

```
## shree ganeshaya namaha
```

✓ Linear Regression

Refernce:

1. <https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/>
2. <https://spotintelligence.com/2024/03/27/regression-metrics-for-machine-learning/>
3. <https://www.geeksforgeeks.org/ml-linear-regression/>
4. <https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/simple-linear-regression.html>- Numericals

```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
print(diabetes_X.shape)
print(diabetes_y.shape)

# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 2]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)

# The coefficients
print("Coefficients: \n", regr.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test, diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(diabetes_y_test, diabetes_y_pred))

# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test, color="black")
plt.plot(diabetes_X_test, diabetes_y_pred, color="blue", linewidth=3)

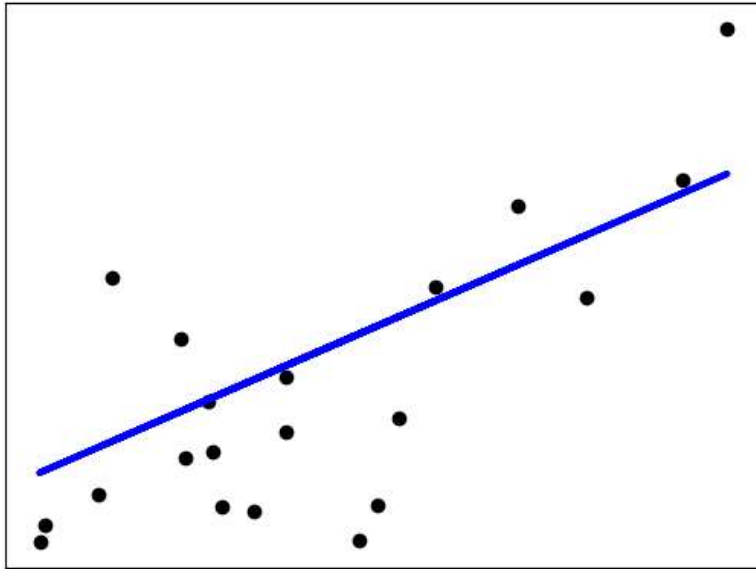
plt.xticks(())
plt.yticks(())

plt.show()
```

```

(442, 10)
(442,)
Coefficients:
 [938.23786125]
Mean squared error: 2548.07
Coefficient of determination: 0.47

```



```
print(diabetes_X[:5,:], diabetes_y[:5])
```

```

[[ 0.06169621]
 [-0.05147406]
 [ 0.04445121]
 [-0.01159501]
 [-0.03638469]] [151.  75. 141. 206. 135.]

```

✓ Logistic Regression

Refrence

1. <https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/>
2. <https://www.analyticsvidhya.com/blog/2021/05/20-questions-to-test-your-skills-on-logistic-regression/>
3. <https://www.geeksforgeeks.org/understanding-logistic-regression/>
4. <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/>
5. <https://www.analyticsvidhya.com/blog/2021/06/artificial-neural-networks-better-understanding/>

```
from sklearn import datasets
```

```

iris = datasets.load_iris() ## load the dataset
print(len(iris))
X=iris.data
y=iris.target
labels=iris.target_names
print(X.shape)
print(y.shape)
print(labels.shape)
print(labels)
#print(iris.DESCR) - Gives

```

```

(150, 4)
(150,)
(3,)
['setosa' 'versicolor' 'virginica']

```

```
print(iris.DESCR)
```



****Data Set Characteristics:****

```
:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive attributes and the class
:Attribute Information:
  - sepal length in cm
  - sepal width in cm
  - petal length in cm
  - petal width in cm
  - class:
    - Iris-Setosa
    - Iris-Versicolour
    - Iris-Virginica
```

:Summary Statistics:

```
=====
      Min  Max   Mean   SD   Class Correlation
=====
sepal length:  4.3  7.9   5.84   0.83    0.7826
sepal width:   2.0  4.4   3.05   0.43   -0.4194
petal length:  1.0  6.9   3.76   1.76    0.9490 (high!)
petal width:   0.1  2.5   1.20   0.76    0.9565 (high!)
=====
```

```
:Missing Attribute Values: None
:Class Distribution: 33.3% for each of 3 classes.
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988
```

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken from Fisher's paper. Note that it's the same as in R, but not as in the UCI Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

.. topic:: References

- Fisher, R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to Mathematical Statistics" (John Wiley, NY, 1950).
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis. (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.
- Dasarthy, B.V. (1980) "Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions on Information Theory, May 1972, 431-433.
- See also: 1988 MLC Proceedings, 54-64. Cheeseman et al's AUTOCLASS II conceptual clustering system finds 3 classes in the data.
- Many, many more ...

```
X1[:20,:]
```



```
array([[5.1, 3.5],
       [4.9, 3. ],
       [4.7, 3.2],
       [4.6, 3.1],
       [5. , 3.6],
       [5.4, 3.9],
       [4.6, 3.4],
       [5. , 3.4],
       [4.4, 2.9],
       [4.9, 3.1],
       [5.4, 3.7],
       [4.8, 3.4],
       [4.8, 3. ],
       [4.3, 3. ],
       [5.8, 4. ],
       [5.7, 4.4],
       [5.4, 3.9],
```

```
[5.1, 3.5],
[5.7, 3.8],
[5.1, 3.8]])
```

y

```
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, 2, 2])
```

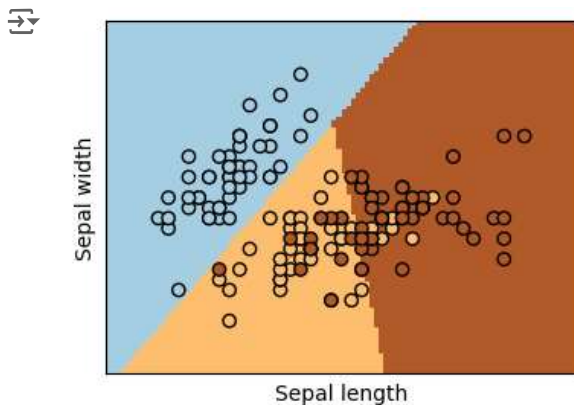
```
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.linear_model import LogisticRegression
# Create an instance of Logistic Regression Classifier and fit the data.
X1=X[:, :2]
logreg = LogisticRegression(C=1e5)
logreg.fit(X1, y)
```

```
_, ax = plt.subplots(figsize=(4, 3))
DecisionBoundaryDisplay.from_estimator(
    logreg,
    X1,
    cmap=plt.cm.Paired,
    ax=ax,
    response_method="predict",
    plot_method="pcolormesh",
    shading="auto",
    xlabel="Sepal length",
    ylabel="Sepal width",
    eps=0.5,
)
```

```
# Plot also the training points
plt.scatter(X1[:, 0], X1[:, 1], c=y, edgecolors="k", cmap=plt.cm.Paired)
```

```
plt.xticks(())
plt.yticks(())
```

```
plt.show()
```



Practice Question

1. download Mnist fashion dataset, and then create a dataset of using any two fashion accessories and then classify the same. try to optimise the same using any Mean square error or R2 error.
2. Use digit dataset to classify the digits 8 and 3 or 6 and 9.

✓ Principal component analysis

Reference:

1. <https://builtin.com/data-science/step-step-explanation-principal-component-analysis>
2. <https://www.analyticsvidhya.com/blog/2022/07/principal-component-analysis-beginner-friendly/>
3. <https://medium.com/analytics-vidhya/understanding-principle-component-analysis-pca-step-by-step-e7a4bb4031d9>
4. <https://www.analyticsvidhya.com/blog/2021/04/principal-component-analysis-introduction-and-practice-problem/>
5. <https://www.analyticsvidhya.com/blog/2016/03/pca-practical-guide-principal-component-analysis-python/>

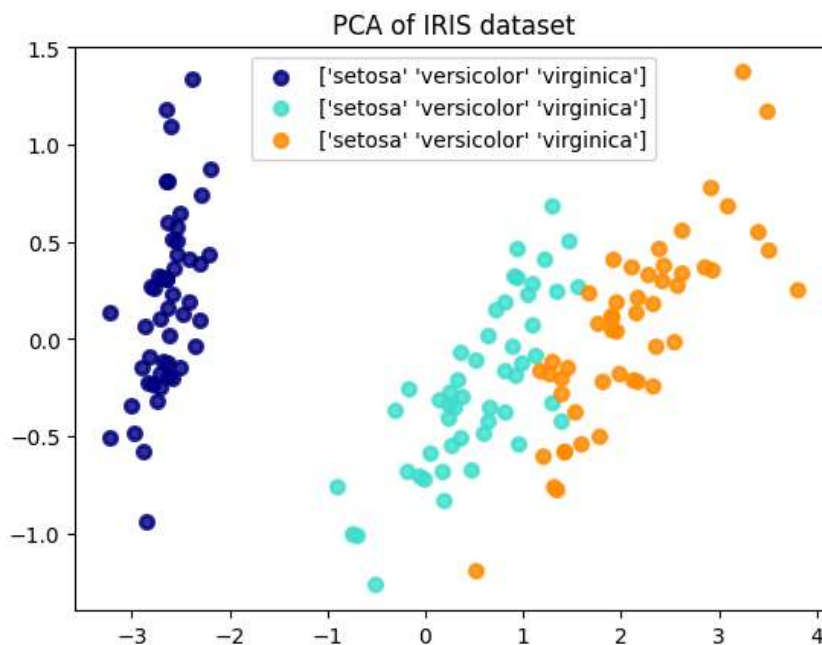
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit(X).transform(X)
print("PCA transformed data", X_pca.shape)
# Percentage of variance explained for each components
print(
    "explained variance ratio (first two components): %s"
    % str(pca.explained_variance_ratio_)
)

plt.figure()
colors = ["navy", "turquoise", "darkorange"]
lw = 2

for color, i, target_name in zip(colors, [0, 1, 2], labels):
    plt.scatter(
        X_pca[y == i, 0], X_pca[y == i, 1], color=color, alpha=0.8, lw=lw, label=labels
    )
plt.legend(loc="best", shadow=False, scatterpoints=1)
plt.title("PCA of IRIS dataset")

plt.show()
```

PCA transformed data (150, 2)
explained variance ratio (first two components): [0.92461872 0.05306648]



Practice question

1. For Mnist Fashion dataset reduce the dimesionaity to say n, where n is accepted from user and try to classify the same using differnt classifiers like Decision trees, SVM, KNN etc

✓ Navie Bayes Classifier

Reference


1. <https://www.geeksforgeeks.org/naive-bayes-classifiers/>
2. <https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>
3. <https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.html>
4. <https://medium.com/@shuv.sdr/na%C3%AFve-bayes-classification-in-python-f869c2e0dbf1>
5. <https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/> - start from scratch
6. <https://www.analyticsvidhya.com/blog/2021/01/a-guide-to-the-naive-bayes-algorithm/>
7. <https://www.geeksforgeeks.org/bayes-theorem/>

```
# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# making predictions on the testing set
y_pred = gnb.predict(X_test)

# comparing actual response values (y_test) with predicted response values (y_pred)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*100)
```

 Gaussian Naive Bayes model accuracy(in %): 95.0

▼ Decision Tree

Reference:


1. <https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/>
2. <https://www.geeksforgeeks.org/decision-tree-introduction-example/>
3. https://www.w3schools.com/python/python_ml_decision_tree.asp
4. <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the input directory

from subprocess import check_output

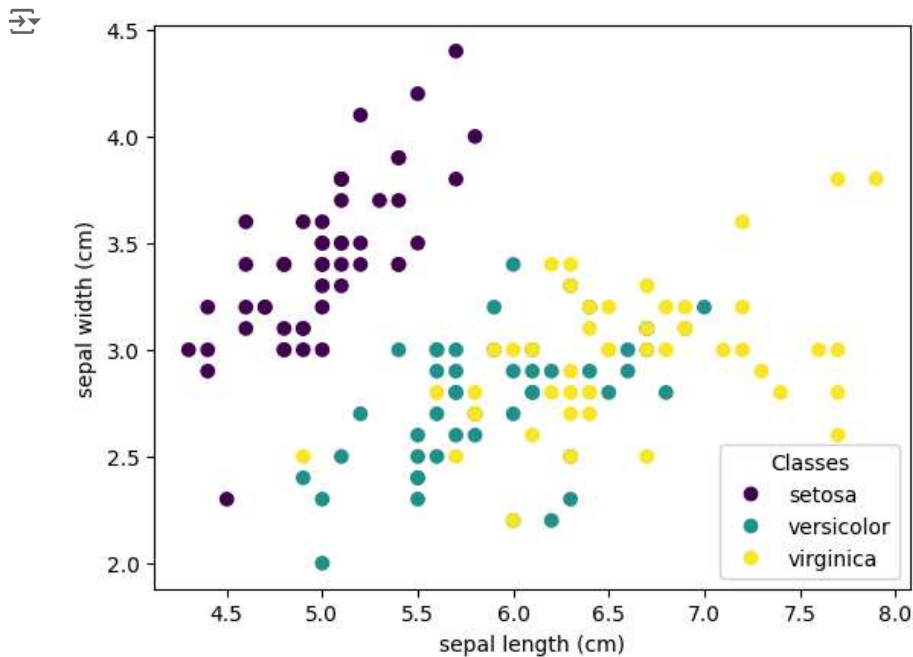
from sklearn import datasets

iris = datasets.load_iris() ## load the dataset
print(len(iris))
X=iris.data
y=iris.target
labels=iris.target_names
print(X.shape)
print(y.shape)
print(labels.shape)
print(labels)
#print(iris.DESCR) - Gives
```

 8
(150, 4)
(150,)
(3,)

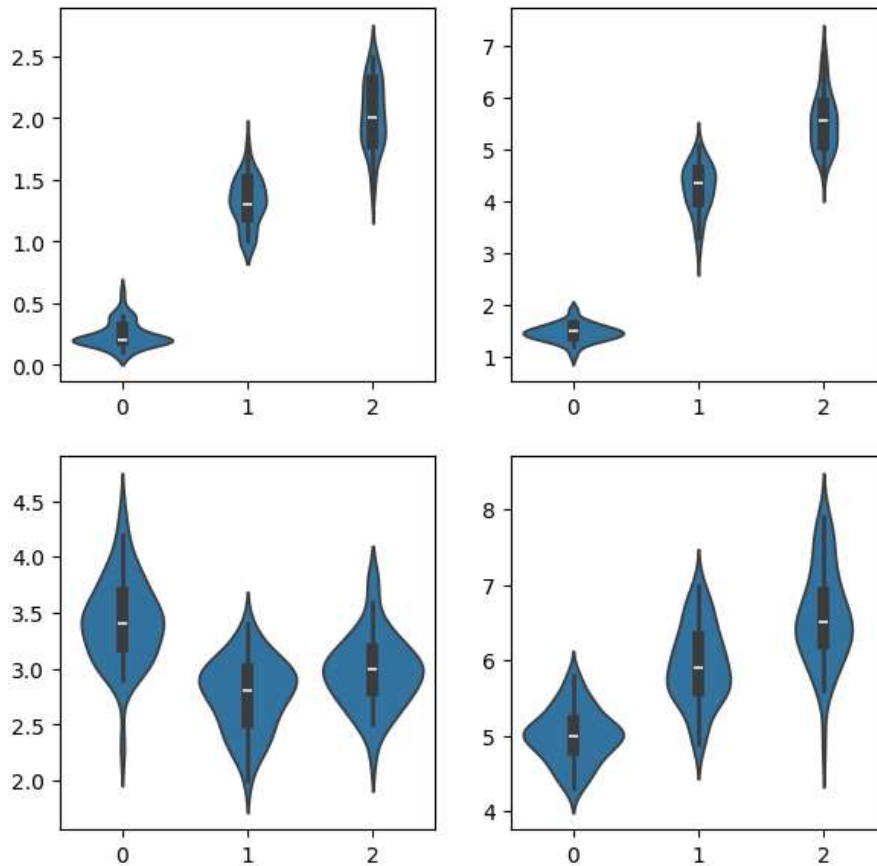
```
['setosa' 'versicolor' 'virginica']
```

```
_, ax = plt.subplots()
scatter = ax.scatter(iris.data[:, 0], iris.data[:, 1], c=iris.target)
ax.set(xlabel=iris.feature_names[0], ylabel=iris.feature_names[1])
_ = ax.legend(
    scatter.legend_elements()[0], iris.target_names, loc="lower right", title="Classes"
)
```



```
plt.figure(figsize=(7,7))
plt.subplot(2,2,1)
sns.violinplot(x=y,y=X[:,3],data=iris)# petal Length
#plt.xlabel('Spices')
plt.subplot(2,2,2)
sns.violinplot(x=y,y=X[:,2],data=iris)# petal Width
plt.subplot(2,2,3)
sns.violinplot(x=y,y=X[:,1],data=iris)# sepal Length
plt.subplot(2,2,4)
sns.violinplot(x=y,y=X[:,0],data=iris)# sepal Width
plt.ylabel
```

→ 'PetalLengthCm'



```
from sklearn import metrics #for checking the model accuracy
from sklearn.tree import DecisionTreeClassifier #for using Decision Tree Algorithm
from sklearn.naive_bayes import CategoricalNB
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.tree import export_text
import graphviz
from sklearn.tree import export_graphviz

#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)# in this our main data is sp.
# the attribute test_size=0.3 splits the data into 70% and 30% ratio. train=70% and test=30%
print(X_train.shape)
print(X_test.shape)
model_dt_al_g=DecisionTreeClassifier()
model_dt_al_g.fit(X_train,y_train)
predict_dt_al_g=model_dt_al_g.predict(X_test)
print('The accuracy of the Decision Tree is',metrics.accuracy_score(predict_dt_al_g,y_test))
con_dt_al_g=confusion_matrix(y_test, predict_dt_al_g)
print("confusion matrix \n",con_dt_al_g, "\n")
cls_dt_al_g=classification_report(y_test, predict_dt_al_g, target_names=labels)
print("classification report \n",cls_dt_al_g)
r = export_text(model_dt_al_g, feature_names=['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm'])
print(r)
```

→ (105, 4)
(45, 4)
The accuracy of the Decision Tree is 0.9555555555555556
confusion matrix
[[14 0 0]
[0 17 1]
[0 1 12]]

classification report

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	14
versicolor	0.94	0.94	0.94	18
virginica	0.92	0.92	0.92	13

accuracy			0.96	45
macro avg	0.96	0.96	0.96	45
weighted avg	0.96	0.96	0.96	45

```

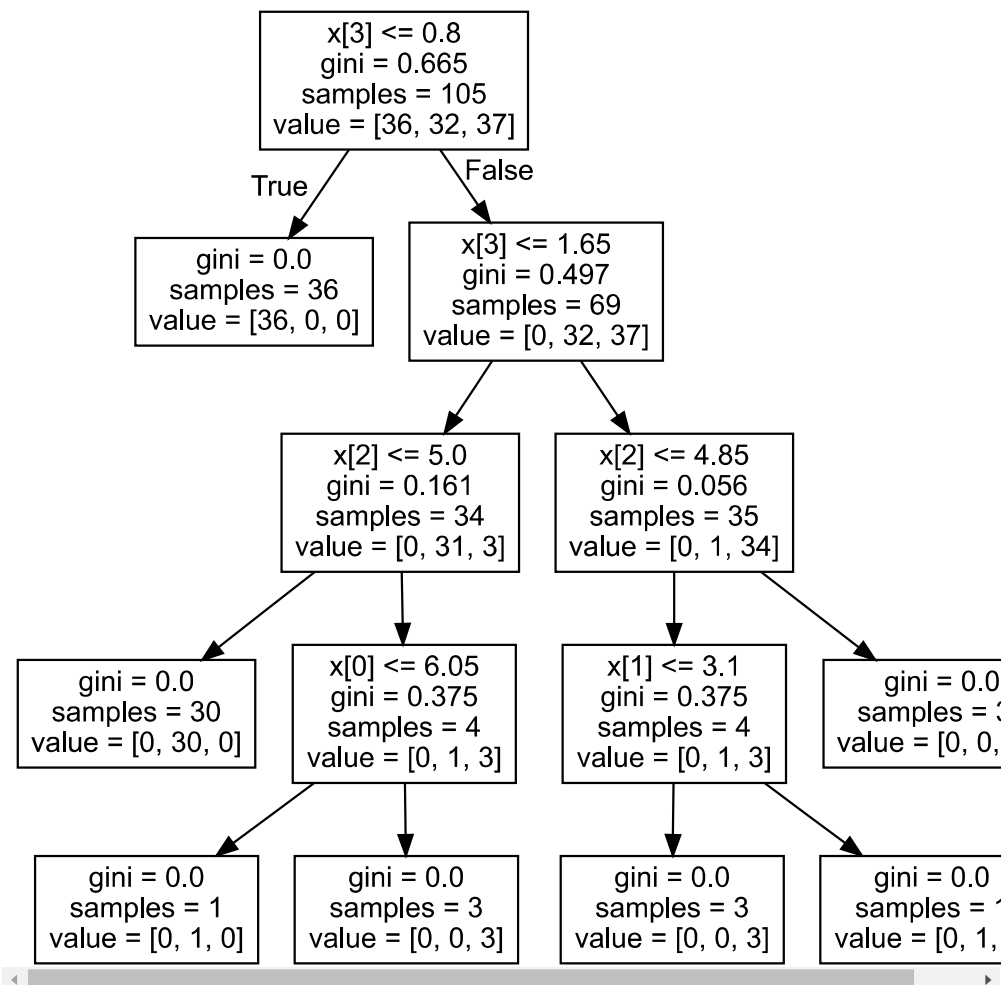
|--- PetalWidthCm <= 0.80
|   |--- class: 0
|--- PetalWidthCm > 0.80
|   |--- PetalWidthCm <= 1.65
|       |--- PetalLengthCm <= 5.00
|           |--- class: 1
|       |--- PetalLengthCm > 5.00
|           |--- SepalLengthCm <= 6.05
|               |--- class: 1
|               |--- SepalLengthCm > 6.05
|                   |--- class: 2
|       |--- PetalWidthCm > 1.65
|           |--- PetalLengthCm <= 4.85
|               |--- SepalWidthCm <= 3.10
|                   |--- class: 2
|               |--- SepalWidthCm > 3.10
|                   |--- class: 1
|           |--- PetalLengthCm > 4.85
|               |--- class: 2

```

```

dot_data = export_graphviz(model_dt_al_g, out_file=None)
graph = graphviz.Source(dot_data)
graph

```



✓ K-Nearest Neighbour

Reference:

```
from sklearn.neighbors import KNeighborsClassifier # for K nearest neighbours
```

```
model_knn_al=KNeighborsClassifier(n_neighbors=3) #this examines 3 neighbours for putting the new data into a class
model_knn_al.fit(X_train,y_train)
predict_knn_al=model_knn_al.predict(X_test)
print('The accuracy of the KNN is',metrics.accuracy_score(predict_knn_al,y_test))
con_knn_al=confusion_matrix(y_test, predict_knn_al)
print("confusion matrix\n",con_knn_al, "\n")
cls_knn_al=classification_report(y_test, predict_knn_al, target_names=labels)
print("classification report \n",cls_knn_al)
```



The accuracy of the KNN is 0.9777777777777777

confusion matrix

```
[[14  0  0]
 [ 0 18  0]
 [ 0  1 12]]
```

classification report

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	14
versicolor	0.95	1.00	0.97	18
virginica	1.00	0.92	0.96	13
accuracy			0.98	45
macro avg	0.98	0.97	0.98	45
weighted avg	0.98	0.98	0.98	45

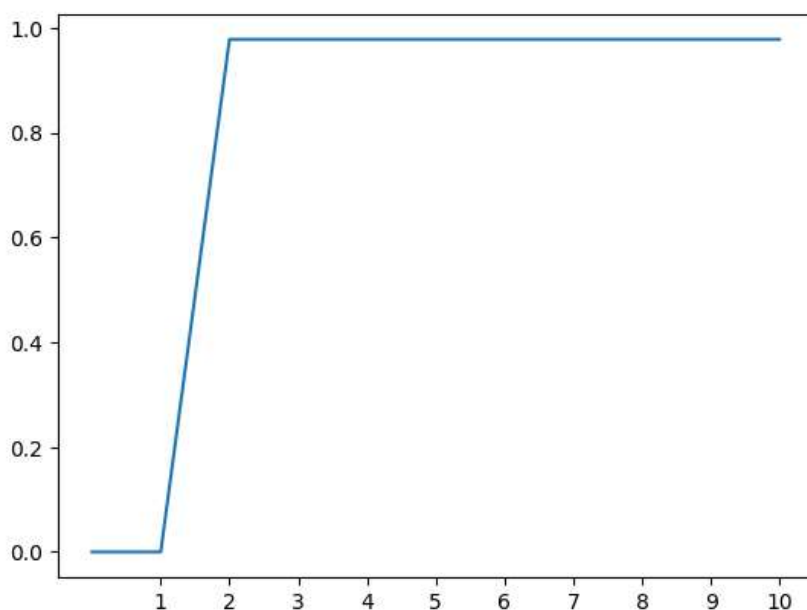
compare KNN for various neighbours

```
a_index=list(range(1,11))
Series=np.zeros(11)
#x=[1,2,3,4,5,6,7,8,9,10]
for i in list(range(2,11)):
    model=KNeighborsClassifier(n_neighbors=i)
    model.fit(X_train,y_train)
    prediction=model.predict(X_test)
    Series[i]=metrics.accuracy_score(prediction,y_test)
    print(Series[i])
plt.plot(range(11), Series)
plt.xticks(x)
```

```

0.9777777777777777
0.9777777777777777
0.9777777777777777
0.9777777777777777
0.9777777777777777
0.9777777777777777
0.9777777777777777
0.9777777777777777
0.9777777777777777
0.9777777777777777
([<matplotlib.axis.XTick at 0x7c45068461d0>,
 <matplotlib.axis.XTick at 0x7c4506845870>,
 <matplotlib.axis.XTick at 0x7c4506844c10>,
 <matplotlib.axis.XTick at 0x7c44c6f97460>,
 <matplotlib.axis.XTick at 0x7c44c6f95630>,
 <matplotlib.axis.XTick at 0x7c44c6f977c0>,
 <matplotlib.axis.XTick at 0x7c44c8b41180>,
 <matplotlib.axis.XTick at 0x7c44c8b41cc0>,
 <matplotlib.axis.XTick at 0x7c44c8b420b0>,
 <matplotlib.axis.XTick at 0x7c44c8b40af0>],
 [Text(1, 0, '1'),
 Text(2, 0, '2'),
 Text(3, 0, '3'),
 Text(4, 0, '4'),
 Text(5, 0, '5'),
 Text(6, 0, '6'),
 Text(7, 0, '7'),
 Text(8, 0, '8'),
 Text(9, 0, '9'),
 Text(10, 0, '10')])

```



✓ Support Vector Machine

```

from sklearn import datasets, svm
from sklearn.inspection import DecisionBoundaryDisplay

C = 1.0 # SVM regularization parameter
models = (
    svm.SVC(kernel="linear", C=C),
    svm.LinearSVC(C=C, max_iter=10000, # dual="auto"),
    svm.SVC(kernel="rbf", gamma=0.7, C=C),
    svm.SVC(kernel="poly", degree=3, gamma="auto", C=C),
)
models = (clf.fit(X_pca, y) for clf in models)

# title for the plots
titles = (
    "SVC with linear kernel",
    "LinearSVC (linear kernel)",
    "SVC with RBF kernel",
    "SVC with polynomial (degree 3) kernel",
)

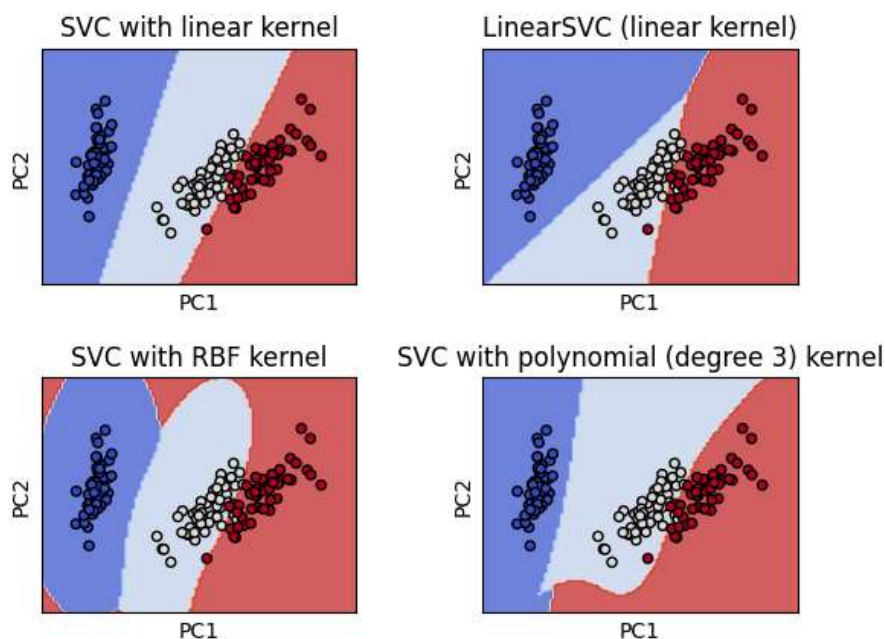
# Set-up 2x2 grid for plotting.
fig, sub = plt.subplots(2, 2)
plt.subplots_adjust(wspace=0.4, hspace=0.4)

#X0, X1 = X[:, 0], X[:, 1]

for clf, title, ax in zip(models, titles, sub.flatten()):
    disp = DecisionBoundaryDisplay.from_estimator(
        clf,
        X_pca,
        response_method="predict",
        cmap=plt.cm.coolwarm,
        alpha=0.8,
        ax=ax,
        xlabel='PC1',
        ylabel='PC2',
    )
    ax.scatter(X_pca[:,0], X_pca[:,1], c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(title)

plt.show()

```



✓ Artificial Neural Network

Reference:

1. <https://www.geeksforgeeks.org/implementing-ann-training-process-in-python/>- toy example
2. <https://k21academy.com/datascience-blog/deep-learning/artificial-neural-network/>
3. <https://medium.com/@Coursesteach/guide-to-supervised-learning-with-scikit-learn-part-3-c31b01c547f9>
4. <https://www.springboard.com/blog/data-science/beginners-guide-neural-network-in-python-scikit-learn-0-18/>
5. <https://towardsdatascience.com/the-simplest-way-to-train-a-neural-network-in-python-17613fa97958>
6. <https://www.analyticsvidhya.com/blog/2021/10/implementing-artificial-neural-networkclassification-in-python-from-scratch/>
7. <https://www.mltut.com/implementation-of-artificial-neural-network-in-python/>

```
import sklearn.neural_network
# Create an instance of the MLPClassifier class
neural_network = sklearn.neural_network.MLPClassifier(hidden_layer_sizes=(10, 20), activation='relu', random_state=1)
# Fit the model to the training data
neural_network.fit(X_train, y_train)
# Predict the labels of new data
y_pred = neural_network.predict(X_test)
print('The accuracy of the ANN is', metrics.accuracy_score(y_pred, y_test))
con_ann = confusion_matrix(y_test, y_pred)
print("confusion matrix\n", con_ann, "\n")
cls_ann = classification_report(y_test, y_pred, target_names=labels)
print("classification report \n", cls_ann)
```



The accuracy of the ANN is 0.7833333333333333

confusion matrix

```
[[19  0  0]
 [ 0  8 13]
 [ 0  0 20]]
```

classification report

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	19
versicolor	1.00	0.38	0.55	21
virginica	0.61	1.00	0.75	20