BHATTI, KOC, PANGANIBAN, TAN

Project Documentation: Predicting Online Shoppers' Purchasing Intentions

SS 2024

1. INTRODUCTION

- **1.1 Project Overview**
  - Overview of the project's aim and significance.
- **1.2 Dataset Description**
  - Source, nature, and characteristics of the dataset used.
- **1.3 Project Objectives**
  - Specific goals and expected outcomes of the project.

2. DATA PROCESSING AND PREPARATION

- **2.1 Key Steps in Data Preparation**
  - **2.1.1 Loading and Cleaning Data**
    - Techniques and methods used for initial data handling.
  - **2.1.2 Encoding Categorical Variables**
    - Approach for transforming categorical data into a machine-readable format.
  - **2.1.3 Feature Scaling**
    - Methods applied to normalize or standardize features.

3. MODEL DEVELOPMENT AND EVALUATION

- **3.1 Overview of Modeling Approach**
  - Summary of the predictive models developed.
- **3.2 Random Forest Classifier**
  - **3.2.1 Model Training (RFC_Train.py)**
    - Details of the training process and configurations.
  - **3.2.2 Model Testing (RFC_Test.py)**
    - Evaluation metrics and results from testing.
- **3.3 Logistic Regression**
  - **3.3.1 Basic Model Training and Testing (LR_Train.py, LR_Test.py)**
  - **3.3.2 Advanced Model with SMOTE (LR2_Train_Test.py)**
    - Modifications and enhancements to improve model performance.

4. VISUALIZATIONS AND ANALYTICAL TECHNIQUES

- **4.1 Visualization Techniques**
  - **4.1.1 Use of Visualization Scripts (Visualize.py and Ace_Visualization.py)**
- **4.2 Key Visualizations**
  - **4.2.1 Correlation Heatmaps**
    - Insights from feature interdependencies.
  - **4.2.2 Pie Charts for Revenue Distribution**
    - Visual representation of target variable classes.

5. PROJECT ARCHITECTURE

- **5.1 Feature Engineering and Preprocessing Pipeline (Build_Features.py)**
- **5.2 Model Training with Pipelines (RFC_Train.py)**

1. INTRODUCTION

PROJECT OVERVIEW

This project uses machine learning to predict whether visitors to an e-commerce site will make a purchase. The goal is to understand the factors that influence online shopping behaviors to enhance marketing strategies and increase conversion rates.

The dataset comprises 12,330 instances with 18 features, providing a comprehensive view of the factors that influence purchase decisions.

DATASET

- **Source:** UCI Machine Learning Repository
- **Features:** 18 (including Administrative, ProductRelated, BounceRates, ExitRates, PageValues, and Month)
- **Target Variable:** Revenue (indicates whether a purchase was made)

PROJECT OBJECTIVES

- **Hypothesis:** Website interaction metrics and user behaviors are significant predictors of a visitor's purchase intention.
- **Goal:** To create a predictive model that accurately identifies potential purchases, allowing for targeted strategies to enhance sales effectiveness.

2. DATA PROCESSING AND PREPARATION

Data preparation is crucial for the success of any machine learning project. Here, we focus on cleaning the data, handling missing values, encoding categorical variables, and scaling numerical features to prepare the dataset for modeling.

KEY STEPS IN DATA PREPARATION (PREPROCESS.PY):

- LOADING AND CLEANING DATA:

```
# Load the dataset
df = pd.read_csv('../../data/raw/online_shoppers_intention.csv')
print("Initial shape of the dataset:", df.shape)

# Display missing values count
missing_values = df.isnull().sum()
print("Missing values in each column:\n", missing_values)

# Remove duplicate rows and explicitly create a copy to avoid SettingWithCopyWarning
df_cleaned = df.drop_duplicates().copy()
print("Shape of the dataset after removing duplicates:", df_cleaned.shape)
```

This code loads the raw dataset and removes duplicate entries to ensure the uniqueness of the data. Plus we assess and address missing values in the dataset, ensuring that our models train on complete and accurate data.

- ENCODING CATEGORICAL VARIABLES:

```
# Encoding categorical variables
encoder = OneHotEncoder(sparse_output=False)
categorical_columns = ['VisitorType', 'Month']  # Encode both 'VisitorType' and 'Month'
df_encoded = pd.DataFrame(encoder.fit_transform(df_cleaned[categorical_columns]), columns=encoder.get_feature_names_out())
df_encoded.index = df_cleaned.index  # Ensure indices are aligned by setting df_encoded index to match df_cleaned
```

Categorical variables such as 'VisitorType' and 'Month' are encoded using one-hot encoding to convert them into a format that can be used by machine learning models.

- FEATURE SCALING:

```
# Feature scaling
scaler = StandardScaler()
scaled_columns = ['Administrative_Duration', 'Informational_Duration', 'ProductRelated_Duration']  # Example of columns to scale
df_cleaned[scaled_columns] = scaler.fit_transform(df_cleaned[scaled_columns])
print("Sample data after scaling:\n", df_cleaned[scaled_columns].head())
```

Numerical features are scaled to have a mean of zero and a standard deviation of one, which helps in optimizing the model's performance.

3. MODEL DEVELOPMENT AND EVALUATION

We developed and evaluated several models, focusing on their ability to predict purchasing intentions accurately.

RANDOM FOREST AND LOGISTIC REGRESSION MODELS

- RANDOM FOREST CLASSIFIER (RFC_TRAIN.PY AND RFC_TEST.PY)

```
# Create a pipeline that first preprocesses the data and then fits the model
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))])
```

The Random Forest algorithm is used for its ability to handle large datasets with higher dimensionality. It operates by building multiple decision trees and voting on the most popular output class.

- LOGISTIC REGRESSION (LR_TRAIN.PY, LR_TESET.PY, LR2_TRAIN_TEST.PY)

```
# Creating the final pipeline
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('classifier', LogisticRegression(max_iter=10000))])
```

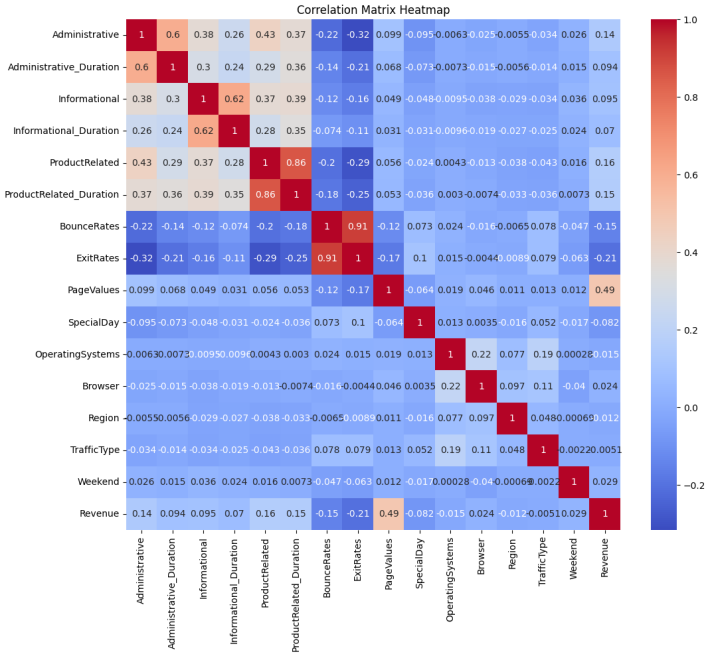
Logistic Regression is used for its efficiency and interpretability, particularly in binary classification tasks. The model is integrated into a pipeline that includes preprocessing steps.

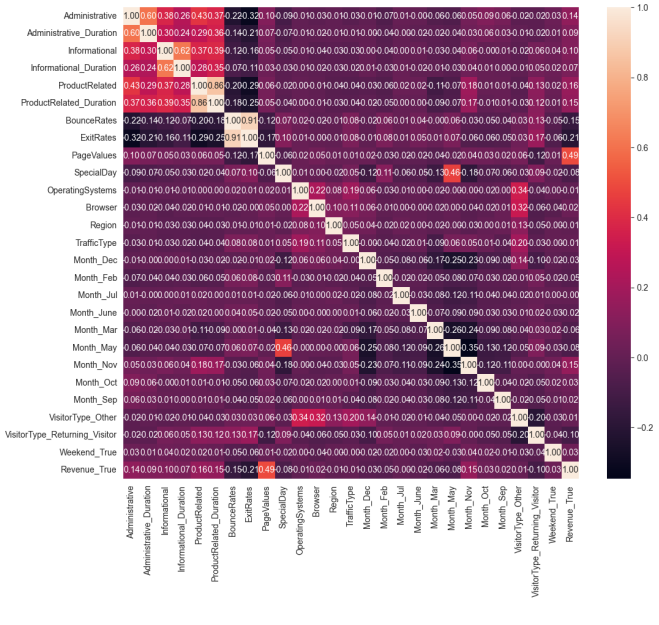
4. VISUALIZATIONS AND ANALYTICAL TECHNIQUES

We employ several visualization techniques to analyze the data and interpret the results, enhancing the understanding of model performance and data characteristics.

VISUALIZATION TECHNIQUES (VISUALIZE.PY AND ACE_VISUALIZATION.PY)

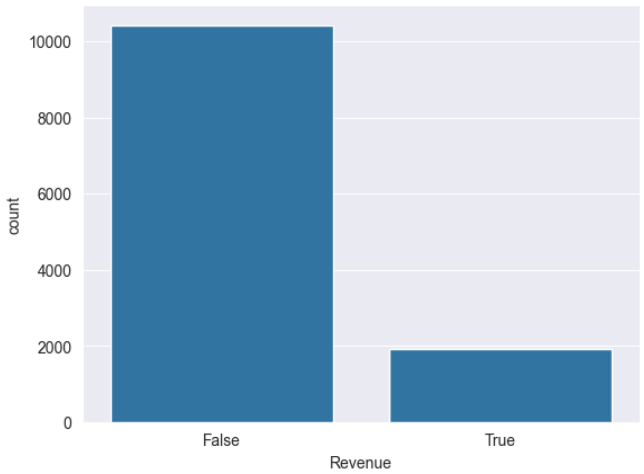
- CORRELATION HEATMAPS

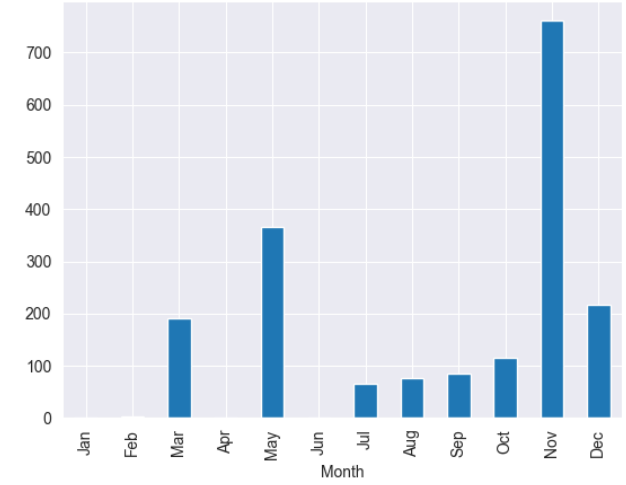




Heatmaps provide insights into the correlation between different features, helping identify which features are most related to the target variable.

- PIE CHARTS FOR REVENUE DISTRIBUTION





Pie charts help visualize the distribution of classes within the target variable, crucial for understanding class imbalance.

5. PROJECT ARCHITECTURE

The architecture of this project is carefully designed to foster modularity, scalability, and reproducibility, which are crucial for maintaining and extending the machine learning workflow. The structure consists of dedicated directories for each aspect of the project, ranging from raw data to processed datasets, and from initial exploratory notebooks to final scripts for training and evaluating models.

- **Data directory:** Organized into raw and processed subdirectories. Raw data remains untouched, serving as a baseline for all operations. Processed data includes cleaned, scaled, and engineered datasets ready for modeling.
- **Scripts and Notebooks:** Split into directories for source code (src), notebooks (notebooks), and documentation (docs). This separation helps distinguish between different stages of the project, such as exploratory data analysis, preprocessing routines, feature engineering, and modeling.
- **Model directory:** Contains serialized versions of trained models, allowing for easy deployment and testing without retraining. This is critical for real-world applications where model inference needs to be efficient and on-demand.

This structured approach not only simplifies the management of data and code but also ensures that each component of the project can be independently developed, tested, and improved.

6. CODE IMPLEMENTATION HIGHLIGHTS

Throughout this project, we utilized Python's powerful libraries and best practices to ensure clean, efficient, and robust code implementation. Below are detailed explanations of key code snippets that illustrate the project's complexity and functionality:

- FEATURE ENGINEERING AND PREPROCESSING PIPELINE (BUILD_FEATURES.PY):

```
# Polynomial Features
poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
poly_features = poly.fit_transform(df_cleaned[['Administrative', 'Informational', 'ProductRelated']])
poly_features_df = pd.DataFrame(poly_features, columns=poly.get_feature_names_out(['Administrative', 'Informational', 'ProductRelated']))
```

This snippet demonstrates the creation of polynomial and interaction features, which are crucial for uncovering non-linear relationships between features that might enhance model performance.

- MODEL TRAINING WITH PIPELINES (RFC_TRAIN.PY):

```
# Create a pipeline that first preprocesses the data and then fits the model
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))])

# Define the cross-validation strategy
kfold = KFold(n_splits=10, shuffle=True, random_state=42)
```

Here, a Scikit-learn pipeline encapsulates data preprocessing and model training in a single step, ensuring that all steps in the process are applied consistently. This prevents common mistakes such as data leakage and ensures that the model can be easily validated or tested with new data.

- MODEL EVALUATION AND VISUALIZATION (PERFORMANCE.PY):

```
# Plot performance metrics
metrics_df.set_index('Model', inplace=True)
ax = metrics_df.plot(kind='bar', figsize=(14, 8))
plt.title('Model Performance on Cleaned Data')
```

This code visualizes the performance of different models on cleaned data, providing immediate visual feedback on metrics such as accuracy, precision, recall, and F1-score. Visual comparisons are essential for communicating results effectively to stakeholders.

7. CONCLUSION

The results from this project clearly demonstrate the potential of machine learning in enhancing the understanding of online shopping behaviors. The Random Forest model emerged as the most effective, providing high accuracy and robustness against overfitting. Moreover, logistic regression models, especially those enhanced with SMOTE and class weighting, showed improved recall, indicating their utility in scenarios where identifying as many positive instances as possible is crucial.

In conclusion, this project lays a strong foundation for future developments in predicting online shopper behavior, providing both actionable insights and a robust methodology for continued innovation in e-commerce analytics.