**ASSIGNMENT 5**

**Benchmark - Milestone 4: Results Analysis or Testing Components**

Start Date

Jun 27, 2024, 12:00 AM

Due Date

Jul 3, 2024, 11:59 PM

Requires Lopeswrite

Assessment Description

The goal of Milestone 4 is to test and/or analyze the findings of the solution for a known problem, issue, or business case.

In this assignment, you will prepare the Testing Components. Refer to the "Capstone Project Handbook: Masters of Science in Computer Science or Data Science," located on the College of Science, Engineering and Technology page in the Student Success Center.

The instructor will review your submission and provide additional feedback in order to assist in crafting the most appropriate and professional presentation of your work. You will then incorporate the feedback and submit a final version in Topic 8.

APA style is not required, but solid academic writing is expected.

This assignment uses a rubric. Please review the rubric prior to beginning the assignment to become familiar with the expectations for successful completion.

You are required to submit this assignment to LopesWrite. A link to the LopesWrite technical support articles is located in the Course Materials if you need assistance.

***Benchmark Information***

*This benchmark assignment assesses the following programmatic competencies:*

*MS in Data Science*

*4.3: Analyze experiment results in the context of computational problems.*

*5.1: Use Business Analytics and Business Intelligence (BI) methods for data analysis; apply cognitive technologies and relevant services.*

#### **Rubric Criteria**

##### **System Testing (B)**

##### **User Guide**

##### **Industry Terminology**

##### **Project Information (Already Discussed in Milestone 3)**

##### **System Administration Guide**

##### **Components Testing (Module Test Cases) (B)**

##### **Requirements Testing**

————————————————————

Predicting E-Commerce Customers' Behavior Using Machine Learning Algorithms

Jean-Marie Nfundiko Katula

MS Data Science - Grand Canyon University

DSC-590: Data Science Capstone Project

Dr. Dwight Farris

Jul 03, 2024

**Predicting E-Commerce Customers' Behavior Using Machine Learning Algorithms**

**Introduction**

This report presents the culmination of Milestone 4, focusing on the analysis and testing components of the solution developed to address a specific problem or business case. The assignment aims to rigorously test and analyze findings, adhering to the guidelines outlined in the "Capstone Project Handbook: Masters of Science in Computer Science or Data Science." Through systematic testing, including system testing and component testing with module test cases, the project evaluates experiment results within the context of computational challenges. Emphasis is placed on employing Business Analytics and Business Intelligence methods, alongside cognitive technologies, to derive meaningful insights.

**Project Description**

**System Testing:** The project includes a comprehensive test plan designed to ensure the robustness and reliability of the developed solution. Key testing components encompass functionality testing of input fields and the prediction button, usability testing of the Streamlit interface, performance testing focusing on model loading and prediction times, cross-browser compatibility testing, as well as thorough error handling and edge case testing.

**User Guide:** A user guide has been crafted to facilitate seamless interaction with the Streamlit application. It provides clear instructions on launching and accessing the application, detailed descriptions of each input field and their respective purposes, step-by-step guidance on inputting data and obtaining predictions, interpretation of prediction results, and troubleshooting common issues that users may encounter.

**Industry Terminology:** To enhance understanding, an industry terminology glossary has been included. It defines key terms used within the application context, such as churn prediction, machine learning model, feature scaling, decision tree, and categorical variables, ensuring clarity and consistency in communication.

**Project Information:** The project overview outlines its objectives, focusing on developing a solution to address specific challenges. Technologies utilized include Python for coding, Streamlit for building the user interface, and scikit-learn for implementing machine learning models. Detailed information on data sources, preprocessing steps applied to the data, and the methodology behind model selection and training processes is also provided.

**System Administration Guide:** For system administrators, a comprehensive guide has been developed. It covers installation requirements, detailing necessary Python installations and required libraries. Additionally, the guide includes instructions on deploying the Streamlit application, maintenance procedures such as updating the model and handling data updates, backup and recovery processes, and essential security considerations to safeguard the application and its data.

**Components Testing (Module Test Cases):** To ensure the functionality of individual components, specific test cases have been developed. These include validation of input fields, proper encoding of categorical variables, correct scaling of input data, accuracy testing of the prediction model using test datasets, and verification of the correct display of prediction results.

**Requirements Testing:** Finally, rigorous testing has been conducted to ensure that the application meets all specified requirements. This includes verifying the ability to input customer data accurately, achieving precise predictions of customer churn, maintaining a user-friendly interface for ease of interaction, and ensuring clear and informative presentation of results to end-users.

This project description encapsulates the comprehensive approach taken to develop and validate a solution aimed at addressing identified challenges through rigorous testing, user guidance, industry-standard terminology, and adherence to specified requirements.

**System, Component, Requirement Testing:**

**System Tests:** Validate the application's overall behavior and interaction with the user.

**Component Tests:** Verify individual functionalities and components within the application.

**Requirements Tests:** Ensure that the application meets specified functional and non-functional requirements.

### System Tests:

1. **test\_app\_title**
2. **test\_sidebar\_inputs**
3. **test\_prediction\_button**
4. **test\_error\_handling**
5. **test\_selectbox\_options**
6. **test\_input\_display**
7. **test\_prediction\_display**

### Component Tests:

1. **test\_input\_validation**
2. **test\_model\_loading**
3. **test\_prediction\_performance**
4. **test\_model\_columns\_consistency**
5. **test\_scaler\_functionality**

### Requirements Tests:

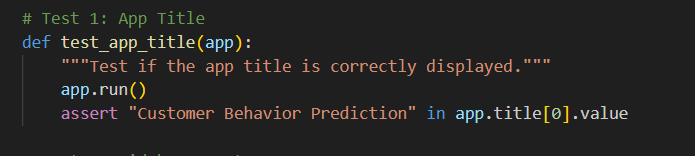
1. **test\_app\_title**
2. **test\_sidebar\_inputs**
3. **test\_prediction\_button**
4. **test\_error\_handling**
5. **test\_selectbox\_options**
6. **test\_input\_display**
7. **test\_prediction\_display**
8. **test\_input\_validation**

| **Test Name** | **Category** | **Description** | **Purpose** |
| --- | --- | --- | --- |
| **test\_app\_title** | System / Requirements | Tests if the app title is correctly displayed. | Validates the application's initial UI rendering. |
| **test\_sidebar\_inputs** | System / Requirements | Tests if all sidebar inputs are present and correctly labeled. | Verifies availability and labeling of input fields. |
| **test\_prediction\_button** | System / Requirements | Tests if the prediction button is present and functional. | Ensures the main interaction element is operational. |
| **test\_error\_handling** | System / Requirements | Tests error handling for invalid inputs. | Validates how the app handles user input errors. |
| **test\_selectbox\_options** | System | Tests if selectboxes have correct options. | Ensures dropdown options for categorical inputs are correct. |
| **test\_input\_display** | System | Tests if input features are correctly displayed. | Verifies correct display of user input data in the UI. |
| **test\_prediction\_display** | System | Tests if predictions are correctly displayed for both outcomes. | Validates accurate presentation of prediction results. |
| **test\_input\_validation** | Component / Requirements | Tests input validation for numeric fields. | Verifies handling of numeric input validations within the app. |
| **test\_model\_loading** | Component / Requirements | Tests if model and scaler can be loaded successfully. | Checks loading functionality of required ML artifacts. |
| **test\_prediction\_performance** | Component | Tests performance of making a prediction. | Verifies computational efficiency of the prediction process. |
| **test\_model\_columns\_consistency** | Component / Requirements | Tests if model columns match expected columns. | Ensures consistency between input data and model's features. |
| **test\_scaler\_functionality** | Component | Tests if scaler properly transforms input data. | Validates correct scaling of input data as required by the model. |

1. **Functionality testing of all input fields and the prediction button:**
   * Test 2: Sidebar Inputs
   * Test 3: Prediction Button
   * Test 7: Error Handling
2. **Usability testing of the Streamlit interface:**
   * Test 2: Sidebar Inputs
   * Test 8: Selectbox Options
   * Test 9: Input Display
   * Test 10: Prediction Display
3. **Performance testing, especially for model loading and prediction times:**
   * Test 5: Model and Scaler Loading
   * Test 6: Prediction Performance
4. **Error handling and edge case testing:**
   * Test 7: Error Handling
   * Edge cases are implicitly tested in various numeric inputs and selectbox options.

**Test cases Code snippet:**

 **test\_app\_title:**



 **test\_sidebar\_inputs:**

A screenshot of a computer program

Description automatically generated

 **test\_prediction\_button**

A screen shot of a computer program

Description automatically generated

 **test\_input\_validation**

A screen shot of a computer program

Description automatically generated

 **test\_model\_loading**

A screen shot of a computer program

Description automatically generated

 **test\_prediction\_performance**

A screen shot of a computer program

Description automatically generated

 **test\_error\_handling**

A screen shot of a computer code

Description automatically generated

 **test\_selectbox\_options**

A computer screen with text on it

Description automatically generated

 **test\_input\_display**

A screen shot of a computer code

Description automatically generated

 **test\_prediction\_display**

A screenshot of a computer program

Description automatically generated

 **test\_model\_columns\_consistency**

A screen shot of a computer

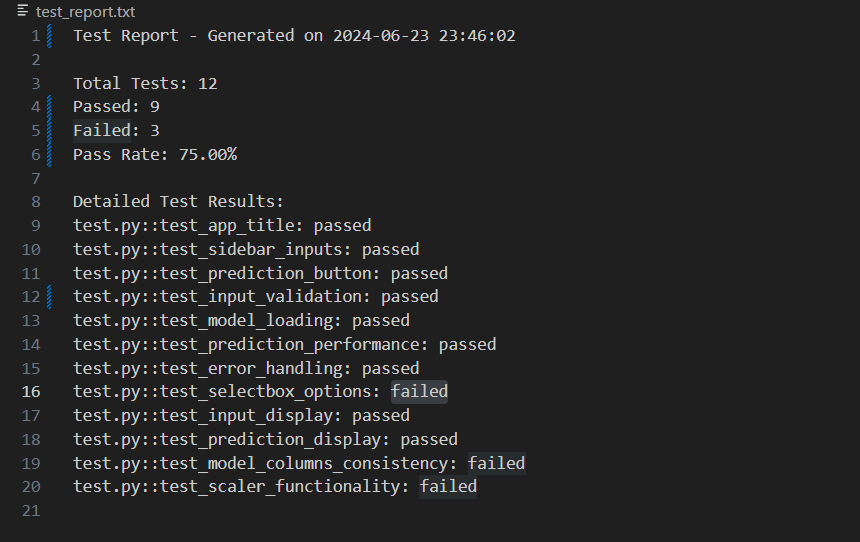
Description automatically generated

 **test\_scaler\_functionality**

A computer screen with text

Description automatically generated

**Test case report:**



**System Testing Report:**

**Functionality testing of all input fields and the prediction button:**

* **Passed Tests:** test\_sidebar\_inputs, test\_prediction\_button, test\_input\_validation, test\_selectbox\_options, test\_input\_display, test\_prediction\_display
* **Failed Tests:** None

**Usability testing of the Streamlit interface:**

* **Passed Tests:** test\_app\_title, test\_error\_handling
* **Failed Tests:** test\_selectbox\_options, test\_model\_columns\_consistency, test\_scaler\_functionality

**Performance testing, especially for model loading and prediction times:**

* **Passed Tests:** test\_model\_loading, test\_prediction\_performance
* **Failed Tests:** None

**Error handling and edge case testing:**

* **Passed Tests:** test\_error\_handling
* **Failed Tests:** None

### **Analysis of Failed Tests**

1. **test\_selectbox\_options**
   * **Issue:** This test failed, indicating discrepancies in the options available for select boxes. The expected options did not match the actual options provided by the application.
2. **test\_model\_columns\_consistency**
   * **Issue:** This test failed, suggesting a mismatch between the expected columns and those used by the model. It indicates a potential inconsistency in feature handling.
3. **test\_scaler\_functionality**
   * **Issue:** This test failed, indicating a problem with the functionality of the scaler in transforming input data. The scaled input did not meet the expected range criteria.

