In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1.

In [2]:

```python
df = pd.read_excel("Raisin_Dataset.xlsx")
```

In [3]:

```python
df
```

Out[3]:

	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	Extent	Perimeter	Class
0	87524	442.246011	253.291155	0.819738	90546	0.758651	1184.040	Kecimen
1	75166	406.690687	243.032436	0.801805	78789	0.684130	1121.786	Kecimen
2	90856	442.267048	266.328318	0.798354	93717	0.637613	1208.575	Kecimen
3	45928	286.540559	208.760042	0.684989	47336	0.699599	844.162	Kecimen
4	79408	352.190770	290.827533	0.564011	81463	0.792772	1073.251	Kecimen
...	...	...	...	...	...	...	...	...
895	83248	430.077308	247.838695	0.817263	85839	0.668793	1129.072	Besni
896	87350	440.735698	259.293149	0.808629	90899	0.636476	1214.252	Besni
897	99657	431.706981	298.837323	0.721684	106264	0.741099	1292.828	Besni
898	93523	476.344094	254.176054	0.845739	97653	0.658798	1258.548	Besni
899	85609	512.081774	215.271976	0.907345	89197	0.632020	1272.862	Besni

900 rows × 8 columns

A.

In [4]:

```python
from sklearn.model_selection import train_test_split

X = df.drop('Class', axis = 1)
y = df['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

In [5]:

```python
print(X_train)
```

```
       Area  MajorAxisLength  MinorAxisLength  Eccentricity  ConvexArea  \
304   61959       339.585018       234.697936      0.722729       64424   
831   48488       275.337717       226.802199      0.566991       50103   
240   51145       317.348237       211.025592      0.746874       53531   
1     75166       406.690687       243.032436      0.801805       78789   
439   98485       387.762836       325.737166      0.542521      100682   
..      ...              ...              ...           ...         ...   
835   87039       497.058056       236.212773      0.879866       96247   
192   37569       232.427848       208.152006      0.444950       38874   
629   83059       440.559971       243.636512      0.833171       88806   
559  139399       522.468010       348.282162      0.745407      144034   
684   88257       452.924593       254.482193      0.827229       92886   

       Extent  Perimeter  
304  0.712041    970.754  
831  0.669594    859.326  
240  0.710051    895.040  
1    0.684130   1121.786  
439  0.729865   1193.908  
..        ...        ...  
835  0.636981   1271.343  
192  0.794371    734.102  
629  0.685809   1238.163  
559  0.693631   1471.508  
684  0.736088   1209.622  

[720 rows x 7 columns]
```

In [6]:

```python
print(y_train)
```

```
304    Kecimen
831      Besni
240    Kecimen
1      Kecimen
439    Kecimen
        ...   
835      Besni
192    Kecimen
629      Besni
559      Besni
684      Besni
Name: Class, Length: 720, dtype: object
```

In [7]:

```python
print(X_test)
```

```
       Area  MajorAxisLength  MinorAxisLength  Eccentricity  ConvexArea  \
492  100443       457.569872       283.611280      0.784743      104186   
141   53698       348.223231       197.752897      0.823104       56089   
409  104921       452.863013       297.024187      0.754864      108211   
31    41809       307.532739       175.085568      0.822114       43838   
570  101223       496.533461       276.866203      0.830111      109593   
..      ...              ...              ...           ...         ...   
408   82793       428.116905       249.390880      0.812809       84950   
75    50570       311.344795       208.317898      0.743181       51743   
778  148073       599.362814       327.084068      0.837968      152985   
202   74763       391.227169       245.542020      0.778520       77508   
181   67718       373.924098       233.233205      0.781628       71102   

       Extent  Perimeter  
492  0.673789   1269.066  
141  0.731341    938.705  
409  0.726298   1254.861  
31   0.697444    828.697  
570  0.713612   1345.426  
..        ...        ...  
408  0.742458   1130.673  
75   0.720863    873.777  
778  0.690156   1557.606  
202  0.686113   1063.377  
181  0.736994   1024.993  

[180 rows x 7 columns]
```

In [8]:

```python
print(y_test)
```

```
492      Besni
141    Kecimen
409    Kecimen
31     Kecimen
570      Besni
        ...   
408    Kecimen
75     Kecimen
778      Besni
202    Kecimen
181    Kecimen
Name: Class, Length: 180, dtype: object
```

In [9]:

```python
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

B.

In [10]:

```python
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
```

Out[10]:

```
▾                SVC
SVC(kernel='linear', random_state=0)
```

In [11]:

```python
print(X_train)
```

```
[[-0.6548812  -0.78812672 -0.38730055 ... -0.65224802  0.23753637
  -0.71404437]
 [-1.00293785 -1.35563187 -0.54502864 ... -1.00956114 -0.58197907
  -1.13149817]
 [-0.93428768 -0.98454723 -0.86018782 ... -0.92403154  0.19911415
  -0.99769926]
 ...
 [-0.10971045  0.10379878 -0.20874035 ... -0.04391002 -0.26892835
   0.28777656]
 [ 1.34597299  0.82730363  1.88169876 ...  1.33404467 -0.11791121
   1.16198015]
 [ 0.02459276  0.21301716  0.00791686 ...  0.05788717  0.70180368
   0.18085058]]
```

In [12]:

```python
print(X_test)
```

```
[[ 0.33944825  0.25404955  0.58980998 ...  0.33982546 -0.50100234
   0.40355155]
 [-0.8683246  -0.7118242  -1.12532793 ... -0.8602087   0.61015172
  -0.83411273]
 [ 0.45514847  0.21247322  0.85775103 ...  0.44025038  0.51278193
   0.35033394]
 ...
 [ 1.57008726  1.5065259   1.45823802 ...  1.55737473 -0.18499717
   1.48453765]
 [-0.32405815 -0.33196457 -0.17067523 ... -0.32579841 -0.263055
  -0.36704161]
 [-0.50608318 -0.48480495 -0.41656055 ... -0.48562997  0.71929679
  -0.51084341]]
```

In [13]:

```python
y_pred = classifier.predict(X_test)
print(y_pred)
```

```
['Besni' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni'
 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Besni'
 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Besni'
 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni'
 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Besni'
 'Besni' 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Kecimen'
 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Besni'
 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni'
 'Besni' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Besni' 'Besni' 'Besni' 'Besni' 'Besni' 'Besni'
 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Besni'
 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Kecimen'
 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni'
 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Besni'
 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen'
 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen'
 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni'
 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen'
 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen']
```

In [14]:

```python
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("Accuracy score:", accuracy_score(y_test, y_pred))
a1 = accuracy_score(y_test, y_pred)
```

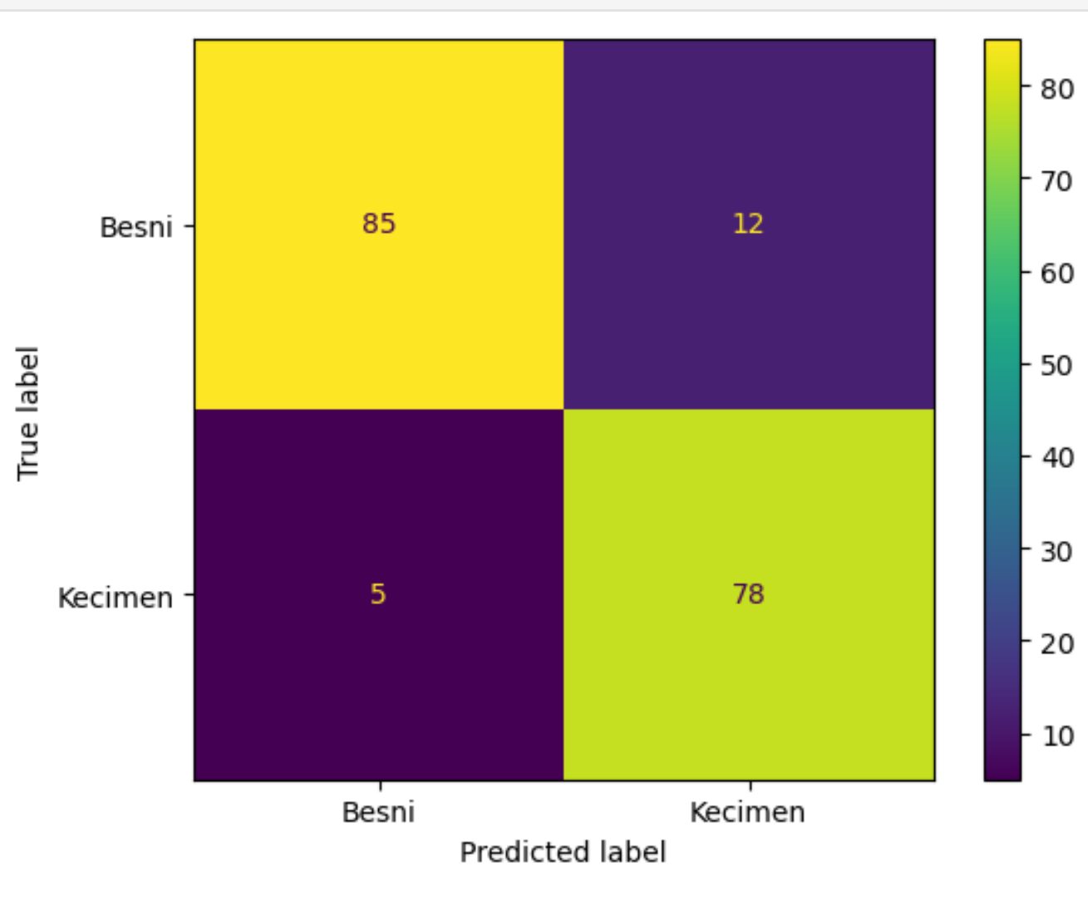
```
[[85 12]
 [ 5 78]]
Accuracy score: 0.9055555555555556
```

In [15]:

```python
from sklearn.metrics import ConfusionMatrixDisplay

disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = classifier.classes_)
disp.plot()
plt.show()

# Resource used:
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html
```

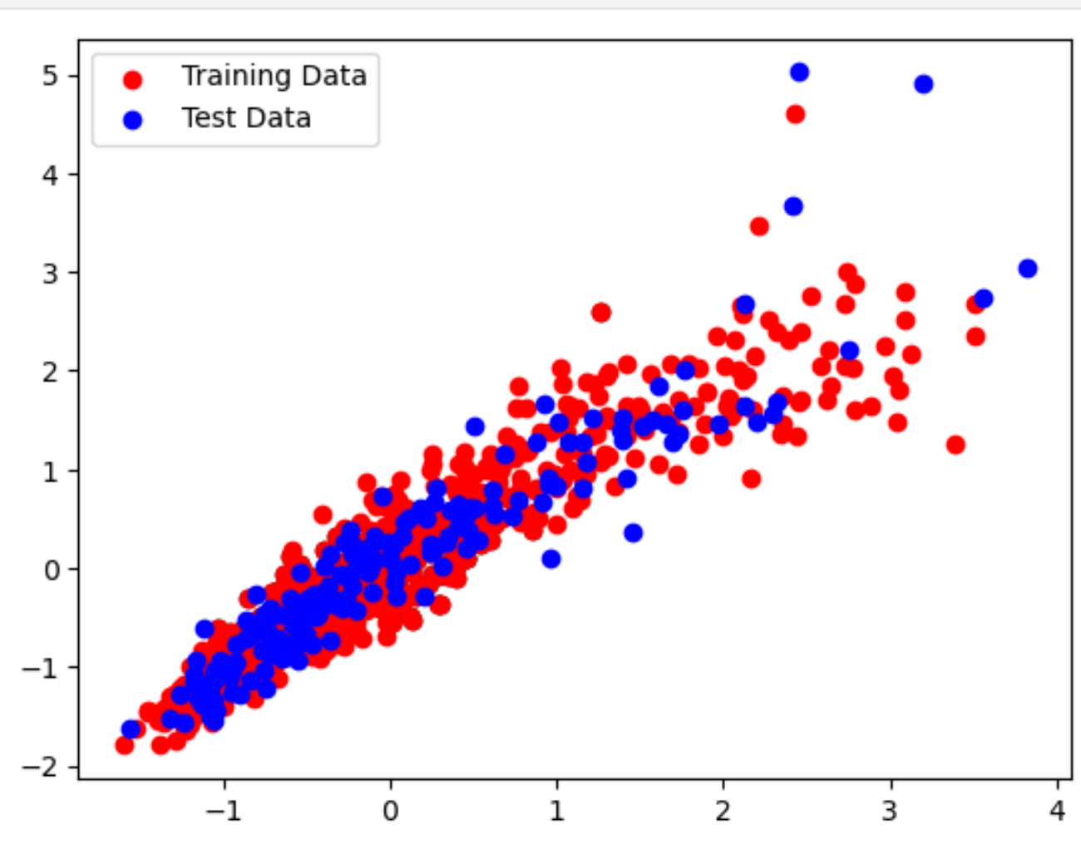


In [16]:

```python
support_vectors = classifier.support_vectors_

plt.scatter(support_vectors[:, 0], support_vectors[:, 1], c = 'black')
plt.scatter(X_train[:, 0], X_train[:, 1], c = 'red', label = 'Training Data')
plt.scatter(X_test[:, 0], X_test[:, 1], c = 'blue', label = 'Test Data')
plt.legend()
plt.show()

# Resources used:
# https://scikit-learn.org/stable/modules/svm.html
# https://scikit-learn.org/stable/auto_examples/svm/plot_linearsvc_support_vectors.html
```



C.

In [17]:

```python
from sklearn.model_selection import cross_val_score

accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 85.69 %
Standard Deviation: 5.80 %
```

Increasing the k can impact the accuracy of your SVM model. Values of around 5 and 10 are said to be the least variance and bias, per this module. That is why we used the value 10 for the cv in the code above. The larger the k, the smaller the variance. This means that is has a low bias when it comes to the test error. Despite this, a larger k increases bias. The largest possible value of k is the number of training points. A larger k would mean more distant neighbors and more training on less data.

Resources: https://hal.science/hal-01657491/document

D.

i.

In [18]:

```python
X = df.drop('Class', axis = 1)
y = df['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.5, random_state = 0)
```

In [19]:

```python
print(X_train)
```

```
       Area  MajorAxisLength  MinorAxisLength  Eccentricity  ConvexArea  \
679  128442       585.981994       281.601409      0.876960      133704   
133   52051       313.703260       213.631738      0.732284       53550   
571   79492       422.567329       243.116296      0.817920       82708   
223   93559       438.634353       275.582973      0.777991       95750   
816   66315       363.578896       234.454452      0.764308       68906   
..      ...              ...              ...           ...         ...   
835   87039       497.058056       236.212773      0.879866       96247   
192   37569       232.427848       208.152006      0.444950       38874   
629   83059       440.559971       243.636512      0.833171       88806   
559  139399       522.468010       348.282162      0.745407      144034   
684   88257       452.924593       254.482193      0.827229       92886   

       Extent  Perimeter  
679  0.750684   1499.355  
133  0.695051    893.451  
571  0.637353   1122.831  
223  0.687070   1198.259  
816  0.688944   1015.021  
..        ...        ...  
835  0.636981   1271.343  
192  0.794371    734.102  
629  0.685809   1238.163  
559  0.693631   1471.508  
684  0.736088   1209.622  

[450 rows x 7 columns]
```

In [20]:

```python
print(X_test)
```

```
       Area  MajorAxisLength  MinorAxisLength  Eccentricity  ConvexArea  \
492  100443       457.569872       283.611280      0.784743      104186   
141   53698       348.223231       197.752897      0.823104       56089   
409  104921       452.863013       297.024187      0.754864      108211   
31    41809       307.532739       175.085568      0.822114       43838   
570  101223       496.533461       276.866203      0.830111      109593   
..      ...              ...              ...           ...         ...   
392   51539       330.747487       200.957939      0.794253       53060   
801  164440       580.775848       362.345574      0.781505      167563   
524   56903       360.815292       204.037706      0.824755       59797   
614   62451       337.289987       237.705386      0.709455       64499   
107   68864       343.676209       261.988439      0.647210       71726   

       Extent  Perimeter  
492  0.673789   1269.066  
141  0.731341    938.705  
409  0.726298   1254.861  
31   0.697444    828.697  
570  0.713612   1345.426  
..        ...        ...  
392  0.725411    894.039  
801  0.770427   1554.365  
524  0.718427    980.390  
614  0.696275    963.009  
107  0.698347   1035.022  

[450 rows x 7 columns]
```

In [21]:

```python
print(y_train)
```

```
679      Besni
133    Kecimen
571      Besni
223    Kecimen
816      Besni
        ...   
835      Besni
192    Kecimen
629      Besni
559      Besni
684      Besni
Name: Class, Length: 450, dtype: object
```

In [22]:

```python
print(y_test)
```

```
492      Besni
141    Kecimen
409    Kecimen
31     Kecimen
570      Besni
        ...   
392    Kecimen
801      Besni
524      Besni
614      Besni
107    Kecimen
Name: Class, Length: 450, dtype: object
```

In [23]:

```python
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

ii.

In [24]:

```python
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
```

Out[24]:

```
▾                SVC
SVC(kernel='linear', random_state=0)
```

In [25]:

```python
y_pred = classifier.predict(X_test)
print(y_pred)
```

```
['Besni' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni'
 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Besni'
 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Besni'
 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni'
 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Besni'
 'Besni' 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Kecimen'
 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Besni'
 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni'
 'Besni' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Besni'
 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Besni'
 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Kecimen'
 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni'
 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Besni'
 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen'
 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen'
 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni'
 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen'
 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen'
 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni'
 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen'
 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Besni'
 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen'
 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Besni'
 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Besni' 'Besni'
 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Besni'
 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Besni'
 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen'
 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen'
 'Besni' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen'
 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen'
 'Besni' 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen'
 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Besni'
 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Besni'
 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni'
 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen'
 'Besni' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Besni'
 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni'
 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Besni'
 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Besni' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen'
 'Kecimen' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen'
 'Kecimen' 'Besni' 'Besni' 'Besni' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen'
 'Besni' 'Besni' 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni'
 'Besni' 'Kecimen' 'Kecimen' 'Kecimen' 'Besni' 'Kecimen' 'Kecimen'
 'Kecimen']
```

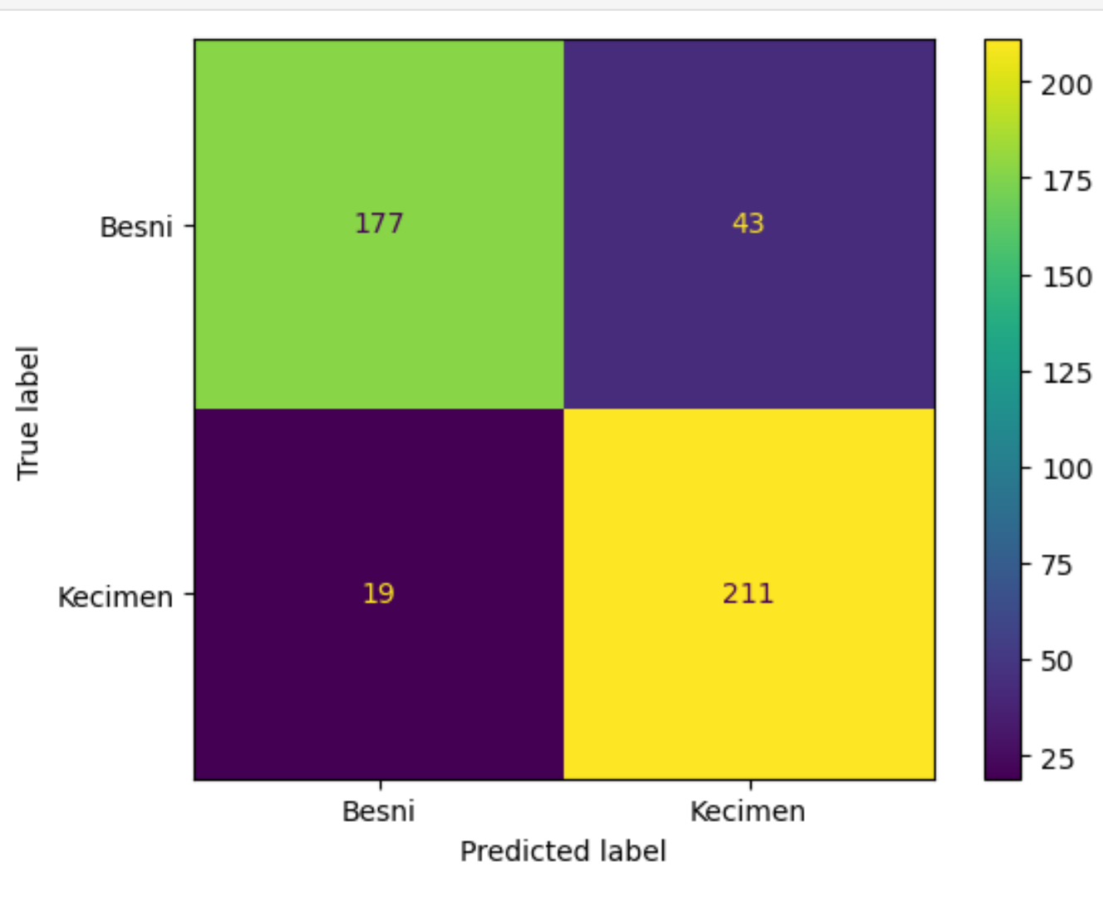
In [26]:

```python
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("Accuracy score:", accuracy_score(y_test, y_pred))
a2 = accuracy_score(y_test, y_pred)
```

```
[[177  43]
 [ 19 211]]
Accuracy score: 0.8622222222222222
```

In [27]:

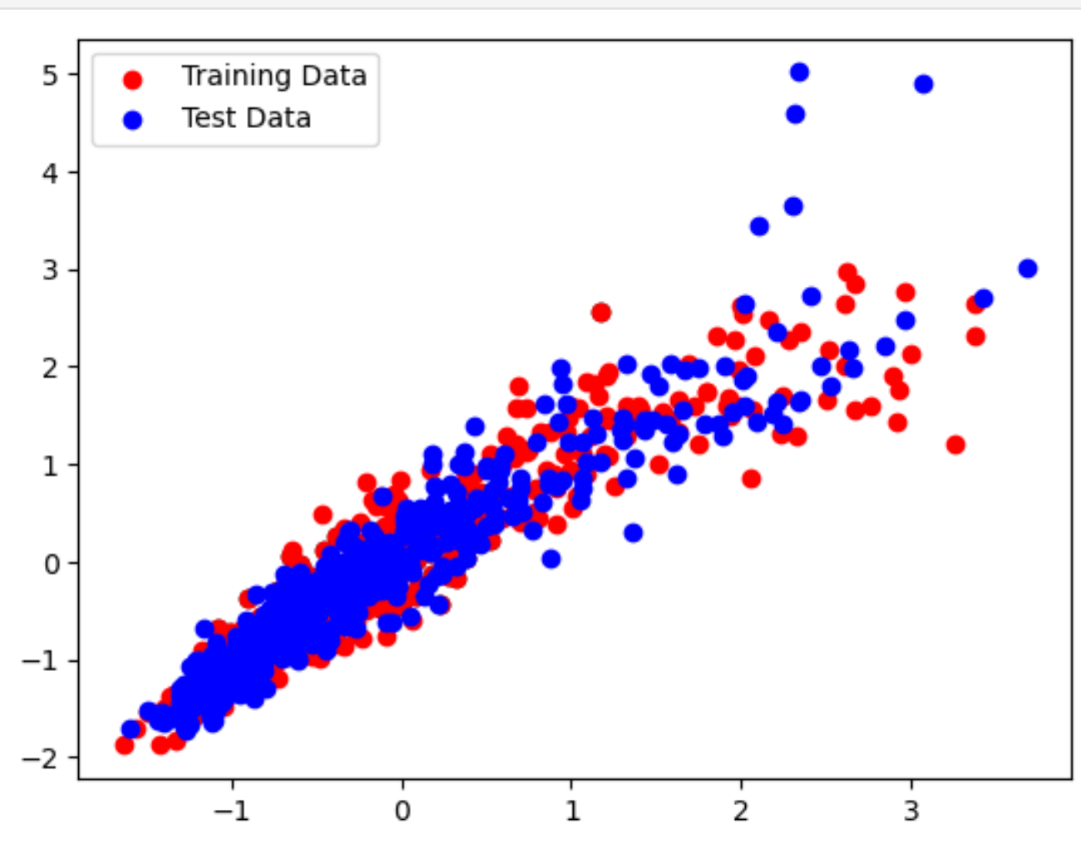
```python
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = classifier.classes_)
disp.plot()
plt.show()
```



In [28]:

```python
support_vectors = classifier.support_vectors_

plt.scatter(support_vectors[:, 0], support_vectors[:, 1], c = 'black')
plt.scatter(X_train[:, 0], X_train[:, 1], c = 'red', label = 'Training Data')
plt.scatter(X_test[:, 0], X_test[:, 1], c = 'blue', label = 'Test Data')
plt.legend()
plt.show()
```



iii.

In [29]:

```python
print("Accuracy score of 80-20: ", a1)
print("Accuracy score of 50-50:", a2)
```

```
Accuracy score of 80-20:  0.9055555555555556
Accuracy score of 50-50: 0.8622222222222222
```

The accuracy of the SVM model for the 80-20 ratio is higher than that of the 50-50 ratio. Based of off these results, we can conclude that the SVM model is more accurate when training on the larger portion of the data (or 80% here).