# BUILD YOUR OWN IMAGE RECOGNITION SYSTEM

# Table of Contents

# Flowers Recognition System

## Abstract

Kaggle has opensource flower dataset which contains 4242 images of flower. Each class is composed of around 800 images and it can be used for classification and clustering. The dataset has five classes of flowers: chamomile, tulip, rose, sunflower, dandelion (Mameav, 2018).

In this paper, we would like to take this opportunity to build a flower recognition system using this dataset. As this dataset is large, we have taken only 400 images per class to test KNN, SVM and AdaBoost algorithms. Due to the longer training time we conducted classification only on 4 classes: Tulip, Rose, Sunflower and Daisy.

All the parameter values for each classifier is recorded and evaluated to see the effect of parameter and its value. The best parameter is used on validation data to find out the best accuracy and use same value of parameter to train the model and test the test dataset. All three classifiers performed similar on validation dataset with the accuracy between 50-60%. However, on the test data, those classifiers could not generalize.

The SVM classifier achieved the accuracy of 46.22% on test dataset.

## Introduction

The inspiration of this system is to identify the species of flower. The dataset contains labeled flowers type. This system can be used to recognize flower type from the photos.

We will be using Bag-of-words (BoW) model to build dictionary class. In addition, K-means clustering will be used to construct word histograms for a given list of images. In this paper we aim to test KNN, SVM and Adaboost algorithms performance while using Bag-of-Words (BoW) model. We will evaluate the performance of each model.

## Data Description

The dataset is obtained from https://www.kaggle.com/alxmamaev/flowers-recognition . The data collection was based on scraped data from flickr, google images, and yandex images.

The original dataset is a set of 4242 RGB images of a fixed resolution about 320x240 pixels. However, we have chosen to use only 400 images in total where every class have 100 images each. Each image contains one of the following polygon flower type: Daisy, Tulip, Rose and Sunflower. Each image is saved in a jpg file. Image size is not equal for all images. These

images have different augmentation such as different resolutions (sizes), viewpoints, lighting conditions, occlusions, etc.

Example of Train/Test/Validation images:

[Daisy, tulip, rose, sunflower]




| Class | Daisy | Tulip | Rose | Sunflower |
|---|---|---|---|---|
| **Train** | 40 | 40 | 40 | 40 |
| **Validation** | 30 | 30 | 30 | 30 |
| **Test** | 30 | 30 | 30 | 30 |
| **Total** | **100** | **100** | **100** | **100** |

# Experiment Setup

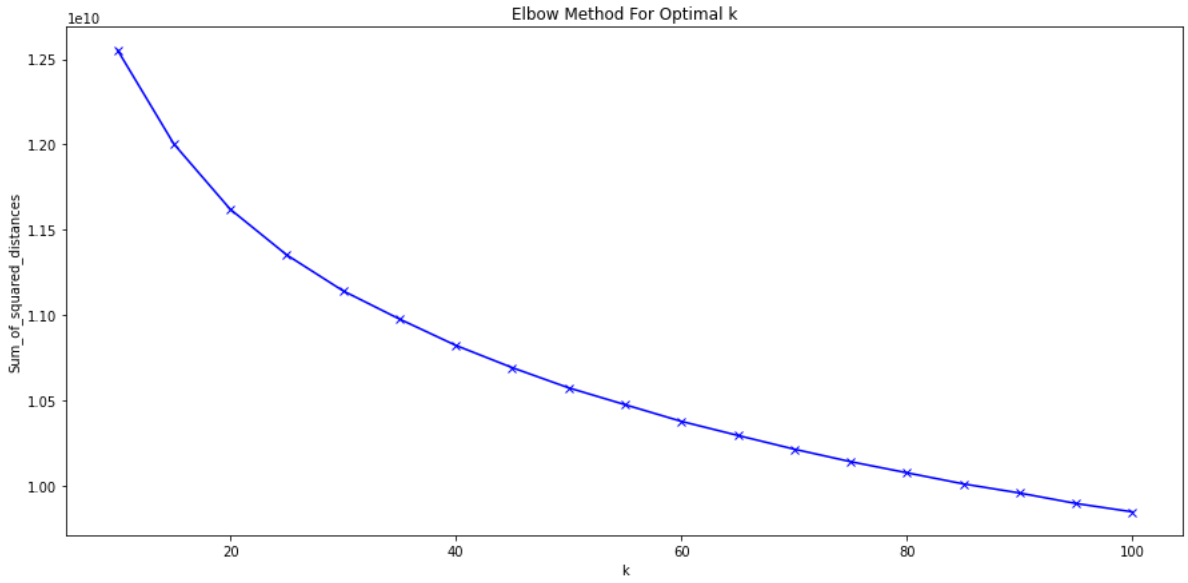
We test the accuracy of the SVM, KNN and Adaboost classifier on the flower dataset.

We divide the dataset into train, validation and test sets with 40%, 30% and 30% images respectively. The training set will be used to train classifiers, SVM, KNN and AdaBoost. The validation set will be used to tune hyperparameters for those classifiers. The test set will be used only for testing our flower recognition application.

# Methodology

## Clustering

We have used an elbow method to find the optimal value for K in K-Means clustering. Here it the graph with x axis representing k values and y axis with the sum of squared distances (inertia_).




From the plot above, we can conclude that 40 is the optimal value of k.

## KNN

N_neighbors = [10, 15, 20, 25, 30] will be used on validation images to find the higher accuracy. The validation set includes 30 images.

## SVM

The c value of hyperparameter for SVM will be c = [10, 20, 30, 40, 50, 100].

## Adaboost

n_estimators = [50, 100, 150, 200, 250] will be the parameters to find the higher accuracy of model in validation images.

# Results

## KNN on Validation Data:

```
When No. of Nearest Neighbours (K) is 10
Accuracy score is 56.67%
Recognition accuracy is 55/120
Confusion Matrix:
[[14  8  6  2]
 [ 2 25  1  2]
 [ 3  7 16  4]
```

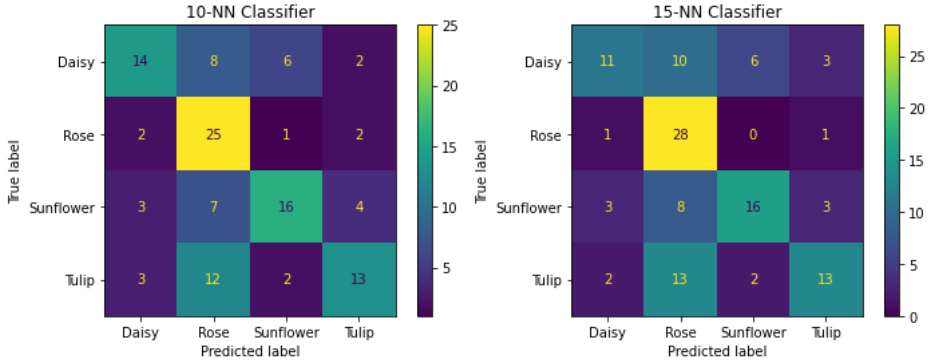
```
 [ 3 12  2 13]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4ed81b70>
Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.47      0.54        30
           1       0.48      0.83      0.61        30
           2       0.64      0.53      0.58        30
           3       0.62      0.43      0.51        30

    accuracy                           0.57       120
   macro avg       0.59      0.57      0.56       120
weighted avg       0.59      0.57      0.56       120
```

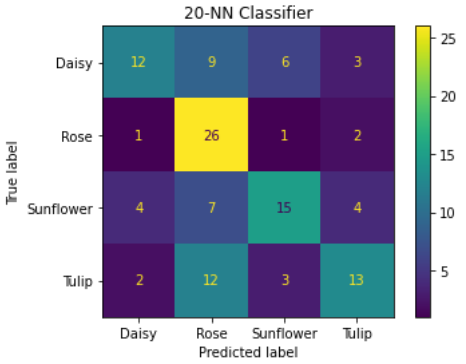
```
When No. of Nearest Neighbours (K) is 15
Accuracy score is 56.67%
Recognition accuracy is 55/120
Confusion Matrix:
[[11 10  6  3]
 [ 1 28  0  1]
 [ 3  8 16  3]
 [ 2 13  2 13]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4d4a4048>
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.37      0.47        30
           1       0.47      0.93      0.63        30
           2       0.67      0.53      0.59        30
           3       0.65      0.43      0.52        30

    accuracy                           0.57       120
   macro avg       0.61      0.57      0.55       120
weighted avg       0.61      0.57      0.55       120
When No. of Nearest Neighbours (K) is 20
Accuracy score is 55.00%
Recognition accuracy is 53/120
Confusion Matrix:
[[12  9  6  3]
 [ 1 26  1  2]
 [ 4  7 15  4]
 [ 2 12  3 13]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4eb99390>
Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.40      0.49        30
           1       0.48      0.87      0.62        30
           2       0.60      0.50      0.55        30
           3       0.59      0.43      0.50        30

    accuracy                           0.55       120
   macro avg       0.58      0.55      0.54       120
weighted avg       0.58      0.55      0.54       120
```

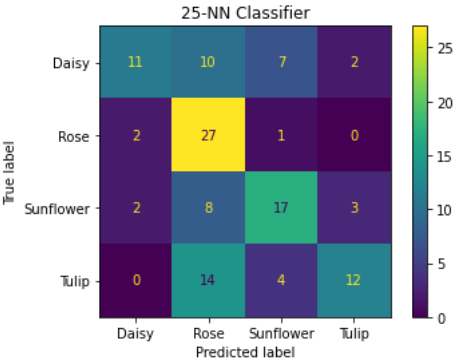
```
When No. of Nearest Neighbours (K) is 25
Accuracy score is 55.83%
Recognition accuracy is 55/120
Confusion Matrix:
[[11 10  7  2]
 [ 2 27  1  0]
 [ 2  8 17  3]
 [ 0 14  4 12]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4ea0fa20>
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.37      0.49        30
           1       0.46      0.90      0.61        30
           2       0.59      0.57      0.58        30
           3       0.71      0.40      0.51        30

    accuracy                           0.56       120
   macro avg       0.62      0.56      0.55       120
weighted avg       0.62      0.56      0.55       120
```

