In [81]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVC
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import VotingRegressor
```

In [82]:
```
df=pd.read_csv('iris.csv')
df.head()
```

Out[82]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [83]:
```
df.isnull().sum()
```

Out[83]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [84]:
```
df_num=df.select_dtypes(['int','float'])
df_num
```

Out[84]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [85]:
```
df_cat=df.select_dtypes(object)
df_cat.head()
```

Out[85]:

	species
0	Iris-setosa
1	Iris-setosa
2	Iris-setosa
3	Iris-setosa
4	Iris-setosa

In [86]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in df_cat:
    df_cat[i]=le.fit_transform(df_cat[i])
```

In [87]:

```
df=pd.concat([df_num,df_cat],axis=1)
df
```

Out[87]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

In [88]:

```
le.classes_
```

Out[88]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [89]:

```
x=df[['sepal_length','sepal_width','petal_length','species']]
x
```

Out[89]:

	sepal_length	sepal_width	petal_length	species
0	5.1	3.5	1.4	0
1	4.9	3.0	1.4	0
2	4.7	3.2	1.3	0
3	4.6	3.1	1.5	0
4	5.0	3.6	1.4	0
...	...	...	...	...
145	6.7	3.0	5.2	2
146	6.3	2.5	5.0	2
147	6.5	3.0	5.2	2
148	6.2	3.4	5.4	2
149	5.9	3.0	5.1	2

150 rows × 4 columns

In [90]:

```
y=df[['petal_width']]
y
```

Out[90]:

	petal_width
0	0.2
1	0.2
2	0.2
3	0.2
4	0.2
...	...
145	2.3
146	1.9
147	2.0
148	2.3
149	1.8

150 rows × 1 columns

In [91]:
```
df.head()
```

Out[91]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [92]:
```
l=['sepal_length','sepal_width','petal_length','petal_width','species']
from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder(categories=[l])
```

In [93]:
```
oe=OrdinalEncoder(categories=[l])
#step1:import module.---------LabelEncoder

#step 2: initialize a class
le=OrdinalEncoder()
#step3:apply encoder
df_cat[['species']]=le.fit_transform(df_cat[['species']])
```

In [94]:
```
df_cat[['species']]
```

Out[94]:

	species
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0
...	...
145	2.0
146	2.0
147	2.0
148	2.0
149	2.0

150 rows × 1 columns

In [95]:
```
df.head()
```

Out[95]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [96]:
```
df['sepal_length']=df['sepal_length'].astype(int)
```

In [97]:
```
df['sepal_length']
```

Out[97]:
```
0      5
1      4
2      4
3      4
4      5
      ..
145    6
146    6
147    6
148    6
149    5
Name: sepal_length, Length: 150, dtype: int32
```

In [98]:
```
df.head()
```

Out[98]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5	3.5	1.4	0.2	0
1	4	3.0	1.4	0.2	0
2	4	3.2	1.3	0.2	0
3	4	3.1	1.5	0.2	0
4	5	3.6	1.4	0.2	0

In [99]:
```
df['sepal_width']=df['sepal_width'].astype(int)
df['petal_length']=df['petal_length'].astype(int)

df['petal_width']=df['petal_width'].astype(int)
```

In [100]:
```
df.head()
```

Out[100]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5	3	1	0	0
1	4	3	1	0	0
2	4	3	1	0	0
3	4	3	1	0	0
4	5	3	1	0	0

In [101]:
```
x=df[['sepal_length','sepal_width','petal_length','species']]
x
```

Out[101]:

	sepal_length	sepal_width	petal_length	species
0	5	3	1	0
1	4	3	1	0
2	4	3	1	0
3	4	3	1	0
4	5	3	1	0
...	...	...	...	...
145	6	3	5	2
146	6	2	5	2
147	6	3	5	2
148	6	3	5	2
149	5	3	5	2

150 rows × 4 columns

In [102]:
```
y=df[['petal_width']]
y
```

Out[102]:

	petal_width
0	0
1	0
2	0
3	0
4	0
...	...
145	2
146	1
147	2
148	2
149	1

150 rows × 1 columns

In [103]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.20,random_state=1)
```

In [104]:
```
def mod(m):
     m.fit(xtrain,ytrain)
     ypred=m.predict(xtest)
     #print(m,"---",r2_score(ytest,ypred))
     return r2_score(ytest,ypred)
```

In [105]:
```
lr=LinearRegression()
rf=RandomForestRegressor()
sv=SVC()
```

In [106]:
```
mod(lr)
```

Out[106]:
```
0.8697106654342677
```

In [107]:
```
mod(rf)
```

Out[107]:
```
0.8566317591317424
```

In [113]:
```
from sklearn.svm import SVC
svm=SVC()
svm.fit(xtrain,ytrain)
ypred=svm.predict(xtest)
```

In [109]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
pparameters={'C':[1,10],'gamma':[0.1,0.001],'kernel':['rbf']}
```

In [114]:
```
ypred
```

Out[114]:
```
array([0, 1, 1, 0, 2, 1, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 0, 2, 1, 0, 0, 1, 1])
```

In [115]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
accuracy_score(ytest,ypred)
```

Out[115]:
```
0.9666666666666667
```

In [116]:
```
confusion_matrix(ytest,ypred)
```

Out[116]:
```
array([[11,  0,  0],
       [ 0, 14,  1],
       [ 0,  0,  4]], dtype=int64)
```

In [117]:
```
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.93      0.97        15
           2       0.80      1.00      0.89         4

    accuracy                           0.97        30
   macro avg       0.93      0.98      0.95        30
weighted avg       0.97      0.97      0.97        30
```

In [118]:
```
parameters={'C':[1,10],'gamma':[0.1,0.001],'kernel':['rbf']}
```

In [119]:
```
from sklearn.model_selection import GridSearchCV
```

In [120]:

```
grid=GridSearchCV(SVC(),parameters,verbose=2)
grid.fit(xtrain,ytrain)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
[CV] END ........................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ........................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ........................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ........................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ........................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ......................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END ......................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END ......................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END ......................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END ......................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .....................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .....................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .....................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .....................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .....................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
```

Out[120]:

```
▸ GridSearchCV
▸ estimator: SVC
      ▸ SVC
```

In [121]:

```
grid.best_params_
```

Out[121]:

```
{'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
```

In [122]:

```
ypred=grid.predict(xtest)
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.93      0.97        15
           2       0.80      1.00      0.89         4

    accuracy                           0.97        30
   macro avg       0.93      0.98      0.95        30
weighted avg       0.97      0.97      0.97        30
```

In [123]:

```
l1=[]
c=["squared_error","friedman_mse","absolute_error","poisson"]
#hyper=[max_depth,min_samples_split,min_samples_leaf]
for j in c:
    l1=[]
    for i in range(1,100):
        #print(k)
        dt=RandomForestRegressor(n_estimators=10,criterion=j,max_depth=i)
        l1.append(mod(dt))
        print(max(l1),"----",1+l1.index(max(l1)))
```

```
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
0.8370083759711867 ---- 3
```

In [124]:

```
l1=[]
#hyper=[max_depth,min_samples_split,min_samples_leaf]
for j in c:
     l1=[]
     for i in range(2,100):
     #print(k)
         dt=RandomForestRegressor(n_estimators=10,criterion=j,min_samples_split=i)
         l1.append(mod(dt))
         print(max(l1),"----",2+l1.index(max(l1)))
```

```
0.8333704775667616 ---- 2
0.8527767920759326 ---- 3
0.8618610350389649 ---- 4
0.8730758582397592 ---- 5
0.8730758582397592 ---- 5
0.8848791701184673 ---- 7
0.8848791701184673 ---- 7
0.8848791701184673 ---- 7
0.9167638783836152 ---- 10
0.9167638783836152 ---- 10
0.9167638783836152 ---- 10
0.9167638783836152 ---- 10
0.9167638783836152 ---- 10
0.9167638783836152 ---- 10
0.9167638783836152 ---- 10
0.9167638783836152 ---- 10
0.9167638783836152 ---- 10
0.9167638783836152 ---- 10
0.9167638783836152 ---- 10
0.9167638783836152 ---- 10
```

In [ ]:

In [ ]: