**Training And Testing Available Data**

**We have a dataset containing prices of used BMW cars. We are going to analyze this dataset and build a prediction function that can predict a price by taking mileage and age of the car as input. We will use sklearn train\_test\_split method to split training and testing dataset**

**import** pandas **as** pd

df **=** pd**.**read\_csv("carprices.csv")

df**.**head()

|  | **Mileage** | **Age(yrs)** | **Sell Price($)** |
| --- | --- | --- | --- |
| **0** | 69000 | 6 | 18000 |
| **1** | 35000 | 3 | 34000 |
| **2** | 57000 | 5 | 26100 |
| **3** | 22500 | 2 | 40000 |
| **4** | 46000 | 4 | 31500 |

**import** matplotlib.pyplot **as** plt

**%matplotlib** inline

|  |  |
| --- | --- |
| **Car Mileage Vs Sell Price ($)**  plt**.**scatter(df['Mileage'],df['Sell Price($)']) | **Car Age Vs Sell Price ($)**  plt**.**scatter(df['Age(yrs)'],df['Sell Price($)']) |
| Chart, scatter chart  Description automatically generated | Chart, scatter chart  Description automatically generated |

**Linear Regression pattern**

**The approach we are going to use here is to split available data in two sets**

1. **Training: We will train our model on this dataset**
2. **Testing: We will use this subset to make actual predictions using trained model**

**The reason we don't use same training set for testing is because our model has seen those samples before, using same samples for making predictions might give us wrong impression about accuracy of our model. It is like you ask same questions in exam paper as you tought the students in the class.**

X **=** df[['Mileage','Age(yrs)']]

y **=** df['Sell Price($)']

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y,test\_size**=**0.3)

X\_train y y\_train tiene el 70%

X\_test y y\_test tiene el 30%

**Lets run linear regression model now**

**from** sklearn.linear\_model **import** LinearRegression

clf **=** LinearRegression()

clf**.**fit(X\_train, y\_train)

clf**.**predict(X\_test)

Out[26]:

array([ 38166.23426912, 25092.95646646, 16773.29470749, 24096.93956163,

22602.44614295, 15559.98266172])

clf**.**score(X\_test, y\_test)

Out[27]:

0.92713

**random\_state argument**

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y,test\_size**=**0.3,random\_state**=**10)

X\_test

Out[35]:

|  | **Mileage** | **Age(yrs)** |
| --- | --- | --- |
| **7** | 72000 | 6 |
| **10** | 83000 | 7 |
| **5** | 59000 | 5 |
| **6** | 52000 | 5 |
| **3** | 22500 | 2 |
| **18** | 87600 | 8 |

random\_state is used for initializing the internal random number generator, which will decide the splitting of data into train and test indices in your case. In the [documentation](http://scikit-learn.org/stable/developers/utilities.html), it is stated that:

If random\_state is None or np.random, then a randomly-initialized RandomState object is returned.

If random\_state is an integer, then it is used to seed a new RandomState object.

when random\_state set to an **integer**, train\_test\_split will return **same** results for each execution.

when random\_state set to an **None**, train\_test\_split will return **different** results for each execution.

from sklearn.model\_selection import train\_test\_split

X\_data = range(10)

y\_data = range(10)

for i in range(5):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_data, y\_data, test\_size = 0.3,random\_state = 0) # zero or any other integer

print(y\_test)

print("\*"\*30)

for i in range(5):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_data, y\_data, test\_size = 0.3,random\_state = None)

print(y\_test)

when random\_state set to an **integer**, train\_test\_split will return **same** results for each execution.

when random\_state set to an **None**, train\_test\_split will return **different** results for each execution.

see below example:

from sklearn.model\_selection import train\_test\_split

X\_data = range(10)

y\_data = range(10)

for i in range(5):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_data, y\_data, test\_size = 0.3,random\_state = 0) # zero or any other integer

print(y\_test)

print("\*"\*30)

for i in range(5):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_data, y\_data, test\_size = 0.3,random\_state = None)

print(y\_test)

**Output**:

[2, 8, 4]

[2, 8, 4]

[2, 8, 4]

[2, 8, 4]

[2, 8, 4]

[4, 7, 6]

[4, 3, 7]

[8, 1, 4]

[9, 5, 8]

[6, 4, 5]