# REPORT

# ON

# DRY-BEANS CLASSIFICATION

## IBM-312: Data Mining

Submitted by

**PINAKI BASU (19115087)**

**PARITOSH SANADHAYA (19115083)**

**GURSIMRAN (19115057)**


**DEPARTMENT OF MANAGEMENT STUDIES**

**INDIAN INSTITUTE OF TECHNOLOGY ROORKEE**

**ROORKEE-247667, UK (INDIA)**

**Spring, 2021-22**

IIT ROORKEE

## MOTIVATION

The exponential growth of Machine Learning as a domain has had a profound impact on all fields of society. Data analysis in the pre machine-learning era relied on conventional and fundamental methods which were highly specialized domains, and often only restricted to subject-experts. Machine Learning, with its uncanny ability to figure out patterns and 'learn' intricate nuances of a dataset, has made it possible for everyone to observe, infer and perform meaningful operations on the dataset. From stock-prices forecasting to housing-price regression, the wings of Machine Learning have spread across various interdisciplinary fields. One such problem statement we will be analyzing in our project is Dry-Beans Classification , where given an extensive dataset of different types of Dry-Beans, our model would learn from the data given and then try to identify the class of dry-bean accurately from the given features.

## DATASET

The dataset for Dry-Beans Classification problem was taken from UCI Machine Learning Repository (*https://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset*). The dataset contains 13611 samples of dry-beans, each classified into one of 7 labels. Each dry-bean is described by a set of 16 features, ranging from dimension metrics like area and axes length to various shape metrics. 80% of the data has been taken for training, while 20% is kept for testing and evaluating the model.
A snapshot of our dataset containing the first 10 elements is attached below: -

| | Area | Perimeter | MajorAxisLength | MinorAxisLength | AspectRation | Eccentricity | ... | Compactness | ShapeFactor1 | ShapeFactor2 | ShapeFactor3 | ShapeFactor4 | Class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 28395 | 610.29 | 208.18 | 173.89 | 1.20 | 0.55 | ... | 0.91 | 7.33e-03 | 3.15e-03 | 0.83 | 1.00 | SEKER |
| 1 | 28734 | 638.02 | 200.52 | 182.73 | 1.10 | 0.41 | ... | 0.95 | 6.98e-03 | 3.56e-03 | 0.91 | 1.00 | SEKER |
| 2 | 29380 | 624.11 | 212.83 | 175.93 | 1.21 | 0.56 | ... | 0.91 | 7.24e-03 | 3.05e-03 | 0.83 | 1.00 | SEKER |
| 3 | 30008 | 645.88 | 210.56 | 182.52 | 1.15 | 0.50 | ... | 0.93 | 7.02e-03 | 3.21e-03 | 0.86 | 0.99 | SEKER |
| 4 | 30140 | 620.13 | 201.85 | 190.28 | 1.06 | 0.33 | ... | 0.97 | 6.70e-03 | 3.66e-03 | 0.94 | 1.00 | SEKER |
| 5 | 30279 | 634.93 | 212.56 | 181.51 | 1.17 | 0.52 | ... | 0.92 | 7.02e-03 | 3.15e-03 | 0.85 | 1.00 | SEKER |
| 6 | 30477 | 670.03 | 211.05 | 184.04 | 1.15 | 0.49 | ... | 0.93 | 6.92e-03 | 3.24e-03 | 0.87 | 1.00 | SEKER |
| 7 | 30519 | 629.73 | 213.00 | 182.74 | 1.17 | 0.51 | ... | 0.93 | 6.98e-03 | 3.16e-03 | 0.86 | 1.00 | SEKER |
| 8 | 30685 | 635.68 | 213.53 | 183.16 | 1.17 | 0.51 | ... | 0.93 | 6.96e-03 | 3.15e-03 | 0.86 | 1.00 | SEKER |
| 9 | 30834 | 631.93 | 217.23 | 180.90 | 1.20 | 0.55 | ... | 0.91 | 7.05e-03 | 3.01e-03 | 0.83 | 1.00 | SEKER |

10 rows × 17 columns

# FEATURE ENGINEERING – EXPLORATORY DATA ANALYSIS

Before diving straight into various Machine-Learning models, it is of utmost importance that we understand the nature and the properties of data we are handling. Only then can we evaluate and judge which ML models would perform better for our given use case.

Taking this into regard, we have performed 4 steps of data visualization and pre-processing :-

I. **Bar Chart –** This provides a frequency-based visual and intuitive view regarding the number of examples in each label. If a particular label has too few examples, it becomes difficult for the algorithm to properly extract the features specific to that label, which makes training difficult and thus requires adequate steps to overcome that. Our dataset contains adequate examples of every label so we can proceed with further analysis without any modification here.

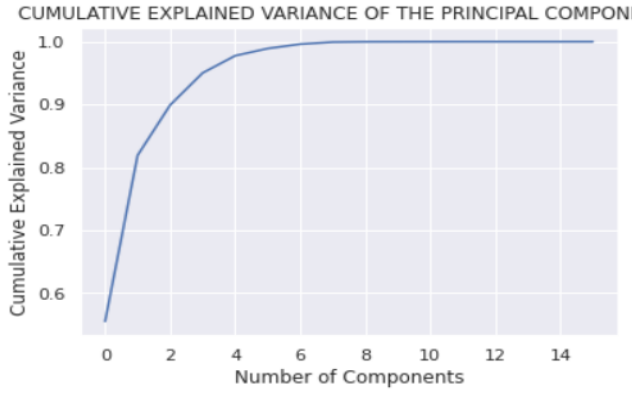
II. **Correlation Matrix –** This matrix quantitatively describes how independent two features (pairwise) are in the given dataset. In case of higher dependency, some features can be compressed without much loss of generality, thus making the algorithm computationally efficient. Some cells in our correlation Matrix have a relatively high value, which signals that feature selection methods like PCA can be applied to the dataset.

III. **Scaling –** As evident from the values in each column of the dataset, the average numerical value of each column differs greatly. This may result in some columns inherently receiving greater weightage as compared to others, which is undesired. Therefore we scale the dataset with the help of Standard scaler, which converts it into a normal distribution with mean 0 and variance 1.

IV. **Principal Component Analysis (PCA) –** As the name suggests, PCA involves taking some primary or ‘principal’ features, which are a combination of the given feature space, so that some correlated features or non-contributing features get compressed and thus the cardinality of the feature space is reduced. The aim of PCA is to minimize the number of features used to describe a dataset while trying to retain the variance of the given data as much as possible. It is essentially a trade-off between cutting-off the number of features for computational efficiency vs retaining the originality of the model. PCA achieves this with the help of similarity transformation. The largest eigenvalues and their corresponding eigenvectors are used to transform the matrix from a larger dimensionality to a smaller one.

The results obtained for explained variance vs number of components is attached below: -

```
Total variance covered taking  1 Most dominant eigenvectors :  0.5546643861267669
Total variance covered taking  2 Most dominant eigenvectors :  0.8189741179133915
Total variance covered taking  3 Most dominant eigenvectors :  0.899039760198745
Total variance covered taking  4 Most dominant eigenvectors :  0.950180563039863
Total variance covered taking  5 Most dominant eigenvectors :  0.9775734920568954
Total variance covered taking  6 Most dominant eigenvectors :  0.9890711014604846
Total variance covered taking  7 Most dominant eigenvectors :  0.9960476088161605
Total variance covered taking  8 Most dominant eigenvectors :  0.9992984338041777
Total variance covered taking  9 Most dominant eigenvectors :  0.9998147001676273
Total variance covered taking  10 Most dominant eigenvectors :  0.9999055682893362
Total variance covered taking  11 Most dominant eigenvectors :  0.9999714550823996
Total variance covered taking  12 Most dominant eigenvectors :  0.9999898290123473
Total variance covered taking  13 Most dominant eigenvectors :  0.9999991286766473
Total variance covered taking  14 Most dominant eigenvectors :  0.9999997543184199
Total variance covered taking  15 Most dominant eigenvectors :  0.9999998884505051
Total variance covered taking  16 Most dominant eigenvectors :  0.9999999999999999
```

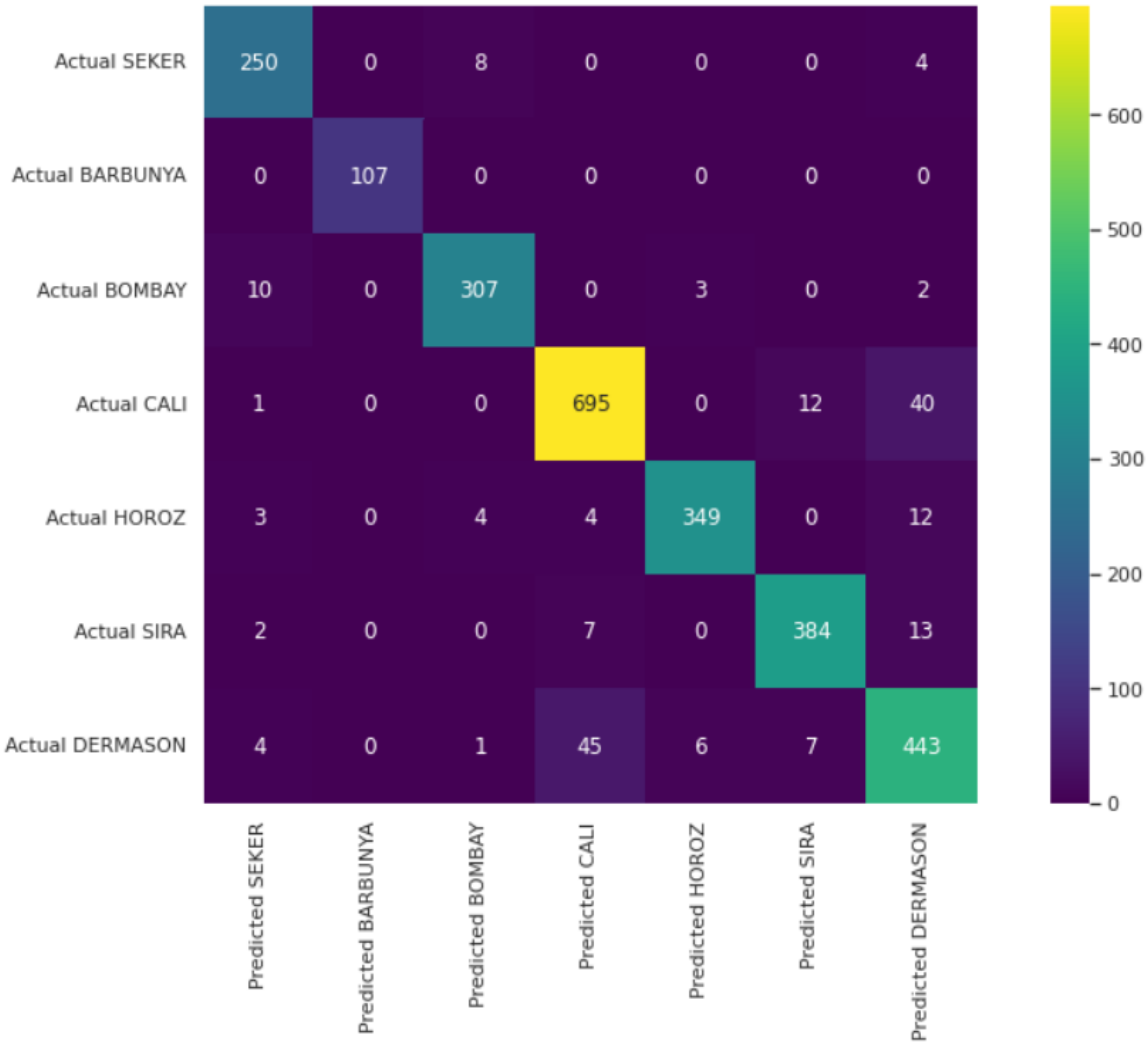



## MODEL TRAINING

### I. Support Vector Classification

Support Vector Classification (SVC) is a supervised machine-learning algorithm which aims to find the hyperplane in an N-dimensional data space that distinctly classifies the points. The hyperplane is chosen such that the distance from it to the nearest data point on each side of it is maximized. SVC algorithm is designed to ignore outliers, thus making it robust to the presence of any. SVC is memory efficient as it uses a subset of training points in the decision function called support vectors.

The correlation matrix formed after applying SVC to the dataset is as shown below: -

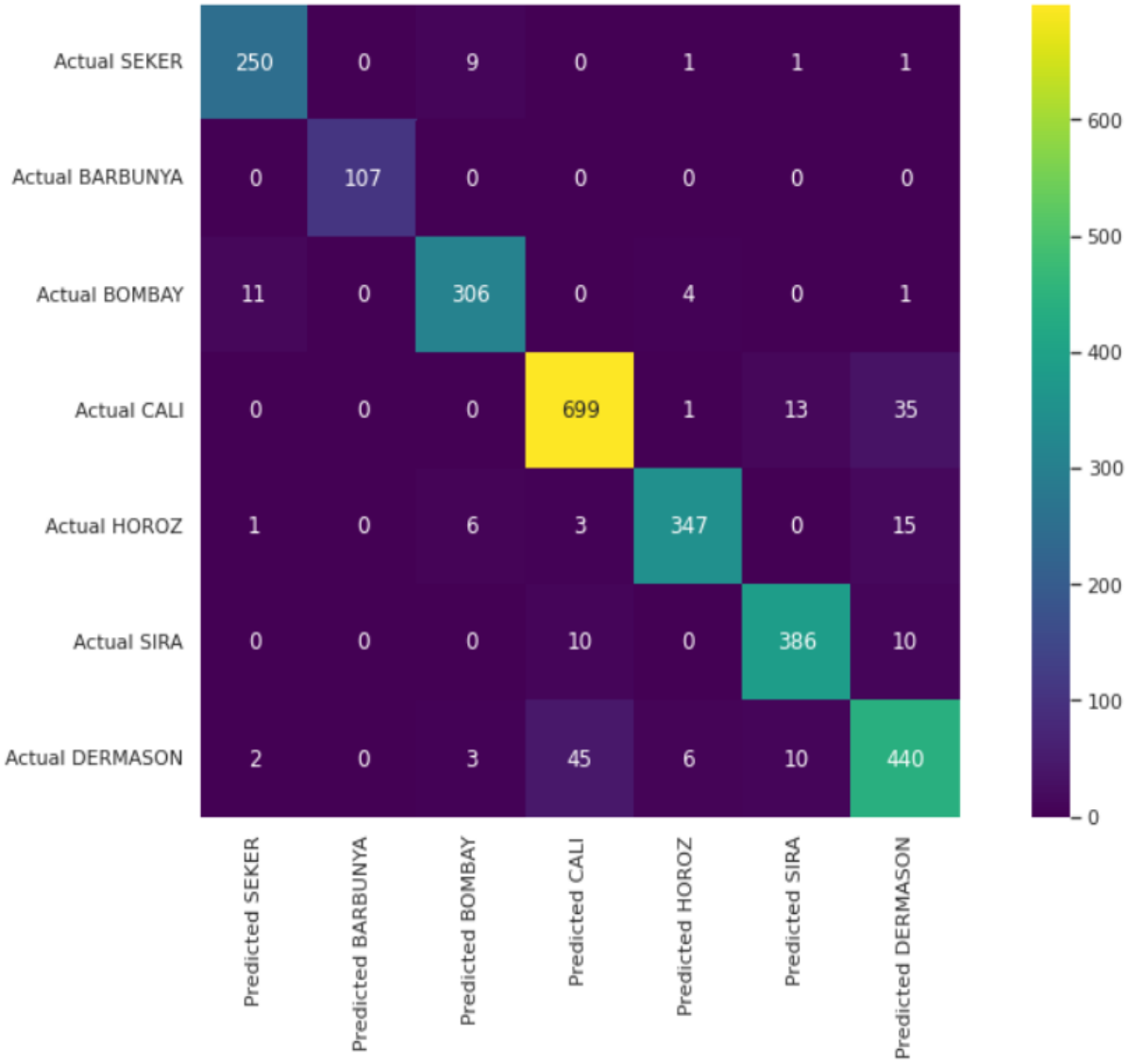



Result : The accuracy of the model comes out to be 0.931.

## II. Random Forest Classification

Random Forest Classifier is an ensemble learning based Machine-Learning algorithm, which outputs the class selected by most decision-trees. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. Random Forests apply the technique of Bagging (Bootstrap-Aggregating), method of randomly creating samples of data out of a population with replacement to estimate a population parameter. It takes several weak models, aggregating the predictions to select the best prediction. The weak models specialize in distinct sections of the feature space, which enables bagging leverage predictions to come from every model to reach the

utmost purpose. Bagging deals with higher dimensional data efficiently and significantly reduces overfitting, thus improving the model performance.
The correlation matrix formed after applying Random-Forest Classifier to the dataset is as shown below: -




Result : The accuracy of the model comes out to be 0.931.

**a. Model Performance Enhancement by SMOTE**

SMOTE, or Synthetic Minority oversampling Technique addresses the problem of imbalanced datasets. Imbalanced datasets result in poor performance in minority classes due to comparatively less data available. SMOTE augments this data by synthesizing new examples from existing ones. It first selects a minority class

instance at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b.
After applying SMOTE, the accuracy of the SVC Model improves to 0.939 while the accuracy of the Random Forest Model improves to 0.948.

**b. Model Performance Enhancement by PCA**

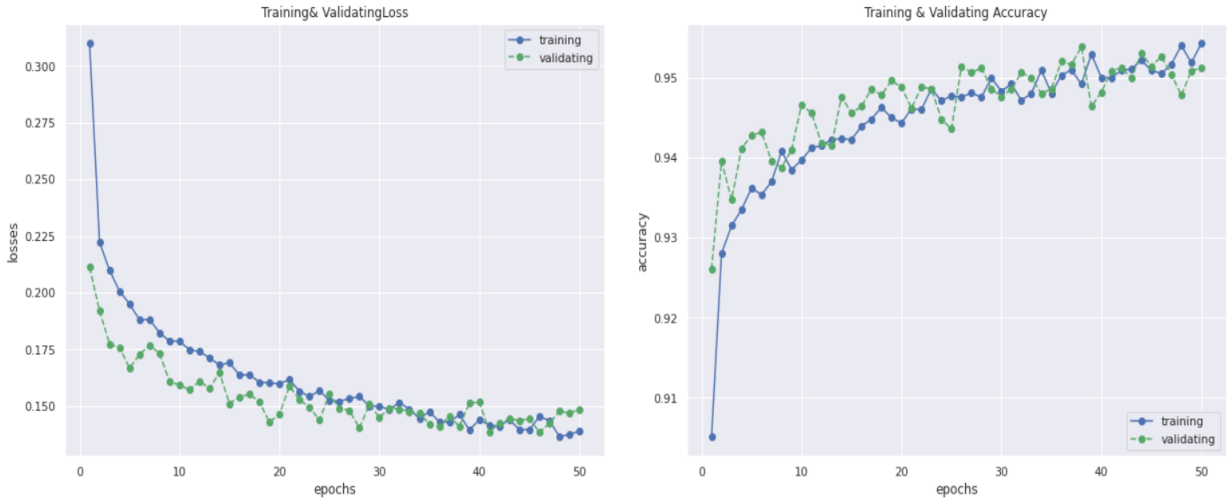
Given the high correlation among some features in the dataset, feature extraction mechanisms like PCA can be used to simplify the feature space and boost the performance of the model. PCA removes correlated features and reduces overfitting of the dataset thereby improving the accuracy of the model. In this model, a variance cutoff of 99% has been taken, which results in taking 7 features, lesser from the earlier taken 16.
After applying PCA, the accuracy of the SVC Model improves to 0.940 while the accuracy of the Random Forest Model improves to 0.953.

## III. Artificial Neural Network

Our third model uses a completely new approach, that of a neural network. An Artificial Neural Network (ANN) is a collection of connected nodes called neurons. The neurons process the information from the previous layer with the help of a weight matrix and pass it to the next layer via an activation function in the forward propagation pass. The network learns its weights for each layer through gradient descent in the back propagation pass. Our network uses 5 layers, each followed by a dropout layer with probability of 20% as a regularization metric to reduce overfitting, followed by a softmax layer which maps the information into one of 7 possible labels. Further, early-stopping is also used to reduce overfitting during training. The model learns via Adam optimization, an enhancement of Gradient Descent, in batch-sizes of 32. The model was trained for 50 epochs, with the aim of minimizing sparse_categorical_crossentropy loss function, with metric being accuracy. The code and graph for training-validation loss is attached alongwith: -

```
[222] model = Sequential()
      model.add(Dense(512, activation='tanh', input_shape=(7,)))
      model.add(Dropout(0.2))
      model.add(Dense(256, activation='tanh'))
      model.add(Dropout(0.2))
      model.add(Dense(128, activation='tanh'))
      model.add(Dropout(0.2))
      model.add(Dense(64, activation='tanh'))
      model.add(Dropout(0.2))
      model.add(Dense(32, activation='tanh'))
      model.add(Dropout(0.2))
      model.add(Dense(7, activation='softmax'))
```

```
[223] model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
      early_stopp = EarlyStopping(patience=6, restore_best_weights=True, monitor='val_loss')
      history = model.fit(X_train, y_train, epochs=50, validation_data=(X_val, y_val), batch_size=32)
```




Result : The accuracy of the model comes out to be 0.951.

## CONCLUSION AND FUTURE WORKS

Despite excellent success in various performance metrics of all the 3 models made, the performance can still be further enhanced, and will be our domain of future work in this project. One such enhancement is to use layers of Convolutional Neural Networks (CNNs). CNNs are powerful Neural Network Architectures specifically designed for image processing, with powerful feature extracting mechanisms. Incorporating CNNs may result in an even higher accuracy for this dataset.