Machine Learning and Statistics Assignment 1

Importing Required Libraries

import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns from sklearn.preprocessing import Imputer from sklearn import neighbors, datasets, model_selection, metrics from sklearn.neighbors import KNeighborsClassifier from sklearn.cross_validation import train_test_split

Importing the predefined dataset Iris in sklearn

In [3]:

```
iris=datasets.load_iris()
iris
```

Out[3]:

```
{'DESCR': 'Iris Plants Database\n====================\n\nNotes\n----
-\nData Set Characteristics:\n    :Number of Instances: 150 (50 in e
ach of three classes)\n    :Number of Attributes: 4 numeric, predict
ive attributes and the class\n    :Attribute Information:\n        -
sepal length in cm\n        - sepal width in cm\n        - petal len
gth in cm\n        - petal width in cm\n        - class:\n
      - Iris-Setosa\n                - Iris-Versicolour\n
     - Iris-Virginica\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  (h
igh!)\n    petal width:    0.1  2.5   1.20  0.76     0.9565  (high!)
\n    ============== ==== ==== ======= ===== ====================\n
\n    :Missing Attribute Values: None\n    :Class Distribution: 33.
3% for each of 3 classes.\n    :Creator: R.A. Fisher\n    :Donor: Mi
chael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988
\n\nThis is a copy of UCI ML iris datasets.\nhttp://archive.ics.uci.
edu/ml/datasets/Iris\n\nThe famous Iris database, first used by Sir
R.A Fisher\n\nThis is perhaps the best known database to be found in
the\npattern recognition literature.  Fisher\'s paper is a classic i
n the field and\nis referenced frequently to this day.  (See Duda &
Hart, for example.)  The\ndata set contains 3 classes of 50 instance
s each, where each class refers to a\ntype of iris plant.  One class
is linearly separable from the other 2; the\nlatter are NOT linearly
separable from each other.\n\nReferences\n----------\n   - Fisher,R.
A. "The use of multiple measurements in taxonomic problems"\n     An
nual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
\n     Mathematical Statistics" (John Wiley, NY, 1950).\n   - Duda,
R.O., & Hart,P.E. (1973) Pattern Classification and Scene Analysi
s.\n     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See pag
e 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 Patt
ern Analysis and Machine\n     Intelligence, Vol. PAMI-2, No. 1, 67-
71.\n   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  I
EEE Transactions\n     on Information Theory, May 1972, 431-433.\n
- See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLAS
S II\n     conceptual clustering system finds 3 classes in the dat
a.\n   - Many, many more ...\n',
 '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]
```

```
       [ 5.7,  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,  0.1],
       [ 5. ,  3.2,  1.2,  0.2],
       [ 5.5,  3.5,  1.3,  0.2],
       [ 4.9,  3.1,  1.5,  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]]),
 'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
 '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, 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, 2, 2, 2]),
 'target_names': array(['setosa', 'versicolor', 'virginica'],
       dtype='<U10')}
```

Merging the dataset into dataframes

In [30]:

```
iris_dataf = pd.DataFrame(data= np.c_[iris['data'], iris['target']], columns= iris['fea
ture_names'] + ['target'])
```

First 5 rows of data from the Dataframes

In [31]:

```
iris_dataf.head()
```

Out[31]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

Last 5 rows of data from the Dataframes

In [32]:

```
iris_dataf.tail()
```

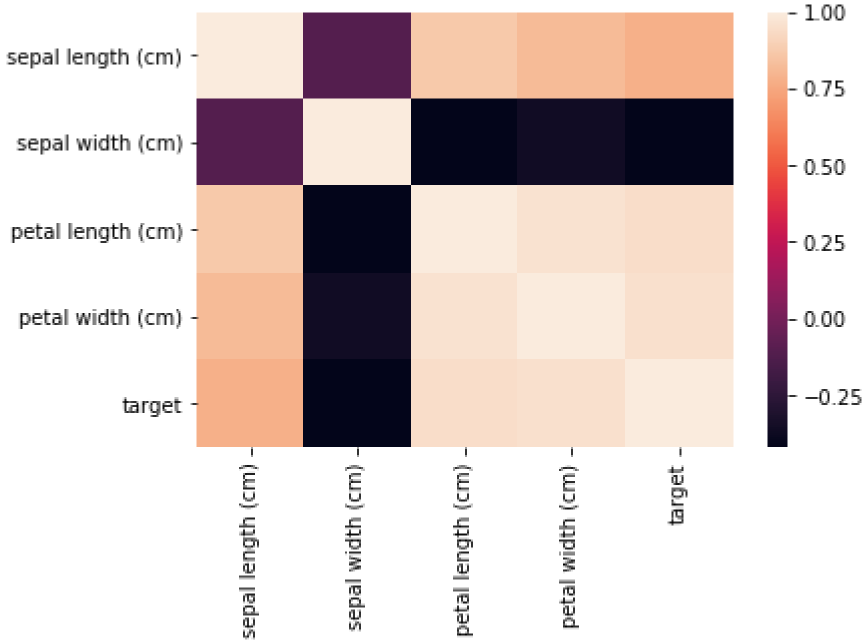
Out[32]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
145	6.7	3.0	5.2	2.3	2.0
146	6.3	2.5	5.0	1.9	2.0
147	6.5	3.0	5.2	2.0	2.0
148	6.2	3.4	5.4	2.3	2.0
149	5.9	3.0	5.1	1.8	2.0

Heatmap for Iris Dataset

In [33]:

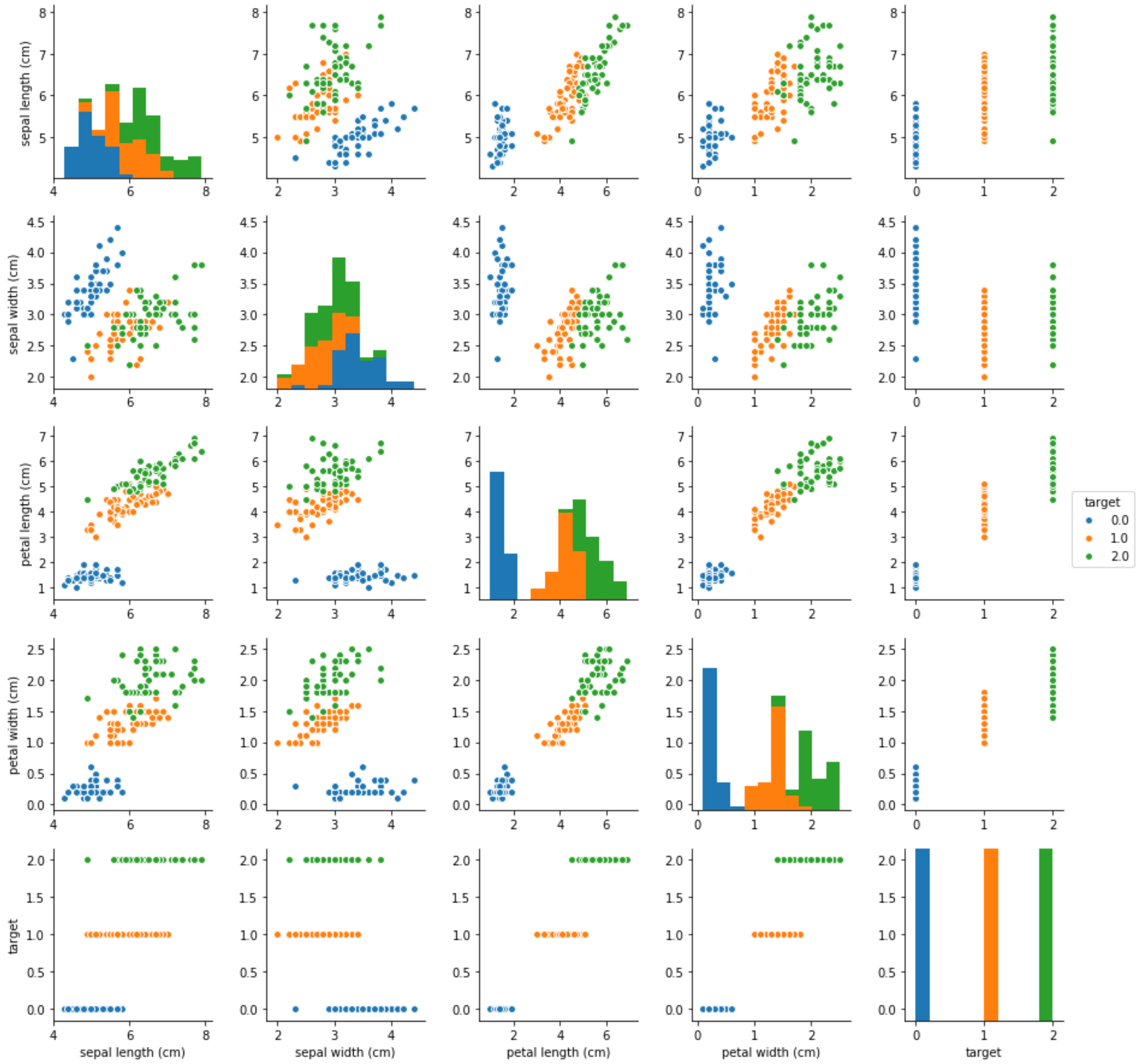
```
corr= iris_dataf.corr()
sns.heatmap(corr)
plt.show()
```



Plot Classfication of Iris V/S Independent variables

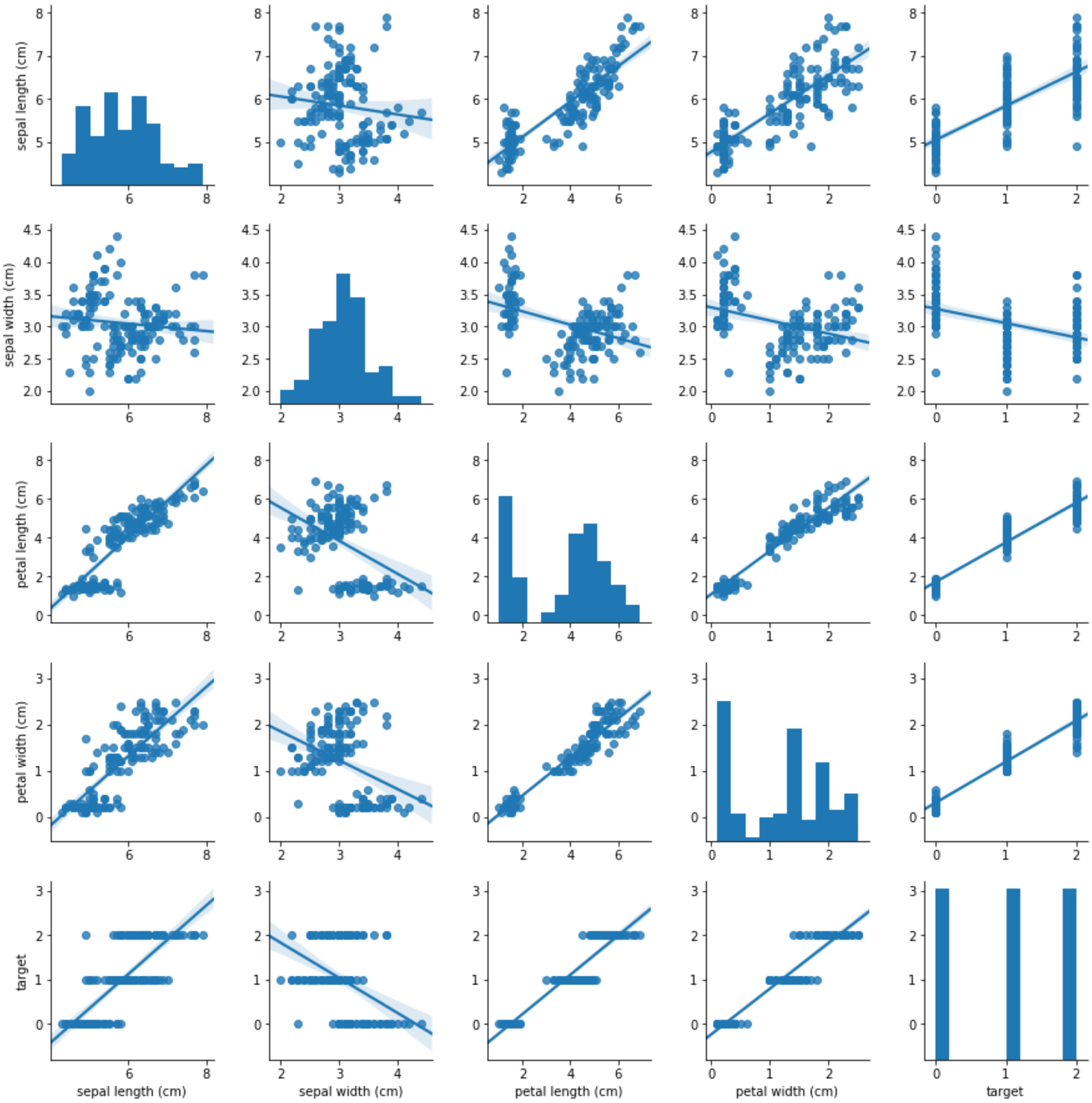
In [34]:

```
sns.pairplot(iris_dataf.dropna(),hue="target")
plt.show()
```



In [35]:

```
sns.pairplot(iris_dataf.dropna(),kind="reg")
plt.show()
```



K Nearest Neighbour's and Cross-Vaidation

In [36]:

```
features, target = datasets.load_iris(return_X_y=True)
```

Differenciating the data into training and testing dataset by using a 29% test size

In [37]:

```
X_train, X_test, y_train, y_test = model_selection.train_test_split(features,
target,test_size=0.29)
```

