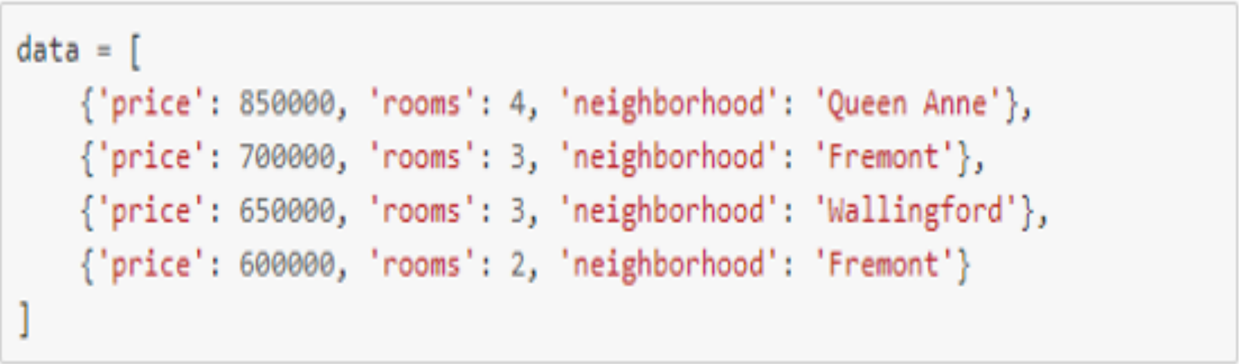
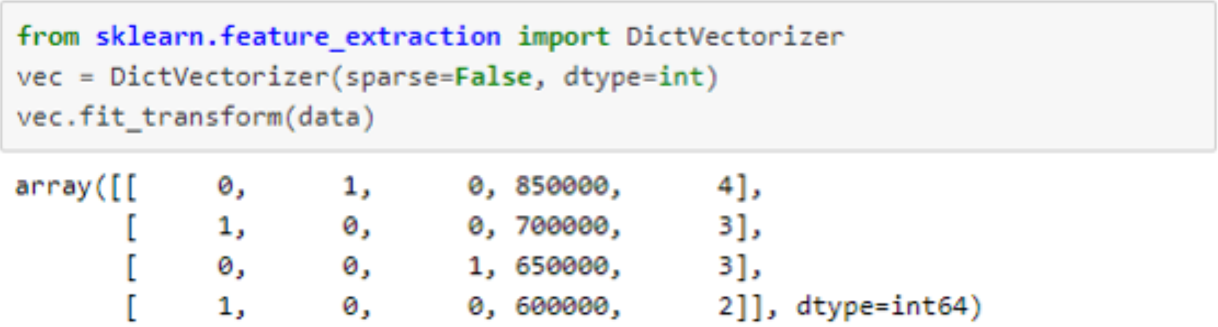


```
data = [
    {'price': 850000, 'rooms': 4, 'neighborhood': 'Queen Anne'},
    {'price': 700000, 'rooms': 3, 'neighborhood': 'Fremont'},
    {'price': 650000, 'rooms': 3, 'neighborhood': 'Wallingford'},
    {'price': 600000, 'rooms': 2, 'neighborhood': 'Fremont'}
]
```

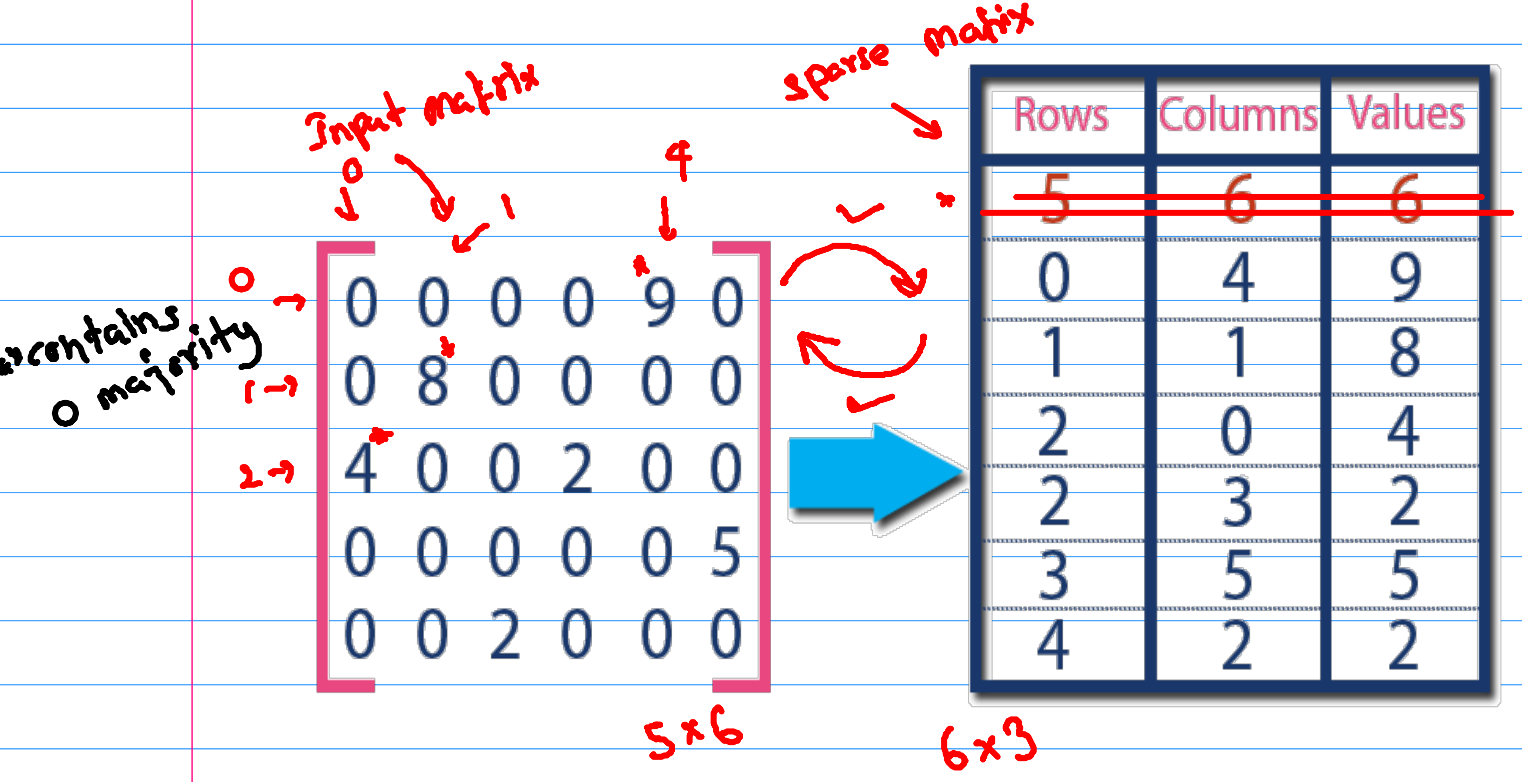


```
from sklearn.feature_extraction import DictVectorizer
vec = DictVectorizer(sparse=False, dtype=int)
vec.fit_transform(data)
```

```
array([[     0,      1,      0, 850000,      4],
       [     1,      0,      0, 700000,      3],
       [     0,      0,      1, 650000,      3],
       [     1,      0,      0, 600000,      2]], dtype=int64)
```

Queen Ann - [0 1 0]   wallingford - [0 0 1]
Fremont - [1 0 0]

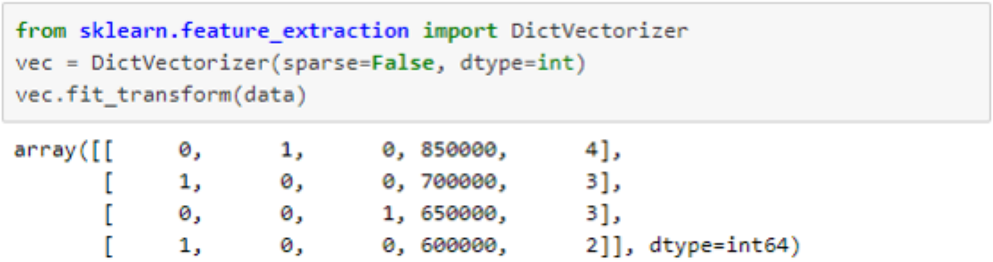
Sparse matrix.



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix}$$

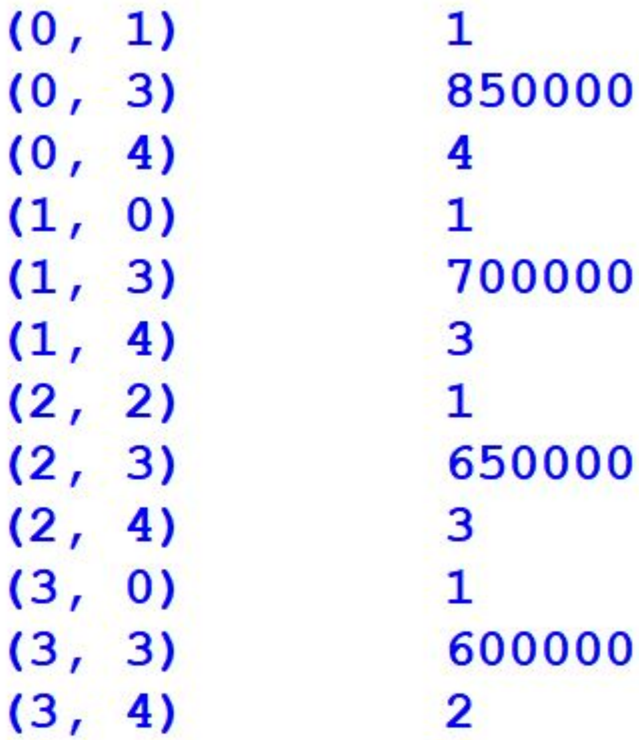
Rows	Columns	Values
~~5~~	~~6~~	~~6~~
0	4	9
1	1	8
2	0	4
2	3	2
3	5	5
4	2	2

when sparse true



```
from sklearn.feature_extraction import DictVectorizer
vec = DictVectorizer(sparse=False, dtype=int)
vec.fit_transform(data)
```

```
array([[      0,       1,       0, 850000,       4],
       [      1,       0,       0, 700000,       3],
       [      0,       0,       1, 650000,       3],
       [      1,       0,       0, 600000,       2]], dtype=int64)
```



```
(0, 1)        1
(0, 3)        850000
(0, 4)        4
(1, 0)        1
(1, 3)        700000
(1, 4)        3
(2, 2)        1
(2, 3)        650000
(2, 4)        3
(3, 0)        1
(3, 3)        600000
(3, 4)        2
```

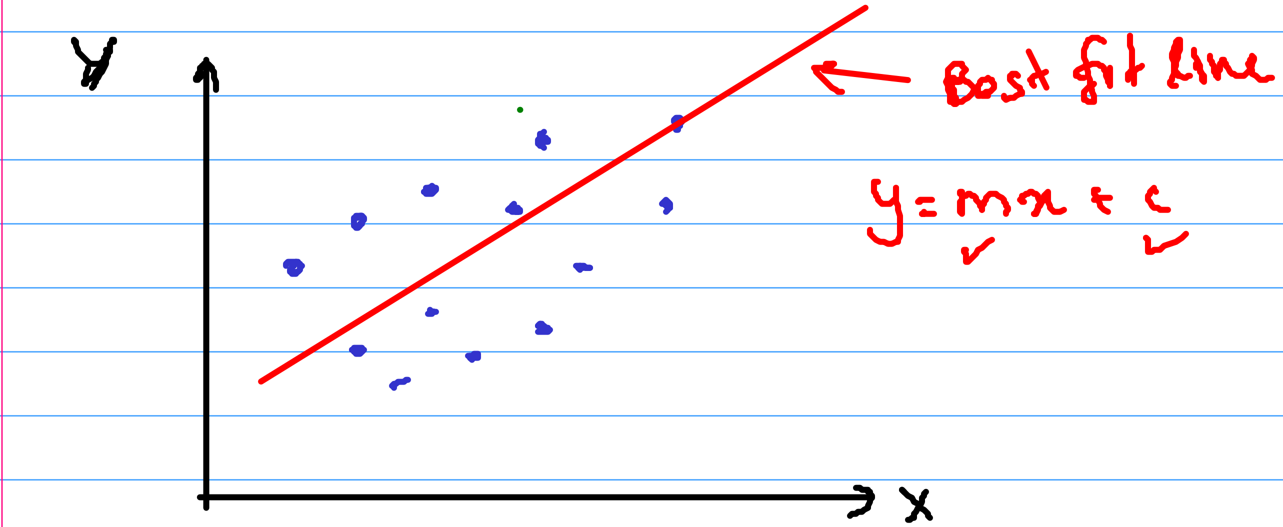
machine Learning $Y \rightarrow$ label x - Features

$$\begin{tikzcd} y_0 & & \\ y_1 & y \arrow[lu] \arrow[l] \arrow[ld] = \overset{*}{f}(x) \arrow[r] & x_0, x_1, x_2, x_3 \\ y_2 & & \end{tikzcd}$$

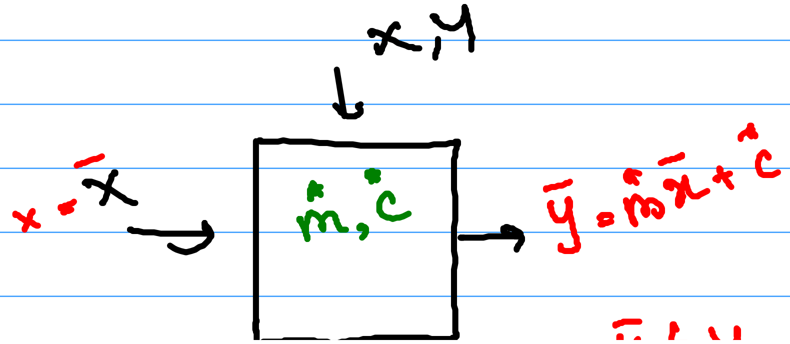
example

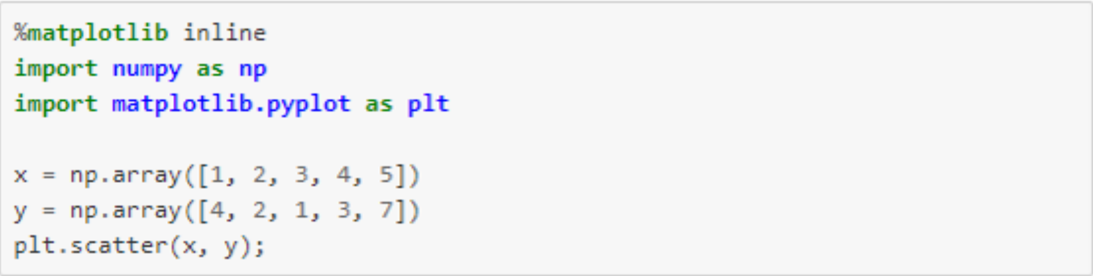
$$y = \overset{\checkmark}{a_0}\overset{*}{x_0} + \overset{\checkmark}{a_1}\overset{*}{x_1} + \overset{\checkmark}{a_2}\overset{*}{x_2} + \overset{\checkmark}{a_3}\overset{*}{x_3}$$

Linear Regression

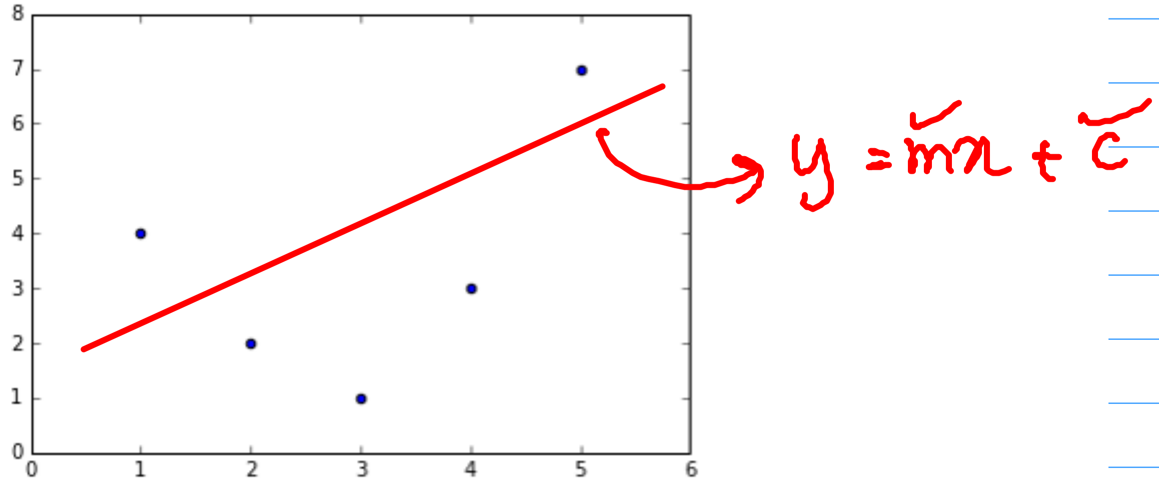


Case I

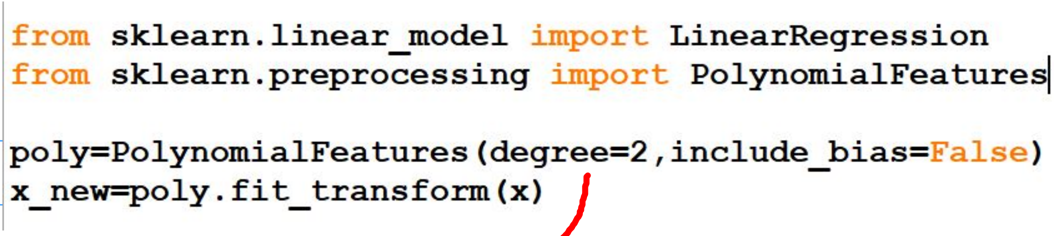

x, Y
$x = \bar{x}$
$\overset{*}{m}, \overset{*}{c}$
$\bar{y} = \hat{m}\bar{x} + \hat{c}$



```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4, 5])
y = np.array([4, 2, 1, 3, 7])
plt.scatter(x, y);
```

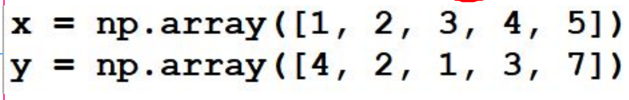


Case II.



```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

poly=PolynomialFeatures(degree=2,include_bias=False)
x_new=poly.fit_transform(x)
```

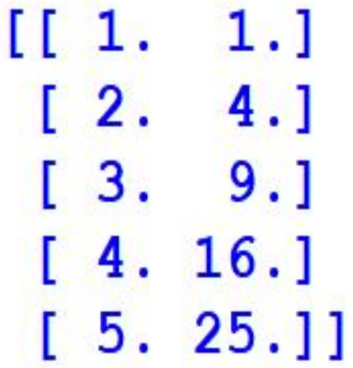
x_new



```
x = np.array([1, 2, 3, 4, 5])
y = np.array([4, 2, 1, 3, 7])
```

x

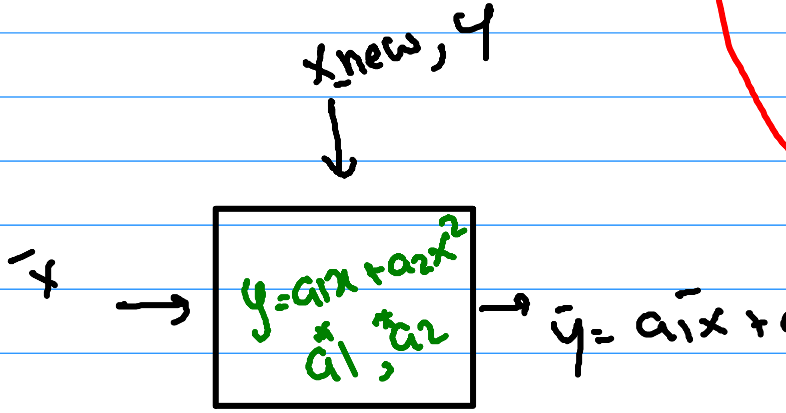
x_new



```
[[ 1.   1.]
 [ 2.   4.]
 [ 3.   9.]
 [ 4.  16.]
 [ 5.  25.]]
```

x

x^2



$\bar{y} = a_1^* x + a_2^* x + c$ → bias

optional

Installing Anaconda.

1. Download & Install anaconda for python 3.7
2. Open anaconda prompt
3. Run command conda install python=3.6
4. Install general libraries.

pip install sklearn ✓ ✓ ✓ ✓ } anaconda prompt

PCA.

variance Ranking

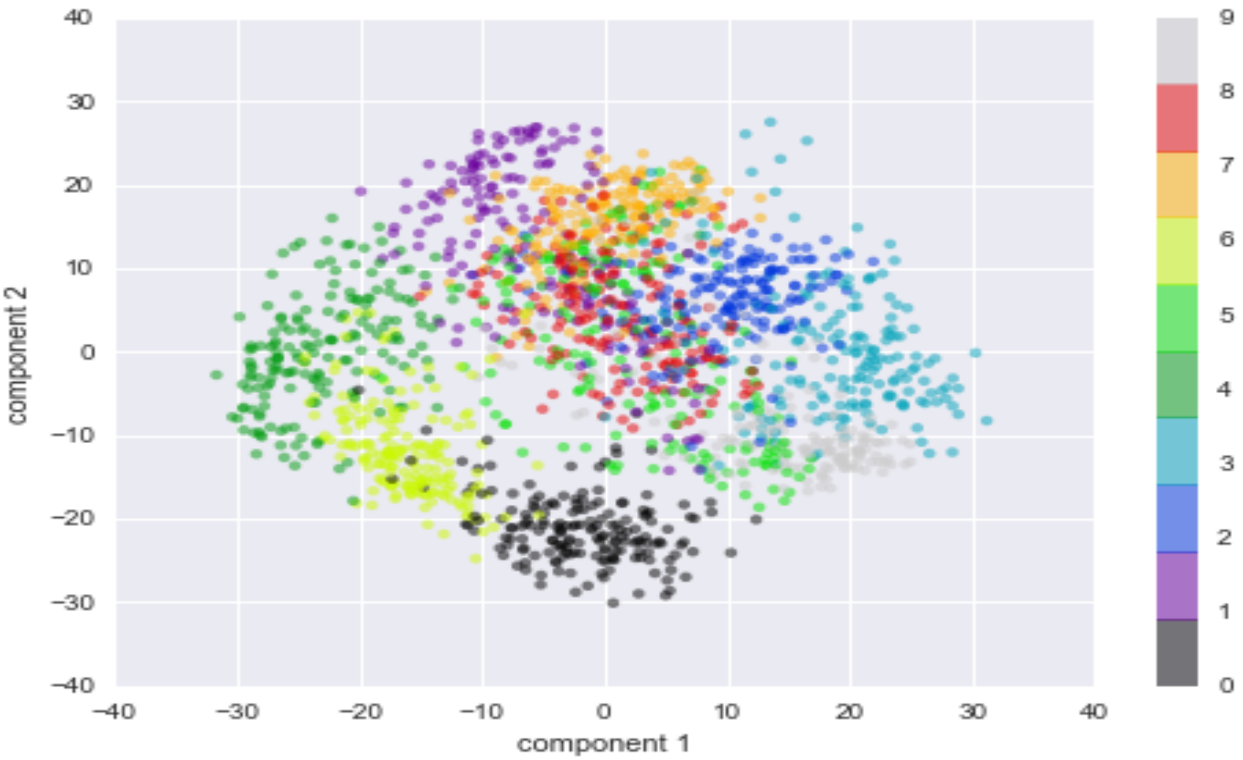
① ③ ④ ②

5.1 S/W 2.4 S/L P/W P/L ← features.

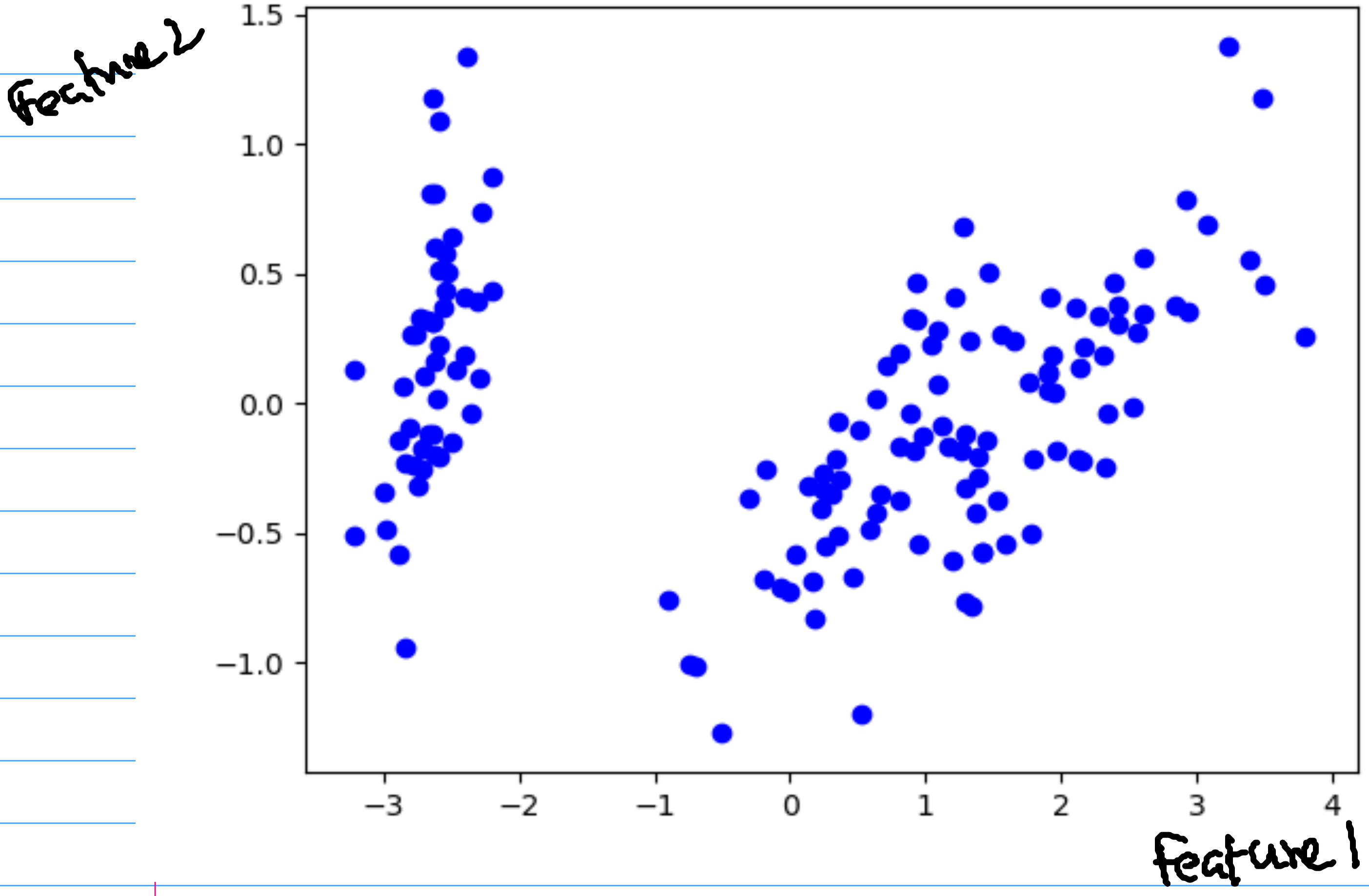
80% 20% ↓ PCA 70/ 30%

F1 F2.

PCA - 2 components to digits dataset.



unsupervised ML → No labels



ML

Supervised

- classification
- Regression

✓

(train)
Features ✓
labels ✓

Unsupervised

Clustering

Kmeans

Meanshift
(today)

Dimensionality
Reduction
(later)

features ✓
labels ✗
(train)

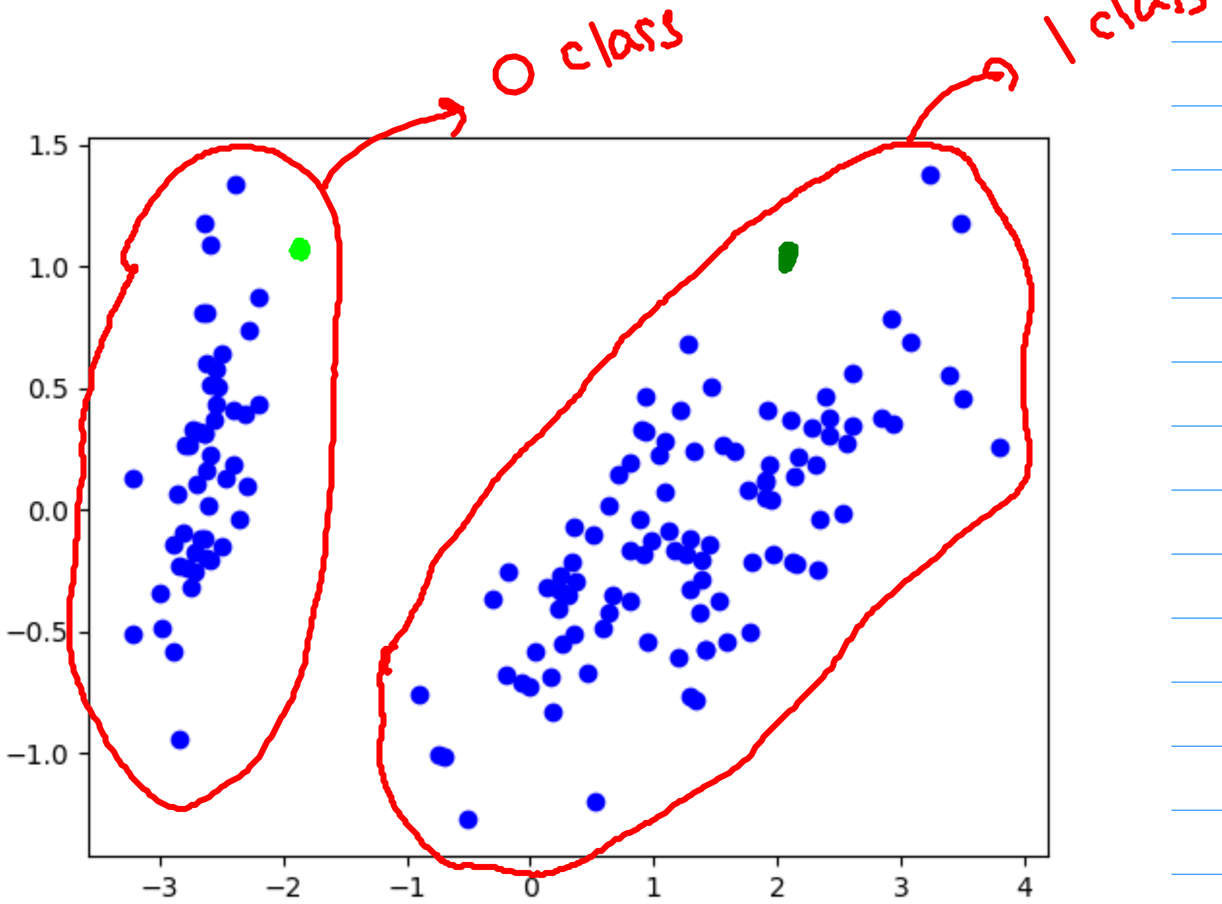
Reinforcement

Next week

Kmeans Clustering

n_clusters to be given

* n_cluster = 2

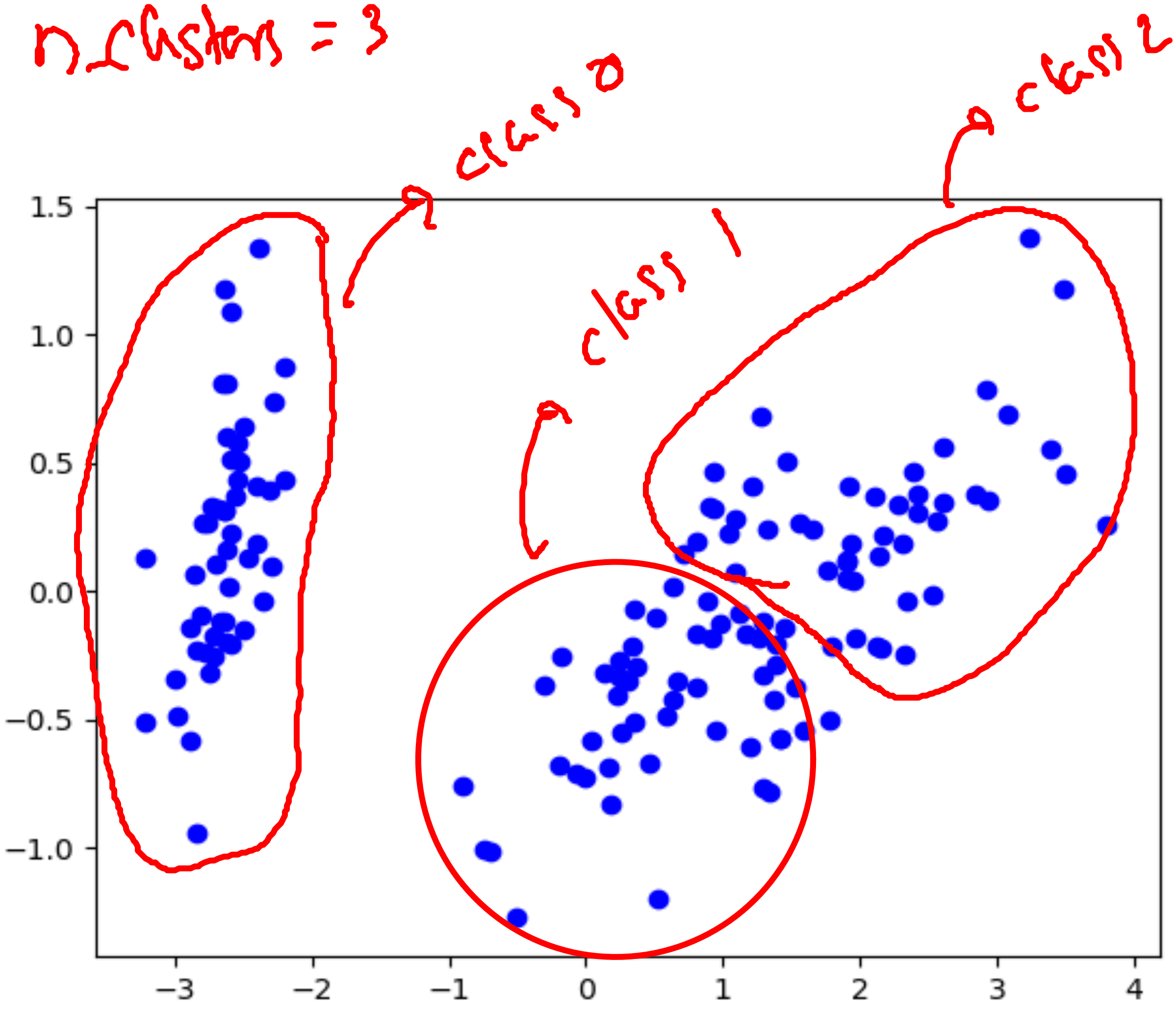


steps

(1) labelling data (clustering)

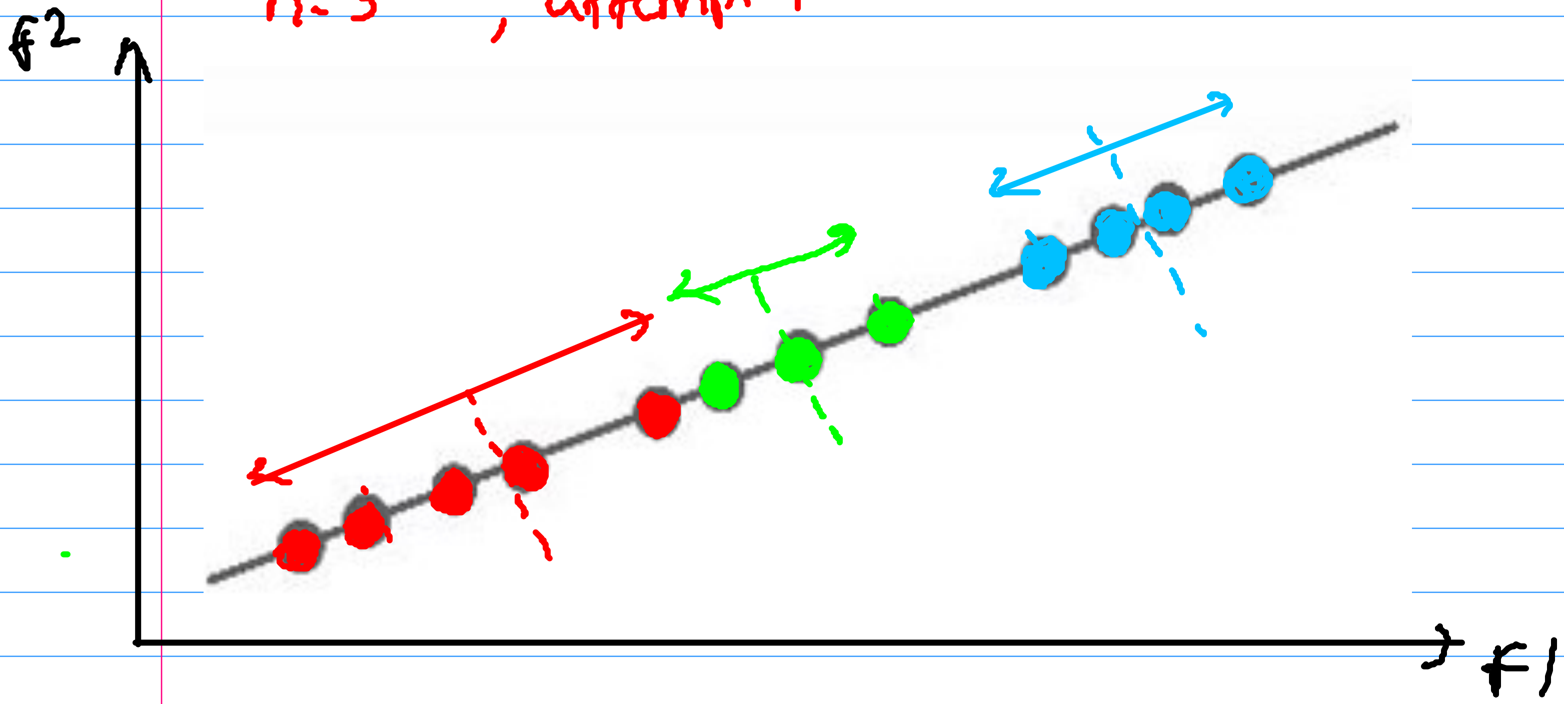
(2) testing

n_clusters = 3

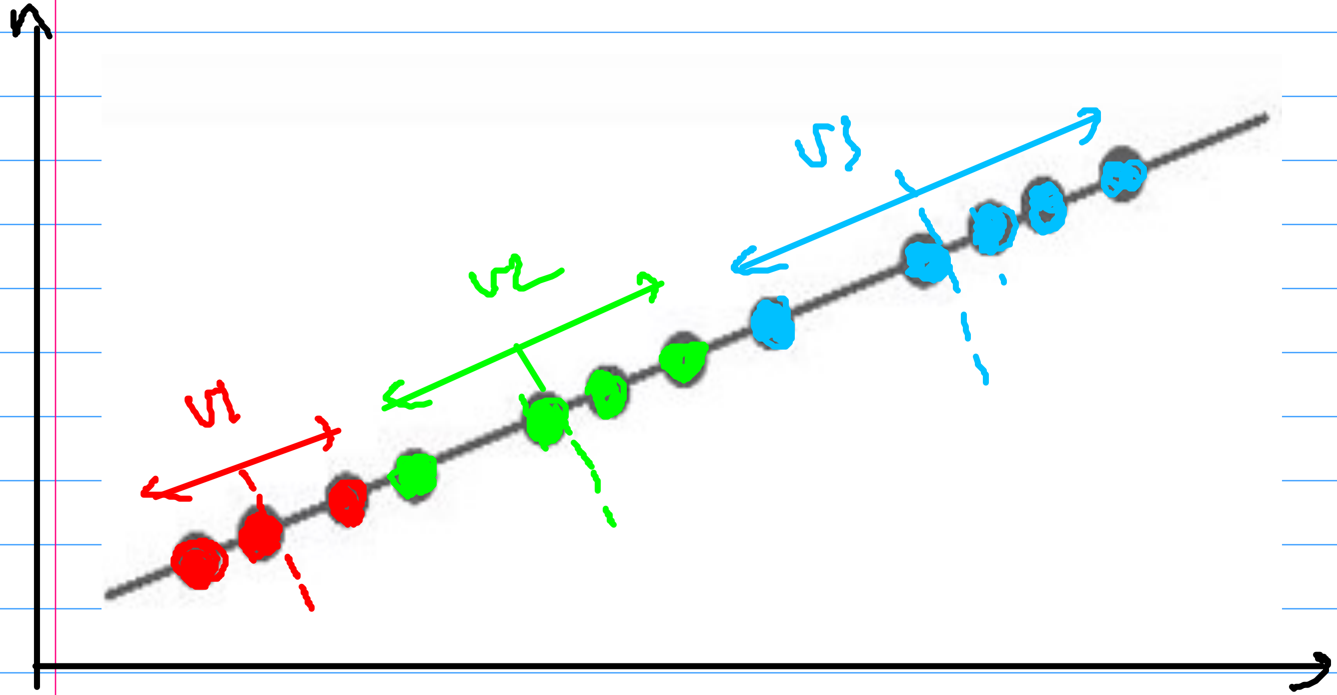


How Kmeans works?

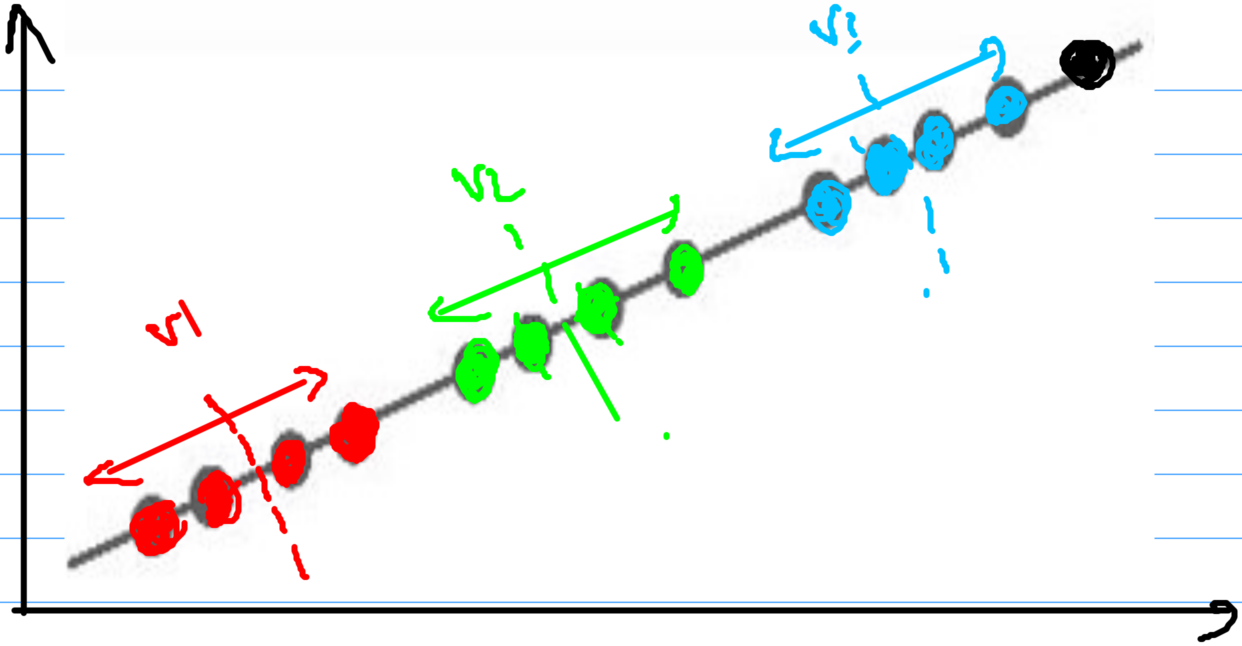
n=3 , attempt 1



attempt 2

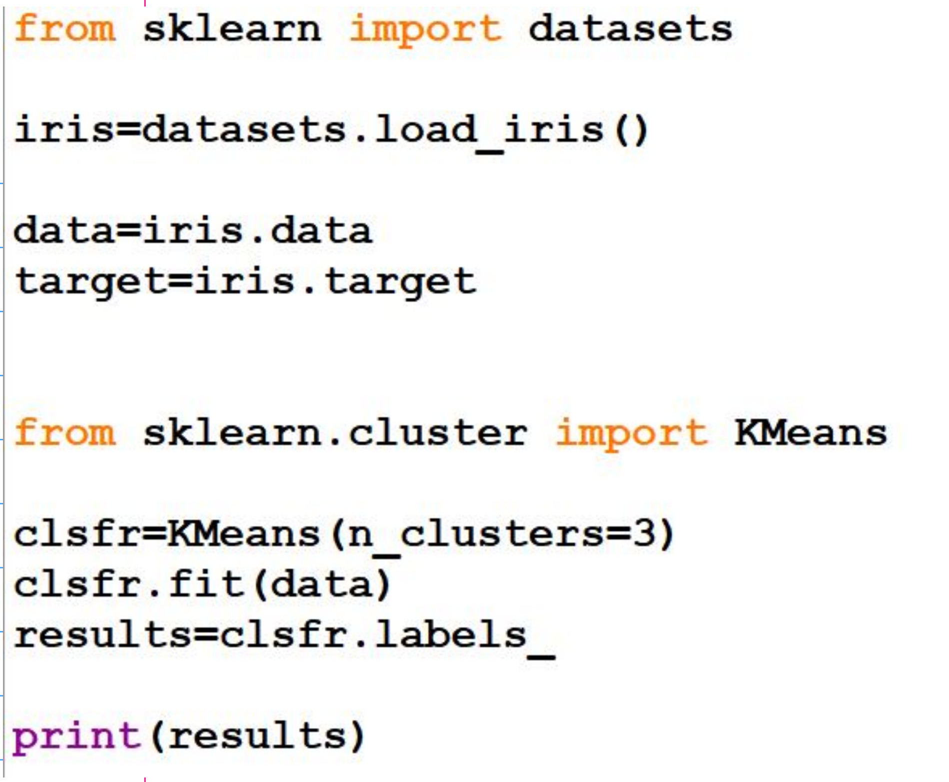


attempt 3

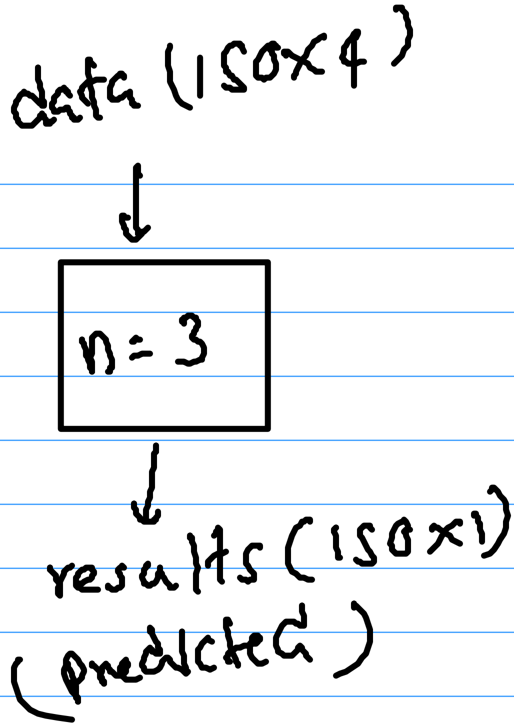


objective – **Total** variance → minimum

$T/V = V_1 + V_2 + V_3 \longrightarrow$ minimum



```
from sklearn import datasets

iris=datasets.load_iris()

data=iris.data
target=iris.target


from sklearn.cluster import KMeans

clsfr=KMeans(n_clusters=3)
clsfr.fit(data)
results=clsfr.labels_

print(results)
```


data (150x4)
n=3
results (150x1)
(predicted)