***Selected topics of programming languages***

**Project documentation**

**Project idea: Flowers species recognition**

**Project Number: 3**

**Team ID: 9**

**Team leader: Mariam Amr Fawzy**

**Team Leader ID: 202000881**

**Project description document**

***First Requirement:***

**General information on image dataset:**

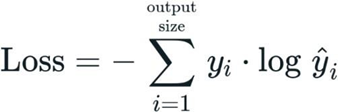
* **The name of dataset**: *Flower Species Recognition*
* **Data source**: <https://www.kaggle.com/competitions/oxford-102-flower-pytorch/data>
* **Number of classes**: 102 classes
* **Number of labels**: 102 labels
* **Example on labels and classes names**: {'21': 'fire lily', '3': 'canterbury bells', '45': 'bolero deep blue', '1': 'pink primrose', '34': 'mexican aster', '27': 'prince of wales feathers', '7': 'moon orchid', '16': 'globe-flower', '25': 'grape hyacinth', '26': 'corn poppy', '79': 'toad lily', '39': 'siam tulip', '24': 'red ginger', '67': 'spring crocus', '35': 'alpine sea holly', '32': 'garden phlox', '10': 'globe thistle', '6': 'tiger lily', '93': 'ball moss', '33': 'love in the mist', '9': 'monkshood', '102': 'blackberry lily', '14': 'spear thistle', '19': 'balloon flower', '100': 'blanket flower', '13': 'king protea', '49': 'oxeye daisy', '15': 'yellow iris', '61': 'cautleya spicata', '31': 'carnation', '64': 'silverbush', '68': 'bearded iris', '63': 'black-eyed susan', '69': 'windflower', '62': 'japanese anemone', '20': 'giant white arum lily', '38': 'great masterwort', '4': 'sweet pea', '86': 'tree mallow', '101': 'trumpet creeper', '42': 'daffodil', '22': 'pincushion flower', '2': 'hard-leaved pocket orchid', '54': 'sunflower', '66': 'osteospermum', '70': 'tree poppy', '85': 'desert-rose', '99': 'bromelia', '87': 'magnolia', '5': 'english marigold', '92': 'bee balm', '28': 'stemless gentian', '97': 'mallow', '57': 'gaura', '40': 'lenten rose', '47': 'marigold', '59': 'orange dahlia', '48': 'buttercup', '55': 'pelargonium', '36': 'ruby-lipped cattleya', '91': 'hippeastrum', '29': 'artichoke', '71': 'gazania', '90': 'canna lily', '18': 'peruvian lily', '98': 'mexican petunia', '8': 'bird of paradise', '30': 'sweet william', '17': 'purple coneflower', '52': 'wild pansy', '84': 'columbine', '12': "colt's foot", '11': 'snapdragon', '96': 'camellia', '23': 'fritillary', '50': 'common dandelion', '44': 'poinsettia', '53': 'primula', '72': 'azalea', '65': 'californian poppy', '80': 'anthurium', '76': 'morning glory', '37': 'cape flower', '56': 'bishop of llandaff', '60': 'pink-yellow dahlia', '82': 'clematis', '58': 'geranium', '75': 'thorn apple', '41': 'barbeton daisy', '95': 'bougainvillea', '43': 'sword lily', '83': 'hibiscus', '78': 'lotus', '88': 'cyclamen', '94': 'foxglove', '81': 'frangipani', '74': 'rose', '89': 'watercress', '73': 'water lily', '46': 'wallflower', '77': 'passion flower', '51': 'petunia'}
* **The total number of samples**: 8189
* **The size of each image**: 80\*80 pixels
* **The number used in training**: 6552 images
* **The number used in validation**: 818 images
* **The number used in testing**: 819 images

***Second Requirement:***

**Implementation details on image dataset:**

* Cross-validation is used in this model.
* **Number of folds used is** 10, It is the common thing to do of course! Not 9 or 11, but 10, and sometimes 5, and sometimes n-1 folds (i.e. leave-one-out cross-validation).
* **Ratio of training images**: 80%
* **Ratio of validation images**:10%
* **Ratio of testing images**: 10%
* **Batch size used**: 32 is sufficient for training purpose
* **Number of epochs used**: 20 epochs
* **Regularization used in the model**:

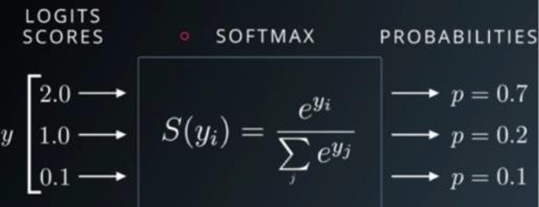
1. **Loss function**: sparse cross entropy

****

1. **Optimizer**: Adam

****

1. **Activation function used for input and hidden layers**: sigmoid
2. **Activation function used for output layer**: softmax

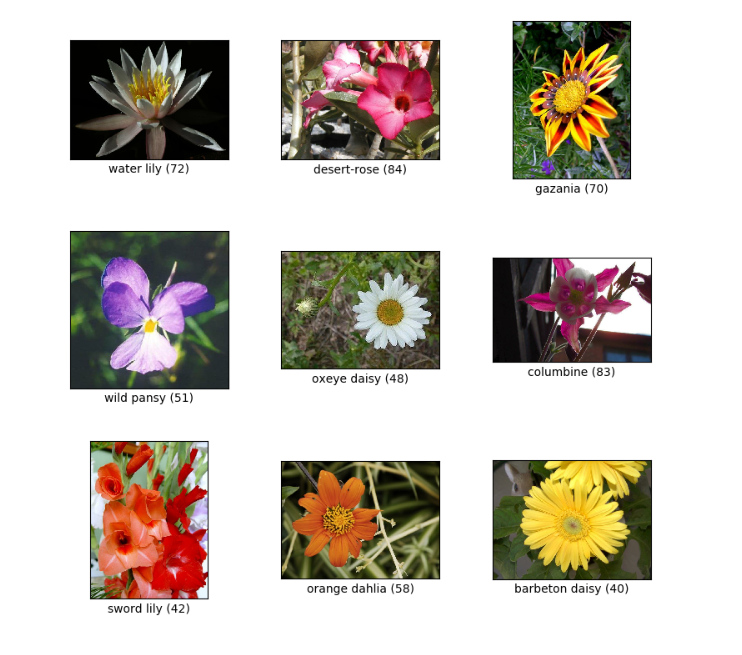
****

1. **Input shape**: 6400
2. **Hidden layers**: 2 layers
3. **Dense of input and hidden layer**: 512
4. **Dense of output layer**: 102

***Third Requirement:***

**Results details:**

* **Predicted result of image output:**

****

**Numerical dataset**

***First Requirement:***

**General information on image dataset:**

* **The name of dataset**: *Mobile Price Classification*
* **Data source:**

**https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification?select=train.csv**

* **Number of classes**: 21classes
* **Number of labels**: 21 labels
* **Example on labels and classes names**: 
* **The total number of samples**: 2000
* **The number used in training**: 1600
* **The number used in testing**: 400

***Second Requirement:***

**Implementation details on numerical dataset:**

* Ratio Of Training Samples: 80%
* Ratio Of Test Samples: 20%

**Hyperparameters**

* \*Regularization Parameter C: 100
* \*Kernel Mode: Linear
* \*Probability: True
* \*Break Ties: True
* \*Random State: 50

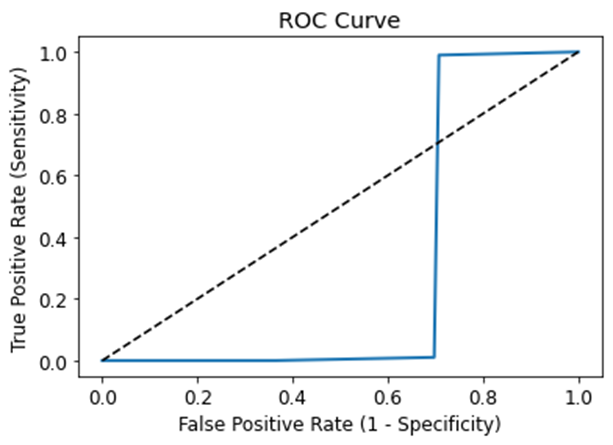
***Third Requirement:***

**Results details:**

Text

Description automatically generated

**ROC curve:**

****

**Support Vector Machine (SVM) image**

***First Requirement:***

**General information on image dataset:**

* **Number of classes**: 102 classes
* **Number of labels**: 102 labels
* **The total number of samples**: 8189
* **The size of each image**: 150\*150 pixels
* **The number used in training**: 6552 images
* **The number used in validation**: 818 images
* **The number used in testing**: 819 images

***Second Requirement:***

**Implementation details on image dataset:**

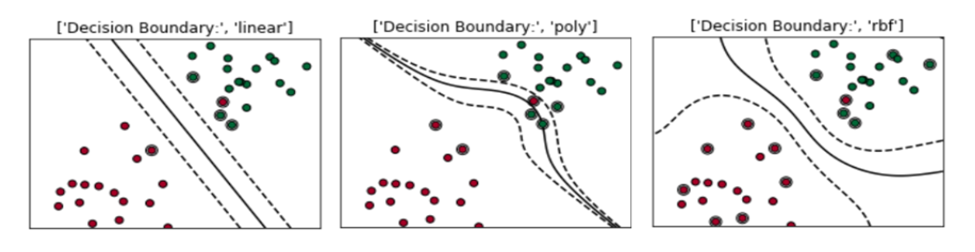
* Ratio of training images: 80%
* Ratio of validation images:10%
* Ratio of testing images: 10%
* Cross-validation is used in this model.

“Support Vector Machine” (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

Some of the key parameters in SVM are:  
→ **Gamma**: defines how far the influence of single training examples reaches values leads to biased results.

**→ C:**Controls the cost of miscalculations  
Small C — makes the cost of misclassification LOW  
Large C — makes the cost of misclassification HIGH

→ **Kernel:**SVM algorithms use a set of mathematical functions that are defined as the kernel.  
Types of Kernels are: Linear, RBF(Radial Basis Function), Polynomial Kernel

****

# ****Grid Search CV****

It is a library function that is a member of sklearn’s model\_selection package. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters.

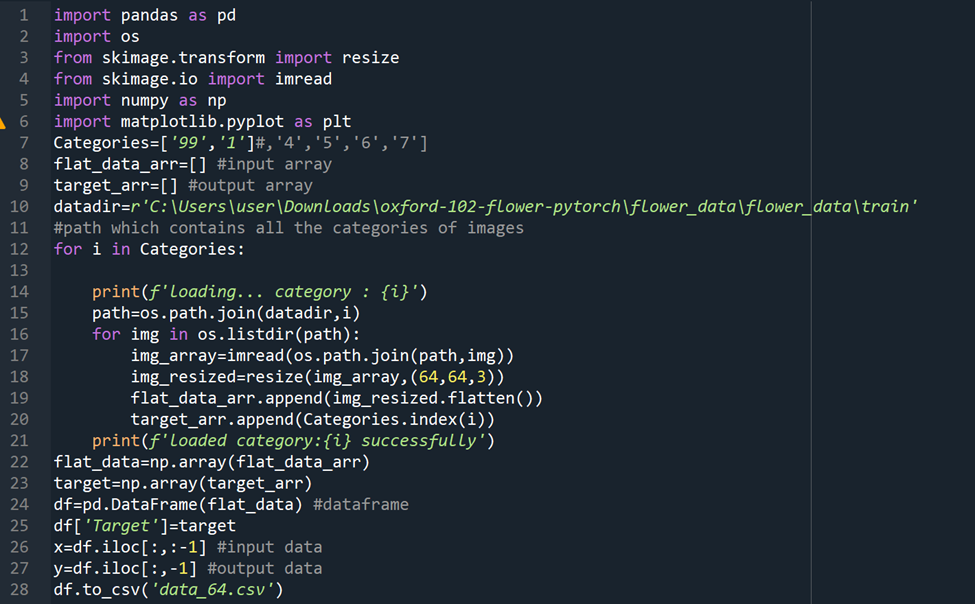
Enough of theory, let’s get started with the coding part.

# ****Process****

It is one of the ways of machine learning where the model is trained by input data and expected output data.  
To create such a model, it is necessary to go through the following phases:

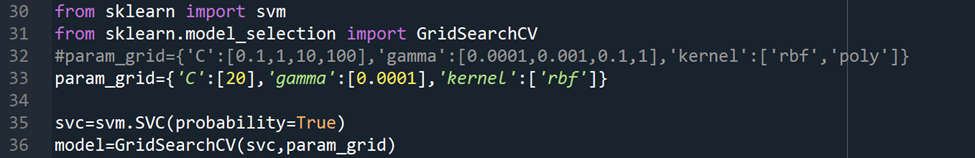
**1.Taking input  
2. Model construction  
3. Model training  
4. Model testing  
5.Model evaluation**

**Taking input:** 3 Different categories of images(Cars, Ice cream cone, Cricket ball) are read and labeled as 0,1,2 in the following way:

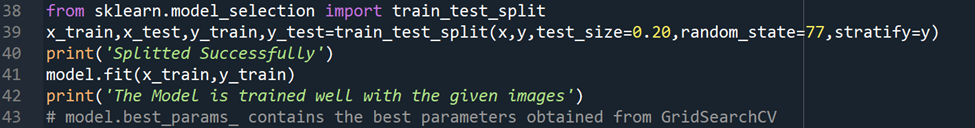
****

Since SVM receives inputs of the same size, all images need to be **resized** to a fixed size before inputting them to the SVM. df is the data frame created using pandas and x and y are input and output data respectively

**Model construction:** In this project case, the model is Support vector machine.  
The algorithm for model construction looks like this:  
1. **Create a support vector classifier:**→ svc=svm.SVC()  
2. **With the help of GridSearchCV** and parameters grid, create a model: →model=GridSearchCV(svc,parameters\_grid)

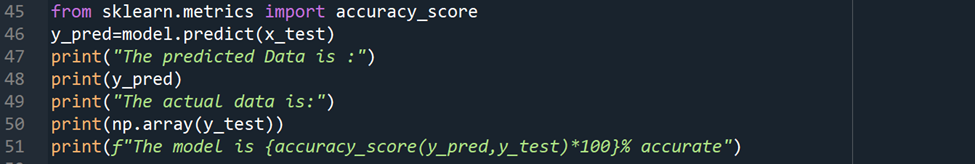
****

**Model training:**The data is split into two categories: training data and testing data. Training data is used to train the model whereas testing data is used to test the trained model.  
For splitting the data into training and testing, **train\_test\_split()** from sklearn library is used.  
Model is trained using training data in this way  
→ model.fit(training\_data,expected\_output)

****

**Model testing:**Now the model is tested using testing data in this way  
→ model.predict(testing\_data)

The accuracy of the model can be calculated using the **accuracy\_score()**method from sklearn.metrics

****

***Third Requirement:***

**Results details:**

* **Predicted result of image output:**

Text

Description automatically generated

**Logistic regression model**

***Second requirement:***

**Implementation details**

**1-Feature extraction**

As it's just a model applied in a tabular logistic dataset, so the feature extraction here where all data transforms are applied in parallel to raw input data and combined together to create one large dataset.

**2-Cross validation**

No, we didn't use cross validation in this model, we only counted on the splitting of data into train a test and the splitting was 80% for training, 20 % for testing

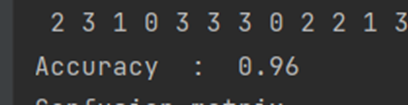
**3-Hyperparameters**

No hyperparameters applied to this model

***Third Requirement:***

**Results details:**

1-Accuracy



2-Sensitivity and specificity

Graphical user interface, text

Description automatically generated

3-Confusion matrix

Text

Description automatically generated

4-ROC plot

Chart, line chart

Description automatically generated

***Thank you😊***