Aim: Implementation of dimensionality reduction techniques: Normalization, Transformation, Principal Components Analysis.

A) NORMALIZATION:-

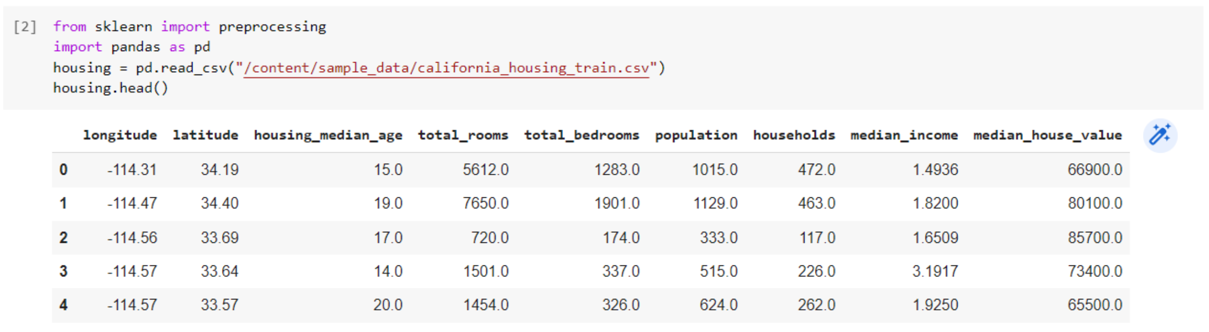
Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. Normalization is also required for some algorithms to model the data correctly.

For example, assume your input dataset contains one column with values ranging from 0 to 1, and another column with values ranging from 10,000 to 100,000. The great difference in the scale of the numbers could cause problems when you attempt to combine the values as features during modeling.

Normalization avoids these problems by creating new values that maintain the general distribution and ratios in the source data, while keeping values within a scale applied across all numeric columns used in the model.

SOURCE CODE:

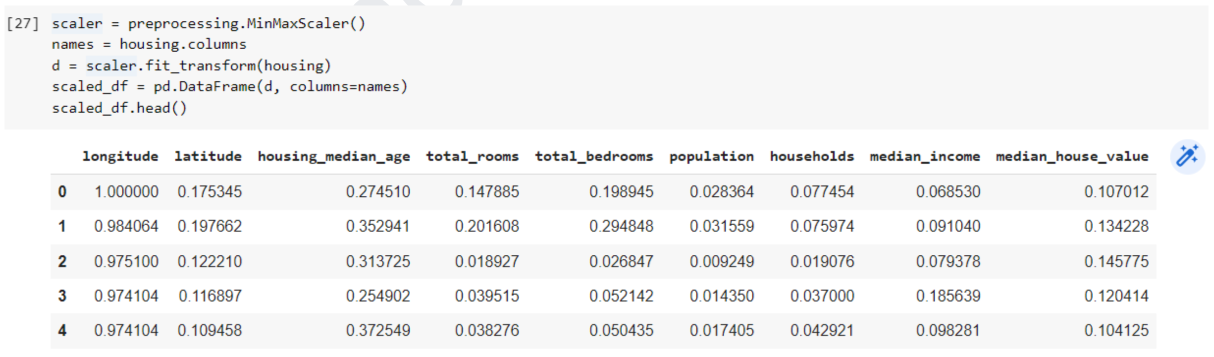
1) Importing Datasets:



```
[2] from sklearn import preprocessing
    import pandas as pd
    housing = pd.read_csv("/content/sample_data/california_housing_train.csv")
    housing.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	472.0	1.4936	66900.0
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	463.0	1.8200	80100.0
2	-114.56	33.69	17.0	720.0	174.0	333.0	117.0	1.6509	85700.0
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226.0	3.1917	73400.0
4	-114.57	33.57	20.0	1454.0	326.0	624.0	262.0	1.9250	65500.0

2) Normalizing- MinMaxScalar Transformation of Dataset:



