Introduction

The objective of this report is to address the problem statement of developing a system for the automatic detection of seven types of dry bean seeds using data captured through a high-resolution camera. To tackle this challenge, two machine learning models, namely Logistic Regression and Random Forest Classifier, have been employed. Prior to model development, an exploratory data analysis was conducted, followed by pre-processing steps to ensure the data's suitability for modelling. The subsequent sections present the methods used in model implementation, results obtained from the implemented models and conclusive insights and findings.

Exploratory Data Analysis

Data Description

The dataset consists of a total of 13,611 samples of seven different kinds of dry beans. The data was collected by using a computer vision system to take images, which were pre-processed and segmented, and then transformed into a binary format using Otsu's global thresholding method. As dry beans do not have a distinctive colour feature, dimensional and shape features were determined by feature analysis. Each sample has 16 features in total, 12 dimensional features and 4 shape features [1]. The labels which represents the different kinds of dry beans are categorical.

Granted, the dataset provides geometric information like size and shape, it couldn't cover all pertinent facets of the beans' characteristics. While these traits offer insightful information, adding more contextual information could improve classification accuracy and broaden understanding. This collection also lacks data on external variables including climate, geographical location, soil conditions, and growing conditions that may have an impact on the bean traits(including but not limited to the size and shape).

Data Cleaning

First, the provided .xlsx file was read and stored in a pandas DataFrame object `bean`. Checking the information of the data frame showed there were no null values. Next, the data frame was inspected for duplicate values and 68 duplicate rows found were removed.

Data Exploration

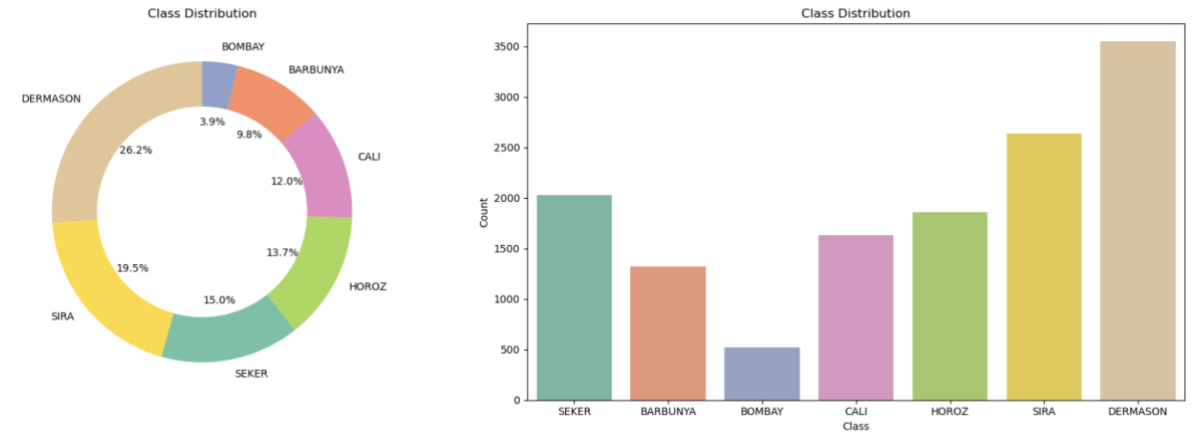

FIG 1 – CLASS DISTRIBUTION

DERMASON has the highest number of samples in the dataset, while BOMBAY has the least. FIG 1 above shows the dataset is imbalanced, and so, the level of skewness was investigated in FIG 2.

While checking for the relationships between features, some features had positive or negative correlation of 0.5 and above with 40 to 60% of the remaining features. This suggests the presence of linear relationships between them.

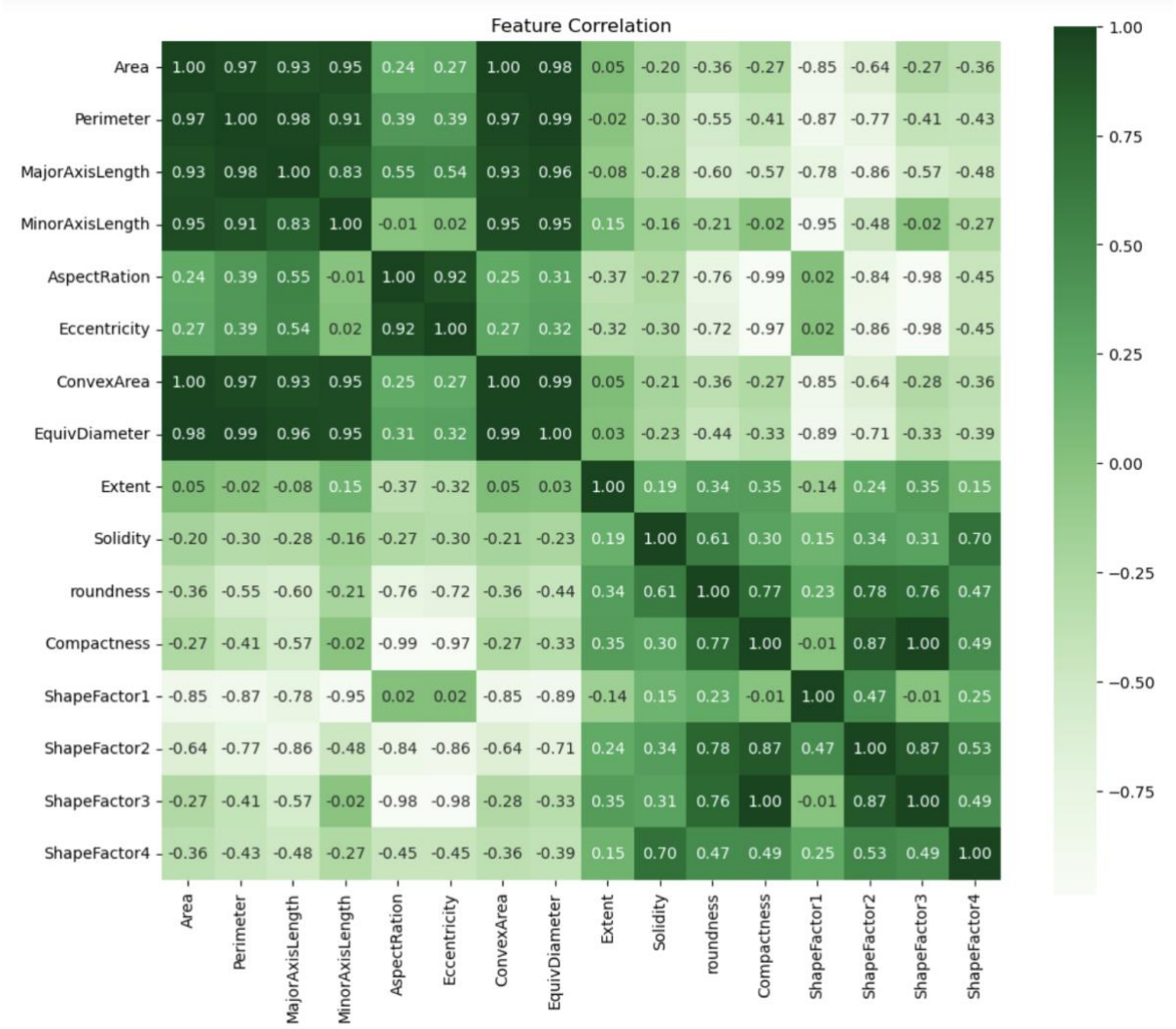

FIG 2 – FEATURE CORRELATION

Feature Engineering

Shape Factors 1 to 4 are features created by modifying existing ones [1]. As such, no additional feature engineering was done.

Data Splitting

The data was first split into two parts for training and testing. The training set was further split into two for training and validation. A random state of 23 was maintained to enable reproducibility of outputs.

Data Balancing – Oversampling

As the dataset was imbalanced, oversampling the minority classes as many times as needed to match the majorities was implemented using SMOTE(Synthetic Minority Over-sampling Technique) [2]. This was applied to the training data only, to ensure that the validation and test datasets remain as they are to ensure they reflect the real distribution of the dataset. Alternatives here are undersampling or hybrid-sampling. Oversampling was chosen to ensure there isn't a further reduction to the dataset.

Data Scaling

Since the features in the dataset had different scales, it was pertinent to normalize to bring these features into similar scale. This would ensure some features do not dominate the learning process. The scaling method used is the Standard scaling. An alternative here are robust scaling, min-max scaling and unit vector scaling.

Primary Model

The primary model for the problem of automatic detection of seven types of dry bean seeds is Logistic Regression(LR). It is a supervised learning algorithm used for binary classification. LR models the relationship between features and the probability of instances belonging to specific types of dry bean seeds using the logistic(sigmoid) function. This model is chosen due to the observed linear relationship between certain features.

The model development started with fitting the model on the training dataset. The dataset was imbalanced and unscaled, also, all hyperparameters was set to defaults. Next, the model was fitted on the oversampled dataset to observe performance. Following the observation here, the model was fitted on the scaled dataset and then the scaled oversampled dataset.

Next the model was fitted on the imbalanced dataset again to begin hyperparameter tuning, then again on the scaled imbalanced dataset with hyperparameter tuning. Alternatives for LR are Support Vector Machines, Gradient Boosting Models(XGBoost and LightGBM) and Neural Networks. LR also assumes feature independence, is sensitive to outliers, and may struggle with imbalanced data.

C-Regularization parameter

The first parameter that was tuned is C. This is the regularization parameter; it controls the inverse of the regularization strength [3]. A low value of C increases the regularization strength, while a higher value decreases it. This value is important to avoid overfitting to the training set, and that informed the decision to tune. This tuning was carried out on both unscaled and scaled datasets.

penalty-Regularization type

Logistic regression uses two types of penalty for regularization, L1(Lasso) and L2(Ridge) regularization [3]. L1(Lasso) regularization adds the absolute values of the coefficients as a penalty term to the loss function. The goal here is to encourage sparsity in the model by shrinking some coefficients to zero. L2(Ridge) regularization adds the squared values of the coefficients as a penalty term to the loss function. This makes the coefficient small, however they do not become zero and as such lead to a more continuous shrinking. This was tuned in models that was fitted on both unscaled and scaled data.

solver

The solver determines the algorithm used in the model's optimization. The default for this model is "lfbgs", however this solver only supports L2 regularization[3] and hence the choice to tune this parameter to include one that supports L1 regularization as well. Both "lfbgs" and "liblinear" as solvers were tested to find the best fit for the model trained on both unscaled and scaled datasets.

max_iter-Maximum number of iterations

This establishes the maximum iterations allowed for the solver to converge [3]. The model may not be adequately fitting the data if the solver doesn't converge before the predetermined number. To obtain convergence, this parameter may need to be changed. This was tuned find an optimal value for convergence. This was done with both unscaled and scaled dataset as well.

Supplementary Model

The additional model developed to address the problem statement is Random Forest Classifier. This classifier is an ensemble learning algorithm renowned for its efficiency in both classification and regression problems. Multiple decision trees are used to produce predictions, and the concept of majority voting is used to choose the final class label.
For the automatic recognition of seven different varieties of dried bean seeds, the Random Forest classifier is also appropriate option. It's a good model to address this issue since it can handle high-dimensional data, capture non-linear correlations, is robust against overfitting, and is computationally efficient.
The model development started with fitting the model on the training dataset. The dataset was imbalanced and unscaled, also, all hyperparameters was set to defaults. Next, the model was fitted on the oversampled dataset to observe performance. Following the observation here, the model was fitted on the scaled dataset.
Next the model was fitted on the imbalanced dataset again to begin hyperparameter tuning.

n_estimators-Number of Decision Trees

The hyperparameter `n_estimators` determines the quantity of decision trees utilized in the ensemble [4]. Each tree is constructed using a random subset of the training data, and the value of n_estimators governs how many trees are generated and combined for making predictions. Enhancing performance is typically achieved by increasing the number of trees, allowing the ensemble to capture intricate patterns within the data. However, there comes a point of diminishing returns, where adding more trees may not yield substantial performance improvements. By tuning this parameter, a balance can be struck between model performance and the available computational resources, ensuring optimal utilization of both.

max_depth

The hyperparameter `max_depth` defines the maximum allowed depth for each decision tree in the ensemble [4]. Increasing max_depth enables trees to grow deeper and capture intricate relationships within the training data. However, deep trees run the risk of overfitting by memorizing noise or irrelevant patterns in the training set. Tuning max_depth allows for controlling the depth of the trees, preventing overfitting, and enhancing the model's capacity to generalize to new, unseen data.

Results

The class with the best classification result is BOMBAY, this is because its features are distinctive from the other classes as seen in the `pairplot`(cell 8). Both models achieved scores of 1 for this class.

A 5-fold cross validation was implemented with grid search [5], the overall model evaluation results obtained from both models are presented below:

Model	Metric	Unscaled Data	Oversampled Data	Scaled Data	Scaled-Oversampled Data
LR	accuracy	0.70	0.70	0.92	0.77
	precision	0.71	0.72	0.93	0.79
	recall	0.70	0.72	0.93	0.83
	f1-score	0.70	0.72	0.93	0.79
RFC	accuracy	0.92	0.92	0.92	-
	precision	0.93	0.93	0.93	-
	recall	0.93	0.93	0.93	-
	f1-score	0.93	0.93	0.93	-

TABLE 1 – TRAINING RESULTS WITHOUT HYPERPARAMETER TUNING.

Model	Metric	Best C	Best penalty & solver	Best max_iter	Best parameters
LR	accuracy	0.70	0.90	0.91	0.91
	precision	0.72	0.92	0.92	0.92
	recall	0.70	0.91	0.92	0.92
	f1-score	0.71	0.91	0.92	0.92

TABLE 2 – TRAINING RESULTS WITH UNSCALED DATASETS AND TUNED HYPERPARAMETERS - LR

Model	Metric	Best C	Best penalty & solver	Best max_iter	Best parameters
LR	accuracy	0.92	0.92	0.92	0.92
	precision	0.93	0.93	0.93	0.93
	recall	0.93	0.93	0.93	0.93
	f1-score	0.93	0.93	0.93	0.93

TABLE 3 – TRAINING RESULTS WITH SCALED DATASETS AND TUNED HYPERPARAMETERS - LR

Model	Metric	Best n_estimators	Best max_depth	Best parameters
RFC	accuracy	0.92	0.92	0.92
	precision	0.93	0.93	0.93
	recall	0.93	0.93	0.93
	f1-score	0.93	0.93	0.93

TABLE 4 – TRAINING RESULTS WITH SCALED DATASETS AND TUNED HYPERPARAMETERS - RFC

TABLE 1 shows that using scaled data leads to improved performance for the LR model across all metrics, with higher accuracy, precision, and F1-score compared to unscaled, oversampled and scaled-oversampled data. As scaled data eliminates the risk of some features dominating the learning process.

TABLE 2 shows that tuning C alone did not improve the results significantly. To obtain the best penalty & solver combination, liblinear was tested with l1 and l2 respectively and the combination solver=liblinear, penalty=l1 outperformed solver=lbfgs, penalty=l2 as the lbfgs solver only supports l2 penalty type. Also, tuning max_iter alone achieved a significant effect on the model performance at the cost of having the solver converge after 7513 iterations. A combination of all these parameters through a grid search also improved the performance of the model, however it can be inferred that only the penalty, solver and max_iter hyperparameter tuning significantly improved the model's performance each time.

TABLE 3 shows a consistent LR model performance with hyperparameter tuning on the scaled dataset. No significant change was recorded in the model's performance with hyperparameter tuning. However, the best penalty & solver combination here was l2 and lbfgs. Tuning max_iter however showed that convergence was achieved after 100 iterations, and after 613 iterations when combined with other parameters. Both numbers are better than what was achieved with the unscaled dataset.

TABLE 4 shows a consistent performance from the RFC model with hyperparameter tuning on the original dataset. Scaling or oversampling did not affect this model as the results remained consistent.

Best hyperparameters

Model	Data type	Best C	Best penalty & solver	Best max_iter	Best parameters
LR	Unscaled	0.01	l1, liblinear	10000	100, l1|liblinear, 10000

	Scaled	1	l2, lbfgs	100	10, l2|lbfgs, 1000

TABLE-5 BEST HYPERPARAMETERS – LR

Model	Data type	Best n_estimators	Best max_depth	Best parameters
RFC	Scaled	300	None	400, None

TABLE-5 BEST HYPERPARAMETERS – RFC

Models Comparison

The comparison between the LR and RFC models reveals important insights. Scaling the dataset had a significant impact on the LR model, confirming its sensitivity to feature scales and the benefits of scaling in capturing relationships effectively. In contrast, the RFC model performed well without the need for data scaling, indicating its robustness in efficiently capturing relationships regardless of feature scales.

Comparison with previous work

The best model in [1] is SVM with and accuracy of 0.93 whereas RFC from this project achieved 0.92. This slight increase in SVM's performance could have been related to its ability to effectively capture non-linear relationships between features and targets using kernel functions. Additionally, SVM is more effective in handling unbalanced datasets by properly taking into account the minority class due to its capacity to modify class weights. As a result of potential bias towards the majority class in decision tree development, RFC could struggle with imbalanced data.

Model limitations

The models developed in this project may struggle to generalize if the training set is not a real world distribution as they depend heavily on the quantity and quality of data provided. Good feature engineering is only possible with good domain knowledge which wasn't considered in building these models, as such inadequate feature selection here may affect performance with another dataset.

Conclusion

The LR model's dependence on scaling suggests that it is particularly suited for datasets with similar scales and linear relationships. On the other hand, the RFC model's ability to perform effectively without scaling makes it a suitable choice for datasets with various feature sizes and non-linear connections. These findings highlight the need to consider the specific algorithm's sensitivity to feature scales when deciding whether to scale the data.

The final LR model had the following parameters `C = 100, max_iter = 1000, penalty = "l2", solver = "lbfgs".` The value of `C` here is high which means the regularization strength is reduced and the model could be vulnerable to overfitting.

As only `n_estimators` improved the RFC model performance slightly, the value for the final model was set to 400, because this provided a better result than when combined with other tuned hyperparameters.

The RFC model was selected as the best model because it produced the similar result as the best LR model, it did not require data scaling, and with the least hyperparameter tuning. Overall, understanding the characteristics of the dataset and the algorithms is crucial in making informed decisions about data pre-processing and model selection.

References

1. Koklu, M., Ozkan, I.A.: Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture. 174, 105507 (2020). https://doi.org/10.1016/j.compag.2020.105507
2. Olugbade, T.: Machine Learning Lecture Note - Model Validation I. (2023)
3. sklearn.linear_model.LogisticRegression, https://scikit-learn/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
4. sklearn.ensemble.RandomForestClassifier, https://scikit-learn/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
5. sklearn.model_selection.GridSearchCV, https://scikit-learn/stable/modules/generated/sklearn.model_selection.GridSearchCV.html