1. Introduction

This report analyses a dataset of dry bean images, including seven different types of beans with similar features such as form, shape, type, and structure. This research aimed to develop a computer vision system that can distinguish between the seven types of beans to achieve uniform seed classification. The dataset consists of high-resolution images of 13,611 grains of the seven types of dry beans, which were used to train the classification model.

One potential application of this model is in the agricultural industry, where it can be used to identify the types of dry beans in a large batch. This can help in the sorting and grading beans based on their type, which can ultimately improve the quality of the product and lead to increased revenue for farmers and suppliers.

However, it is essential to note that this dataset has its limitations. The feature extracted from the grains is limited to 16 features, 12 dimensions and four shape forms, which may not capture all the variations in the beans. Additionally, the dataset may not represent the entire population of dry beans, and the model may not perform well on beans that differ significantly from the ones in this dataset. Therefore, it is crucial to consider these limitations when interpreting the results of this study.

2. Methodology

SVM (support vector machine) algorithm was selected as the primary classification method because it effectively handles high-dimensional datasets with few instances. I will explain why I chose SVM and discuss the initial output that influenced this decision.

2.1 EDA

After importing the dataset, the first step was removing any duplicate rows and identifying and removing any null values present. Subsequently, a bar graph was created to visualise the frequency count of each bean type. A correlation matrix was plotted to explore the dataset further to investigate the relationship between the different variables. The correlation matrix was visualised using a heatmap to facilitate the rapid identification of positively or negatively correlated variables and the extent of correlation between them. This information helped determine the variables to be used in the development process and identify potential issues such as multicollinearity. Next, I plotted a grid of histograms with a bell-shaped curve superimposed on each. This allows for visualising the distribution of each variable in the dataset and identifying any potential outliers or non-normal distributions. Fitting each distribution to a normal distribution curve helps identify each variable's mean and standard deviation. This information can help select appropriate machine-learning models and set model parameters.

A pair plot is created to show scatter plots of all possible combinations of features(columns) of the input dataset' **df'**, dropping some columns. Then a boxplot is created to visualise the data distribution for each feature and identify any potential outliers that may need to be

addressed in the data pre-processing step. However, the outliers in the dataset were removed using the IQR method. **'Class'** was used to colour code the plot points, and **'height'** and **'diag_kind'** were set for subplot customisation.

2.2 Data Splitting and Feature Scaling

To evaluate the performance of our machine learning model, we have split the data into training and testing sets. The training set is utilised for training the model, whereas the testing test is evaluated the accuracy of the model's prediction on unseen data. I split the dataset using the **'train_test_split'** function from scikit-learn. The '**X**' variable contains all features except the target variable, stored in the **'y'** variable. The '**test_size'** parameter determines the portion of data used for testing, in this case, 20%, while the remaining 80% is used for training.

Feature scaling is a pre-processing step often applied to datasets before fitting a machine learning model. Feature scaling aims to avoid bias introduced by features with significantly different scales or units. The method which I used here is StandardScalar. It standardises features by subtracting the mean and dividing by the standard deviation. This method transforms the features into a mean of zero and a standard deviation of one. Standard Scalar is helpful for specific algorithms like SVM that assume the data is normally distributed. The code **sc.fit_transform(X_train)** fits the scalar to the training data (**X_train**) and applies the transformation. The code **sc.transform**(**X_test**) applies the transformation learned from the training data to the testing data(**X_test**).

2.3. Modelling – Support Vector Machines (SVM)

Support Vector Machines (SVMs) are well-known and widely used machine learning algorithms for classification and regression tasks. Although they are often used for binary classification, SVMs can also be used for multiclass classification. One of the main advantages of SVMs is their ability to high-dimensional data and complex classification problems, making them an excellent choice for modelling. In the dataset with multiple features and classes, SVMs have been shown to perform exceptionally well. Furthermore, SVMs can also handle non-linear decision boundaries using kernel methods, improving the model's accuracy. I tested various models before deciding that SVM was the model that best fit my task.

For the SVM classifier model and do the following task:

1. Instantiate an SVM classifier model with a linear kernel and set the random state to 42
2. Fit the model to the training set data.
3. Use the trained model to predict the class labels for the test data.
4. Evaluate the model's performance by calculating the accuracy of the predicted labels against the actual labels of the test set data.
5. Print the calculated accuracy score.

After printing the accuracy score, I decided to do the hyperparameter tuning for the SVM model for better accuracy. Hyperparameter involves trying different combinations of

hyperparameters and evaluating the model's performance for each combination until the best set of hyperparameters is found. Hyperparameter tuning is finding the optimal set of hyperparameters for a machine learning model that can maximise its performance on a given dataset.

Hyperparameter tuning for SVM.

I am performing hyperparameter tuning for the SVM model using GridSearchCV. Hyperparameters are parameters set before training and not learned from the data during training. In SVM, hyperparameters such as the kernel, C, and gamma can significantly impact the model's performance. To perform hyperparameter tuning with GridSearchCV, we establish a dictionary of hyperparameters and their values. GridSearchCV then conducts a thorough search all over all possible hyperparameter combinations to identify the optimal ones. The number of cross-validation folds to use during the search is specified by the 'cv' parameter. In contrast, the '**n_jobs'** parameter sets the number of CPUs to perform the computation in parallel. The parameters we are tuning in this example are kernel functions **('linear','rbf', or 'poly'**), the regularisation parameter C(0.1, 1, or 10) and the kernel coefficient gamma(0.1,1 or 'scale).

I also tried hyperparameter tuning with Bayesian optimisation for SVM. The optimisation uses the BayesSearchCV object from the Skopt package, which searches within the hyperparameter space specified by the params dictionary. Three hyperparameters- C, kernel, and gamma and their corresponding search ranges are included in the params dictionary. Gamma is a continuous hyperparameter with a range of 0.1 to 1 with a logarithmic prior distribution, C is a continuous hyperparameter with a range of 0.1 to 10, and kernel is a categorical hyperparameter with three possible values (**'liner', 'rbf', 'poly'**). The '**BayesSearchCV**' object is then fit to the training data, using 5- fold cross validation and 10 iterations of hyperparameter search.

3. Result and Analysis

Various Models Performance for Training Set

	Accuracy	F1-Score	Precision	Recall
Decision Tree	1.0000	1.0000	1.0000	1.0000
GradientBoosting Classifier	0.9626	0.962595	0.962635	0.962618
SVM	0.9276	0.927674	0.92779	0.927635
MLP	0.9323	0.932324	0.932519	0.932250

Various Models performance for Test set

	Accuracy	F1-Score	Precision	Recall
Decision Tree	0.8933	0.8933	0.8934	0.8933
GradientBoosting Classifier	0.9221	0.9221	0.9223	0.9221

SVM	0.9269	0.9270	0.9272	0.9269
MLP	0.9284	0.9286	0.9289	0.9284

The decision tree model exhibits 100% accuracy on the training set, indicating that the data may have been overfitted. With an accuracy of 0.8933, it still performs poorly on the test set. The Gradient Boosting model outperforms the Decision Tree model regarding accuracy on the training set, suggesting it may be less prone to overfitting. With an accuracy of 0.9221, it is also more accurate on the test set than the Decision Tree model.

The SVM model performs about as well as the previous two models; however, it has a lower accuracy on the training set (0.9276) than the former two. Additionally, it has an accuracy of 0.9269 on the test set, which is comparable. Compared to the SVM model, the MLP model has a slightly higher accuracy on the training set (0.9323). Additionally, it has an accuracy of 0.9284 on the test set, which is marginally better than the SVM model's accuracy.

The Gradient Boosting model, which has the highest accuracy on the test set, can be said to be the model that performs the best overall based on the accuracy metric. We might also wish to consider other metrics like F1 score, precision, and recall to understand the model performance better.

SVM

```
Best hyperparameters: {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
Accuracy: 0.9328165374677002
classification report
              precision    recall  f1-score   support

    BARBUNYA       0.95      0.92      0.93       259
      BOMBAY       1.00      1.00      1.00       114
        CALI       0.93      0.96      0.95       305
    DERMASON       0.92      0.93      0.93       707
       HOROZ       0.98      0.95      0.97       376
       SEKER       0.95      0.95      0.95       414
        SIRA       0.88      0.89      0.89       534

    accuracy                           0.93      2709
   macro avg       0.94      0.94      0.94      2709
weighted avg       0.93      0.93      0.93      2709
```

After using GridSearchCV to perform hyperparameter tuning, SVM produced the above results. The outcomes show that the accuracy of the SVM model with the hyperparameters tuned using GridSearchCV achieved an accuracy of 0.9269. The model correctly predicted most of the samples for each class, as seen from the high precision, recall, and F1- scores for most classes. However, as seen by the lower precision, recall, and F1- score values compared to other classes, it had trouble classifying the SIRA class. The best hyperparameters founded by GridSearchCV are C=1, gamma=0.1 and kernel='rbf'.

```
[[237   0  14   0   0   1   7]
 [  0 114   0   0   0   0   0]
 [  8   0 293   0   1   1   2]
 [  0   0   0 656   0  10  41]
 [  0   0   7   5 359   0   5]
 [  4   0   0   8   0 393   9]
 [  1   0   0  42   8   8 475]]
```

The confusion matrix above demonstrates that the model predicted instances correctly. Diagonal elements show the number of cases that were correctly classified. The class ‘SIRA’ exhibited the highest misclassifications, with 42. Class ‘BOMBAY’ has the lowest amount of misclassification since this class has no misclassification. Misclassification in other classes is comparatively rare.

The results for SVM when using BayesSearchCV are as follows.

```
Best hyperparameters: OrderedDict([('C', 7.366877378057127), ('gamma', 0.869700535740794), ('kernel', 'linear')])
Accuracy: 0.9265411590992987
classification report
              precision    recall  f1-score   support

    BARBUNYA       0.93      0.93      0.93       259
      BOMBAY       1.00      1.00      1.00       114
        CALI       0.93      0.94      0.94       305
    DERMASON       0.92      0.91      0.92       707
       HOROZ       0.97      0.95      0.96       376
       SEKER       0.96      0.94      0.95       414
        SIRA       0.87      0.89      0.88       534

    accuracy                           0.93      2709
   macro avg       0.94      0.94      0.94      2709
weighted avg       0.93      0.93      0.93      2709
```

The two findings appear similar when considering the precision, recall and f1-score measures. However, there are some minor differences in the scores for some classes. For example, in the first set of results, the precision and recall for class ‘BARBUNYA’ are slightly lower in the result of BayesSearchCV. On the other hand, the precision and recall for class ‘HOROZ’ are slightly higher for GridSearchCV method. Overall, the macro and weighted averages for the two methods are identical, demonstrating that the model’s overall performance is similar in both cases.

It is challenging to choose between BayesSearchCV and GridSearchCV. Both techniques' precision, recall and f1-score are slightly different, producing 93% accurate and comparable results. However, BayesSearchCV is an advanced optimisation technique that uses Bayesian optimisation to search for the best hyperparameters, whereas GridSearchCV exhaustively searches through a specified subset of hyperparameters. So, in general, BayesSearchCV may be a better option if ample space for hyperparameters needs to be explored.

4. Extra Task- GradientBoosting Classifier

The Gradient Boosting Classifier (GBC) is a supervised learning algorithm that combines several weak learning models into a single robust model. The boosting ensemble method used to generate this model entails building and training a series of weak models iteratively to enhance the model’s overall performance. GBC incrementally increased the model’s decision tree count, training each new tree to rectify the mistakes of the prior ones.

Compared to the primary model, the GBC model is a different algorithm with different hyperparameters from the primary model. The GBC model is a powerful ensemble method that can handle complex non-linear relationships and high dimensional data. It is known to be less prone to overfitting than other ensemble methods like Random Forest. On the other hand, the primary model is also a robust algorithm for classification but with different strengths and weaknesses.

I utilised the same dataset and pre-processed it in the same manner as the primary model to create and assess the GBC model. After that, I tuned the GBC model's hyperparameters using GridSearchCV. The learning rate, the number of trees, and the minimum number of samples needed to divide a node are among the modified hyperparameters. I used 5-fold cross-validation to assess how well the GBC model performed.

The precision, recall and F1-score assessment measures used to assess the primary model's performance were also used to assess the performance of the GBC model. The accuracy score for the GBC model was 0.9268. However, the GBC model had higher precision and recall for the 'SIRA' and 'HOROZ' classes.

In conclusion, the SVM model outperformed the GBC model in terms of accuracy and most evaluation metrics, with accuracy scores of 0.929 and 0.933; the GBC model and SVM model both performed well in categorising the bean dataset. The SVM model also achieved higher precision, recall and F1-score for most of the classes, which suggests that it is a better model overall.

5. Conclusion

In conclusion, this report explores the use of machine learning algorithms to classify dry beans. The primary model used is the Support Vector Machines (SVM) model, with hyperparameter tuning performed using Grid SearchCV to improve accuracy. The SVM model achieved an accuracy score of 0.9269, indicating its effectiveness in identifying the different types of dry beans in the dataset. Other metrics, such as precision, recall, and F1 score, were also considered to evaluate the model's performance comprehensively. A GradientBoosting Classifier model was also developed and evaluated, achieving an accuracy score of 0.9294.

Furthermore, some recommendations for future work, such as incorporating data augmentation to produce more training data and increase the generalisation of the models, data augmentation techniques like picture flipping, rotation and zooming can be applied and investigating misclassification of classes. Some classes were trickier to classify than others. The potential application of these models would be in the agricultural industry, such as identifying the type of dry beans in a large batch, making them valuable tools for improving efficiency and reducing costs.

6. Reference

Koklu, M. and Ozkan, I.A. (2020). Multiclass classification of dry beans using computer vision and machine learning techniques. *Computers and Electronics in Agriculture*, 174, p.105507. doi:https://doi.org/10.1016/j.compag.2020.105507.

Nicosia, G., Ojha, V., La Malfa, E., Jansen, G., Sciacca, V., Pardalos, P., Giuffrida, G. and Umeton, R. eds., (2020). Machine Learning, Optimization, and Data Science. Lecture Notes in Computer Science. Cham: Springer International Publishing. doi:https://doi.org/10.1007/978-3-030-64583-0.