NIT TRICHY

FEBRAUARY 2022

MACHINE LEARNING ALGORITHMS

PART 1

Computers are able to hear, see and learn. Welcome to the Future.

PRESENTED TO

Sishaj P Simon

PRESENTED BY

Surendra Srinivas

Acknowledgement

FIRSTLY, I WOULD LIKE TO EXPRESS MY SINCERE GRATITUDE TO THE DISTINGUISHED ADVISOR PROF.SISHAJ P SIMON, FACULTY IN ELECTRICAL AND ELECTRONICS DEPARTMENT AT NIT TRICHY, FOR THE CONTINUOUS SUPPORT OF MY CONCISE REPORT ON ALGORITHMS, FOR HIS PATIENCE, MOTIVATION, AND IMMENSE KNOWLEDGE. HIS GUIDANCE HELPED ME IN ALL THE TIMES OF DIFFICULTIES AND THE OBSTACLES I FACED. I COULD NOT HAVE IMAGINED HAVING A BETTER ADVISOR AND MENTOR FOR MY REPORT MAKING.

BESIDES MY ADVISOR, I WOULD LIKE TO THANK MY SENIORS FOR THEIR INSIGHTFUL COMMENTS AND ENCOURAGEMENT. I THANK MY FELLOW FRIENDS IN FOR THE PROGRAMMING DISCUSSIONS, FOR THE SLEEPLESS NIGHTS WE WERE WORKING TOGETHER BEFORE DEADLINES, AND FOR ALL THE FUN WE HAVE HAD IN THE JOURNEY.

LAST BUT NOT THE LEAST, I WOULD LIKE TO THANK MY FAMILY: MY PARENTS AND TO MY BROTHERS AND SISTER FOR SUPPORTING ME SPIRITUALLY THROUGHOUT WRITING THIS REPORT AND MY LIFE IN GENERAL.

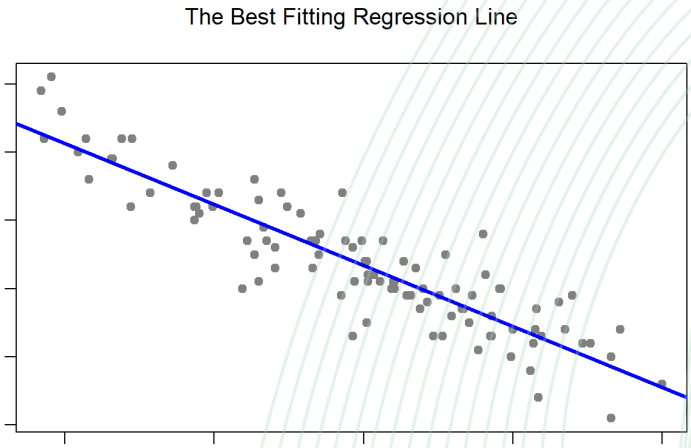
MATTERS OF THE DOCKET

" Machine intelligence is the last invention that humanity will ever need to make. "

LINEAR REGRESSION

Sir Francis Galton

1894



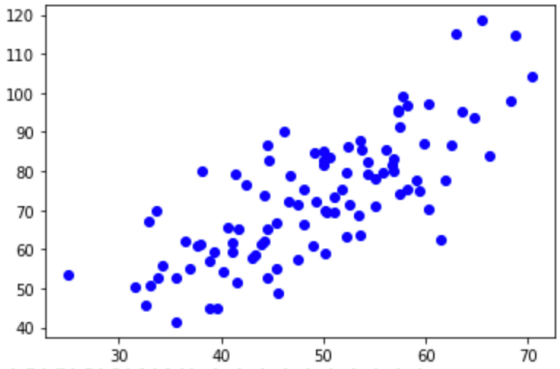
Linear regression, is a way of calculating the relationship between two variables. It assumes that there's a direct correlation between the two variables, and that this relationship can be represented with a straight line.

For the Straight line of the form Y = A*X+B,the coefficients A and B are calculated to minimize the error between the models predictions and the actual data in the training set.

LINEAR REGRESSION IMPLEMENTATION IN PYTHON

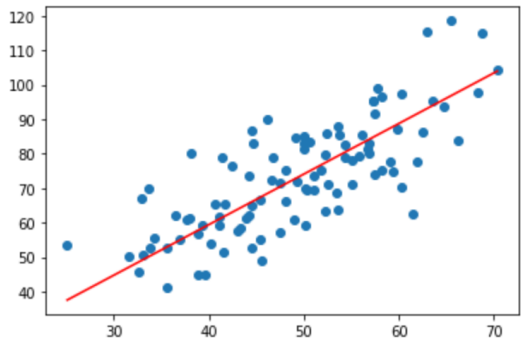
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('LR.csv')
print(data)
X = data.iloc[:,0]
Y = data.iloc[:,1]
plt.scatter(X,Y,color='b')
plt.show()

a = 0
b = 0
alpha = 0.0001
n = float(len(X))

for i in range(10000):
    Y_pred = a*X+b
    a_Derivative = (-2/n)*sum(X*(Y-Y_pred))
    b_Derivative = (-2/n)*sum(Y-Y_pred)
    a = a - alpha*a_Derivative
    b = b - alpha*b_Derivative

print("Y = ",a," *X+",b)
Y_pred = a*X+b
plt.scatter(X,Y)
plt.plot([min(X),max(X)],[min(Y_pred),max(Y_pred)],color = 'red')
plt.show()
```

Before Regression



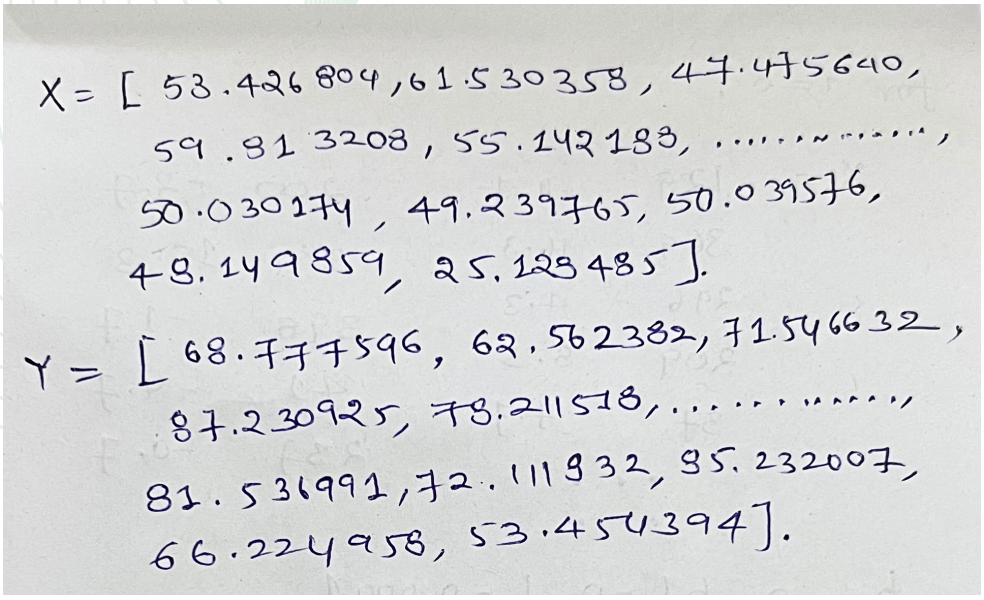
After Regression

