Dataset Introduction/Description:

Images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera.
A total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains. Seven different types of dry beans were used in this research, taking into account the features such as form, shape, type, and structure by the market situation. A computer vision system was developed to distinguish seven different registered varieties of dry beans with similar features in order to obtain uniform seed classification. For the classification model, images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. Bean images obtained by computer vision system were subjected to segmentation and feature extraction stages, and a total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains.

Attribute Information
1.) Area (A): The area of a bean zone and the number of pixels within its boundaries.
2.) Perimeter (P): Bean circumference is defined as the length of its border.
3.) Major axis length (L): The distance between the ends of the longest line that can be drawn from a bean.
4.) Minor axis length (l): The longest line that can be drawn from the bean while standing perpendicular to the main axis.
5.) Aspect ratio (K): Defines the relationship between L and l.
6.) Eccentricity (Ec): Eccentricity of the ellipse having the same moments as the region.
7.) Convex area (C): Number of pixels in the smallest convex polygon that can contain the area of a bean seed.
8.) Equivalent diameter (Ed): The diameter of a circle having the same area as a bean seed area.
9.) Extent (Ex): The ratio of the pixels in the bounding box to the bean area.
10.)Solidity (S): Also known as convexity. The ratio of the pixels in the convex shell to those found in beans.
11.) Roundness (R): Calculated with the following formula: (4piA)/(P^2)
12.) Compactness (CO): Measures the roundness of an object: Ed/L
13.) ShapeFactor1 (SF1)
14.) ShapeFactor2 (SF2)
15.) ShapeFactor3 (SF3)
16.) ShapeFactor4 (SF4)
17.) Class (Seker, Barbunya, Bombay, Cali, Dermosan, Horoz and Sira)

Limitations of The Dataset: Prediction will be slow with multiple labels to classify, as load on computer processor will be high.

K NEAREST NEIGHBOR MODEL: The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning). Also known as lazy algorithm. It falls under classification algorithm (supervised learning). Other alternative classification models are Logistic regression, decision trees, Support vector machine (SVM), random forest, and Naïve Bayes.

Factors that affect the performance of a KNN model:

1. Calculating the distance between the nearest neighbors
   There is the Minkowski (main formular for knn distance): with the formular

| (x1 - y1)^p | ^p

When p = 2 we have the Euclidean distance, When p = 1 we have Manhattan. For our project we are using Euclidean distance since our dataset has small dimension. But it also as the issue of the result being influenced by outliers.

2. The type of tree algorithm used in implementing KNN
3. The no_of_neighbour chosen.

Merit of the KNN Model:

1. simply "remember" all of its training data (this possibly transformed into a fast indexing structure such as a Ball Tree or KD Tree).
2. It is so easy to implement. As only the distance (no of neighbors) and tree algorithm used is important. Thus it is often called the lazy algorithm.

Limitations of KNN Algorithm:

1. It becomes computationally expensive when we have thousands of data points
2. It is prone to outliers
3. Takes lot of storage, as it need to keep record of neighbor

HOW WE IMPROVE THE PERFORMANCE OF KNN:

1. To reduce the cost of computational expenses, we introduce the concept of trees. There is KD-tree, Ball-trees. Ball trees shine when the total attributes in a data set is in the 100 or thousands, and also for sparse datasets. In our case, the total attributes in out dry-bean is 17. So we are going with KD_tree algorithm.
2. Using Euclidean distance (where P= 2)
3. Application of preprocessing mechanism like StandardScaler() function which fit, transformed, and standardize the features.
4. Implementation of GridSearchCV function for cross validation

Real-World Application of the KNN Model: Knn can be used in real word to build a recommendation system, for example, movie recommendation on Movie websites. It can also be use to build a customer segmentation system. Knowing customers and categorizing them base on those unique attributes

METHODOLOGY AND TECHNIQUES EMPLOYED

1. Deciding on Analysis tools
   Did research on classification algorithms like Logistic regression, decision trees, Support vector machine (SVM), random forest, K-nearest neighbor (KNN) and Naïve Bayes. I ended up choosing two out of the six models I studied for my analysis. KNN as my primary and SVM as secondary.

2. Choice of Python Library and Coding Environment.
   Using the website scikit-learn.org, I studied the various functionalities that skLearn provide for data analysis. Choose scikit-learn as it allows us to define machine learning algorithms and compare them to one another, as well as offers tools to preprocess data.
   Jupyter notebook was my chosen IDE, launched from inside Anaconda navigator. This way I am able to run create and manage multiple sci-kit learn environment without running into broken issue, or library version not supported errors.

3. Import various SkLearn libraries that I would be using to work on my dry-Bean datasets. So that I can get access to their inbuilt classes and functions that my choice of model provides. Few of them are:

```
#importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing
from sklearn.metrics import classification_report,accuracy_score
from sklearn.preprocessing import StandardScaler, QuantileTransformer,
from sklearn.metrics import ConfusionMatrixDisplay
```

4. Data Exploration
   Load my dataset into Jupiter notebook. And started running and outputting python codes that give me a detail and better understanding of dataset. Some of the few command I run and their uses is listed below.

```
df.head(10)                 # print the first 10 rows of my dataset
df.shape                    # total rows(entries) and columns(attributes)
df.describe()               # output mean, std, counts of each attributes
df['Class'].unique()        # output all 7 categories of dry-beans
df['Class'].value_counts()  # no of each bean type in the class attribute
```

5. Data visualization
   Used Seabon.countplot() function to Show the counts of observations in each categorical bin of dry-beans using bars. A count plot can be thought of as a histogram.

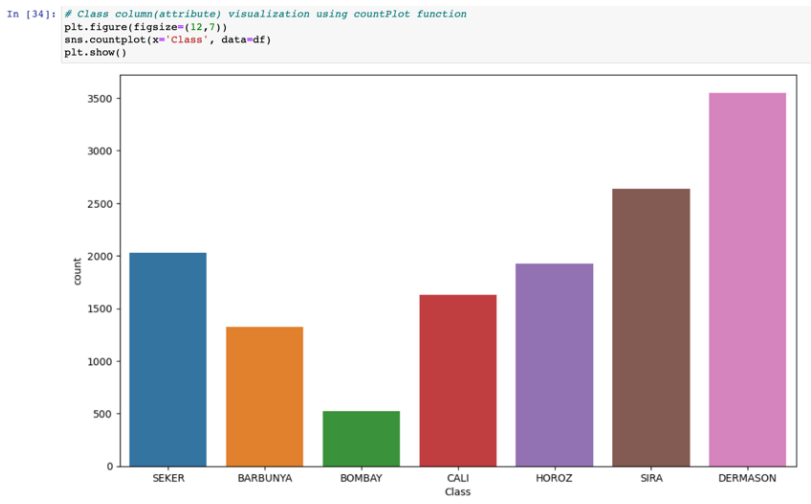

Figure 1: Observation of bean types in class attribute using the countplot() function

Correlation matrix using pearson method, with heatmap for result visualization. We considered computing the standard correlation coefficient (using pearson's method) between every pair of attributes, and displayed on a table. . The correlation was visualized using a heatmap.

```
# Syntax – df (DataFram) represent our dataset
>>> correlationMatrix = df.corr( method = 'pearson' )
>>> seaborn.heatmap(correlation)
```

The correlation coefficient which we got ranged from -1 to 1. When it is close to 1 (vmax), it means that there is a strong positive correlation. When the coefficient is close to -1 (vmin), it means there is a strong negative correlation. While coefficients close to 0 (center) rep means that there is no linear correlation. The correlation table that contains attributes and heatmap to visualize the value of the matrix is provided visible in our implementation code.

Figure 2: Correlation matrix visualization using Heatmap from Seaborn library

Since we are working with a large dataset, correlation matrix and heatMap prove to be a powerful tool in helping us identify and visualize patterns in the given data and thus reach a good summary.

Visualised all the numeric attributes (16 of them) using an histogram.

6. Split Data into Train and Test Subsets

7. Time for Pre-processing of Data.

   Label Encoding: "Class" column/attribute with our list of dry-beans is in string (in object form) so we need to convert the labels into a numeric form. LabelEncoder library in skLearn was used.

   Data standardization and Normalization: The standardScaler class in sklearn was used to scale the data. This removes outliers, fit, and transform the data so as to improve our performance when our data finally gets to the chosen model.
8. Model training using KNN and SVM
9. Classification Reporting and Accuracy scoring

10. Metrix visualization using ConfusionMetrix

PRESENTATION OF RESULTS

K-NEAREST NEIGHBOR REPORT AND ACCURACY SCORE

Accuracy score is 0.92325 which is 92 %

Table 1: KNN classification Report

Accuracy: 0.92325

	precision	recall	f1-score	support
BARBUNYA	0.96	0.90	0.93	261
BOMBAY	1.00	1.00	1.00	117
CALI	0.92	0.96	0.94	317
DERMASON	0.89	0.91	0.90	671
HOROZ	0.97	0.95	0.96	408
SEKER	0.97	0.95	0.96	413
SIRA	0.86	0.87	0.86	536
accuracy			0.92	2723
macro avg	0.94	0.93	0.94	2723
weighted avg	0.92	0.92	0.92	2723

KNN VISUALIZATION USING CONFUSION-MATIX DISPLAY

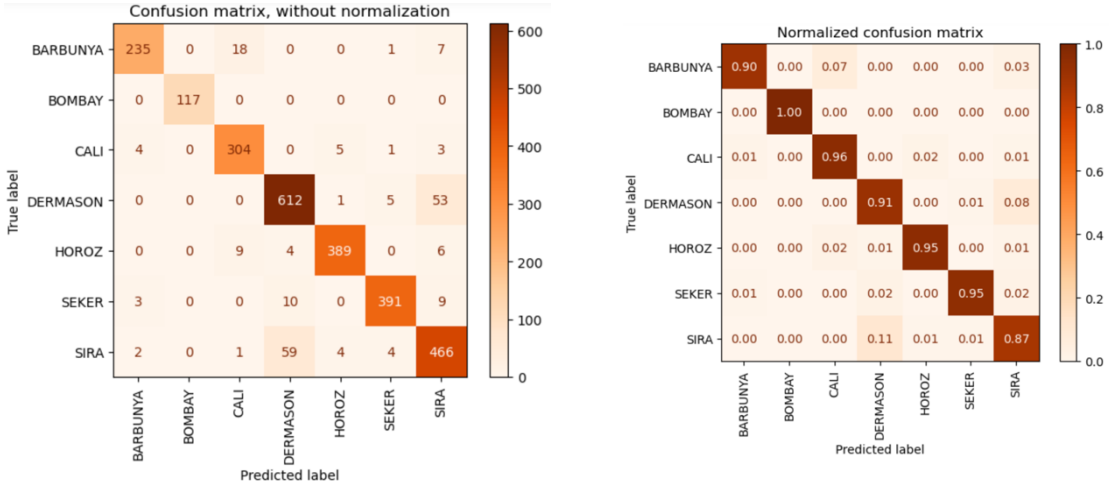

Figure 3: KNN confusion matrix and KNN Confusion matrix with normalization

DISCUSSIONS AND FINDINGS: Using KNN, Bombay got the best performance, while Sira got the worst.

COMPARISON OF KNN (primary model) and SVM RESULT

1. Accuracy in KNN is 92% while in SVM it is 93%
2. In both KNN and SVM Bombay got the best performance, while Sira got the worst.

EXTRA TASK: Support Vector Machine (SVM) as my secondary model for the dry-beans problem:

SVMs decision function depends on some subset of the training data, called the support vectors.

The advantages of support vector machines are:
1.Effective in high dimensional spaces.
2. Still effective in cases where number of dimensions is greater than the number of samples.
3. Uses subset of training points in the decision function (support vectors), which is memory efficient.
4. Versatile: different Kernel functions can be specified for the decision function.

The disadvantages of support vector machines include:
1. If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
2. SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

PRESENTATION OF RESULTS FOR SVM
SUPPORT VECTOR MACHINE(SVM) REPORT AND ACCURACY SCORE

Accuracy score is 0.93390 which is 93 %

Table 2: SVM classification Report

Accuracy: 0.93390

	precision	recall	f1-score	support
BARBUNYA	0.93	0.92	0.93	261
BOMBAY	1.00	1.00	1.00	117
CALI	0.94	0.95	0.94	317
DERMASON	0.91	0.93	0.92	671
HOROZ	0.98	0.96	0.97	408
SEKER	0.97	0.95	0.96	413
SIRA	0.88	0.90	0.89	536
accuracy			0.93	2723
macro avg	0.95	0.94	0.94	2723
weighted avg	0.93	0.93	0.93	2723

SVM VISUALIZATION USING CONFUSION-MATIX DISPLAY

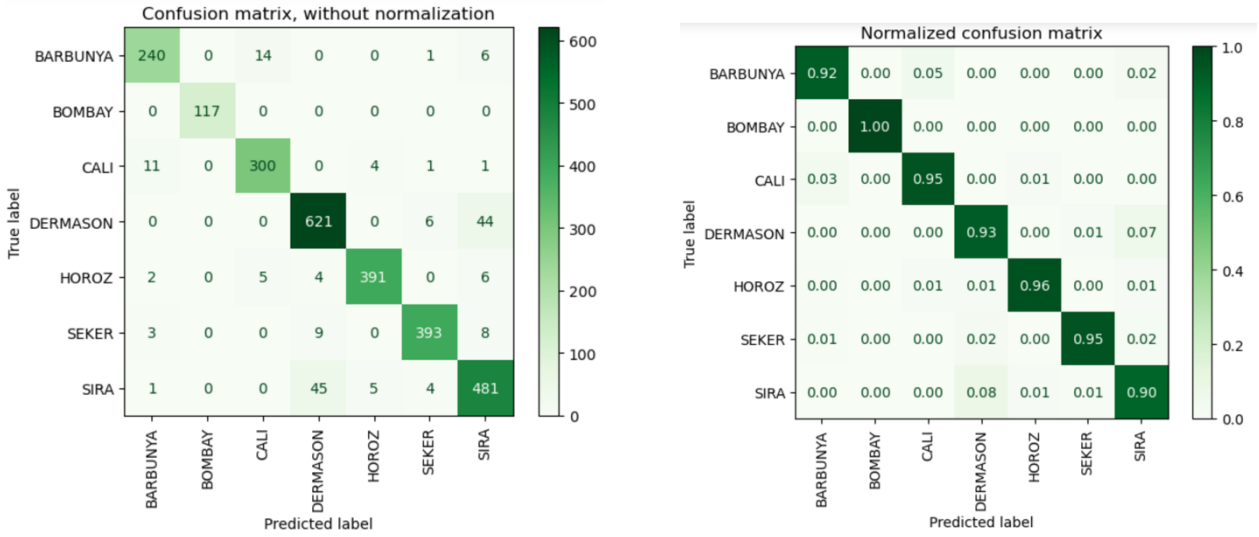

Figure 3: SVM confusion matrix and SVM Confusion matrix with normalization

REFERENCES AND RESOURCES

Dataset and problem description

• Problem statement – To develop a system for automatic detection of 7 types of dry bean seeds based on data captured using a high-resolution camera.

• Dataset (Dry Bean dataset) – Download from https://archivebeta.ics.uci.edu/dataset/602/dry+bean+dataset

• Research publication (on dataset and previous work) –

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, https://doi.org/10.1016/j.compag.2020.105507