BDA - Lab 3 : Machine Learning

Student1: Akshay Gurudath(Aksgu350)

Student2: Keshav Padiyar Manuru(Kespa139)

In [ ]:

```
from __future__ import division
from math import radians, cos, sin, asin, sqrt, exp
from datetime import datetime
from pyspark import SparkContext
import sys

# Set up Spark Context
sc = SparkContext(appName = "BDA Lab3")

# Methods Section

def is_leap_year(year):
    return (year % 4 == 0 and year % 100 != 0) or year % 400 == 0

def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    km = 6367 * c
    return km

def date_diff(date1,date2):
    date1 = datetime.strptime(date1, "%Y-%m-%d")
    date2 = datetime.strptime(date2, "%Y-%m-%d")
    diff = abs(date1 - date2).days
    ret = 1 if diff<=2190 else 0
    return ret


def date_distance (date1, date2):
    """
    Calculates the number of days between the dates from
    data and the prediction date.

    Algorithm keeps track about the shortest distance between
    the 2 dates, considering the leap years as well.
    """
    date1 = datetime.strptime(date1, "%Y-%m-%d")
    year = date1.year
    date2 = datetime.strptime(date2, "%Y-%m-%d").replace(year=year)

    if is_leap_year(year):
        fix_year = datetime.strptime(str(year)+'-01'+'-01', "%Y-%m-%d")
        date_diff = (date1 - fix_year).days
        pred_diff = (date2 - fix_year).days
        diff = abs(pred_diff - date_diff)
        dif = diff if diff<183 else 366 - diff
```

```python
    else:
        fix_year = datetime.strptime(str(year)+'-01'+'-01', "%Y-%m-%d")
        date_diff = (date1 - fix_year).days
        pred_diff = (date2 - fix_year).days
        diff = abs(pred_diff - date_diff)
        diff = diff if diff<182 else 365 - diff

    return diff

def time_distance(time1, time2):
    """
    Calculates the time differences in Hours
    """
    time1 = datetime.strptime(time1, '%H:%M:%S').hour
    time2 = datetime.strptime(time2, '%H:%M:%S').hour
    diff = abs(time1- time2)
    diff = diff if diff<=12 else 24 - diff
    return diff


def gaussian_kernel(distance, h):
    """
    Gaussian Kernel
    """
    return(exp(-((distance**2)/h)))



def k_sum (k1,k2,k3):
    return k1+k2+k3

def k_prod(k1, k2, k3):
    return k1 * k2 * k3


# Kernel Parameters
h_distance = 300000 # Up to you
h_date = 1000 # Up to you
h_time = 31 # Up to you

#
a = 58.4274 # Up to you
b = 14.826 # Up to you
date = "2013-07-04" # Up to you

rdd_tempReadings = sc.textFile("file:///home/x_kesma/Lab1/input_data/temperature-r
eadings.csv") \
                            .map(lambda line: line.split(";"))
# Trim data till given date and time
rdd_tempReadings = rdd_tempReadings.filter(lambda line: (line[1]<date) & (date_dif
f(line[1],date)!=0))

rdd_stations = sc.textFile("file:///home/x_kesma/Lab1/input_data/stations.csv")\
                            .map(lambda line: line.split(";"))

tempReadings = rdd_tempReadings.map(lambda line: (line[0],(line[1],line[2], float(
line[3]))))
stations = rdd_stations.map(lambda line: (line[0],(line[3],line[4])))
```

```
# Distance Between Stations:
dist_stations = stations.map(lambda line: (line[0],\
                                           gaussian_kernel(haversine(float(line[1]
[1]),\
                                                                     float(line[1]
[0]),\
                                                                     b,a),\
                                                           h_distance)))

dict_dist_stations = sc.broadcast(dist_stations.collectAsMap())

# Date Distances:
dist_dates = tempReadings.map(lambda line:(line[0],(line[1][1],line[1][2],gaussian
_kernel(date_distance(line[1][0], date),\

h_date))))

# Combine Geo-distance and Date distance

dist_Station_dates_join = dist_dates.map(lambda line: (line[0],(line[1][0],line[1]
[2],\
                                                                dict_dist_stations
.value[line[0]],\
                                                                line[1][1])))#.cac
he()

sumOut = []
prodOut = []
i = 0
#24,22,20,18,16,14,12,10,8,6,4
times = ["00:00:00", "22:00:00", "20:00:00", "18:00:00", "16:00:00", "14:00:00",\
             "12:00:00", "10:00:00", "08:00:00", "06:00:00", "04:00:00"]
for time in times:
    print("Executing for - {}".format(time))
    kMatrix = dist_Station_dates_join.map(lambda line:(line[0],(line[1][1],line[1]
[2],\
                                                                gaussian_kernel(ti
me_distance(line[1][0],time),h_time),\
                                                                line[1][3] )))

    kTransform = kMatrix.map(lambda line: (k_sum(line[1][0],line[1][1],line[1][2
]),\
                                           k_prod(line[1][0],line[1][1],line[1][2
]), line[1][3]))
    totalSum = kTransform.map(lambda line: (line[0] * line[2],line[0])).reduce(lam
bda a,b: (a[0]+b[0],a[1]+b[1]))
    prodSum  = kTransform.map(lambda line: (line[1] * line[2],line[1])).reduce(lam
bda a,b: (a[0]+b[0],a[1]+b[1]))
    sumOut.append(totalSum[0]/totalSum[1])
    prodOut.append(prodSum[0]/prodSum[1])
    print(sumOut)
    print(prodOut)
    i = i+1


with open("/home/x_kesma/Lab1/input_data/results/BDA_LAB3/Output.txt", "w") as out
put:
```

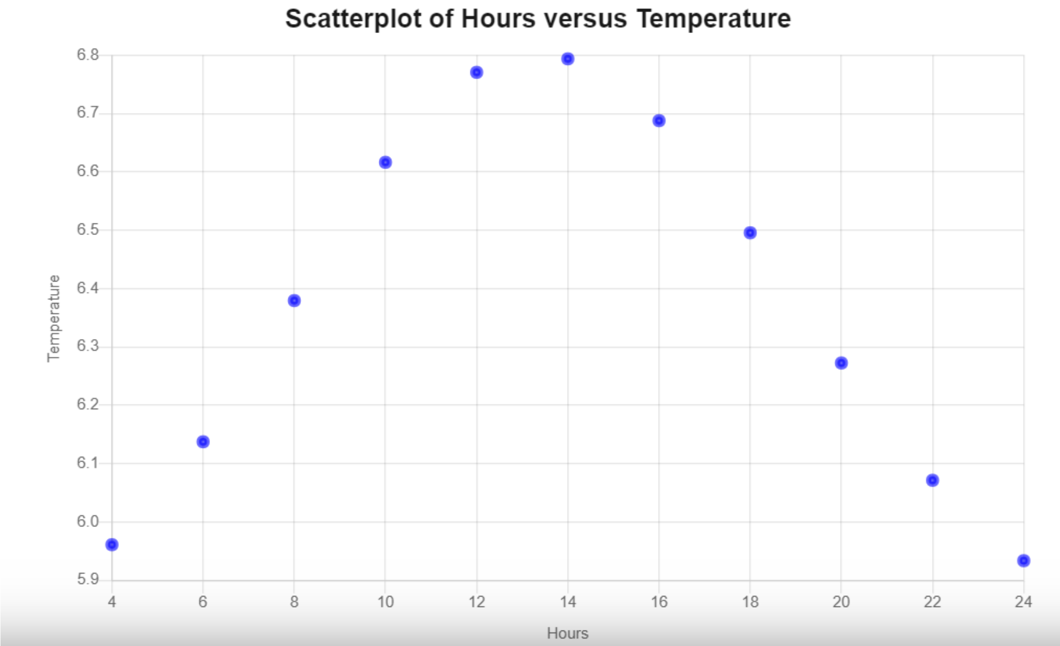
```
    output.write(str(sumOut))
    output.write("\n")
    output.write(str(prodOut))

sys.exit(0)
```

Result:

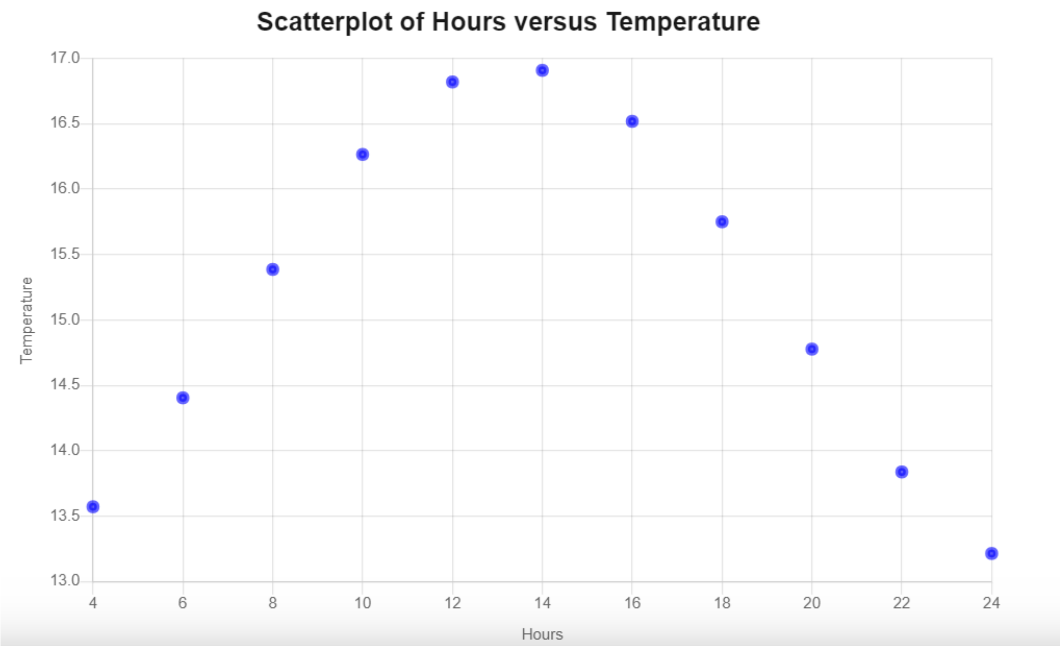
Sum of Kernels: [5.93366287206746, 6.071460586739478, 6.272508775348385, 6.495681068923899, 6.687857955604679, 6.793800389977985, 6.770658221493563, 6.616436685920447, 6.379472539603611, 6.137476539006086, 5.961064016641998]

Highest temperature observed at 14:00 - 6.793800389977985



Product Of Kernels: [13.214303350649812, 13.837661264208363, 14.776440763847974, 15.749926630659463, 16.517560476009848, 16.907384219646037, 16.818710257003158, 16.264490516499457, 15.385649093723485, 14.404241617759448, 13.571464771209166]

Highest temperature observed at 14:00 - 16.907384219646037



The above graph is obtained when we multiply all the kernels together. We see that the temperatures we get now are different compared to the summation kernel. This is because when we multiply the kernels together, the gaussian kernel is made narrower For example, if we take the individual weights returned by the three gaussian kernels as [0.8,0.6,0.1]. Effectively, this is 50% weight (1.5/3)). But when these weights are multiplied together, the effective weight is 0.048, which is roughly 4.8%. We see that now because one of the kernels have low weight, the effective kernel weight has reduced a lot.

For distances, we decided to weigh distances that are 400 kms away by 60%. Substituting this in the formula for weights of the kernel, we obtained h = 300,000.

For days, we decided to weigh days that are 50 days away by 70%. And then substituting this in the formula for weights of the kernel, we obtained h=1000

For time, the maximum difference between any two times is 12 hours. We decided to weight this maximum difference of 12 hours by 1%. Substituting this, we obtained h=31.