Using KNN Classifier on the dataset

In [38]:

```
knn = KNeighborsClassifier(n_neighbors=4)
```

Defining a KNN model using the previously defined training data

In [39]:

```
knn.fit(X_train, y_train)
```

Out[39]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=4, p=2,
           weights='uniform')
```

Checking the precision of the model utilizing the testing data

In [40]:

```
pred = knn.predict(X_test)
```

In [41]:

```
pred
```

Out[41]:

```
array([1, 1, 0, 0, 2, 0, 0, 1, 0, 1, 2, 0, 2, 1, 0, 2, 2, 1, 1, 1, 2, 0,
2,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 2, 0, 0, 2, 2, 1, 2, 2, 2, 2])
```

In [42]:

```
y_test
```

Out[42]:

```
array([1, 1, 0, 0, 2, 0, 0, 1, 0, 1, 2, 0, 2, 2, 0, 2, 2, 1, 1, 1, 2, 0,
2,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 2, 1, 2, 0, 0, 2, 2, 1, 2, 2, 2, 2])
```

In [48]:

```
metrics.accuracy_score(y_test, pred)
```

Out[48]:

```
0.954545454545459
```

In [49]:

```
print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
```

```
Test set score: 0.95
```

Check the accuracy of K values from 1-25

In [50]:

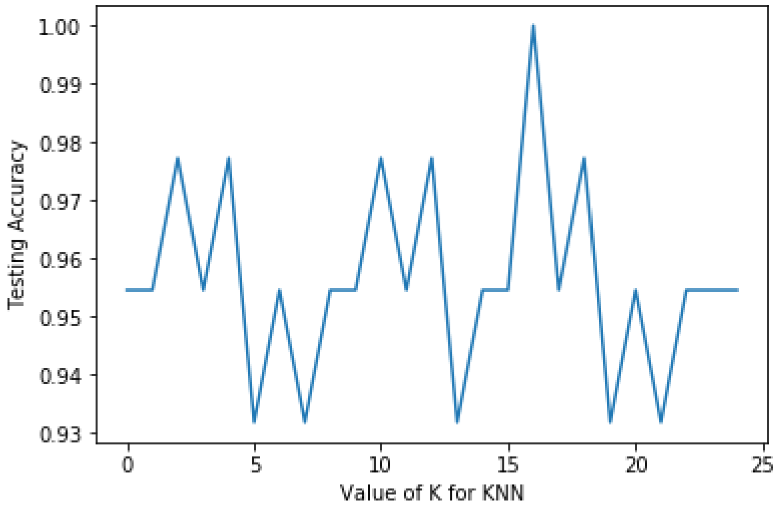
```
k_range = range(0,25)

scores = []

for K in k_range:
 K_value = K+1
 neigh = neighbors.KNeighborsClassifier(n_neighbors = K_value, weights ='uniform', algo
rithm='auto')
 neigh.fit(X_train, y_train)
 y_pred = neigh.predict(X_test)
 print ("Accuracy is",metrics.accuracy_score(y_test,y_pred)*100, "% for K-Value:", K_va
lue)
 scores.append(metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy is 95.4545454545 % for K-Value: 1
Accuracy is 95.4545454545 % for K-Value: 2
Accuracy is 97.7272727273 % for K-Value: 3
Accuracy is 95.4545454545 % for K-Value: 4
Accuracy is 97.7272727273 % for K-Value: 5
Accuracy is 93.1818181818 % for K-Value: 6
Accuracy is 95.4545454545 % for K-Value: 7
Accuracy is 93.1818181818 % for K-Value: 8
Accuracy is 95.4545454545 % for K-Value: 9
Accuracy is 95.4545454545 % for K-Value: 10
Accuracy is 97.7272727273 % for K-Value: 11
Accuracy is 95.4545454545 % for K-Value: 12
Accuracy is 97.7272727273 % for K-Value: 13
Accuracy is 93.1818181818 % for K-Value: 14
Accuracy is 95.4545454545 % for K-Value: 15
Accuracy is 95.4545454545 % for K-Value: 16
Accuracy is 100.0 % for K-Value: 17
Accuracy is 95.4545454545 % for K-Value: 18
Accuracy is 97.7272727273 % for K-Value: 19
Accuracy is 93.1818181818 % for K-Value: 20
Accuracy is 95.4545454545 % for K-Value: 21
Accuracy is 93.1818181818 % for K-Value: 22
Accuracy is 95.4545454545 % for K-Value: 23
Accuracy is 95.4545454545 % for K-Value: 24
Accuracy is 95.4545454545 % for K-Value: 25
```

To plot the relation between precision of testing and K

In [51]:

```
plt.plot(k_range, scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Testing Accuracy')
plt.show()
```



KNN performs best for k=17 as the accuracy is 100%