

Citrus Fruit Classification

Sai Sree Doddapuneni - CS5300

May 5, 2023

Contents

| | | |
|----------|--|----------|
| 1 | Introduction | 3 |
| 2 | Dataset | 3 |
| 2.1 | Visualization of the distribution of each input features | 4 |
| 2.2 | Equal Distribution plot of data | 5 |
| 3 | Data Processing | 6 |
| 3.1 | Data Splitting | 6 |
| 3.2 | Normalization of data | 6 |
| 4 | Modelling | 6 |
| 4.1 | Training data using Keras, TensorFlow | 6 |
| 4.2 | Plotting of testing and training accuracy using Keras Tensorflow | 7 |
| 4.3 | Accuracy on both training and validation dataset | 7 |
| 5 | Model Evaluation | 8 |
| 5.1 | Training models with one feature at a time | 8 |
| 5.2 | Plot of validation accuracies | 8 |
| 6 | Challenges Faced | 9 |
| 7 | Conclusion | 9 |

1 Introduction

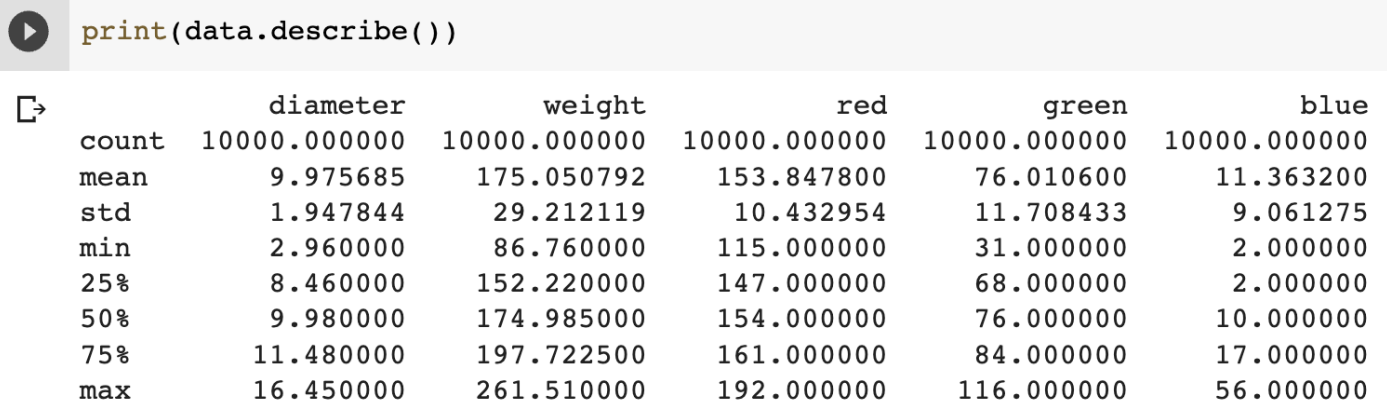
Although it seems very obvious to a human to distinguish between oranges and grapefruit, there are still some mistakes in manual observation. This dataset creates a larger dataset with a wide range of values that are "oranges" and "grapefruit" and uses the color, weight, and diameter of an "average" orange and grapefruit.

2 Dataset

The "Citrus Dataset" Downloaded the citrus dataset from kaggale website which is a good repository of datasets. The dataset contains columns diameter,weight,red,green and blue:

- **Diameter:** This aspect represents the diameter of the fruit.
- **Weight:** This feature represents the weight of the fruit.
- **red:** This indicates the colour of the fruit.
- **Green:**This indicates the colour of the fruit.
- **Blue:** This indicates the colour of the fruit.

Describing the dataset



```
print(data.describe())
```

| | diameter | weight | red | green | blue |
|-------|--------------|--------------|--------------|--------------|--------------|
| count | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 |
| mean | 9.975685 | 175.050792 | 153.847800 | 76.010600 | 11.363200 |
| std | 1.947844 | 29.212119 | 10.432954 | 11.708433 | 9.061275 |
| min | 2.960000 | 86.760000 | 115.000000 | 31.000000 | 2.000000 |
| 25% | 8.460000 | 152.220000 | 147.000000 | 68.000000 | 2.000000 |
| 50% | 9.980000 | 174.985000 | 154.000000 | 76.000000 | 10.000000 |
| 75% | 11.480000 | 197.722500 | 161.000000 | 84.000000 | 17.000000 |
| max | 16.450000 | 261.510000 | 192.000000 | 116.000000 | 56.000000 |

Convert the text in the first row to binary

```
[9] data['target'] = data['name'].map({'orange': 0, 'grapefruit': 1})
```

Figure 1:

The dataset contains 10001 data samples. Converting the text into two possible values - 0 (negative) and 1 (positive). The input features pertain to certain fields.

- Diameter, Width, Red, Green, Blue.

2.1 Visualization of the distribution of each input features

The histogram plot before normalization of each info highlights indicating their most extreme and least value as well as how they are distributed can be found in the pictures given underneath.

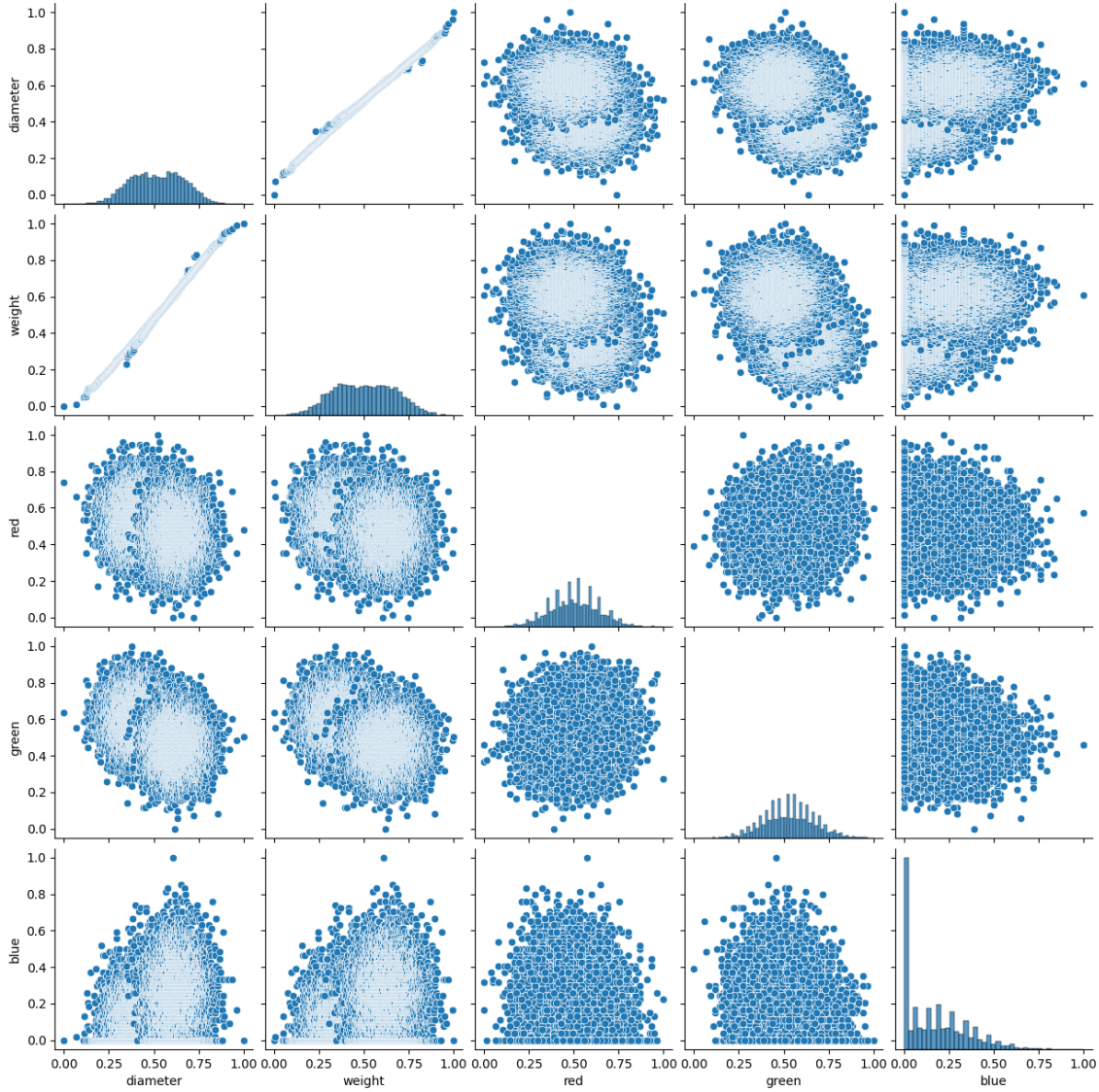


Figure 2: Input Data Distribution Histograms - Before Normalization

2.2 Equal Distribution plot of data

Notice that the data is balanced after resampling (such as oversampling, undersampling) and making equal ratio of Oranges and Grapes as shown below:

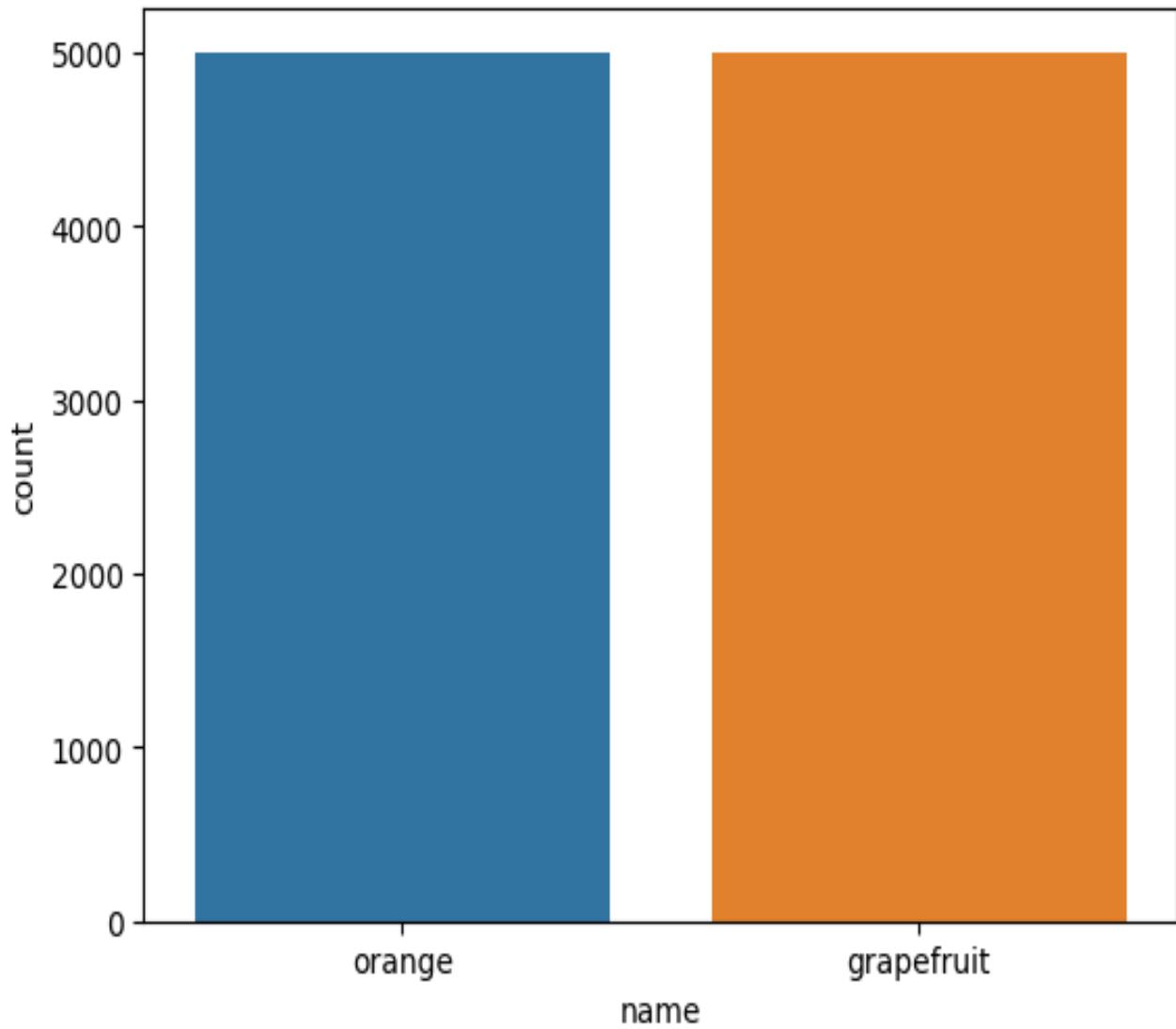


Figure 3: Output Data Distribution - After resampling

3 Data Processing

3.1 Data Splitting

The data was taken randomly and the dataset was split into training and validation, where 70% of the dataset was allocated for training and 30% was allocated for validation or testing.

3.2 Normalization of data

Prior to data mining, data pretreatment is crucial to resolving the uneven distribution of the data. In order to accomplish this, normalization techniques are utilized to strengthen training and increase the numerical stability of the optimization problem. All values should fall between 0 and 1, and outliers should be discernible in the normalized data, thanks to normalization. Both of the available normalizing methods—each with distinct side effects—can be employed at the moment.

Mean Normalization Formula

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Z-Score Normalization

$$X_{normalized} = \frac{X - X_{mean}}{X_{standard_deviation}}$$

4 Modelling

A feed forward artificial neural network architectures was used to create the model.

NOTE: Data is shuffle. Thus, the result will vary every time. All models were compiled and fit on May 10, 2022.

4.1 Training data using Keras, TensorFlow

After training the data using keras tensorflow with 150 epochs there is loss of 0.04 and accuracy of 0.98

```

305/313 [=====>.] - ETA: 0s - loss: 0.0463 - accuracy: 0.9803
Epoch 146: accuracy did not improve from 0.98440
313/313 [=====] - 1s 2ms/step - loss: 0.0480 - accuracy: 0.9812
Epoch 147/150
305/313 [=====>.] - ETA: 0s - loss: 0.0473 - accuracy: 0.9825
Epoch 147: accuracy did not improve from 0.98440
313/313 [=====] - 1s 2ms/step - loss: 0.0470 - accuracy: 0.9825
Epoch 148/150
295/313 [=====>..] - ETA: 0s - loss: 0.1074 - accuracy: 0.9676
Epoch 148: accuracy did not improve from 0.98440
313/313 [=====] - 1s 2ms/step - loss: 0.1038 - accuracy: 0.9690
Epoch 149/150
308/313 [=====>.] - ETA: 0s - loss: 0.0469 - accuracy: 0.9828
Epoch 149: accuracy did not improve from 0.98440
313/313 [=====] - 1s 2ms/step - loss: 0.0464 - accuracy: 0.9829
Epoch 150/150
307/313 [=====>.] - ETA: 0s - loss: 0.0440 - accuracy: 0.9838
Epoch 150: accuracy did not improve from 0.98440
313/313 [=====] - 1s 2ms/step - loss: 0.0442 - accuracy: 0.9836

