Summary:

On the internet, every action on the website is monitored, allowing us to gather extensive information about users. This data is then analyzed to make predictions. The goal of this project is to build a system that predicts the likelihood of a specific user making a purchase under certain conditions while browsing the website. To achieve this goal, four steps were taken: data analysis, data processing, model embedding, and evaluation.

The dataset used in this project consists of users on the Commerce-E website, containing 23 features, some of which are anonymous. In the exploration stage, the data was examined to understand its content and determine how to proceed with the subsequent steps. The second stage involved data processing, which included converting categorical values to numbers, removing features, and adjusting measurements. Subsequently, the dataset was split into two parts: train and test data.

In the next stage, models were selected and evaluated to choose the best one for prediction on the test data. The Random Forest classifier was determined to be the most suitable model. During the exploration phase, various aspects of the data were examined, such as its size, different types of features (object, int, float), and numerical and categorical values. Distributions of each feature were analyzed, and it was discovered that two variables almost follow a normal distribution. Based on the distribution of labels, it was observed that the data is imbalanced, with 85% of sessions not resulting in a purchase.

Additionally, visualizations were created to examine the relationships between certain features, such as admin_page_duration and num_of_admin_pages. Furthermore, a correlation matrix was generated to determine the influence of features on each other. Strong correlations (above 0.80) were observed, and as a result, the decision was made to remove two features, BounceRates and duration_total, due to their high correlation increasing redundancy.

Preprocessing

Missing Values/Zeroes/Infinities

After exploring the data, I noticed that there are many missing values in multiple columns. I decided to fill these missing values with random values from the existing data. Additionally, I chose to remove features with missing values exceeding 0.9 (feature D). Furthermore, I examined the percentage of zeroes in each feature and also decided to remove the feature "holiday_to_closeness" because 85% of its values are zeroes. As for infinite values, they are not present in the given data.

Data Normalization

During the data analysis, I found that the data contains features with different ranges. In order to determine which model to choose, it is easier for models to deal with normalized data. Initially, I did not normalize the data, which took a long time for GridSearchCV to find the best hyperparameters for the SVM model. Therefore, I eventually decided to normalize the data using MinMaxScaler. This is especially important since it is a classification problem where data normalization is necessary to achieve good performance.

Outliers

Removing outliers can help stabilize the training process in the future. To identify these outliers, I created box plots for each variable. After examining the distribution plots in the X-axis variable, I decided to handle the features with normal or approximately normal distributions (B & EsitRates) and remove outliers based on the IQR (interquartile range). I noticed that the feature B (with a normal distribution) contains only a small percentage of the outliers. Since removing outliers implies removing the rows in which they exist, I chose not to lose more data at this stage, as there might be important outliers present.

Feature Selection
To select the best features, I performed two models: RandomForestClassifier and calssif_info_mutual. Both models returned the same top-performing features, from which I decided which ones to choose. Consequently, I selected the top 19 features.

Feature Engineering
To obtain uniform values that can be consistently interpreted, I converted all object-type values to numbers. This was achieved either through custom functions I created or using the dummies_get function. Applying this function transformed all categorical values I passed into numerical values, generating many columns (new features) to fulfill the requirement.

Model Execution Summary:

The first two models we chose were:
1. Logistic Regression
   - Hyper-Parameters: C=10.0, fit_intercept=True, max_iter=100, penalty='l2', random_state=None, solver='lbfgs'
2. K Neighbors Classifier
   - Hyper-Parameters: algorithm='brute', metric='euclidean', n_neighbors=22, weights='distance'

The advanced models we selected were:
3. Random Forest Classifier
   - Hyper-Parameters: criterion='entropy', max_depth=8, max_features=7, n_estimators=100
4. SVM
   - Hyper-Parameters: C=10, gamma=0.01, kernel='rbf', probability=True

Model Evaluation:
Confusion Matrix:
I constructed a confusion matrix for the Random Forest Classifier model. According to the matrix, the model successfully predicts 97% of the sessions that ended with a purchase (positive true). It correctly predicts 70% of the sessions that did not end with a purchase (positive true).

K-Fold Cross Validation & Overfitting:
Since we obtained very similar AUC scores, we can conclude that our models will perform similarly on the test set and that there is no overfitting. Additionally, relying on the AUC scores, we can see that the final model, Random Forest, had the highest score. Therefore, I chose it for the final predictions.

Summary:
In this project, we used machine learning models to predict whether a session will end with a purchase or not. This was done alongside a thorough exploration and analysis of the data to select the appropriate models for achieving maximum AUC performance.

Initially, we received noisy and inconsistent data, which we consolidated. Then, through visualization of the features, I drew conclusions about their importance. Next, I evaluated four different models using K-Fold Cross Validation, and ultimately chose the best-performing model, the Random Forest Classifier, for making predictions on the test data. Finally, I created a pipeline function that takes the raw files (test and train) and generates a new CSV file containing the predictions for the test data. Within the function, preprocessing, training, and prediction are performed.