Exercise 9

Take the iris flower dataset and use different Classifier model such as Random Forest, Decision Tree, SVM and logistic Regression and use cross_val_score method to measure the performance of the best classifier

In [102…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
```

In [103…
```
data = load_iris()
dir(data)
```

Out[103]:
```
['DESCR',
 'data',
 'data_module',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

In [104…
```
df = pd.DataFrame(data.data, columns=data.feature_names)
df.head()
```

Out[104]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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

In [105…
```
df['target'] = data.target
df['flower name'] = df.target.apply(lambda x: data.target_names[x])
df.head()
```

Out[105]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

In [106…
```
X = df.drop(['target', 'flower name'], axis=1)
y = df['target']
```

In [107…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [108…
```
kf = StratifiedKFold()
kf
```

Out[108]:
```
StratifiedKFold(n_splits=5, random_state=None, shuffle=False)
```

In [109…
```
lr = cross_val_score(LogisticRegression(max_iter=1000), X, y, cv=kf)
lr
```

Out[109]:
```
array([0.96666667, 1.        , 0.93333333, 0.96666667, 1.        ])
```

In [110…
```
svm = cross_val_score(SVC(kernel="linear"), X, y, cv=kf)
svm
```

Out[110]:
```
array([0.96666667, 1.        , 0.96666667, 0.96666667, 1.        ])
```

In [111…
```
dt = cross_val_score(DecisionTreeClassifier(), X, y, cv=kf)
dt
```

Out[111]:
```
array([0.96666667, 0.96666667, 0.9       , 0.93333333, 1.        ])
```

In [112…
```
rf = cross_val_score(RandomForestClassifier(n_estimators=40), X, y, cv=kf)
rf
```

Out[112]:
```
array([0.96666667, 0.96666667, 0.93333333, 0.93333333, 1.        ])
```

In [113…
```
print(f"Logistic Regression avg score : {np.mean(lr)}")
print(f"SVM avg score : {np.mean(svm)}")
print(f"Decision tree avg score : {np.mean(dt)}")
print(f"Random Forest Classifier avg score : {np.mean(rf)}")
```

```
Logistic Regression avg score : 0.9733333333333334
SVM avg score : 0.9800000000000001
Decision tree avg score : 0.9533333333333334
Random Forest Classifier avg score : 0.96
```

SVM performed the best

In [114…
```
X_test.head()
```

Out[114]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
54	6.5	2.8	4.6	1.5
27	5.2	3.5	1.5	0.2
119	6.0	2.2	5.0	1.5
110	6.5	3.2	5.1	2.0
99	5.7	2.8	4.1	1.3

In [115…
```
df.iloc[13]
```

Out[115]:
```
sepal length (cm)       4.3
sepal width (cm)        3.0
petal length (cm)       1.1
petal width (cm)        0.1
target                    0
flower name          setosa
Name: 13, dtype: object
```

In [116…
```
model = SVC(kernel="linear")
model.fit(X_train, y_train)
```

Out[116]:
```
SVC(kernel='linear')
```

In [117…
```
predict = model.predict(X_test)
predict
```

Out[117]:
```
array([1, 0, 2, 2, 1, 2, 2, 2, 1, 0, 1, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1,
       2, 0, 1, 0, 2, 2, 1, 0])
```

In [118…
```
predict1 = predict.astype(np.int32)  # Adjust the data type if necessary
y_test1 = y_test.astype(np.int32)
right = np.sum(predict1 == y_test1)
wrong = np.sum(predict1 != y_test1)

print(f"Num of Correct predictions {right}")
print(f"Num of Wrong predictions {wrong}")
```

```
Num of Correct predictions 28
Num of Wrong predictions 2
```

In [120…
```
cm = confusion_matrix(y_test, predict)
plt.figure(figsize=(10, 7))
sn.heatmap(cm, annot=True)
plt.xlabel("Predicted")
plt.ylabel("Truth")
```

Out[120]:
```
Text(95.72222222222221, 0.5, 'Truth')
```