Assignment 1 Report

Data Mining and Machine Learning

Anjan Mondal
MDS202208

February 12, 2023

1 Introduction

In this assignment, we were tasked with fitting two machine learning classification models to two datasets, one of a bank and the other of a movie. The first model is a decision tree and the second one is a Naive Bayes model. The goal of these tasks is to compare the performance of the two models and determine which one is more suitable for each type of data.

2 Task 1: Bank Data Classification

In this task, we use the bank data to perform a classification task. We are given a set of attributes like Age, Education, Credit Card Defualt etc. and we need to predict whether a client subscribes to a term deposit. The data was preprocessed to prepare it for the modeling stage. This involved converting categorical variables into numerical variables, specifically One Hot Vectors using the `get_dummies` function in pandas. The final preprocessed data was split into a training set and a testing set in a ratio of 80:20. We use `train_test_split` function from the `sklearn` library and set the parameter `stratify=y` to gurantee equal proportion of all class labels in the test and the training sets. `random_state=40` is used for the split.

2.1 Modeling

In this section, we describe the steps taken to fit the decision tree and Naive Bayes models to the preprocessed bank data.

2.1.1 Decision Tree

The decision tree model was fit to the training data using `sklearn` library's `GridSearchCV` function with `DecisionTreeClassifier()` as its estimator. The parameters `max_depth`, `min_samples_split`, `min_samples_leaf` were fitted for a range of values to obtain a best combination of the parameters on the training

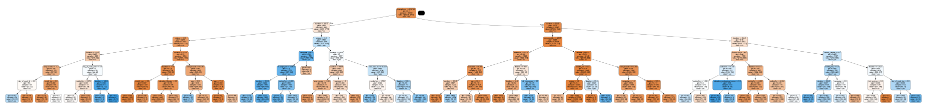


Figure 1: Bank Data Decision Tree

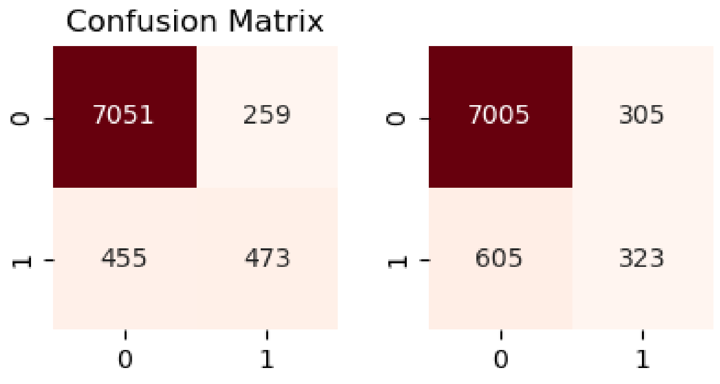
set. The defualt 5-fold cross validation was implemented for each combination of the parameters. The model was trained using the default Gini impurity as the splitting criterion. The best parameters turned out to be the following : `{'max_depth': 6, 'min_samples_leaf': 6, 'min_samples_split': 3}`. The time required to fit the model and find the best parameters is: `120.95 seconds` , the memory required for training the dataset is `24.27 MB`.

2.1.2 Naive Bayes

The Naive Bayes model was fit to the training data using the `sklearn` library's `MultninomialNB`. Before fitting the model, both the train and test data were scaled by `MinMaxScaler()`, which scales the data between 0 and 1 by default. This scaling is necessary becuase `MultinomialNB` doesn't fit for negative feature values. The time required to fit the model and find the best parameters is: `0.08 seconds` , the memory occupied by the training and test dataset is `19.84 MB`.

2.2 Evaluation

In this section, we compare the performance of the two models using various evaluation metrics such as accuracy, precision, recall, and F1 score.



(a) Decision Tree Classifier (b) Naive Bayes

Figure 2: Confusion Matrices for models trained on Bank Data

The accuracy of the models was calculated by comparing the predictions of the models with the actual values in the testing set. The accuracy of the decision tree model was found to be , while the accuracy of the Naive Bayes model was .

The precision, recall, and F1 score were also calculated for both models. The following results were obtained:

Table 1: Decision Tree Results

Metric	Yes	No
Accuracy	0.913	0.913
Precision	0.646	0.939
Recall	0.510	0.965
F1 Score	0.570	0.952

Table 2: Naive Bayes Results

Metric	Yes	No
Accuracy	0.890	0.890
Precision	0.514	0.920
Recall	0.348	0.958
F1 Score	0.415	0.939

3 Task 2: Movie Data Prediction

In this task, we use a bollywood movie dataset to perform a task of classification if a movie would be hit or flop based on features like Whether it is a remake, Genre, Lead Actor etc. For this dataset we had to create a separate column indicating whether the film was a hit or a flop. This target variable is set to `'hit'` if the revenue is greater than the budget and `'flop'` otherwise.
The data was preprocessed in a similar way as the bank data. The final preprocessed data was split into a training set and a testing set in a ratio of 80:20.

3.1 Modeling

In this section, we describe the steps taken to fit the decision tree and Naive Bayes models to the preprocessed bank data.

3.1.1 Decision Tree

The decision tree model was fit to in a manner similar to the Task 1. The best parameters turned out to be the following : `{'max_depth': 8, 'min_samples_leaf': 2, 'min_samples_split': 2}`. The time required to fit the model and find the best parameters is: `124.84 seconds`, the memory occupied by the training and test dataset is `19.37 MB`.

3.1.2 Naive Bayes

The Naive Bayes model was fit to in a manner similar to the Task 1. Here too `MultninomialNB` was used. Also `MinMaxScaler()` was used to scale the data. The time required to train the model is `0.03 seconds`, the memory occupied by the training and test dataset is `57.79 MB`.

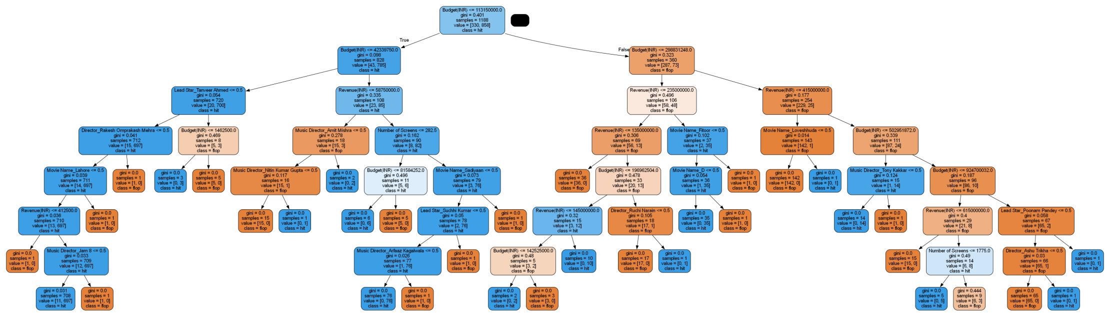
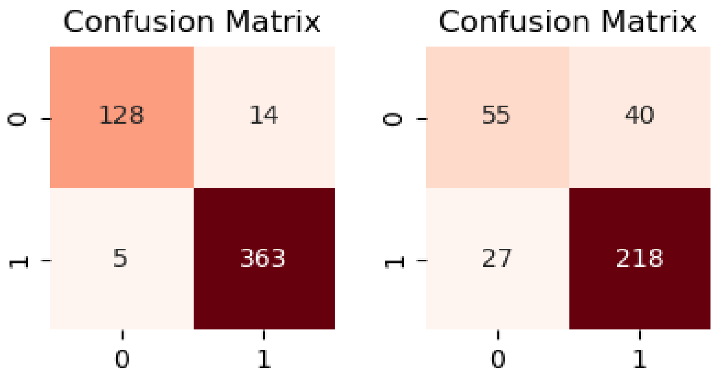


Figure 3: Movie Data Decision Tree



(a) Decision Tree Classifier (b) Naive Bayes

Figure 4: Confusion Matrices for models trained on Movie Data

3.2 Evaluation

In this section, we compare the performance of the two models using various evaluation metrics such as accuracy, precision, recall, and F1 score.
The accuracy of the models was calculated by comparing the predictions of the models with the actual values in the testing set.
The precision, recall, and F1 score were also calculated for both models.

Table 3: Decision Tree Results

Metric	Flop	Hit
Accuracy	0.963	0.963
Precision	0.962	0.963
Recall	0.901	0.986
F1 Score	0.931	0.974

Table 4: Naive Bayes Results

Metric	Flop	Hit
Accuracy	0.803	0.803
Precision	0.671	0.845
Recall	0.579	0.890
F1 Score	0.621	0.867

4 Conclusion

The results show that the decision tree model had a higher precision, recall, and F1 score compared to the Naive Bayes model for both the tasks. This is because the Naive Bayes model assumes independence between features, meaning that the presence or absence of a particular feature does not affect the presence or absence of another. This assumption is a strong one and can lead to poor accuracy, especially when the data contains complex relationships between the features.

Decision Trees are a more flexible and robust method for making classifications. They can easily capture complex relationships between variables by dividing the data into smaller subsets based on their characteristics.
If the training dataset is imbalanced, meaning that it contains a disproportionate number of samples from one class compared to others, the model may be biased towards the majority class and perform poorly on the minority class. Hence for the Bank data, where the number of `no`s far outweigh the number of `yes`ses, we see poor metrics for the `yes` class.