```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
%matplotlib inline
```

```
df = pd.read_csv('/content/Iris.csv')
df
```

	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

150 rows × 6 columns

```
df.head()
```

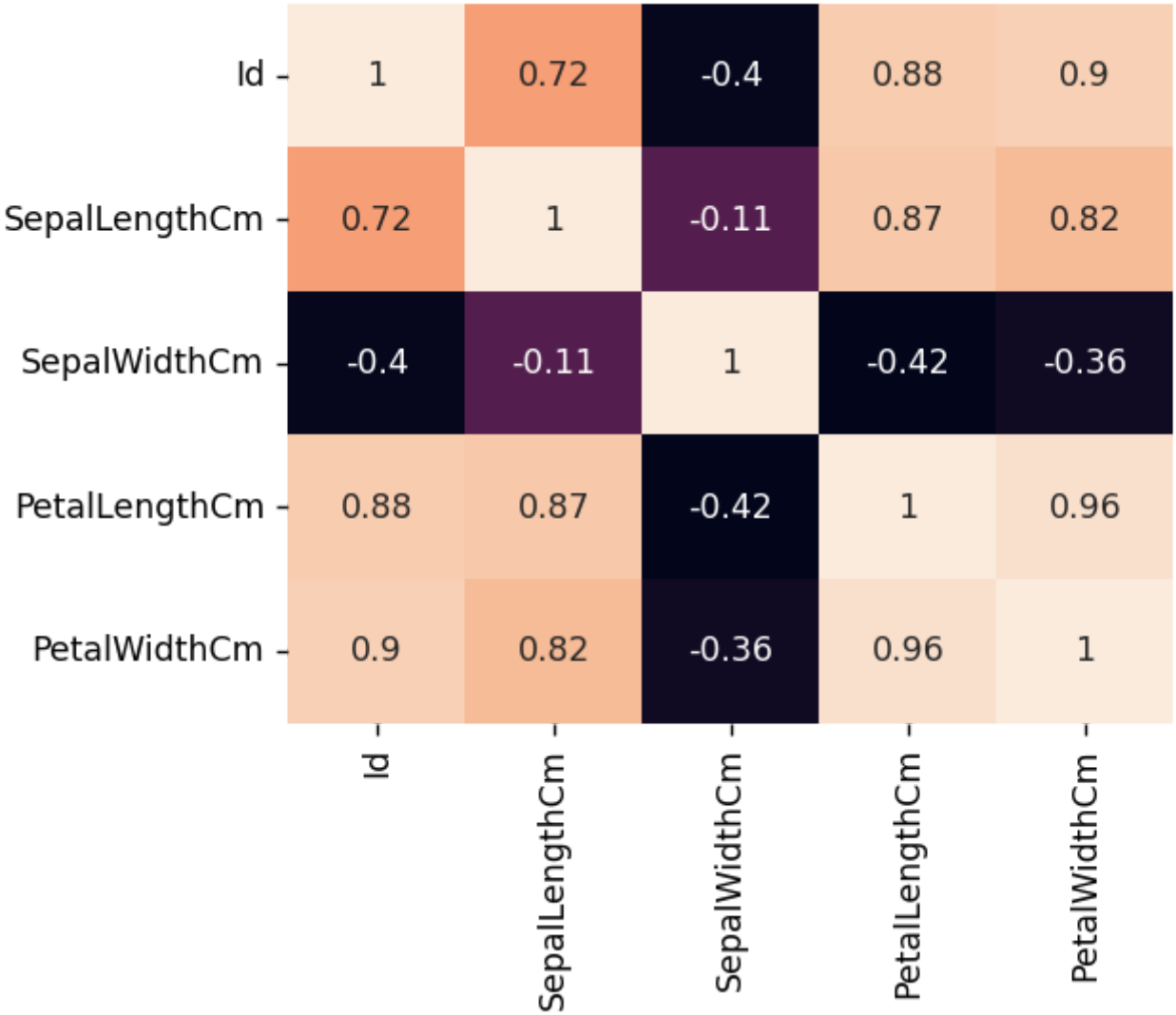
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
df.isnull().sum()
```

```
Id              0
SepalLengthCm   0
SepalWidthCm    0
PetalLengthCm   0
PetalWidthCm    0
```

```
Species          0
dtype: int64
```

```
# Heatmap shows the correlation
plt.figure(dpi = 125)#dpi gives clarity of the figure (dpi- dotts per inch)
sns.heatmap(np.round (df.corr(), 2), annot = True)
plt.show()
```




```
X = df.iloc[:,0:4].values
y = df.iloc[:,4].values
```

```
X
```

```
[ 93. ,   5.8,   2.6,   4. ],
[ 94. ,   5. ,   2.3,   3.3],
[ 95. ,   5.6,   2.7,   4.2],
[ 96. ,   5.7,   3. ,   4.2],
[ 97. ,   5.7,   2.9,   4.2],
[ 98. ,   6.2,   2.9,   4.3],
[ 99. ,   5.1,   2.5,   3. ],
[100. ,   5.7,   2.8,   4.1],
[101. ,   6.3,   3.3,   6. ],
[102. ,   5.8,   2.7,   5.1],
[103. ,   7.1,   3. ,   5.9],
```

```
[105. ,   7.1,   3. ,   5.9],
[104. ,   6.3,   2.9,   5.6],
[105. ,   6.5,   3. ,   5.8],
[106. ,   7.6,   3. ,   6.6],
[107. ,   4.9,   2.5,   4.5],
[108. ,   7.3,   2.9,   6.3],
[109. ,   6.7,   2.5,   5.8],
[110. ,   7.2,   3.6,   6.1],
[111. ,   6.5,   3.2,   5.1],
[112. ,   6.4,   2.7,   5.3],
[113. ,   6.8,   3. ,   5.5],
[114. ,   5.7,   2.5,   5. ],
[115. ,   5.8,   2.8,   5.1],
[116. ,   6.4,   3.2,   5.3],
[117. ,   6.5,   3. ,   5.5],
[118. ,   7.7,   3.8,   6.7],
[119. ,   7.7,   2.6,   6.9],
[120. ,   6. ,   2.2,   5. ],
[121. ,   6.9,   3.2,   5.7],
[122. ,   5.6,   2.8,   4.9],
[123. ,   7.7,   2.8,   6.7],
[124. ,   6.3,   2.7,   4.9],
[125. ,   6.7,   3.3,   5.7],
[126. ,   7.2,   3.2,   6. ],
[127. ,   6.2,   2.8,   4.8],
[128. ,   6.1,   3. ,   4.9],
[129. ,   6.4,   2.8,   5.6],
[130. ,   7.2,   3. ,   5.8],
[131. ,   7.4,   2.8,   6.1],
[132. ,   7.9,   3.8,   6.4],
[133. ,   6.4,   2.8,   5.6],
[134. ,   6.3,   2.8,   5.1],
[135. ,   6.1,   2.6,   5.6],
[136. ,   7.7,   3. ,   6.1],
[137. ,   6.3,   3.4,   5.6],
[138. ,   6.4,   3.1,   5.5],
[139. ,   6. ,   3. ,   4.8],
[140. ,   6.9,   3.1,   5.4],
[141. ,   6.7,   3.1,   5.6],
[142. ,   6.9,   3.1,   5.1],
[143. ,   5.8,   2.7,   5.1],
[144. ,   6.8,   3.2,   5.9],
[145. ,   6.7,   3.3,   5.7],
[146. ,   6.7,   3. ,   5.2],
[147. ,   6.3,   2.5,   5. ],
[148. ,   6.5,   3. ,   5.2],
[149. ,   6.2,   3.4,   5.4],
[150. ,   5.9,   3. ,   5.1]])
```

y

```
array([0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.2, 0.1,
       0.1, 0.2, 0.4, 0.4, 0.3, 0.3, 0.3, 0.2, 0.4, 0.2, 0.5, 0.2, 0.2,
       0.4, 0.2, 0.2, 0.2, 0.2, 0.4, 0.1, 0.2, 0.1, 0.2, 0.2, 0.1, 0.2,
       0.2, 0.3, 0.3, 0.2, 0.6, 0.4, 0.3, 0.2, 0.2, 0.2, 0.2, 1.4, 1.5,
       1.5, 1.3, 1.5, 1.3, 1.6, 1. , 1.3, 1.4, 1. , 1.5, 1. , 1.4, 1.3,
       1.4, 1.5, 1. , 1.5, 1.1, 1.8, 1.3, 1.5, 1.2, 1.3, 1.4, 1.4, 1.7,
       1.5, 1. , 1.1, 1. , 1.2, 1.6, 1.5, 1.6, 1.5, 1.3, 1.3, 1.3, 1.2,
       1.4, 1.2, 1. , 1.3, 1.2, 1.3, 1.3, 1.1, 1.3, 2.5, 1.9, 2.1, 1.8,
```

```
       2.2, 2.1, 1.7, 1.8, 1.8, 2.5, 2. , 1.9, 2.1, 2. , 2.4, 2.3, 1.8,
       2.2, 2.3, 1.5, 2.3, 2. , 2. , 1.8, 2.1, 1.8, 1.8, 1.8, 2.1, 1.6,
       1.9, 2. , 2.2, 1.5, 1.4, 2.3, 2.4, 1.8, 1.8, 2.1, 2.4, 2.3, 1.9,
       2.3, 2.5, 2.3, 1.9, 2. , 2.3, 1.8])
```

```
from sklearn.preprocessing import StandardScaler
X_std = StandardScaler().fit_transform(X)
```

```
X_std
```

```
       [-0.93532822, -1.14301691,  0.10644536, -1.2844067 ],
       [-0.9122337 , -1.02184904,  0.33784833, -1.45500381],
       [-0.88913917, -0.41600969,  1.03205722, -1.39813811],
       [-0.86604465, -1.14301691,  0.10644536, -1.2844067 ],
       [-0.84295013, -1.74885626, -0.1249576 , -1.39813811],
       [-0.8198556 , -0.90068117,  0.80065426, -1.2844067 ],
       [-0.79676108, -1.02184904,  1.03205722, -1.39813811],
       [-0.77366655, -1.62768839, -1.74477836, -1.39813811],
       [-0.75057203, -1.74885626,  0.33784833, -1.39813811],
       [-0.72747751, -1.02184904,  1.03205722, -1.227541  ],
       [-0.70438298, -0.90068117,  1.72626612, -1.05694388],
       [-0.68128846, -1.26418478, -0.1249576 , -1.3412724 ],
       [-0.65819393, -0.90068117,  1.72626612, -1.227541  ],
       [-0.63509941, -1.50652052,  0.33784833, -1.3412724 ],
       [-0.61200489, -0.65834543,  1.49486315, -1.2844067 ],
       [-0.58891036, -1.02184904,  0.56925129, -1.3412724 ],
       [-0.56581584,  1.40150837,  0.33784833,  0.53529583],
       [-0.54272131,  0.67450115,  0.33784833,  0.42156442],
       [-0.51962679,  1.2803405 ,  0.10644536,  0.64902723],
       [-0.49653227, -0.41600969, -1.74477836,  0.1372359 ],
       [-0.47343774,  0.79566902, -0.58776353,  0.47843012],
       [-0.45034322, -0.17367395, -0.58776353,  0.42156442],
       [-0.42724869,  0.55333328,  0.56925129,  0.53529583],
       [-0.40415417, -1.14301691, -1.51337539, -0.26082403],
       [-0.38105965,  0.91683689, -0.35636057,  0.47843012],
       [-0.35796512, -0.7795133 , -0.8191665 ,  0.08037019],
       [-0.3348706 , -1.02184904, -2.43898725, -0.14709262],
       [-0.31177607,  0.06866179, -0.1249576 ,  0.25096731],
       [-0.28868155,  0.18982966, -1.97618132,  0.1372359 ],
       [-0.26558703,  0.31099753, -0.35636057,  0.53529583],
       [-0.2424925 , -0.29484182, -0.35636057, -0.09022692],
       [-0.21939798,  1.03800476,  0.10644536,  0.36469871],
       [-0.19630345, -0.29484182, -0.1249576 ,  0.42156442],
       [-0.17320893, -0.05250608, -0.8191665 ,  0.1941016 ],
       [-0.15011441,  0.4321654 , -1.97618132,  0.42156442],
       [-0.12701988, -0.29484182, -1.28197243,  0.08037019],
       [-0.10392536,  0.06866179,  0.33784833,  0.59216153],
       [-0.08083083,  0.31099753, -0.58776353,  0.1372359 ],
       [-0.05773631,  0.55333328, -1.28197243,  0.64902723],
       [-0.03464179,  0.31099753, -0.58776353,  0.53529583],
       [-0.01154726,  0.67450115, -0.35636057,  0.30783301],
       [ 0.01154726,  0.91683689, -0.1249576 ,  0.36469871],
       [ 0.03464179,  1.15917263, -0.58776353,  0.59216153],
       [ 0.05773631,  1.03800476, -0.1249576 ,  0.70589294],
       [ 0.08083083,  0.18982966, -0.35636057,  0.42156442],
       [ 0.10392536, -0.17367395, -1.05056946, -0.14709262],
       [ 0.12701988, -0.41600969, -1.51337539,  0.02350449],
       [ 0.15011441, -0.41600969, -1.51337539, -0.03336121],
       [ 0.17320893, -0.05250608, -0.8191665 ,  0.08037019],
```

```
[ 0.17926895,  0.05250608, -0.8191665 ,  0.08357819],
[ 0.19630345,  0.18982966, -0.8191665 ,  0.76275864],
[ 0.21939798, -0.53717756, -0.1249576 ,  0.42156442],
[ 0.2424925 ,  0.18982966,  0.80065426,  0.42156442],
[ 0.26558703,  1.03800476,  0.10644536,  0.53529583],
[ 0.28868155,  0.55333328, -1.74477836,  0.36469871],
[ 0.31177607, -0.29484182, -0.1249576 ,  0.1941016 ],
[ 0.3348706 , -0.41600969, -1.28197243,  0.1372359 ],
[ 0.35796512, -0.41600969, -1.05056946,  0.36469871],
[ 0.38105965,  0.31099753, -0.1249576 ,  0.47843012],
[ 0.40415417,  0.05250608, -1.05056946,  0.1372359 ],
```

