Dimension Reduction

↓

Reducing the number of features in Raw data

1. Feature selection Technique
2. Feature Extraction.

* **Feature Selection :-**
  * Filter methods
  * Wrapper methods
  * Essemble methods

1. **Filter methods :-**

* **Low Variance** :- Every value is unique in Input variable. How much it is having impact on o/p.

Ex:- I/p

it doesn't have any impact on output. Because. for Every record. it having same value. So, if it is there also, not there also, it will have no change.

So, which ever the column having low variance, we drop that column.

* it is applicable for Continuous data.

for regression :

RMSE ↓ parameter

* No R^2 ✗

as no. of Feature increases R^2 also increase

* **Unique Values** :- Every value is unique values, so, we drop that columns.

Ex:- | name | age | sex | job |

unique rows

* **Low correlation** :- When we have Low correlation with Output variable. we drop that columns.
  * it applicable for Continuous Data.

* **Low I.G** [information Gain] :- When Ever we have got low information Gain, we are going to drop that column.
  * information Gain = 1 - Entropy
  * Entropy = $-\Sigma p \log_2 p$

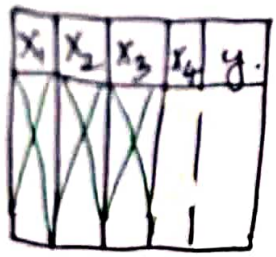
Ex:- | x_1 | x_2 | x_3 | x_4 | y_1 |

* when x_3 having information gain as "0". So, "0" why we required, from that we are not get any information, it is useless, So, we going to drop.
  * it is applicable for Discrete data

2. **Wrapper methods** : Applicable only for Linear Regress
   * Forward Feature Elimination
   * Backward Feature Elimination
   * Recursive Feature Elimination.

• Forward Feature Elimination:

Ex:-

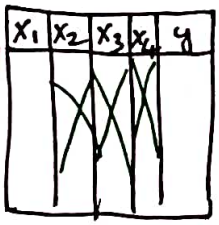


* We have I/p: X_1, X_2, X_3, X_4 and Y variable O/p
* First, we remove X_1: And we see how the relation with $X_2 X_3 X_4$ and "Y"
* Second, we remove x_2: Again, we see relation with $x_3 x_4$ and "y"
* third, we remove x_3: . . . . . .

∴ For Each variable, we remove "forward direction" what, would be the values. How relation Coefficient changes, How slope value change

* Backward Feature Elimination:

Ex:-



* We delete feature from "Backward direction" and we see relation with other variables.

* Recurssive Feature Elimination:

Ex:- in multiple linear regression:

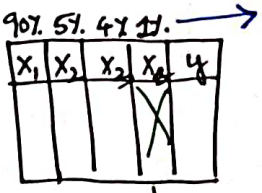
We Taken different "p" Values, which Ever The "p" value is not Significant : which ever the 'p' value Having "p" value is > 0.05 We removed that particular variable : Variable Significance, From That we are going to select remove Them.

* We are going to take subset of Features by using we Eliminate them.
* which ever Feature is having the "p" Value High. it will deleted.
* It is not forward direction (or) Backward direction. we deleted variable from Center Value (or) any were.

3. Essemble methods

* important feature from model.
∴ model.feature_importances_

Ex:- Bagging
Boosting.



Que :- What is difference between "Feature selection" and "Feature Extraction" ?

A:- **Feature Selection** : we dropping of Feature (column) which are not required for analysis, directly we are dropping that column, without converting (or) Transform we just drop data. it is called "Feature Selection".

* **Feature Extraction** :- Here, we Extract the data, from One column and saves it through covariance and correlation matrix and Eigen vectors by using Principal component analysis. and thats we delete the column (or) row after Extracting data.

⇒ Feature Selection :-

1. filter methods
   (Before modelling)
   - low variance
   - Unique
   - low correlation
   - low I.G [Information gain]

2. Wrapper methods
   - → forward Feature Elimination
   - → Backward Feature Elimination
   - → recussive Feature Elimination.

