Machine Learning

Question 1.

The objective of last week's exercise was to familiarize yourself with using NumPy in order to perform basic analysis of a bike sharing dataset. This question focuses specifically on array based indexing in NumPy.

The following are the details of the various fields from the bike dataset.

1. instant: record index
2. season : season (1:springer, 2:summer, 3:fall, 4:winter)
3. yr : year (0: 2011, 1:2012)
4. mnth : month (1 to 12)
5. hr : hour (0 to 23)
6. holiday : weather day is holiday or not (extracted from [Web Link])
7. weekday : day of the week
8. workingday : if day is neither weekend nor holiday is 1, otherwise is 0.
9. + weathersit :
    i. 1: Clear, Few clouds, Partly cloudy, Partly cloudy
    ii. 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    iii. 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
    iv. 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
10. temp : Normalized temperature in Celsius. The values are divided to 41 (max)
11. atemp: Normalized feeling temperature in Celsius. The values are divided to 50 (max)
12. hum: Normalized humidity. The values are divided to 100 (max)
13. windspeed: Normalized wind speed. The values are divided to 67 (max)
14. casual: count of casual users
15. registered: count of registered users
16. cnt: count of total rental bikes including both casual and registered

Objectives

.

(i) Read the dataset bikeSharing.csv into a NumPy array.

Write a program that will compare the average number of total users (column index 15) on days that are holidays (1) with the average number of total users on days that are not holidays (0). Note that you should use array indexing to perform this task.

The above question focuses on both total users, which includes both registered and casual users. Determine if there is a difference if you only consider casual users (column index 13).

```
data = np.genfromtxt('bikeSharing.csv', delimiter=',')

compareHolidays(data, 0)
compareHolidays(data, 1)

def compareHolidays(data, holiday):
    subset = data[data[:, 5] == holiday]
    print "Number of entries ", len(subset)
    print "Mean", np.mean(subset[:, 15])
```

(ii) You will notice that the following columns in the dataset are normalized:

- temp : Normalized temperature in Celsius. The values are divided to 41 (max)
- atemp: Normalized feeling temperature in Celsius. The values are divided to 50 (max)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)

Your objective is to produce a new NumPy array. The new NumPy array should be a copy of the old with the real-values replacing the normalized values for each of the above columns.

```
# The following sample shows how to normalize the temp column.
    data = np.genfromtxt('bikeSharing.csv', delimiter=',')

    newdata = np.copy(data)

    print newdata[:, 9]

    newdata[:, 9] *= 9

    print newdata[:, 9]
```

(iii) Generally on a given day the number of registered users outnumber the number of casual users. Determine the percentage of the days in the dataset where the casual users outnumber the registered users (You should be able to do this in 2 or 3 lines of code using a relational operator).

```
data = np.genfromtxt('bikeSharing.csv', delimiter=',')

result = data[:, 13]>data[:, 14]

print "Percentage of time where causal users > registered ",
(len(data[result])*100.0)/len(data)
```

(iv) In this question you should provide a new implementation of one of last week's questions using array indexing. The objective of this task is to investigate the impact of weather conditions on the popularity of the bike scheme. For each of the 4 possible weather conditions calculate the average number of rental bikes.

See sample output of program below.

```
Average number of bikes is  189.463087635
Average bikes for condition  Clear  is  204.869271883
Average bikes for condition  Mist and Cloudy  is  175.165492958
Average bikes for condition  Light Rain  is  111.579281184
Average bikes for condition  Heavy Rain  is  74.3333333333
```

```
def averageNumRentalBikesPerCondition(data):

    conditions = {1:"Clear", 2:"Misty", 3:"Light Rain", 4:"Heavy Rain"}

    for key in conditions:
        subsetData = data[data[:,8]==key]
        print np.mean(subsetData[:, 15])

def main():

    data = np.genfromtxt('bikeSharing.csv', delimiter=',')

    averageNumRentalBikesPerCondition(data)
```

(v) The objective of this question is to look at the relationship between temperature and the number of casual users.

Your code should work out the average number of casual users for the following temperature ranges:

1, 5
6, 10
11, 15
16, 20
21, 25
26, 30
31, 35
36, 40

Please note the temperature range specified in the file have been normalised by dividing by 41

The following is the sample output:

```
For temp in range  1 to 5 the mean number of casual users was  49.2954545455
For temp in range  6 to 10 the mean number of casual users was  73.6670630202
For temp in range  11 to 15 the mean number of casual users was  130.681770652
For temp in range  16 to 20 the mean number of casual users was  169.066772655
For temp in range  21 to 25 the mean number of casual users was  211.700074516
For temp in range  26 to 30 the mean number of casual users was  242.172678691
For temp in range  31 to 35 the mean number of casual users was  337.473005641
For temp in range  36 to 40 the mean number of casual users was  314.991111111
```

```
def main():

        data = np.genfromtxt('bikeSharing.csv', delimiter=',')

        for temp in range(1, 40, 5):
            analyseTemp(data, temp, temp+4)

def analyseTemp(data, minValue, maxValue):
    # the temperature values stored in the array are multiplied by 41

    higherTempCondition = (data[:,9]*41)>=minValue
    lowerTempCondition = (data[:,9]*41)<=maxValue

    subset = data[higherTempCondition & lowerTempCondition]
```

```
    meanValue = np.mean(subset[:, 15])
    print "For temp in range ", minValue, "to", maxValue, "the mean number of casual
users was ", meanValue
```