```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Social_Network_Ads.csv to Social_Network_Ads.csv
User uploaded file "Social_Network_Ads.csv" with length 10926 bytes
```

```
# Import necessary modules
#sci-kit library
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris #cleaned


# Loading data
irisData = load_iris()
print(irisData.DESCR)
print(irisData.feature_names) #input
print(irisData.target_names) #output
# Create feature and target arrays
X = irisData.data # 4 attributes petal length, width, sepal len, width
Y = irisData.target # class 0,1,2
print(X)
print(Y)
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**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.
          - Dasarathy, 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.
```

```
# Split into training and test set
knn = KNeighborsClassifier(n_neighbors=5)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=30)
knn.fit(X_train, Y_train) #training
# Predict on dataset which model has not seen before
Y_pred=knn.predict(X_test)
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
acc= metrics.accuracy_score(Y_test, Y_pred)
print("Accuracy:",acc)
print( metrics.classification_report(Y_test, Y_pred) )
metrics.confusion_matrix(Y_test, Y_pred)
```

```
Accuracy: 0.9333333333333333
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      0.78      0.88         9
           2       0.82      1.00      0.90         9

    accuracy                           0.93        30
   macro avg       0.94      0.93      0.92        30
weighted avg       0.95      0.93      0.93        30

array([[12,  0,  0],
       [ 0,  7,  2],
       [ 0,  0,  9]])
```

```
# Split into training and test set
testing_accuracy = []
n_neighbors=range(1,11)
for i in n_neighbors:
    knn = KNeighborsClassifier(n_neighbors=i)
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=42)
    knn.fit(X_train, Y_train) #training
    # Predict on dataset which model has not seen before
    Y_pred=knn.predict(X_test)
    #Import scikit-learn metrics module for accuracy calculation
    from sklearn import metrics
    # Model Accuracy, how often is the classifier correct?
    acc= metrics.accuracy_score(Y_test, Y_pred)
    testing_accuracy.append(acc)
    print("i=",i , "Accuracy:",acc)
    #print( metrics.classification_report( y_test, Y_pred) )
    #metrics.confusion_matrix(Y_test, Y_pred,labels=[0, 1, 2])
import matplotlib.pyplot as plt
plt.plot(n_neighbors,testing_accuracy)
```

```
i= 1 Accuracy: 1.0
i= 2 Accuracy: 1.0
i= 3 Accuracy: 1.0
i= 4 Accuracv: 1.0
```

```
# Import necessary modules
#sci-kit library
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
# Loading data
cancerData = load_breast_cancer()
print(cancerData.feature_names)
print(cancerData.target_names)
# Create feature and target arrays
X = cancerData.data
Y = cancerData.target
print(X)
print(Y)
```

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
['malignant' 'benign']
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 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 0 0 0 0 0 0 1]
```

```
# Split into training and test set
testing_accuracy = []
n_neighbors=range(1,11)
for i in n_neighbors:
   knn = KNeighborsClassifier(n_neighbors=i)
   X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)
   knn.fit(X_train, Y_train) #training
   # Predict on dataset which model has not seen before
   Y_pred=knn.predict(X_test)
   #Import scikit-learn metrics module for accuracy calculation
   from sklearn import metrics
   # Model Accuracy, how often is the classifier correct?
   acc= metrics.accuracy_score(Y_test, Y_pred)
   testing_accuracy.append(metrics.accuracy_score(Y_test, Y_pred))
   print("i=",i , "Accuracy:",acc)
   #print( metrics.classification_report( y_test, Y_pred) )
   #metrics.confusion_matrix(Y_test, Y_pred,labels=[0, 1, 2])
import matplotlib.pyplot as plt
plt.plot(n_neighbors,testing_accuracy)
```

```
i= 1 Accuracy: 0.9298245614035088
i= 2 Accuracy: 0.956140350877193
i= 3 Accuracy: 0.9649122807017544
i= 4 Accuracy: 0.9298245614035088
i= 5 Accuracy: 0.9473684210526315
i= 6 Accuracy: 0.9035087719298246
i= 7 Accuracy: 0.9298245614035088
i= 8 Accuracy: 0.9122807017543859
i= 9 Accuracy: 0.9473684210526315
i= 10 Accuracy: 0.9298245614035088
[<matplotlib.lines.Line2D at 0x7fc59d7e8210>]
```


```
import pandas as pd
df = pd.read_csv('Social_Network_Ads.csv')
df
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

```
# Import necessary modules
#sci-kit library
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df.iloc[:,1] = le.fit_transform(df.iloc[:,1])
# Create feature and target arrays
X = df.iloc[:,0:4]
Y = df.iloc[:,4]
print(X)
print(Y)
```

```
      User ID  Gender  Age  EstimatedSalary
0    15624510       1   19            19000
1    15810944       1   35            20000
2    15668575       0   26            43000
3    15603246       0   27            57000
4    15804002       1   19            76000
..        ...     ...  ...              ...
395  15691863       0   46            41000
396  15706071       1   51            23000
397  15654296       0   50            20000
398  15755018       1   36            33000
399  15594041       0   49            36000

[400 rows x 4 columns]
0      0
1      0
2      0
3      0
4      0
      ..
```

```
395    1
396    1
397    1
398    0
399    1
Name: Purchased, Length: 400, dtype: int64
```

```
# Split into training and test set
from sklearn.preprocessing import StandardScaler
testing_accuracy = []
n_neighbors=range(1,11)
for i in n_neighbors:
  knn = KNeighborsClassifier(n_neighbors=i)
  X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)
  scaler = StandardScaler()
  scaler.fit(X_train)

  X_train = scaler.transform(X_train)
  X_test = scaler.transform(X_test)
  print(X_train)
  knn.fit(X_train, Y_train) #training
  # Predict on dataset which model has not seen before
  Y_pred=knn.predict(X_test)
  #Import scikit-learn metrics module for accuracy calculation
  from sklearn import metrics
  # Model Accuracy, how often is the classifier correct?
  acc= metrics.accuracy_score(Y_test, Y_pred)
  testing_accuracy.append(metrics.accuracy_score(Y_test, Y_pred))
  print("i=",i , "Accuracy:",acc)
  #print( metrics.classification_report( y_test, Y_pred) )
  #metrics.confusion_matrix(Y_test, Y_pred,labels=[0, 1, 2])
import matplotlib.pyplot as plt
plt.plot(n_neighbors,testing_accuracy)
```

```
[[ 1.00380933  0.98142253 -1.02172412  0.54695863]
 [-0.33374107  0.98142253 -1.20989777  0.60795419]
 [ 1.32736875 -1.01892912  0.29549144  0.30297639]
 ...
 [-0.333797    0.98142253  0.95409922 -1.13041929]
 [-0.2074658  -1.01892912 -1.11581095 -1.67937934]
 [-0.56287365 -1.01892912  1.612707    1.82786541]]
i= 1 Accuracy: 0.8375
[[-1.02273085 -1.01257911  0.79305752  0.3713813 ]
 [-0.50189982 -1.01257911 -1.43137211  0.3713813 ]
```

```
 [ 0.8661458   0.98757716 -1.91494377 -0.50166118]
 [ 0.86436443 -1.01257911 -1.04451478  0.77877312]
 [ 0.77126005  0.98757716 -0.56094312  0.89517079]]
i= 2 Accuracy: 0.825
[[ 0.59571626 -1.          0.02997906 -0.52811954]
 [ 1.44699037 -1.         -0.06595393  0.34255668]
 [-1.5620589   1.          0.22184503  0.28451159]
 ...
 [-1.1449309  -1.          2.14050474 -0.64420971]
 [-0.17582281 -1.         -1.12121677 -1.54390847]
 [ 1.67690512 -1.          0.60557697 -0.84736749]]
i= 3 Accuracy: 0.8625
[[ 0.21487933 -0.96317747 -1.11548628 -0.99026973]
 [ 0.33238432  1.03823026  0.9625829  -0.81735977]
 [ 1.0985174   1.03823026  0.39583676 -0.12571995]
 ...
 [ 1.2327474   1.03823026 -0.83211321 -0.75972312]
 [ 1.07313865 -0.96317747  0.86812521 -0.55799484]
 [-0.75994194  1.03823026  2.09607517 -0.78854145]]
i= 4 Accuracy: 0.85
[[ 1.22877335 -1.          1.42907345 -1.07550541]
 [-1.52657559  1.         -0.62077926  0.10840147]
 [ 1.24151617  1.          0.0314466   0.01960846]
 ...
 [-1.17697062 -1.         -1.08665487 -0.54274731]
 [ 1.70674062  1.          0.21779685  1.08512465]]
```