```
print('Covariance matrix \n')
cov_mat= np.cov(X_std, rowvar=False)
cov_mat
```

```
Covariance matrix

array([[ 1.00671141,  0.72148618, -0.40039813,  0.8886718 ],
       [ 0.72148618,  1.00671141, -0.11010327,  0.87760486],
       [-0.40039813, -0.11010327,  1.00671141, -0.42333835],
       [ 0.8886718 ,  0.87760486, -0.42333835,  1.00671141]])
```

```
cov_mat = np.cov(X_std.T)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)
```

```
Eigenvectors
[[ 0.55318314  0.31153594 -0.77256222 -0.00902118]
 [ 0.51774664  0.48025478  0.56930389 -0.42093567]
 [-0.28847469 -0.16889872 -0.2641027  -0.90471285]
 [ 0.58541369 -0.80235523  0.09638701 -0.06501105]]

Eigenvalues
[2.83122907 0.04725055 0.22729518 0.92107083]
```

```
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i])
for i in range(len(eig_vals))]
print(type(eig_pairs))
#Sort the (eigenvalue, eigenvector) tuples from high to low eig_pairs.sort()
eig_pairs.reverse()
print("\n",eig_pairs)
#Visually confirm that the list is correctly sorted by decreasing eigenvalues
print('\n\n\nEigenvalues in descending order:')
for i in eig_pairs:
    print(i[0])
```

```
<class 'list'>

 [(0.9210708329025815, array([-0.00902118, -0.42093567, -0.90471285, -0.06501105])),
```

```
Eigenvalues in descending order:
0.9210708329025815
```

```
0.22729518173179655
0.04725054797568571
2.8312290749738294
```

```
tot = sum(eig_vals)
print("\n",tot)
var_exp = [(i / tot)*100 for i in sorted(eig_vals, reverse=True)]
print("\n\n1. Variance Explained\n",var_exp)
cum_var_exp = np.cumsum(var_exp)
print("\n\n2. Cumulative Variance Explained\n",cum_var_exp)
print("\n\n3. Percentage of variance the first two principal components each contain\n ",v
print("\n\n4. Percentage of variance the first two principal components together contain\n
```

```
 4.026845637583893


1. Variance Explained
 [70.30885536185009, 22.87325901708077, 5.644497013006281, 1.1733886080628617]


2. Cumulative Variance Explained
 [ 70.30885536  93.18211438  98.82661139 100.        ]


3. Percentage of variance the first two principal components each contain
  [70.30885536185009, 22.87325901708077]


4. Percentage of variance the first two principal components together contain
 93.18211437893086
```

```
print(eig_pairs[0][1])
print(eig_pairs[1][1])
matrix_w = np.hstack((eig_pairs[0][1].reshape(4,1), eig_pairs[1][1].reshape(4,1)))
#hstack: Stacks arrays in sequence horizontally (column wise). print('Matrix W:\n', matrix
```

```
[-0.00902118 -0.42093567 -0.90471285 -0.06501105]
[-0.77256222  0.56930389 -0.2641027   0.09638701]
```

```
Y = X_std.dot(matrix_w)
principalDf = pd.DataFrame(data = Y , columns = ['principal component 1', 'principal compo
principalDf.head()
```

	principal component 1	principal component 2

```
finalDf = pd.concat([principalDf,pd.DataFrame(y,columns = ['species'])], axis = 1)
finalDf.head()
```

	principal component 1	principal component 2	species
0	-0.451868	0.414614	0.2
1	0.696698	0.564380	0.2
2	0.383488	0.280866	0.2
3	0.636243	0.266119	0.2
4	-0.611050	0.213151	0.2

```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X_std)
principalDf = pd.DataFrame(data = principalComponents , columns = ['principal component 1'
principalDf.head(5) # prints the top 5 rows
```

	principal component 1	principal component 2
0	-2.501021	0.451868
1	-2.279945	-0.696698
2	-2.559435	-0.383488
3	-2.476060	-0.636243
4	-2.579407	0.611050

```
finalDf = pd.concat([principalDf, finalDf[['species']]], axis = 1)
finalDf.head(5)
```

	principal component 1	principal component 2	species
0	-2.501021	0.451868	0.2
1	-2.279945	-0.696698	0.2
2	-2.559435	-0.383488	0.2
3	-2.476060	-0.636243	0.2
4	-2.579407	0.611050	0.2