week 7

Feature Construction: (Use packages that are applicable)

1. Dummy coding categorical(nominal) variables.
2. Encoding categorical(ordinal) variables.
3. Transforming numeric(continuous)features to categorical features Feature Extraction: (Use packages that are applicable)
4. Principal Component Analysis (PCA)
5. Singular Value Decomposition (SVD)
6. Linear Discriminant Analysis (LDA)
7. Feature Subset Selection Compiler Design

Data set :IRIS attributes: petal length, petal width, sepal length, sepal width

In [12]:
```
#reading the files
import pandas as pd
import numpy as np
df= pd.read_csv("C:/Users/saile/Desktop/IoT/datasets/iris.csv")
```

In [13]:
```
df
```

Out[13]:

	Id	Sepal Length	Sepal Width	Petal Length	Petal Width	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

dummy coding categorical variables

In [15]:
```
#using pandas to create dummy variables
dummies =pd.get_dummies(df.Species)
```

In [16]:
```
dummies
```

Out[16]:

	Iris-setosa	Iris-versicolor	Iris-virginica
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
...	...	...	...
145	0	0	1
146	0	0	1
147	0	0	1
148	0	0	1
149	0	0	1

150 rows × 3 columns

In [18]:
```
#merging dummy and original dataset
Iris =pd.concat([df,dummies],axis='columns')
Iris
```

Out[18]:

	Id	Sepal Length	Sepal Width	Petal Length	Petal Width	Species	Iris-setosa	Iris-versicolor	Iris-virginica
0	1	5.1	3.5	1.4	0.2	Iris-setosa	1	0	0
1	2	4.9	3.0	1.4	0.2	Iris-setosa	1	0	0
2	3	4.7	3.2	1.3	0.2	Iris-setosa	1	0	0
3	4	4.6	3.1	1.5	0.2	Iris-setosa	1	0	0
4	5	5.0	3.6	1.4	0.2	Iris-setosa	1	0	0
...	...	...	...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica	0	0	1
146	147	6.3	2.5	5.0	1.9	Iris-virginica	0	0	1
147	148	6.5	3.0	5.2	2.0	Iris-virginica	0	0	1

	Id	Sepal Length	Sepal Width	Petal Length	Petal Width	Species	Iris-setosa	Iris-versicolor	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica	0	0	1
149	150	5.9	3.0	5.1	1.8	Iris-virginica	0	0	1

150 rows × 9 columns

In [19]:
```
Iris =Iris.drop(['Species'],axis='columns')
Iris
```

Out[19]:

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

150 rows × 8 columns

In [20]:
```
#Labelling encoding of nomial data usig skelarn
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
iris_1=df
iris_1.Species=le.fit_transform(iris_1.Species)
iris_1
```

Out[20]:

	Id	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0

	Id	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	2
146	147	6.3	2.5	5.0	1.9	2
147	148	6.5	3.0	5.2	2.0	2
148	149	6.2	3.4	5.4	2.3	2
149	150	5.9	3.0	5.1	1.8	2

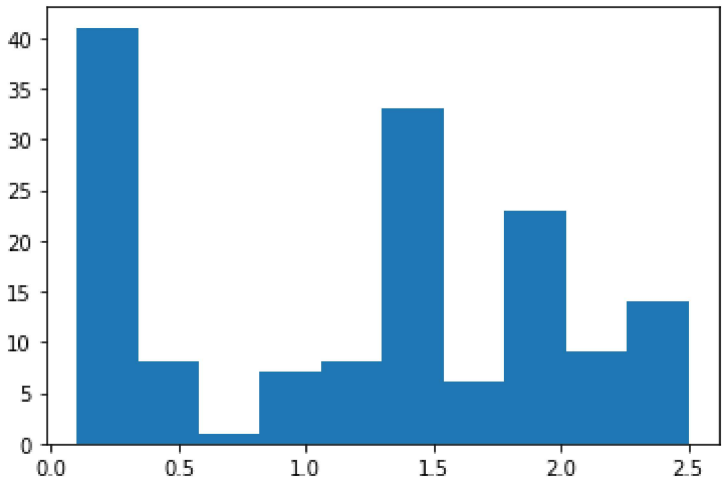
150 rows × 6 columns

2 encoding categorical variables

3 transforming numerical features to categorical features

In [34]:
```
import matplotlib.pyplot as plt
x=df['Petal Width ']
plt.hist(x)
plt.show
```

Out[34]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [42]:
```
df['Petal category']=pd.cut(df['Petal Width '] ,bins=[0,1,2,2.5], labels=['small','medi
print(df['Petal category'])
print(df['Petal category'].value_counts())
```

```
0       small
1       small
2       small
3       small
4       small
       ...
```

```
145      long
146    medium
147    medium
148      long
149    medium
Name: Petal category, Length: 150, dtype: category
Categories (3, object): ['small' < 'medium' < 'long']
medium    70
small     57
long      23
Name: Petal category, dtype: int64
```

Observation

since categorization of the flower can be done through the petal width we have categorised the continous data into 3 categories as small medium and long

Encoding categorical variables

In [47]:
```
from sklearn.preprocessing import OrdinalEncoder
encoder=OrdinalEncoder()
df['Petal category'] =encoder.fit_transform(df[['Petal category']])
df
```

Out[47]:

	Id	Sepal Length	Sepal Width	Petal Length	Petal Width	Species	Petal category
0	1	5.1	3.5	1.4	0.2	0	2.0
1	2	4.9	3.0	1.4	0.2	0	2.0
2	3	4.7	3.2	1.3	0.2	0	2.0
3	4	4.6	3.1	1.5	0.2	0	2.0
4	5	5.0	3.6	1.4	0.2	0	2.0
...	...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	2	0.0
146	147	6.3	2.5	5.0	1.9	2	1.0
147	148	6.5	3.0	5.2	2.0	2	1.0
148	149	6.2	3.4	5.4	2.3	2	0.0
149	150	5.9	3.0	5.1	1.8	2	1.0

150 rows × 7 columns

observations

scikitlearn Ordinal Encoder is used to convert the categorical data to continous data

feature extraction

Feature Extraction: (Use packages that are applicable)

1. Principal Component Analysis (PCA)
2. Singular Value Decomposition (SVD)
3. Linear Discriminant Analysis (LDA)
4. Feature Subset Selection

In [49]:
```
# read the csv files

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA
df= datasets.load_iris()
```

In [50]:
```
df
```

Out[50]:
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
```

```
[4.9, 3.1, 1.5, 0.2],
[5. , 3.2, 1.2, 0.2],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.6, 1.4, 0.1],
[4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
[6.9, 3.1, 4.9, 1.5],
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
```

```
       [5.6, 2.7, 4.2, 1.3],
       [5.7, 3. , 4.2, 1.2],
       [5.7, 2.9, 4.2, 1.3],
       [6.2, 2.9, 4.3, 1.3],
       [5.1, 2.5, 3. , 1.1],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 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, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
 'frame': None,
 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set
Characteristics:**\n\n    :Number of Instances: 150 (50 in each of three classes)\n    :
Number of Attributes: 4 numeric, predictive attributes and the class\n    :Attribute Inf
ormation:\n        - sepal length in cm\n        - sepal width in cm\n        - petal le
ngth in cm\n        - petal width in cm\n        - class:\n                - Iris-Setosa
\n                - Iris-Versicolour\n                - Iris-Virginica\n
\n    :Summary Statistics:\n\n    ============== ==== ==== ======= ===== ==============
=====\n                    Min  Max   Mean    SD   Class Correlation\n    ==============
==== ==== ======= ===== ====================\n    sepal length:   4.3  7.9   5.84   0.83
0.7826\n    sepal width:    2.0  4.4   3.05   0.43   -0.4194\n    petal length:   1.0
6.9   3.76   1.76    0.9490  (high!)\n    petal width:    0.1  2.5   1.20   0.76    0.95
65  (high!)\n    ============== ==== ==== ======= ===== ====================\n\n    :Mis
sing Attribute Values: None\n    :Class Distribution: 33.3% for each of 3 classes.\n
:Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n
:Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fisher. The datase
t is taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but not as in the UC
I\nMachine Learning Repository, which has two wrong data points.\n\nThis is perhaps the
best known database to be found in the\npattern recognition literature.  Fisher\'s paper
is a classic in the field and\nis referenced frequently to this day.  (See Duda & Hart,
for example.)  The\ndata set contains 3 classes of 50 instances each, where each class r
efers to a\ntype of iris plant.  One class is linearly separable from the other 2; the\n
latter are NOT linearly separable from each other.\n\n.. topic:: References\n\n   - Fish
er, R.A. "The use of multiple measurements in taxonomic problems"\n     Annual Eugenics,
7, Part II, 179-188 (1936); also in "Contributions to\n     Mathematical Statistics" (Jo
hn Wiley, NY, 1950).\n   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Sc
ene Analysis.\n     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n
- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n     Structure
and Classification Rule for Recognition in Partially Exposed\n     Environments".  IEEE
Transactions on Pattern Analysis and Machine\n     Intelligence, Vol. PAMI-2, No. 1, 67
-71.\n   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n
on Information Theory, May 1972, 431-433.\n   - See also: 1988 MLC Proceedings, 54-64.
Cheeseman et al"s AUTOCLASS II\n     conceptual clustering system finds 3 classes in th
e data.\n   - Many, many more ...',
 'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
 'filename': 'iris.csv',
 'data_module': 'sklearn.datasets.data'}
```

Principle component analysis

In [52]:

```
# Determine the initial dimension of the data
X=df.data
Y=df.target
print(X.shape,Y.shape)
```

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

In [54]:

```
#PCA for taregt dimension of the dataset
pca=PCA(n_components =2)
pca.fit(X)
```

Out[54]:
```
PCA
PCA(n_components=2)
```

In [55]:
```
#visualizing  principle components
pca.components_
```

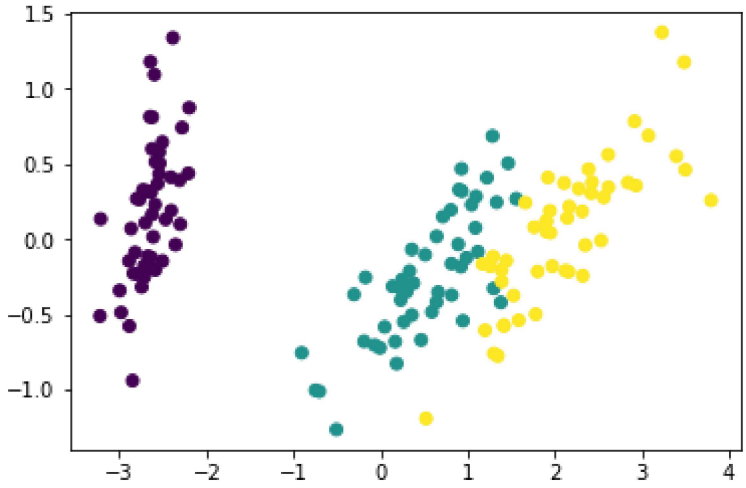
Out[55]:
```
array([[ 0.36138659, -0.08452251,  0.85667061,  0.3582892 ],
       [ 0.65658877,  0.73016143, -0.17337266, -0.07548102]])
```

In [56]:
```
#transforming the data from 4-D to 2-D using PCA
z=pca.transform(X)
z.shape
```

Out[56]:
```
(150, 2)
```

In [59]:
```
#scatter plot
plt.scatter(z[:,0],z[:,1],c=Y)
```

Out[59]:
```
<matplotlib.collections.PathCollection at 0x2ca29cd5760>
```



In [61]:
```
#variance ratio of the target dimensions

pca.explained_variance_ratio_
```

Out[61]:
```
array([0.92461872, 0.05306648])
```

observations

4-D data converted to 2-D data Total variance of the data is equal to the sum of variance of the principle components

2 .Linear Discriminant analysis(LDA)

In [62]:
```
# Determine the initial dimension of the data
X=df.data
Y=df.target
print(X.shape,Y.shape)
```

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

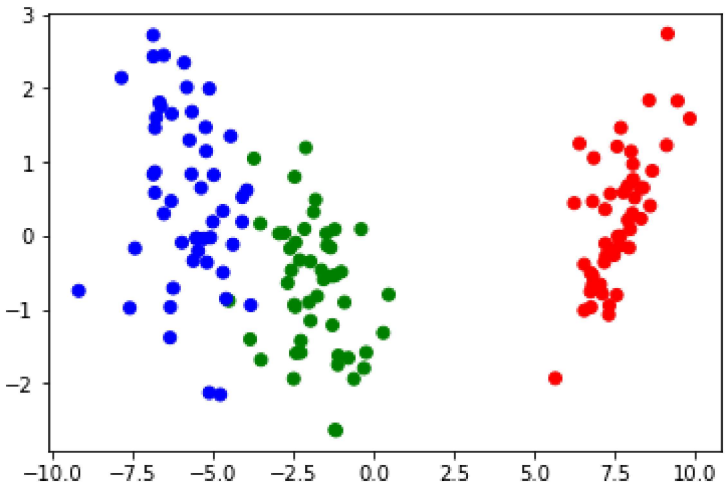
In [64]:
```
#Decomposing 4D to 2D using LDA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda=LinearDiscriminantAnalysis(n_components=2)
X_r2=lda.fit(X,Y).transform(X)
```

In [65]:
```
#getting the variance ratio
lda.explained_variance_ratio_
```

Out[65]:
```
array([0.9912126, 0.0087874])
```

In [69]:
```
#visualizing the 2d data in the form of scater plot
colors=['red','green','blue','yellow']
vectorizer=np.vectorize(lambda x:colors[x % len(colors)])
plt.scatter(X_r2[:,0],X_r2[:,1],c=vectorizer(Y))
```

Out[69]:
```
<matplotlib.collections.PathCollection at 0x2ca29d6d880>
```



Observation

4-D data converted to 2-D data using LDA

Feature subset selection

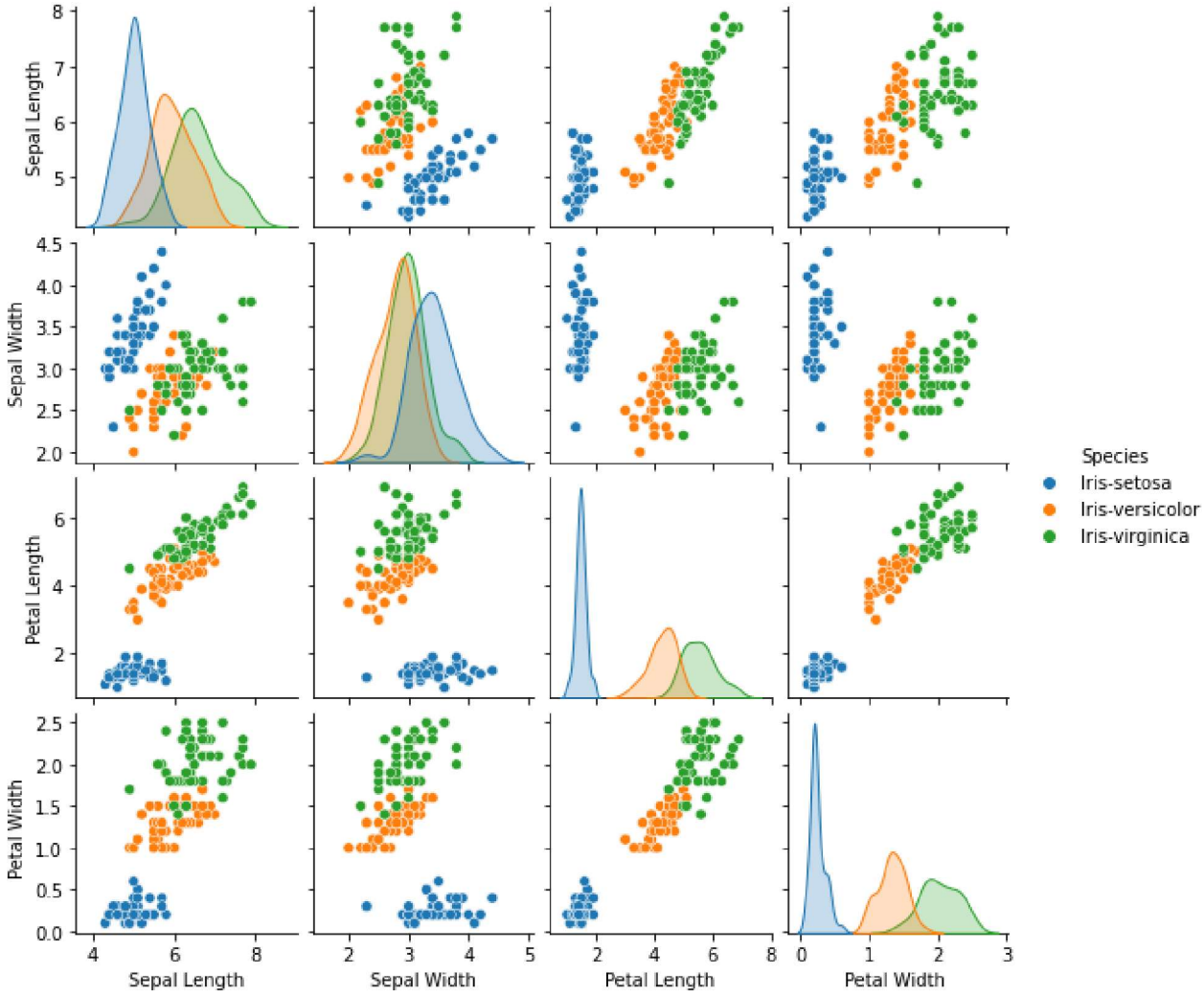
Filter Approach

In [75]:
```
iris= pd.read_csv("C:/Users/saile/Desktop/IoT/datasets/iris.csv")
```

```
#visualizing using pair plot
import seaborn as sns
sns.pairplot(iris.drop(['Id'],axis =1),
             hue='Species',height=2)
```

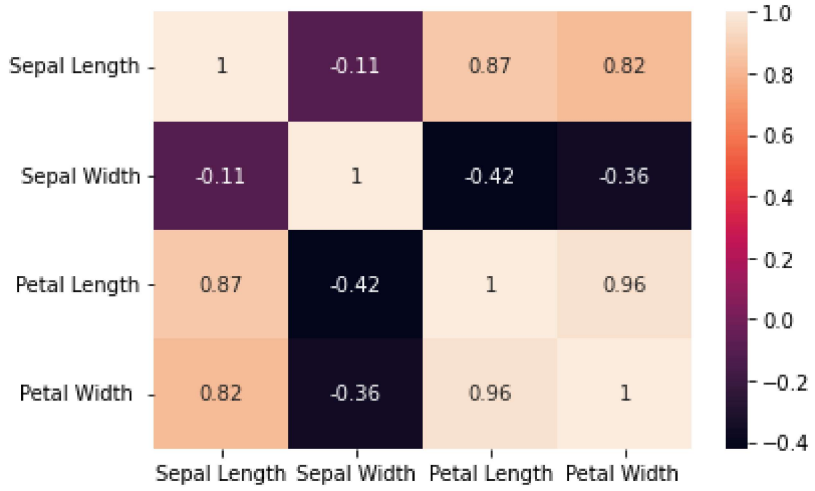
Out[75]:

```
<seaborn.axisgrid.PairGrid at 0x2ca2938a730>
```



In [76]:

```
#visualization correlation using heatmap
sns.heatmap(iris.corr(method='pearson').drop(['Id'],axis =1).drop(['Id'],axis =0),annot
plt.show()
```



Observations

In filter approach we use statistics in feature selction Therefore pearsons correlation is used in the above to selct features with high correlation Since petal leghth and petal width have high correlation ,these two can be used to classify the flower species

In [ ]: