1. **Description of the selected forecasting problem**

Iris flower can be classified into three related species - Setosa, Versicolor and Virginica. The specific type of the species can be judged according to the width and length of its Petal and Sepal. Hence, the goal of this assignment is to judge the type of the iris flower based on the input width and length number.

**2.Description of available data**

The data set this assignment uses has four attributes - Petal width, Petal length, Sepal width, Sepal length with a quantity of 150 rows in total. The data type of the attributes is float numbers ranged from 0 to 10. For the output, there are three categories : ‘Iris-setosa’, ‘Iris-versicolor’and ‘Iris-virginica’.

**3.Short overview of the selected algorithms**

1.) KNearest Neighbor

KNearest Neighbor is an algorithm that can be applied for both classification and regression in machine learning. Instead of calculating parameters during the learning process, KNN predicts the results based on the distances of its K Nearest neighbors.

1. ) Multi-layer perceptron

Multi-layer perceptron is a specific algorithm of Artificial Neural Network. Multi-layer perceptron is consist of three parts, the input layer, the hidden layer and the output layer. The input layer takes the input data from outside and performs several computations in one or more hidden layers, which are some non-linear functions. Finally, the algorithm outputs the result from the output layer.

1. ) Support Vector Machine

Support Vector Machine is a supervised learning algorithm which can be applied for both classification and regression. In classification, the purpose of SVM is to determine a widest hyperplane that can divide the data into several categories.

**4.Specifics about how algorithms were applied and the evaluation procedure.**

The algorithms are applied by python in this assignment. Python has a library called “sklearn” which provides several machine-learning algorithms functions for user to implement. In order to perform evaluation, this assignment separates 150 rows of raw data into training set and test set in a proportion of 8:2 by setting a random number of 0.8 during data processing. The training set will then be applied to a standard function call “fit(trainX, trainY)” while the test set is used to generate forecasting result through a function called “predict(testX).

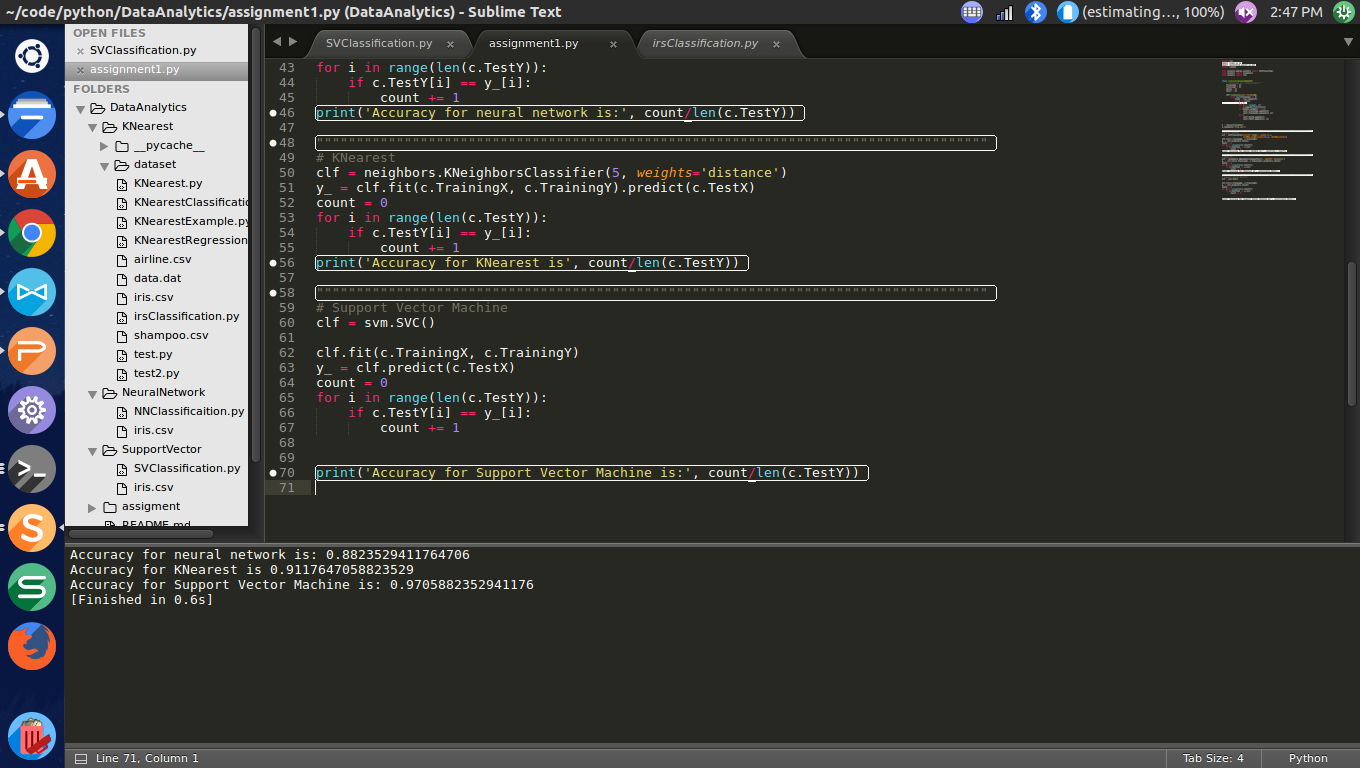
For KNN, this assignment sets the K factor as 5.

For SVM, the kernel is set to be “rbf”.

For MLP, one hidden layer with 15 nodes is defined in the algorithm.

1. **Accuracy Comparison.**

The result of the forecasting is shown as below:



The accuracy for Neural Network, KNearest and Support Vector Machine are approximately 88%, 91% and 97% respectively.

1. **Code**

import csv

import numpy as np

import matplotlib.pyplot as plt

import random

from sklearn.neural\_network import MLPClassifier

from sklearn import neighbors

from sklearn import svm

class classification(object):

"""docstring for classification"""

TrainingX = []

TrainingY = []

TestX = []

TestY = []

def readData(self, filename):

with open(filename) as f:

lines = csv.reader(f)

for i in lines:

l = []

for j in range(1, 5):

l.append(float(i[j]))

if random.random() < 0.8:

self.TrainingX.append(l)

self.TrainingY.append(i[-1])

else:

self.TestX.append(l)

self.TestY.append(i[-1])

c = classification()

c.readData('iris.csv')

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

# Neural Network

clf = MLPClassifier(solver='lbfgs', alpha=1e-5,

hidden\_layer\_sizes=(15,), random\_state=1)

clf.fit(c.TrainingX, c.TrainingY)

y\_ = clf.predict(c.TestX)

count = 0

for i in range(len(c.TestY)):

if c.TestY[i] == y\_[i]:

count += 1

print('Accuracy for neural network is:', count/len(c.TestY))

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

# KNearest

clf = neighbors.KNeighborsClassifier(5, weights='distance')

y\_ = clf.fit(c.TrainingX, c.TrainingY).predict(c.TestX)

count = 0

for i in range(len(c.TestY)):

if c.TestY[i] == y\_[i]:

count += 1

print('Accuracy for KNearest is', count/len(c.TestY))

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

# Support Vector Machine

clf = svm.SVC()

clf.fit(c.TrainingX, c.TrainingY)

y\_ = clf.predict(c.TestX)

count = 0

for i in range(len(c.TestY)):

if c.TestY[i] == y\_[i]:

count += 1

print('Accuracy for Support Vector Machine is:', count/len(c.TestY))