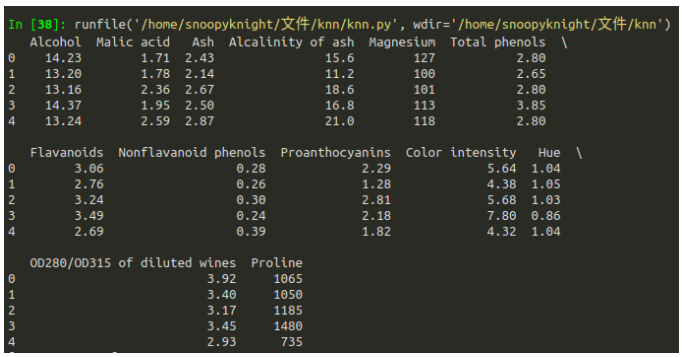
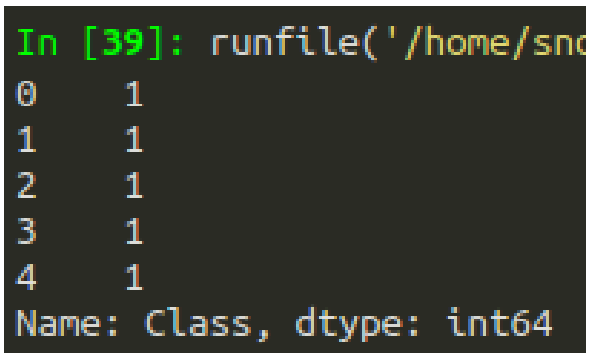
KNN-classifier implement

1. Store UCI dataset - Wine Data Set as pandas dataframe, and store every feature and label into variable X and y respectively.

```
import pandas as pd

def load_feature(filename):
    df = pd.read_csv(filename)
    feature = df.drop('Class',1)
    return feature

def load_label(filename):
    df = pd.read_csv(filename)
    label = df.Class
    return label

def main():
    X = load_feature('wine.data')
    y = load_label('wine.data')
```



print(X.head())



print(y.head())

2. Split training data and testing data.(based on sklearn.model_selection.train_test_split) The store them in X_train, X_test, y_train, y_test.

```
from sklearn.model_selection import train_test_split
def main():
    X = load_feature('wine.data')
    y = load_label('wine.data')
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
0.5)
```

3. Calculate cosine similarity of all testing data and training data, then store them in the cs_array(row:X_test_arr, col: X_train).

```
from sklearn.model_selection import train_test_split
import numpy as np

def cos_sim(X_train, X_test, y_train, y_test):
    X_train_arr = np.array(X_train.values)
    X_test_arr = np.array(X_test.values)
    cs_array = cosine_similarity(X_test_arr,X_train_arr)
    return cs_array
```

```
In [42]: runfile('/home/snoopyknight/文件/knn/knn.py', wdir='/home/snoopyknight
[[0.99974582 0.98767391 0.99962477 ... 0.99976859 0.99965528 0.9996475 ]
 [0.99885173 0.99555036 0.99892307 ... 0.9961601  0.99565948 0.99565911]
 [0.99979148 0.99307442 0.99983312 ... 0.9980913  0.99778233 0.99774827]
 ...
 [0.99878215 0.98303509 0.99856005 ... 0.99996635 0.99999747 0.9999962 ]
 [0.99991292 0.9919165  0.99991913 ... 0.99862683 0.99836576 0.99833441]
 [0.99944678 0.99438196 0.99955029 ... 0.99721853 0.99686059 0.99680792]]
```

print(cs_array)

4. **knn_classify**
   a. choose k largest values of cosine similarity in each row. (It means that k training data nodes which is close to the current testing data node.)
   b. Compare labels of this k nodes and choose the most common label as the result of the prediction of the testing data
   c. **k** is the parameter that we can decide the number of neighbors.

```
def knn_classify(X_train, X_test, y_train, y_test, k):
    cs_array = cos_sim(X_train, X_test, y_train, y_test)
    k_list = []
    y_pred_list = []
    for i in range(len(cs_array)):
        k_list = heapq.nlargest(k, range(len(cs_array[i])), cs_array[i].take)
        #k_list stores index of k largest cosine similarity value
        class_list = []
        for idx in k_list:
            class_list.append(y_train.iloc[idx])
            #class_list stores k prediction classes of each node.
        a = np.array(class_list)
        counts = np.bincount(a)
        print(np.argmax(counts))
        #choose the most common class as the result prediction
```

```
        y_pred_list.append(np.argmax(counts))
        print("====================")
    y_pred = pd.Series(y_pred_list)
    return y_pred
```

```
In [47]: runfile('/home/snoopyknight/文件/knn/knn.py', wdir='/home/snoopyknight/文件/knn')
[3, 3, 2, 3, 2]
3
====================
[2, 3, 2, 2, 2]
2
====================
[2, 1, 2, 2, 2]
2
====================
[2, 2, 2, 2, 2]
2
====================
[2, 2, 3, 1, 3]
2
====================
[2, 2, 2, 2, 2]
2
====================
[1, 1, 1, 1, 1]
1
====================
[3, 3, 2, 2, 3]
3
```

5. Check the classifier report and accuracy

```
from sklearn.metrics import classification_report as clf_report
from sklearn.metrics import accuracy_score

y_pred = knn_classify(X_train, X_test, y_train, y_test, 5)
accurancy = accuracy_score(y_test,y_pred)
report = clf_report(y_test,y_pred)
```

```
In [3]: runfile('C:/Users/dmlab/Desktop/knn_classifier-master/
knn.py', wdir='C:/Users/dmlab/Desktop/knn_classifier-master')
             precision    recall  f1-score   support

          1       0.85      0.94      0.89        31
          2       0.76      0.82      0.79        34
          3       0.83      0.62      0.71        24

avg / total       0.81      0.81      0.80        89

==========================================================
testing data accurancy =  0.808988764045
==========================================================
execution time :  0.0175299644447021484
```

report and accuracy

6. Execution time

```
import time
if __name__ == "__main__":
    start = time.time()
    main()
    end = time.time()
    print("============================================================")
    print("ececution time : ",end - start
```

Execution time : 0.03158402442932129 seconds

The code is on: https://github.com/SnoopyKnight/knn_classifier

Reference:

- http://enginebai.logdown.com/posts/241676/knn
- https://pandas.pydata.org/pandas-docs/stable/
- http://scikit-learn.org/stable/
- http://www.numpy.org/
- https://pythonhow.com/measure-execution-time-python-code/