```
[27] scaler = preprocessing.MinMaxScaler()
     names = housing.columns
     d = scaler.fit_transform(housing)
     scaled_df = pd.DataFrame(d, columns=names)
     scaled_df.head()
```

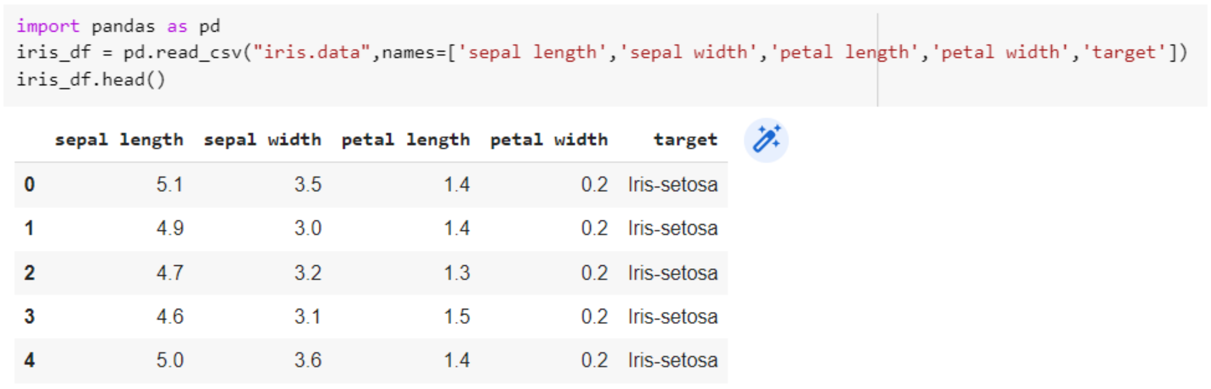
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	1.000000	0.175345	0.274510	0.147885	0.198945	0.028364	0.077454	0.068530	0.107012
1	0.984064	0.197662	0.352941	0.201608	0.294848	0.031559	0.075974	0.091040	0.134228
2	0.975100	0.122210	0.313725	0.018927	0.026847	0.009249	0.019076	0.079378	0.145775
3	0.974104	0.116897	0.254902	0.039515	0.052142	0.014350	0.037000	0.185639	0.120414
4	0.974104	0.109458	0.372549	0.038276	0.050435	0.017405	0.042921	0.098281	0.104125

B) Principal Component Analysis:-

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation that converts a set of correlated variables to a set of uncorrelated variables. PCA is the most widely used tool in exploratory data analysis and in machine learning for predictive models. Moreover, PCA is an unsupervised statistical technique used to examine the interrelations among a set of variables. It is also known as a general factor analysis where regression determines a line of best fit.

SOURCE CODE:

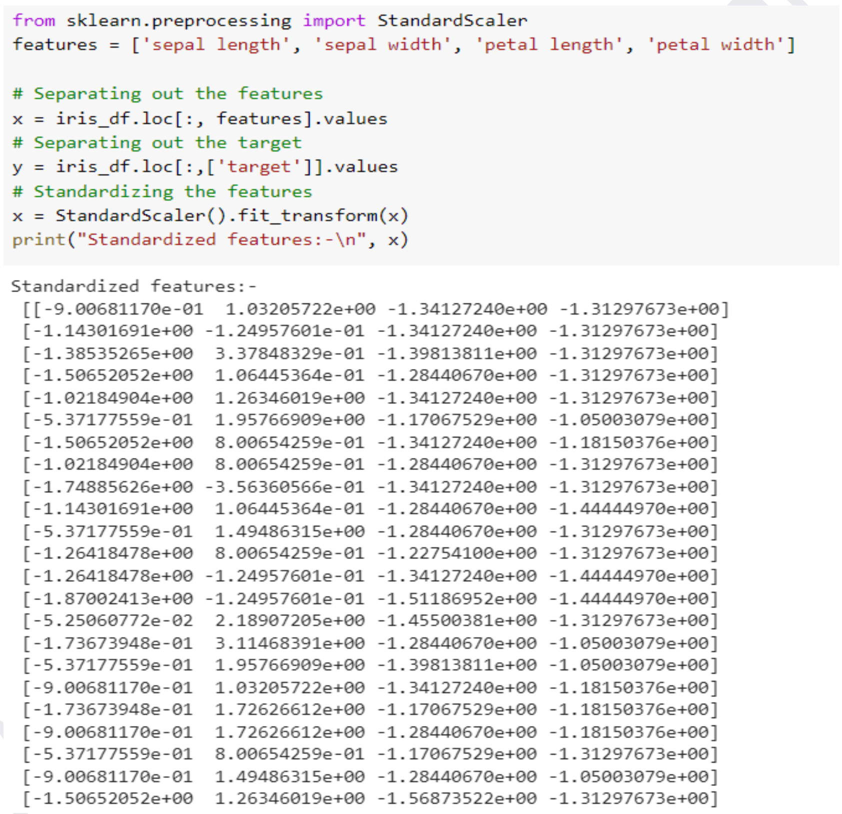
1) Importing Dataset:



```python
import pandas as pd
iris_df = pd.read_csv("iris.data",names=['sepal length','sepal width','petal length','petal width','target'])
iris_df.head()
```

	sepal length	sepal width	petal length	petal width	target
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

2) Standardize the Data:



```python
from sklearn.preprocessing import StandardScaler
features = ['sepal length', 'sepal width', 'petal length', 'petal width']

