# Using Machine Learning to Classify Dry Beans

Valentine Osegbo
*MSc Data Science and Computational Intelligence*
*Coventry University*
Coventry, United Kingdom
osegbov@uni.coventry.ac.uk

***Abstract*—Maintaining the purity of crop variety is an important tool in the global agricultural sector, in order to increase the germination rate, yield of crops during harvesting, and improve the overall quality of the seeds. This is typically done by manually sorting the seeds into various classes and this can be a difficult and labour-intensive task. The aim of this report is to use machine learning techniques to accurately classify commercial dry beans into seven categories; Cranberry Beans (Barbunya), Bombay Kidney beans, Cali Beans, Dermason Kidney Beans, Horoz Beans, Seker Beans and Sira Beans. The dataset used for this paper consists of 16 features that were pre-obtained by extracting the shape forms and dimensions from the images of 13611 sample seeds acquired via a computer vision system. The experiment will involve feature visualization and extraction using Principal Component Analysis (PCA) and the application of five machine learning techniques; Support Vector Machine (SVM), Logistic Regression, K-Nearest Neighbours (KNN), Random Forest Trees and the XGBoost, to create models that will accurately classify the dataset. Each individual machine learning method will be investigated and the performance of the models will be compared. XGBoost had the best performance with an accuracy of 95.7%.**

***Keywords—Dry beans, Machine learning, Classification, SVM, KNN, Logistic Regression, Random Forest, Naïve Bayes, PCA***

## I. INTRODUCTION

Dry beans, a.k.a. *Phaseolus Vulgaris* is the most popular and earliest cultivated of the thousands of legume species as it accounts for half of the legume grained consumed [1]. Dry beans are the primary source of protein for a majority of the world's population especially in developing countries. It is also rich in complex carbohydrate, fiber and essential vitamins and minerals to the diet, yet are low in fat and sodium and contain no cholesterol [2]. It was first cultivated in the upland regions of Latin America more than 7000 years ago (Kaplan, 1965; Kaplan and Kaplan, 1988; Gepts and Debouck, 1991) and spread to most parts of the americas though immigration and to Africa during the slave trade and colonial period [3]. Dry beans seed comes in different varieties, such as colour, shape, texture, and can range in size from 50mg to about 2000mg. Modern technology has made such a massive impact in the global agriculture, and has revolutionised what was once perceived to technology-hostile sector. From the application of control systems in monitoring and adjusting the environmental factors responsible with sensors and automated actuators with the goal of improving crop quality and yield to the application of biotechnology to increase crop production by a targeted modification of the genetic structure of the crop, making it resistant to certain diseases and the effects of herbicides and pesticides, we can all see that modern technology is playing a massive role in the struggle to meet the global food consumption needs. In 2016 alone, the United Nations Food Agricultural Organization (FAO) estimated the quantity of dry beans produced globally to be 26.8 million tonnes with Myanmar and India on record as the largest producers of this species of common beans, both yielding 5.2 million tonnes and 3.8 million tonnes of dry beans respectively [4]. In recent times, the advent of big data technologies has paved a way for machine learning to be applied in the field of agriculture mainly for the optimization of cultivation processes which ultimately gives rise to significant increase of efficiency of the systems. Ramos P.J. et al. attempted to count, measure the weights and maturity percentage using computer vision system and classify coffee fruits into three categories; harvestable, not harvestable and fruits with disregarded maturation stage, with the aim of providing coffee farmers with useful information that would be beneficial especially for the optimization of certain agricultural processes [5]. Of the three machine learning techniques applied in the above paper, Support Vector Machine had the best performance with an overall classification success rate of 88.02%. Moushou D. et al designed a model using Quadratic Discriminant Analysis, Artificial Neural Networks and spectral reflectance that classifies wheat into healthy and infected with the purpose of detecting the yellow rust disease which can help improve the target precision of pesticides used in the field [6]. The results of this paper showed that the Multi-Layer Perceptron model performed better as it was able to correctly classify 98.9% and 99.4% of the healthy and diseased wheat canopies respectively. Grinblat G.L. et al developed a method of classifying legumes into three labels; white beans, red beans and soy beans, based on their respective leaf vein pattern [7]. For this paper, Convolutional Neural Networks (CNN) was used to create the image classification model for this dataset with an outstanding 96.9% classification success rate. The importance of modern technology and machine learning in agriculture cannot be over-emphasized and it is crucial for world governments to encourage more research in this field to get a better understanding of how to provide adequate and quality crops for human consumption through efficient and sustainable means.

## II. PROBLEM DEFINITION AND DATASET

### A. Problem

One of the major factors influencing the yield of Dry Beans crop is the costly side-effect of seedborne diseases, therefore the quality of the seeds plays an important role in increasing productivity by controlling the likelihood of the seeds getting infected [8]. Seed purity is one way of ensuring

good quality of the beans crop during harvest because distribution of pesticides developed for the particular crop species will work more effectively across a uniform field and, cultivation of a singular type of Dry beans per area will reduce the risk of spreading diseases from one species to another which is a common occurrence. Traditionally, the task is performed by employing labour in the form of human experts, to identify and appropriately sort each dry beans seed by its species. However, this method has several limitations including; the high financial cost of employing human experts, the productivity cost that may arise due to human error, for example, due to fatigue, and the time cost implications of completing the task by human experts as it may be very time consuming. To reduce these limitations, this paper proposes a method of classifying Dry Beans seeds into seven classes using 5 machine learning techniques. The classes can identify as Cranberry Beans (Barbunya), Bombay Kidney beans, Cali Beans, Dermason Kidney Beans, Horoz Beans, Seker Beans and Sira Beans. These machine learning models can be further used to detect the class to which an unidentified dry beans seed belongs to. Support Vector Machine (SVM), Logistic Regression, K-Nearest Neighbours (KNN), Random Forest Trees and the Naïve Bayes ML methods will be applied and their individual performances will be compared and further discussed. Learning will be supervised in this paper, and the dataset will be initially cleaned and pre-processed to improve performance of the models.

### *B. Dataset*

The dataset was originally sourced from UCI Machine Repository and the link to this webpage can be found in the Appendix. It consists of 17 attributes including the target, extracted by analysing the images of seven classes of Dry beans using Computer Vision System. The geometric features of the Dry bean seeds were extracted, using MATLAB software from the images. [9] These input features include:

- Area ($A$): Area of the beans is computed by measuring the number of pixels describing the bean seed within the image and can be expressed as:

$$A = \sum_{r,c \in R} 1$$

Where $r, c$ is the size of the region $R$

- Perimeter ( $P$ ): This is defined as the circumference of the seed and is a measure of the length of the pixel boundaries.
- Major Axis Length ($L$): This is defined as the length of the distance between the two furthest points on the seed.
- Minor Axis Length ($l$): This is the length of the distance between the two furthest points perpendicular to the major axis.
- Aspect Ratio ( $K$ ): This is defined as the relationship between the Major Axis length and the Minor Axis length and can be expressed as:

$$K = L/l$$

- Eccentricity ($Ec$): This can be defined as the eccentricity of the ellipse used to describe the region.
- Convex Area ($C$): This is the number of pixel in the smallest convex polygon that can be used to represent the area of the bean seed.
- Equivalent Diameter ($Ed$): This is defined as the diameter of a circle having an are equal to that of the bean seed and is expressed as:

$$Ed = \sqrt{\frac{4 * A}{\pi}}$$

- Extent ( $Ex$ ): The ratio of the pixels in the bounding box to the pixels in the bean seed and is expressed as:

$$Ex = \frac{A}{Ab}$$

Where $Ab$ is the Area of the bounding rectangle

- Solidity (S): This is defined as the ratio of the pixels in the convex shells to those found in the beans. It is also referred to as Convexity and can be expressed as:

$$S = \frac{A}{C}$$

- Roundness (R): This can be expressed as:

$$R = \frac{4\pi A}{P^2}$$

- Compactness ($CO$): This can be described as the degree of roundness of the bean seed and is expressed as:

$$CO = \frac{Ed}{L}$$

- Shape Factor 1 (SF1):This is expressed as:

$$SF1 = L/A$$

- Shape Factor 2 (SF2):This is expressed as:

$$SF2 = l/A$$

- Shape Factor 3 (SF3):This is expressed as:

$$SF3 = \frac{A}{\left(\frac{L}{2}\right) * \left(\frac{L}{2}\right) * \pi}$$

- Shape Factor 3 (SF3):This is expressed as:

$$SF3 = \frac{A}{\left(\frac{L}{2}\right) * \left(\frac{l}{2}\right) * \pi}$$

From figure 1 below, it is observable that some features have a rather strong correlation, which could either be positive or negative, to other features in the dataset. This dataset will initially be subjected to pre-processing and feature extraction before it can be used to train the ML models so as to increase

the models' accuracy in correctly classifying the dry bean seeds.

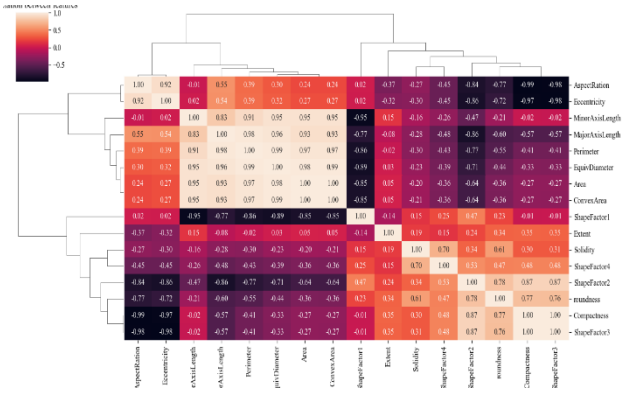


Fig 1. CORRELATION BETWEEN THE FEATURES

## III. DATA PREPARATION

Data pre-processing is an essential phase in Machine Learning as the performance of the model is mostly affected by the quality of data. This typically involves the transforming the original data into an object that is more easily readable by the machine. For this paper, pre-processing was performed in Python so as to take advantage of its rich library of resources, the dataset that was used contained no missing values and all values of the independent variables were real numbers. The next step in pre-processing is detection of outliers in the dataset. An outlier can be generally described as a value or observation that is distinct from others or doesn't fit the overall trend of the data. While in some cases this could signify a unique occurrence in the data, for example, in Fraud Detection, however in Machine Learning, the presence of outliers can reduce the performance of the models as it affects the mean and standard deviation of the distribution of each feature. One approach of detecting outliers is by defining statistical boundaries where all data should not exceed. This code was written in python (see Appendix) and the upper and lower boundaries are expressed as follows:

$$Upper\ boundary = Q3 + (1.5 * IQR)$$
$$Lower\ boundary = Q1 - (1.5 * IQR)$$

Where
$Q3$ is the Third Quartile (75th percentile)
$Q1$ is the First Quartile (25th percentile)
$IQR$ is the Inter-Quartile Range ($Q3 - Q1$)

### A. Feature Scaling

Earlier, as a part of pre-processing, we discussed about restricting the range of the distribution of each feature in the dataset in order to detect anomalies or outliers present in the features as these outliers can cause a shift in the mean and spread of the distribution which can lead to unreliable results from the ML models. Features in a dataset are measured in units of various numerical sizes and this usually creates a performance problem for some ML techniques because the features with significantly larger values tend to have a bigger effect on the result than those with relatively smaller values, thus introducing a bias into the model and could have a negative impact on the performance. Feature scaling is a crucial step in machine learning because it reduces the size of feature space to a specific range in an attempt to balance the contribution of each feature.

### B. Standardization

A popular method of feature scaling and also the preferred method for this paper is Standardisation or Z-Normalization. This method assumes the samples in the dataset can be described by a gaussian distribution and so recentres the data by fitting the features into a gaussian distribution of mean 0 and standard deviation of 1. This can be mathematically expressed as:

$$\hat{X}[:,i] = \frac{X[:,i] - \mu_i}{\sigma_i}, \left( \mu_i = \sum_{k=1}^{N} X[k,i], \quad \sigma_i = \sqrt{\frac{1}{N-1} * \sum_{k=1}^{N} (X[k,i] - \mu_i)^2} \right)$$

Where

$X$ is the dataset

$N$ is the number of rows or samples

$D$ is the number of columns or features

$X[:,i]$ is the ith feature

$\mu_i$ is the mean of the ith feature

$\sigma_i$ is the standard deviation of the $i$th feature

Although this method assumes the features are gaussian distributed, it is still as effective with non-gaussian distribution and is not sensitive to outliers.

### C. Class Imbalance

The target column comprises of cells containing the class of Dry bean seed each sample belongs to and will be used to both train (supervised learning) and measure the performance of the machine learning models. From the bar chart illustrated below, it can be observed that there is an imbalance in the sizes of each class in the target feature. This can have a negative effect on the models as it might introduce a bias against the classes with lower number of samples. Different techniques of data pre-processing have been proposed to handle the issue of class imbalance but for this paper, we will apply Synthetic Minority Oversampling Technique (SMOTE) and also investigate its effect on the performance on the ML models. The effect of SMOTE can be visualized in the bar charts below.

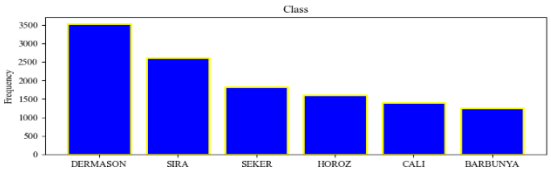



Fig 2. CLASS BALANCE BEFORE APPLYING SMOTE

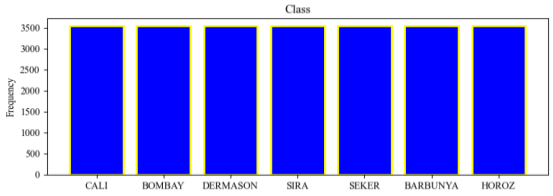



Fig 3. CLASS BALANCE AFTER APPLYING SMOTE

### D. *Feature Extraction*

Dimensionality reduction is a critical step in machine learning because it can help improve the performance of the models by identifying and extracting the principal or more important features in the dataset under consideration while also minimizing information loss. This is especially useful when the dataset has a high number of features and is plagued by the curse of dimensionality. The curse of dimensionality is a phenomenon in machine learning that explains the non-linear relationship between change in the number of features and samples respectively. In other words, the number of samples required to train a model increases proportionately as the number of dimensions in the feature space increases but not beyond a certain degree. The method of feature extraction that will be investigated for this paper is the Principal Component Analysis (PCA) Dimensionality Reduction. Principal component analysis performs dimensionality reduction by transforming the variables in the dataset, usually very large, into n number of Principal Components by measuring and comparing the variations of the features in the dataset. The number of principal components required to sufficiently represent the data can be determined by visualizing the explained variance ratio (which can be described as the amount of information stored) of each principal component as shown in the first two figures below. It is noticeable that Principal Components 1 and 2 account for 81.9% of the explained variance and also, 99% of the explained variance is contained in the first 8 components. Thus, the dataset can be reduced from 16 to 8 dimensions using PCA while still keeping 99% of the information. We can also visualize the relationship between the features in the dataset and the principal components PC1 and PC2 with the two highest explained variance ratios using a 2D PCA Bi-plot as demonstrated below. We can observe from the bi-plot that the samples appear to be grouped implying that PC1 and PC2 can be successfully used to train a classifier. An investigation of the effect of PCA on the performance of machine learning models under consideration will be performed as part of this paper.

