iris_multi_linear_regression_assignment

January 19, 2023

1 Multilinear Reression

[ ]:
```
#Import Libraris
import pandas as pd
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import numpy as np
```

[ ]:
```
#Import dataset
iris = sns.load_dataset('iris')
```

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

[ ]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

[ ]:
```
iris.isnull().sum() / len(iris)*100
```

[ ]:
```
sepal_length    0.0
sepal_width     0.0
petal_length    0.0
petal_width     0.0
species         0.0
dtype: float64
```

[ ]:
```
iris.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  150 non-null    float64
 1   sepal_width   150 non-null    float64
```

```
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

[ ]:
```
iris = iris[['sepal_length','sepal_width','petal_length','petal_width']]
iris.head()
```

[ ]:

	sepal_length	sepal_width	petal_length	petal_width
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

[ ]:
```
X = iris[['sepal_length','sepal_width','petal_length']]
y = iris['petal_width']
```

[ ]:
```
model = LinearRegression().fit(X, y)
model
```

[ ]:
```
LinearRegression()
```

[ ]:
```
model.predict([[2.3, 3.4, 5.6]])
```

```
c:\Users\muham\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\base.py:409: UserWarning: X does not have valid feature names,
but LinearRegression was fitted with feature names
  warnings.warn(
```

[ ]:
```
array([2.97546313])
```

[ ]:
```
model.score
```

[ ]:
```
<bound method RegressorMixin.score of LinearRegression()>
```

[ ]:
```
reg = LinearRegression().fit(X_test, y_test)
```

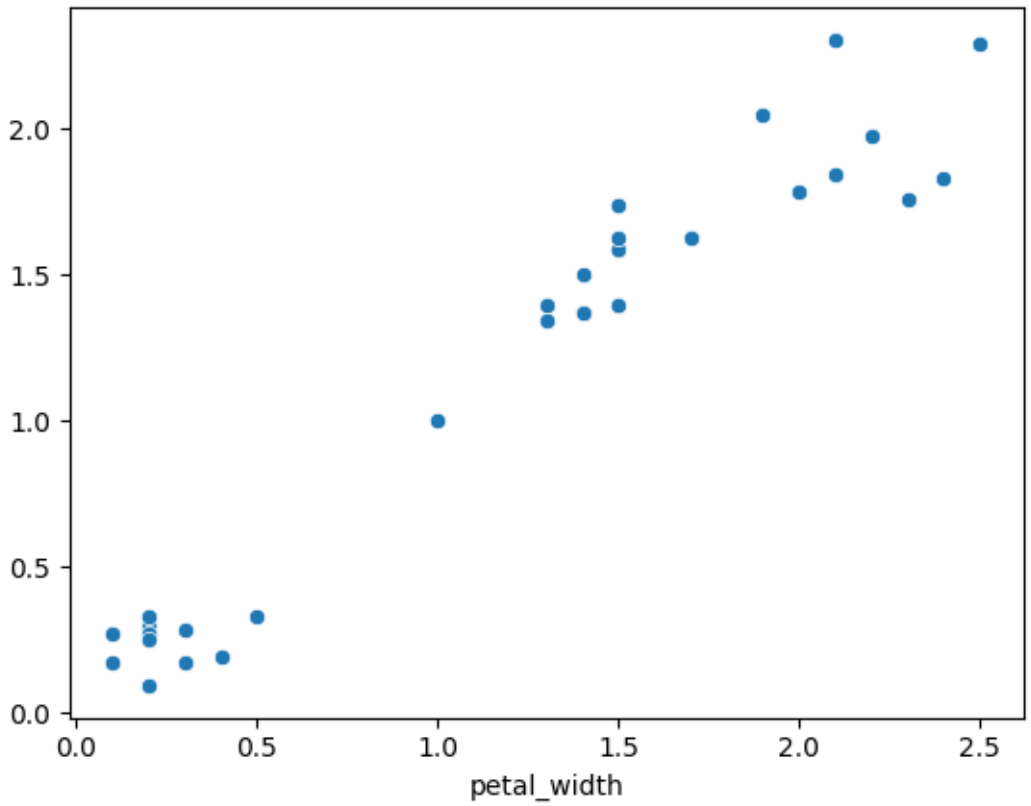
[ ]:
```
# Model evaluation
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.8)

model = LinearRegression()

model = LinearRegression().fit(X_train, y_train)
prediction = model.predict(X_test)
```

[ ]:
```
sns.scatterplot(x=y_test, y= prediction)
```

[ ]: <AxesSubplot: xlabel='petal_width'>



[ ]:
```
model.score(X_test, y_test)
```

[ ]: 0.9377546979291058

[ ]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

MAE = mean_absolute_error(y_true = y_test, y_pred = prediction)
MSE = mean_squared_error (y_true = y_test, y_pred = prediction)
RMSE = mean_squared_error (y_true = y_test, y_pred = prediction, squared=True)
R2 = r2_score(y_true = y_test, y_pred = prediction)
```

[ ]:
```
print('MAE: ', MAE)
print('MSE: ', MSE)
print('RMSE: ', RMSE)
print('R_squared: ', R2)

# Near to Zero is best value for MAE,MSE,RMSE and 1 is best for R2
```

```
MAE:  0.15091753986043377
MSE:  0.0395195422848107
```

```
RMSE:  0.0395195422848107
R_squared:  0.9377546979291058
```

2 Prediction has good resultd