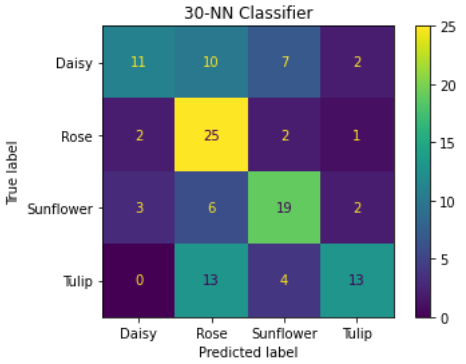
```
When No. of Nearest Neighbours (K) is 30
Accuracy score is 56.67%
Recognition accuracy is 55/120
Confusion Matrix:
[[11 10  7  2]
 [ 2 25  2  1]
 [ 3  6 19  2]
 [ 0 13  4 13]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4eda77b8>
Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.37      0.48        30
           1       0.46      0.83      0.60        30
           2       0.59      0.63      0.61        30
           3       0.72      0.43      0.54        30

    accuracy                           0.57       120
   macro avg       0.62      0.57      0.56       120
weighted avg       0.62      0.57      0.56       120
```

## Confusion Matrix for KNN (Validation Dataset):

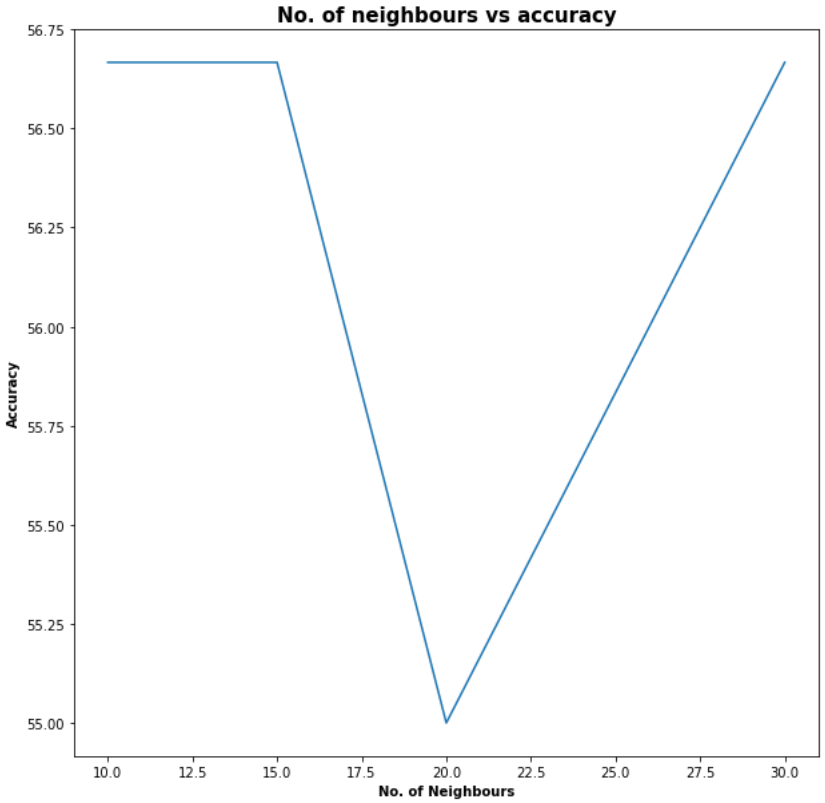




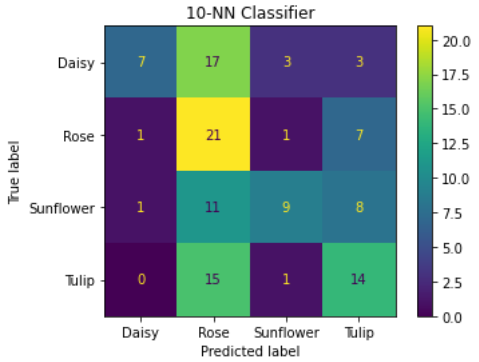




## Varying Accuracy on No. of Neighbours:



## KNN on Test Dataset:

```
When No. of Nearest Neighbours (K) is 10
Accuracy score is 42.86%
Recognition accuracy is 37/119
Confusion Matrix:
[[ 7 17  3  3]
 [ 1 21  1  7]
 [ 1 11  9  8]
 [ 0 15  1 14]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d440f2dd8>
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.23      0.36        30
           1       0.33      0.70      0.45        30
           2       0.64      0.31      0.42        29
           3       0.44      0.47      0.45        30

    accuracy                           0.43       119
   macro avg       0.55      0.43      0.42       119
weighted avg       0.55      0.43      0.42       119
```




## SVM on Validation Dataset:

```
When C value (C) is 10
Accuracy score is 58.33%
Recognition accuracy is 50/120
Confusion Matrix:
[[13  5  6  6]
 [ 1 20  1  8]
 [ 5  1 17  7]
 [ 0  5  5 20]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4cfc2be0>
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.43      0.53        30
           1       0.65      0.67      0.66        30
           2       0.59      0.57      0.58        30
           3       0.49      0.67      0.56        30

    accuracy                           0.58       120
   macro avg       0.60      0.58      0.58       120
weighted avg       0.60      0.58      0.58       120
```

```
When C value (C) is 20
Accuracy score is 59.17%
Recognition accuracy is 53/120
Confusion Matrix:
[[13  7  6  4]
 [ 2 22  1  5]
 [ 5  1 18  6]
 [ 2  6  4 18]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4dc0a2e8>
Classification Report:
              precision    recall  f1-score   support

           0       0.59      0.43      0.50        30
           1       0.61      0.73      0.67        30
           2       0.62      0.60      0.61        30
           3       0.55      0.60      0.57        30

    accuracy                           0.59       120
   macro avg       0.59      0.59      0.59       120
weighted avg       0.59      0.59      0.59       120
```

```
When C value (C) is 30
Accuracy score is 57.50%
Recognition accuracy is 50/120
Confusion Matrix:
[[13  6  6  5]
 [ 2 21  1  6]
 [ 6  2 16  6]
 [ 2  6  3 19]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4e149a20>
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.43      0.49        30
           1       0.60      0.70      0.65        30
           2       0.62      0.53      0.57        30
           3       0.53      0.63      0.58        30

    accuracy                           0.57       120
   macro avg       0.58      0.57      0.57       120
weighted avg       0.58      0.57      0.57       120
```

```
When C value (C) is 40
Accuracy score is 58.33%
Recognition accuracy is 51/120
Confusion Matrix:
[[13  8  5  4]
 [ 2 22  1  5]
 [ 6  2 16  6]
 [ 3  6  2 19]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4d529208>
Classification Report:
              precision    recall  f1-score   support
```

```
              0       0.54      0.43      0.48        30
              1       0.58      0.73      0.65        30
              2       0.67      0.53      0.59        30
              3       0.56      0.63      0.59        30

    accuracy                           0.58       120
   macro avg       0.59      0.58      0.58       120
weighted avg       0.59      0.58      0.58       120
```

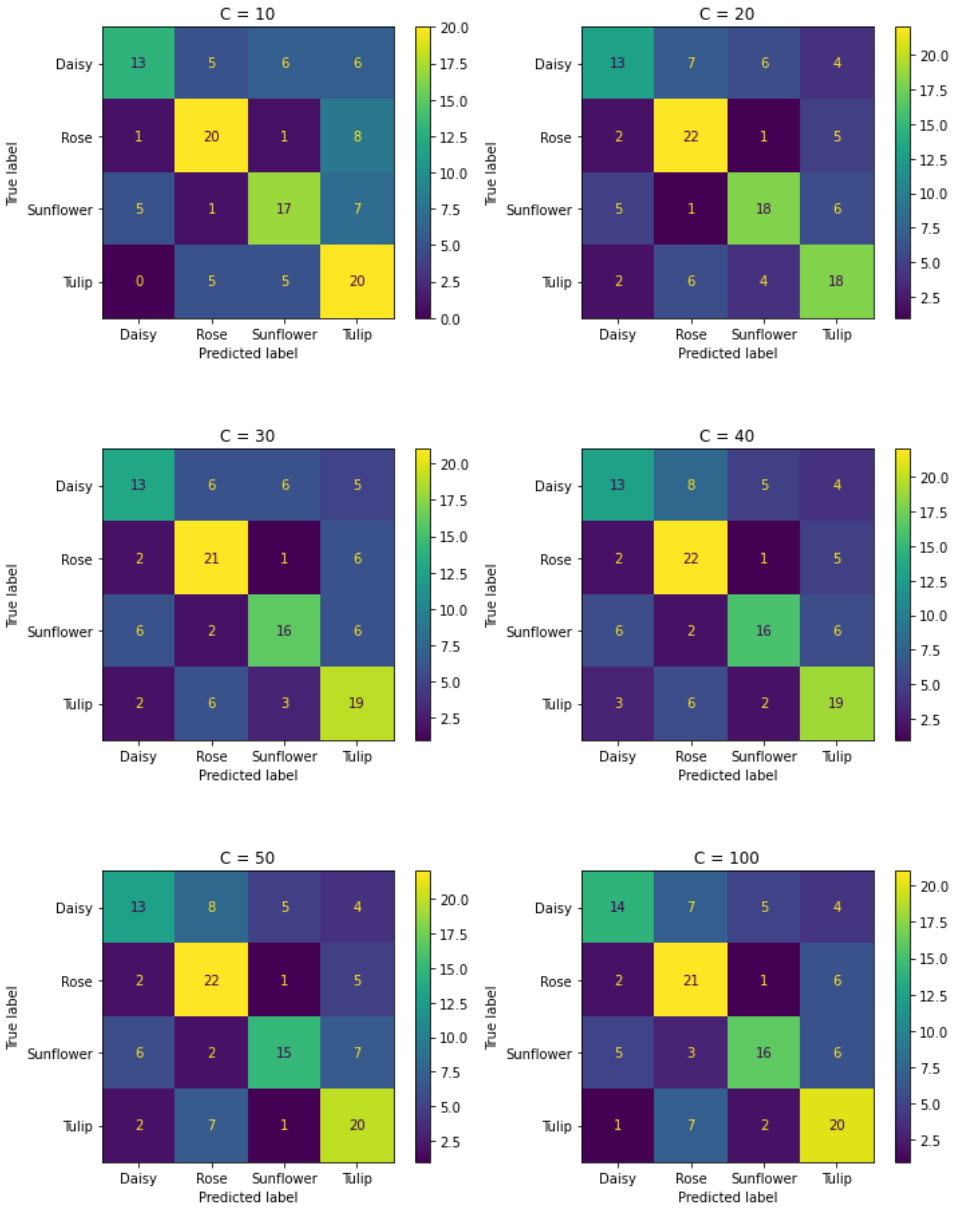
When C value **(C)** is **50**
Accuracy score is **58.33%**

```
Recognition accuracy is 50/120
Confusion Matrix:
[[13  8  5  4]
 [ 2 22  1  5]
 [ 6  2 15  7]
 [ 2  7  1 20]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4d098f98>
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.43      0.49        30
           1       0.56      0.73      0.64        30
           2       0.68      0.50      0.58        30
           3       0.56      0.67      0.61        30

    accuracy                           0.58       120
   macro avg       0.59      0.58      0.58       120
weighted avg       0.59      0.58      0.58       120
```

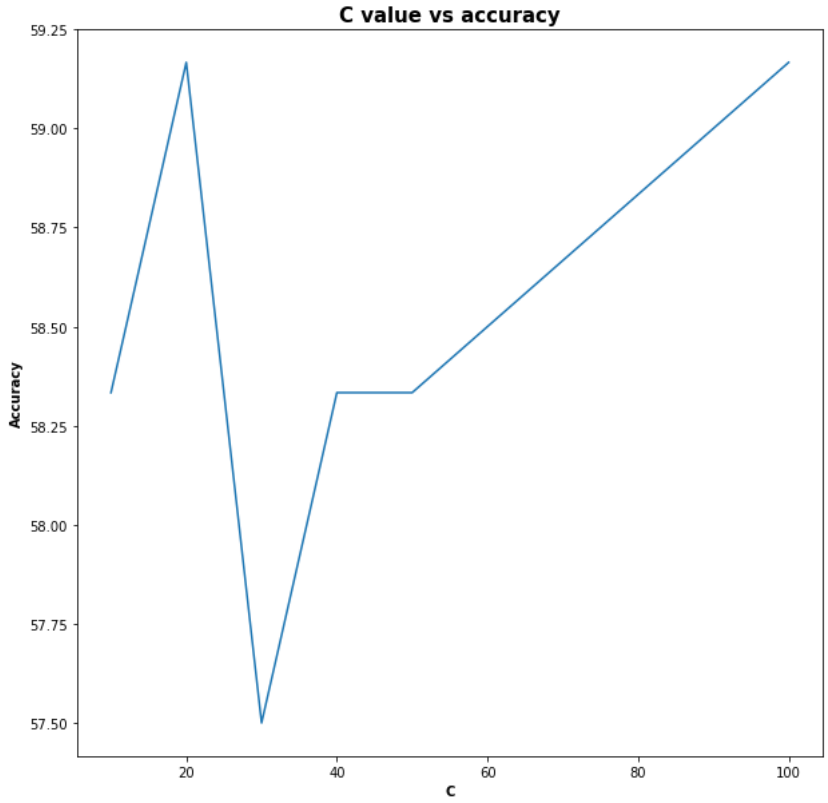
When C value (**C**) is **100**
Accuracy score is **59.17%**

```
Recognition accuracy is 51/120
Confusion Matrix:
[[14  7  5  4]
 [ 2 21  1  6]
 [ 5  3 16  6]
 [ 1  7  2 20]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4cfc2358>
Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.47      0.54        30
           1       0.55      0.70      0.62        30
           2       0.67      0.53      0.59        30
           3       0.56      0.67      0.61        30

    accuracy                           0.59       120
   macro avg       0.60      0.59      0.59       120
weighted avg       0.60      0.59      0.59       120
```

## SVM Confusion Matrix (Validation Dataset):

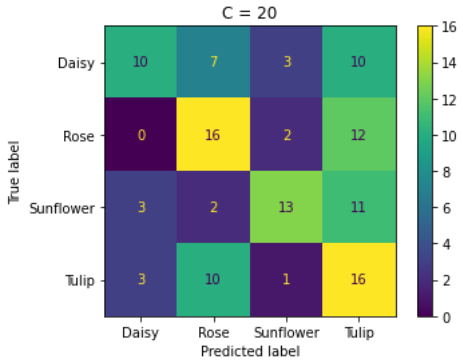


## Varying accuracy with C:




## SVM on Test Dataset:

```
When C value (C) is 20
Accuracy score is 46.22%
Recognition accuracy is 39/119
Confusion Matrix:
[[10  7  3 10]
 [ 0 16  2 12]
 [ 3  2 13 11]
 [ 3 10  1 16]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d440f2d68>
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.33      0.43        30
           1       0.46      0.53      0.49        30
           2       0.68      0.45      0.54        29
           3       0.33      0.53      0.41        30

    accuracy                           0.46       119
   macro avg       0.52      0.46      0.47       119
weighted avg       0.52      0.46      0.47       119
```




## AdaBoost on Validation Dataset:

```
When n_estimators is 50
Accuracy score is 40.00%
Recognition accuracy is 32/120
Confusion Matrix:
[[ 8  6  5 11]
 [ 4 18  3  5]
 [ 4  6  6 14]
 [ 4  6  4 16]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4e9fc940>
Classification Report:
              precision    recall  f1-score   support

           0       0.40      0.27      0.32        30
           1       0.50      0.60      0.55        30
           2       0.33      0.20      0.25        30
           3       0.35      0.53      0.42        30

    accuracy                           0.40       120
   macro avg       0.40      0.40      0.38       120
weighted avg       0.40      0.40      0.38       120
```

```
When n_estimators is 100
Accuracy score is 43.33%
Recognition accuracy is 42/120
Confusion Matrix:
[[12  5  6  7]
 [ 3 19  2  6]
 [ 4  9 11  6]
 [ 4 12  4 10]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d4f0ebd30>
Classification Report:
              precision    recall  f1-score   support

           0       0.52      0.40      0.45        30
           1       0.42      0.63      0.51        30
           2       0.48      0.37      0.42        30
           3       0.34      0.33      0.34        30

    accuracy                           0.43       120
   macro avg       0.44      0.43      0.43       120
weighted avg       0.44      0.43      0.43       120
```

When **n_estimators** is **150**
Accuracy score is **43.33%**
Recognition accuracy is 43/120
Confusion Matrix:
```
[[12  4  6  8]
 [ 4 18  3  5]
 [ 3 10 13  4]
 [ 4 11  6  9]]
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7f5d4d7e2978>
Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.52 | 0.40 | 0.45 | 30 |
| 1 | 0.42 | 0.60 | 0.49 | 30 |
| 2 | 0.46 | 0.43 | 0.45 | 30 |
| 3 | 0.35 | 0.30 | 0.32 | 30 |
| accuracy | | | 0.43 | 120 |
| macro avg | 0.44 | 0.43 | 0.43 | 120 |
| weighted avg | 0.44 | 0.43 | 0.43 | 120 |

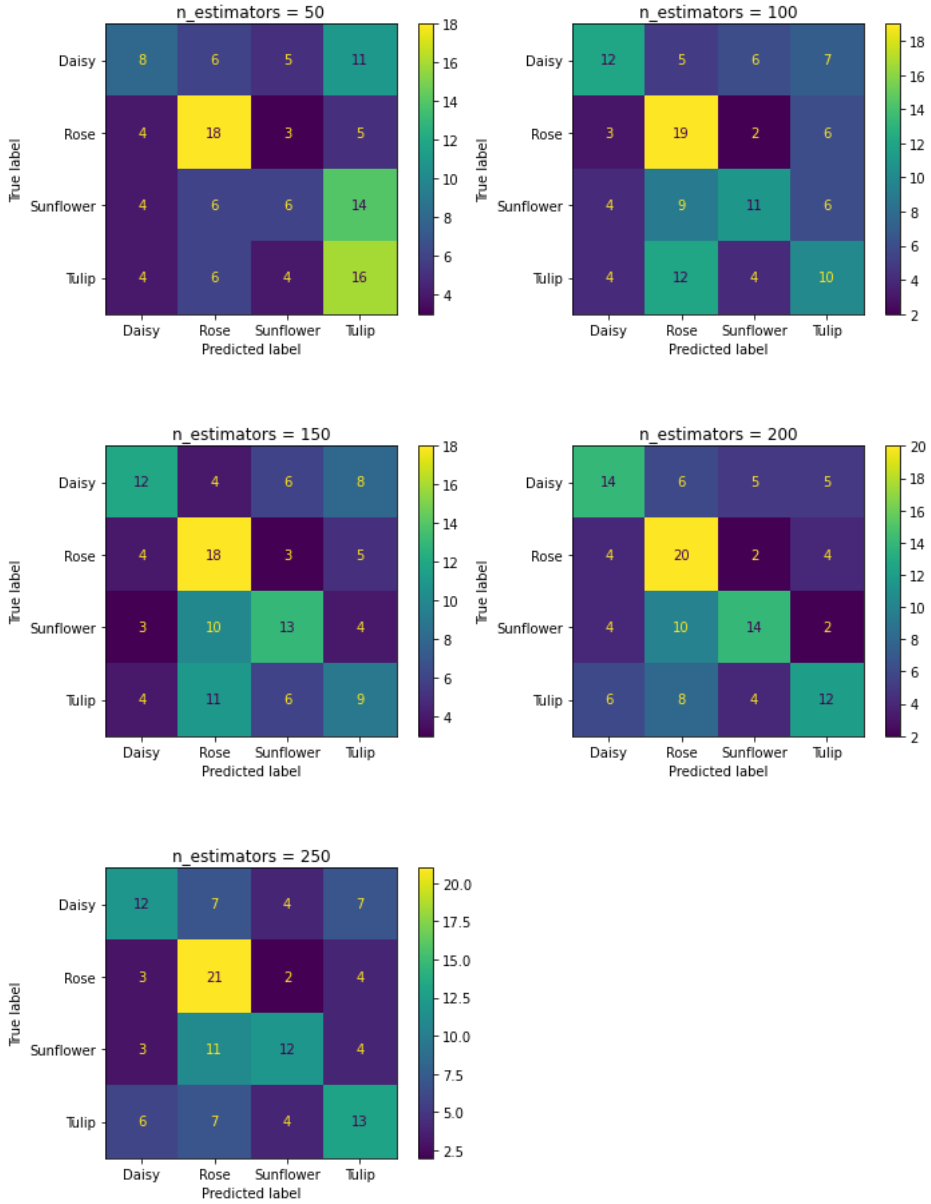
When **n_estimators** is **200**
Accuracy score is **50.00%**
Recognition accuracy is 48/120
Confusion Matrix:
```
[[14  6  5  5]
 [ 4 20  2  4]
 [ 4 10 14  2]
 [ 6  8  4 12]]
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7f5d44122438>
Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.50 | 0.47 | 0.48 | 30 |
| 1 | 0.45 | 0.67 | 0.54 | 30 |
| 2 | 0.56 | 0.47 | 0.51 | 30 |
| 3 | 0.52 | 0.40 | 0.45 | 30 |
| accuracy | | | 0.50 | 120 |
| macro avg | 0.51 | 0.50 | 0.50 | 120 |
| weighted avg | 0.51 | 0.50 | 0.50 | 120 |

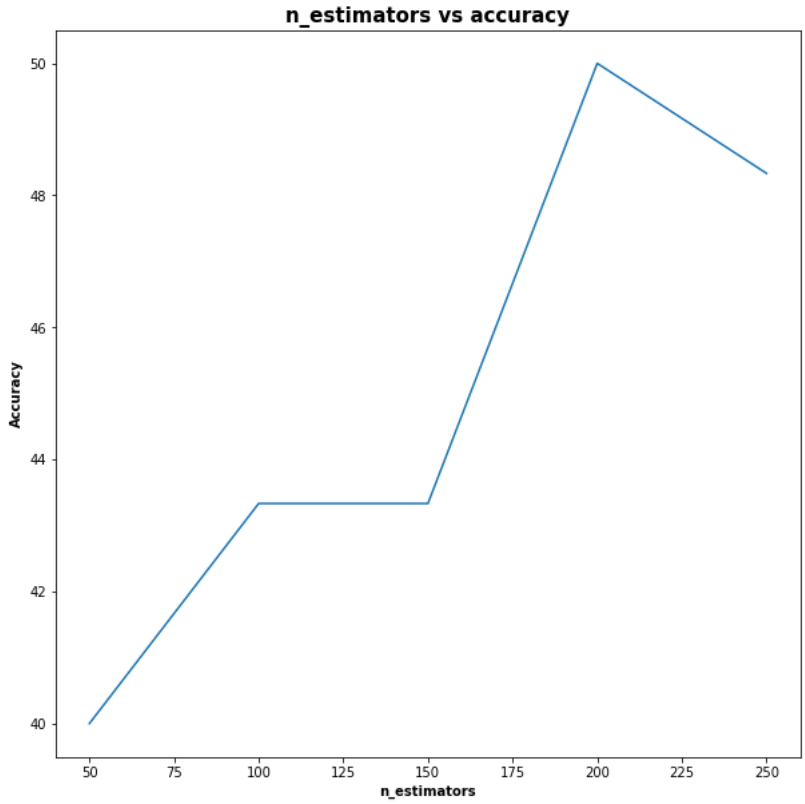
When **n_estimators** is **250**
Accuracy score is **48.33%**
Recognition accuracy is 45/120
Confusion Matrix:
```
[[12  7  4  7]
 [ 3 21  2  4]
 [ 3 11 12  4]
 [ 6  7  4 13]]
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7f5d51e1ce48>
Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.50 | 0.40 | 0.44 | 30 |
| 1 | 0.46 | 0.70 | 0.55 | 30 |
| 2 | 0.55 | 0.40 | 0.46 | 30 |
| 3 | 0.46 | 0.43 | 0.45 | 30 |
| accuracy | | | 0.48 | 120 |
| macro avg | 0.49 | 0.48 | 0.48 | 120 |
| weighted avg | 0.49 | 0.48 | 0.48 | 120 |

## AdaBoost Confusion Matrix (Validation Dataset):



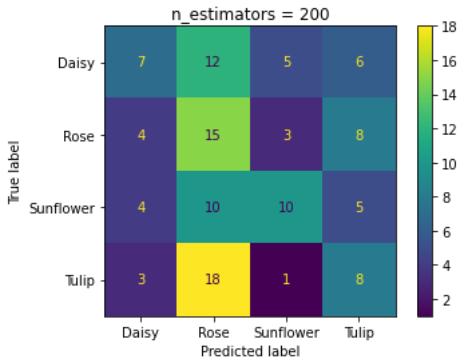
## Varying n_estimators on Validation Set:



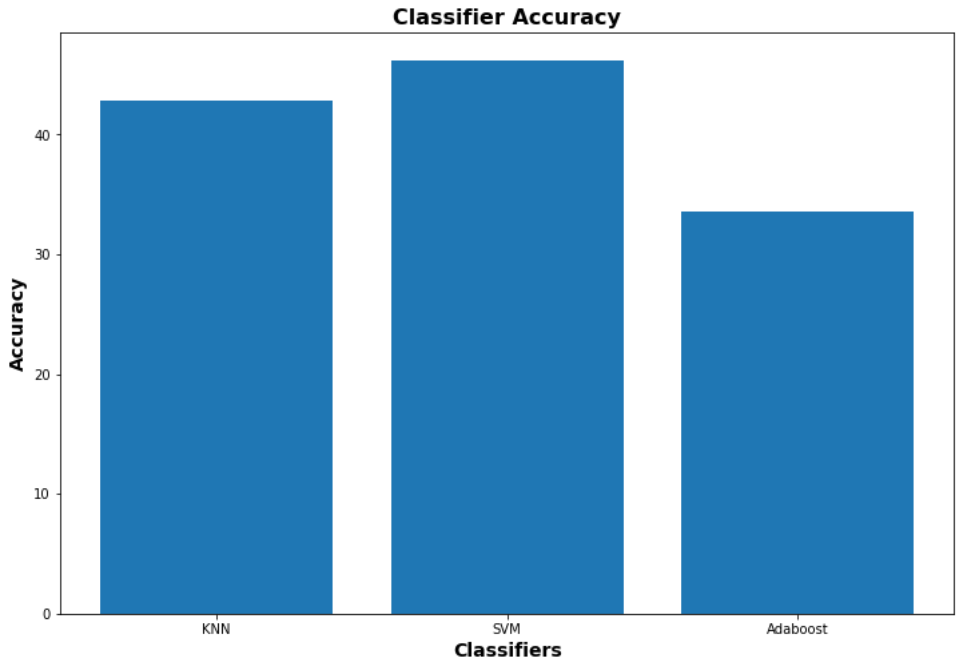

When the value of n_estimator was 200, it achieved the higher accuracy in validation dataset. The accuracy was 50%.

## AdaBoost on Test Set:

```
When n_estimators is 200
Accuracy score is 33.61%
Recognition accuracy is 32/119
Confusion Matrix:
[[ 7 12  5  6]
 [ 4 15  3  8]
 [ 4 10 10  5]
 [ 3 18  1  8]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f5d456b96a0>
Classification Report:
              precision    recall  f1-score   support

           0       0.39      0.23      0.29        30
           1       0.27      0.50      0.35        30
           2       0.53      0.34      0.42        29
           3       0.30      0.27      0.28        30

    accuracy                           0.34       119
   macro avg       0.37      0.34      0.34       119
weighted avg       0.37      0.34      0.33       119
```




## Comparison of accuracy for all classifiers:




From above bar plot, we can clearly see that SVM classifier achieved higher accuracy compared to KNN and Adaboost.

## Summary:

| Classifier | KNN | SVM | AdaBoost |
|---|---|---|---|
| **Validation (Best)** | 56.67 | **59.17** | 50 |
| **Test** | 42.86 | **46.22** | 33.61 |

# Conclusions

We used 160 images to train the BoW model and it took long time to train as it has also found the optimal value of k for Kmeans clustering. The optimal value of k is 40.

We have successfully experimented the KNN, SVM and AdaBoost classifiers using BoW model. The SVM classifier has the higher accuracy of 46.22% on test data and AdaBoost has the lowest accuracy on test data with 33.61%.

This experiment was to know the basic of how classifiers work with BoW model but if we were to implement this for actual system, we would train with all the images on the training dataset so that it can classify with higher accuracy.

For further work, more hyperparameters for the classifiers can be tuned. This paper has only tried single parameter for each classifier. In addition, these parameters can be tuned in less range as we have selected in higher difference such as 10, 20 rather than 10,11 or 12. For higher accuracy and precision, it should be trained with more images.

# Acknowledgements

We would like to thank Alexender Mamaev for providing clean dataset for classification and clustering. The dataset is publicly available at https://www.kaggle.com/alxmamaev/flowers-recognition. In addition, we would also like to thank Kaggle for sharing the data.

# References

Mameav, A., 2018. *Flowers Recognition.* [Online]
Available at: https://www.kaggle.com/alxmamaev/flowers-recognition
[Accessed 12 August 2020].