IE 7300 Statistical Learning for Engineering Fall 2022

Group 14

Project Report

Dry Beans Classification

Group Members

Ajay Athitya Ramanathan

Monil Rawka

Himani Thakker

Riddhi Gupta

Contribution: All members contributed equally

INTRODUCTION

The most frequently cultivated edible bean crop in the world, the dry bean, contains a large range of genetic diversity. There is no denying that crop production is impacted by seed quality. Seed categorization is essential for both marketing and production to provide the building blocks of sustainable agricultural systems. More than 13,000 dried bean samples from 7 distinct species were captured, and their geometry was established using computer vision methods. This output dataset produced by the computer vision algorithm will be used. The primary objective of this research is to provide a method for obtaining stable seed varieties using population-based agricultural production, which precludes the seeds from being confirmed to be of a single variety.

ABSTRACT

The most frequently cultivated edible bean crop in the world, the dry bean, contains a large range of genetic diversity. There is no denying that crop production is impacted by seed quality. Seed categorization is essential for both marketing and production to provide the building blocks of sustainable agricultural systems. More than 13,000 dried bean samples from 7 distinct species were captured, and their geometry was established using computer vision methods. This output dataset produced by the computer vision algorithm will be used. The primary objective of this research is to provide a method for obtaining stable seed varieties using population-based agricultural production, which precludes the seeds from being confirmed to be of a single variety.

A high-resolution camera was used to capture photos of 13,611 grains from 7 distinct registered dry bean varieties for the classification model. Following feature segmentation and extraction processes on bean pictures collected using a computer vision system (CVS), a total of 16 features—12 dimension-based and 4 shape-based—were extracted from the grains. Performance metrics for the classification models of Logistic Regression, Support Vector Machine (SVM), k-Nearest Neighbors (kNN), Naive Bayes, and Neural Networks (NN) were compared.

PROBLEM STATEMENT

For bean growers and the market, choosing the finest seed breed is one of the top priorities. The best grain variety must be separated from the population of mixed dry beans since distinct genotypes are grown all over the world. If this is not done, the net worth of these hybrid species of beans may decline dramatically.

To execute a multi-classification of dry bean species gathered through population farming on a single farm, the goal of this research is to design a supervised machine learning algorithm. When developing a function that will yield the desired result when presented with fresh, unlabeled data, supervised machine learning algorithms rely on labeled input data.

AIM

The aim of this project is to develop a supervised machine learning algorithm to perform a multi-classification of dry beans species harvested from population cultivation from a single farm. A supervised machine learning algorithm is one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data.

DATA DESCRIPTION

Dataset was collected from UCI Machine Learning Repository and here is the link attached for it.

Link: https://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset

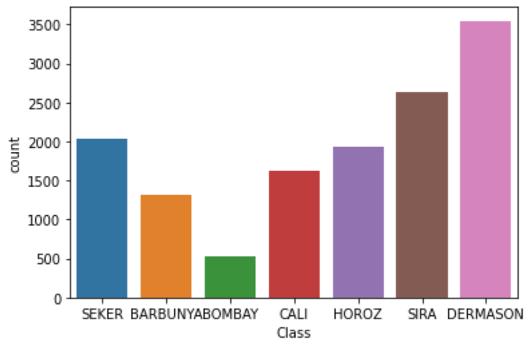
Number of Instances = 13,611

Number of Attributes = 17

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 (Y): Seker, Barbunya, Bombay, Cali, Dermosan, Horoz and Sira.

EXPLORATORY DATA ANALYSIS

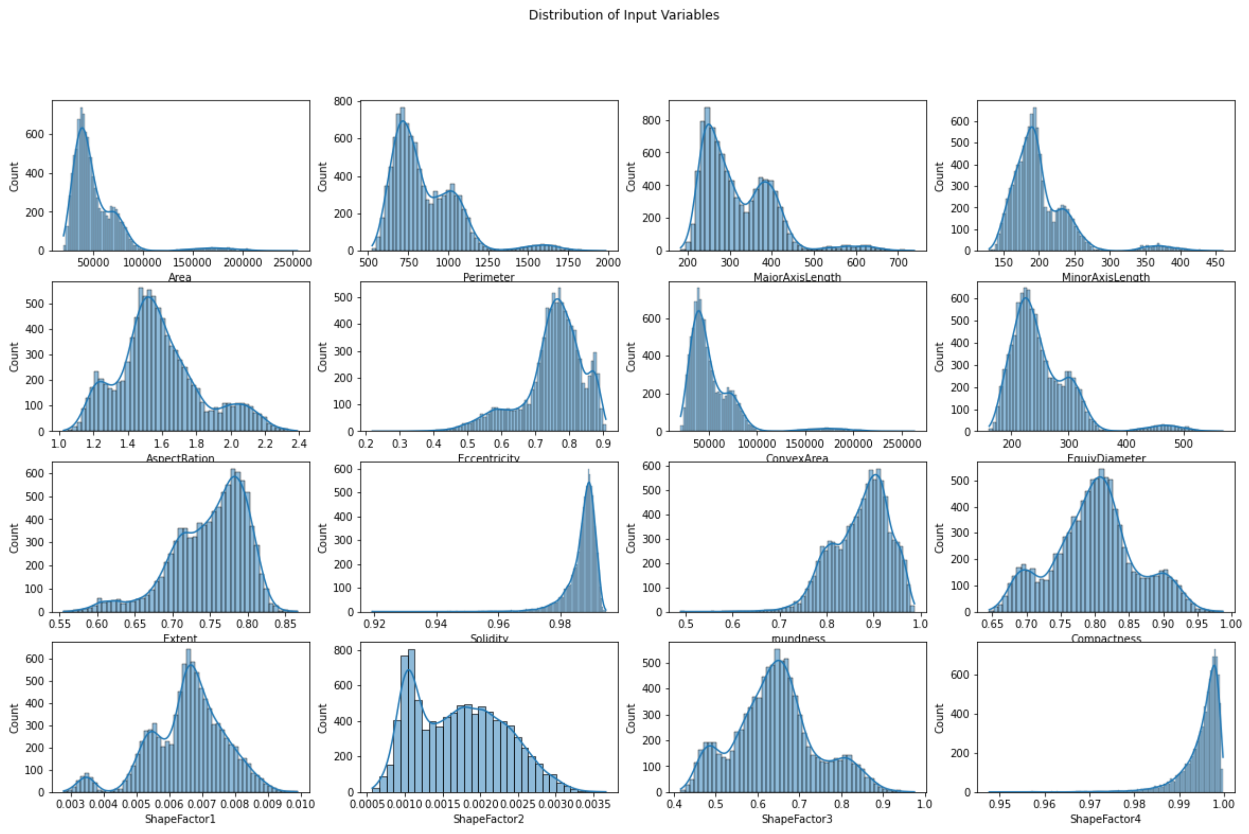
- Target variable distribution



DERMASON	3546
SIRA	2636
SEKER	2027
HOROZ	1928
CALI	1630
BARBUNYA	1322
BOMBAY	522

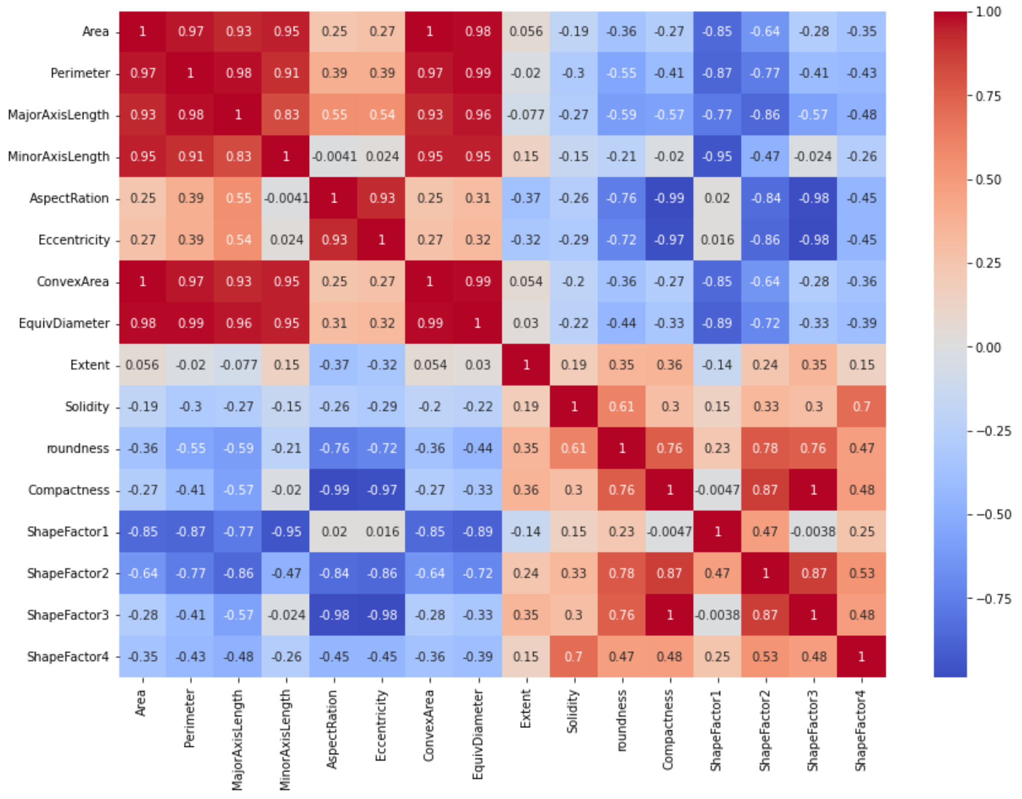
According to the graph above, there are different counts for each class. With 3546 beans, Dermason is the most prevalent class. Bombay has 522 beans, making it the least
common class. Two classes have a significant distinction that should be considered while developing a model.

- Input variable distribution

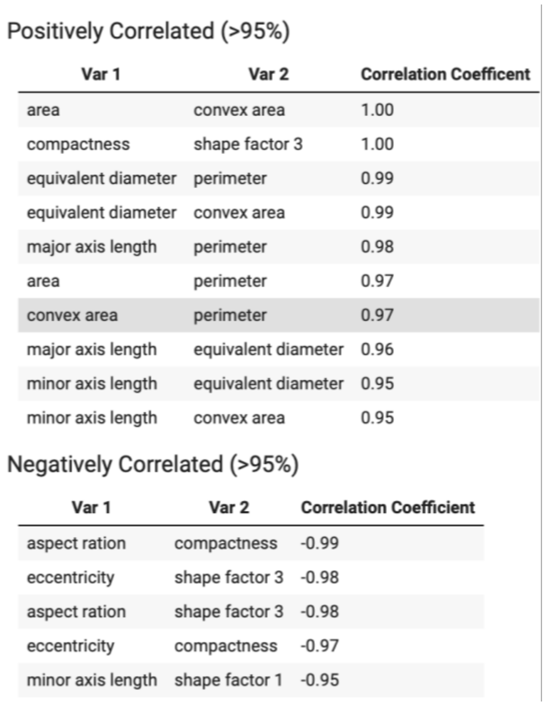


These points may imitate a special type of dry beans because of the skewness and outliers that characterize their distribution in large amounts. We go at some of the extremely effective columns' impact further.

- Input variable correlation



	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4
Area	1	0.97	0.93	0.95	0.25	0.27	1	0.98	0.056	-0.19	-0.36	-0.27	-0.85	-0.64	-0.28	-0.35
Perimeter	0.97	1	0.98	0.91	0.39	0.39	0.97	0.99	-0.02	-0.3	-0.55	-0.41	-0.87	-0.77	-0.41	-0.43
MajorAxisLength	0.93	0.98	1	0.83	0.55	0.54	0.93	0.96	-0.077	-0.27	-0.59	-0.57	-0.77	-0.86	-0.57	-0.48
MinorAxisLength	0.95	0.91	0.83	1	-0.0041	0.024	0.95	0.95	0.15	-0.15	-0.21	-0.02	-0.95	-0.47	-0.024	-0.26
AspectRation	0.25	0.39	0.55	-0.0041	1	0.93	0.25	0.31	-0.37	-0.26	-0.76	-0.99	0.02	-0.84	-0.98	-0.45
Eccentricity	0.27	0.39	0.54	0.024	0.93	1	0.27	0.32	-0.32	-0.29	-0.72	-0.97	0.016	-0.86	-0.98	-0.45
ConvexArea	1	0.97	0.93	0.95	0.25	0.27	1	0.99	0.054	-0.2	-0.36	-0.27	-0.85	-0.64	-0.28	-0.36
EquivDiameter	0.98	0.99	0.96	0.95	0.31	0.32	0.99	1	0.03	-0.22	-0.44	-0.33	-0.89	-0.72	-0.33	-0.39
Extent	0.056	-0.02	-0.077	0.15	-0.37	-0.32	0.054	0.03	1	0.19	0.35	0.36	-0.14	0.24	0.35	0.15
Solidity	-0.19	-0.3	-0.27	-0.15	-0.26	-0.29	-0.2	-0.22	0.19	1	0.61	0.3	0.15	0.33	0.3	0.7
roundness	-0.36	-0.55	-0.59	-0.21	-0.76	-0.72	-0.36	-0.44	0.35	0.61	1	0.76	0.23	0.78	0.76	0.47
Compactness	-0.27	-0.41	-0.57	-0.02	-0.99	-0.97	-0.27	-0.33	0.36	0.3	0.76	1	-0.0047	0.87	1	0.48
ShapeFactor1	-0.85	-0.87	-0.77	-0.95	0.02	0.016	-0.85	-0.89	-0.14	0.15	0.23	-0.0047	1	0.47	-0.0038	0.25
ShapeFactor2	-0.64	-0.77	-0.86	-0.47	-0.84	-0.86	-0.64	-0.72	0.24	0.33	0.78	0.87	0.47	1	0.87	0.53
ShapeFactor3	-0.28	-0.41	-0.57	-0.024	-0.98	-0.98	-0.28	-0.33	0.35	0.3	0.76	1	-0.0038	0.87	1	0.48
ShapeFactor4	-0.35	-0.43	-0.48	-0.26	-0.45	-0.45	-0.36	-0.39	0.15	0.7	0.47	0.48	0.25	0.53	0.48	1



Positively Correlated (>95%)

Var 1	Var 2	Correlation Coefficent
area	convex area	1.00
compactness	shape factor 3	1.00
equivalent diameter	perimeter	0.99
equivalent diameter	convex area	0.99
major axis length	perimeter	0.98
area	perimeter	0.97
convex area	perimeter	0.97
major axis length	equivalent diameter	0.96
minor axis length	equivalent diameter	0.95
minor axis length	convex area	0.95

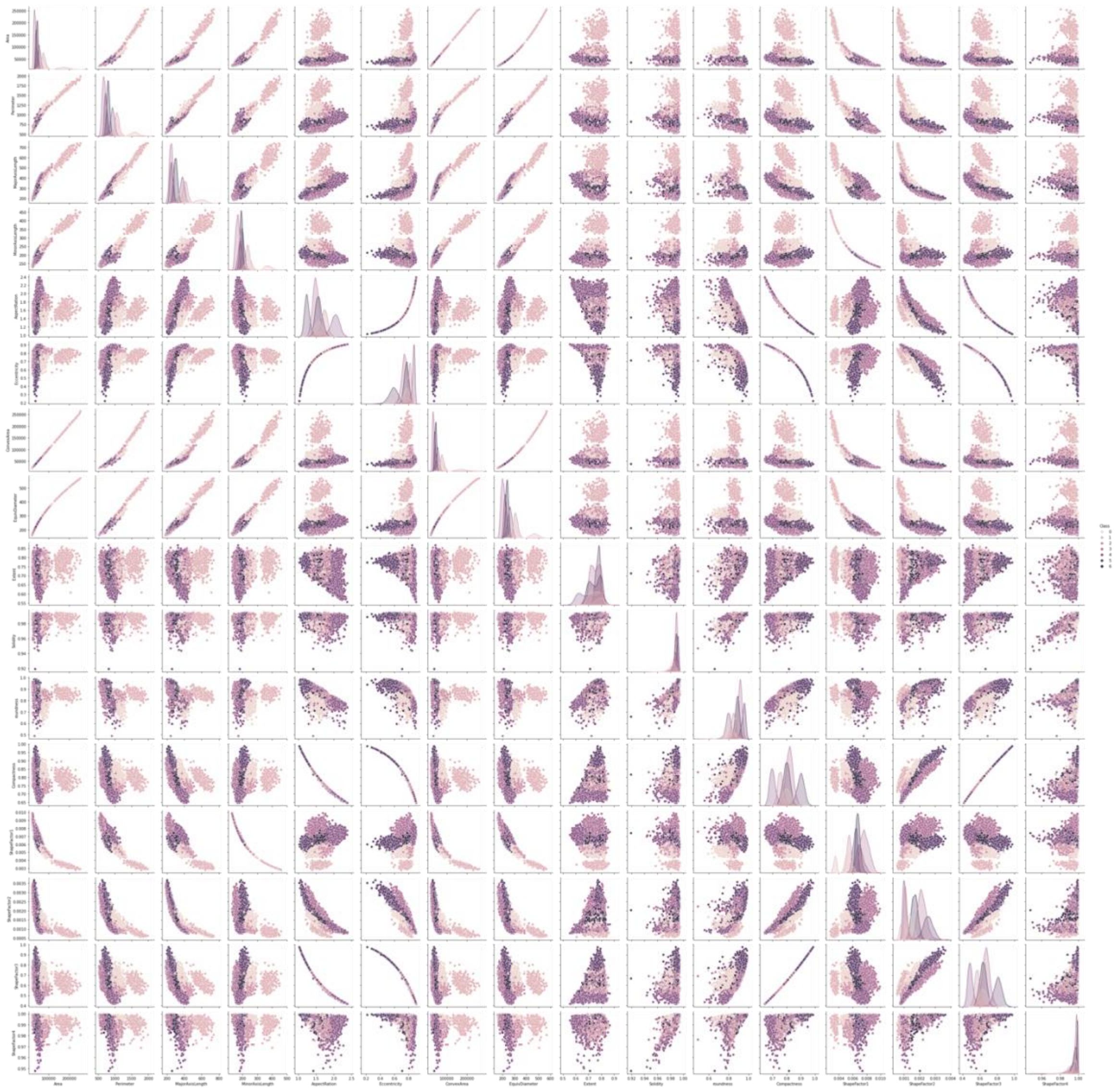
Negatively Correlated (>95%)

Var 1	Var 2	Correlation Coefficient
aspect ration	compactness	-0.99
eccentricity	shape factor 3	-0.98
aspect ration	shape factor 3	-0.98
eccentricity	compactness	-0.97
minor axis length	shape factor 1	-0.95

It's intriguing to see that certain characteristics show a high (linear) correlation with one another. For instance, among "compactness" and "shape factor 3," and between "convex area" and "area." This is expected given how closely connected the "area" and the "convex area" are. Although the calculation of "shape factor 3" and the other "shape factors" is not entirely obvious, it is reasonable to believe that the "compactness" of the beans had some sort of effect.

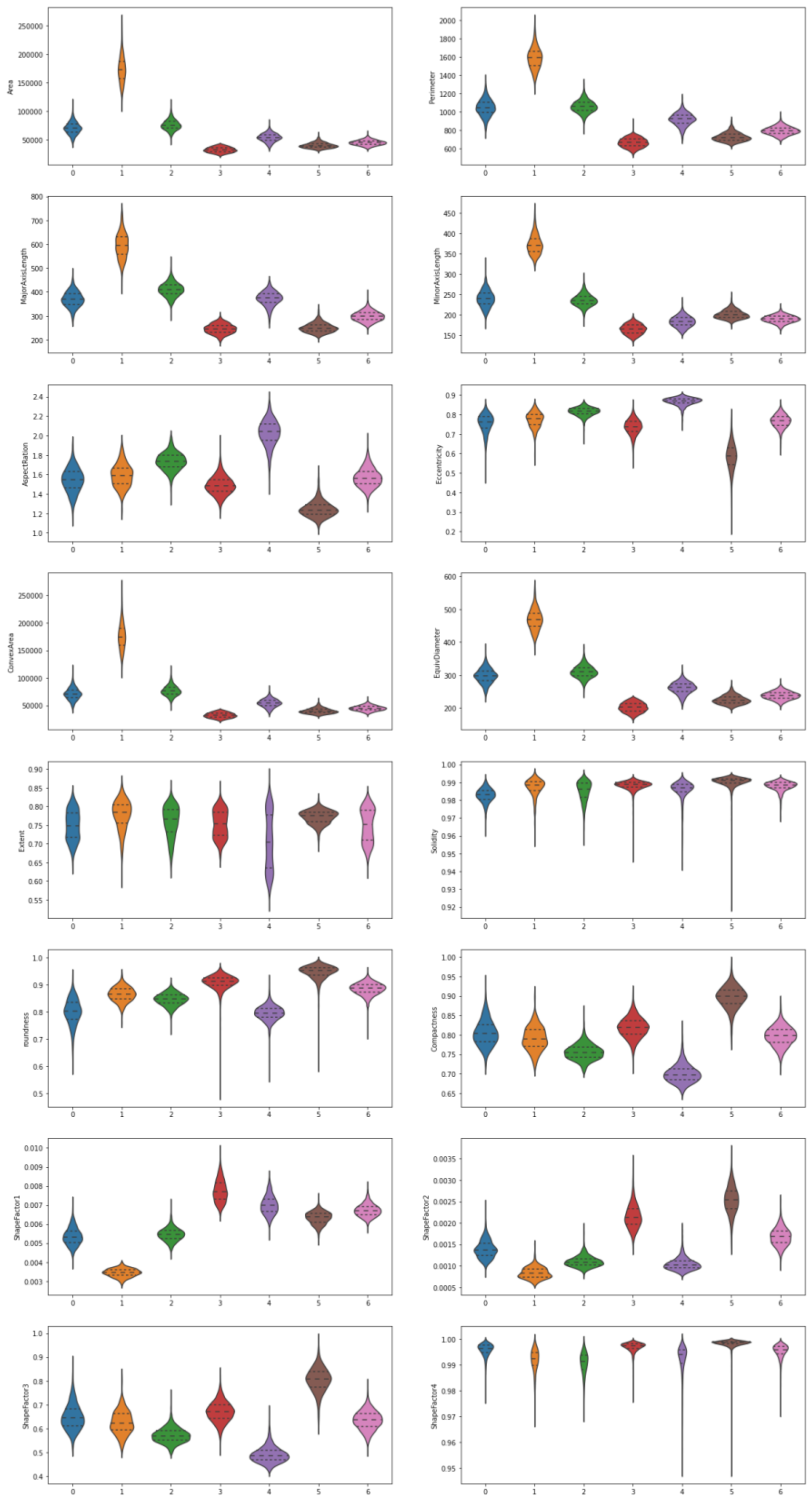
Overall, there is a significant link between traits that can be both favorable and negative. The most associated ones should be eliminated; however, the performance of the neural network was unaffected by less characteristics. The finished model thus functions with all characteristics.

- Pair plot



The pair plot shows us how the data is linearly separable in different dimensions.

- Violin Plot



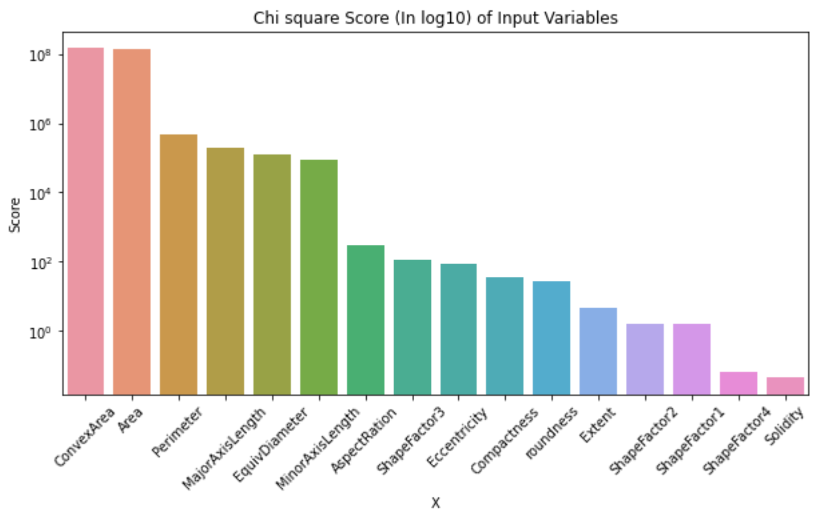
The Violin plot shows the distribution of input variable with respect to each class

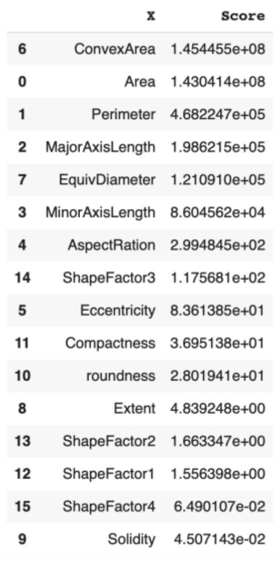
DATA PREPROCESSING

Feature Engineering

- **Chi square test**

  Chi square test can be used to identify features that are independent of the response and need not be considered for training.

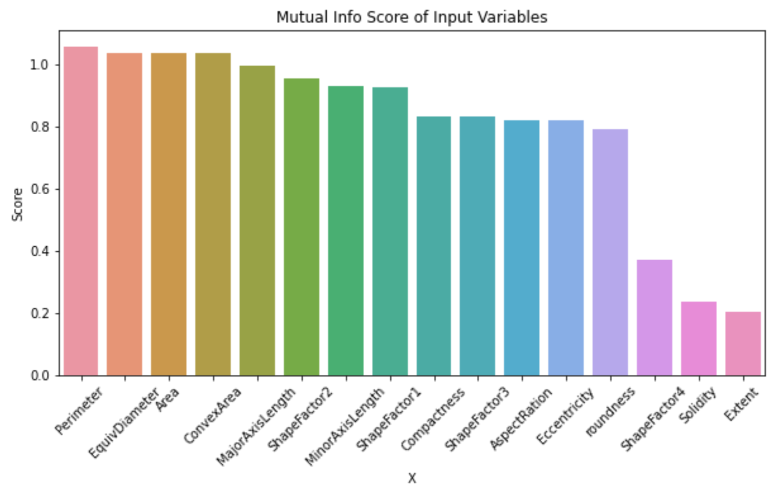


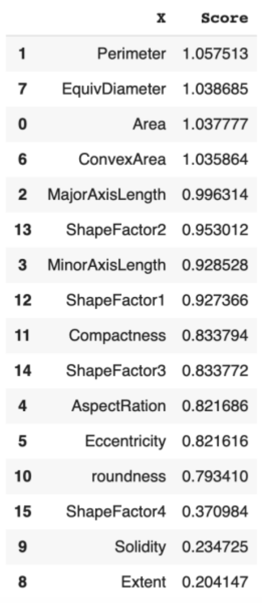


	X	Score
6	ConvexArea	1.454455e+08
0	Area	1.430414e+08
1	Perimeter	4.682247e+05
2	MajorAxisLength	1.986215e+05
7	EquivDiameter	1.210910e+05
3	MinorAxisLength	8.604562e+04
4	AspectRation	2.994845e+02
14	ShapeFactor3	1.175681e+02
5	Eccentricity	8.361385e+01
11	Compactness	3.695138e+01
10	roundness	2.801941e+01
8	Extent	4.839248e+00
13	ShapeFactor2	1.663347e+00
12	ShapeFactor1	1.556398e+00
15	ShapeFactor4	6.490107e-02
9	Solidity	4.507143e-02

- **Mutual Information**

  Mutual information between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency.





	X	Score
1	Perimeter	1.057513
7	EquivDiameter	1.038685
0	Area	1.037777
6	ConvexArea	1.035864
2	MajorAxisLength	0.996314
13	ShapeFactor2	0.953012
3	MinorAxisLength	0.928528
12	ShapeFactor1	0.927366
11	Compactness	0.833794
14	ShapeFactor3	0.833772
4	AspectRation	0.821686
5	Eccentricity	0.821616
10	roundness	0.793410
15	ShapeFactor4	0.370984
9	Solidity	0.234725
8	Extent	0.204147

Both chi square test and Mutual Information suggests removing of features like shapefactor4, solidity, extent, etc. But for the baseline model we will keep all the features so that we can compare it with PCA.

Encoding

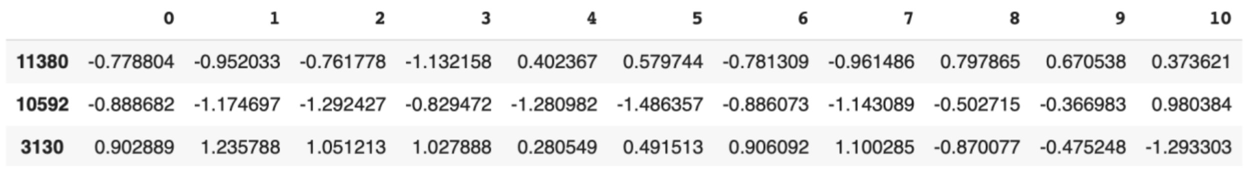
The target variable ‘class’ is encoded using label encoder. The names are converted to class numbers from 0 to 6 representing 7 classes.

Test Train Split

The dataset is split into 3 parts train, test, and valid with size 70%,15%,15% respectively.

Feature Scaling

We have applied standardization which is a scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation. As our dataset follows gaussian distribution this is very useful. Standardization is the first step before doing PCA. Dataset (first 10 features) after standardization



	0	1	2	3	4	5	6	7	8	9	10
11380	-0.778804	-0.952033	-0.761778	-1.132158	0.402367	0.579744	-0.781309	-0.961486	0.797865	0.670538	0.373621
10592	-0.888682	-1.174697	-1.292427	-0.829472	-1.280982	-1.486357	-0.886073	-1.143089	-0.502715	-0.366983	0.980384
3130	0.902889	1.235788	1.051213	1.027888	0.280549	0.491513	0.906092	1.100285	-0.870077	-0.475248	-1.293303

Principle Components Analysis (PCA)

A method for lowering the dimensionality of datasets, improving interpretability, and minimizing information loss is principal component analysis (PCA). This is accomplished by producing fresh, uncorrelated variables that maximize variance one after the other. The components are created as linear mixtures or combinations of the basic variables, acting as new variables. The top four components account for more than 95% of the variance in the dataset, according to the principal component analysis shown below. Nearly all the volatility in the data can be explained by the first four components.

Use the following algorithm to create a PCA.

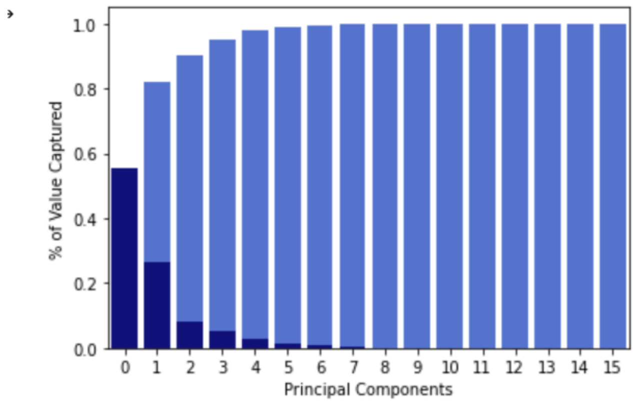
Step 1: Separate the data into X and Y where X will be the training set and Y will be the validation set.

Step 2: Give your data a structure, make the values mean centered and then standardize the data.

Step 3: Find the covariance and Eigenvectors for the data.

Step 4: Sort the Eigenvectors and select top n EV.

Step 5: Plot all these values in a graph in such a manner we can see the number of components and the variance for each component.



The cutoff for PCA is at principal component 3 in which we achieve over 95% value of the original dataset cumulatively. So, we will PC0, PC1, PC2 and PC3 as our training data

MODEL IMPLEMENTATION

We are targeting to solve the problem using Machine learning models such as Logistic Regression, Neural Network, and Naive Bayes

1) Logistic Regression

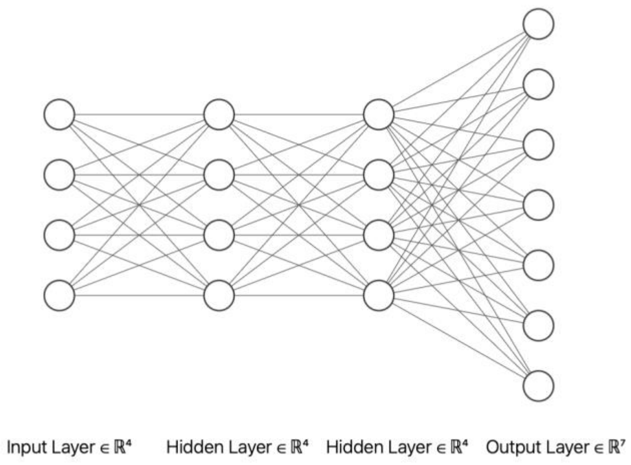
Logistic regression by its nature is used mainly for binary classification. It does this by predicting categorical outcomes, unlike linear regression that predicts a continuous outcome. In the simplest case there are two outcomes, which is called binomial, an example of which is predicting if a tumor is malignant or benign. Other cases have more than two outcomes to classify, in this case it is called multinomial. Instead of building Multinomial logistic regression classifier we have created 7 binary classifiers as a baseline model. We have transformed the dataset for each class and followed 1 vs others approach in training. For example, if classifier 1is trained for class 1 then target variable is set to 1 for all the rows belonging to class 1 rest 6 classes the target variable is set to 0. The Logistic regression model is trained with both PCA as well as scaled dataset without PCA applied. The formula to calculate the probability of belonging to class 1 is

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p)}}$$

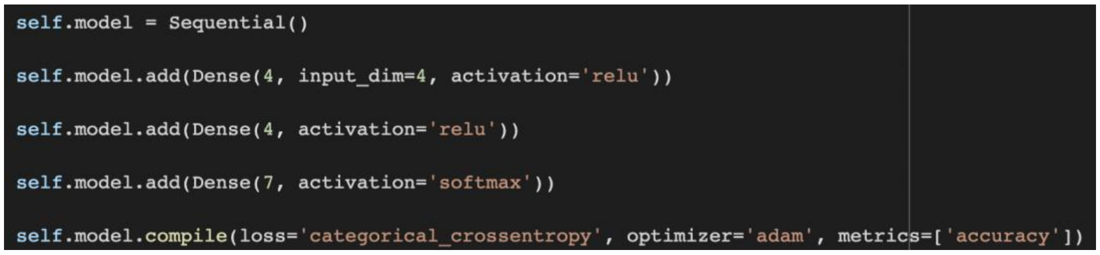
A possible problem with this approach is when 2 or more classes are predicted as 1. There we fail to interpret which class the model belongs. To overcome this we have created predict_proba function which doesn't round off the probability with respect to threshold. So now the 7 models outputs 7 individual probability score of belonging to class 1. We choose the model with highest probability score and save it as y_hat for the row.

2) Neural Network

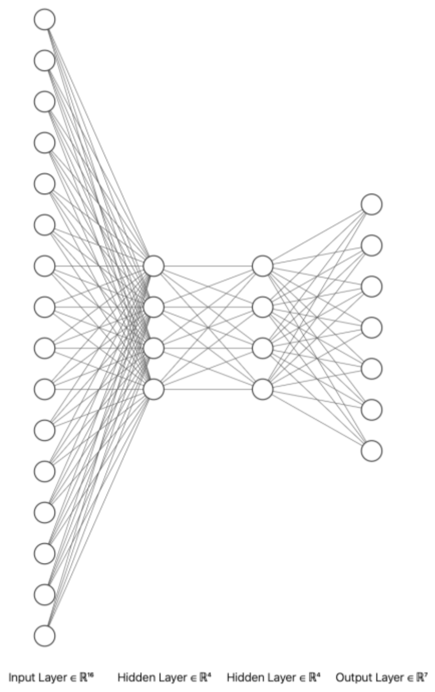
Neural Networks rely on training data to improve their accuracy over time. To practically perform it, we have implemented a Neural Network consisting of layers with input dimensions = 4 for PCA, activation function ReLu and outputs a value between 0.0 to 1.0



Neural Network Architecture with PCA (input_dim = 4)



```
self.model = Sequential()

self.model.add(Dense(4, input_dim=4, activation='relu'))

self.model.add(Dense(4, activation='relu'))

self.model.add(Dense(7, activation='softmax'))

self.model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```



Neural Network Architecture without PCA (input_dim = 16)

The neural network has several other components in it,

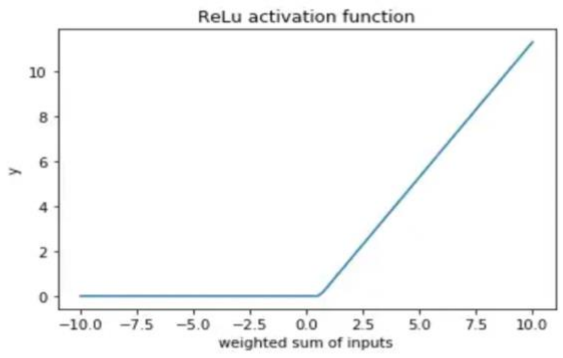
- ReLu Activation Function:

  It's the most used function because of its simplicity and it should always be used in the hidden layers of a neural network.

  Its defined as:

$$f(x) = \max(0, a) = \max(0, \textstyle\sum_{i=1}^{i=n} w_i x_i + b)$$

If the input is a positive number, the function returns the number itself and if the input is a negative number, then the function returns 0.



The derivative is given as below:

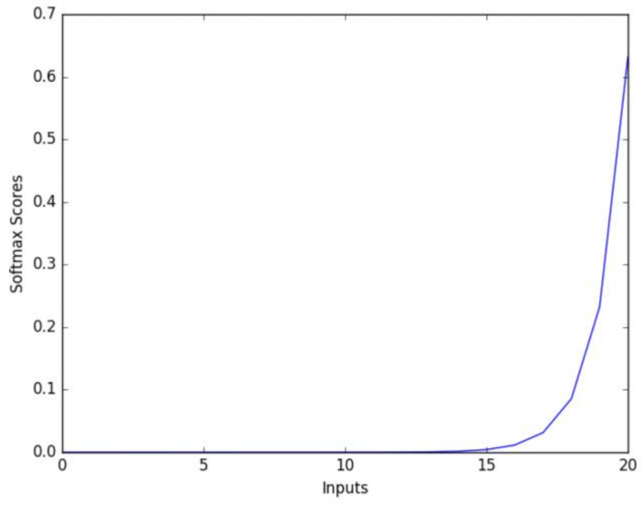
$$f'(x) = \frac{\partial f(x)}{\partial x} = \begin{cases} 0 & if\ x < 0 \\ 1 & if\ x > 0 \end{cases}$$

- SoftMax Activation Function:

  The SoftMax function is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution. The soft max activation function calculates the relative probabilities.

$$softmax(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

  Here, the Z represents the values from the neurons of the output layer. The exponential acts as the nonlinear function. Later these values are divided by the sum of exponential values to normalize and then convert them into probabilities.

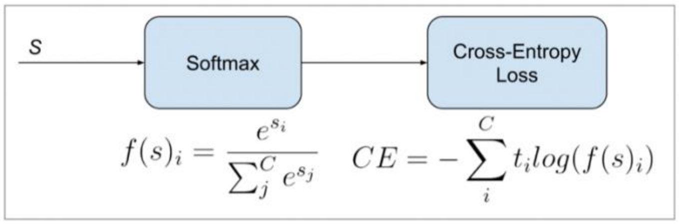


Softmax Graph

  The figure shows the fundamental property of the SoftMax function. The high value will have a high probability.

- Loss Function (Categorical Cross Entropy):

  Also called SoftMax Loss. It is a SoftMax activation plus a Cross-Entropy loss. If we use this loss, we will train a neural network model to output a probability over the C classes for each neuron. It is used for multi-class classification.


s
Softmax
Cross-Entropy Loss
$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$
$CE = -\sum_i^C t_i log(f(s)_i)$

In the specific (and usual) case of Multi-Class classification the labels are one-hot, so only the positive class C_p keeps its term in the loss. There is only one element of the Target vector t which is not zero $t_i = t_p$. So discarding the elements of the summation which are zero due to target labels, we can write:

$$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$

When Softmax loss is used is a multi-label scenario, the gradients get a bit more complex, since the loss contains an element for each positive class. Consider M are the positive classes of a sample. The CE Loss with Softmax activations would be:

$$CE = \frac{1}{M}\sum_p^M -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$

3) Gaussian Naïve Bayes

Naive Bayes is among one of the very simple and powerful algorithms for classification based on Bayes Theorem with an assumption of independence among the predictors. The Naive Bayes classifier assumes that the presence of a feature in a class is not related to any other feature. Naive Bayes is a classification algorithm for binary and multi-class classification problems.

Bayes Theorem

```
P(H|E) = P(E|H)*P(H)/P(E)
```

In Gaussian NB, the likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

are estimated using maximum likelihood.

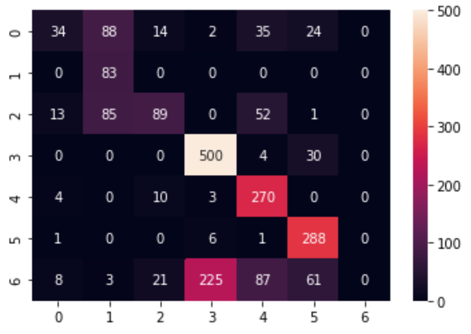
MODEL PERFORMANCE

- Logistic Regression

With PCA

Train Accuracy: 62.98%
Test Accuracy: 61.9%

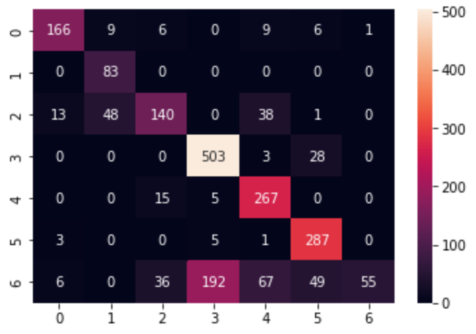
```
              precision    recall  f1-score   support

           0       0.57      0.17      0.26       197
           1       0.32      1.00      0.49        83
           2       0.66      0.37      0.48       240
           3       0.68      0.94      0.79       534
           4       0.60      0.94      0.73       287
           5       0.71      0.97      0.82       296
           6       0.00      0.00      0.00       405

    accuracy                           0.62      2042
   macro avg       0.51      0.63      0.51      2042
weighted avg       0.51      0.62      0.53      2042
```



Without PCA

Train Accuracy: 73.2%
Test Accuracy: 73.51%

```
              precision    recall  f1-score   support

           0       0.88      0.84      0.86       197
           1       0.59      1.00      0.74        83
           2       0.71      0.58      0.64       240
           3       0.71      0.94      0.81       534
           4       0.69      0.93      0.79       287
           5       0.77      0.97      0.86       296
           6       0.98      0.14      0.24       405

    accuracy                           0.74      2042
   macro avg       0.76      0.77      0.71      2042
weighted avg       0.78      0.74      0.68      2042
```

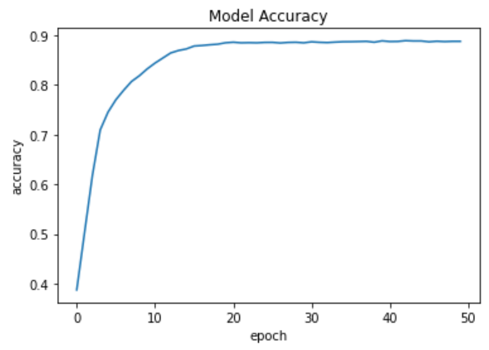


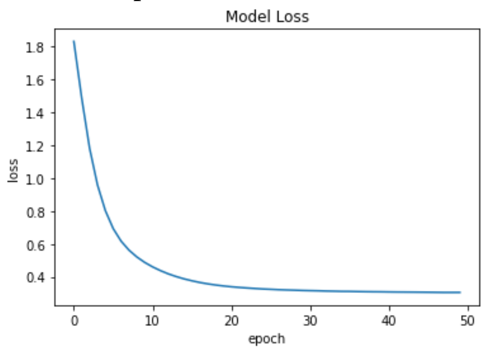
- Neural Network:

With PCA

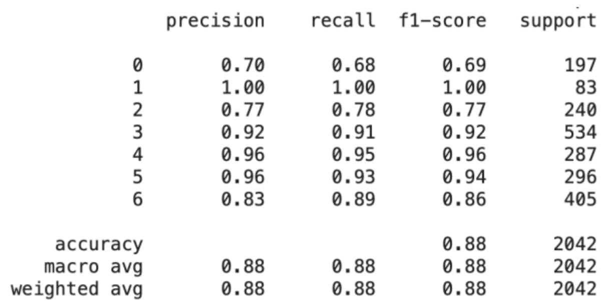
Train Accuracy: 88.91%
Test Accuracy: 88.0%

- Plotting the Model accuracy and loss vs epochs

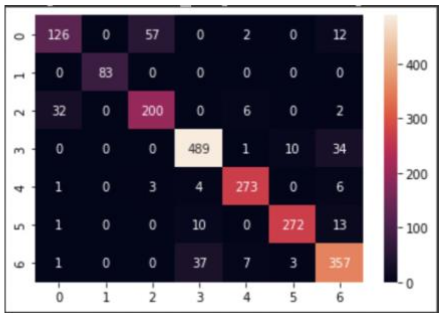




- Confusion Matrix and Classification report



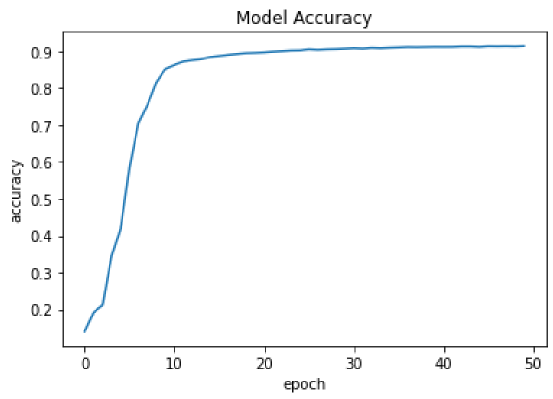
```
              precision    recall  f1-score   support

           0       0.70      0.68      0.69       197
           1       1.00      1.00      1.00        83
           2       0.77      0.78      0.77       240
           3       0.92      0.91      0.92       534
           4       0.96      0.95      0.96       287
           5       0.96      0.93      0.94       296
           6       0.83      0.89      0.86       405

    accuracy                           0.88      2042
   macro avg       0.88      0.88      0.88      2042
weighted avg       0.88      0.88      0.88      2042
```

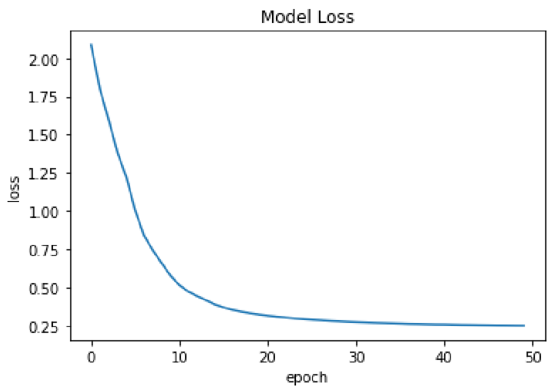


Without PCA:

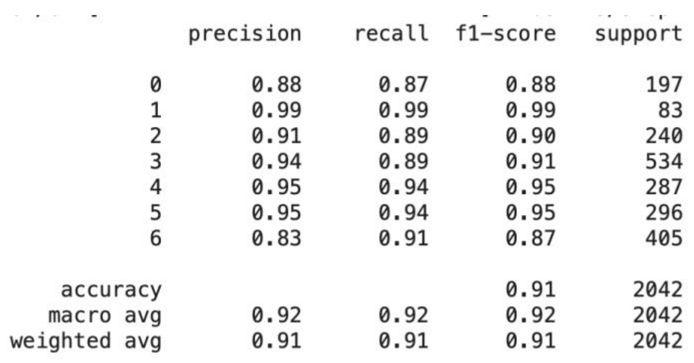
Train Accuracy: 91.14%
Test Accuracy: 91.14%

- Plotting the Model accuracy and loss vs epochs

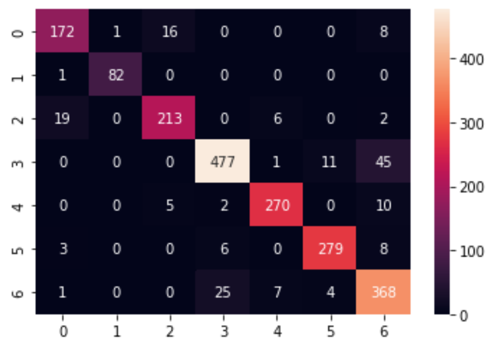




- Confusion Matrix and Classification report



```
              precision    recall  f1-score   support

           0       0.88      0.87      0.88       197
           1       0.99      0.99      0.99        83
           2       0.91      0.89      0.90       240
           3       0.94      0.89      0.91       534
           4       0.95      0.94      0.95       287
           5       0.95      0.94      0.95       296
           6       0.83      0.91      0.87       405

    accuracy                           0.91      2042
   macro avg       0.92      0.92      0.92      2042
weighted avg       0.91      0.91      0.91      2042
```

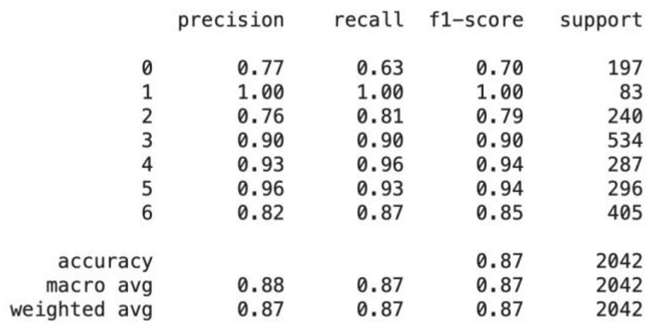


	0	1	2	3	4	5	6
0	172	1	16	0	0	0	8
1	1	82	0	0	0	0	0
2	19	0	213	0	6	0	2
3	0	0	0	477	1	11	45
4	0	0	5	2	270	0	10
5	3	0	0	6	0	279	8
6	1	0	0	25	7	4	368

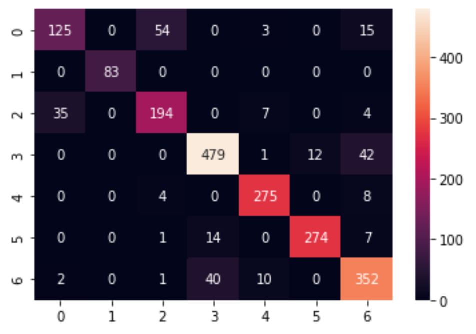
- Gaussian Naïve Bayes:

With PCA:

Train Accuracy: 86.43%
Test Accuracy: 87.27%



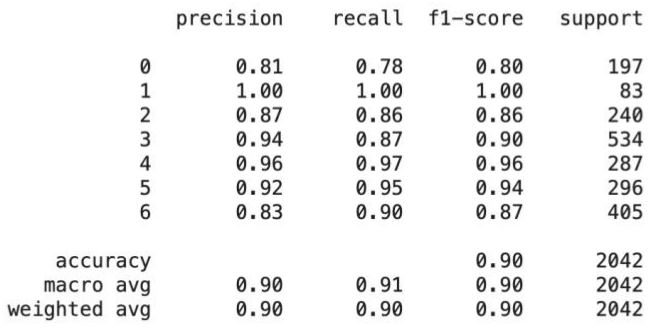
```
              precision    recall  f1-score   support

           0       0.77      0.63      0.70       197
           1       1.00      1.00      1.00        83
           2       0.76      0.81      0.79       240
           3       0.90      0.90      0.90       534
           4       0.93      0.96      0.94       287
           5       0.96      0.93      0.94       296
           6       0.82      0.87      0.85       405

    accuracy                           0.87      2042
   macro avg       0.88      0.87      0.87      2042
weighted avg       0.87      0.87      0.87      2042
```



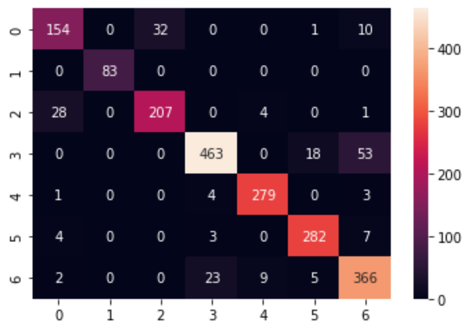
	0	1	2	3	4	5	6
0	125	0	54	0	3	0	15
1	0	83	0	0	0	0	0
2	35	0	194	0	7	0	4
3	0	0	0	479	1	12	42
4	0	0	4	0	275	0	8
5	0	0	1	14	0	274	7
6	2	0	1	40	10	0	352

Without PCA:

Train Accuracy: 89.55%
Test Accuracy: 89.81%



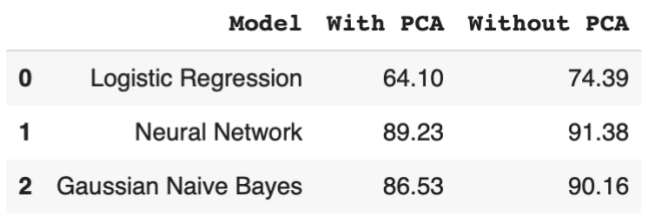
```
              precision    recall  f1-score   support

           0       0.81      0.78      0.80       197
           1       1.00      1.00      1.00        83
           2       0.87      0.86      0.86       240
           3       0.94      0.87      0.90       534
           4       0.96      0.97      0.96       287
           5       0.92      0.95      0.94       296
           6       0.83      0.90      0.87       405

    accuracy                           0.90      2042
   macro avg       0.90      0.91      0.90      2042
weighted avg       0.90      0.90      0.90      2042
```



	0	1	2	3	4	5	6
0	154	0	32	0	0	1	10
1	0	83	0	0	0	0	0
2	28	0	207	0	4	0	1
3	0	0	0	463	0	18	53
4	1	0	0	4	279	0	3
5	4	0	0	3	0	282	7
6	2	0	0	23	9	5	366

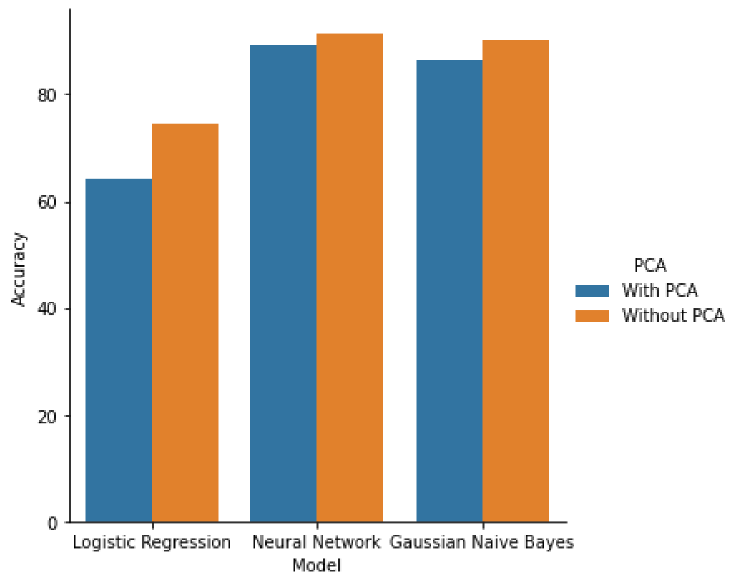
MODEL SELECTION

The final accuracy scores obtained from all the models using the validation set is as below.



	Model	With PCA	Without PCA
0	Logistic Regression	64.10	74.39
1	Neural Network	89.23	91.38
2	Gaussian Naive Bayes	86.53	90.16

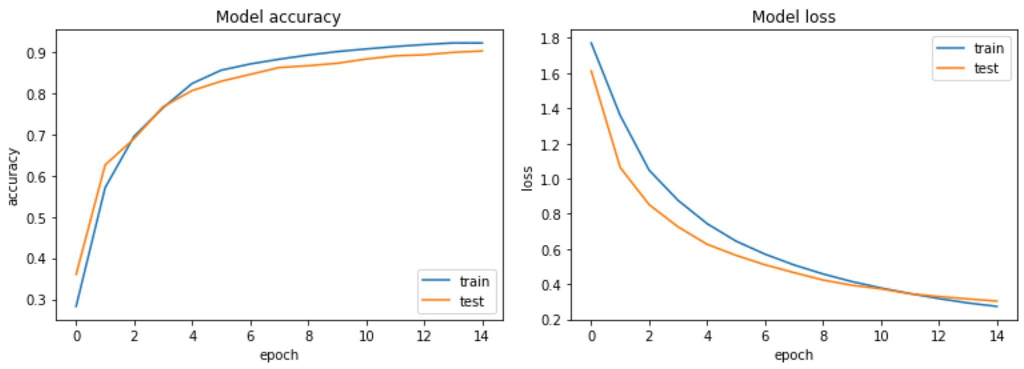
- As we can see PCA decreases model accuracy from 2-10%.



- The Neural network by far had the best performance among all models.

MODEL TUNING

The Neural Network model was retrained with reduced number of epochs, 15 from 50 since the previous models stopped learning after 15 epochs. Another drawback from initial neural network model was that it struggled classifying class 0 which resulted in low f1-score and precision for the class. To improve it, we balanced the data and retrained the model. The shape of balanced dataset is (17521, 16), (17521,1) for X_train and y_train respectively. The result of the model after retraining are presented below.



Classification Report:

Training Data

```
              precision    recall  f1-score   support

           0       0.96      0.91      0.93      2503
           1       0.99      1.00      0.99      2503
           2       0.92      0.96      0.94      2503
           3       0.89      0.88      0.89      2503
           4       0.92      0.94      0.93      2503
           5       0.93      0.96      0.95      2503
           6       0.85      0.82      0.83      2503

    accuracy                           0.92     17521
   macro avg       0.92      0.92      0.92     17521
weighted avg       0.92      0.92      0.92     17521
```

Validation Data

```
              precision    recall  f1-score   support

           0       0.93      0.90      0.92       198
           1       0.95      0.99      0.97        78
           2       0.90      0.95      0.93       239
           3       0.92      0.87      0.89       509
           4       0.91      0.93      0.92       301
           5       0.91      0.95      0.93       323
           6       0.83      0.82      0.83       394

    accuracy                           0.90      2042
   macro avg       0.91      0.92      0.91      2042
weighted avg       0.90      0.90      0.90      2042
```

We can see the improved f1-score, and precision for class 0, and class 2 meaning that balancing the data accounts significant improvement.

Fit of the Model:

To test if the model is overfitting or underfitting the error is calculated using average zero_one_loss which returns 0 if y_true[i] == y_pred[i], 1 if otherwise. On calculating the error we get,

Training Data Error: 7.64%

Test Data Error: 9.79%

Validation Data Error: 10.09%

We can see that ~2% error difference between the train and test data meaning there isn’t much overfitting in the model and it is able to generalize its prediction.