CH : - 5 Regression Analysis

1.Supervised Learning ¶

Regression(integer or float)

- Linear
- 1.)simple
- 2.)multiple
- polynomial
- 1.)simple
- 2.)multiple

Classification(categorical data)

- KNN
- Decision Tree

equation

- y = mx+b (linear)
- y = a+bx+cx1+dx2+ex3 (linear)
- y = a+bx+cx^2+dx^3 (polynomial)
- y = a+bx1+cx2^2+dx3^3 (polynomial)

Simple Linear Regression

scikit library

- preprocessing

In [65]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.read_csv("Book1.csv")
df.head()
```

Out[65]:

	cgpa	package
0	6.89	3.26
1	5.12	1.98
2	7.82	3.25
3	7.42	3.67
4	6.94	3.57

In [66]:
```
x = df.iloc[:,0:1]
y = df.iloc[:,-1]
# x = df.drop("package",axis=1)
# x = df[['cgpa','x1','x2']]
# x = df['cgpa']
# y = df['package']
```

In [67]:
```
x.shape
```

Out[67]: (200, 1)

In [68]:
```
y.shape
```

Out[68]: (200,)

- processing

In [69]:
```
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)
x_train.shape
```

Out[69]: (160, 1)

In [70]:
```
help(train_test_split)
```

```
Help on function train_test_split in module sklearn.model_selection._split:

train_test_split(*arrays, **options)
    Split arrays or matrices into random train and test subsets

    Quick utility that wraps input validation and
    ``next(ShuffleSplit().split(X, y))`` and application to input data
    into a single call for splitting (and optionally subsampling) data in a
    oneliner.

    Read more in the :ref:`User Guide <cross_validation>`.

    Parameters
    ----------
    *arrays : sequence of indexables with same length / shape[0]
        Allowed inputs are lists, numpy arrays, scipy-sparse
        matrices or pandas dataframes.

    test_size : float or int, default=None
        If float, should be between 0.0 and 1.0 and represent the proportion
        of the dataset to include in the test split. If int, represents the
        absolute number of test samples. If None, the value is set to the
        complement of the train size. If ``train_size`` is also None, it wil
l
        be set to 0.25.

    train_size : float or int, default=None
        If float, should be between 0.0 and 1.0 and represent the
        proportion of the dataset to include in the train split. If
        int, represents the absolute number of train samples. If None,
        the value is automatically set to the complement of the test size.

    random_state : int or RandomState instance, default=None
        Controls the shuffling applied to the data before applying the spli
t.
        Pass an int for reproducible output across multiple function calls.
        See :term:`Glossary <random_state>`.


    shuffle : bool, default=True
        Whether or not to shuffle the data before splitting. If shuffle=Fals
e
        then stratify must be None.

    stratify : array-like, default=None
        If not None, data is split in a stratified fashion, using this as
        the class labels.

    Returns
    -------
    splitting : list, length=2 * len(arrays)
        List containing train-test split of inputs.

        .. versionadded:: 0.16
            If the input is sparse, the output will be a
            ``scipy.sparse.csr_matrix``. Else, output type is the same as th
e
            input type.

    Examples
    --------
```

```
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(10).reshape((5, 2)), range(5)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]

>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
...
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]

>>> train_test_split(y, shuffle=False)
[[0, 1, 2], [3, 4]]
```

In [71]:
```
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)
x_test.shape
```

Out[71]: (40, 1)

In [72]:
```
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)
y_train.shape
```

Out[72]: (160,)

In [73]:
```
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)
y_test.shape
```

Out[73]: (40,)

In [74]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
lr=LinearRegression()
model = lr.fit(x_train,y_train) # creating a model or equation
lr.coef_
```

Out[74]: array([0.56342815])

In [75]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
lr=LinearRegression()
model = lr.fit(x_train,y_train) # creating a model or equation
lr.intercept_
```

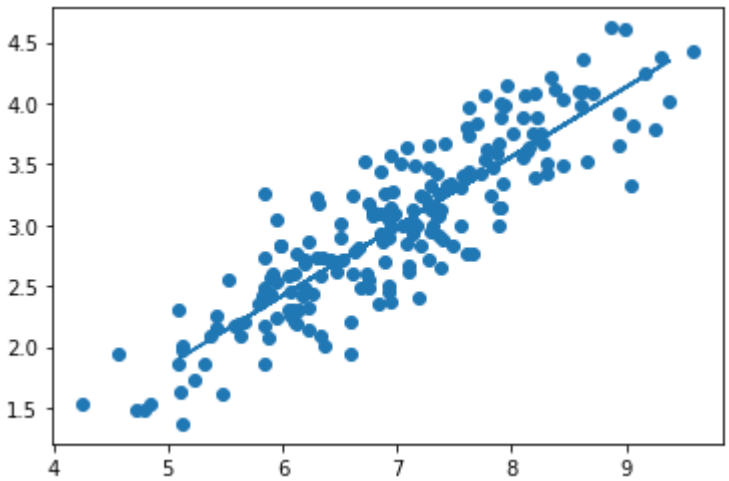
Out[75]:
```
-0.9147111511222294
```

- posting

In [76]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
lr=LinearRegression()
model = lr.fit(x_train,y_train) # creating a model or equation
y_pred = lr.predict(x_test)
y_pred
```

Out[76]:
```
array([1.74592711, 3.73167108, 2.78634508, 2.98212265, 2.60175479,
       1.94729833, 4.095258  , 3.0380591 , 2.79193872, 2.25494881,
       3.78201388, 3.44080154, 2.80871966, 2.97093536, 2.56259928,
       3.8211694 , 2.62412937, 1.96407927, 2.93177985, 2.71922134,
       2.22138694, 1.95848562, 2.98212265, 3.92185501, 4.12881987,
       2.52344376, 2.99330994, 2.52903741, 3.38486509, 2.95415443,
       3.27858583, 2.93737349, 2.98212265, 3.07721462, 3.21705574,
       2.79753237, 3.93863594, 2.56259928, 3.68692192, 4.30781651])
```

In [77]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
lr=LinearRegression()
model = lr.fit(x_train,y_train) # creating a model or equation
y_pred = lr.predict(x_test)
plt.scatter(df['cgpa'],df['package'])
plt.plot(x_test,y_pred)
plt.show()
```



In [78]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import math
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
lr=LinearRegression()
model = lr.fit(x_train,y_train)
y_pred = lr.predict(x_test)
print("MAE",metrics.mean_absolute_error(y_test,y_pred))
print("MSE",metrics.mean_squared_error(y_test,y_pred))
print("RMSE",math.sqrt(metrics.mean_squared_error(y_test,y_pred)))
print("R2score",metrics.r2_score(y_test,y_pred))
```

```
MAE 0.2429127222310849
MSE 0.1030403845741497
RMSE 0.3209990413913252
R2score 0.8058369354814652
```

In [79]:
```
import numpy as np
test_data = np.array([9.3,7.8])
print(test_data)
print(test_data.shape)
```

```
[9.3 7.8]
(2,)
```

In [80]:
```
import numpy as np
test_data = test_data.reshape(-1,1)
print(test_data)
print(test_data.shape)
```

```
[[9.3]
 [7.8]]
(2, 1)
```

In [81]:
```
lr.predict(test_data)
```

Out[81]:
```
array([4.28608987, 3.44111045])
```

In [ ]: