ILLINOIS

Statistical Learning: HW2

due 2023-02-24

- Yu-Ching Liao ycliao3@illinois.edu

Basic Import

In [70]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

from sklearn.datasets import load_iris
from sklearn import preprocessing
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
```

In [71]:
```
def get_sensitivity_and_specificity(cm, i):
    TP = cm[i, i]
    FN = sum(cm[i])-TP
    FP = sum(cm[:,i])-TP
    TN = sum(sum(cm)) - TP - FN - FP
    sensitivity = TP/(TP+FN)
    specificity = TN/(TN+FP)
    return sensitivity, specificity
```

TP: The actual value and predicted value should be the same. So concerning Setosa class, the value of cell 1 is the TP value.

FN: The sum of values of corresponding rows except the TP value

FP : The sum of values of corresponding column except the TP value.

TN: The sum of values of all columns and row except the values of that class that we are calculating the values for.

Sensitivity = TP/(TP+FN)

Specificity = TN/(TN+FP)

Problem 2

Loading, Preparing and Visualizing Data

In [75]:
```
iris = load_iris()
X_iris, y_iris = iris.data, iris.target

print(X_iris.shape, y_iris.shape)
print(X_iris[0], y_iris[0])
```

```
(150, 4) (150,)
[5.1 3.5 1.4 0.2] 0
```

In [76]:
```
X, y = X_iris[:, :4], y_iris

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
print(X_train.shape, y_train.shape)
```

```
(112, 4) (112,)
```

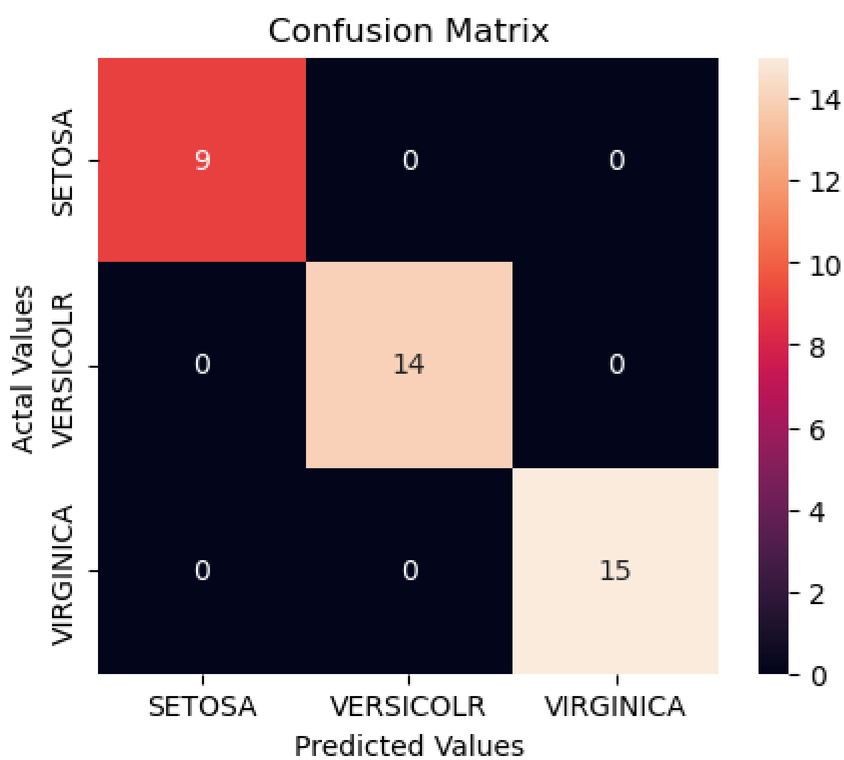
In [77]:
```
scaler = preprocessing.StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

LDA

In [82]:
```
clf = LDA()
clf.fit(X_train, y_train)

print("Coefficients: \n", clf.coef_, '\n')
print("Intercepts: \n", clf.intercept_, '\n')

y_train_pred = clf.predict(X_train)
print("Training Set Accuracy:", metrics.accuracy_score(y_train, y_train_prec

y_pred = clf.predict(X_test)
print("Testing Set Accuracy:", metrics.accuracy_score(y_test, y_pred), '\n')

print("Confusion Matrix:")
cm = metrics.confusion_matrix(y_test, y_pred)
cm_df = pd.DataFrame(cm,
                     index = ['SETOSA','VERSICOLR','VIRGINICA'],
                     columns = ['SETOSA','VERSICOLR','VIRGINICA'])
```

```
plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')
plt.show()

res = []
name = ['setosa', 'versicolor', 'virginica']

for l in [0, 1, 2]:
    res.append([name[l],get_sensitivity_and_specificity(cm,l)[0],get_sensiti

display(pd.DataFrame(res, columns=['Classes', 'Sensitivities', 'Specificitie

print('\n',"Additional Report:")
print(
    metrics.classification_report(y_test,
                                  y_pred,
                                  target_names=iris.target_names))
```

```
Coefficients:
 [[  6.27365149   4.9123596  -35.85726139 -12.16412295]
 [ -2.08999655  -2.13055863  12.81205311   2.24574484]
 [ -5.19942386  -3.56304665  28.82610872  11.93949219]]

Intercepts:
 [-29.02055384  -4.2409419  -21.22344971]

Training Set Accuracy: 0.9732142857142857
Testing Set Accuracy: 1.0

Confusion Matrix:
```



	Classes	Sensitivities	Specificities
0	setosa	1.0	1.0
1	versicolor	1.0	1.0
2	virginica	1.0	1.0

```
 Additional Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         9
  versicolor       1.00      1.00      1.00        14
   virginica       1.00      1.00      1.00        15

    accuracy                           1.00        38
   macro avg       1.00      1.00      1.00        38
weighted avg       1.00      1.00      1.00        38
```

Problem 3

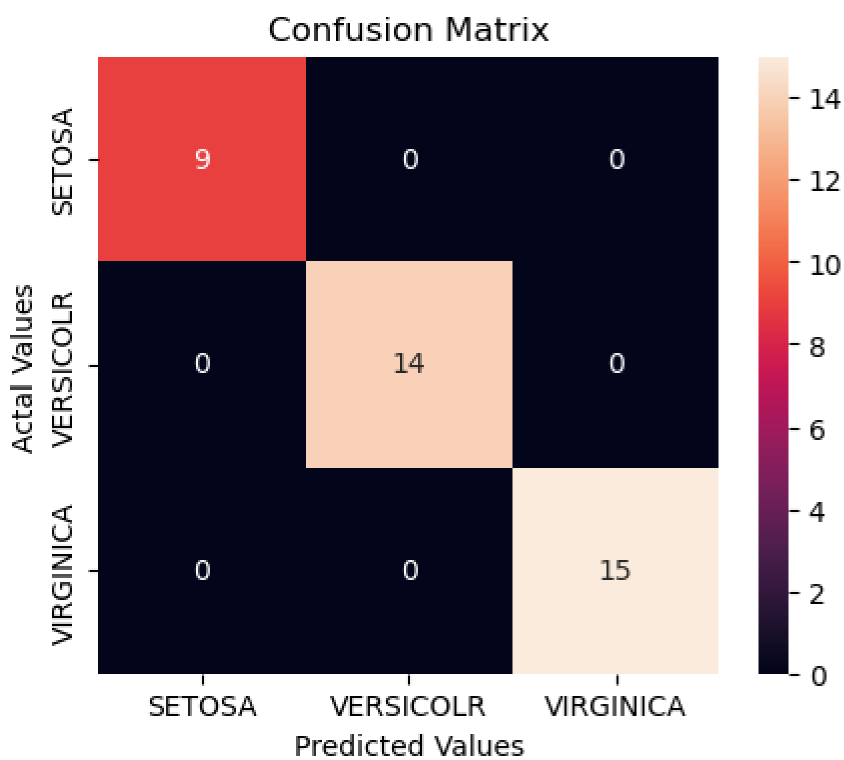
Logistic Regression

In [83]:

```
clf2 = LogisticRegression()
clf2.fit(X_train, y_train)

print("Coefficients: \n", clf2.coef_, '\n')
print("Intercepts: \n", clf2.intercept_, '\n')

y_train_pred = clf2.predict(X_train)
print("Training Set Accuracy:", metrics.accuracy_score(y_train, y_train_prec

y_pred = clf2.predict(X_test)
print("Testing Set Accuracy:", metrics.accuracy_score(y_test, y_pred), '\n')

print("Confusion Matrix:")
cm = metrics.confusion_matrix(y_test, y_pred)
cm_df = pd.DataFrame(cm,
                     index = ['SETOSA','VERSICOLR','VIRGINICA'],
                     columns = ['SETOSA','VERSICOLR','VIRGINICA'])
plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')
plt.show()

res = []
name = ['setosa', 'versicolor', 'virginica']

for l in [0, 1, 2]:
    res.append([name[l],get_sensitivity_and_specificity(cm,l)[0],get_sensiti
```

```
display(pd.DataFrame(res, columns=['Classes', 'Sensitivities', 'Specificitie

print('\n',"Additional Report:")
print(
    metrics.classification_report(y_test,
                                  y_pred,
                                  target_names=iris.target_names))
```

```
Coefficients:
 [[-1.02139469  1.04243172 -1.79730333 -1.64049312]
 [ 0.50596642 -0.31739578 -0.24296152 -0.6608599 ]
 [ 0.51542827 -0.72503594  2.04026485  2.30135303]]

Intercepts:
 [ 0.09726616  1.84192521 -1.93919137]

Training Set Accuracy: 0.9553571428571429
Testing Set Accuracy: 1.0

Confusion Matrix:
```



	Classes	Sensitivities	Specificities
0	setosa	1.0	1.0
1	versicolor	1.0	1.0
2	virginica	1.0	1.0

```
 Additional Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         9
  versicolor       1.00      1.00      1.00        14
   virginica       1.00      1.00      1.00        15

    accuracy                           1.00        38
   macro avg       1.00      1.00      1.00        38
weighted avg       1.00      1.00      1.00        38
```

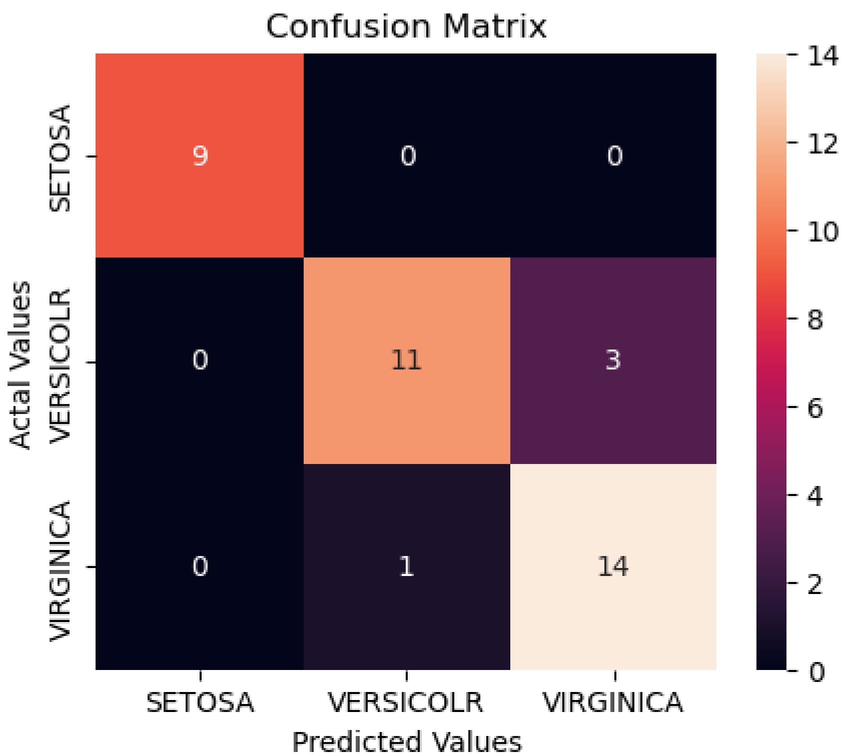
From the confusion matrix and other outcome, I would say LDA and Logistic Regression are both working well, though the training accuracy of LDA is still slightly better.

Logistic Regression without Intercept

In [85]:

```
clf3 = LogisticRegression(fit_intercept=False)
clf3.fit(X_train, y_train)

print("Coefficients: \n", clf3.coef_, '\n')
print("Intercepts: \n", clf3.intercept_, '\n')

y_train_pred = clf3.predict(X_train)
print("Training Set Accuracy:", metrics.accuracy_score(y_train, y_train_pred
y_pred = clf3.predict(X_test)
print("Testing Set Accuracy:", metrics.accuracy_score(y_test, y_pred) ,'\n')

print("Confusion Matrix:")
cm = metrics.confusion_matrix(y_test, y_pred)
cm_df = pd.DataFrame(cm,
                     index = ['SETOSA','VERSICOLR','VIRGINICA'],
                     columns = ['SETOSA','VERSICOLR','VIRGINICA'])
plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')
plt.show()

res = []
name = ['setosa', 'versicolor', 'virginica']

for l in [0, 1, 2]:
    res.append([name[l],get_sensitivity_and_specificity(cm,l)[0],get_sensiti

display(pd.DataFrame(res, columns=['Classes', 'Sensitivities', 'Specificitie

print('\n',"Additional Report:")
print(
    metrics.classification_report(y_test,
                                  y_pred,
                                  target_names=iris.target_names))
```

```
Coefficients:
 [[-0.57008429  0.98903539 -1.31341185 -1.08930229]
 [ 0.37645691 -1.02448055  0.61782732 -0.33310061]
 [ 0.19362738  0.03544516  0.69558454  1.4224029 ]]

Intercepts:
 [0. 0. 0.]

Training Set Accuracy: 0.8482142857142857
Testing Set Accuracy: 0.8947368421052632

Confusion Matrix:
```



	Classes	Sensitivities	Specificities
0	setosa	1.000000	1.000000
1	versicolor	0.785714	0.958333
2	virginica	0.933333	0.869565

```
 Additional Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         9
  versicolor       0.92      0.79      0.85        14
   virginica       0.82      0.93      0.87        15

    accuracy                           0.89        38
   macro avg       0.91      0.91      0.91        38
weighted avg       0.90      0.89      0.89        38
```

Without the intercept, there is no significantly difference on the accuracy. Yet we can see the performance of sensitivity and specificity is slighly lower than the outcome with the intercept in general.

ILLINOIS

```
In [ ]:
```