DMML - Assignment 1

Ishita Pethkar (BMC202128), Siddhant Shah (BMC202171)

17^{th} March, 2024

Contents

1 Task 1

1.1 The Data

For this task, we refer to the data set *Customer Churn*, stored at `Data/Customer Churn.csv`.
The data set has the following dimensions: 1000 rows × 14 columns, stored as a `csv` with a size of 71 KB.
Overview of the Data Types:

- CustomerID - int64
- Age - int64
- Gender - object
- AnnualIncome - float64
- TotalSpend - float64
- YearsAsCustomer - int64
- NumOfPurchases - int64
- AvgTransactionAmount - float64
- NumOfReturns - int64
- NumOfSupportQueries - int64
- SatisfactionScore - int64
- LastPurchaseDaysAgo - int64
- EmailOptIn - bool
- PromotionResponse - object
- Churn - bool

We drop the `CustomerID` column as it is unique and doesn't offer any insights into the Churn. To be able to use `AdaBoost`, we need to convert the `object` type columns to numerical column. We do so using the `OrdinalEncoder()` from `scikit-learn`.

We then proceed to split the data into training and test sets (70-30 split), stratifying over `gender`.

1.2 `AdaBoost Classifier`

We start by finding the ideal hyper-paramters using `GridSearch`, with a *Repeated Stratified* 10-*Fold* and *accuracy* as the scoring metric.
We iterate over different values of `n_estimators`, `learning_rate` and `algorithm`. This takes a total time of approximately 150 seconds.

`GridSearch` gives us that the ideal values of the hyper-paramters are:

- `algorithm`: 'SAMME'
- `learning_rate`: 1e-05,
- `n_estimators`: 50

This configuration gives us an accuracy of 0.5343.

We train the final model with these hyper-parameter values, on the training data set. This takes 0.0557 seconds.

1.3 `Random Forest Classifier`

We start by finding the ideal hyper-paramters using `GridSearch`, with a *Repeated Stratified* 10-*Fold* and *accuracy* as the scoring metric.
We iterate over different values of `n_estimators`, `max_features`, `max_depth` and `max_leaf_nodes`. This takes a total time of approximately 450 seconds.

`GridSearch` gives us that the ideal values of the hyper-paramters are:

- `n_estimators`: 200
- `max_features`: 'log2'
- `max_depth`: 12
- `max_leaf_nodes`: 2

This configuration gives us an accuracy of 0.5252.

We train the final model with these hyper-parameter values, on the training data set. This takes 0.1235 seconds.

1.4 Model Comparison

We make predictions and then evaluate them using a confusion matrix for both models.

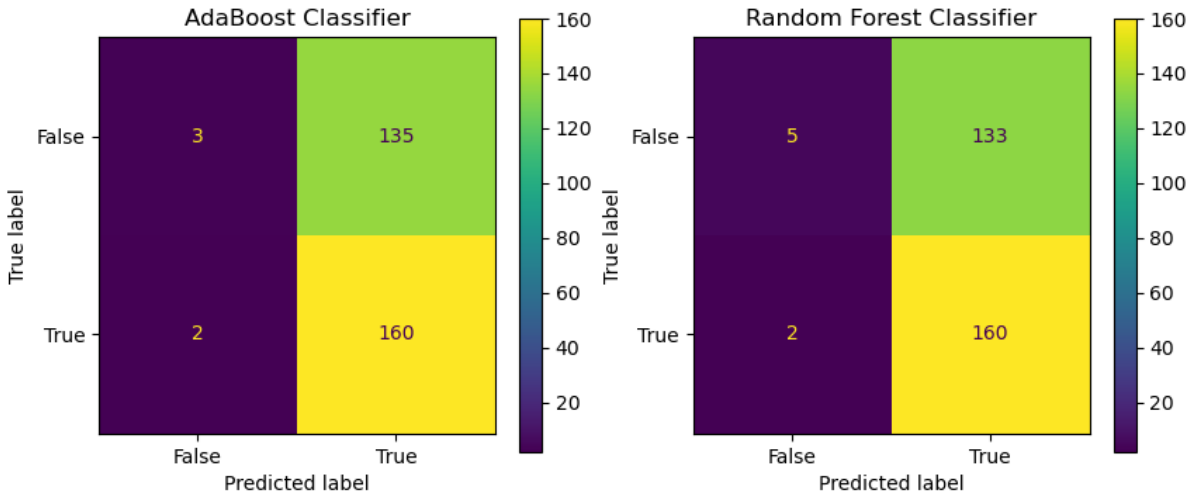

Figure 1: Confusion Matrix for Task 1 on Test Data

As the prediction is a binary value, we compare accuracy, recall & precision for these two models:

Metric	AdaBoost Classifier	Random Forest Classifier
Accuracy	0.5433	0.5500
Recall	0.9877	0.9877
Precision	0.5424	0.5461
F1 Score	0.7002	0.7033

2 Task 2

2.1 Gender Prediction

2.1.1 The Data

For this task, we refer to the data set *Supermarket Sales*, stored at `Data/Supermarket Sales.csv`.
The data set has the following dimensions: 1000 rows × 11 columns, stored as a `csv` with a size of 80 KB.
Overview of the Data Types:

- InvoiceID - object
- Branch - object
- CustomerType - object
- Gender - object
- ProductType - object
- UnitPrice - float64
- Quantity - int64
- Tax - float64
- Total - float64
- PaymentType - object
- Rating - float64

We drop the `InvoiceID` column as it is unique and doesn't offer any insights into the Churn. To be able to use these models, we need to convert the `object` type columns to numerical columns. We do so using the `OrdinalEncoder()` from `scikit-learn`.

We then proceed to split the data into training and test sets (70-30 split).

2.1.2 `Decision Tree Classifier`

We start by finding the ideal hyper-paramters using `GridSearch`, with a *Repeated Stratified* 10-*Fold* and *accuracy* as the scoring metric.
We iterate over different values of `max_features`, `max_depth`, `max_leaf_nodes` and `min_samples_leaf`. This takes a total time of approximately 1 second.

`GridSearch` gives us that the ideal values of the hyper-paramters are:

- `max_depth`: 9
- `max_features`: 'log2'
- `max_leaf_nodes`: 10
- `min_samples_leaf`: 5

This configuration gives us an accuracy of 0.5033.

We train the final model with these hyper-parameter values, on the training data set. This takes 0.0044 seconds.

2.1.3 `Random Forest Classifier`

We start by finding the ideal hyper-paramters using `GridSearch`, with a *Repeated Stratified* 10-*Fold* and *accuracy* as the scoring metric.
We iterate over different values of `n_estimators`, `max_features`, `max_depth` and `max_leaf_nodes`. This takes a total time of approximately 180 seconds.

`GridSearch` gives us that the ideal values of the hyper-paramters are:

- `n_estimators`: 100
- `max_features`: 'log2'
- `max_depth`: 6
- `max_leaf_nodes`: 10

This configuration gives us an accuracy of 0.5224.

We train the final model with these hyper-parameter values, on the training data set. This takes 0.0917 seconds.

2.1.4 Model Comparison

We make predictions and then evaluate them using a confusion matrix for both models.

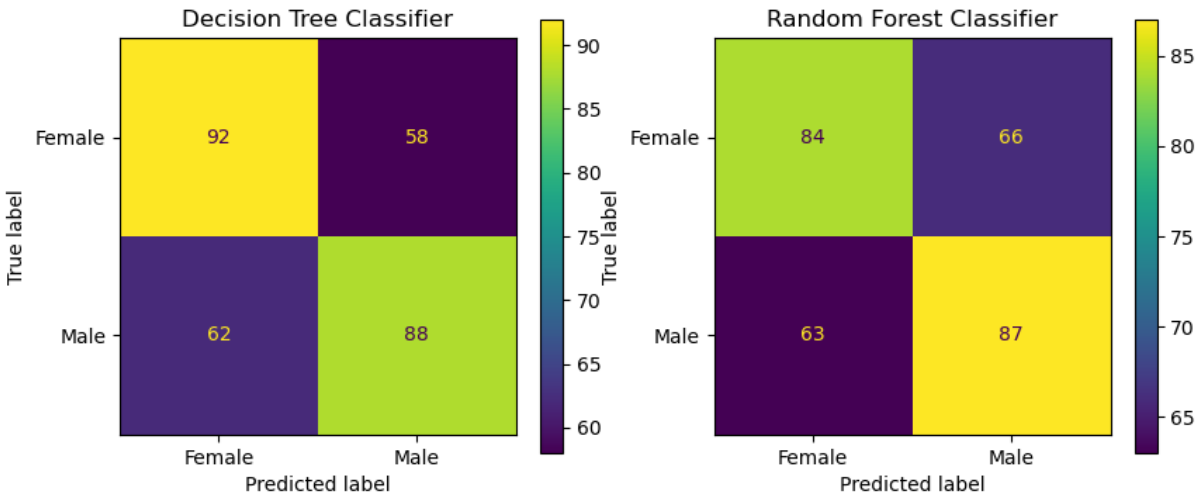

Figure 2: Confusion Matrix for Task 2A on Test Data

As the prediction is a binary value, we compare accuracy, recall & precision for these two models:

Metric	Decision Tree Classifier	Random Forest Classifier
Accuracy	0.6000	0.5700
Recall	0.5867	0.5800
Precision	0.6027	0.5686
F1 Score	0.5964	0.5743

2.2 Ratings Prediction

2.2.1 The Data

For this task, we refer to the data set *Supermarket Sales*, stored at `Data/Supermarket Sales.csv`.
The data set has the following dimensions: 1000 rows × 11 columns, stored as a `csv` with a size of 80 KB.
Overview of the Data Types:

- InvoiceID - object
- Branch - object
- CustomerType - object
- Gender - object
- ProductType - object
- UnitPrice - float64
- Quantity - int64
- Tax - float64
- Total - float64
- PaymentType - object
- Rating - float64

We drop the `InvoiceID` column as it is unique and doesn't offer any insights into the Churn. To be able to use these models, we need to convert the `object` type columns to numerical columns. We do so using the `OrdinalEncoder()` from `scikit-learn`.

We then proceed to split the data into training and test sets (70-30 split).

2.2.2 `Linear Regressor`

As Linear Regression has no hyper-parameters, we run it directly on the training set and get the values of the intercept and coefficients to be as follows:

- Intercept: 6.547981457474257
- Coefficients: `[ 5.66160866e-02 -1.27878875e-02 -1.16013065e-02  8.59907186e-03 1.00556066e-01 -9.26284356e-0`

This takes 0.0060 seconds.

2.2.3 `Decision Tree Regressor`

We start by finding the ideal hyper-paramters using `GridSearch`, with `neg_mean_squared_error` as the scoring metric. We iterate over different values of `splitter`, `max_depth`, `min_samples_leaf`, `min_weight_fraction_leaf`, `max_features`, and `max_leaf_nodes`. This takes a total time of approximately 75 seconds.

`GridSearch` gives us that the ideal values of the hyper-paramters are:

- `max_depth`: 3
- `max_features`: 'log2'
- `max_leaf_nodes`: 2
- `min_samples_leaf`: 2
- `min_weight_fraction_leaf`: 0.2
- `splitter`: 'random'

This configuration gives us an MSE of 2.9333.

We train the final model with these hyper-parameter values, on the training data set. This takes 0.0029 seconds.

2.2.4 Model Comparison

As the prediction is a continuous value, we compare the RMSE (Root Mean Squared Error):

Metric	Linear Regression	Decision Tree Regressor
RMSE	1.7418	1.7257