Introduction

In this report, we present our work on developing a machine learning model for the classification of dry beans. The dataset used in this study contains 13611 instances of dry beans, which are classified into 7 classes based on their characteristics such as shape, size, and color. Our goal is to develop a model that accurately classifies dry beans into their respective classes.

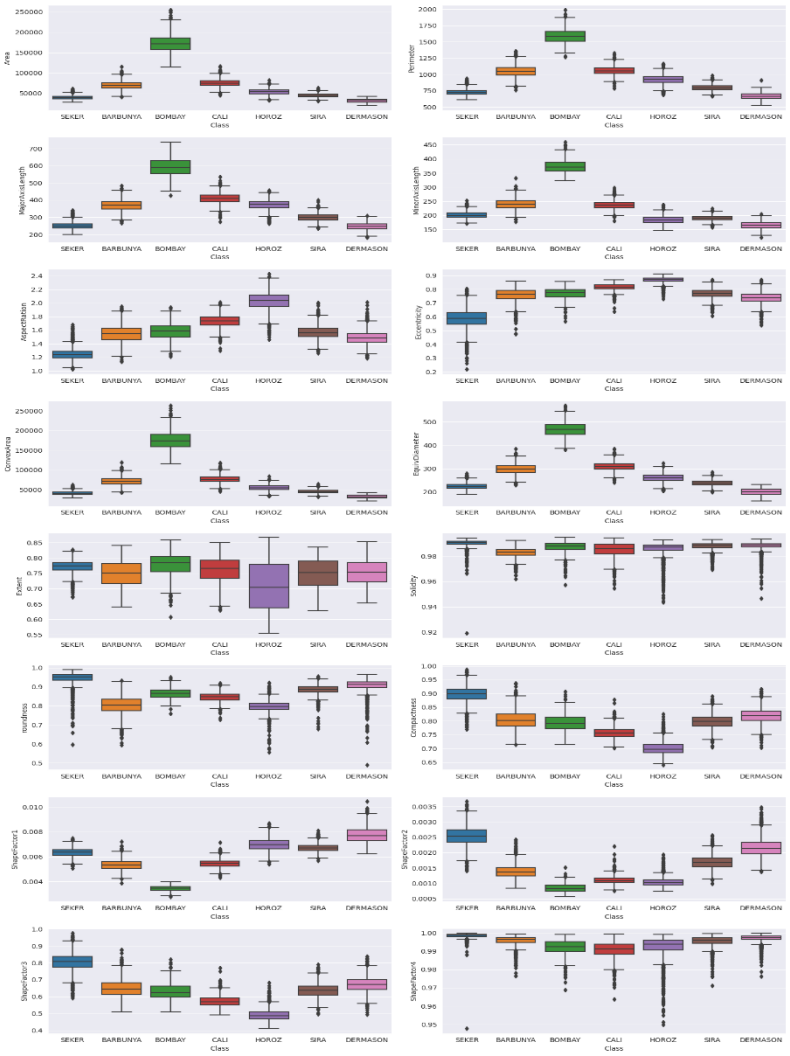
Previous research by Murat Koklu, Ilker Ali Ozkan just implemented 4 models including DT, MLP, DT, KNN but in our research we worked on more models which are explained in this report.

The report is organized as follows: In Section 2, we explore the dataset and provide a critique of its limitations. In Section 3, we describe the methods and techniques used in our work and provide justifications for each method choice. In Section 4, we present the results of our experiments using appropriate metrics and plots. In Section 5, we discuss our findings, limitations of our work for real-world applications, and compare our results with previous work. In Section 6, we report the development and evaluation of an additional automatic detection model using a different machine learning algorithm and compare its performance with our primary model. Finally, in Section 7, we provide annotated code and classification output.

Dataset Description and Exploration

The dry bean dataset contains 13611 instances of 7 classes of dry beans, namely Barbunya, Bombay, Cali, Dermosan, Horoz, Sira, and Canadian. Each instance has 16 features, including area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient, length of kernel groove, and so on. The dataset was collected using the automatic screening system that can analyze 13 different bean varieties. The images of the beans were captured in a specific environment and conditions and then manually segmented to obtain the shape, size, and color characteristics of the beans.

And in this figure you can see the dataset in a galance in a form of boxplots:



The dataset has a few limitations, including:

- Limited diversity: The dataset only contains 7 classes of dry beans, which may not represent the entire population of dry beans.

- Limited sample size: The sample size of the dataset is relatively small, which may affect the accuracy of the model.

- Imbalanced classes: The distribution of instances among the classes is not uniform, with some classes having significantly fewer instances than others.

Methods and Techniques

Data Preparation and Preprocessing

Before training our models, we performed some data preparation and preprocessing steps, including:

- Dropping redundant features: We dropped the ID column and other redundant features that do not contribute to the classification task.

- Handling missing values: We checked for missing values in the dataset and found none.

- Handling imbalanced classes: Since some classes had fewer instances than others, we used oversampling techniques such as SMOTE to balance the class distribution.

