Principal Component Analysis

Why PCA?

- Visualisation
- Dimensionality Reduction
- Multicollinearity
- Noise reduction in the dataset

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('Iris.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	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

PCA
↓
Reduce Dim.
+
Preserve Informat^n

In [4]:
```
label = df['Species']
data = df.drop("Species",axis=1)
data = data.drop("Id",axis=1)
```

In [5]:
```
print(data.shape)
print(label.shape)
```

```
(150, 4)
(150,)
```

In [6]:
```
data.head()
```

Out[6]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [7]:
```
# Data-preprocessing: Standardizing the data
```

numpy array

```
from sklearn.preprocessing import StandardScaler
standardized_data = StandardScaler().fit_transform(data)
print(standardized_data.shape)
```

Always Standardization

```
(150, 4)
```

In [8]:
```
standardized_data[:5]
```

Out[8]:
```
array([[-0.90068117,  1.03205722, -1.3412724 , -1.31297673],
       [-1.14301691, -0.1249576 , -1.3412724 , -1.31297673],
       [-1.38535265,  0.33784833, -1.39813811, -1.31297673],
       [-1.50652052,  0.10644536, -1.2844067 , -1.31297673],
       [-1.02184904,  1.26346019, -1.3412724 , -1.31297673]])
```

In [9]:
```
# Try to find the mean and variance of each column
```

PCA using SCIKIT-LEARN

In [10]:
```
# initializing the pca
from sklearn.decomposition import PCA
pca = PCA()
```

Internally SVD

In [11]:
```
pca.fit(standardized_data)
```

only X as it is unsupervised

Out[11]:
```
PCA()
```

In [12]:
```
pca.components_
```

(Eigen Vectors)

Out[12]:
```
array([[ 0.52237162, -0.26335492,  0.58125401,  0.56561105],
       [ 0.37231836,  0.92555649,  0.02109478,  0.06541577],
       [-0.72101681,  0.24203288,  0.14089226,  0.6338014 ],
       [-0.26199559,  0.12413481,  0.80115427, -0.52354627]])
```

4 Components

In [13]:
```
pca.explained_variance_ratio_
```

(Eigen Values)

Out[13]:
```
array([0.72770452, 0.23030523, 0.03683832, 0.00515193])
```

72.7% 23.0% 3% 0.5%

In [14]:
```
np.cumsum(pca.explained_variance_ratio_)
```

→ { 1,2,3,4,5 → Cumulative Sum → 1 3 6 10 15 (2+1) (3+3) (6+4) (10+5)

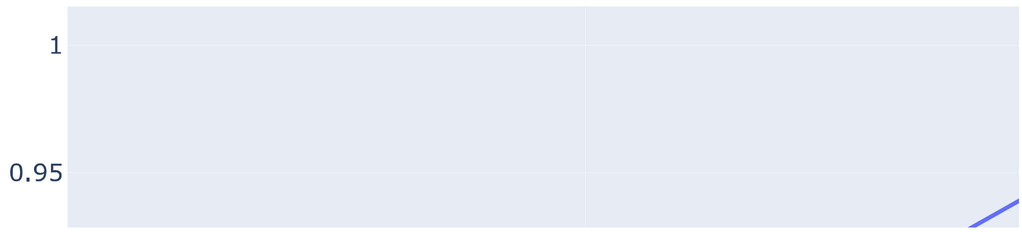
Out[14]:
```
array([0.72770452, 0.95800975, 0.99484807, 1.        ])
```

72.7% 95.8%

In [15]:
```
np.arange(len(pca.explained_variance_ratio_))
```

Out[15]:
```
array([0, 1, 2, 3])
```

In [16]:
```
import plotly.express as px

px.line(x = np.arange(len(pca.explained_variance_ratio_)), y = np.cumsum(pca.exp
```



0 1 2 3

features 1 2 3 4

OBSERVATION: As we can see that only 2 PCA components are explaining 95.8% variance, so lets go ahead and consider these two components.

In [17]:

```
# configuring the parameteres
# number of components = 2
pca_new = PCA(n_components = 2)
pca_new_data = pca_new.fit_transform(standardized_data)

# Lets look at the shape of data after PCA
print("shape = ", pca_new_data.shape)
```

Variance Thresholding internally

Feature transform

PCA
fit → Learns Eigen Values & Eigen Vectors

```
shape =  (150, 2)
```

In [18]:

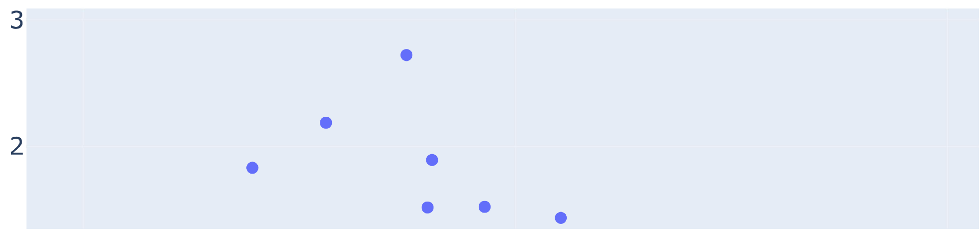
```
pca_df = pd.DataFrame(data=pca_new_data, columns=("1st_principal", "2nd_principa

pca_df["Label"] = label.values

pca_df.head()
```

Out[18]:

	1st_principal	2nd_principal	Label
0	-2.264542	0.505704	Iris-setosa
1	-2.086426	-0.655405	Iris-setosa
2	-2.367950	-0.318477	Iris-setosa
3	-2.304197	-0.575368	Iris-setosa
4	-2.388777	0.674767	Iris-setosa

In [19]:

```
px.scatter(pca_df, x = "1st_principal",
           y = "2nd_principal",
           color = "Label")
```



In [ ]: