Machine learning algorithms for classification and regression on iris in Python

In [19]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd
import numpy as np


# Convert 'iris.data' numpy array to 'iris.dataframe' pandas dataframe
# complete the iris dataset by adding species
iris = datasets.load_iris()
iris = pd.DataFrame(
    data= np.c_[iris['data'], iris['target']],
    columns= iris['feature_names'] + ['target']
    )

species = []

for i in range(len(iris['target'])):
    if iris['target'][i] == 0:
        species.append("setosa")
    elif iris['target'][i] == 1:
        species.append('versicolor')
    else:
        species.append('virginica')


iris['species'] = species
iris
```

Out[19]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	species
0	5.1	3.5	1.4	0.2	0.0	setosa
1	4.9	3.0	1.4	0.2	0.0	setosa
2	4.7	3.2	1.3	0.2	0.0	setosa
3	4.6	3.1	1.5	0.2	0.0	setosa
4	5.0	3.6	1.4	0.2	0.0	setosa
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2.0	virginica
146	6.3	2.5	5.0	1.9	2.0	virginica
147	6.5	3.0	5.2	2.0	2.0	virginica
148	6.2	3.4	5.4	2.3	2.0	virginica
149	5.9	3.0	5.1	1.8	2.0	virginica

150 rows × 6 columns

1. splitting the dataset into training and test sets

In [62]:
```
X = iris.iloc[:, 0:4]
y = iris.iloc[:, 4]
class_names = iris.iloc[:, 5]

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
import random

random.seed(2023)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=40, train_s
```

2. Saving a copy of the different datasets in .csv files

In [63]:
```
# 2. save entire dataset, training and testing datasets
# save a copy of the dataset in .csv
iris.to_csv('C:/Users/julia/OneDrive/Desktop/github/24. Machine learning toolbox Py

iris.to_csv('C:/Users/julia/OneDrive/Desktop/github/24. Machine learning toolbox Py
          index = False)

iris.to_csv('C:/Users/julia/OneDrive/Desktop/github/24. Machine learning toolbox Py
          index = False)
```

1. Naive Bayes classifier

1.1 train the model

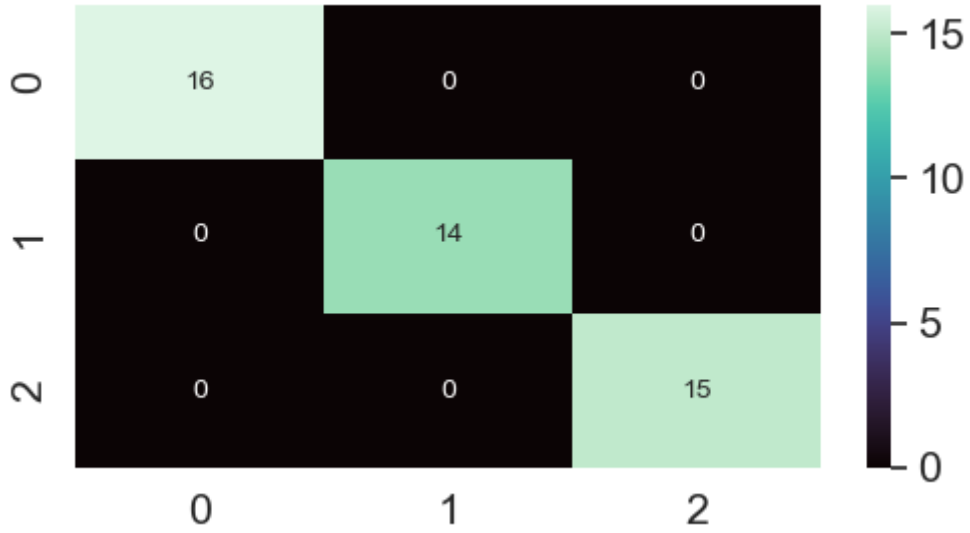
In [92]:
```
from sklearn.naive_bayes import GaussianNB

# create a Gaussian RF classifier
nb_model = GaussianNB()

# fit the model to the iris dataset
nb_model.fit(X_train,y_train)

# make predictions on test set
y_pred_nb = nb_model.predict(X_test)
```

1.2 Confusion matrix

Now that we have predictions, we can compute a confusion matrix and the accuracy of our trained SVM classifier on the testing set.

In [93]:
```
from sklearn.metrics import confusion_matrix

cm_nb = confusion_matrix(y_test, y_pred_nb)
cm_nb

df_cm_nb = pd.DataFrame(cm_nb, range(len(class_names.unique())), range(len(class_n

plt.figure(figsize=(6,3))
```

```
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm_nb, annot=True, annot_kws={"size": 10}, cmap = sns.color_palette

plt.show()
```



1.3 Accuracy of the Naive Bayes classifier

In [94]:
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

accuracy_test_nb = round(accuracy_score(y_test, y_pred_nb)* 100, 2)
accuracy_train_nb = round(nb_model.score(X_train, y_train) * 100, 2)
print("Accuracy testing: %.3f"  % accuracy_test_nb)
print("Accuracy training: %.3f" % accuracy_train_nb)
```

```
Accuracy testing: 100.000
Accuracy training: 95.240
```

2. Random forest classifier

2.1 train the model

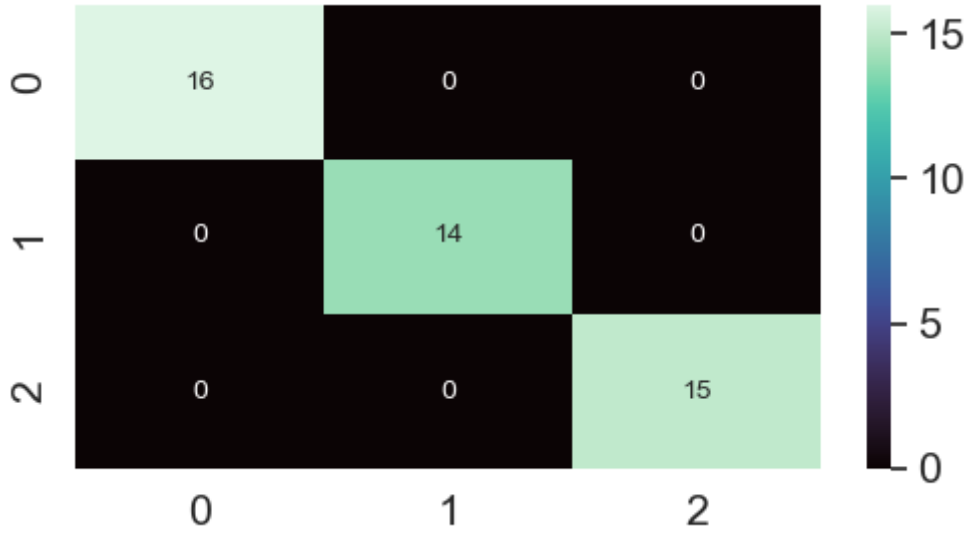
In [95]:
```
from sklearn.ensemble import RandomForestClassifier

# create a Gaussian RF classifier
rf_model = RandomForestClassifier(n_estimators=1000)

# fit the model to the iris dataset
rf_model.fit(X_train,y_train)

# make predictions on test set
y_pred_rf = rf_model.predict(X_test)
```

2.2 Confusion matrix and accuracy

Now that we have predictions, we can compute a confusion matrix and the accuracy of our trained SVM classifier on the testing set.

In [98]:
```
from sklearn.metrics import confusion_matrix
```

```
cm_rf = confusion_matrix(y_test, y_pred_rf)
cm_rf

df_cm_rf = pd.DataFrame(cm_rf, range(len(class_names.unique())), range(len(class_na

plt.figure(figsize=(6,3))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm_rf, annot=True, annot_kws={"size": 10}, cmap = sns.color_palette

plt.show()
```



2.3 Accuracy of the Random forest classifier

In [99]:
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

accuracy_test_rf = round(accuracy_score(y_test, y_pred_rf)* 100, 2)
accuracy_train_rf = round(rf_model.score(X_train, y_train) * 100, 2)
print("Accuracy testing: %.3f"  % accuracy_test_rf)
print("Accuracy training: %.3f" % accuracy_train_rf)
```

```
Accuracy testing: 100.000
Accuracy training: 100.000
```

3. Multinomial Logistic Regression classifier

3.1 train the model

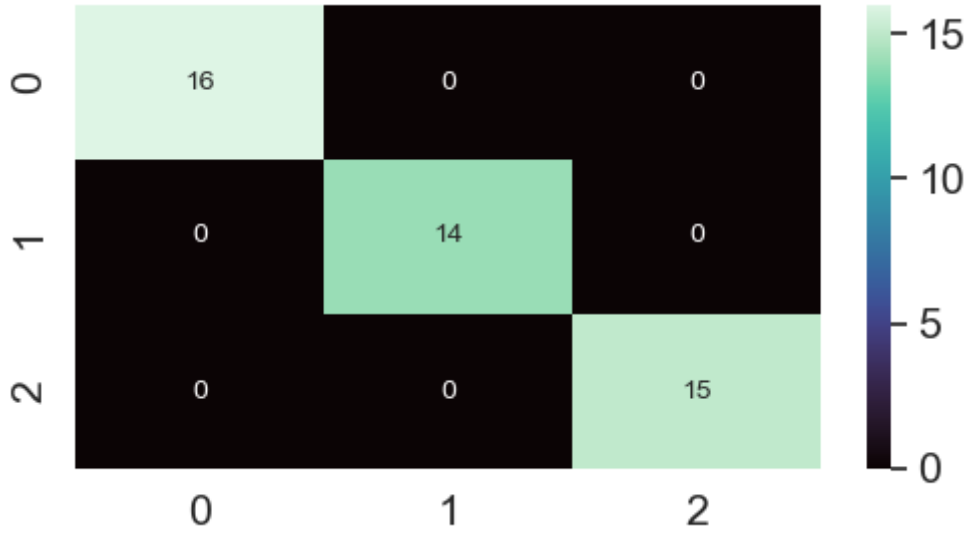
In [103…
```
from sklearn.linear_model import  LogisticRegression
from sklearn.metrics import accuracy_score

# create a Logistic regresion model
lr_model = LogisticRegression(solver= 'lbfgs', max_iter=400, multi_class = 'multin

# fit the model to the iris dataset
lr_model.fit(X_train, y_train)

# make predictions on test set
y_pred_lr = lr_model.predict(X_test)
```

3.2 Confusion matrix and accuracy

Now that we have predictions, we can compute a confusion matrix and the accuracy of our trained SVM classifier on the testing set.

In [106…

```
from sklearn.metrics import confusion_matrix
cm_lr = confusion_matrix(y_test, y_pred_lr)
cm_lr

df_cm_lr = pd.DataFrame(cm_lr, range(len(class_names.unique())), range(len(class_n

plt.figure(figsize=(6,3))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm_lr, annot=True, annot_kws={"size": 10}, cmap = sns.color_palette

plt.show()
```



3.3 Accuracy of the Multinomial Logistic Regression classifier

In [108…

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

accuracy_test_lr = round(accuracy_score(y_test, y_pred_lr)* 100, 2)
accuracy_train_lr = round(rf_model.score(X_train, y_train) * 100, 2)
print("Accuracy testing: %.3f"  % accuracy_test_lr)
print("Accuracy training: %.3f" % accuracy_train_lr)
```

```
Accuracy testing: 100.000
Accuracy training: 100.000
```

4. Support Vector Machines classifier

4.1 train the model

In [100…

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# create a SVM model
svm_model = SVC(kernel = 'linear', random_state = 0)
```

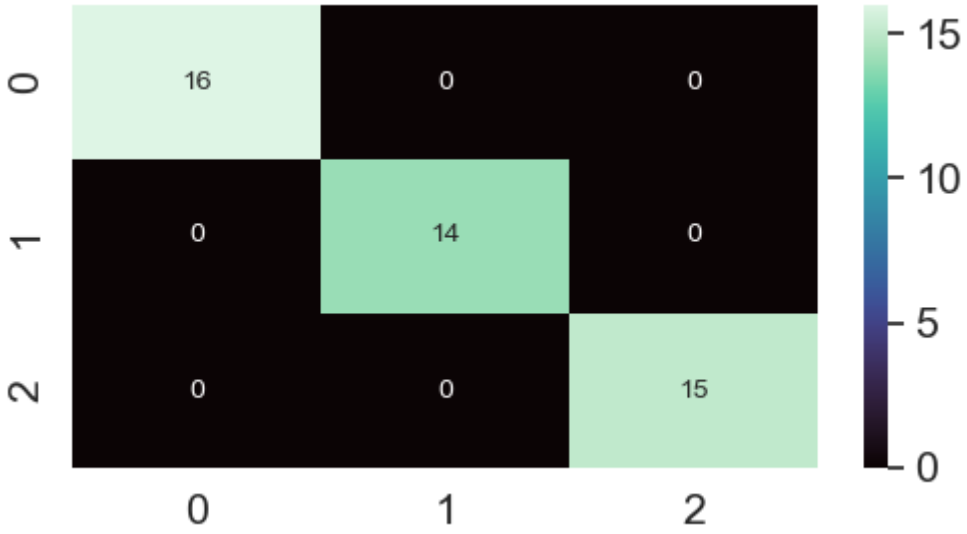
```
# fit the model to the iris dataset
svm_model.fit(X_train, y_train)

# make predictions on test set
y_pred_svm = svm_model.predict(X_test)
```

4.2 Confusion matrix and accuracy

Now that we have predictions, we can compute a confusion matrix and the accuracy of our trained SVM classifier on the testing set.

In [101…

```
from sklearn.metrics import confusion_matrix
cm_svm = confusion_matrix(y_test, y_pred_svm)
cm_svm

df_cm_svm = pd.DataFrame(cm_svm, range(len(class_names.unique())), range(len(class_

plt.figure(figsize=(6,3))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm_svm, annot=True, annot_kws={"size": 10}, cmap = sns.color_palett

plt.show()
```



4.3 Accuracy of the SVM classifier

In [102…

```
from sklearn.model_selection import cross_val_score

accuracy_test_svm = round(accuracy_score(y_test, y_pred_svm)* 100, 2)
accuracy_train_svm = round(svm_model.score(X_train, y_train) * 100, 2)
print("Accuracy testing: %.3f"  % accuracy_test_svm)
print("Accuracy training: %.3f" % accuracy_train_svm)
```

```
Accuracy testing: 100.000
Accuracy training: 97.140
```

In [ ]: