Logistic_regression_withsvm

November 24, 2024

```
[2]: import pandas as pd
```

```
[8]: iris = pd.read_csv("iris.csv")
```

```
[10]: iris.head()
```

```
[10]:    SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
      0            5.1           3.5            1.4           0.2  Iris-setosa
      1            4.9           3.0            1.4           0.2  Iris-setosa
      2            4.7           3.2            1.3           0.2  Iris-setosa
      3            4.6           3.1            1.5           0.2  Iris-setosa
      4            5.0           3.6            1.4           0.2  Iris-setosa
```

```
[12]: iris.drop("Id",axis=1, inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[12], line 1
----> 1 iris.drop("Id",axis=1, inplace=True)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:5581, in DataFrame.
 ↪drop(self, labels, axis, index, columns, level, inplace, errors)
   5433 def drop(
   5434     self,
   5435     labels: IndexLabel | None = None,
   (…)
   5442     errors: IgnoreRaise = "raise",
   5443 ) -> DataFrame | None:
   5444     """
   5445     Drop specified labels from rows or columns.
   5446
   (…)
   5579             weight  1.0     0.8
   5580     """
-> 5581     return super().drop(
   5582         labels=labels,
   5583         axis=axis,
   5584         index=index,
   5585         columns=columns,
   5586         level=level,
   5587         inplace=inplace,
   5588         errors=errors,
   5589     )

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4788, in NDFrame.
 ↪drop(self, labels, axis, index, columns, level, inplace, errors)
   4786 for axis, labels in axes.items():
   4787     if labels is not None:
-> 4788         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4790 if inplace:
   4791     self._update_inplace(obj)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4830, in NDFrame.
 ↪_drop_axis(self, labels, axis, level, errors, only_slice)
   4828         new_axis = axis.drop(labels, level=level, errors=errors)
   4829     else:
-> 4830         new_axis = axis.drop(labels, errors=errors)
   4831     indexer = axis.get_indexer(new_axis)
   4833 # Case for non-unique axis
   4834 else:

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:7070, in Index.
 ↪drop(self, labels, errors)
   7068 if mask.any():
   7069     if errors != "ignore":
-> 7070         raise KeyError(f"{labels[mask].tolist()} not found in axis")
   7071     indexer = indexer[~mask]
   7072 return self.delete(indexer)

KeyError: "['Id'] not found in axis"
```

[18]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
import seaborn as sns
```

[19]:

```python
iris.shape
```

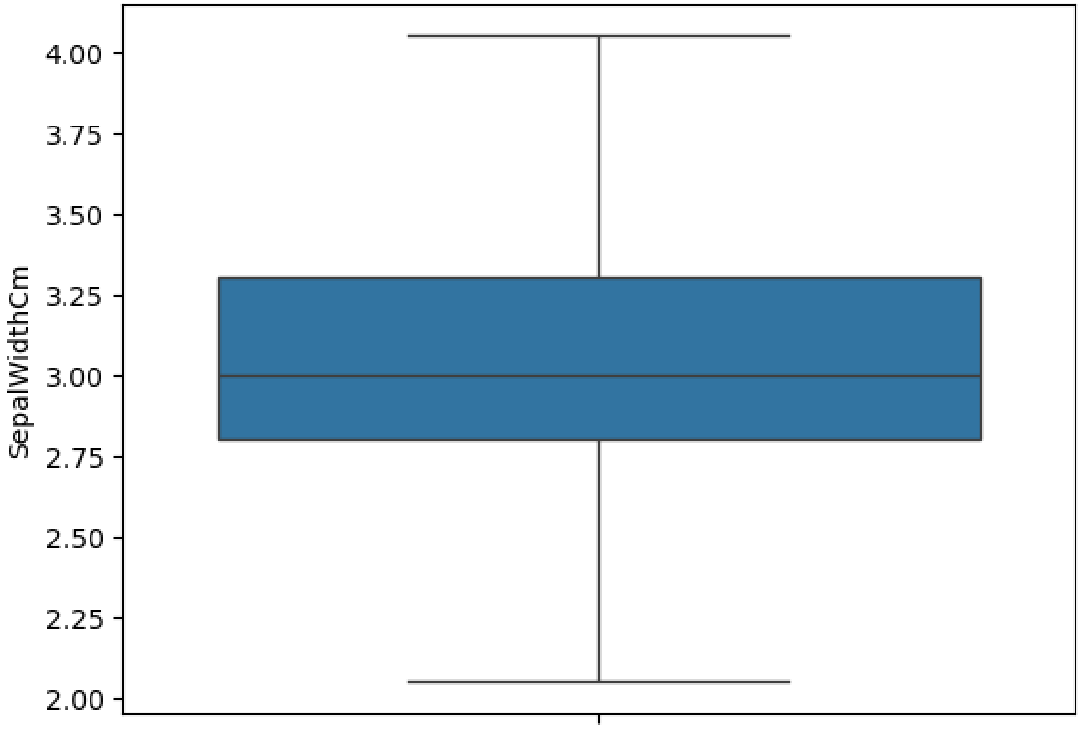
[19]:

```
(150, 5)
```

[35]:

```python
sns.boxplot(iris['SepalWidthCm'])
```

```
[35]: <Axes: ylabel='SepalWidthCm'>
```



```
[37]: train, test=train_test_split(iris, test_size=0.3)
```

```
[39]: train_X=train[['PetalWidthCm','PetalLengthCm','SepalWidthCm','SepalLengthCm']]
      train_y = train.Species

      test_X = test[['PetalWidthCm','PetalLengthCm','SepalWidthCm','SepalLengthCm']]
      test_y = test.Species
```

```
[41]: iris = LogisticRegression()
      iris.fit(train_X, train_y)
      prediction=iris.predict(test_X)


      from sklearn.metrics import accuracy_score
      print("accuracy of logestc regretion is..",accuracy_score(prediction,test_y))

      accuracy of logestc regretion is.. 0.9777777777777777
```

```
[43]: iris=svm.SVC()
      iris.fit(train_X,train_y)
      prediction=iris.predict(test_X)
```

```
print("the accuracy of svm is:", metrics.accuracy_score(prediction,test_y))
```

```
the accuracy of svm is: 0.9777777777777777
```

[45]:

```
iris = DecisionTreeClassifier()
iris.fit(train_X,train_y)
prediction=iris.predict(test_X)
print("the accuracy of dessision tree ie :",metrics.
 ↪accuracy_score(prediction,test_y))
```

```
the accuracy of dessision tree ie : 0.9111111111111111
```

[47]:

```
iris = KNeighborsClassifier(n_neighbors=4)
iris.fit(train_X,train_y)
prediction=iris.predict(test_X)
print("the accuracy of KNN is :",metrics.accuracy_score(prediction,test_y))
```

```
the accuracy of KNN is : 0.9555555555555556
```