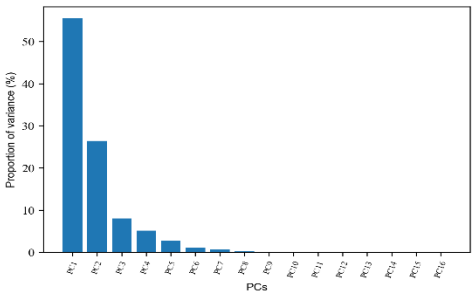



Fig. 4 SCREE PLOT OF THE PRINCIPAL COMPONENTS

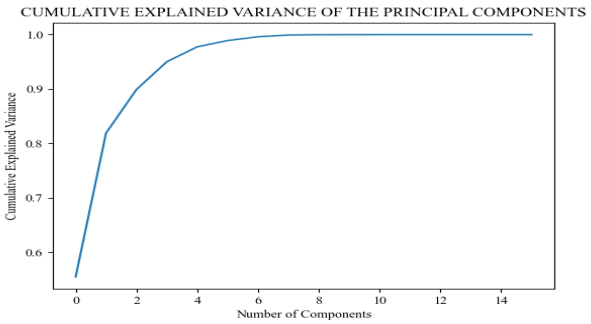



Fig 5. CUMULATIVE EXPLAINED VARIANCE RATIO PER PRINCIPAL COMPONENT

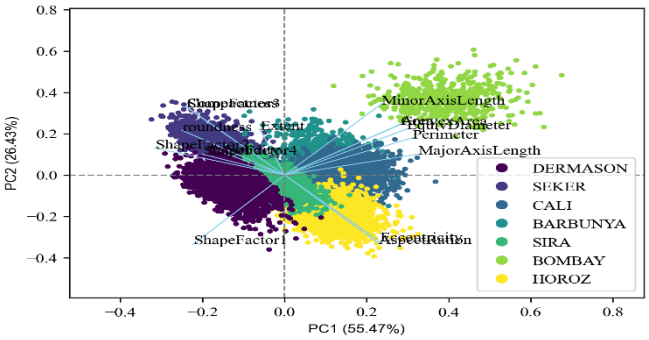



Fig 6. 2D BI-PLOT OF PRINCIPAL COMPONENTS 1 AND 2

## IV. METHODOLOGY

### A. *Random Forest Decision Trees*

Before we can talk about Random Forest, we need to first understand the Decision tree ML algorithm works. Simply put, it is a supervised machine learning algorithm that can be applied to solve regression and classification tasks. Its ease of understanding and application makes it the one of the most popular algorithms (Surjeet Kumar et al, 2012). At each node, the algorithm makes a decision based on the attributes under consideration and splits the node (partitions the data). The attribute with the highest Gini impurity or information gain is selected for this step and the process is repeated until one of the stopping criteria is met. This could either be if a "pure node" is reached or there is no increase in information gain for the attributes. Random Forest can be described as a set of multiple decision trees because each node in the Random Forest algorithm represents a random subset of features which will be used to build a decision tree. The output of the decision trees will then be combined to generate a more reliable final output, making it also a type of ensemble learning as shown in the figure 7 below.

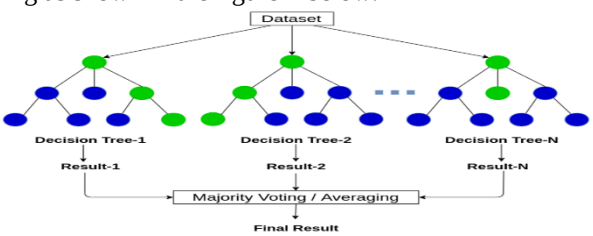



Fig 7. STRUCTURE OF A RANDOM FOREST LEARNING ALGORITHM

### B. Support Vector Machine

The Support Vector Machine is a machine learning algorithm which performs classification on a dataset by locating the optimum hyperplane that maximizes the degree of separation between the classes in the dataset while also reducing the misclassification error (in the case of non-linearly separable classes). This is initiated by mapping the dataset into a high dimensional space where it is linearly separable using a kernel function.

### C. Logistic Regression

The Logistic regression ML technique defines the relationship between the feature and target variables as a sigmoid function, assuming Y follows a binomial distribution (for binary classification) or a multinomial distribution (for multi-class classification). For binary classification in a 2-dimensional feature space, the output Y can be mathematically expressed as:

$$P(Y = 1|X = x) = \frac{1}{1 + e^{-(\beta_0 + x\beta_1)}}$$

$$P(Y = 0|X = x) = 1 - P(Y = 1|X = x) = \frac{1}{1 + e^{\beta_0 + x\beta_1}}$$

For $K$-class classification, the output is defined as follows:

$$P(Y = 1|X = x) = P(Y = K)e^{\beta_1 . x_i}$$
$$P(Y = 2|X = x) = P(Y = K)e^{\beta_2 . x_i}$$
$$P(Y = K - 1|X = x) = P(Y = K)e^{\beta_{K-1} . x_i}$$

Where

$$P(Y = K) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\beta_k . x_i}}$$

$x_i$ is a row vector containing the input variables for the $i$th sample

The optimum value of the parameter $\hat{\theta} = [\beta_0, \beta_1, \dots, \beta_k]$ can be found by estimating the maximum a posterior (MAP).

### D. K-Nearest Neighbours

The K-Nearest Neighbours algorithm is an instance based, supervised machine learning algorithm that can be used on both classification and regression problems and is particularly useful in dealing with complex target functions. The algorithm determines the class of a test input by calculating the distance between the query input and the training samples and selecting the most common class from the K samples closest to the query input. The most popular method of finding the distance is through calculating the Euclidean distance as shown in the equation below:

$$d\left(x^{(i)}, x^{(j)}\right) = \sqrt{\sum_{f=1}^{n} (x^{(i)}{}_f - x^{(j)}{}_f)^2}$$

Where $n$ is the number of features

### E. Extreme Gradient Boosting Ensemble

Gradient boosting can be described as an ensemble of the decision trees ML algorithm. It generally involves applying gradient descent method to optimize the decision trees by evaluating each split based on a specified loss function. Extreme Gradient boosting algorithm or XGBoost is a python library package that implements a variant of gradient boosting to a high degree of performance while also increasing the speed of computation.

## V. EXPERIMENTAL SETUP AND RESULTS

For the purpose of this paper, the dataset will be split to training and testing into a ratio 80:20. The hyper-parameters are then optimized for the models by combining a Randomized Search grid with 5-Fold Cross Validation. Doing this significantly reduces any bias in the dataset, particularly useful for smaller datasets. The randomised search of 30 iterations will be used to tune selected hyper-parameters in the machine learning models under consideration and the results are displayed below in TABLE 1, TABLE 3, TABLE 5, TABLE 7 and TABLE 9, and the performance of the models using these optimized hyper-parameters will be further investigated as shown in TABLE 2, TABLE 4, TABLE 6, TABLE 8 and TABLE 10

### A. Random Forest Classification

TABLE I HYPER-PARAMETER OPTIMIZATION

| **Hyper-parameters *(Default value)*** | **Optimized values** | |
|---|---|---|
| | ***Original Dataset*** | ***SMOTE Dataset*** |
| *n_estimators (100)* | 789 | 1069 |
| *max_features (auto)* | log2 | sqrt |
| *max_depth (none)* | 50 | 30 |
| *min_samples_split (2)* | 10 | 4 |
| *min_samples_leaf (1)* | 5 | 1 |

The table below shows the result of training the Random Forest Model using the dataset:

TABLE 2 RANDOM FOREST PERFORMANCE RESULTS

| | **Original Dataset** | | | **SMOTE Dataset** | | |
|---|---|---|---|---|---|---|
| | ***Unextracted Features*** | | ***PCA n_component = 8*** | ***Unextracted Features*** | | ***PCA n_component = 8*** |
| ***Performance metrics*** | ***Default Settings*** | ***Optimized Hyper-parameters*** | ***Default Settings*** | ***Default Settings*** | ***Optimized Hyper-parameters*** | ***Default Settings*** |
| *Accuracy* | 0.923 | 0.920 | 0.921 | 0.951 | 0.951 | 0.954 |
| *F1-Measure* | 0.935 | 0.932 | 0.934 | 0.951 | 0.952 | 0.955 |
| *Precision* | 0.937 | 0.935 | 0.936 | 0.952 | 0.952 | 0.955 |
| *Recall* | 0.932 | 0.930 | 0.932 | 0.951 | 0.952 | 0.955 |

From the TABLE 2. above, it can be seen that balancing the class using SMOTE improved the performance of the model and also, extracting 8 Principal components from the SMOTE balanced dataset gave the best results when further

used to train the random forest model. The ideal number of PCs can be evaluated by plotting the performance of the RF model cross-validation using varying principal components as shown in the Figure 8 below. 5-Fold cross-validation was used for this experiment and the results show that there is not much f1-measure improvement upwards of 5 Components. For results of this experiment on the other machine learning models, please see appendix.

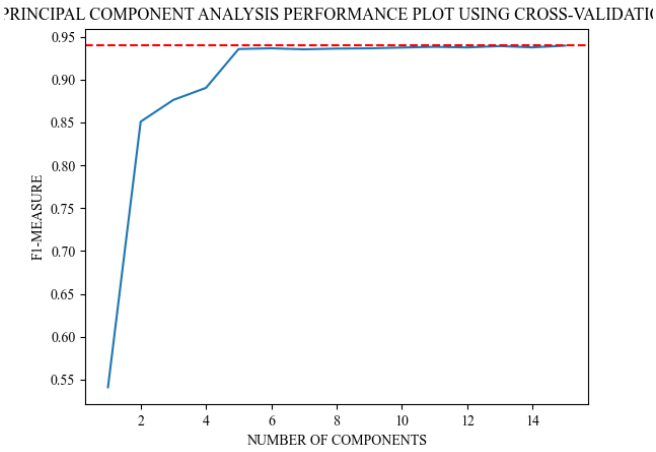



Fig 8. RANDOM FOREST MODEL EVALUATION OF PRINCIPAL COMPONENTS

The method above can also be used to investigate the effects of increasing the number of folds for K-Fold cross-validation on the performance of the model in order to select the optimum number of folds as shown in Figure 9 below.

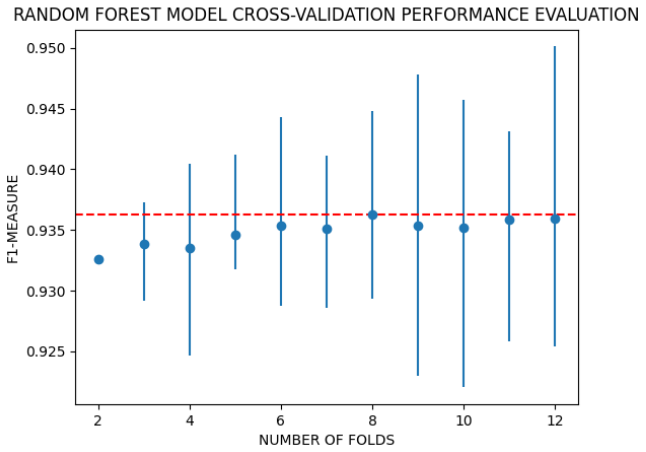



Fig 9. RANDOM FOREST MODEL EVALUATION OF K-FOLD CROSS VALIDATION

## B. *Support Vector Machine*

TABLE 3 HYPER-PARAMETER OPTIMIZATION

| **Hyper-parameters** ***(Default value)*** | **Optimized values** | |
|---|---|---|
| | ***Original Dataset*** | ***SMOTE Dataset*** |
| *C (1)* | 1 | 1 |
| *gamma (scale)* | 0.1 | 1 |
| *kernel (rbf)* | rbf | rbf |
| *degree (3)* | 5 | 7 |

TABLE 4 SUPPORT VECTOR MACHINE PERFORMANCE RESULTS

| | **Original Dataset** | | | **SMOTE Dataset** | | |
|---|---|---|---|---|---|---|
| | ***Unextracted Features*** | | ***PCA n_component = 8*** | ***Unextracted Features*** | | ***PCA n_component = 8*** |
| ***Performance metrics*** | ***Default Settings*** | ***Optimized Hyper-parameters*** | ***Default Settings*** | ***Default Settings*** | ***Optimized Hyper-parameters*** | ***Default Settings*** |
| *Accuracy* | 0.928 | 0.930 | 0.928 | 0.940 | 0.952 | 0.941 |
| *F1-Measure* | 0.940 | 0.942 | 0.940 | 0.941 | 0.952 | 0.942 |
| *Precision* | 0.943 | 0.945 | 0.943 | 0.941 | 0.952 | 0.942 |
| *Recall* | 0.937 | 0.939 | 0.937 | 0.941 | 0.952 | 0.942 |

From the results in TABLE 4, we can observe that Support Vector Machine algorithm performed slightly better on the model with optimized hyper-parameters than with the default hyper-parameters shown in TABLE 2. So far, applying SMOTE on the dataset improves the performance of our model.

## C. *Logistic Regression*

TABLE 5 HYPER-PARAMETER OPTIMIZATION

| **Hyper-parameters** ***(Default value)*** | **Optimized values** | |
|---|---|---|
| | ***Original Dataset*** | ***SMOTE Dataset*** |
| *C (1)* | 0.0011 | 2.4955 |
| *max_iter (100)* | 800 | 800 |
| *penalty (l2)* | none | none |
| *solver (lbfgs)* | lbfgs | lbfgs |

TABLE 6 LOGISTIC REGRESSION PERFORMANCE RESULTS

| | Original Dataset | | | SMOTE Dataset | | |
|---|---|---|---|---|---|---|
| | *Unextracted Features* | | *PCA n_component = 8* | *Unextracted Features* | | *PCA n_component = 8* |
| ***Performance metrics*** | ***Default Settings*** | ***Optimized Hyper-parameters*** | ***Default Settings*** | ***Default Settings*** | ***Optimized Hyper-parameters*** | ***Default Settings*** |
| *Accuracy* | 0.925 | 0.928 | 0.925 | 0.935 | 0.938 | 0.935 |
| *F1-Measure* | 0.936 | 0.940 | 0.936 | 0.936 | 0.939 | 0.936 |
| *Precision* | 0.938 | 0.942 | 0.938 | 0.936 | 0.939 | 0.936 |
| *Recall* | 0.935 | 0.938 | 0.935 | 0.936 | 0.938 | 0.935 |

As shown in TABLE 6 above, there was a small improvement in the logistic regression model after applying SMOTE to the dataset and tuning the hyper-parameters.

*D. K-Nearest Neighbours*

TABLE 7 HYPER-PARAMETER OPTIMIZATION

| **Hyper-parameters *(Default value)*** | **Optimized values** | |
|---|---|---|
| | ***Original Dataset*** | ***SMOTE Dataset*** |
| *n_neighbor (5)* | 16 | 4 |
| *leaf_size (30)* | 5 | 11 |

TABLE 8 K-NEAREST NEIGHBORS PERFORMANCE RESULTS

| | Original Dataset | | | SMOTE Dataset | | |
|---|---|---|---|---|---|---|
| | *Unextracted Features* | | *PCA n_component = 8* | *Unextracted Features* | | *PCA n_component = 8* |
| ***Performance metrics*** | ***Default Settings*** | ***Optimized Hyper-params*** | ***Default Settings*** | ***Default Settings*** | ***Optimized Hyper-params*** | ***Default Settings*** |
| *Accuracy* | 0.921 | 0.916 | 0.921 | 0.944 | 0.946 | 0.944 |
| *F1-Measure* | 0.933 | 0.929 | 0.933 | 0.945 | 0.947 | 0.945 |
| *Precision* | 0.937 | 0.933 | 0.937 | 0.945 | 0.947 | 0.945 |
| *Recall* | 0.930 | 0.926 | 0.930 | 0.945 | 0.947 | 0.945 |

From TABLE 8 above, it is also evident that applying SMOTE technique on the dataset increases the performance of the KNN model.

*E. Extreme Gradient Boosting Ensemble*

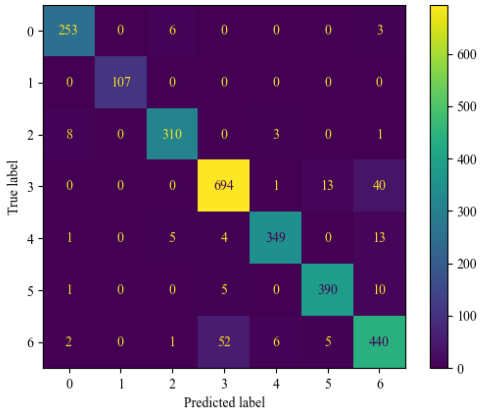
TABLE 9 HYPER-PARAMETER OPTIMIZATION

| **Hyper-parameters *(Default value)*** | **Optimized values** | |
|---|---|---|
| | ***Original Dataset*** | ***SMOTE Dataset*** |
| *min_child_weight(5)* | 5 | 1 |
| *max_depth (30)* | 4 | 15 |
| *learning_rate (5)* | 0.1 | 0.25 |
| *gamma (30)* | 0.0 | 0.0 |
| *colsample_bytree (30)* | 0.7 | 0.7 |

TABLE 10 EXTREME GRADIENT BOOST PERFORMANCE RESULTS

| | **Original Dataset** | | | **SMOTE Dataset** | | |
|---|---|---|---|---|---|---|
| | ***Unextracted features*** | | ***PCA =8*** | ***Unextracted features*** | | ***PCA =8*** |
| ***Metrics*** | ***Default*** | ***Optimized*** | ***Default*** | ***Default*** | ***Optimized*** | ***Default*** |
| *Accuracy* | 0.927 | 0.925 | 0.923 | 0.954 | 0.957 | 0.955 |
| *F1-Measure* | 0.941 | 0.939 | 0.937 | 0.955 | 0.957 | 0.956 |
| *Precision* | 0.944 | 0.941 | 0.940 | 0.955 | 0.958 | 0.956 |
| *Recall* | 0.939 | 0.936 | 0.934 | 0.955 | 0.957 | 0.956 |

CONFUSION MATRIX PLOT OF XGBOOST MODEL




From the overall results of all the experiments carried out shown in the illustrations above, it is visible that the Extreme Gradient boost model had the best performance on the dataset of all the machine learning models discussed in this paper. In addition, applying SMOTE class balancing technique on the

dataset slightly improved the performance of the models in all cases although the results obtained from optimization of the hyper-parameters and PCA dimension reduction gave mixed results as some models performed better without PCA and hyper-parameter optimization. For more visual illustrations including the confusion matrix plots of the other ML techniques under investigation, please refer to the appendix.

## VI. DISCUSSIONS AND CONCLUSION

It is important to note that the performance of the model does not depends on several significant factors including machine learning technique, hyper-parameter optimization and the quality of the data. In other words, it is impossible to presume which algorithm will perform better on a given dataset (Ho and Pepyne, 2002). The python Gradient Boosting implementation XGBoost yielded the best results in terms of accurately predicting the class to which an unknown dry beans seed belongs to while also ensuring the model doesn't overfit the data. The K-Nearest Neighbours ML algorithm, although with the worst performance of the five algorithms under investigation, yielded high results and also speeds up the model by processing the applying the training dataset on the model only at the point of query processing. This can also be referred to as lazy learning. A more exhaustive search such as a grid search can yield better optimized hyper-parameters which could increase the performance of the models. Principal component analysis (PCA) also helps in speeding up algorithms by reducing the complexity of the problem dataset although the data might be difficult to interpret after transformation. It is important to cross-validate our models when optimising the hyper-parameters in order to reduce the bias that might be present in the especially smaller datasets. Traditionally for KFold cross-validation, k=5 or k=10 is preferred but we can determine the optimum number of folds by evaluating the model's performance using different number of folds as we can observe in Figure 9. Please refer to the appendix for KFold evaluations of other four models. The F1-measure was the preferred metric for evaluating the models because it is more reliable than the other 3 metrics considered for this paper (accuracy, precision and recall).

To conclude, accurate classification of dry beans seeds will not only help to reduce the cost of labour that would have been needed to manually sort the seeds but also improve the overall quality of the seeds by reducing human error. This could have positive economic and humanitarian implications of the availability of more quality beans crops. Please note that one of the main aspects of this problem, which is image analysis using a computer vision system, was performed outside of the scope of this paper. Please see reference for link to the dataset for more information.

## REFERENCES


[1] Broughton, W.J., Hernández, G., Blair, M. et al. Beans (Phaseolus spp.) – model food legumes. Plant and Soil 252, 55–128 (2003). https://doi.org/10.1023/A:1024146710611

[2] P B Geil & J W Anderson (1994) Nutrition and health implications of dry beans: a review., Journal of the American College of Nutrition, 13:6, 549-558, DOI: 10.1080/07315724.1994.10718446

[3] P.H. Graham, P. Ranalli, Common bean (Phaseolus vulgaris L.), Field Crops Research, Volume 53, Issues 1–3, 1997, Pages 131-146, ISSN 0378-4290, https://doi.org/10.1016/S0378-4290(97)00112-3.

[4] "Green bean production in 2016, Crops/Regions/World list/Production Quantity (pick lists)". UN Food and Agriculture Organization, Corporate Statistical Database (FAOSTAT). 2017. Retrieved 27 September 2018.

[5] Ramos, P.J.; Prieto, F.A.; Montoya, E.C.; Oliveros, C.E. Automatic fruit count on coffee branches using computer vision. Comput. Electron. Agric. 2017, 137, 9–22.

[6] Moshou, D.; Bravo, C.; West, J.; Wahlen, S.; McCartney, A.; Ramon, H. Automatic detection of "yellow rust" in wheat using reflectance measurements and neural networks. Comput. Electron. Agric. 2004, 44, 173–188.

[7] Grinblat, G.L.; Uzal, L.C.; Larese, M.G.; Granitto, P.M. Deep learning for plant identification using vein morphological patterns. Comput. Electron. Agric. 2016, 127, 418–424.

[8] P.H. Graham, P. Ranalli, Common bean (Phaseolus vulgaris L.), Field Crops Research, Volume 53, Issues 1–3, 1997, Pages 131-146, ISSN 0378-4290, https://doi.org/10.1016/S0378-4290(97)00112-3.

[9] MacKay, Information Theory, Inference and Learning Algorithms, Cambridge Univ Press, 2003.

[10] A. Smola, Introduction to Machine Learning, Cambridge University Press, 2008.

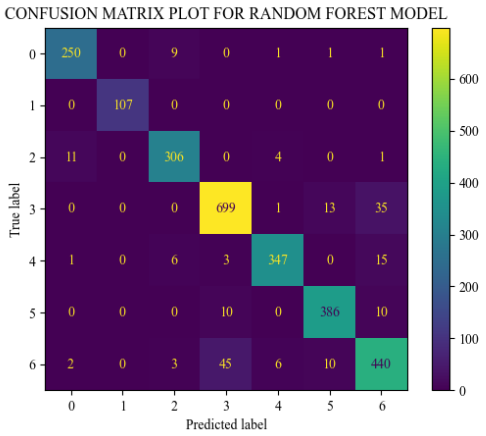
[11] Kevin Murphy, "Machine Learning : A Probabilistic Perspective", MIT Press, 2012.

[12] Ian Goodfellow, Yoshua Bengio and Aaron Courville, "Deep Learning, 2016

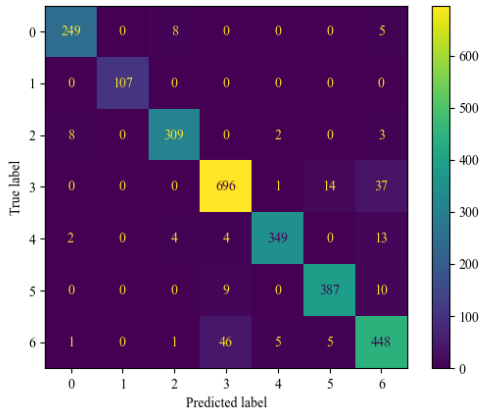
[13] T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning", Springer 2011.

APPENDIX

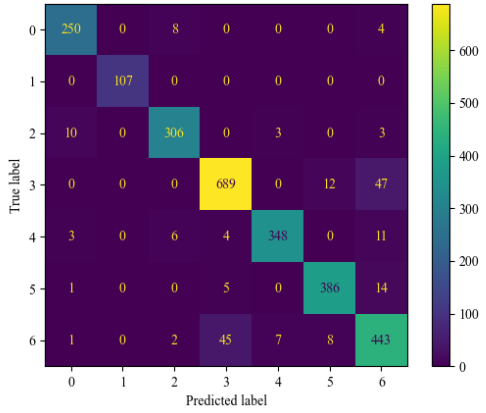
## A. *CONFUSION MATRIX PLOT FOR RANDOM FOREST MODEL*



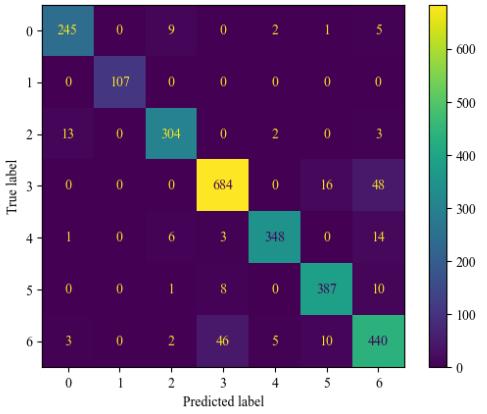

## B. *CONFUSION MATRIX PLOT FOR SUPPORT VECTOR MACHINE MODEL*



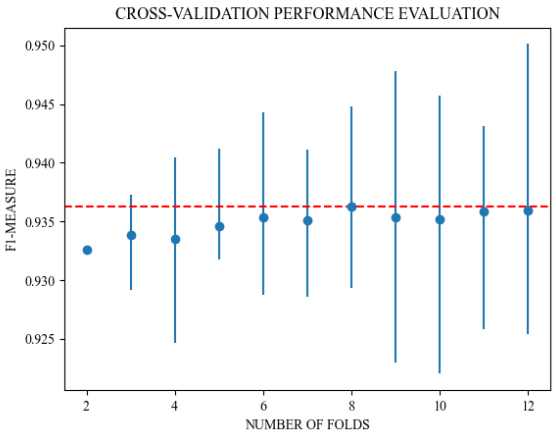

## C. *CONFUSION MATRIX PLOT FOR LOGISTIC REGRESSION MODEL*




## D. *CONFUSION MATRIX PLOT FOR K-NEAREST NEIGHBOURS MODEL*

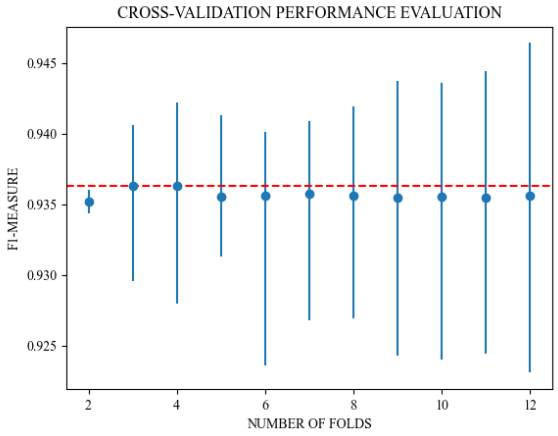



## E. *KFOLD CROSS-VALIDATION PLOT FOR RANDOM FOREST MODEL*

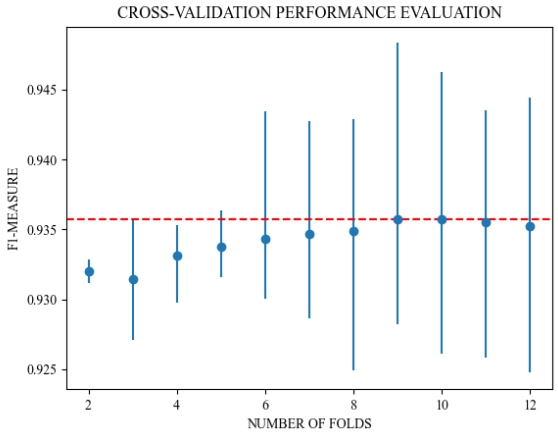



## F. *KFOLD CROSS-VALIDATION PLOT FOR SUPPORT VECTOR MODEL*

### G. *KFOLD CROSS-VALIDATION PLOT FOR LOGISTIC REGRESSION MODEL*




### H. *KFOLD CROSS-VALIDATION PLOT FOR KNN MODEL*




### I. *LINK TO THE DATASET*

The link to the original dataset and metadata can be found here:
http://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset

### J. *LINK TO THE FULL SOURCE CODE*

The visualization, preprocessing, feature extraction and machine learning codes were all written in Python language on a HP Pavilion Notebook 15 machine with Intel® Core™ i3-6100U processor (2.3 GHz, 3 MB cache, 2 cores), 8 GB DDR4-2133 SDRAM (1 x 8 GB) and Intel® HD Integrated Graphics 520. The link to the full source code can be found here:
https://drive.google.com/drive/folders/1Q_aTvhaXitMD1W5tjEQoDXmvAIOiSkju?usp=sharing