while an AUC score of 1 represents perfect discrimination. Therefore, an AUC of 0.51 suggests that the model is relatively good at ranking predictions but still requires improvement. It’s similar to the MSE but conveyed in the same units as the target variable, which makes it easier to interpret the error magnitude.

2. Discuss the results and implications of your prediction analysis.

The AdaBoosted Decision Tree model achieved a moderate accuracy of 59% and an AUC score of 0.51. These results indicate that while the model is somewhat effective at distinguishing between the two classes of anxiety, there is room for improvement. The confusion matrix and classification report further highlight that the model performs better at predicting the "No Anxiety" class compared to the "Anxiety" class, with higher precision, recall, and F1-scores for the former.

Implications of this analysis include the potential for using this model in a real-world organizational context to identify individuals at risk of anxiety. However, the model's moderate performance suggests that it should be used as a supplementary tool rather than a primary diagnostic tool. Enhancing the model's ability to accurately predict the "Anxiety" class could significantly benefit targeted interventions and support programs.

3. **Discuss one limitation of your data analysis.**

One limitation of the data analysis is the imbalanced nature of the dataset, with a higher number of "No Anxiety" cases compared to "Anxiety" cases. This imbalance likely contributed to the model's lower performance in predicting the "Anxiety" class. Additionally, the model's performance metrics, such as accuracy and AUC, may not fully capture its effectiveness in real-world scenarios, where the cost of misclassifying an anxious individual could be high.

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

Since the model showed more data on no anxiety, I think it would be great to gather more information on anxiety. So that hospitals and healthcare providers can use the analysis insights to create focused interventions or support systems for patients with anxiety. These may involve specialized mental health services, educational resources, or preventive measures. By analyzing more data, organizations can adjust their strategies and allocate resources to recognize similarities among patients and learn more about anxiety.

Part VI: Demonstration

F. Provide a Panopto video recording that includes a demonstration of the functionality of the code used for the analysis and a summary of the programming environment.

My video link is here: Gabriela_Howell_D209_Task2_V2

G. Record the web sources used to acquire data or segments of third-party code to support the analysis. Ensure the web sources are reliable.

GeeksforGeeks. (2023, April 18). *Label encoding in Python*. https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/

1.10. decision trees. scikit. (n.d.-a). https://scikit-learn.org/stable/modules/tree.html

H. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

Centers for Disease Control and Prevention. (2024, May 16). *Mental health - household pulse survey - covid-19*. Centers for Disease Control and Prevention. https://www.cdc.gov/nchs/covid19/pulse/mental-health.htm

Rawale, S. (2018, May 30). *Understanding decision tree, algorithm, drawbacks and advantages.* Medium. https://medium.com/@sagar.rawale3/understanding-decision-tree-algorithm-drawbacks-and-advantages-4486efa6b8c3

What is a decision tree?. Data Basecamp. (2023, April 7). https://databasecamp.de/en/ml/decision-trees

1.10. decision trees. scikit. (n.d.-a). https://scikit-learn.org/stable/modules/tree.html