Exercise 8

Use famous iris flower dataset from sklearn.datasets to predict flower species using random forest classifier.

1. Measure prediction score using default n_estimators (10)
2. Now fine tune your model by changing number of trees in your classifer and tell me what best score you can get using how many trees

```
In [222...  import pandas as pd
           import matplotlib.pyplot as plt
           import numpy as np
           from sklearn.datasets import load_iris
           from sklearn.model_selection import train_test_split
           from sklearn.ensemble import RandomForestClassifier
```

```
In [223...  data = load_iris()
```

```
In [224...  dir(data)
```

```
Out[224]: ['DESCR',
           'data',
           'data_module',
           'feature_names',
           'filename',
           'frame',
           'target',
           'target_names']
```

```
In [225...  data.data
```

```
Out[225]: 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],
                 [5.4, 3.9, 1.7, 0.4],
                 [4.6, 3.4, 1.4, 0.3],
                 [5. , 3.4, 1.5, 0.2],
                 [4.4, 2.9, 1.4, 0.2],
                 [4.9, 3.1, 1.5, 0.1],
                 [5.4, 3.7, 1.5, 0.2],
                 [4.8, 3.4, 1.6, 0.2],
                 [4.8, 3. , 1.4, 0.1],
                 [4.3, 3. , 1.1, 0.1],
                 [5.8, 4. , 1.2, 0.2],
                 [5.7, 4.4, 1.5, 0.4],
                 [5.4, 3.9, 1.3, 0.4],
                 [5.1, 3.5, 1.4, 0.3],
                 [5.7, 3.8, 1.7, 0.3],
                 [5.1, 3.8, 1.5, 0.3],
                 [5.4, 3.4, 1.7, 0.2],
                 [5.1, 3.7, 1.5, 0.4],
                 [4.6, 3.6, 1. , 0.2],
                 [5.1, 3.3, 1.7, 0.5],
                 [4.8, 3.4, 1.9, 0.2],
                 [5. , 3. , 1.6, 0.2],
                 [5. , 3.4, 1.6, 0.4],
                 [5.2, 3.5, 1.5, 0.2],
                 [5.2, 3.4, 1.4, 0.2],
                 [4.7, 3.2, 1.6, 0.2],
                 [4.8, 3.1, 1.6, 0.2],
                 [5.4, 3.4, 1.5, 0.4],
                 [5.2, 4.1, 1.5, 0.1],
                 [5.5, 4.2, 1.4, 0.2],
                 [4.9, 3.1, 1.5, 0.2],
                 [5. , 3.2, 1.2, 0.2],
                 [5.5, 3.5, 1.3, 0.2],
                 [4.9, 3.6, 1.4, 0.1],
                 [4.4, 3. , 1.3, 0.2],
                 [5.1, 3.4, 1.5, 0.2],
                 [5. , 3.5, 1.3, 0.3],
                 [4.5, 2.3, 1.3, 0.3],
                 [4.4, 3.2, 1.3, 0.2],
                 [5. , 3.5, 1.6, 0.6],
                 [5.1, 3.8, 1.9, 0.4],
                 [4.8, 3. , 1.4, 0.3],
                 [5.1, 3.8, 1.6, 0.2],
                 [4.6, 3.2, 1.4, 0.2],
```
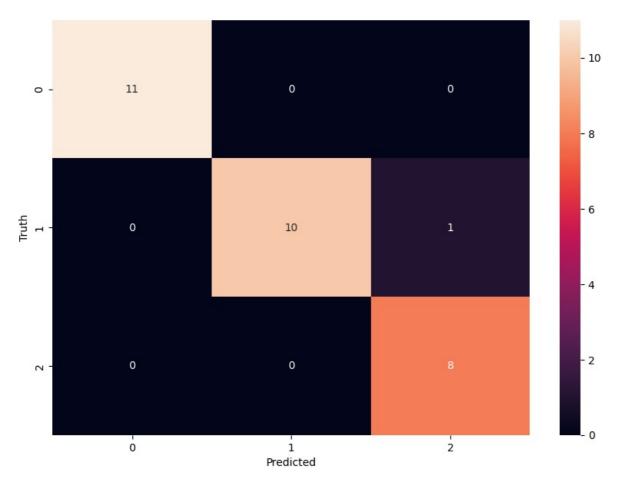
```
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
       [6.3, 3.3, 4.7, 1.6],
       [4.9, 2.4, 3.3, 1. ],
       [6.6, 2.9, 4.6, 1.3],
       [5.2, 2.7, 3.9, 1.4],
       [5. , 2. , 3.5, 1. ],
       [5.9, 3. , 4.2, 1.5],
       [6. , 2.2, 4. , 1. ],
       [6.1, 2.9, 4.7, 1.4],
       [5.6, 2.9, 3.6, 1.3],
       [6.7, 3.1, 4.4, 1.4],
       [5.6, 3. , 4.5, 1.5],
       [5.8, 2.7, 4.1, 1. ],
       [6.2, 2.2, 4.5, 1.5],
       [5.6, 2.5, 3.9, 1.1],
       [5.9, 3.2, 4.8, 1.8],
       [6.1, 2.8, 4. , 1.3],
       [6.3, 2.5, 4.9, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [6.4, 2.9, 4.3, 1.3],
       [6.6, 3. , 4.4, 1.4],
       [6.8, 2.8, 4.8, 1.4],
       [6.7, 3. , 5. , 1.7],
       [6. , 2.9, 4.5, 1.5],
       [5.7, 2.6, 3.5, 1. ],
       [5.5, 2.4, 3.8, 1.1],
       [5.5, 2.4, 3.7, 1. ],
       [5.8, 2.7, 3.9, 1.2],
       [6. , 2.7, 5.1, 1.6],
       [5.4, 3. , 4.5, 1.5],
       [6. , 3.4, 4.5, 1.6],
       [6.7, 3.1, 4.7, 1.5],
       [6.3, 2.3, 4.4, 1.3],
       [5.6, 3. , 4.1, 1.3],
       [5.5, 2.5, 4. , 1.3],
       [5.5, 2.6, 4.4, 1.2],
       [6.1, 3. , 4.6, 1.4],
       [5.8, 2.6, 4. , 1.2],
       [5. , 2.3, 3.3, 1. ],
       [5.6, 2.7, 4.2, 1.3],
       [5.7, 3. , 4.2, 1.2],
       [5.7, 2.9, 4.2, 1.3],
       [6.2, 2.9, 4.3, 1.3],
       [5.1, 2.5, 3. , 1.1],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
```

```
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

In [226… `data.target`

```
Out[226]: 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, 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])
```

In [227…
```python
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
```

In [228…
```python
df['flower name'] = df.target.apply(lambda x: data.target_names[x])
```

In [229…
```python
df.head()
```

Out[229]:

	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 [230…
```python
X = df.drop(['target', 'flower name'], axis=1)
y = df['target']
```

In [231…
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [232…
```python
model = RandomForestClassifier(n_estimators=20)
```

In [233…
```python
model.fit(X_train, y_train)
model.get_params();
```

In [234…
```python
model.score(X_test, y_test)
```

Out[234]: 0.9666666666666667

Highest score

```
0.9666666666666667
```

In [235…
```python
predict = model.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, predict)
```

In [236…
```python
import seaborn as sn
plt.figure(figsize=(10, 7))
sn.heatmap(cm, annot=True)
plt.xlabel("Predicted")
plt.ylabel("Truth")
```

Out[236]: Text(95.72222222222221, 0.5, 'Truth')

```
In [243... obj = {
         "actual":y_test,
         "predict" : predict
         }

         newdf = pd.DataFrame(obj)
         newdf
```

	actual	predict
10	0	0
90	1	1
95	1	1
136	2	2
108	2	2
103	2	2
74	1	1
125	2	2
85	1	1
21	0	0
23	0	0
117	2	2
89	1	1
35	0	0
13	0	0
105	2	2
47	0	0
46	0	0
27	0	0
145	2	2
94	1	1
62	1	1
31	0	0
86	1	1
28	0	0
91	1	1
77	1	2
61	1	1
135	2	2
12	0	0

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js