FUNDAMENTALS OF MACHINE LEARNING IN DATA SCIENCE

CSIS 3290

CLASSIFICATION USING REGRESSION

IN SKLEARN

FATEMEH AHMADI

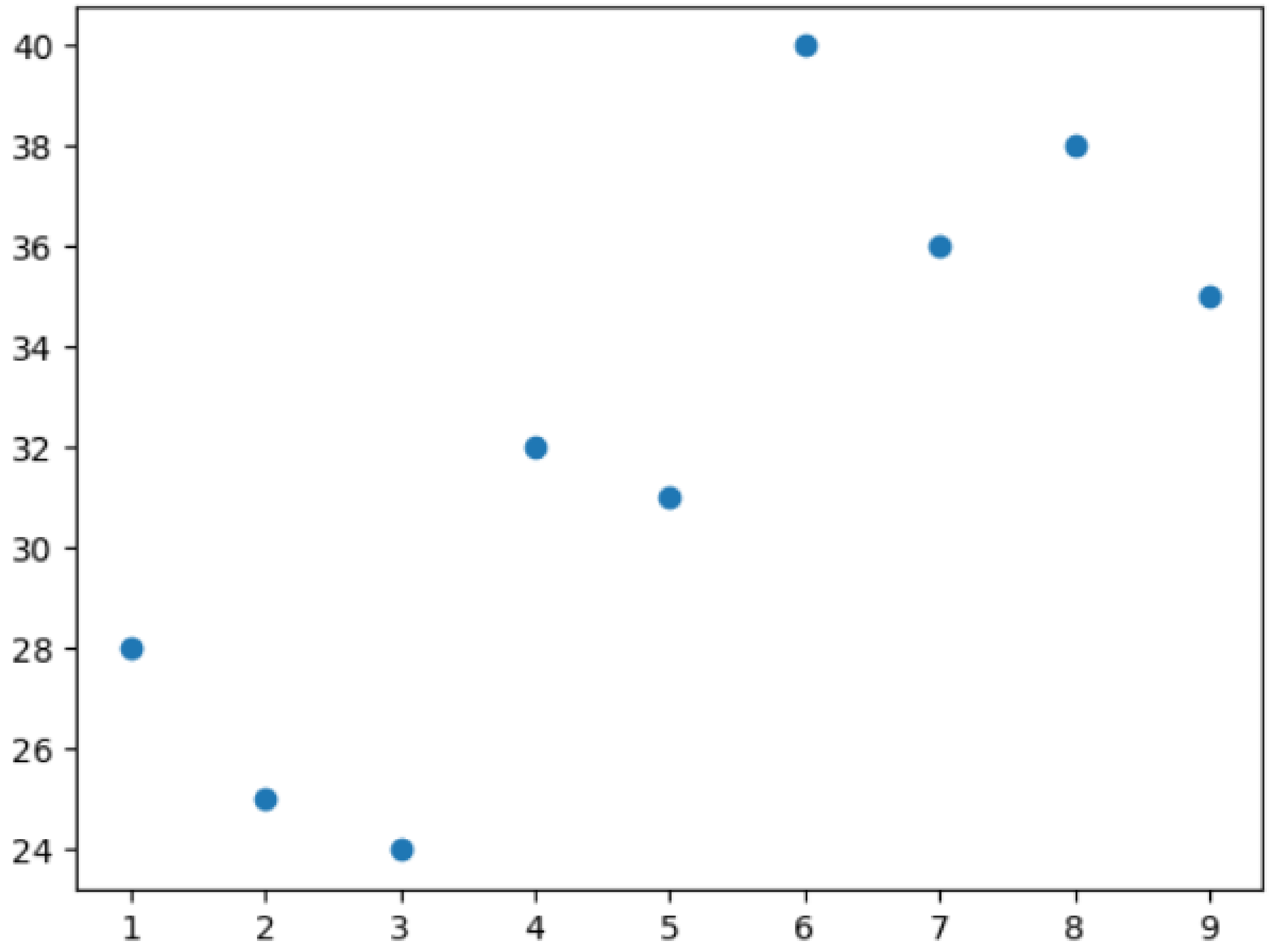
Linear Regression

```
In [62]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         from sklearn import datasets
         from sklearn.datasets import load_wine
         from sklearn.linear_model import LinearRegression
         from sklearn.linear_model import LogisticRegression
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import confusion_matrix, classification_report
         from sklearn.preprocessing import normalize
         from sklearn.metrics import mean_squared_error
         from sklearn.model_selection import cross_val_score
```

3

Linear Regression

```
In [43]: x1=np.arange(1,10)
         y1=np.array([28,25,24,32,31,40,36,38,35])
         plt.scatter(x1,y1)
         plt.show()
```



Linear Regression

Reshape(-1,1): is a **NumPy** function that reshapes an array with one column and as many rows as necessary to accommodate the data.

```
In [44]: x1=x1.reshape(-1,1)
         y1=y1.reshape(-1,1)

In [45]: x1

Out[45]: array([[1],
                [2],
                [3],
                [4],
                [5],
                [6],
                [7],
                [8],
                [9]])

In [46]: y1

Out[46]: array([[28],
                [25],
                [24],
                [32],
                [31],
                [40],
                [36],
                [38],
                [35]])
```

Linear Regression

```
In [47]: reg1=LinearRegression()

In [48]: reg1.fit(x1,y1)

Out[48]: ▾ LinearRegression
         LinearRegression()

In [49]: y_pred=reg1.predict(x1)

In [50]: y_pred

Out[50]: array([[25.51111111],
                [27.16111111],
                [28.81111111],
                [30.46111111],
                [32.11111111],
                [33.76111111],
                [35.41111111],
                [37.06111111],
                [38.71111111]])
```

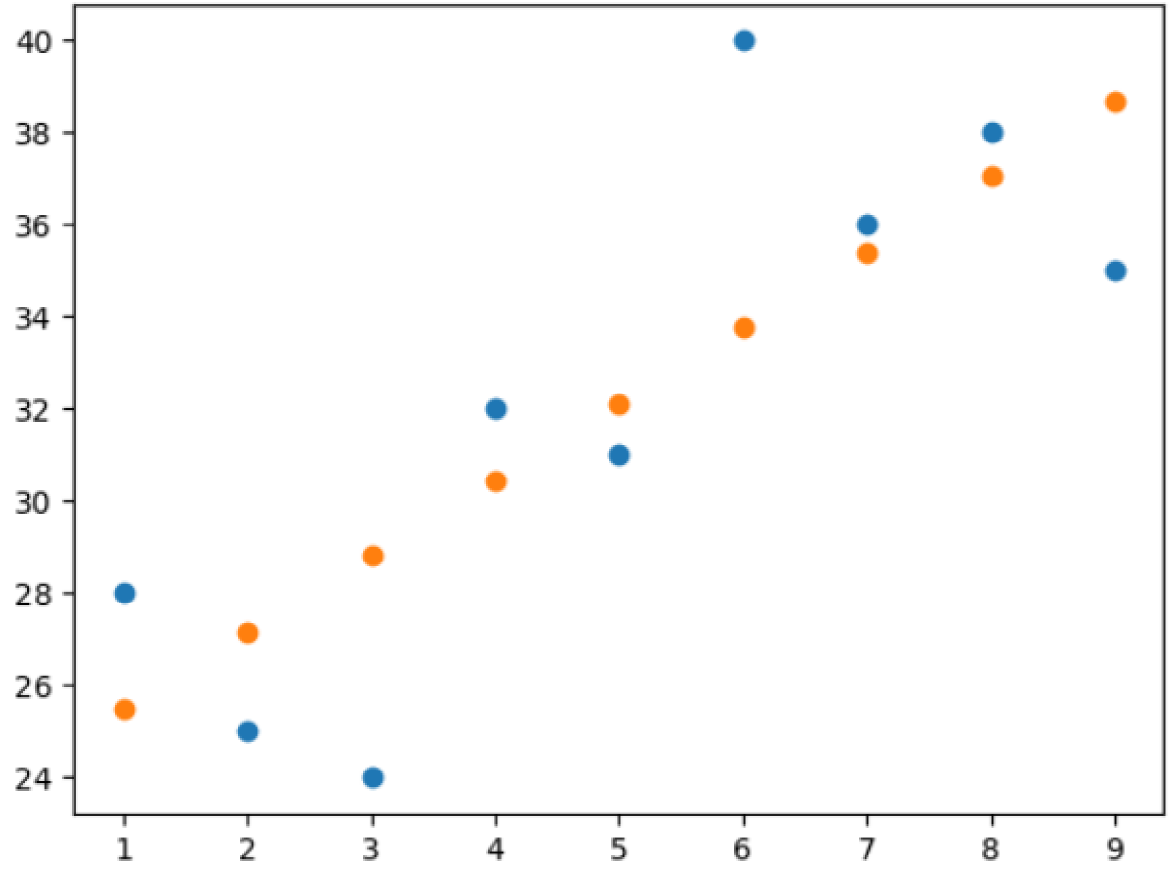
Linear Regression

```
In [60]: mse1=mean_squared_error(y1,y_pred)

In [61]: print(mse1)

10.170987654320983
```

```
In [51]: plt.scatter(x1,y1)
         plt.scatter(x1,y_pred)
         plt.show()
```



K-Fold Cross-Validation with a Linear Regression Model

```
In [63]: reg3=LinearRegression()

In [66]: cv1_score=cross_val_score(reg3,x1,y1,cv=4)

In [67]: print(cv1_score)

[-11.70841444  -6.50948487  -6.52740211  -9.67375283]

In [68]: print(np.mean(cv1_score))

-8.604763563381962
```

Number of K

Four MSE

Average values for MSE

Logistic Regression


```
In [31]: import pandas as pd
         from sklearn import datasets
         from sklearn.datasets import load_wine
         from sklearn.linear_model import LinearRegression
         from sklearn.linear_model import LogisticRegression
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import confusion_matrix, classification_report
         from sklearn.preprocessing import normalize
```

Logistic Regression

```
In [12]: data1=load_wine()
         wineDF=pd.DataFrame(data1.data, columns=data1.feature_names)
         wineDF['target']=data1.target

In [25]: x=data1.data
         y=data1.target
         data1.data
         y
         data1.target_names

Out[25]: array(['class_0', 'class_1', 'class_2'], dtype='<U7')

In [19]: x_train, x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
```

Logistic Regression

```
In [20]: reg2=LogisticRegression()

In [21]: reg2.fit(x_train,y_train)

D:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

Out[21]: ▾ LogisticRegression
         LogisticRegression()

In [22]: pre2=reg2.predict(x_test)

In [29]: cm=confusion_matrix(y_test,pre2)

In [30]: print(cm)

[[18  1  0]
 [ 0 21  0]
 [ 0  0 14]]
```

Logistic Regression

L1: It may be defined as the normalization technique that modifies the dataset values in a way that in each row the **sum of the absolute values will always be up to 1**. It is also called **Least Absolute Deviations**.

```
In [29]: cm=confusion_matrix(y_test,pre2)

In [30]: print(cm)

[[18  1  0]
 [ 0 21  0]
 [ 0  0 14]]

In [33]: cm1=normalize(cm,norm='l1',axis=1)

In [34]: cm1Df=pd.DataFrame(cm1,columns=data1.target_names,index=data1.target_names)

In [36]: print(cm1Df)

          class_0   class_1  class_2
class_0  0.947368  0.052632      0.0
class_1  0.000000  1.000000      0.0
class_2  0.000000  0.000000      1.0
```

norm{'l1', 'l2', 'max'}, default='l2'

L1 and L2 normalization are methods to keep the coefficients of the model small and reduce the complexity of the model. <u>L1-norm minimizes the sum of the absolute differences between the target value and the estimated values</u>. <u>L2-norm minimizes the sum of the squared differences between the target value and the estimated values</u>. <u>L1-norm is more robust to outliers, while L2-norm optimizes the mean cost</u>.