MSIS 672 Fall 2023 Project

UMass  | Boston

GROUP MEMBERS:

ARPITA SRIVAS (02075452), SAMARTH adhikary (01943125), Yash alpeshbhai patel (02071032)

2023

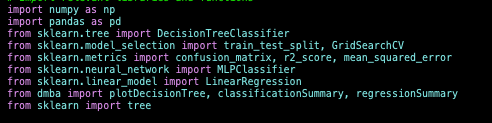
**Introduction**

In this project, we aimed to create models that can predict two key variables based on a sample of 2000 records from a larger dataset provided by CatalogCom: (1) whether a customer will make a purchase or not, and (2) how much they are likely to spend if they do make a purchase.

To achieve this, we developed two separate models: a decision tree model to predict the target variable "Purchase," and a linear regression model to predict "Spending" for purchasers. By using these models, we can gain insights into customer behavior and identify potential high-value customers for targeted marketing campaigns.

**Data exploration and data preparation**

Imported Libraries and Functions:



The dataset was imported into a variable named ‘df’ using the Pandas library. Our first step was to check the structure of the data with df.shape, which revealed that our DataFrame contained 2000 rows by 25 columns. As a next step, we checked whether the dataset contained null values. Using df.info(), we verified that the data did not contain null or empty values, and that the data are numerical (int64).

There was no need for concern about the categorical variables, since they had already been converted into dummy variables in the CSV file.

**Data analysis**

1. **First Part**

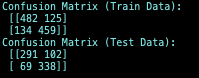
For the first part of the project, we used the Decision Tree Classification method to predict the class label of a new instance based on its features.

Next, we split the train and test data using the function train\_test\_split imported from the scikit-learn library. The train-test split is 60/40. While spitting data into the training and testing set, we assign feature variables to *X*, but we drop *Purchase* and *Spending* as purchase is the target variable and spending is not needed for our prediction. We assign the target variable *Purchase* to *y*.

For the decision tree model, we then set up a GridSearchCV to identify the optimal hyperparameters. Maximum depth, minimum sample splits, and minimum leaf samples are included in hyperparameters. By using cross-validation, GridSearchCV searches over the defined hyperparameters, returning the optimal hyperparameters for the model based on the given scoring metric. We then use DecisionTreeClassifier() and GridSearchCV() functions to train our model. After training the model, we use it to make predictions on the test set and calculate the confusion matrix using the actual and predicted values. Finally, we print the confusion matrix to evaluate the performance of the model.



According to the confusion matrix on the test data, the decision tree model correctly predicted 291 instances of class 0 (people who did not purchase) or **True Negatives (TN)** and 338 instances of class 1 (people who purchased) or **True Positives (TP)**. The model predicted 102 instances as class 1 when they were actually class 0, which is **False Positives (FP)**, and 69 instances as class 0 when they were actually class 1, which is **False Negatives (FN)**.



Based on the confusion matrices, the model is performing reasonably well on both the training and test sets, with similar accuracy scores. However, there seems to be a slight over-prediction of true positives and under-prediction of true negatives, indicated by the higher false positive rate in the test set. This may suggest overfitting, as the model is performing slightly worse on the test set compared to the training set.

To further assess overfitting, we compared the training and test set accuracy scores.

Accuracy score = (TP + TN) / Total

or

Train Data Accuracy Score= (482+459)/1200 = 0.7842,

and

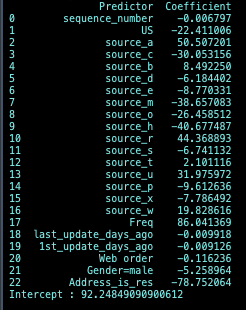
Test Data Accuracy Score= (291+338)/800 = 0.7875

The training set accuracy is 0.7842, while the test set accuracy is 0.7875, which is very similar. This suggests that overfitting is not a major concern in this model. However, further analysis and fine-tuning of the model may be necessary to improve its performance and reduce any potential overfitting.

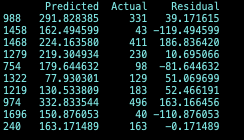
1. **Second Part**

The second part of the project involved performing regression. As we are predicting the spending of purchasers in this case, we must do the regression on the dataset by selecting only the rows where *Purchase = 1*. Therefore, we subset the dataset with the condition *Purchase = 1* *(df[df['Purchase']== 1)*. The 'Spending' column in the subset dataset is treated as the dependent variable, and the remaining columns are treated as independent variables. As we already subset the data with all Purchase values as 1, we eliminated the Purchase variable as well from the list of independent variables.

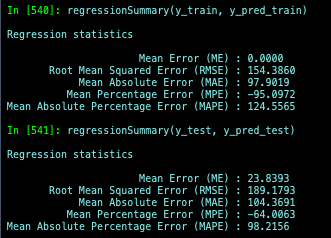
We do an 80/20 split of the data for training and testing. The linear regression was performed and fitted on the train data *(X\_train and y\_train)*. Next, we used *.predict* in our linear regression model to predict the *'y'* values for the test data. We printed the table containing our linear regression model columns. The DataFrame table shows the predictor variables and their coefficients.



To assess the performance of our model, we used *regressionSummary()* imported from *dmba* library *(Data Mining for Business Analytics)*. The function generated a summary of the regression results by comparing the actual coefficients of the dependent variable (*y\_test*) with the predicted coefficients (*y\_pred*).



The regression statistics for both the train and test sets show that the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) of the test set are higher than those of the train set. This suggests that the model may be overfitting the train data and may not be able to generalize well to new data.



Another indication of overfitting is that the Mean Percentage Error (MPE) for the test set is greater than that for the train set, indicating that the model underestimates the target values for the test set. It is worth noting that both the Mean Absolute Percentage Error (MAPE) and the Mean Absolute Percentage Error (MAPE) are high for both the train and test sets, suggesting that the model's predictions are not very accurate.

We utilized scikit-learn's mean\_squared\_error() and r2\_score() functions and observed that the MSE value of 35788.81 implies that the model's mean prediction error deviates approximately 35788.81 units from the actual values. A smaller MSE score suggests that the model's predictions are more precise and closer to the actual values, indicating improved performance in forecasting the target variable. Moreover, the R-squared value of 0.48 suggests that the regression model has some capability to anticipate the target variable, but there is still scope for enhancement.

**Findings and Conclusions**

In conclusion, the decision tree model's performance suggests slight overfit while predicting Purchase whereas the linear model is clear overfit to predict spending, and further improvements are necessary to enhance its reliability. The use of more data, the reduction of complexity, and the increase of regularization strength can help prevent overfitting. For Decision Tree Model, Hyperparameter tuning and cross-validation could also help in assessing overfit.

To improve the performance of the linear regression model, one suggestion is to try different regression algorithms such as Decision Tree Regression or Random Forest Regression to see if they can provide better results on the data.