Iris Flower Prediction

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.datasets import load_iris
data=load_iris()
df=pd.DataFrame(data.data,columns=data.feature_names)
data.feature_names
```

Out[1]:

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [2]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

In [43]:

```
df.tail()
```

Out[43]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

In [42]:

```
df.target.unique()
```

Out[42]:

```
array([0, 1, 2])
```

In [3]:

```
df['target']=data['target']
df.head()
```

Out[3]:

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

In [4]:

```
df.describe()
```

Out[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

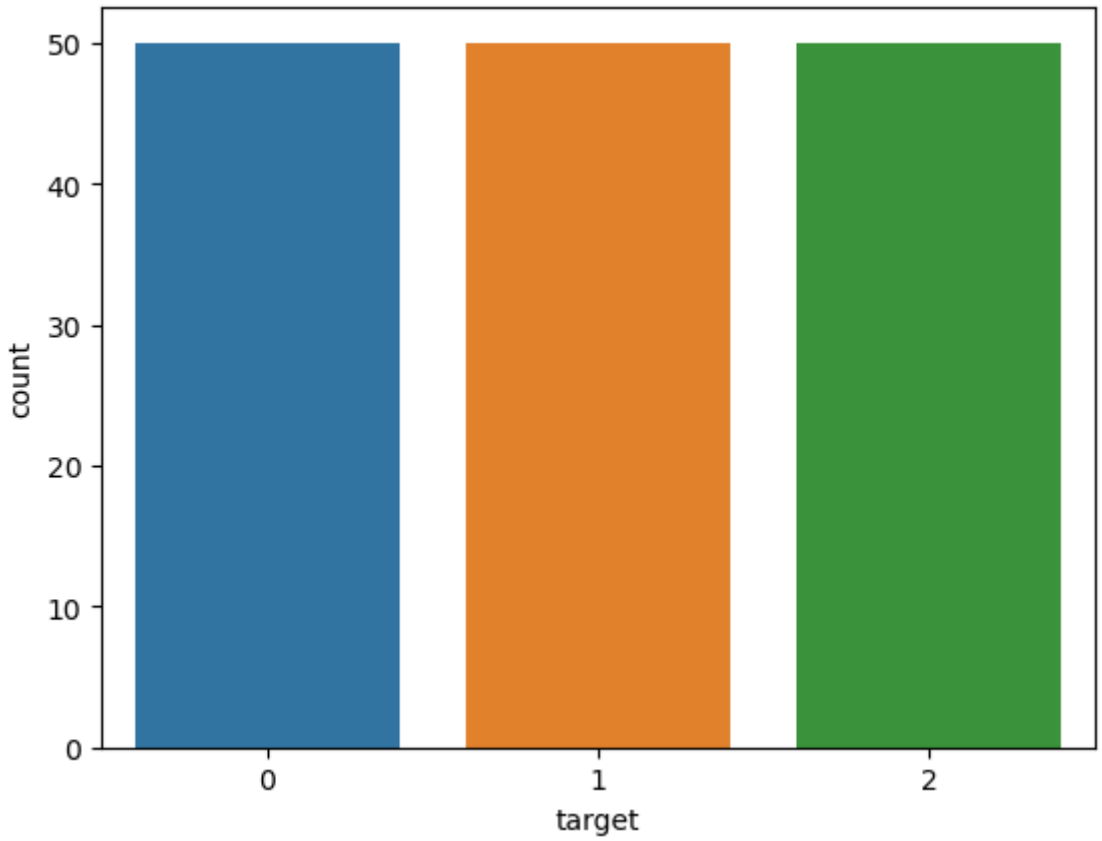
In [5]:

```
import numpy as np
import seaborn as sns
sns.countplot(df['target'])

print(df.target.value_counts())
```

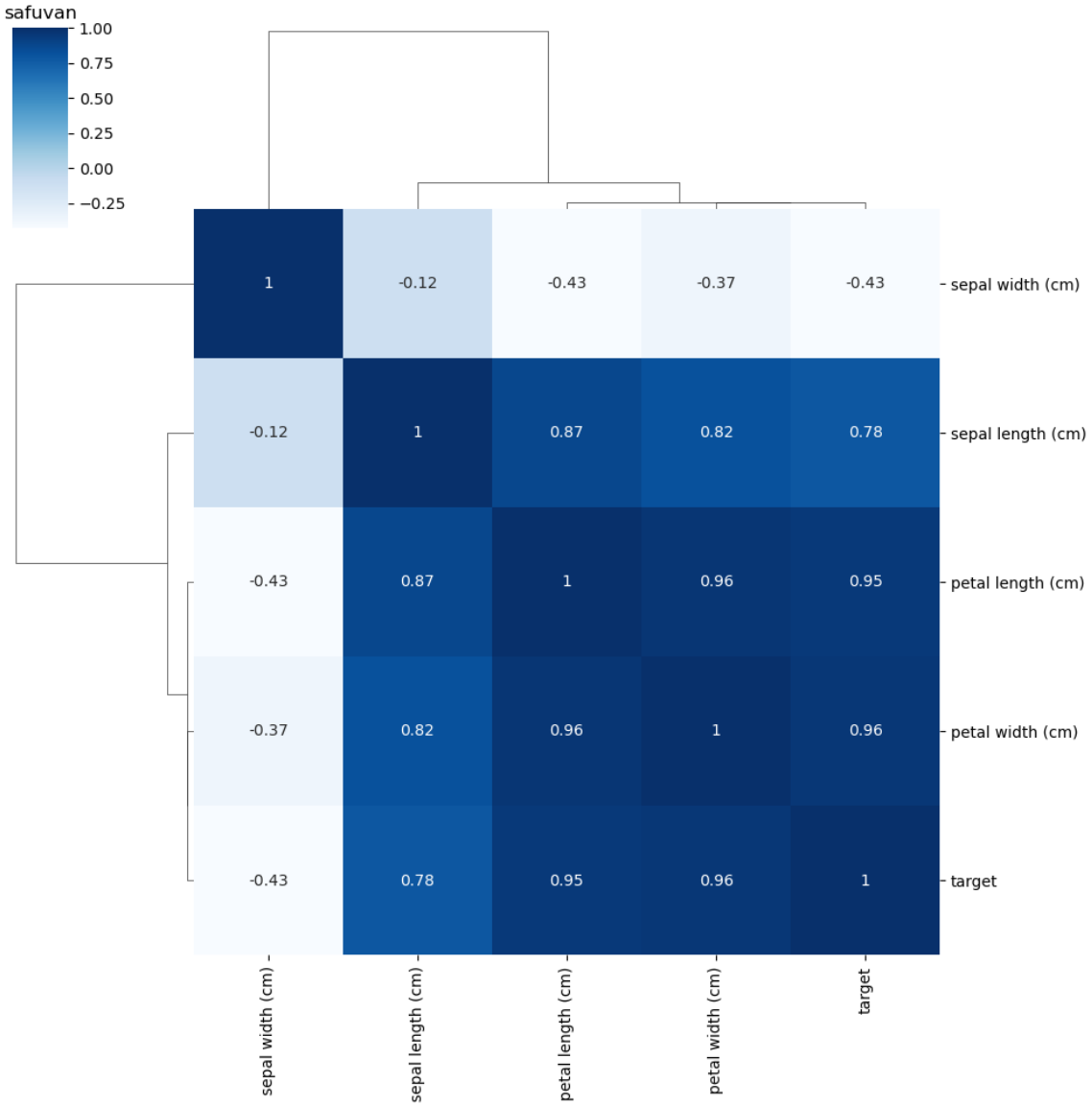
```
D:\Users\Safuvan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fu
tureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(

0    50
1    50
2    50
Name: target, dtype: int64
```



In [6]:

```
import matplotlib.pyplot as plt
corr=df.corr()
sns.clustermap(corr,annot=True,cmap="Blues")
plt.title("safuvan")
plt.show()
```



In [7]:

```
from sklearn.model_selection import train_test_split
x=df.drop(["target"],axis=1).values
y=df["target"].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
print("x_train.:",len(x_train))
print("x_test.shape:",x_test.shape)
print("y_train.shape:",len(y_train))
print("y_test.shape:",len(y_test))
```

```
x_train.: 120
x_test.shape: (30, 4)
y_train.shape: 120
y_test.shape: 30
```

In [8]:

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.linear_model import LogisticRegression
regressor=LogisticRegression()
regressor.fit(x_train,y_train)
regressor.score(x_test,y_test)
y_preds=regressor.predict(x_test)
print("accuracy:",accuracy_score(y_test,y_preds))

print("confusion:",confusion_matrix(y_test,y_preds))
print("cassification",classification_report(y_test,y_preds))
```

```
accuracy: 0.9666666666666667
confusion: [[12  0  0]
 [ 0 10  1]
 [ 0  0  7]]
cassification               precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      0.91      0.95        11
           2       0.88      1.00      0.93         7

    accuracy                           0.97        30
   macro avg       0.96      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

In [9]:

```
print("probabilty of prediction:",np.round(regressor.predict_proba(x_test),2))
```

```
probabilty of prediction: [[0.96 0.04 0.  ]
 [0.97 0.03 0.  ]
 [0.02 0.95 0.02]
 [0.98 0.02 0.  ]
 [0.18 0.82 0.  ]
 [0.   0.79 0.21]
 [0.01 0.97 0.02]
 [0.97 0.03 0.  ]
 [0.02 0.94 0.03]
 [0.97 0.03 0.  ]
 [0.02 0.87 0.11]
 [0.98 0.02 0.  ]
 [0.   0.01 0.99]
 [0.07 0.92 0.01]
 [0.   0.16 0.84]
 [0.   0.12 0.88]
 [0.05 0.94 0.01]
 [0.98 0.02 0.  ]
 [0.96 0.04 0.  ]
 [0.02 0.9  0.08]
 [0.96 0.04 0.  ]
 [0.   0.04 0.96]
 [0.   0.   1.  ]
 [0.   0.39 0.61]
 [0.01 0.73 0.26]
 [0.98 0.02 0.  ]
 [0.   0.08 0.92]
 [0.98 0.02 0.  ]
 [0.97 0.03 0.  ]
 [0.   0.01 0.99]]
```

In [13]:

```
value = [[4.9,3.0,1.4,0.2]]
```

In [14]:

```
regressor.predict(value)
```

Out[14]:

```
array([0])
```

In [15]:

```
import pickle
```

In [16]:

```
file = 'train_model.sav'
```

In [17]:

```
pickle.dump(regressor,open(file,'wb'))
```

In [18]:

```
model = pickle.load(open('train_model.sav','rb'))
```

In [ ]:

```

```