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 [100]:

```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.read_csv("Book1.csv")
df.head()
```

Out[100]:

	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 [101]:

```
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 [102]:

```
x.shape
```

Out[102]:

```
(200, 1)
```

In [103]:

```
y.shape
```

Out[103]:

```
(200,)
```

- processing

In [104]:

```
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[104]:

```
(160, 1)
```

```
In [105]: 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 [106]:

```
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[106]:

```
(40, 1)
```

In [107]:

```
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[107]:

```
(160,)
```

In [108]:

```
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[108]:

```
(40,)
```

In [109]:

```
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[109]:

```
array([0.56004401])
```

In [110]:
```
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_
```

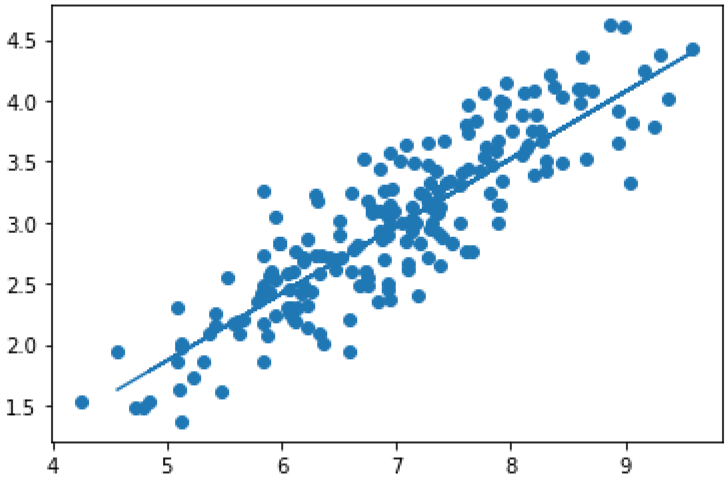
Out[110]:
```
-1.057454060871304
```

- posting

In [111]:
```
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[111]:
```
array([2.55731678, 3.16105985, 3.90881503, 2.9671974 , 2.3579154 ,
       3.6595633 , 2.97273633, 4.28546208, 3.49893441, 3.32168874,
       3.21644913, 3.49893441, 3.42692836, 1.95357372, 3.16659878,
       2.68471211, 2.92842491, 4.04174928, 3.07243702, 2.9671974 ,
       1.9701905 , 2.82872422, 3.53216798, 3.73156936, 2.13081939,
       2.61270606, 3.37153909, 2.98381418, 2.52962215, 3.43246729,
       3.35492231, 3.60971296, 2.65701747, 3.10567058, 2.9671974 ,
       2.97273633, 2.38561004, 2.96165847, 1.95911264, 3.52662905])
```

In [112]:
```
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 [113]:

```
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)) # x depend on y
```

```
MAE 0.28331977292553595
MSE 0.13470894990622276
RMSE 0.3670271787023718
R2score 0.6419528555237273
```

In [114]:

```
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 [115]:

```
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 [116]:

```
lr.predict(test_data)
```

Out[116]:

```
array([4.3834217 , 3.48369105])
```

Multiple Linear Regression

In [1]:

```
# Target is selling_price.
import pandas as pd
df = pd.read_csv("car data.csv")
df.isna().sum()
```

Out[1]:

```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

In [7]:

```
import pandas as pd
df = pd.read_csv("car data.csv")
df = df.drop("Car_Name",axis=1)
df
```

Out[7]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owı
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	
...	...	...	...	...	...	...	...	
296	2016	9.50	11.60	33988	Diesel	Dealer	Manual	
297	2015	4.00	5.90	60000	Petrol	Dealer	Manual	
298	2009	3.35	11.00	87934	Petrol	Dealer	Manual	
299	2017	11.50	12.50	9000	Diesel	Dealer	Manual	
300	2016	5.30	5.90	5464	Petrol	Dealer	Manual	

301 rows × 8 columns

In [8]:

```
df["Age"] = 2025-df["Year"]
df
```

Out[8]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owı
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	
...	...	...	...	...	...	...	...	
296	2016	9.50	11.60	33988	Diesel	Dealer	Manual	
297	2015	4.00	5.90	60000	Petrol	Dealer	Manual	
298	2009	3.35	11.00	87934	Petrol	Dealer	Manual	
299	2017	11.50	12.50	9000	Diesel	Dealer	Manual	
300	2016	5.30	5.90	5464	Petrol	Dealer	Manual	

301 rows × 9 columns

In [9]:
```
df = df.drop("Year",axis=1)
df
```

Out[9]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	A
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	
...	...	...	...	...	...	...	...	
296	9.50	11.60	33988	Diesel	Dealer	Manual	0	
297	4.00	5.90	60000	Petrol	Dealer	Manual	0	
298	3.35	11.00	87934	Petrol	Dealer	Manual	0	
299	11.50	12.50	9000	Diesel	Dealer	Manual	0	
300	5.30	5.90	5464	Petrol	Dealer	Manual	0	

301 rows × 8 columns

In [10]:
```
pd.get_dummies(df)
```

Out[10]:

IG	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Dealer	Seller_Type_Individual	Transmission_A
0	0	1	1	0	
0	1	0	1	0	
0	0	1	1	0	
0	0	1	1	0	
0	1	0	1	0	
...	...	...	...	...	
0	1	0	1	0	
0	0	1	1	0	
0	0	1	1	0	
0	1	0	1	0	
0	0	1	1	0	

In [11]:
```
df = pd.get_dummies(df,drop_first=True) # only one column remove from each cat
df
```

Out[11]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol
0	3.35	5.59	27000	0	11	0	1
1	4.75	9.54	43000	0	12	1	0
2	7.25	9.85	6900	0	8	0	1
3	2.85	4.15	5200	0	14	0	1
4	4.60	6.87	42450	0	11	1	0
...	...	...	...	...	...	...	...
296	9.50	11.60	33988	0	9	1	0
297	4.00	5.90	60000	0	10	0	1
298	3.35	11.00	87934	0	16	0	1
299	11.50	12.50	9000	0	8	1	0
300	5.30	5.90	5464	0	9	0	1

301 rows × 9 columns

In [12]:
```
# target is selling price
y = df['Selling_Price']
x = df.drop('Selling_Price',axis=1)
x.shape
```

Out[12]:
```
(301, 8)
```

In [13]:
```
y = df['Selling_Price']
x = df.drop('Selling_Price',axis=1)
y.shape
```

Out[13]:
```
(301,)
```

In [30]:
```
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_stat
x_train.shape
```

Out[30]:
```
(240, 8)
```

In [16]:
```
x_test.shape
```

Out[16]:
```
(61, 8)
```

In [17]:
```
y_train.shape
```

Out[17]:
```
(240,)
```

In [18]:
```
y_test.shape
```

Out[18]:
```
(61,)
```

In [31]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
model = lr.fit(x_train,y_train)
lr.coef_
```

Out[31]:
```
array([ 4.28802981e-01, -5.92316903e-06, -8.49009727e-01, -4.06411225e-01,
        2.86056285e+00,  7.03103550e-01, -1.07477347e+00, -1.50897975e+00])
```

In [32]:
```
print(x.columns)
```

```
Index(['Present_Price', 'Kms_Driven', 'Owner', 'Age', 'Fuel_Type_Diesel',
       'Fuel_Type_Petrol', 'Seller_Type_Individual', 'Transmission_Manual'],
      dtype='object')
```

In [33]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
model = lr.fit(x_train,y_train)
lr.intercept_
```

Out[33]:
```
6.850757321843104
```

In [52]:
```
from sklearn import metrics
import math
y_pred = lr.predict(x_test)
y_pred
```

Out[52]:
```
array([10.57889241,  0.71754255,  4.23613904,  5.17855444,  9.75533583,
        4.20615652,  2.6748089 ,  7.63221623,  0.17032381,  5.13283747,
        6.15727726,  6.44545506,  2.11900126,  7.6644102 ,  1.91710055,
        1.71680788,  2.02672159,  1.85294787,  9.56717323,  4.23076317,
        1.48724915,  9.36189904,  1.46103587,  9.54572517,  0.82954545,
        8.32266051,  1.53698563, -3.16293717,  4.2127377 ,  2.10385587,
        3.42279444,  3.71284702,  5.54538889,  7.6834828 , -1.89757368,
        7.06591847,  8.46027656,  5.77704136,  6.10531922,  6.52919752,
       16.08564344,  2.07410694,  1.05462465, -0.44710144,  7.06940454,
        6.73548589,  0.98609923,  7.08961114, 14.16420523,  3.00434598,
        8.2871215 , -0.87862345,  8.8814273 ,  1.15099681,  2.11848057,
       -0.82641923,  0.69287067,  9.88583992, -0.45444272, -2.41051663,
       10.28627971])
```

In [53]:
```
from sklearn import metrics
import math
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 :  1.2453565634870642
MSE :  2.750689262139753
RMSE :  1.658520202511791
R2score :  0.8502332355855595
```

In [54]:

```
# when column value is given
import numpy as np
test_data = np.array([6.7, 10000,0, 7, 1,0, 0, 1])
test_data = test_data.reshape(-1,8)
test_data
```

Out[54]:

```
array([[6.7e+00, 1.0e+04, 0.0e+00, 7.0e+00, 1.0e+00, 0.0e+00, 0.0e+00,
        1.0e+00]])
```

In [57]:

```
import numpy as np
test_data = np.array([6.7, 10000,0, 7, 1,0, 0, 1])
test_data = test_data.reshape(-1,8)
lr.predict(test_data)
```

Out[57]:

```
array([8.17121013])
```

Advertising csv

In [58]:

```
import pandas as pd
df = pd.read_csv("advertising.csv")
df.isna().sum()
```

Out[58]:

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [66]:

```
y = df["Sales"]
x = df.drop("Sales",axis=1)
```

In [67]:

```
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[67]:

```
(160, 3)
```

In [68]:

```
x_test.shape
```

Out[68]:

```
(40, 3)
```

In [69]:

```
y_train.shape
```

Out[69]:

```
(160,)
```

In [70]:

```
y_test.shape
```

Out[70]:

```
(40,)
```

```
In [74]: from sklearn.linear_model import LinearRegression
         lr = LinearRegression()
         model = lr.fit(x_train,y_train)
         lr.coef_
```

```
Out[74]: array([ 0.05413044,  0.1096093 , -0.00071742])
```

```
In [75]: from sklearn.linear_model import LinearRegression
         lr = LinearRegression()
         model = lr.fit(x_train,y_train)
         lr.intercept_
```

```
Out[75]: 4.601987623229277
```

```
In [76]: from sklearn.linear_model import LinearRegression
         lr = LinearRegression()
         model = lr.fit(x_train,y_train)
         y_pred = lr.predict(x_test)
         y_pred
```

```
Out[76]: array([14.06714329, 17.00172214, 20.74018701, 20.14958878, 21.0236248 ,
                16.76572535, 15.76327412,  8.03654982, 17.11011956, 21.34362676,
                11.28608236, 11.06766208, 15.39394793,  8.81462303, 24.55268079,
                17.07362544, 24.88331835, 10.42926552,  6.09130108, 24.92061384,
                 9.86831195, 13.35784443, 18.10206165, 10.148075  ,  9.51161825,
                10.5969708 , 15.79896773, 23.52774486, 20.65344938, 18.79356837,
                19.4127532 , 15.18136138, 19.20252368, 12.03755096, 20.51207209,
                15.53065475, 16.36903336, 24.50947859, 14.4166112 , 12.55345811])
```

```
In [77]: from sklearn import metrics
         import math
         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 :  1.383308862610568
MSE :  2.886920336419673
RMSE :  1.6990939751584293
R2score :  0.8982801816554089
```

Simple Ploynomial regression

In [78]:

```
import pandas as pd
df = pd.read_csv("temp.csv")
df
```

Out[78]:

	sno	Temperature	Pressure
0	1	0	0.0002
1	2	20	0.0012
2	3	40	0.0060
3	4	60	0.0300
4	5	80	0.0900
5	6	100	0.2700

In [80]:

```
df = df.drop("sno",axis=1)
df
```

Out[80]:

	Temperature	Pressure
0	0	0.0002
1	20	0.0012
2	40	0.0060
3	60	0.0300
4	80	0.0900
5	100	0.2700

In [81]:

```
y = df["Pressure"]
x = df.drop("Pressure",axis=1)
```

In [83]:

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
x_poly = poly.fit_transform(x)
x_poly
```

Out[83]:

```
array([[1.0e+00, 0.0e+00, 0.0e+00],
       [1.0e+00, 2.0e+01, 4.0e+02],
       [1.0e+00, 4.0e+01, 1.6e+03],
       [1.0e+00, 6.0e+01, 3.6e+03],
       [1.0e+00, 8.0e+01, 6.4e+03],
       [1.0e+00, 1.0e+02, 1.0e+04]])
```

In [84]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_poly,y,train_size=0.2)
```

```
In [85]: from sklearn.linear_model import LinearRegression
         lr = LinearRegression()
         lr.fit(x_train,y_train)
         lr.coef_

Out[85]: array([0., 0., 0.])
```

```
In [86]: lr.intercept_

Out[86]: 0.0002
```