Installing SKLEARN

conda install scikit-learn #anaconda users

Pip install scikit-learn

Simple Linear Regression.

IT is the very basic machine learning algorithm to create model and train.

It's based on the equation $y = mx + c$

y = dependent variable

x = independent variable

m = slope

c = y intercept (the value of y when x is 0).

m is the slope of the line (how much y changes for a unit change in x).

Diameter (x)	Price (Y)	Mean (X)	Mean (Y)	Deviations (X)	Deviation (Y)	Product of Deviation	Sum of Product of Deviation	Square of Deviations for X
8	10	10	13	-2	-3	6	12	4
10	13			0	0	0		0
12	16			2	3	6		4

Calculate m = Sum of product of deviations / Sum of square of deviation for X

$= \frac{12}{8} = 1.5$

~~$Y = mX + C$~~

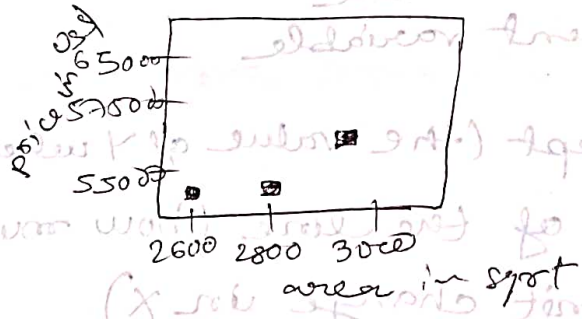
Calculate c = Mean of Y − (m * Mean of X)

$= 13 - (1.5 * 10)$

$= 13 - 15 = -2$

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn import linear_model
data = pd.read_csv('homeprices.csv')
data
```

```
plt.xlabel('area in sqft')
plt.ylabel('prices in US$')
plt.scatter(data.area, data.price, marker='s',
            color='red')
```



Create the model and train the data..

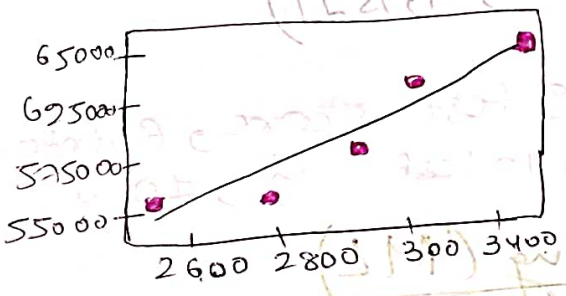
fit ⇒ for train the data

```
reg_model = linear_model.LinearRegression()
reg_model.fit(data[['area']], data.price)
```

```
print(reg-model.coef_)
print (reg-model.intercept_)
predict -> predict future data
reg-model.predict([[2800],[3000]])
```

plotting

```
pre = reg-model.predict(data[['area']])
plt.xlabel('area in sqft')
plt.ylabel('price in US$')
plt.scatter(data.area, data.price, marker='s',
            color='red')
plt.plot(data.area, pre, color='black')
```



Linear regression with multiple variable

It is more than one independent model. i.e the dependent variable value can predict based on more than one independent variable

e.g

area	bedrooms	age	price
2600	3	20	55000

Equation

$$y = m_1x_1 + m_2x_2 + m_3x_3 + b$$

$$price = m_1 area + m_2 bedrooms + m_3 age$$

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn import linear_model
data = pd.read_csv('homeprices.csv')
data
```

Data preprocessing

```
median_ = data.bedrooms.median()
# data.bedrooms = data.bedrooms.fillna(median_
data['bedrooms'].loc[2] = median_
data
```

Create, train and predict

```
model = linear_model.LinearRegression()
model.fit(data[['area', 'bedrooms', 'age']], data.price)
model.predict([[3000, 4, 15]])
```

⑧ 8 yr experience, 9 test score, 6 interview score
12 yr experience, 10 test " , 10 " "

Machine Learning (ML)

Machine learning consists of 2 steps typically.

① train the model.
② Ask questions to the model.

The more you train the model, the model will be more accurate But if the training data is so huge then the training time will be high. So every time training a model is time consuming process.

Saving the Trained Model....

Save a model using pickle model. By using dump function and load.

1) import pickle

```
with open('pickle_model', 'wb') as f:
    pickle.dump(reg_model, f)
with open('pickle_model', 'rb') as f:
    mpdel_ = pickle.load(f)
model.predict([[2000]])
```

2) Saving using joblib module

```
from sklearn.externals import joblib
joblib.dump(reg_model, 'joblib_model')
new_model = joblib.load('joblib_model')
new_model.predict([[30000]])
```

Dummy variables and One hot encoding.

Consider situation like in our data that we are using to train the model, is not number and the reality is it always will not be

town	area	price
Mumbai	2600	55000
"	3000	56500
Delhi	4000	99500

Create the

Create the Data

```
data = pd.read_csv('homeprices.csv')
dummies = pd.get_dummies(data.town)
merge = pd.concat([data, dummies], axis=1)
merge = merge.drop(['town'], axis=1)
merge
```

Train the model

```
X = merge.drop(['price'], axis=1)
Y = merge.price
model = linear_model.LinearRegression()
model.fit(X,Y)
model.predict([[2000, 0, 1, 0]])
```

Finding how accurate your model?

```
model.score(X,Y)
```

Using 1 hot encoder..

Creating X and Y

```
df = data
df
from sklearn.preprocessing import OneHotEncoder
Y = df.price

from sklearn.compose import ColumnTransformer
ts = ColumnTransformer(
    transformers = [
        ("abc", OneHotEncoder(), [1])
    ],
    remainder = 'passthrough')
x = ts.fit_transform(X)
```

Train And Predict

```
model = linear_model.LinearRegression()
model.fit(X,Y)
model.predict([[1, 0, 0, 2000]])
```

8) 1) Predict price of a mercedes benz that is 4yr old with mileage 45000.

2) Predict price of a BMW X5 that is 7yrs old with mileage 86000.

3) Tell me the score (accuracy) of your model.
(Hint: use LinearRegression().score())

Split the data to train and test...

The good practice in machine learning is split the data into 2 parts.
The first part is for train the model.
The second part is for test the model.

```
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.model_selection import train_test_split.
data = pd.read_csv('carprices.csv')
x = data [['Mileage', Age(yrs)']]
y = data [['Sell Price($)']]
xtrain, xtest, ytrain, ytest = train_test_split(x,y,
                                    test_size=.3)
```

Model and predict

```
model = linear_model.LinearRegression()
model.fit(xtrain, ytrain)
model.predict(xtest)
ytest
```