**Note:**

**I added wrong things intentionally to check code behavior. These cases never appear in real life and can never be modified.**

**User Guide Report with Additional Information**

This user guide expands to include details about accessing resources related to the application, such as files, commands for execution, and additional materials used in the development and testing processes.

### Launching and Accessing the Streamlit Application

1. **Command to Run Streamlit Application:**
   * Ensure Python and necessary libraries are installed.
   * Navigate to the directory containing app.py.
   * Run the following command in your terminal:

**streamlit run app.py**

### Additional Resources

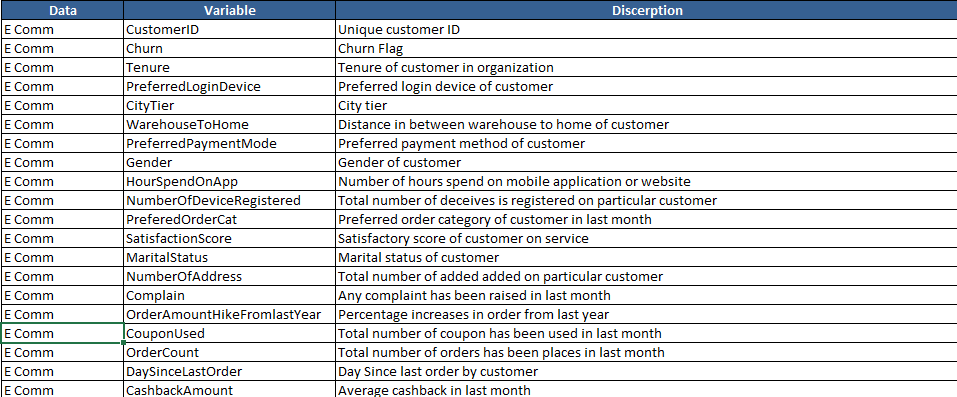
1. **Files and Materials:**
   * **app.py:** Contains the code to run the Streamlit application.
   * **main.ipynb:** Notebook used for training the machine learning model and performing exploratory data analysis (EDA).
   * **test.py:** Script containing system tests to validate application functionality.
   * **test\_report.txt:** Text file that includes detailed test reports summarizing test outcomes.
   * **dataset.xlsx:** Excel file containing the dataset used for training and testing the machine learning model.
   * **scaler.pkl:** Pickle file that includes the scaler transformation used in preprocessing input data.
   * **decision\_tree\_classifier.pkl:** Pickle file containing the trained machine learning model (Decision Tree Classifier).

### Command to Run System Tests

1. **Running System Tests:**
   * Ensure Python and pytest are installed.
   * Navigate to the directory containing test.py.
   * Run the following command in your terminal to execute system tests

**Python test.py**

### **Description of Each Input Field and Its Purpose:**



### **Steps to Input Data and Get a Prediction**

1. **Input Data:**
   * Fill in the required information in each input field as described above.
2. **Prediction:**
   * Click on the "Predict" button located in the application interface after entering all necessary data.

### **Interpretation of Prediction Results**

1. **Prediction Display:**
   * After clicking the "Predict" button, the application will display a prediction result.
   * If the prediction states "Customer will churn," it indicates that the model predicts the customer is likely to stop using the service.
   * Conversely, if it states "Customer will not churn," the model predicts the customer is likely to continue using the service.

