SVM

November 21, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
plt.rcParams['figure.figsize'] = [19, 8]
```

[3]:
```
import warnings
warnings.filterwarnings('ignore')
```

[4]:
```
from sklearn.datasets import load_iris
iris = load_iris()
dir(iris)
```

[4]:
```
['DESCR',
 'data',
 'data_module',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

[5]:
```
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df.head()
```

[5]:

	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

[6]:
```
iris_df['target'] = iris.target
iris_df.head()
```

[6]:

	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
```

```
   target
0       0
1       0
2       0
3       0
4       0
```

[7]:
```
iris_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 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
 4   target             150 non-null    int64  
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

[8]:
```
iris.target_names
```

[8]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

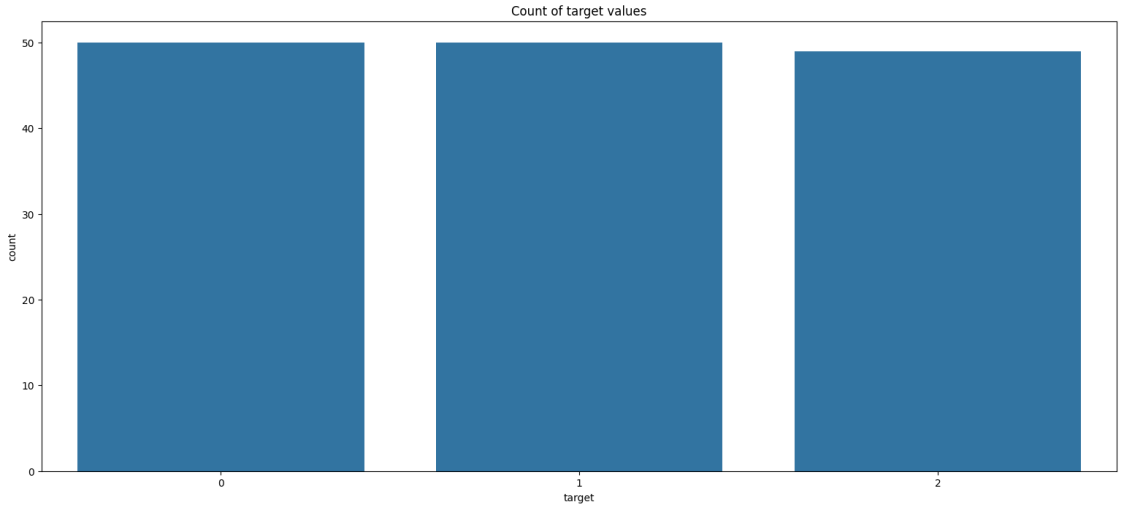
[9]:
```
iris_df.duplicated().sum()
```

[9]:
```
np.int64(1)
```

[10]:
```
iris_df.drop_duplicates(inplace=True)
iris_df.duplicated().sum()
```

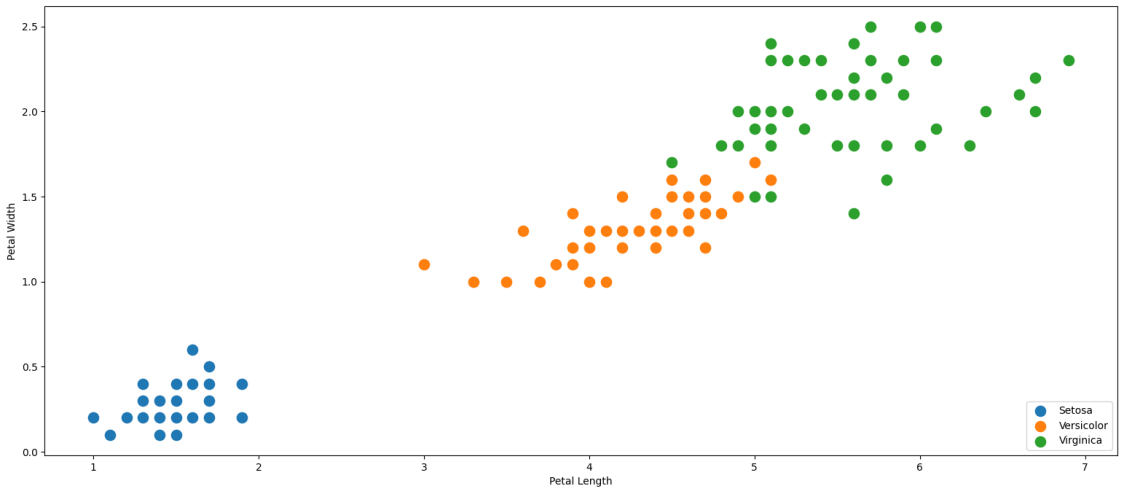
[10]:
```
np.int64(0)
```

[11]:
```
sns.countplot(data=iris_df, x='target')
plt.title("Count of target values")
plt.show()
```

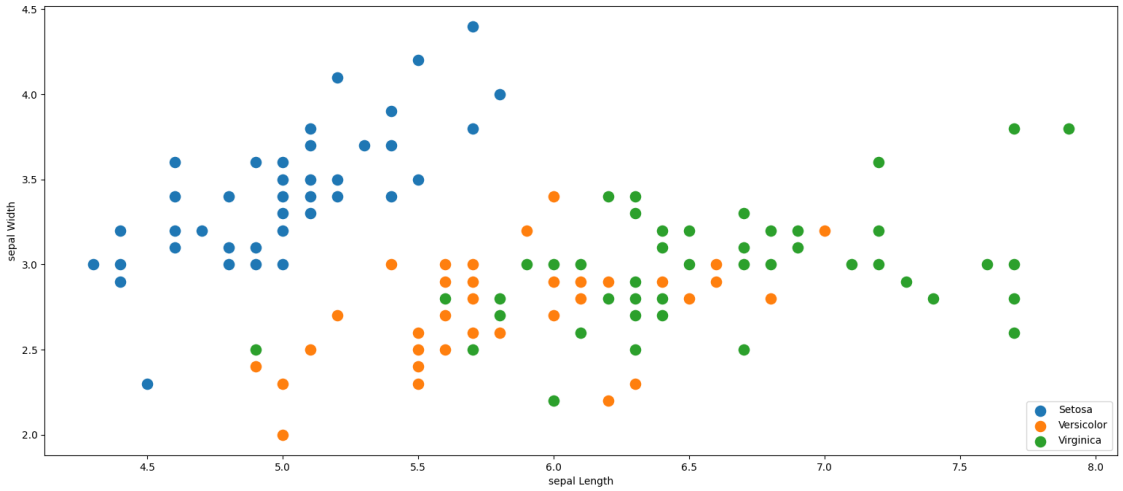


[12]:
```
iris_setosa = iris_df.loc[iris_df['target'] == 0, :]
iris_versicolor = iris_df.loc[iris_df['target'] == 1, :]
iris_virginica = iris_df.loc[iris_df['target'] == 2, :]
```

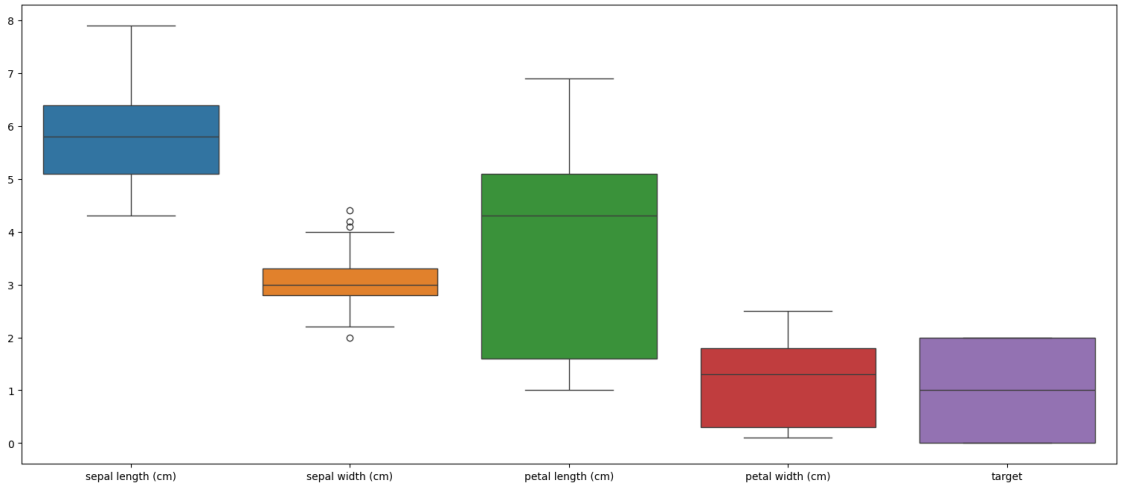
[13]:
```
sns.scatterplot(data=iris_setosa, x='petal length (cm)', y='petal width (cm)',␣
↪s=150)
sns.scatterplot(data=iris_versicolor, x='petal length (cm)', y='petal width␣
↪(cm)', s=150)
sns.scatterplot(data=iris_virginica, x='petal length (cm)', y='petal width␣
↪(cm)', s=150)
plt.legend(['Setosa', 'Versicolor', 'Virginica'], loc='lower right')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()
```



[14]:
```
sns.scatterplot(data=iris_setosa, x='sepal length (cm)', y='sepal width (cm)',␣
↪s=150)
sns.scatterplot(data=iris_versicolor, x='sepal length (cm)', y='sepal width␣
↪(cm)', s=150)
sns.scatterplot(data=iris_virginica, x='sepal length (cm)', y='sepal width␣
↪(cm)', s=150)
plt.legend(['Setosa', 'Versicolor', 'Virginica'], loc='lower right')
plt.xlabel('sepal Length')
plt.ylabel('sepal Width')
plt.show()
```



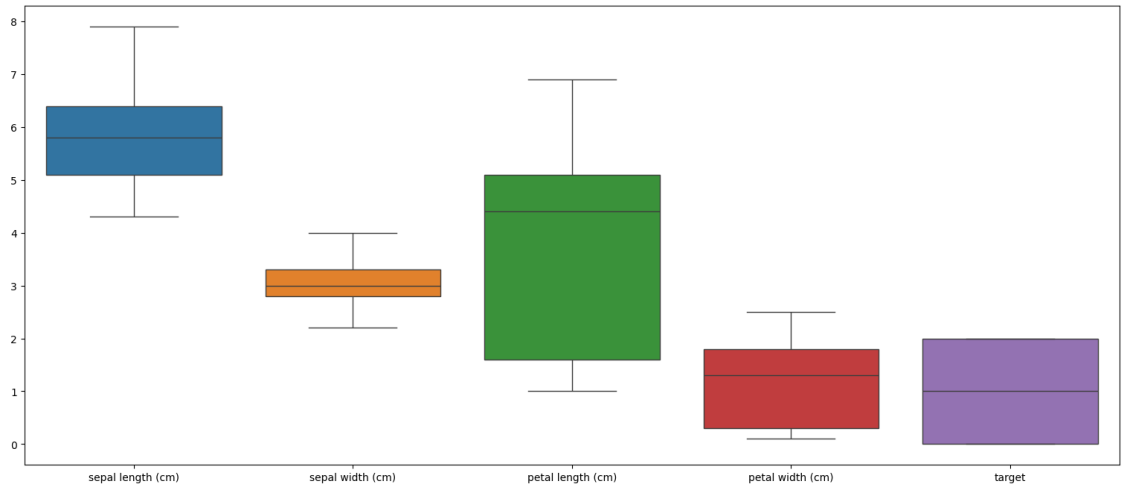
[15]:
```
sns.boxplot(iris_df)
plt.show()
```



[16]:
```
# obtain the first quartile
Q1 = iris_df.quantile(0.25)
# obtain the third quartile
Q3 = iris_df.quantile(0.75)
# obtain the IQR
IQR = Q3 - Q1
# print the IQR
print(IQR)
```

```
sepal length (cm)    1.3
sepal width (cm)     0.5
petal length (cm)    3.5
petal width (cm)     1.5
target               2.0
dtype: float64
```

[17]:
```
ul = Q3 + 1.5 * IQR
ll = Q1 - 1.5 * IQR
iris_df = iris_df[~((iris_df < ll) |(iris_df > ul)).any(axis=1)]
sns.boxplot(iris_df)
plt.show()
```



[18]:
```
X = iris_df.loc[:, :'petal width (cm)'].values
y = iris_df.loc[:, 'target'].values
y
```

[18]:
```
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, 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])
```

[19]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

[20]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=1)
```

[21]:
```
from sklearn.svm import SVC # Support Vector Classifier
```

[22]:
```
model = SVC(kernel='linear')
model.fit(X_train, y_train)
```

[22]:
```
SVC(kernel='linear')
```

[23]:
```
model.score(X_train, y_train)
```

[23]:
```
0.9827586206896551
```

[24]:
```
y_predict = model.predict(X_test)
y_predict
```

[24]:
```
array([0, 2, 0, 2, 1, 0, 0, 2, 0, 1, 1, 1, 1, 2, 0, 2, 0, 0, 0, 1, 2, 1,
       0, 0, 2, 2, 2, 2, 1])
```

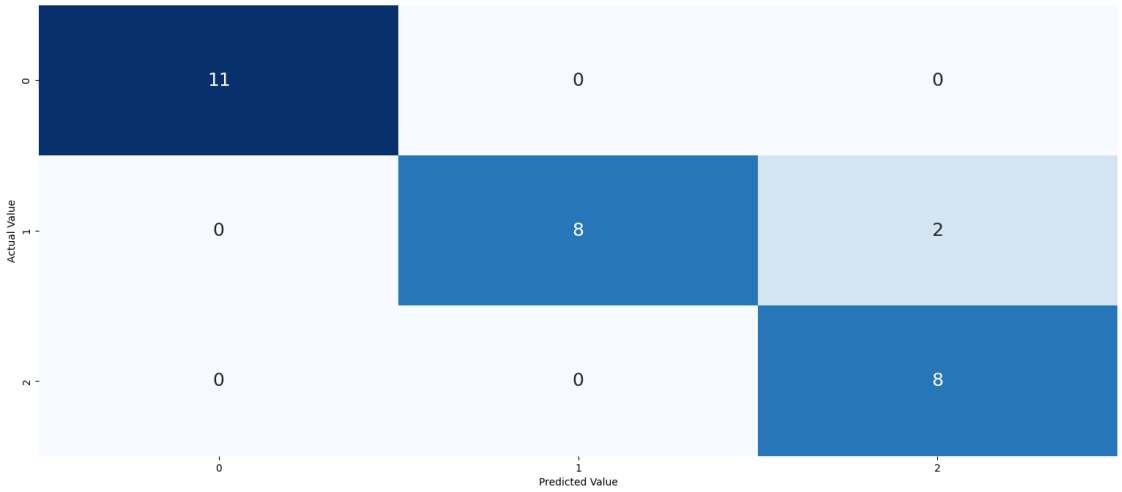
[25]:
```
y_test
```

[25]:
```
array([0, 1, 0, 2, 1, 0, 0, 2, 0, 1, 1, 1, 1, 1, 0, 2, 0, 0, 0, 1, 2, 1,
       0, 0, 2, 2, 2, 2, 1])
```

[26]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predict)
cm
```

[26]:
```
array([[11,  0,  0],
       [ 0,  8,  2],
       [ 0,  0,  8]])
```

[27]:
```
sns.heatmap(cm, annot=True, cmap='Blues', cbar=False, annot_kws={"fontsize":18})
plt.xlabel("Predicted Value")
plt.ylabel("Actual Value")
plt.show()
```


11
0
0
0
8
2
0
0
8
Actual Value
Predicted Value
0
1
2

[ ]: