In [11]:
```
 #Load required libraries
from sklearn import datasets
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix, precision_recall_fscore_support,classifi
cation_report,average_precision_score,precision_recall_curve
from sklearn.svm import SVC
from sklearn.linear_model import LinearRegression
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.utils.fixes import signature
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
```

In [12]:
```
# Load the iris dataset
iris = datasets.load_iris()

# Create our X and y data
X = iris.data
y = iris.target
```

In [13]:
```
# View the first five observations of our y data
y[:5]
```

Out[13]:
```
array([0, 0, 0, 0, 0])
```

In [14]:
```
# View the first five observations of our x data.
# Notice that there are four independent variables (features)
X[:5]
```

Out[14]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

In [15]:
```
# Split the data into 70% training data and 30% test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [16]:
```
# Train the scaler, which standarizes all the features to have mean=0 and unit variance
sc = StandardScaler()
sc.fit(X_train)
```

Out[16]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [17]:
```
# Apply the scaler to the X training data
X_train_std = sc.transform(X_train)

# Apply the SAME scaler to the X test data
X_test_std = sc.transform(X_test)
```

In [18]:
```
# Create a perceptron object with the parameters: 40 iterations (epochs) over the data, and a learnin
g rate of 0.1
ppn = Perceptron(n_iter=40, eta0=0.1, random_state=0)

# Train the perceptron
ppn.fit(X_train_std, y_train)
```

Out[18]:
```
Perceptron(alpha=0.0001, class_weight=None, early_stopping=False, eta0=0.1,
      fit_intercept=True, max_iter=None, n_iter=40, n_iter_no_change=5,
      n_jobs=None, penalty=None, random_state=0, shuffle=True, tol=None,
      validation_fraction=0.1, verbose=0, warm_start=False)
```

In [19]:
```
# Apply the trained perceptron on the X data to make predicts for the y test data
y_pred = ppn.predict(X_test_std)
```

In [20]:
```
# View the predicted y test data
y_pred
```

Out[20]:
```
array([1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 2, 0, 0, 2, 2, 0, 0, 2, 1, 2, 2, 0,
       2, 2, 1, 2, 1, 0, 1, 0, 1, 2, 2, 2, 2, 1, 0, 0, 2, 0, 2, 2, 1, 2,
       0])
```

In [21]:
```
# View the true y test data
y_test
```

Out[21]:
```
array([1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 2, 1, 1, 2, 1, 0, 0, 2, 1, 1, 2, 0,
       2, 2, 1, 2, 1, 0, 1, 0, 1, 2, 2, 1, 1, 1, 1, 0, 2, 0, 2, 2, 1, 2,
       0])
```

In [22]:
```
 #View the accuracy of the model, which is: 1 - (observations predicted wrong / total observations)
 #Accuracy: The amount of correct classifications / the total amount of classifications.
 #The train accuracy: The accuracy of a model on examples it was constructed on.
  #The test accuracy is the accuracy of a model on examples it hasn't seen.
accuracy_test_ppn=round(ppn.score(X_test,y_test)*100,2)
accuracy_train_ppn=round(ppn.score(X_train,y_train)*100,2)
accuracy_ppn=round(accuracy_score(y_test, y_pred)*100,2)
print('Training accuracy of perceptron',accuracy_train_ppn)
print('Testing accuracy of perceptron',accuracy_test_ppn)
print('Accuracy of Perceptron:',accuracy_ppn)
```

```
Training accuracy of perceptron 70.48
Testing accuracy of perceptron 57.78
Accuracy of Perceptron: 84.44
```

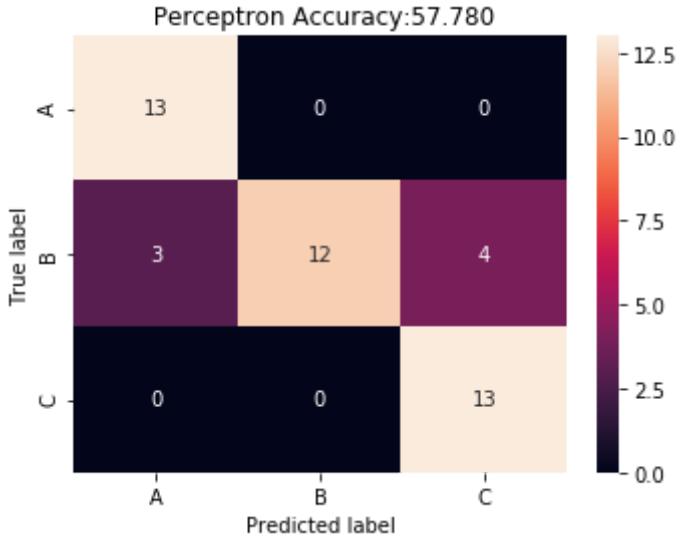
In [23]:
```
#Confusion Matrix for Perceptron
cm = confusion_matrix(y_test, y_pred)
```

In [24]:
```
cm
```

Out[24]:
```
array([[13,  0,  0],
       [ 3, 12,  4],
       [ 0,  0, 13]])
```

In [25]:
```
cm_df = pd.DataFrame(cm,
                     index = ['A','B','C'],
                     columns = ['A','B','C'])
```

In [26]:
```
plt.figure(figsize=(5.5,4))
sns.heatmap(cm_df, annot=True)
plt.title(' Perceptron Accuracy:{0:.3f}'.format(accuracy_test_ppn))
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



In [27]:
```
target_names = ['A', 'B','C']
print(classification_report(y_test, y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

           A       0.81      1.00      0.90        13
           B       1.00      0.63      0.77        19
           C       0.76      1.00      0.87        13

   micro avg       0.84      0.84      0.84        45
   macro avg       0.86      0.88      0.85        45
weighted avg       0.88      0.84      0.84        45
```

In [28]:
```
#SVM implementation with same dataset
svm_clf = SVC()
```

In [29]:
```
svm_clf.fit(X_train,y_train)
```

Out[29]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

In [30]:
```
svm_pred = svm_clf.predict(X_test)
```

In [31]:
```
svm_pred
```

Out[31]:
```
array([1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 2, 1, 1, 2, 2, 0, 0, 2, 1, 2, 2, 0,
       2, 2, 1, 2, 1, 0, 1, 0, 1, 2, 2, 2, 1, 1, 1, 0, 2, 0, 2, 2, 1, 2,
       0])
```

In [32]:
```
y_test
```

Out[32]:
```
array([1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 2, 1, 1, 2, 1, 0, 0, 2, 1, 1, 2, 0,
       2, 2, 1, 2, 1, 0, 1, 0, 1, 2, 2, 1, 1, 1, 1, 0, 2, 0, 2, 2, 1, 2,
       0])
```

In [33]:
```
accuracy_test_svm=round(svm_clf.score(X_test,y_test)*100,2)
accuracy_train_svm=round(svm_clf.score(X_train,y_train)*100,2)
accuracy_svm=round(accuracy_score(y_test, svm_pred)*100,2)
print('Training accuracy of SVM',accuracy_train_svm)
print('Testing accuracy of SVM',accuracy_test_svm)
print('Accuracy of SVM classifier:',accuracy_svm)
```

```
Training accuracy of SVM 100.0
Testing accuracy of SVM 93.33
Accuracy of SVM classifier: 93.33
```

In [34]:
```
target_names = ['A', 'B','C']
print(classification_report(y_test, svm_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

           A       1.00      1.00      1.00        13
           B       1.00      0.84      0.91        19
           C       0.81      1.00      0.90        13

   micro avg       0.93      0.93      0.93        45
   macro avg       0.94      0.95      0.94        45
weighted avg       0.95      0.93      0.93        45
```

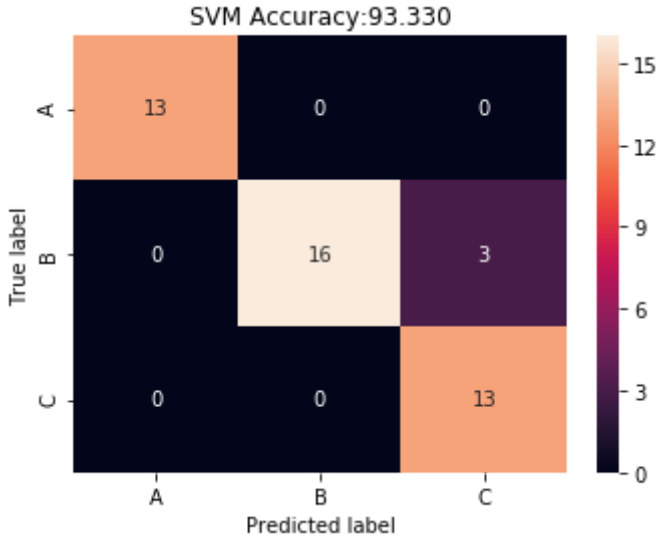
In [35]:
```
#Confusion Matrix for SVM
cm = confusion_matrix(y_test, svm_pred)
```

In [36]:
```
cm
```

Out[36]:
```
array([[13,  0,  0],
       [ 0, 16,  3],
       [ 0,  0, 13]])
```

In [37]:
```
cm_df = pd.DataFrame(cm,
                     index = ['A','B','C'],
                     columns = ['A','B','C'])
```

In [38]:
```
plt.figure(figsize=(5.5,4))
sns.heatmap(cm_df, annot=True)
plt.title('SVM Accuracy:{0:.3f}'.format(accuracy_test_svm))
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



In [39]:
```
target_names = ['A', 'B','C']
print(classification_report(y_test, svm_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

           A       1.00      1.00      1.00        13
           B       1.00      0.84      0.91        19
           C       0.81      1.00      0.90        13

   micro avg       0.93      0.93      0.93        45
   macro avg       0.94      0.95      0.94        45
weighted avg       0.95      0.93      0.93        45
```

In [40]:
```
#Linear Regression Implementation of same dataset
lr_reg_clf = LinearRegression(fit_intercept=True)
```

In [41]:
```
lr_reg_clf.fit(X_train,y_train)
```

Out[41]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [42]:
```
lr_pred = lr_reg_clf.predict(X_test)
```

In [43]:
```
lr_pred
```

Out[43]:
```
array([ 8.98987388e-01, -8.62874600e-02,  1.03975478e+00, -2.26508605e-02,
        1.22065822e-02,  1.34228215e+00,  1.43894562e+00, -5.66522661e-02,
        1.20472678e+00,  2.10748218e-02,  1.65873956e+00,  1.45416351e+00,
        1.05560521e+00,  2.00557918e+00,  1.65965864e+00,  1.42202655e-01,
       -7.89078164e-02,  2.32582383e+00,  1.18169556e+00,  1.62444001e+00,
        1.58854499e+00,  1.47228198e-01,  2.03382508e+00,  2.14633710e+00,
        8.32356553e-01,  1.82340396e+00,  9.38816808e-01, -1.09922573e-01,
        1.27400040e+00, -7.84835854e-02,  1.35277452e+00,  2.23123740e+00,
        1.75366986e+00,  1.51408350e+00,  1.50478157e+00,  1.28635960e+00,
        1.20280450e+00,  5.60021094e-04,  1.82459440e+00, -8.48256571e-02,
        1.80276308e+00,  1.80276308e+00,  1.09232191e+00,  2.07593609e+00,
       -1.50481953e-01])
```

In [44]:
```
y_test
```

Out[44]:
```
array([1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 2, 1, 1, 2, 1, 0, 0, 2, 1, 1, 2, 0,
       2, 2, 1, 2, 1, 0, 1, 0, 1, 2, 2, 1, 1, 1, 1, 0, 2, 0, 2, 2, 1, 2,
       0])
```

In [45]:
```
accuracy_test_lr=round(lr_reg_clf.score(X_test,y_test)*100,2)
accuracy_train_lr=round(lr_reg_clf.score(X_train,y_train)*100,2)
print('Training accuracy of Linear Regression',accuracy_train_lr)
print('Testing accuracy of Linear Regression',accuracy_test_lr)
```

```
Training accuracy of Linear Regression 94.32
Testing accuracy of Linear Regression 88.1
```