```
#connecting to google drive
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

```
#Preprocessing
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from skimage.io import imread
from skimage.transform import resize

target = []                 #list for output(1-D) data(dependent variable)
images = []                 #list for input(2-D) data
flat_data = []              #list for flattened input data(1-D)(independent variable)

DATADIR = '/content/drive/MyDrive/mp trial2/Train'
CATEGORIES = ['Tomato___Leaf_Mold','Tomato___Late_blight','Tomato___healthy','Potato___Lat

for i in CATEGORIES:
  class_num = CATEGORIES.index(i)               #label encoding
  path = os.path.join(DATADIR,i)                #creating a path to use all the images
  for img in os.listdir(path):
    img_array = imread(os.path.join(path,img))
    img_resized = resize(img_array,(150,150,3))  #normalizing each and every image iterati
    flat_data.append(img_resized.flatten())     #flattening the data and storing in the li
    images.append(img_resized)
    target.append(class_num)
    #plt.imshow(img_resized)
    #plt.show()                                 #Use just in case to show the resized imag
flat_data = np.array(flat_data)
images = np.array(images)                         #transforming all the data into a 1-D ar
target = np.array(target)
```

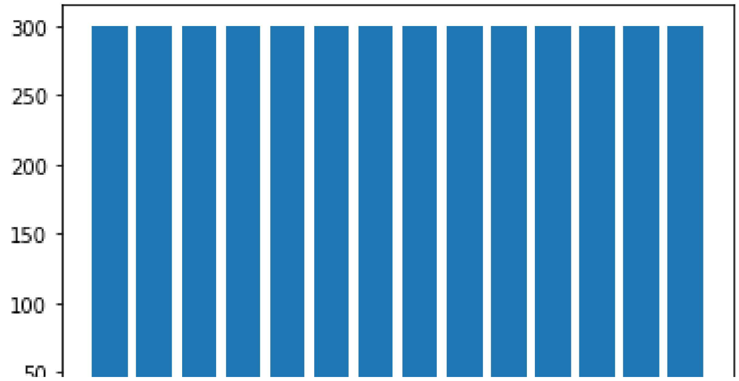
```
target
```

```
array([ 0,  0,  0, ..., 13, 13, 13])
```

```
unique,count = np.unique(target,return_counts=True)
```

```
plt.bar(unique,count)
```

```
<BarContainer object of 14 artists>
```



```
#Splitting the data into training and testing
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(flat_data,target,test_size=0.2,random_state
```

```
import pickle
import warnings
import numpy as np
import pandas as pd

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
warnings.filterwarnings('ignore')
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

SVC

```
from sklearn.svm import SVC
model = SVC(probability=True)
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
y_pred
```

```
array([ 5,  7,  5, 11,  0,  6, 13,  1, 11,  8,  3,  7,  4, 12,  9,  7, 11,
        1,  6,  2, 10,  3,  3,  8,  8, 12,  7,  5,  2,  7, 13,  8,  9,  1,
        2,  6,  1,  3,  7,  2, 12,  8,  2,  0,  4,  0,  8,  7,  3,  2,  5,
        8,  1,  4,  5,  0,  0,  5,  2, 13, 13,  3,  7,  1,  2,  6,  6,  2,
        6, 10, 12, 13,  7,  0,  3, 12,  1,  8,  3,  4,  6,  0,  9,  3,  0,
       10,  6,  2,  8,  7,  0,  0, 13,  2,  6, 12,  5,  5, 10, 13,  0,  3,
       12, 12,  8,  5,  7,  7,  5,  1,  3, 13,  8,  0,  0,  8, 11,  7, 13,
       12, 13,  2,  2,  7,  1,  8, 10,  0,  2,  7,  3,  0,  0,  9,  9,  0,
        8, 11,  1,  9,  1,  9,  2,  6,  8,  3, 11,  9,  0,  0,  5,  3,  5,
        2,  7, 13, 13,  6,  6,  6,  3,  9,  3, 11,  4,  3, 10,  5,  1,  0,
       12, 12,  3,  2,  1,  7, 10, 13,  4, 12,  3, 12,  1,  4,  8,  7,  4,
       12,  6,  2,  2, 10,  4,  0,  1, 12,  3,  9,  4,  2,  0, 10,  7, 11,
        9,  3, 10,  3, 11, 13,  7, 10,  3,  6,  7, 13, 11,  2, 12,  7,  2,
        7,  9,  9,  0,  2, 13,  6,  4,  9,  3,  9,  2,  6, 13, 12,  7,  5,
        6,  4, 12,  8,  6, 13,  9, 11,  9,  1,  2,  2, 12,  3,  4,  1, 10,
        0, 12,  6, 11,  4,  6,  5,  3,  8,  8, 13,  3, 13,  0,  7,  7, 11,
       11,  3,  2,  3,  2,  7,  4,  9,  2,  2,  6,  7,  4, 12,  4, 10,  0,
        0, 12,  8, 13,  4,  9,  5,  9,  2,  5,  3, 11,  0,  7, 11,  5, 10,
        8,  1,  7,  8,  6, 12,  4,  5,  5,  2,  5,  7,  7,  7,  9,  0,  4,
        9,  0, 11,  7, 13, 10,  6,  1, 11,  1,  1,  2,  3, 11, 13,  5,  0,
        0,  7,  3,  0, 11,  4,  3,  9,  7, 13,  1,  9,  4,  0,  0, 12,  3,
        2,  3, 11,  1,  9,  7,  1,  6,  8,  3,  2, 11,  6,  3,  7, 10,  5,
        1,  9,  5, 10, 13, 12,  7, 10,  2,  1,  8,  5,  6, 12,  9,  6, 12,
        2,  6,  7,  8,  1,  3,  6,  8, 11, 12,  8,  4,  8,  0, 12,  7,  8,
       10,  1, 12,  8,  7,  4,  6,  1,  3,  9, 12, 12,  1, 11,  2,  0, 11,
       10, 12,  3, 13,  4, 10,  6,  7,  3,  9,  5, 13,  3,  1, 13,  1,  4,
        3,  5,  1,  0,  2,  2,  0,  8,  3,  7,  5, 11, 12,  0,  1,  5,  2,
        4,  4,  2,  0,  7,  9, 13,  0, 13, 12,  8,  6, 10,  4,  3, 11,  3,
       11,  9,  1,  1, 10,  5,  1, 13, 10,  6,  9,  2,  3,  2, 10,  8,  8,
        3,  3,  7,  3, 11,  1, 11, 11,  0,  9, 10,  8,  7,  0,  6, 10,  7,
        7,  1,  7, 13,  9,  0,  1,  7,  8,  6,  6,  0,  4,  8,  7,  1, 13,
       13,  9,  9,  7,  2,  1,  3, 12,  8, 13,  7,  3,  1, 13, 13,  9,  0,
       10,  7,  1,  9, 11,  3, 10,  3,  5, 11,  6, 11,  4, 12,  7,  8,  5,
        5,  6,  0, 13,  2, 13,  0, 13, 11,  3, 13,  2,  3,  6,  7,  2,  7,
        0,  3,  4,  9, 11,  2,  7,  5,  4,  7,  6,  6,  6,  7,  5,  0,  4,
        1,  5,  2, 10,  6,  5, 11, 10,  8,  2,  3, 12, 10,  2,  6, 12,  8,
        5,  4, 11,  0, 12,  8, 11,  2,  6, 10,  0, 12,  0,  7,  3,  1,  4,
        5,  0,  2,  0,  7, 11,  0,  8, 13,  6,  0,  3, 11,  5, 13, 13,  7,
       12,  7,  9,  7,  5,  7,  0,  8, 11,  5,  1, 10,  1,  3, 11,  3,  4,
       11,  4,  1,  6,  3,  5,  3,  8,  7, 11,  4,  0,  7,  6,  3,  4, 10,
        2,  0,  0,  1, 13,  3, 11,  5,  7,  8,  5,  8,  6, 13,  7,  7,  0,
       10,  2, 12,  8, 13,  9, 13,  2,  1,  6,  7,  5,  7,  3,  7, 13,  1,
        0,  8,  0, 12,  4,  8,  9,  5,  5, 12,  7,  9,  5,  1,  9,  1,  8,
        8,  5,  7, 12,  1,  9,  3,  4, 11,  1,  0,  3, 10,  9,  5,  3, 12,
        7,  5,  8,  9,  2,  8,  5,  0, 12,  1,  3,  0, 11,  5, 10,  9, 12,
        2, 10, 10,  3, 13, 12,  5, 13, 12,  0,  7,  2,  1, 11, 12, 11, 12,
        4,  3,  0,  0,  3, 10,  9, 12,  6,  9,  5, 10, 13,  2, 11,  3,  2,
        1,  2, 10, 10, 10, 13,  1,  4,  5, 10,  8,  4,  2,  7, 12,  9,  2,
        6,  8,  1, 11,  2, 11,  3,  5, 10,  3,  4,  9,  6, 12,  6,  8, 10,
        9,  6,  0,  6,  1,  7, 12])
```

```
accuracy_score(y_pred,y_test)
```

```
0.8
```

```
confusion_matrix(y_pred,y_test)
```

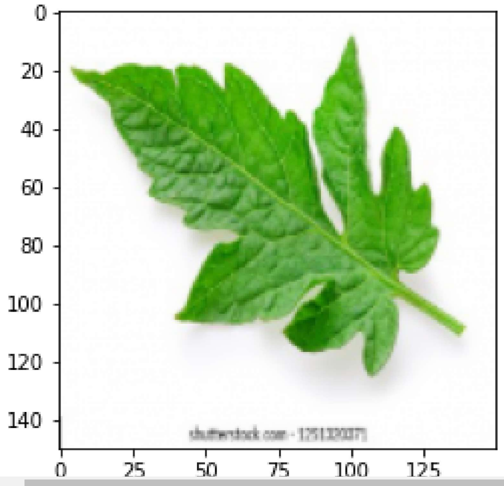
```
array([[60,  3,  0,  0,  1,  0,  0,  0,  0,  1,  0,  0,  5,  2],
       [ 3, 47,  3,  2,  0,  0,  0,  1,  0,  4,  0,  0,  1,  1],
       [ 1,  3, 54,  1,  3,  0,  0,  0,  0,  0,  0,  0,  2,  2],
       [ 7,  1,  1, 49, 12,  3,  2,  0,  0,  1,  0,  0,  1,  0],
       [ 2,  0,  0,  1, 40,  0,  1,  0,  0,  0,  0,  0,  1,  1],
       [ 0,  4,  0,  2,  1, 47,  0,  0,  0,  0,  0,  0,  3,  1],
       [ 0,  0,  0,  0,  8,  0, 46,  1,  2,  0,  0,  0,  0,  0],
       [ 0,  2,  0,  0,  2,  0,  0, 54, 23,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  1,  8, 47,  0,  0,  0,  1,  0],
       [ 0,  5,  2,  2,  0,  0,  0,  0,  0, 42,  1,  0,  0,  1],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 48,  0,  0,  0],
       [ 0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  1, 51,  0,  0],
       [ 4,  2,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0, 46,  4],
       [ 2,  0,  0,  1,  0,  3,  1,  0,  0,  0,  0,  0,  5, 41]])
```

```
#saving the model using pickle
import joblib
joblib.dump(model,'leaf disease detection')
```

```
['leaf disease detection']
```

```
#Testing random image
flat_data = []
url = input('Enter the url of image:')
img = imread(url)
img_resized = resize(img,(150,150,3))
flat_data.append(img_resized.flatten())
flat_data = np.array(flat_data)
print(img.shape)
plt.imshow(img_resized)
y_out = model.predict(flat_data)
y_out = CATEGORIES[y_out[0]]
print(f'Predicted output:{y_out}')
```

```
Enter the url of image:https://image.shutterstock.com/image-photo/tomato-leaves-isolatec
(280, 375, 3)
Predicted output:Tomato___healthy
```



Double-click (or enter) to edit

```
svc_classifier = SVC(kernel='linear')
svc_classifier.fit(x_train, y_train)
```

```
y_pred2 = svc_classifier.predict(x_test)
y_pred2
```

```
array([10., 11.,  9., ...,  9.,  6., 10.])
```

```
accuracy_score(y_pred2,y_test)
```

```
0.7171428571428572
```

✓ 12s completed at 9:13 PM