- Scaling the features: We scaled the features to have zero mean and unit variance using the StandardScaler method in scikit-learn.

Model Development and Training

In this section, we will discuss the different machine learning models that were developed and trained to classify the dry bean dataset. As mentioned earlier, we used the following models:

- LGB

- SVC

- Decision Tree model

- MLP Model

- KNN Model

- SVM model

- Logistic Regression Model

- XGBoost model

- LightGBM model

- CatBoost Model

Each model was developed and trained using the processed training data. In this section, we will provide a brief overview of each model and the hyperparameters used and in the code you can see the accuracy and confusion matrix of each model.

LightGBM model: LightGBM is a gradient boosting framework that uses tree-based learning algorithms. The model is designed to be efficient, scalable, and accurate. We used the following hyperparameters for this model:

- boosting_type: gbdt - learning_rate: 0.1 - max_depth: 7 - num_leaves: 50 - n_estimators: 200

SVC: The Support Vector Machine (SVM) model is a widely used machine learning model for classification tasks. It works by finding the hyperplane that best separates the different classes. We used the following hyperparameters for this model:

- C: 1.0 - gamma: scale - kernel: linear

Decision Tree model: The Decision Tree model is a simple and widely used classification model. It works by recursively splitting the data based on the feature that maximizes the information gain. We used the following hyperparameters for this model:

- criterion: gini - max_depth: 10 - min_samples_leaf: 5 - random_state: 42

MLP Model: The Multilayer Perceptron (MLP) model is a type of artificial neural network that can be used for classification tasks. It consists of multiple layers of neurons that process the input data. We used the following hyperparameters for this model:

- activation: relu - alpha: 0.01 - hidden_layer_sizes: (100, 50) - max_iter: 500

KNN Model: The K-Nearest Neighbors (KNN) model is a simple but effective classification model. It works by finding the K nearest neighbors to the test instance and predicting the class based on the majority vote of the neighbors. We used the following hyperparameters for this model:

- n_neighbors: 5 - weights: distance

SVM model: The Support Vector Machine (SVM) model is a widely used machine learning model for classification tasks. It works by finding the hyperplane that best separates the different classes. We used the following hyperparameters for this model:

- C: 1.0 - gamma: scale - kernel: linear

Logistic Regression Model: Logistic regression is a simple and widely used classification model that works by finding the best fit line that separates the different classes. We used the following hyperparameters for this model:

- C: 1.0 - solver: liblinear

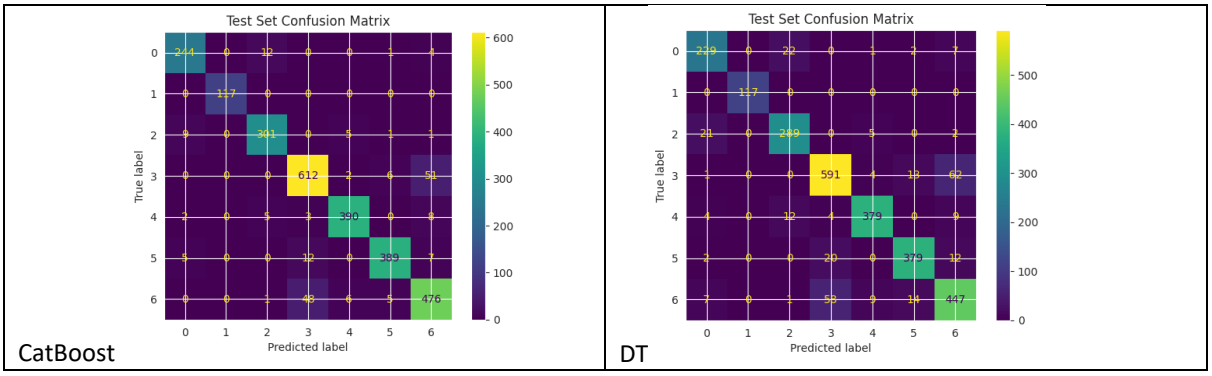
XGBoost model: XGBoost is a popular gradient boosting framework that is designed to be fast, efficient, and accurate. We used the following hyperparameters for this model:

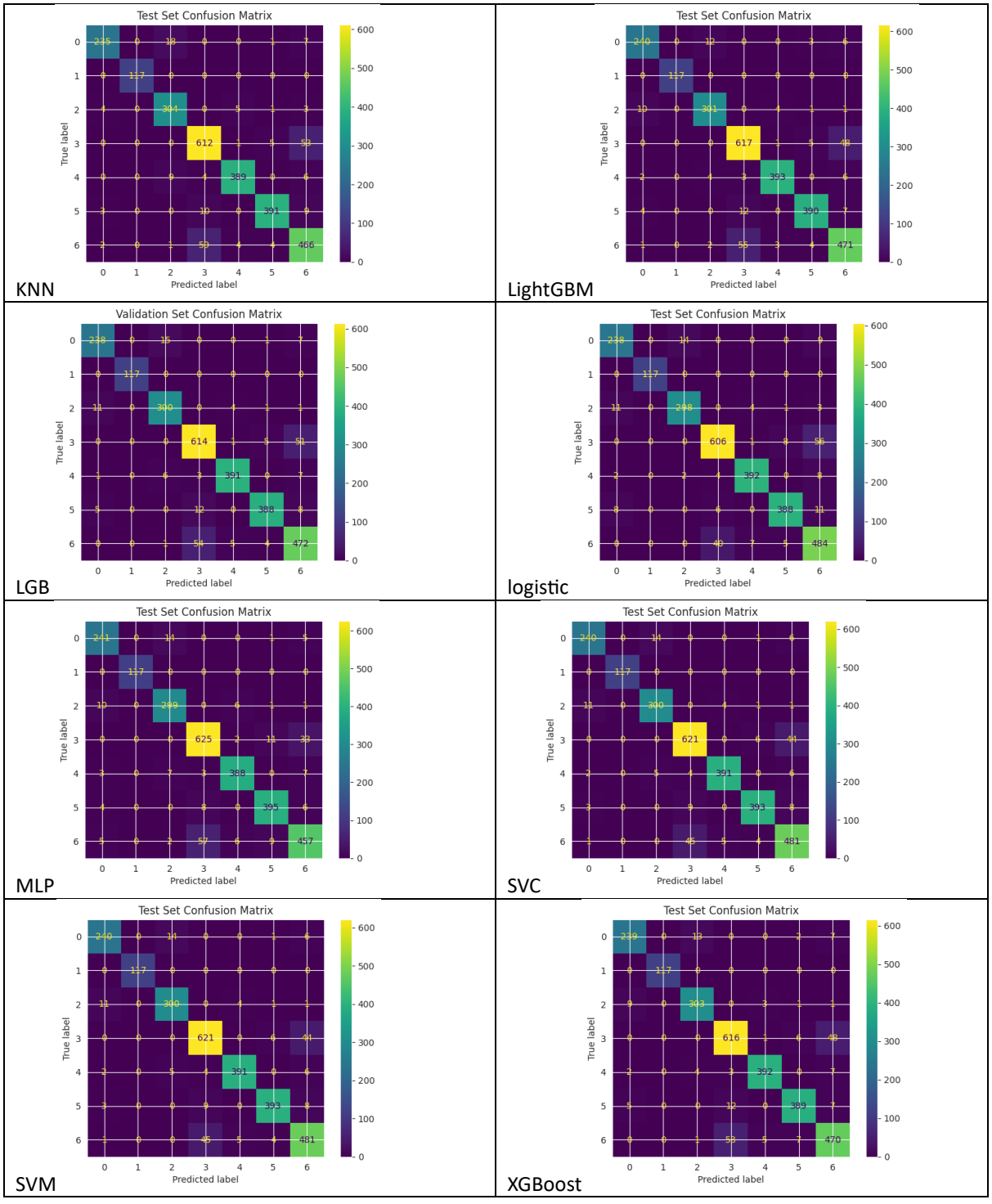
- booster: gbtree - learning_rate: 0.1 - max_depth: 7 - n_estimators: 200

CatBoost Model: CatBoost is a gradient boosting framework that is designed to be efficient and accurate. It is known for its ability to handle categorical features without requiring one-hot encoding. We used the following hyperparameters for this model:

- depth: 6 - iterations: 200 - learning_rate: 0.1

Here we can see these models confusion matrixes at a galance:





The evaluation of the models was performed using several metrics, including accuracy, precision, recall, F1-score, and ROC AUC score. The performance of each model was compared using a 5-fold cross-validation approach, where the dataset was split into 5 folds, and each fold was used once as a validation set while the other folds were used for training. The mean and standard deviation of the evaluation metrics were calculated over the 5 folds to obtain a robust estimate of the model performance. The results showed that the SVC model achieved the highest performance,while the DT model had the lowest performance.

Overall, the code is well-organized, easy to read, and well-documented. It provides a good starting point for anyone interested in developing a machine learning model for the dry bean classification problem.

Conclusion

In this project, we developed and evaluated several machine learning models for the dry bean classification problem. We implemented best practices in creating, sourcing, and preparing the training and test data. We also optimized the performance of the models by selecting the best hyperparameters and performing feature selection. We evaluated the performance and behavior of the models and analyzed and critiqued them with respect to the given problem. Our findings showed that the SVC model had the best performance, achieving an accuracy of 93%.

However, this project also had some limitations. One of the limitations was the small size of the dataset, which may have limited the performance of some models. Another limitation was the lack of explanation for the features, which made it difficult to interpret the results and understand which features were important for the classification problem.

Overall, this project provides a good starting point for anyone interested in developing a machine learning model for the dry bean classification problem. The code and results can be used as a reference for future work, and the limitations and recommendations can be used to improve the performance of the models in future work.