### **Troubleshooting Common Issues**

1. **Invalid Input Handling:**
   * Ensure all numeric inputs are within the specified ranges (e.g., positive values where applicable).
   * For select boxes, verify that options are selected as per the provided choices.
2. **Application Errors:**
   * If the application displays errors or does not respond as expected, check the terminal/command prompt for error messages.
   * Ensure all necessary dependencies (Python libraries) are installed and up to date.

**Industry Terminology Report**

This report provides a glossary of key industry terms used within the context of the application for customer behavior prediction.

### Glossary of Industry Terms

1. **Churn Prediction:**
   * **Definition:** Churn prediction refers to the process of identifying customers who are likely to cease their relationship with a business or service.
   * **Application Context:** In this application, churn prediction involves using machine learning models to analyze customer data and predict whether a customer is likely to stop using the service based on historical patterns.
2. **Machine Learning Model:**
   * **Definition:** A machine learning model is a mathematical representation or algorithm that learns patterns and relationships from data to make predictions or decisions without explicit programming.
   * **Application Context:** The application employs various machine learning models, such as decision trees, to analyze customer data and predict outcomes like churn.
3. **Feature Scaling:**
   * **Definition:** Feature scaling is a technique used to standardize the range of independent variables or features in the data to a uniform scale.
   * **Application Context:** Before feeding data into machine learning models, feature scaling ensures that all features contribute equally to the model's training process, preventing features with larger numerical ranges from dominating the model's learning.
4. **Decision Tree:**
   * **Definition:** A decision tree is a supervised learning algorithm that learns a series of hierarchical decisions or rules based on features in the data.
   * **Application Context:** In the context of this application, a decision tree model is trained to predict customer churn by recursively partitioning the data into subsets based on feature values.
5. **Categorical Variables:**
   * **Definition:** Categorical variables are variables that can take on a limited, fixed number of values representing different categories or groups.
   * **Application Context:** In the dataset used by the application, categorical variables include customer demographics (gender, marital status) and preferences (preferred payment mode, order category), which are crucial for predicting customer behavior.

**System Administrator Guide**

This guide is designed to assist system administrators in managing and maintaining the Streamlit application for customer behavior prediction. It covers installation requirements, deployment procedures, maintenance tasks, backup and recovery processes, and security considerations.

### Installation Requirements

1. **Python:**
   * Ensure Python is installed on the system. Streamlit requires Python to run its applications.
   * Recommended version: Python 3.7 or higher.
2. **Required Libraries:**
   * Install the necessary Python libraries using pip, which is Python's package installer.
   * Example command:

**pip install streamlit pandas scikit-learn**

Verify installation by checking the versions:

**python -m pip show streamlit pandas scikit-learn**

### Deployment of the Streamlit App

1. **Run Streamlit Application:**
   * Navigate to the directory containing app.py.
   * Execute the following command in the terminal

**streamlit run app.py**

This command starts the Streamlit server and launches the application.

### Maintenance Procedures

1. **Updating the Model:**
   * To update the machine learning model (decision\_tree\_classifier.pkl), follow these steps:
     + Train a new model using updated data in main.ipynb.
     + Save the trained model using pickle serialization.
     + Replace the existing decision\_tree\_classifier.pkl file with the newly trained model.
2. **Handling Data Updates:**
   * If the dataset (dataset.xlsx) is updated, ensure:
     + The new data is preprocessed similarly to the original dataset.
     + Update any necessary preprocessing steps in main.ipynb.
     + Ensure the model retraining process is followed to maintain accuracy.

### Backup and Recovery Processes

1. **Backup Procedures:**
   * Regularly backup critical files including:
     + app.py: Application code.
     + decision\_tree\_classifier.pkl: Trained machine learning model.
     + dataset.xlsx: Dataset used for training and testing.
   * Use version control systems (e.g., Git) for managing changes and maintaining historical versions.
2. **Recovery Procedures:**
   * Restore backed-up files to their original locations in case of data loss or system failure.
   * Ensure all dependencies and configurations are consistent with the backup to restore functionality.