```
Y =  1.4673253395092016  *X+ 0.7300121018992646
```



Dataset and Numerical computation for Linear Regression

	32.502345269453031	31.70700584656992
0	53.426804	68.777596
1	61.530358	62.562382
2	47.475640	71.546632
3	59.813208	87.230925
4	55.142188	78.211518
...	...	...
94	50.030174	81.536991
95	49.239765	72.111832
96	50.039576	85.232007
97	48.149859	66.224958
98	25.128485	53.454394



$a = 0,\ b = 0,\ \alpha = 0.0001,\ n = 99$

⇒ for $i = 0$:

$Y\_pred = 0 * X + 0 = 0$

$$a\_Derivative = \frac{-2}{99} \times \Sigma (X)(Y - Y\_pred)^{\nearrow 0}$$

$$= \frac{-2}{99} \Sigma (XY) \leftarrow (\text{array multiplica}$$

$$= -7424.3352.$$

$$b\_Derivative = \frac{-2}{99} \times \Sigma (Y - Y\_pred)^{\nearrow 0}$$

$$= \frac{-2}{99} \Sigma Y$$

$$= -146.2989$$

✓ $a = a - \alpha * a\_Derivative$

$= 0 - 0.0001 \times (-7424.335)$

$= 0.7425$

✓ $b = b - \alpha * b\_Derivative$

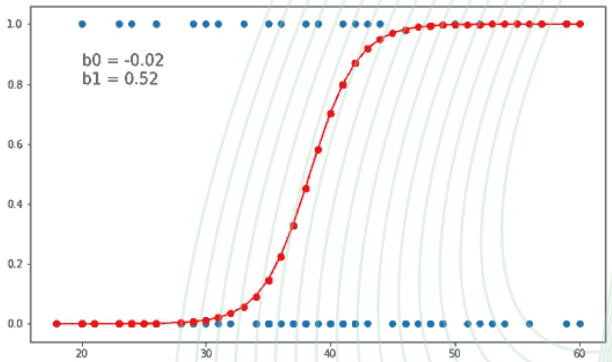
$= 0 - 0.0001 \times (-146.3)$

$= 0.01463.$

LOGISTIC REGRESSION

JOSEPH BERKSON

1944



Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist.

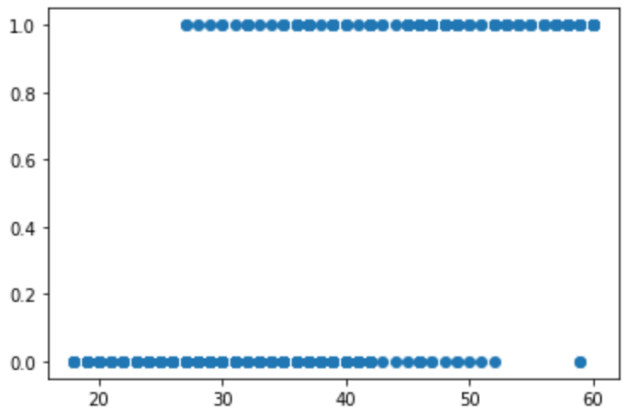
In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

LOGISTIC REGRESSION IMPLEMENTATION IN PYTHON

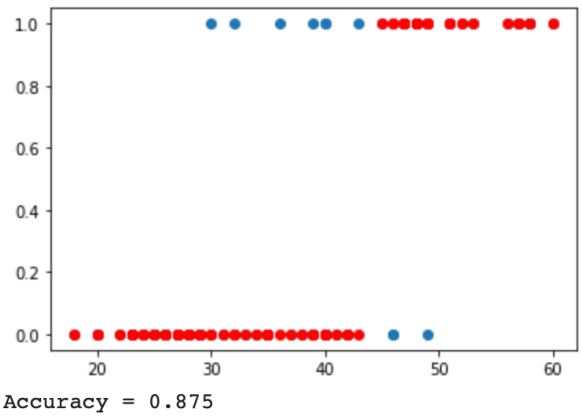
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
from sklearn.model_selection import train_test_split

x=pd.read_csv('Social_Network_Ads.csv')
plt.scatter(x['Age'],x['Purchased'])
#plt.legend()
plt.show()
X_train, X_test, y_train, y_test = train_test_split(x['Age'], x['Purchased'],test_size=0.20)

def normalize(X):
    return X-X.mean()

def predict(X,b0,b1):
    return np.array([1/(1+math.exp(-1*b0+-1*b1*x)) for x in X])

def logistic_Regression(X,Y):
    X = normalize(X)
    b0 = 0
    b1 = 0
    L = 0.001
    epochs = 1500
```

```
    for e in range(epochs):
        y_pred = predict(X,b0,b1)
        D_b0 = -2*sum((Y-y_pred)*y_pred*(1-y_pred))
        D_b1 = -2*sum(X*(Y-y_pred)*(1-y_pred))
        b0 = b0-L*D_b0
        b1 = b1-L*D_b1
    return b0,b1

b0,b1 = logistic_Regression(X_train,y_train)
X_test_norm = normalize(X_test)
y_pred = predict(X_test_norm,b0,b1)
y_pred = [1 if p>=0.5 else 0 for p in y_pred]
plt.clf()
plt.scatter(X_test,y_test)
plt.scatter(X_test,y_pred,c="red")
plt.show()
accuracy = 0

for i in range(len(y_pred)):
    if y_pred[i] == y_test.iloc[i]:
    accuracy += 1
print(f"Accuracy = {accuracy / len(y_pred)}")
```

Before Regression



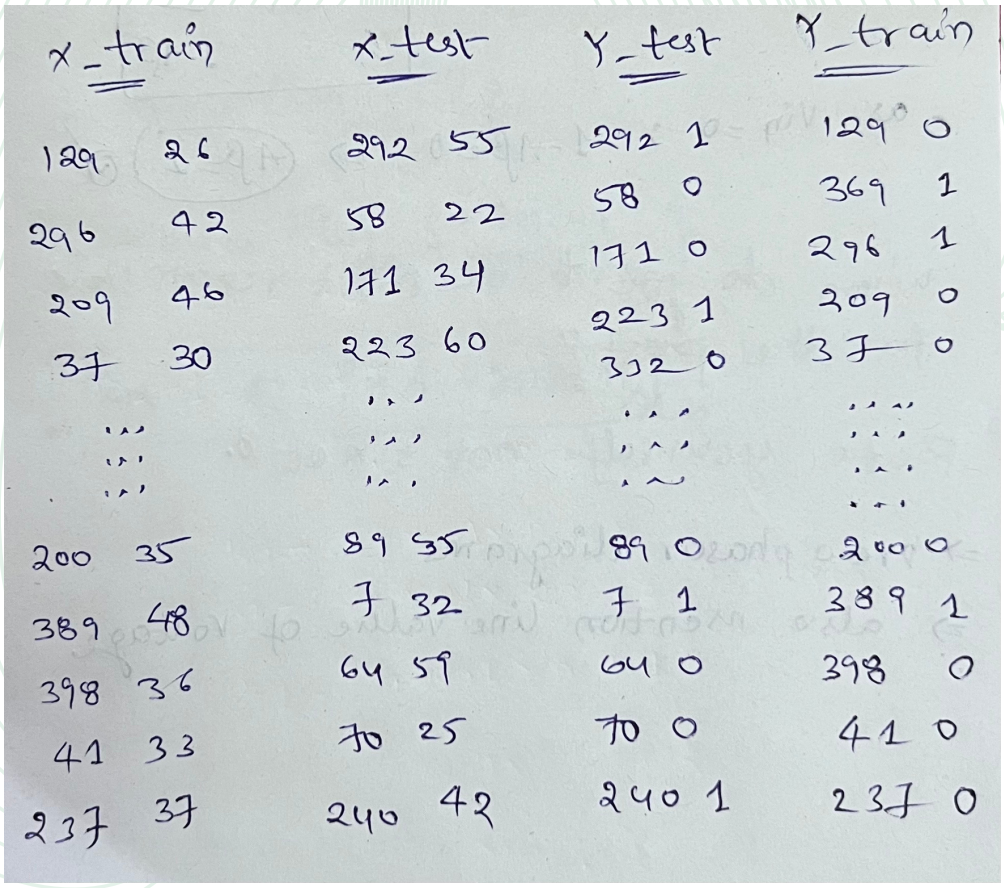
After Regression




```
Accuracy = 0.875
```

Dataset and Numerical computation for Logistic Regression

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
..	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1



for $i = 0$:

$X =$

129	-11.7	200	-3.7
369	16.3	389	10.3
296	4.3	398	-1.7
209	8.3	41	-4.7
37	-7.7	237	-0.7

...

...

$b_0 = 0, b1 = 0, L = 0.0001$

$D\_b0 = -2 * \Sigma [(Y - Y\_pred) * Y\_pred * (1 - Y\_pred)]$

$Y\_pred = \left\{ \frac{1}{1 + e^{-b0 + -b1 * x}} \right\}$ → array.

$= 23.0$

$D\_b1 = -2 * \Sigma [x * (Y - Y\_pred)(1 - Y\_pred)]$

$= 176.31742$

$b0 = b0 - 0.0001 \times D\_b0$

$= 0 - 0.0001 \times 23 = -23 \times 10^{-4}$.

$b1 = b1 - 0.0001 \times D\_b1$

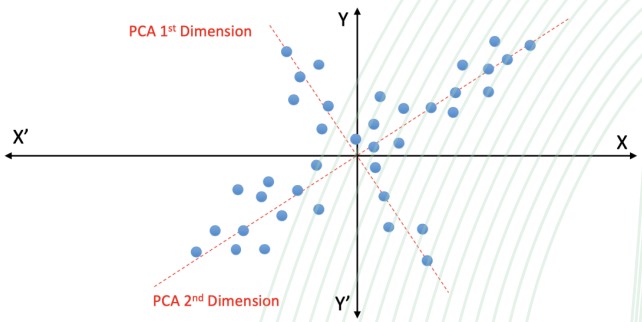
$= 0 - 0.0001 \times 176.32$

$= -176.32 \times 10^{-4}$.

PRINCIPAL COMPONENT ANALYSIS

KARL PEARSON

1901



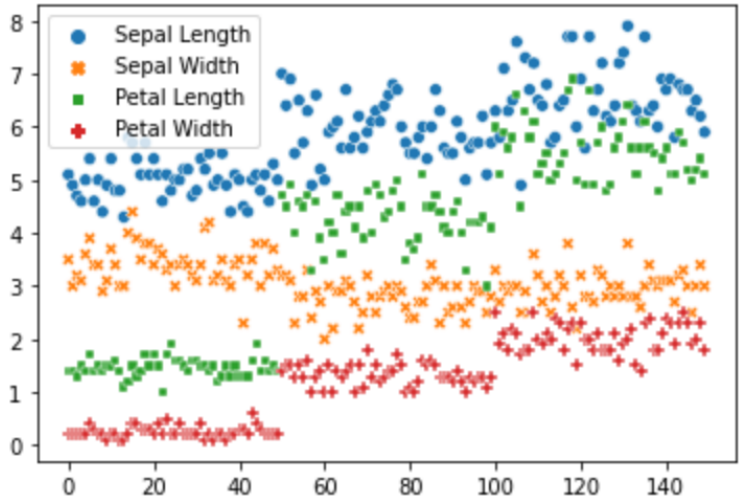
Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

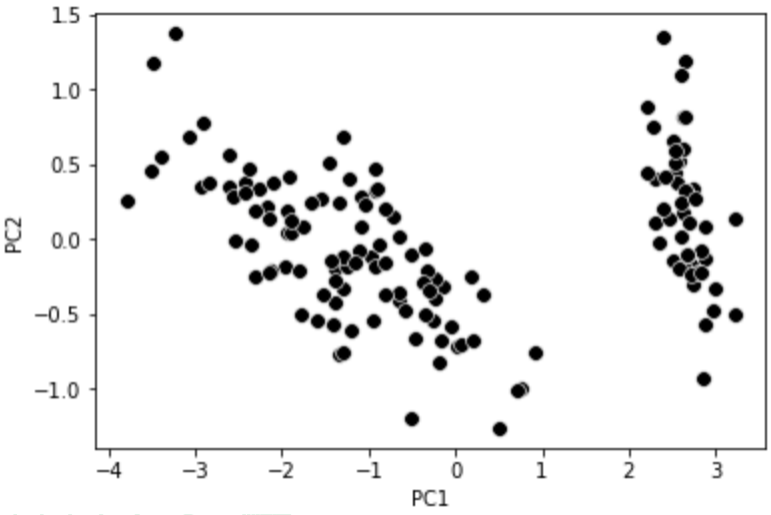
PRINCIPAL COMPONENT ANALYSIS IMPLEMENTATION IN PYTHON

```
import sklearn.datasets as DS
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sb


def PCA(X,num_components):
    X_meaned = X-np.mean(X,axis=0)
    cov_mat = np.cov(X_meaned, rowvar = False)
    eigenvalues, eigenvectors = np.linalg.eigh(cov_mat)
    sorted_index = np.argsort(eigenvalues)[::-1]
    sorted_eigenvalues = eigenvalues[sorted_index]
    sorted_eigenvectors = eigenvectors[:,sorted_index]
    eigenvector_subset = sorted_eigenvectors[:,0:num_components]
    X_reduced = np.dot(eigenvector_subset.transpose(), X_meaned.transpose()).transpose()
    return X_reduced

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
data = pd.read_csv(url, names=['Sepal Length','Sepal Width', 'Petal Length','Petal Width', 'Target'])

X = data.iloc[:,0:4]
target = data.iloc[:,4]

reducedmat = PCA(X,2)
df = pd.DataFrame(reducedmat,columns = ['PC1','PC2'])
df = pd.concat([df,pd.DataFrame(target)],axis=1)
sb.scatterplot(data = df,x ='PC1',y='PC2',s=60,palette = 'icefire')
#sb.scatterplot(data = data) # check plotting this also you will see only essence of the data is plotted.
```