# Separating out the features
x = iris_df.loc[:, features].values
# Separating out the target
y = iris_df.loc[:,['target']].values
# Standardizing the features
x = StandardScaler().fit_transform(x)
print("Standardized features:-\n", x)
```

```
Standardized features:-
 [[-9.00681170e-01  1.03205722e+00 -1.34127240e+00 -1.31297673e+00]
 [-1.14301691e+00 -1.24957601e-01 -1.34127240e+00 -1.31297673e+00]
 [-1.38535265e+00  3.37848329e-01 -1.39813811e+00 -1.31297673e+00]
 [-1.50652052e+00  1.06445364e-01 -1.28440670e+00 -1.31297673e+00]
 [-1.02184904e+00  1.26346019e+00 -1.34127240e+00 -1.31297673e+00]
 [-5.37177559e-01  1.95766909e+00 -1.17067529e+00 -1.05003079e+00]
 [-1.50652052e+00  8.00654259e-01 -1.34127240e+00 -1.18150376e+00]
 [-1.02184904e+00  8.00654259e-01 -1.28440670e+00 -1.31297673e+00]
 [-1.74885626e+00 -3.56360566e-01 -1.34127240e+00 -1.31297673e+00]
 [-1.14301691e+00  1.06445364e-01 -1.28440670e+00 -1.44444970e+00]
 [-5.37177559e-01  1.49486315e+00 -1.28440670e+00 -1.31297673e+00]
 [-1.26418478e+00  8.00654259e-01 -1.22754100e+00 -1.31297673e+00]
 [-1.26418478e+00 -1.24957601e-01 -1.34127240e+00 -1.44444970e+00]
 [-1.87002413e+00 -1.24957601e-01 -1.51186952e+00 -1.44444970e+00]
 [-5.25060772e-02  2.18907205e+00 -1.45500381e+00 -1.31297673e+00]
 [-1.73673948e-01  3.11468391e+00 -1.28440670e+00 -1.05003079e+00]
 [-5.37177559e-01  1.95766909e+00 -1.39813811e+00 -1.05003079e+00]
 [-9.00681170e-01  1.03205722e+00 -1.34127240e+00 -1.18150376e+00]
 [-1.73673948e-01  1.72626612e+00 -1.17067529e+00 -1.18150376e+00]
 [-9.00681170e-01  1.72626612e+00 -1.28440670e+00 -1.18150376e+00]
 [-5.37177559e-01  8.00654259e-01 -1.17067529e+00 -1.31297673e+00]
 [-9.00681170e-01  1.49486315e+00 -1.28440670e+00 -1.05003079e+00]
 [-1.50652052e+00  1.26346019e+00 -1.56873522e+00 -1.31297673e+00]
```

3) PCA Projection to 2D:

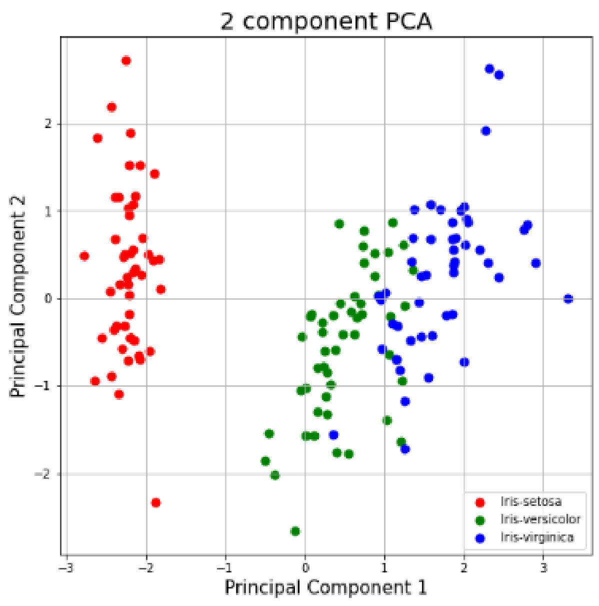
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2'])
finalDf = pd.concat([principalDf, iris_df[['target']]], axis = 1)
finalDf
```

	principal component 1	principal component 2	target
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
...	...	...	...
145	1.870522	0.382822	Iris-virginica
146	1.558492	-0.905314	Iris-virginica
147	1.520845	0.266795	Iris-virginica
148	1.376391	1.016362	Iris-virginica
149	0.959299	-0.022284	Iris-virginica

150 rows × 3 columns

4) Visualize 2D Projection:

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
 indicesToKeep = finalDf['target'] == target
 ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
 , finalDf.loc[indicesToKeep, 'principal component 2']
 , c = color
 , s = 50)
ax.legend(targets)
ax.grid()
```



CONCLUSION:

From this practical, I have learned and implemented dimensionality reduction techniques: Normalization, Transformation, Principal Components Analysis.