```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn import datasets
from io import StringIO
from sklearn.tree import export_graphviz
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn import metrics

# Load data file
bank=pd.read_csv('dataset.csv')
bank.head()
```

```
   SampleIdx Gender   bodyBuild_Size bodyFrame_Breadth
bodyFrame_Length  \
0          1   Male  Weaklydeveloped       Thin/Narrow
Long
1          2   Male    Welldeveloped             Broad
Long
2          3   Male  Weaklydeveloped       Thin/Narrow
Long
3          4   Male  Weaklydeveloped       Thin/Narrow
Long
4          5   Male    Welldeveloped             Broad
Long

  bodyHair_Color chest_Breadth   eye_Color             eye_Size  \
0          Black   Thin/Narrow   DarkBrown  Moderatelydeveloped
1      DarkBrown         Broad   DarkBrown  Moderatelydeveloped
2     LightBrown   Thin/Narrow  LightBrown      Weaklydeveloped
3          Dusky   Thin/Narrow   DarkBrown      Weaklydeveloped
4          Black         Broad       Black  Moderatelydeveloped

    eye_Symmetry  ...    teeth_Appearance1 teeth_Appearance2
voice_clear  \
0  Proportionate  ...  Non_Brittle/Cracked        Non_Loose
Clear
1  Proportionate  ...  Non_Brittle/Cracked        Non_Loose
Clear
2  Proportionate  ...  Non_Brittle/Cracked        Non_Loose
Non_Clear
3  Proportionate  ...  Non_Brittle/Cracked        Non_Loose
Clear
4  Proportionate  ...  Non_Brittle/Cracked        Non_Loose
Non_Clear

  skin_cracked  skin_freckle  skin_mark  skin_mole  skin_pimple
```

```
skin_wrinkled  \
0  Non_Cracked  Non_Freckles  Non_Marks  Non_Moles      Pimples
Non_Wrinkled
1  Non_Cracked  Non_Freckles      Marks  Non_Moles  Non_Pimples
Non_Wrinkled
2      Cracked  Non_Freckles  Non_Marks      Moles      Pimples
Wrinkled
3      Cracked  Non_Freckles  Non_Marks  Non_Moles  Non_Pimples
Wrinkled
4  Non_Cracked  Non_Freckles      Marks      Moles  Non_Pimples
Non_Wrinkled

   class
0   Vata
1  Kapha
2   Vata
3   Vata
4  Kapha

[5 rows x 135 columns]
```

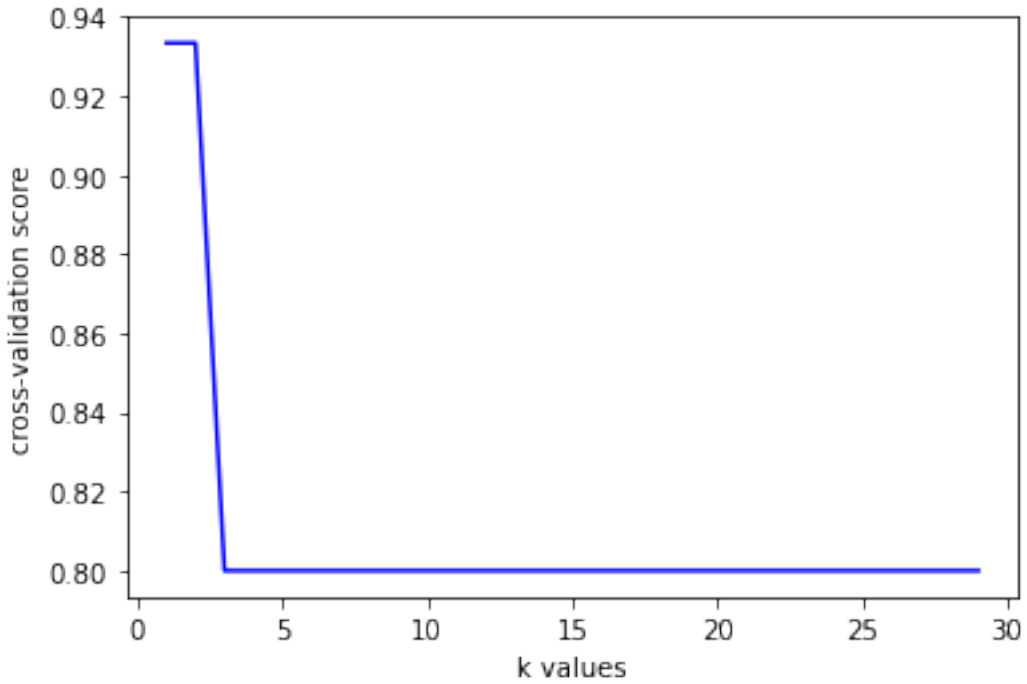
```
# Make a copy for parsing
question_data = bank.copy()
# Check if the data set contains any null values
#question_data[question_data.isnull().any(axis=1)].count()
question_data['class'] = question_data['class'].map( {'Vata':0,
'Kapha':1,'Pitta':2} )
# Convert categorical variables to dummies
question_data_original=question_data
question_data_class=question_data['class']
question_data.drop('class', axis=1, inplace=True)
question_data.drop('SampleIdx', axis=1, inplace=True)

#print(test_final.shape)
question_with_dummies = pd.get_dummies(data=question_data)
#question_with_dummies2 = pd.get_dummies(data=test_final)
question_with_dummies.head()
question_data_class.head()
```

```
0    0
1    1
2    0
3    0
4    1
Name: class, dtype: int64
```

```
X_train=question_with_dummies[:131]
y_train=question_data_class[:131]
X_test=question_with_dummies[132:]
y_test=question_data_class[132:]
```

```
#from sklearn.cross_validation import train_test_split
from sklearn import neighbors
from sklearn.model_selection import cross_val_score

k_range=list(range(1,30))
k_scores=[]
for k in k_range:
    clf = neighbors.KNeighborsClassifier(k, weights='distance')
    clf.fit(X_train, y_train)
    #scores=cross_val_score(clf, X_test, y_test, cv=2)
    scores=clf.score(X_test, y_test)
    k_scores.append(scores.mean())
    #scores=clf.score(X_test, y_test)
  # print(scores)
    #k_scores.append(scores)
print(np.round(k_scores,3)) # to display scores to 3 decimal places
from matplotlib import pyplot as plt
plt.plot(k_range,k_scores,color="blue")
plt.xlabel('k values')
plt.ylabel('cross-validation score')
fig1 = plt.gcf()
plt.draw()
plt.show()
fig1.savefig('knn1_k.png',dpi=200)
```

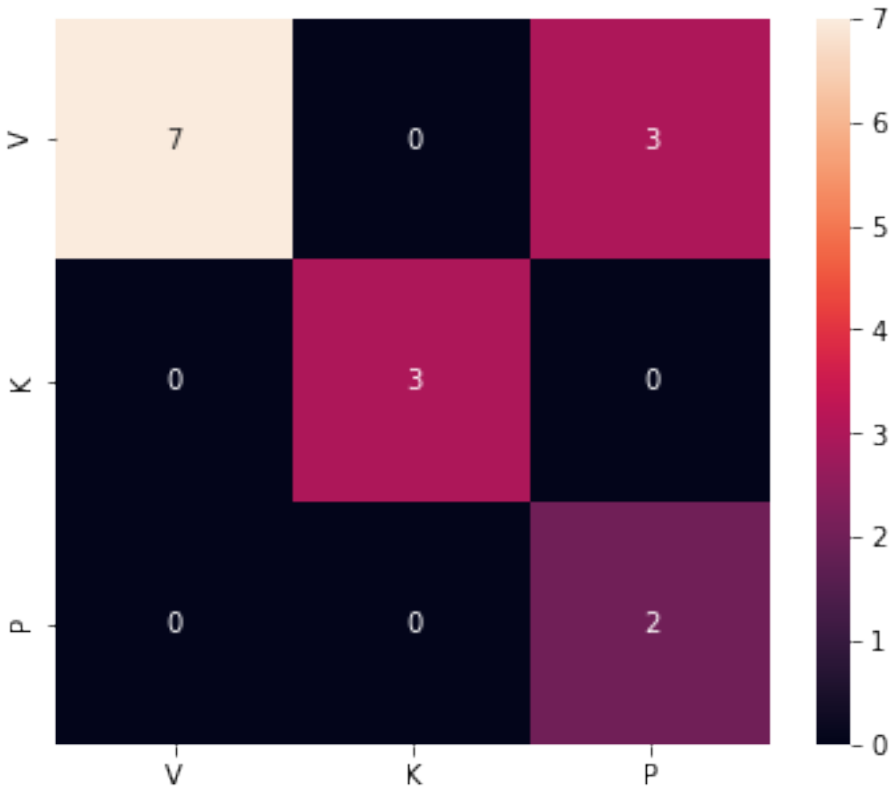
```
[0.933 0.933 0.8   0.8   0.8   0.8   0.8   0.8   0.8   0.8   0.8   0.8
 0.8   0.8   0.8   0.8   0.8   0.8   0.8   0.8   0.8   0.8   0.8   0.8
 0.8   0.8   0.8   0.8   0.8  ]
```



```
from sklearn import neighbors
n_neighbors=3
clf = neighbors.KNeighborsClassifier(n_neighbors, weights='distance')
clf.fit(X_train, y_train)

#predict
y_pred = clf.predict(X_test)
## See how the model performs on the test data.
clf.score(X_test, y_test)
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.70      0.82        10
           1       1.00      1.00      1.00         3
           2       0.40      1.00      0.57         2

    accuracy                           0.80        15
   macro avg       0.80      0.90      0.80        15
weighted avg       0.92      0.80      0.83        15
```

```
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
array = confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(array, index = [i for i in "VKP"],
                  columns = [i for i in "VKP"])
g=plt.figure(figsize = (6,5))
sn.heatmap(df_cm, annot=True)
plt.show()
fig1 = plt.gcf()
plt.draw()
plt.show()
fig1.savefig('knn1_k.png',dpi=200)
```



<Figure size 432x288 with 0 Axes>