Before Dimension Reduction

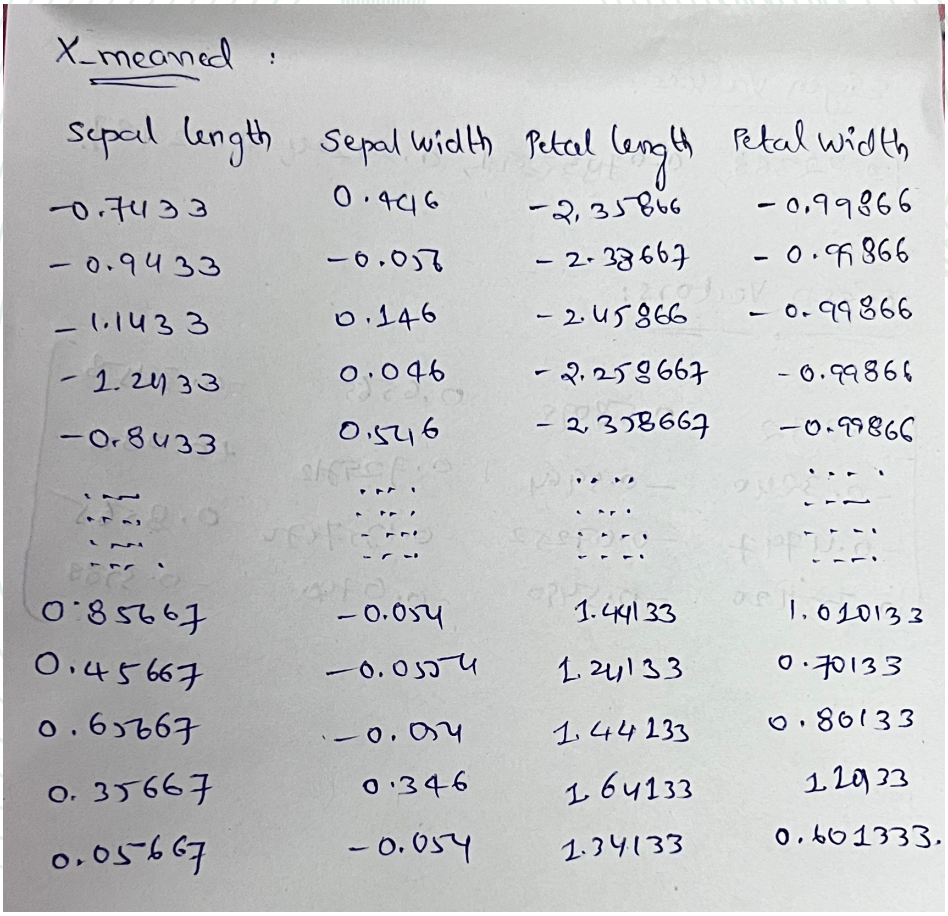


After Dimension Reduction



Dataset and Numerical computation for Principal Component Analysis

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica



cov-mat :

$$\begin{bmatrix} 0.68569 & -0.03926 & 1.27367 & 0.5769039 \\ -0.039268 & 0.1800403 & -0.32149 & -0.1179312 \\ 1.27368\text{2} & -0.3217 & 3.113174\text{42} & 1.296387\text{1} \\ 0.5169 & -0.117 & 1.29638747 & 0.582414\text{7} \end{bmatrix}$$

Eigen Values :

$$[0.02368,\ 0.07852391,\ 0.24224,\ 4.22489].$$

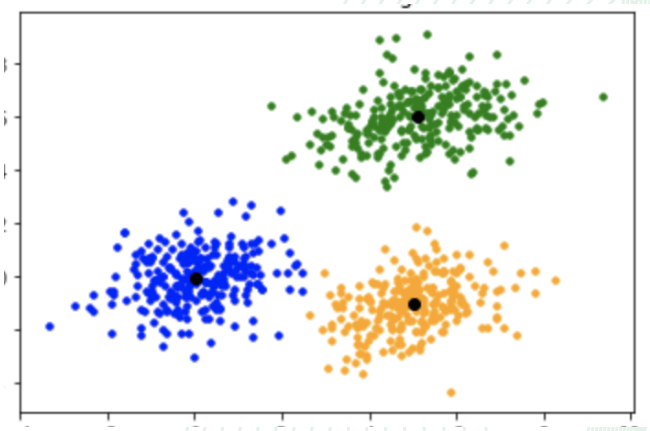
Eigen Vectors :

$$\begin{bmatrix} 0.3172 & 0.58099 & 0.6565 & -0.36158 \\ -0.3240 & -0.5964 & 0.72972 & +0.082 \\ -0.4797 & -0.07252 & -0.1377674 & -0.8565 \\ 0.77120 & -0.5490 & -0.0740 & -0.3588 \end{bmatrix}$$

KMEANS CLUSTERING

JAMES MACQUEEN

1967



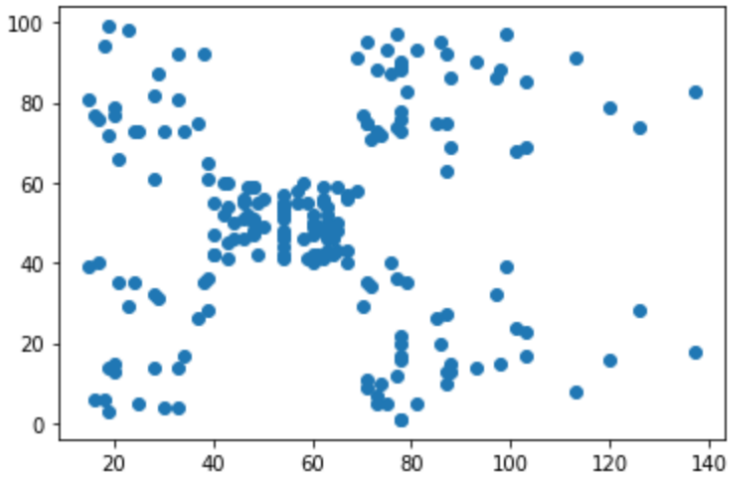
Kmeans algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group.
It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible.

It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

KMEANS CLUSTERING IMPLEMENTATION IN PYTHON

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans


data = pd.read_csv('Mall_Customers.csv')
X = data.iloc[:,3:].values
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i, init='k-means++',max_iter=300, n_init = 10,random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.scatter(X[:,0],X[:,1])
plt.show()
plt.plot (range(1,11),wcss)
plt.title("The Elbow Method")
plt.xlabel("Number of clusters")
plt.ylabel("WCSS")
plt.show()

kmeans = KMeans(n_clusters=5, init = 'k-means++',max_iter = 300,n_init = 10, random_state =0)
y_kmeans = kmeans.fit_predict(X)
plt.scatter(X[y_kmeans==0,0],X[y_kmeans==0,1],s=50,c='red',label = "Cluster 1")
plt.scatter(X[y_kmeans==1,0],X[y_kmeans==1,1],s=50,c='blue',label = "Cluster 2")
plt.scatter(X[y_kmeans==2,0],X[y_kmeans==2,1],s=50,c='green',label = "Cluster 3")
plt.scatter(X[y_kmeans==3,0],X[y_kmeans==3,1],s=50,c='black',label = "Cluster 4")
plt.scatter(X[y_kmeans==4,0],X[y_kmeans==4,1],s=50,c='orange',label = "Cluster 5")

plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=150,c='yellow',label = 'Centroids')
plt.title('Clusters of clients')
plt.xlabel("Annual Income(k$)")
plt.ylabel('Spending score (1-100)')
plt.legend()
plt.show()
```

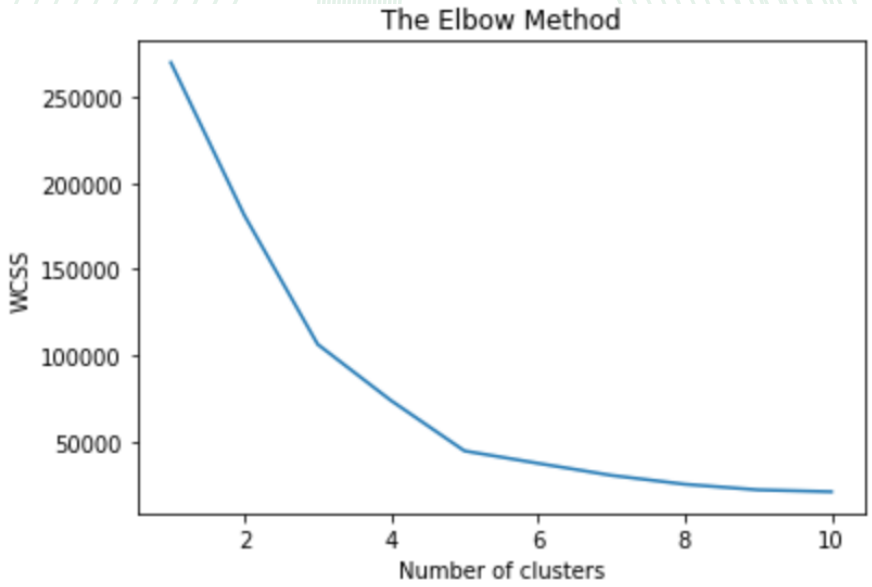
Before Clustering



After Clustering

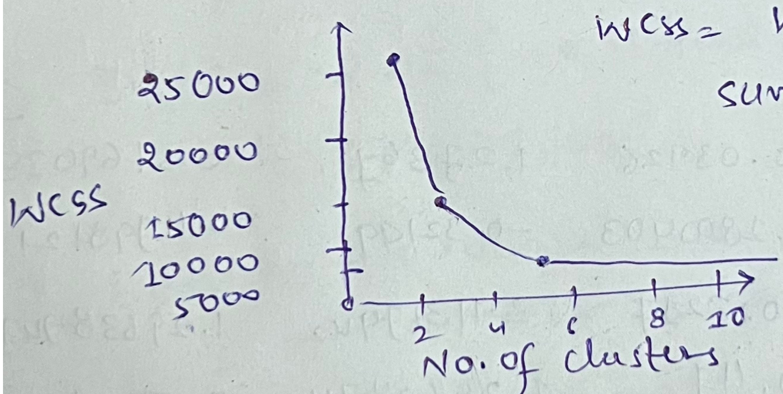
Dataset and Numerical computation for KMeans Clustering

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
..	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83



→ No. of clusters can be observed from the graph:

WCSS = Within cluster sum of squares.



WCSS = [269981.28, 181363.5959, 106348.373, 73679.789, 44448.455, 37265.865, 30259.657, 25095.703, 21830.0419, 20736.6799].

Y_Kmeans = [4 3 4 3 4 3 4 3 4 3
4 3 4 3 4 3 43 4 3 4 3 4 3 4 3
4 3 4 3 4 3 4 3 43 43 43 4 3
4 3 4 1 4 3 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 0 2
2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0
2 0 2 0 2 0 2 0 2 0 2020 2 0 2 0
2 0 2 0 2 0 2 0 2 0].

MEET THE AUTHOR

"We are entering a new world. The technologies of machine learning, speech recognition, and natural language understanding are reaching a nexus of capability. The end result is that we'll soon have artificially intelligent assistants to help us in every aspect of our lives."

The Author, SURENDRA SRINIVAS, Though he loves solving circuits, is a Machine Learning enthusiast, an undergraduate in Electrical and Electronics engineering from National Institute of Tiruchirappalli. He was born in the rural areas of Telangana. His Interests includes Programming, Reading, Travelling, Music, Exploring and Learning . The Author feeds his addiction to badminton by playing in the courts everyday.

" Although still in its infancy, Machine learning will be a game changer.", Author Believes...

Reach him at :

surendranaik242@gmail.com
107120026@nitt.edu
9505245988
https://github.com/Surendra-Srinivas