3. Essemble methods
   (after modelling)
   - → important features from model
   - :. model.feature_importances —

* Feature Extraction.

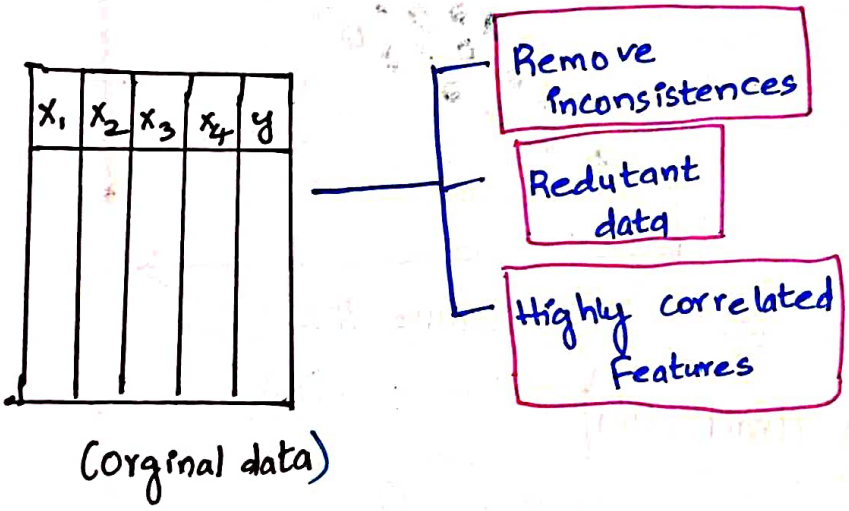
PCA

↓

Principal Component analysis

* Need for PCA :

High dimension data is Extremly Complex to **process** due to **Inconsistencies** in features which **increase** the Computation time and make data processing and EDA more **Convoluted.**



(Orginal data)

New data space with lesser features that requires that retain m of the info.

Ex:-

Feature Selection

(directly remove)

Feature Extraction.

(Extraction of data)

and

Removing Them

"X" person

↓

Knowledge Transfer

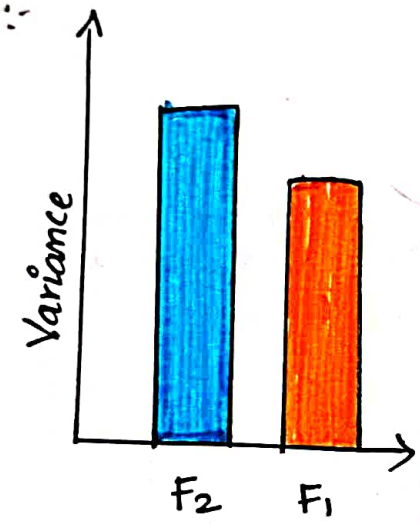
PCA

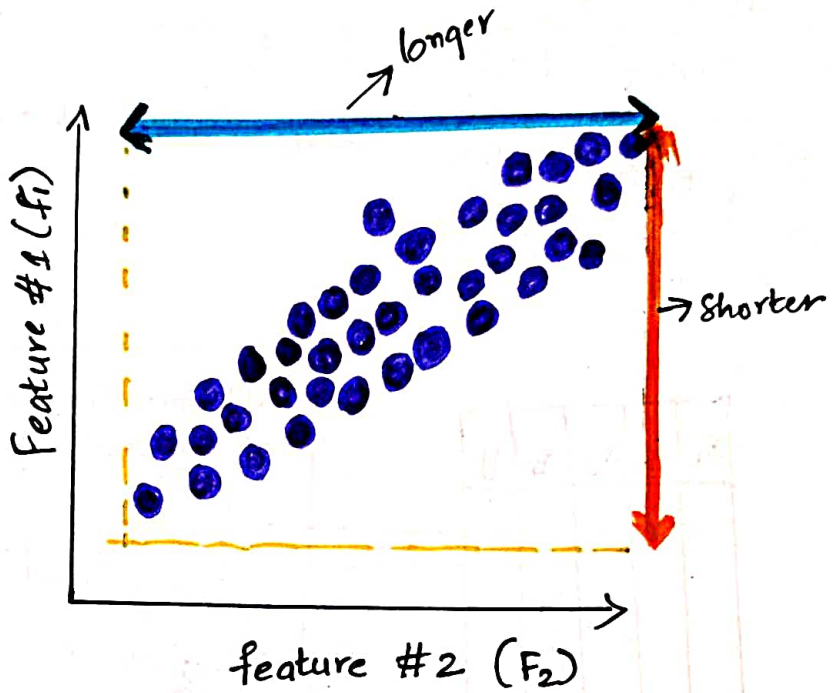
new person

Q:- What is PCA ?

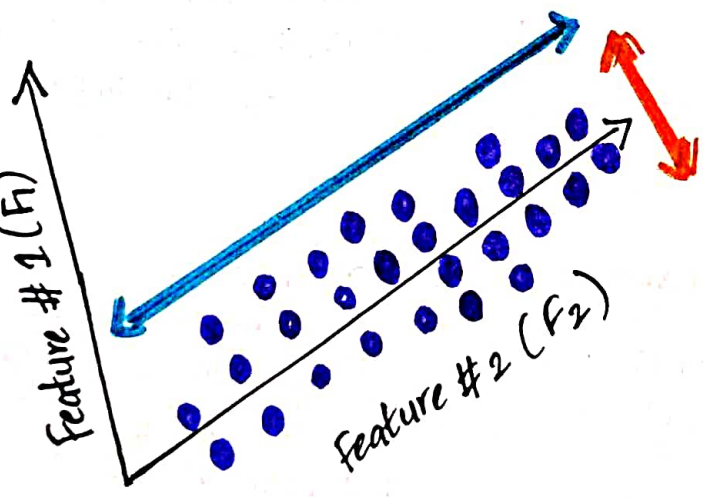
A:- Principal Component analysis (PCA) is a dimensionality reduction Technique that enables You identify correlation and covarience & patterns in a data set so that it can be transformed into a data set of significantly Lower dimension without loss of any important information.

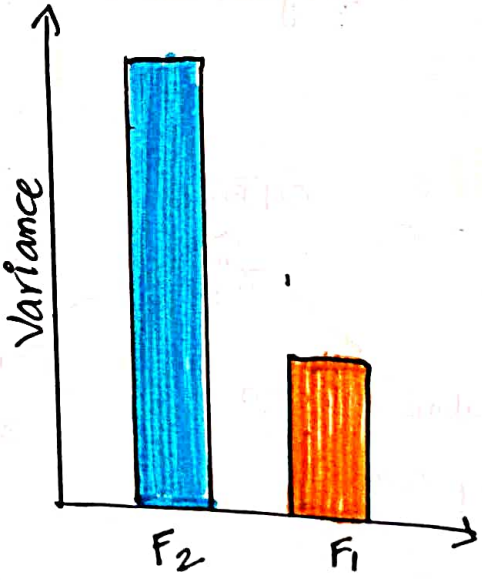
Ex:-



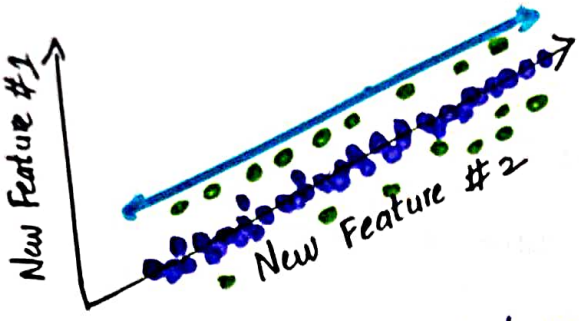


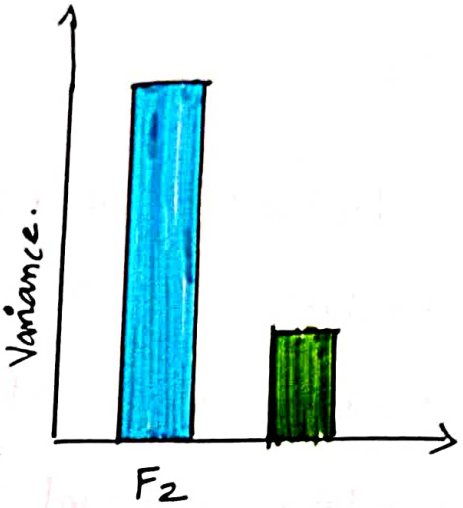
We. Extract data





all the data points projected on F_2 '1' dimension.





all points of data projected on x-axis, we have only 1 dimensional "

Step by Step "PCA"

- Standardization of data
- Computing The covariance Matrix
- Calculating The eigen vectors and Eigen Values.
- Computating the Principal components.
- Reducing the dimensions of data.

* Steps of PCA

1. Standardization of data
2. Computating the covariance matrix
3. Calculating Eigen vectors and values of Eigen
4. Computing poincipal components
5. Reducing the dimensions of data.

STEP:-

(1.) Standardization of data :

Standardization is all about **scaling your data** in such a way that all variables and their values lies with in a **similar range.**

Ex:-

Rating	# of downloads
5	1383
3	668
2	763
5	839

→ Rating Feature ranges between 0-5

→ # of downlods between 100-5000

$$Z = \frac{\text{Variable Value} - \text{mean}}{\text{Standard deviation}}$$

Step2 :- Computing the Covariance matrix

A **Covariance matrix** Express the **correlation** between the different Variables in the dataset. it is Essential to identify **heavily dependent variable** because they contain biased and redudant information which reduces the overall performance of model.

Ex:-

X	Y

:. ⇒ Variance (x) = $\frac{\varepsilon(x-\bar{x})^2}{n-1}$ = $\frac{\varepsilon(x-\bar{x})(x-\bar{x})}{n-1}$ (written as)

* cov(x,x)

⇒ Variance (y) = $\frac{\varepsilon(y-\bar{y})^2}{n-1}$ = $\frac{\varepsilon(y-\bar{y})(y-\bar{y})}{n-1}$

* cov(y,y)

How co-dependent two variables # what is dependency between two variable

* Covariance (x,y)

* cov(x,y) = $\frac{\varepsilon(x-\bar{x})(y-\bar{y})}{n-1}$

Matrix of Covariance

$$\begin{bmatrix} cov(a,a) & cov(a,b) \\ cov(b,a) & cov(b,b) \end{bmatrix}$$

* From scatter plot.

what covariance gives :- (+ve) = relation between x & y is directly proportional → when "x" increases "y" also increases.

(-ve) negative covariance :- relation between x & y is indirectly proportional (when "x" increase, "y" decreases)

6/10/22 5.00 pm

Step 3: Calculating the Eigenvectors and Eigen values.

Eigen vectors and Eigen values are mathematical constructs that must be computed from the covariance matrix in order to determine the principal components of data set.

Principal components are the new set of variables that are obtained from initial set of variables. they compress and posses most of the useful information that are scatted among initial variables.

- **Eigen vectors:** are those vectors when a linear transformation is performed on them then their direction does not change.
- **Eigen values:** simply denote the scalars of the respective eigen vectors.

Step: 4 Computing the principal Components

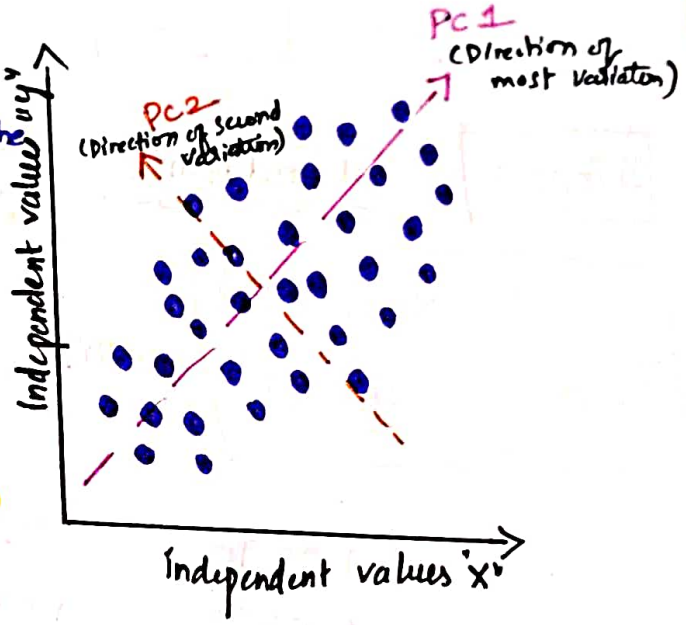
Once we have computed the Eigen vectors and Eigen Values, all we have to do is order them in the descending order, where the eigen vector with highest eigen value is the most significant and thus forms the first principal Component.

PC 1 (Principal Component)

it is most significant and stores the Maximum possible information.

PC 2 (Principal Component)

it is most significant PC and stores remaining maximum info and so on.



Step 5:- Reducing the dimension of data Set

The last step in performing PCA is to re-arrange the Original data, with final principal components which represents the maximum and most significant information of dataset.

Example :

PCA

	X_1	X_2	X_3
Student	Maths	English	Arts
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

Step: 1 Standardize the datapoints.

$$X_{new} = \frac{X - mean(x)}{std(x)} \quad \text{(or)} \quad \left[\frac{X - \bar{X}}{\sigma}\right]_{z\,score}$$

$$\begin{bmatrix} 1.0690 & 0. & 1.11803 \\ 1.0690 & 1.58113 & -1.11803 \\ -0.26726 & 0. & 0. \\ -0.26726 & 0. & 1.11803 \\ -1.6035 & -1.58113 & -1.11803 \end{bmatrix}$$

Step2 : Compute the covariance matrix of whole dataset

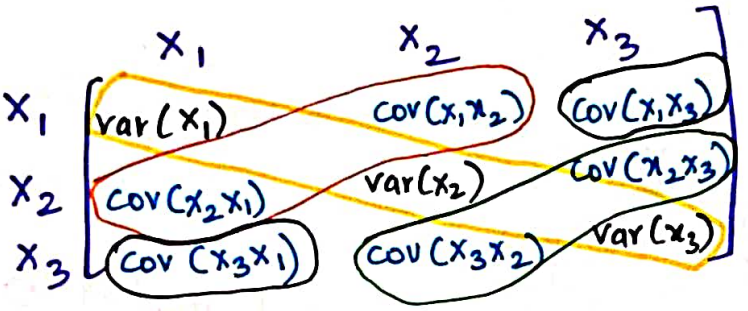
$$cov(x,y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

	X_1	X_2	X_3
X_1	$var(X_1)$ 0.99954	$cov(X_1X_2)$ 0.84489	$cov(X_1X_3)$ 0.35848
X_2	$cov(X_2X_1)$ 0.84489	$var(X_2)$ 0.99982	$cov(X_2X_3)$ 0.0
X_3	$cov(X_3X_1)$ 0.35848	$cov(X_3X_2)$ 0.0	$var(X_3)$ 0.69996

$\therefore$ $cov(X_1, X_1) = Variance(X_1)$

$cov(X_2 X_2) = Var(X_2)$

$cov(X_3 X_3) = var(X_3)$



Upper triangle and lower triangle matrix will be same.

The Center diagonal Element will be Variance of Each parameter.

Step 3 :- Compute Eigen vectors and Eigen values.

Let "A" be a square matrix,

"V" a vector,

"λ" as Scaler

that satisfies $Av = \lambda v$,

Then "λ" is called Eigen Values associated with eigen vector V of A.

Eigen value $\lambda_1 = 1.899$

$$V_1 = \begin{bmatrix} 3.34704 \\ 3.14213 \\ 1 \end{bmatrix}$$

* Eigen value (λ_2) = 0.0501

$$V_2 = \begin{bmatrix} -1.81273 \\ 1.61269 \\ 1 \end{bmatrix}$$

* Eigen value (λ_3) = 0.7493

$$V_3 = \begin{bmatrix} 0.13786 \\ -0.4651 \\ 1 \end{bmatrix}$$

STep 4 : Sorting the Eigen vectors by decreasing Eigen values and choose "k" eigen vectors with largest eigen values to form "d x k" dimensional matrix W.

$$\begin{bmatrix} 1.8998 \\ 0.7493 \\ 0.0501 \end{bmatrix}$$

$$\frac{1.8998}{1.8998+0.7493+0.0501}$$

$\frac{\lambda_1}{\text{Sum of Eigen Values.}}$

* normalization of Eigen values is Called "Principal Components".

Contribution

$$\lambda_1 = \frac{1.8998}{2.6992} = 0.70 = 70\%$$

$$\lambda_2 = \frac{0.7493}{2.6992} = 0.28 = 28\%$$

Consider these two Feature.

$$\lambda_3 = \frac{0.0501}{2.6992} = 0.02 = 2\%$$

delete/ cut this Feature

Feature Exaction :

Now, information has been stored,

How ? we deleted rows ? and columns.

First, we took covariance matrix, calculated Eigen Values.

↓

it already having information of "x" and "x". it stored.

CODE :-

PCA

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# load data.

# df = pd.read_csv("wine.csv")

# df.head()
```

out :

alcohol	Malic_acid	Ash	Ash_alcanity	Magnesium	Total_Phenols	Flavanoids	non flavanoid_phenols	Proanthocyanins	color_intensity	Hue	OD280	Proline	custSegm
14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065	1
13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050	1
13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185	1
14.37	1.95	2.50	16.8	113	3.85	3.40	0.24	2.18	7.80	0.86	3.45	1480	1
13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735	1

```
# df.info()
```

Out: Range index 178 entries 0 to 177
Data columns (total 14 columns)

```
#Out variable, counts. (3 class problem)
# df["Customer_Segment"].Value_counts()
```

Out:
2	71
1	59
3	48

```
# X & y
# x = df.iloc[:,:13]
# y = df.iloc[:,13]

# Train & Test
    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_state=29)
```

Without PCA

```
#model
    from sklearn.linear_model import LogisticRegression.
# model = LogisticRegression()
# model.fit(x_train, y_train)
```

Out: LogisticRegression()

predict

y_pred = model.predict(x_test)

accuracy

from sklearn.metrics import accuracy_score.

accuracy_score(y_test, y_pred)

Out : 0.9166

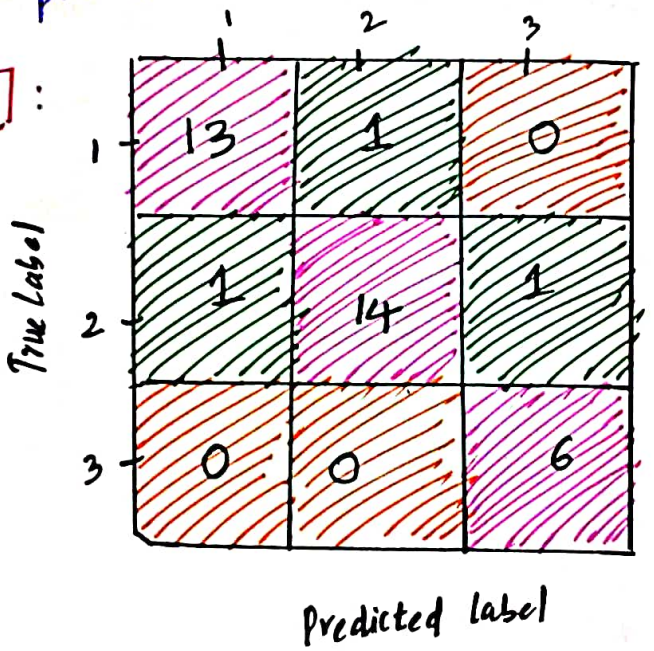
Confusion matrix.

from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(model, x_test, y_test)

plt.show()

Out :



Applying 'PCA'

```
from sklearn.preprocessing import StandardScaler

# sc = StandardScaler()
# X = sc.fit_transform(x)

#train, test
# x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.
                                                      random_state=29)

# model.
from sklearn.decomposition import PCA
# pca_model = PCA(n_components = 0.95)   Ex: 0.99
                                          ↳ which contributes 0.95, we consider that only. remaining 5%. we leave.
# x_train_pca = pca_model.fit_transform(x_train)
# x_test_pca = pca_model.fit_transform(x_test)

# shape
# x_train_pca.shape
```

old: 14 → 13 i/p, 1 o/p

out: (142, 10) (new) (12)

```
= pca_model.explained_variance_ratio_
```

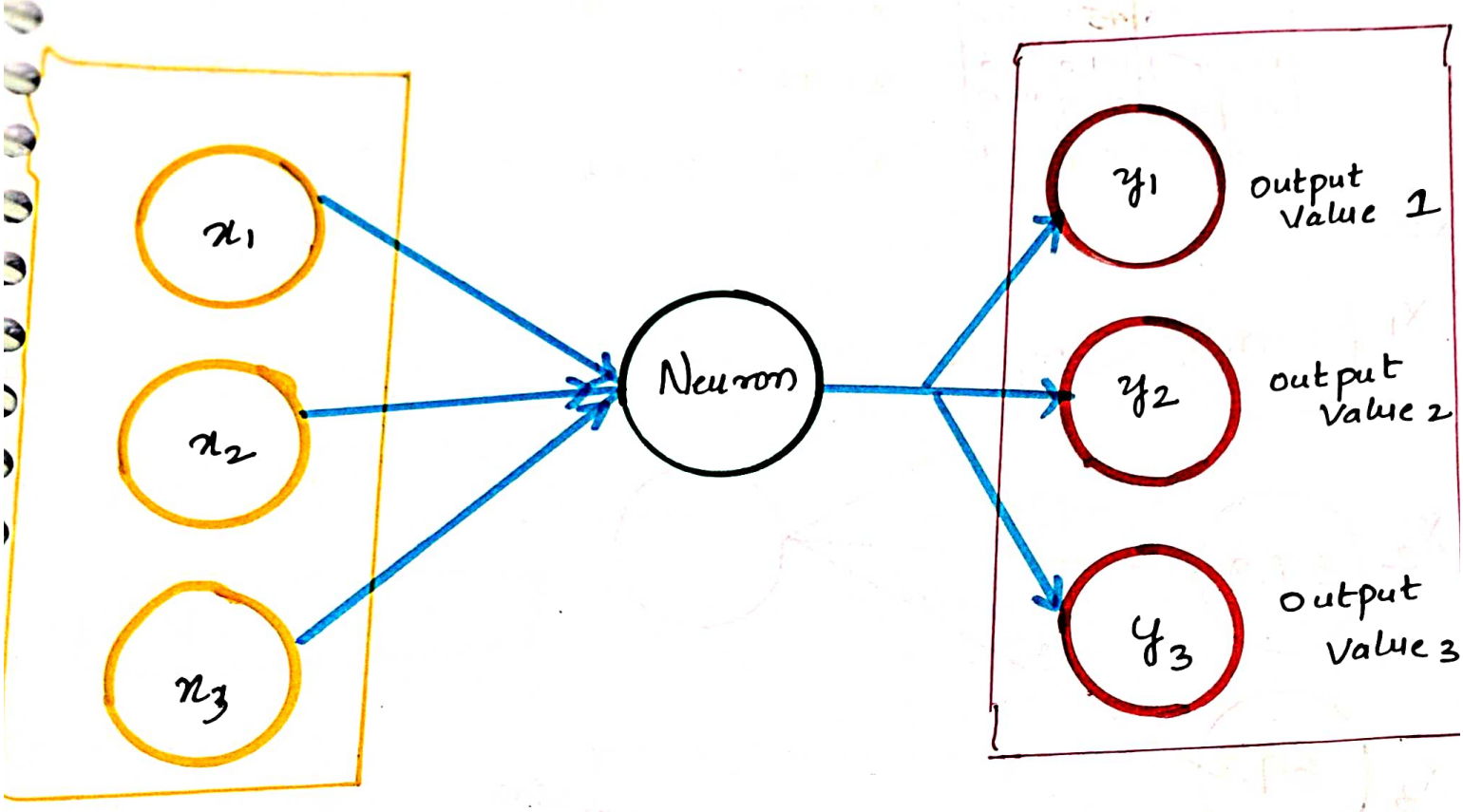
out: array([0.3672 , 0.1923 , 0.1083 , 0.0741 , 0.0628
(1 Eigen value, 2 Eigen value)

0.0505 , 0.0419 , 0.02518 , 0.0222 , 0.0185])

11, 12)

* Binary classification → o/p neuron ⇒ 1 neuron
* Regression → output neuron ⇒ 1 neuron
* Multiclassification → output neuron ⇒ no. of categories (or) classes.

Ex:- Penguin
Age, chi, Gentoo



* **Weights**

Machine learning :- $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 = (y - \hat{y})_{min}$

deep learning : $b + w_1 x_1 + w_2 x_2 + w_3 x_3 = (y - \hat{y})_{min}$

→ It can be written as

$$\sum w_i x_i + b$$

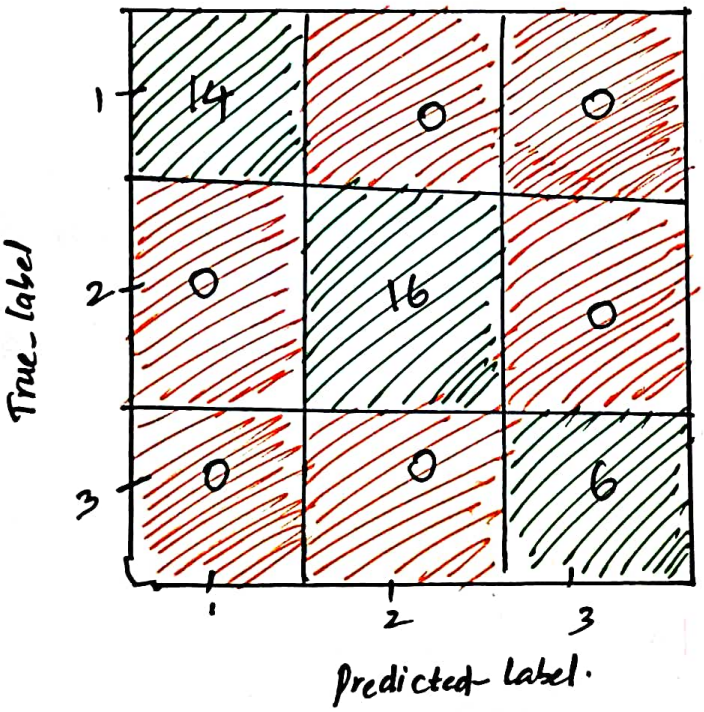
fit and again modelling by "10" features only.

```
from sklearn.linear_model import LogisticRegression
# classifier = LogisticRegression()
# classifier.fit(x_train_pca, y_train)
```

Out: LogisticRegression()

```
# predict
# y_pred = classifier.predict(x_test_pca)
# Confusion matrix.
from sklearn.metrics import plot_confusion_matrix.
# plot_confusion_matrix(classifier, x_test_pca, y_test)
```

Out:



7/05/22
3:30 AM