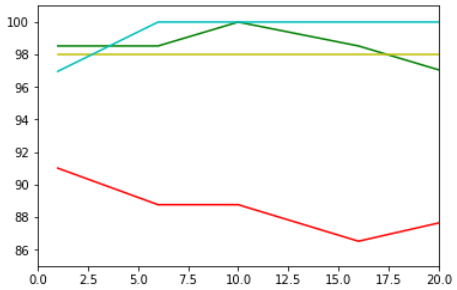
Q1 a) **k-NN on Iris Data set with varying values of K and split.**

```
def main(split, arrayForNearestNeighbours):
    # prepare data
    trainingSet=[]
    testSet=[]
    loadDataset('D:\\CSNL\\Text_Analysis\\Assignment6\\xLect6.Progs\\iris.csv', split, trainingSet, testSet)
    accuracyForK =[]
    for k in arrayForNearestNeighbours:
        # generate predictions
        predictions=[]
        for x in range(len(testSet)):
            neighbors = getNeighbors(trainingSet, testSet[x], k)
            result = getResponse(neighbors)
            predictions.append(result)
        accuracyForK.append(getAccuracy(testSet, predictions))
    return accuracyForK

finalAccuracy = []
arrayForNearestNeighbours = [1,6, 10, 16,20]
for split in [0.33,0.50,0.67,0.80]:
    finalAccuracy.append(main(split,arrayForNearestNeighbours))

print(finalAccuracy)

import matplotlib.pyplot as plt
plt.plot(arrayForNearestNeighbours, finalAccuracy[0], 'r-')
plt.plot(arrayForNearestNeighbours, finalAccuracy[1], 'g-')
plt.plot(arrayForNearestNeighbours, finalAccuracy[2], 'y-')
plt.plot(arrayForNearestNeighbours, finalAccuracy[3], 'c-')
plt.axis([0, 20, 85, 101])
plt.show()
```



Plot of N vs accuracy.

Output : when K = 1,6,10,16,20.

Red line : For Split = 0.33
[91.01123595505618, 88.76404494382022, 88.76404494382022, 86.51685393258427, 87.64044943820225]
Green line : For Split = 0.50
[98.52941176470588, 98.52941176470588, 100.0, 98.52941176470588, 97.05882352941177]
Yellow line : For Split = 0.67
[98.0, 98.0, 98.0, 98.0, 98.0]

Cyan line : For Split = 0.80
[[96.96969696969697, 100.0, 100.0, 100.0, 100.0]

Inference:

1. When the split is 0.33 i.e. 33% we see the graph actually starts with a lower accuracy of 91% and even decreases further . This is because the training set is actually less and this model is not skilled to test the variance in our data.
2. When the split is 0.50 represented by the green line it shows little fluctuations in the data. It shows a good accuracy of 100% at k = 10 but further the accuracy goes on decreasing for higher values of K, as K increases the model becomes simple and accuracy decreases.
3. When the split is 0.80 that is training set contains almost 80% of the data and accuracy increases after k =10 and is 100% for all other values of k which follow . This tells us that the model has become more accurate .But the problem with such a model is that it would produce much inconsistent results for slight variations in our test dataset which is also known as over- fitting.

Q1 b)**K-fold Process:The K-fold process is a process which can help us find how our machine learning model will work on new and independent dataset.i.e when we do K-cross validation our each sample is used for training and testing both .**

```
split_array =[0.33,0.5,0.67,0.80]

k_range = [1,6,10,16,20]

K_values_score=[]
for k in k_range:
    k_for_each_split=[]
    for split_val in split_array:
        X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=split_va
        knn = KNeighborsClassifier(n_neighbors=k)
        scores = cross_val_score(knn, X_train, y_train, cv=5, scoring='accura
        k_for_each_split.append(round(scores.mean()*100,2))
    K_values_score.append(k_for_each_split)
print(K_values_score)
```

```
[[97.99, 98.67, 93.89, 78.86], [97.99, 97.24, 96.18, 93.33], [96.99, 96.07, 9
4.14, 89.81], [93.78, 93.46, 100.0, 79.14], [96.08, 94.46, 84.32, 63.33]]
```

```
avg_scores_dict={}
for i in range(len(K_values_score)):
    temp=0
    for j in range(len(K_values_score[i])):
        temp+=K_values_score[i][j]
    avg_scores_dict.update({"The Average value of n-fold for knn = "+str(k_range[i])+" is ":temp/4})
```

```
for i in avg_scores_dict:
    print(i,avg_scores_dict[i])
```

```
The Average value of n-fold for knn = 1 is  92.3525
The Average value of n-fold for knn = 6 is  96.18499999999999
The Average value of n-fold for knn = 10 is  94.2525
The Average value of n-fold for knn = 16 is  91.595
The Average value of n-fold for knn = 20 is  84.5475
```

Code Snippet to perform 5-fold validation on varying splits

Steps followed for K-cross Validation

1) For Iris data the total length of our dataset is 150,so now we perform 5-fold cross validation process.
2) The 150 records are divided into 5 -chunks, each of 30 records
3) Now suppose if we use 1st 30 records as our Testing set the union of the rest 4parts i.e. in all 120 records will act as our Training dataset.
4) This process is repeated 5 times for each testing set in our records and also at varying values of split.

Observations :

we can infer that the best value of k =6, as it gives us the highest accuracy of 96.18 and should be selected.

References :1) https://machinelearningmastery.com/k-fold-cross-validation/
2)https://www.ritchieng.com/machine-learning-cross-validation/

Q2) **Naïve Bayes : Naïve Bayes classifier considers that all its features are independent of one another and applies Bayes Theorem to get the output.**

Observations for feature 1:

```
def gender_features(word):
    return {'last_letter': word[-1]}
```

Considering the last charcter.

When the last character was selected for classification the classifier gave the accuracy of 76%.

```
In [170]: runfile('D:/CSNL/Text_Analysis/Assignment6/Bayes.py', wdir='D:/CSNL/
Text_Analysis/Assignment6')
Brenitta is: female
Emmy is: female
Alex is: male
Rose is: female
Brian is: male
Most Informative Features
            last_letter = 'k'              male : female =     45.6 : 1.0
            last_letter = 'a'            female : male   =     34.9 : 1.0
            last_letter = 'f'              male : female =     15.9 : 1.0
            last_letter = 'p'              male : female =     11.1 : 1.0
            last_letter = 'v'              male : female =     10.5 : 1.0
0.76
```

Also the actual observed accuracy in the case is 100% for chosen inputs as all samples are classified accurately.

Observations for feature 2:

```
def gender_features(word):
    #return {'last_letter': word[-1]}
    #return {'last_two_letters': word[-2:]}
    return {'last_and_first': word[-1]+word[0]}
```

```
In [165]: runfile('D:/CSNL/Text_Analysis/Assignment6/Bayes.py', wdir='D:/CSNL/
Text_Analysis/Assignment6')
Brenitta is: female
Emmy is: female
Alex is: female
Rose is: female
Brian is: male
Most Informative Features
         last_and_first = 'aC'           female : male   =     53.6 : 1.0
         last_and_first = 'aR'           female : male   =     38.2 : 1.0
         last_and_first = 'aA'           female : male   =     30.6 : 1.0
         last_and_first = 'aM'           female : male   =     27.1 : 1.0
         last_and_first = 'oA'             male : female =     26.9 : 1.0
0.738
```

Considering the first and the last charcter.

After modifying our model for considering the last and the first letter the accuracy drops to 73.8%. Alex though a male is classified as a female.
This tells us that the model is not good , for our selected features.

Observations for feature 3:

```
def gender_features(word):
    #return {'last_letter': word[-1]}
    return {'last_two_letters': word[-2:]}
```

```
In [167]: runfile('D:/CSNL/Text_Analysis/Assignment6/Bayes.py', wdir='D:/CSNL/
Text_Analysis/Assignment6')
Brenitta is: female
Emmy is: female
Alex is: male
Rose is: female
Brian is: male
Most Informative Features
        last_two_letters = 'na'           female : male   =    165.6 : 1.0
        last_two_letters = 'la'           female : male   =     75.3 : 1.0
        last_two_letters = 'ia'           female : male   =     55.3 : 1.0
        last_two_letters = 'sa'           female : male   =     34.3 : 1.0
        last_two_letters = 'ta'           female : male   =     32.9 : 1.0
0.772
```

Considering the last two charcters.
When last two letters are selected for classification the accuracy is highest that is its 77.2 % and the observed accuracy for selected input is 100%.

Inference :

The naïve Bayes Algorithm classifies the input name as male or female by using probablistic method.

The instance of considering the last two characters gives us the highest accuracy and at the same time the feature of last word gives a slightly less classifier accuracy than the last two letters feature. i.e. as features are increasing the accuracy is also increasing.

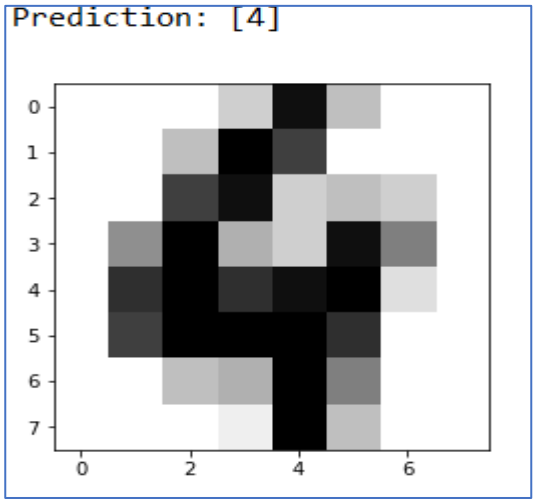
The first and the last letter feature however has a very low accuarcy and is not a good feature for our model although the features have increased .Example : Alex is classified as Female in this model although it should have been 'male',this may have been the case as there have been more instances In our training set where A is 1st letter and X is last letter have more females.

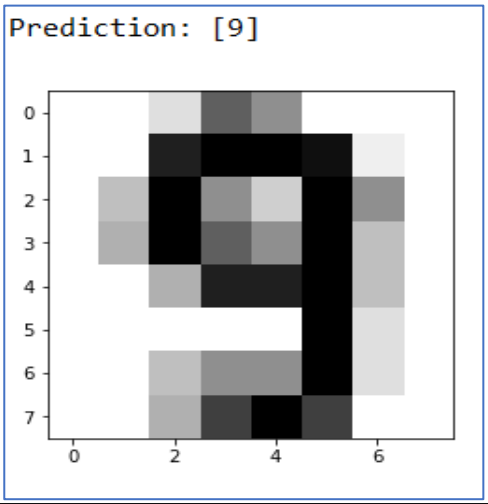
Q3) SVM : The support vector machine algorithm categorizes new datapoints in a specific hyperplane by comparing it with the given labeled data using the supervised learning approach.

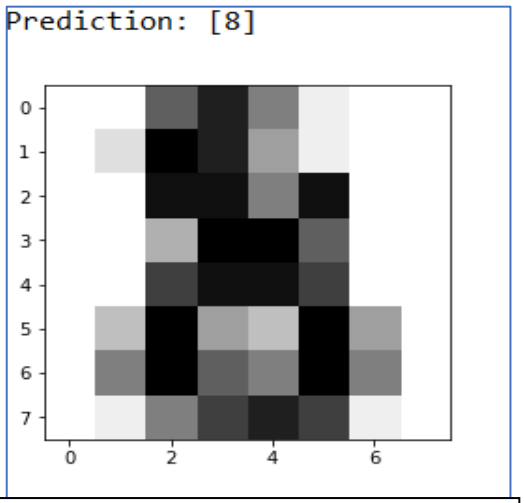
```
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn import svm
digits = datasets.load_digits()
clf = svm.SVC(gamma=0.01,C=100)
x,y= digits.data[:-15],digits.target[:-15]
clf.fit(x,y)
clf.predict(digits.data[-6].reshape(1,-1))
plt.imshow(digits.images[-6], cmap=plt.cm.gray_r, interpolation='nearest')
plt.show()
```

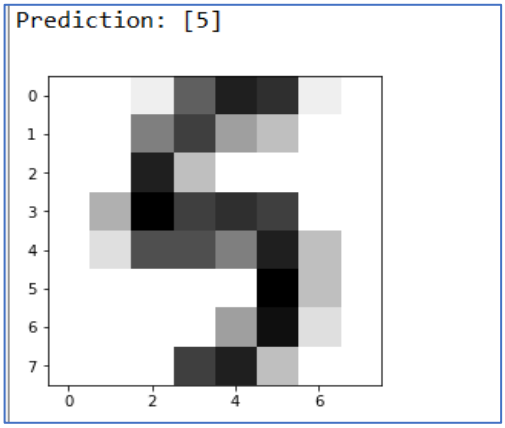
For gamma = 0.01 :

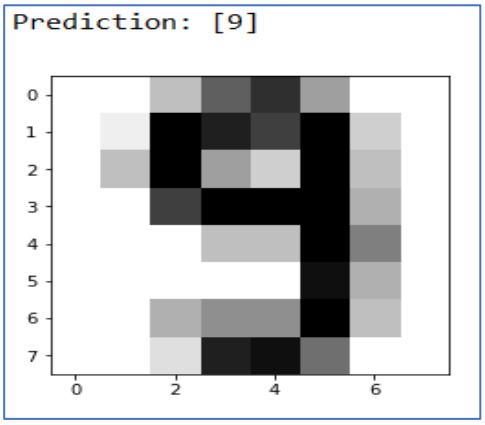






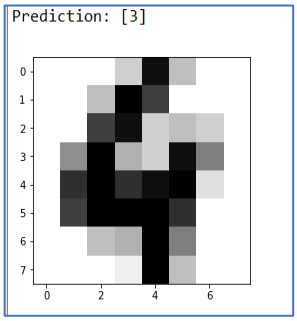
D = -6 , image = 4 D =-2 , image = 9 D = -1 ,image = 8

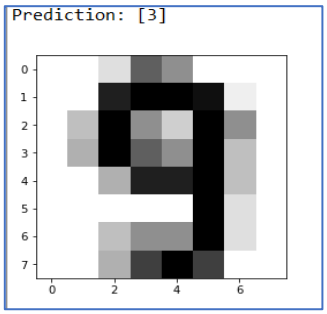


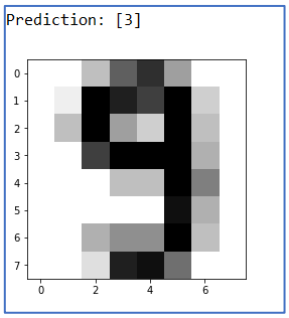


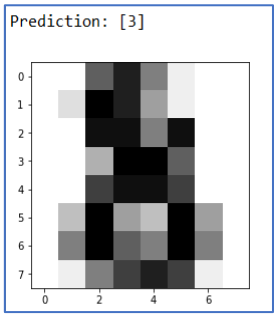
D = -13 , image = 5 D =-5 , image = 9

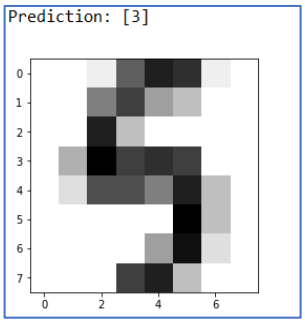
For gamma = 0.2 :





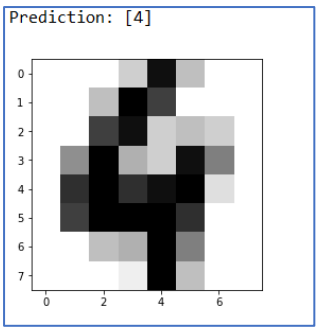


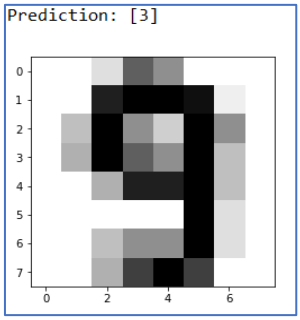


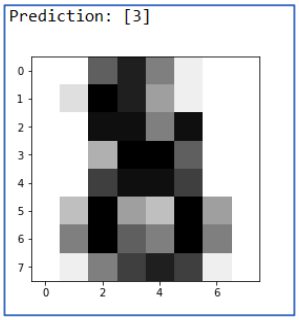


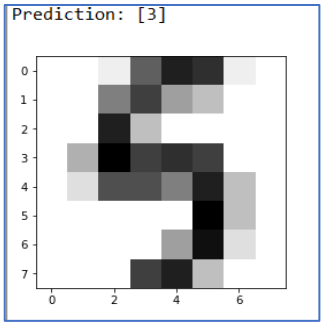
D = -6 , image = 4	D =-2 , image = 9	D = -1 ,image = 9	D = -13 , image = 8	D =-5 , image = 5

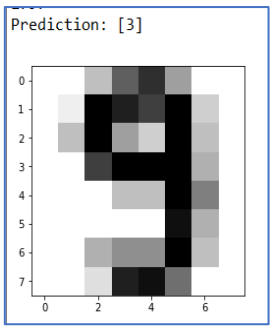
For gamma = 0.02 :











D = -6 , image = 4	D =-2 , image = 9	D = -1 ,image = 8	D = -13 , image = 5	D =-5 , image = 9

Observations :

Gamma	Accuracy
0.01	100%
0.2	0%
0.02	20%

The value of gamma at 0.01 gives us the maximum accuracy .
The accuracy drops when we scale up the value of gamma to 0.2 and almost gives 0 % accuracy.
The value of accuracy decreases as gamma increases, this can be evidently seen from third assumption of 0.02 .i.e a slight increase in the value decreases our accuracy by 80 %.

Inference :

We can infer that the space vector model doesn't give correct predictions for higher values of gamma.
This happens because , If we increase the value of gamma the distant points will also tend to show similarity when they actually are dis similar.
So accurate boundaries cannot be made which can help us differentiate the data points.
Reference : https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/