```

Figure 4: Training the data using Keras Tensorflow

4.2 Plotting of testing and training accuracy using Keras Tensorflow

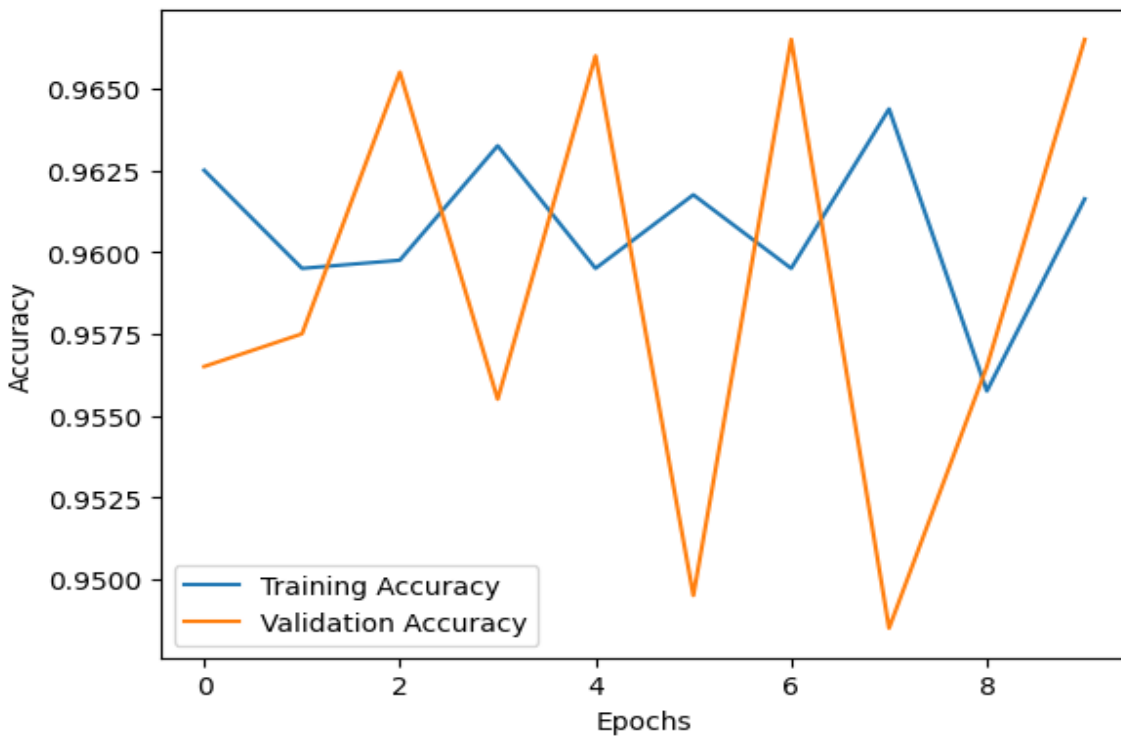


Figure 5: Plotting using Keras Tensorflow

4.3 Accuracy on both training and validation dataset

Training Accuracy : 97.90

Validation Accuracy : 97.80

5 Model Evaluation

Three essential classification model metrics to evaluate. The precision, recall and f1 score for the Keras using Tensorflow.

1. Precision: 98.02
2. Recall: 97.63
3. F1-Score: 97.82.

5.1 Training models with one feature at a time

In Feature Reduction the loss: 0.5716 and val accuracy: 0.7070 using 50 epochs.

5.2 Plot of validation accuracies

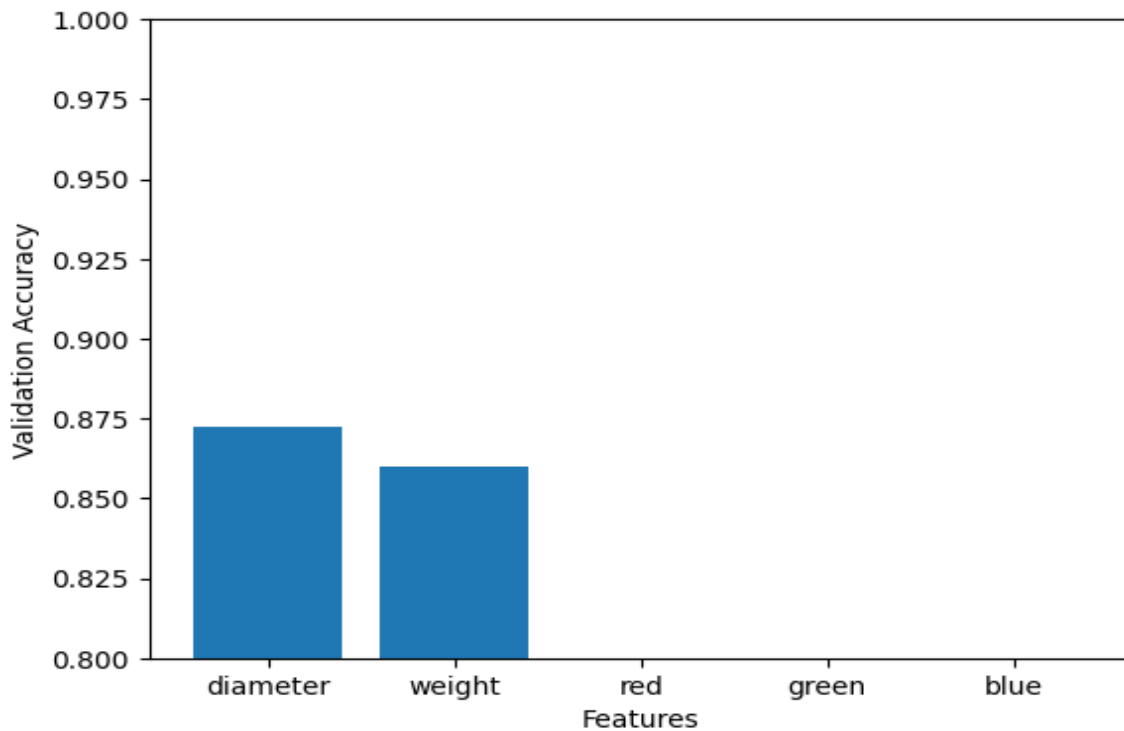


Figure 6: Plot of validation Accuracies

The bar graph between the Validation and features is shown above. Diameter has high validation accuracy when compared to weight as shown.

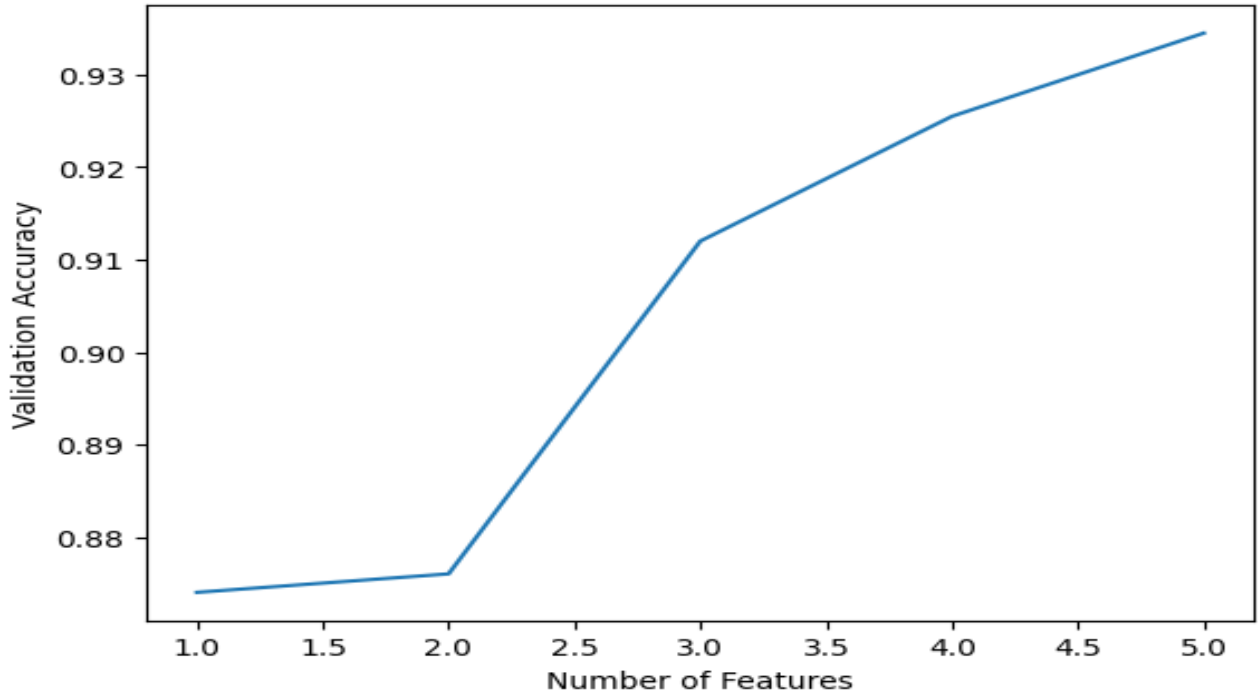


Figure 7: Validation accuracy after feature reduction

As the no of features increases the validation accuracy also increases.

After Removing the unimportant features and train models

$val_{loss} : 0.2618 - val_{accuracy} : 0.8845$

6 Challenges Faced

First i started with US Census Data which has categorical and the data conversion is challenging then i decided to get the simple dataset and i choose citrus dataset which is a simple dataset with minimum features.

7 Conclusion

This projects shows a simple classification with neural networks using tensor flow where oranges and grapes are classified by taking dataset of equal distribution and showing how the accuracy varies with features and also the with the feature reduction.