iris

September 13, 2024

[211]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.metrics import (confusion_matrix , accuracy_score,␣
 ↪ConfusionMatrixDisplay ,classification_report ,
precision_score,recall_score)
from sklearn.ensemble import (AdaBoostClassifier ,BaggingClassifier␣
 ↪,GradientBoostingClassifier,
RandomForestClassifier)
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

[11]:
```
# load iris datasets
iris = pd.read_csv('IRIS.csv')
iris.head()
```

[11]:

	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

[17]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
```

```
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

[21]:
```
iris.isna().sum()
```

[21]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

[25]:
```
iris.describe(include='all')
```

[25]:

	sepal_length	sepal_width	petal_length	petal_width	species
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	50
mean	5.843333	3.054000	3.758667	1.198667	NaN
std	0.828066	0.433594	1.764420	0.763161	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

[27]:
```
iris.nunique()
```

[27]:
```
sepal_length    35
sepal_width     23
petal_length    43
petal_width     22
species          3
dtype: int64
```

[54]:
```
iris.iloc[:,:4].var()
```

[54]:
```
sepal_length    0.685694
sepal_width     0.188004
petal_length    3.113179
petal_width     0.582414
dtype: float64
```

1 Modeling

[48]:
```
iris.columns
```

[48]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

[57]:
```
# convert spicies colum to dummies by labelencoder
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
label.fit_transform(iris['species'])
```

[57]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

[59]:
```
# split our data into features and targets
X = iris.drop('species',axis=1)
y= iris['species']
```

[61]:
```
# train test split our data
X_train,X_test,y_train,y_test = train_test_split(X,y,stratify=y,test_size=0.
↪3,random_state=42)
```

[225]:
```
# SVC -> Model # 1
clf = SVC(C=10,kernel='rbf')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# Evaluation
print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(ConfusionMatrixDisplay(confusion_matrix(y_test,␣
↪y_pred),display_labels=clf.classes_).plot())
print(classification_report(y_test,y_pred))
print(recall_score(y_test,y_pred,average='micro'))
print(precision_score(y_test,y_pred,average='macro'))
```

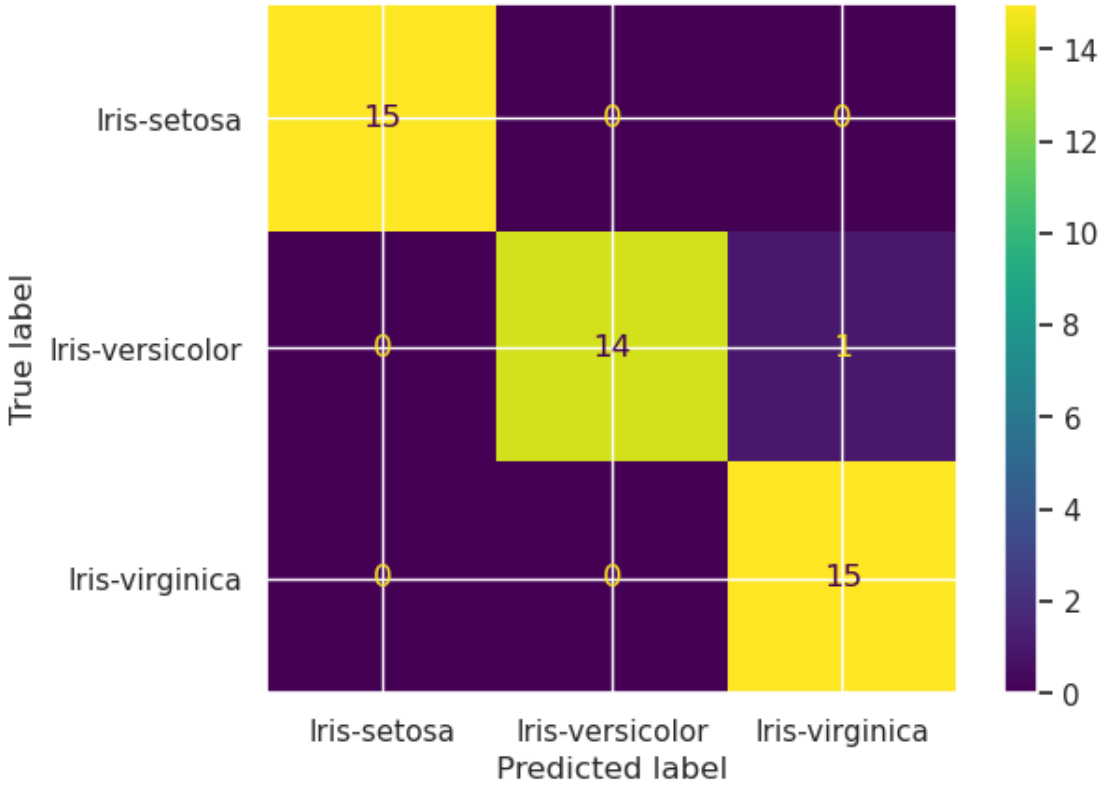
```
0.9777777777777777
[[15  0  0]
 [ 0 14  1]
 [ 0  0 15]]
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f84f14632e0>
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        15
Iris-versicolor       1.00      0.93      0.97        15
 Iris-virginica       0.94      1.00      0.97        15

       accuracy                           0.98        45
      macro avg       0.98      0.98      0.98        45
   weighted avg       0.98      0.98      0.98        45

0.9777777777777777
0.9791666666666666
```



[227]:
```
# KNN -> Model # 2
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

# Evaluation
print(accuracy_score(y_test, y_pred).round(2))
```

```
print()
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(ConfusionMatrixDisplay(confusion_matrix(y_test,␣
 ↪y_pred),display_labels=knn.classes_).plot())
print(classification_report(y_test,y_pred))
print(recall_score(y_test,y_pred,average='micro'))
print(precision_score(y_test,y_pred,average='macro'))
```

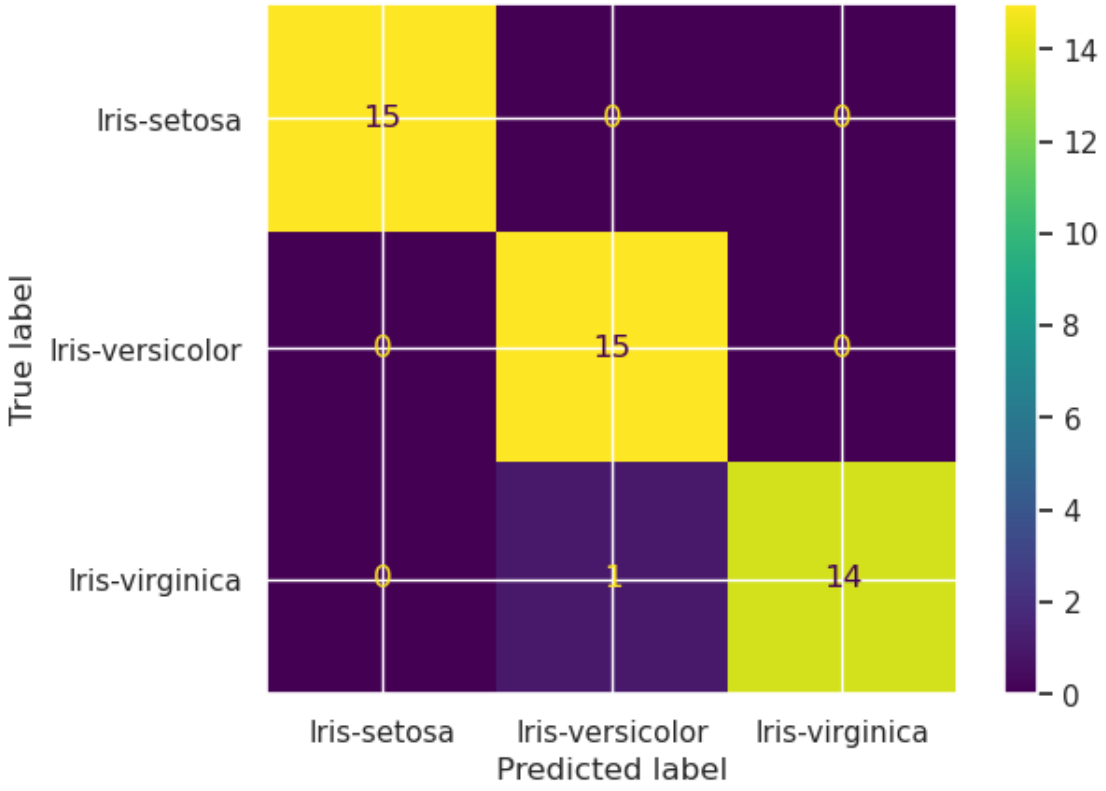
```
0.98

[[15  0  0]
 [ 0 15  0]
 [ 0  1 14]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f84f18f8670>
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        15
Iris-versicolor       0.94      1.00      0.97        15
 Iris-virginica       1.00      0.93      0.97        15

       accuracy                           0.98        45
      macro avg       0.98      0.98      0.98        45
   weighted avg       0.98      0.98      0.98        45

0.9777777777777777
0.9791666666666666
```



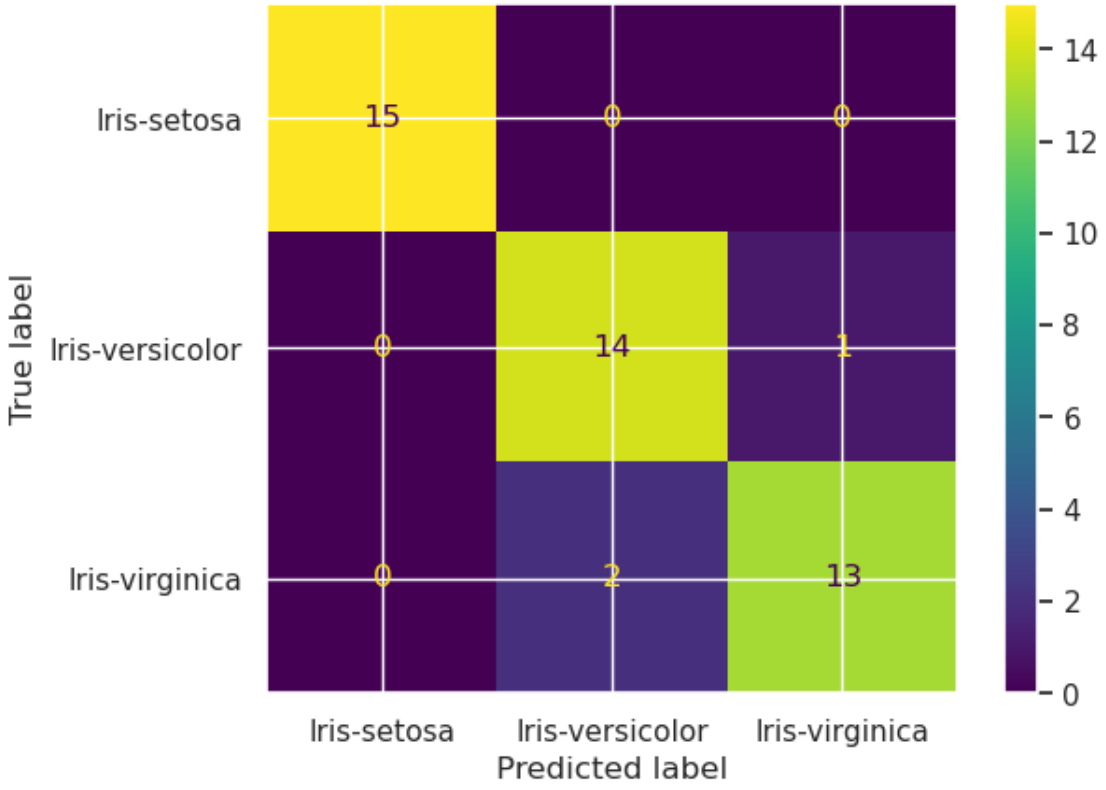
[229]:
```
# Logistic Regression -> Model # 3
log = LogisticRegression(max_iter=10000)
log.fit(X_train, y_train)
y_pred = log.predict(X_test)

# Evaluation
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
print()
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(ConfusionMatrixDisplay(confusion_matrix(y_test,␣
 ↪y_pred),display_labels=log.classes_).plot())
print(recall_score(y_test,y_pred,average='micro'))
print(precision_score(y_test,y_pred,average='macro'))
```

```
0.9333333333333333

[[15  0  0]
 [ 0 14  1]
 [ 0  2 13]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
```

```
0x7f84f18f8e50>
0.9333333333333333
0.9345238095238096
```

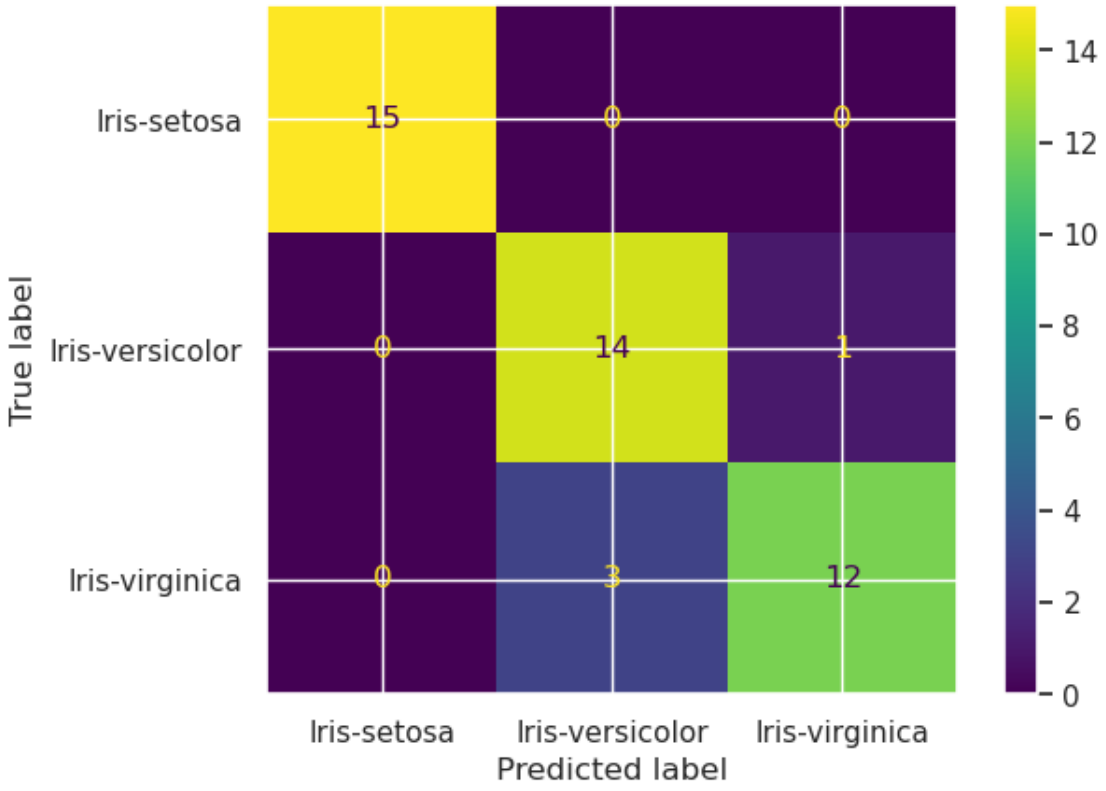


[231]:
```
# Random Forests -> Model 4
rfc = RandomForestClassifier(n_estimators=500,max_depth=50)
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)

# Evaluation
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
print()
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(ConfusionMatrixDisplay(confusion_matrix(y_test,␣
 ↪y_pred),display_labels=rfc.classes_).plot())
print(recall_score(y_test,y_pred,average='micro'))
print(precision_score(y_test,y_pred,average='macro'))
```

```
0.9111111111111111
```

```
[[15  0  0]
 [ 0 14  1]
 [ 0  3 12]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f84f2294940>
0.9111111111111111
0.9155354449472096
```

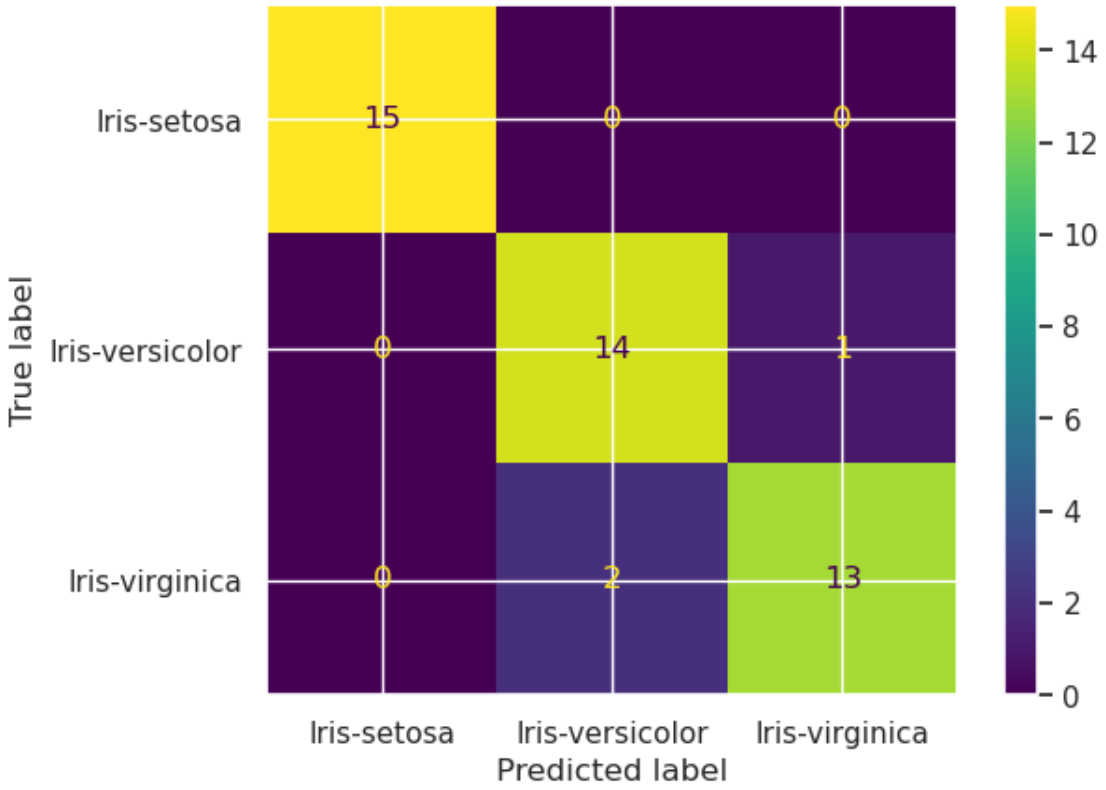


[233]:
```
#  GradientBoostingClassifier -> Model 5
grc = GradientBoostingClassifier()
grc.fit(X_train, y_train)
y_pred = grc.predict(X_test)

# Evaluation
print(accuracy_score(y_test, y_pred))
print()
print(confusion_matrix(y_test, y_pred))
print(ConfusionMatrixDisplay(confusion_matrix(y_test,␣
 ↪y_pred),display_labels=grc.classes_).plot())
print(recall_score(y_test,y_pred,average='micro'))
print(precision_score(y_test,y_pred,average='macro'))
```

```
0.9333333333333333

[[15  0  0]
 [ 0 14  1]
 [ 0  2 13]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f84f22955d0>
0.9333333333333333
0.9345238095238096
```



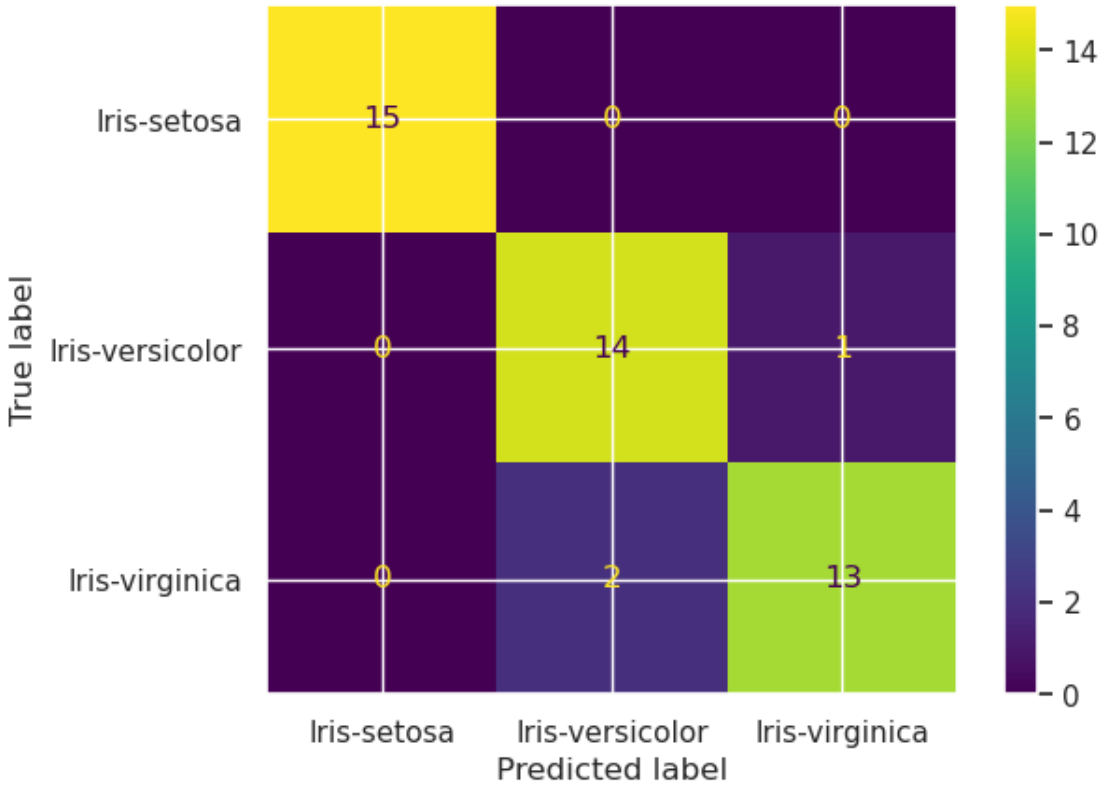
[235]:
```
#  Gradient Boosting Classifier -> Model 6
ada = AdaBoostClassifier()
ada.fit(X_train, y_train)
y_pred = ada.predict(X_test)

# Evaluation
print(accuracy_score(y_test, y_pred))
print()
print(confusion_matrix(y_test, y_pred))
print(ConfusionMatrixDisplay(confusion_matrix(y_test,␣
 ↪y_pred),display_labels=ada.classes_).plot())
print(recall_score(y_test,y_pred,average='micro'))
```

```
print(precision_score(y_test,y_pred,average='macro'))
```

```
0.9333333333333333

[[15  0  0]
 [ 0 14  1]
 [ 0  2 13]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f84f260da80>
0.9333333333333333
0.9345238095238096
```



[239]:
```
#  Gradient Boosting Classifier -> Model 7
bag = BaggingClassifier(n_estimators=100)
bag.fit(X_train, y_train)
y_pred = bag.predict(X_test)

# Evaluation
print(accuracy_score(y_test, y_pred))
print()
print(confusion_matrix(y_test, y_pred))
```

```
print(ConfusionMatrixDisplay(confusion_matrix(y_test,␣
  ↪y_pred),display_labels=bag.classes_).plot())
print(recall_score(y_test,y_pred,average='micro'))
print(precision_score(y_test,y_pred,average='macro'))
```

```
0.9333333333333333

[[15  0  0]
 [ 0 14  1]
 [ 0  2 13]]
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at
0x7f84f2570fa0>
0.9333333333333333
0.9345238095238096
```