BIKE SHARING DATASET

A capstone Report

In partial fulfilment of the degree

Bachelor of technology

In

Computer Science & Engineering

By

V. Sathwila: 2003A52019

Ibrahim Afnan: 2003A52013

Syed Akramuddin: 2003A52017

Gopi Krishna: 2003A52015

Under the Guidance of

Dr. RAMESH

Submitted to

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SR UNIVERSITY, ANANTHASAGAR, WARANGAL

SR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify the project report entitled **“BIKE SHARING DATASET”** is a record of bonafide work carried out by the student(s) **Sathwika (2003A52019)**, **Ibrahim Afnan (2003A52013)**, **Syed Akramuddin (2003A52017)**, **Gopi Krishna (2003A52015)**. During the academic year 2020-2021 in partial fulfilment of the award of the degree of **Bachelor of Technology in Computer Science & Engineering (AI & ML)**.

Supervisor **Head of the Department**

ABSTRACT

This dataset contains the hourly and daily count of rental bikes between the years 2011 and 2012 in the Capital bikeshare system with the corresponding weather and seasonal information. In the current generation where the society is very health conscious and concerned about the nature, the people avoiding the usage of motor transport has become a trend. This analysis is based on the climatic and temporal factors that help us in understanding the scenarios that favour the use of rental bikes among casual and registered riders. Capital bikeshare has over 350 stations in Washington. Bike sharing systems are a new way of traditional bike rentals. The whole process from membership to rental and return back has become automatic. The data was generated by 500 bike-sharing programs and was collected by the Laboratory of Artificial Intelligence and Decision Support (LIAAD), University of Porto.

Bike sharing is a new form of transport and is becoming increasingly popular in cities around the world. This study aims to quantitatively estimate the environmental benefits of bike sharing. Using big data techniques, we estimate the impacts of bike sharing.

TABLE OF CONTENTS:

Contents

1. INTRODUCTION

Bike sharing systems are new generation of traditional bike rentals where whole process from membership, rental and return back has become automatic. Through these systems, user is able to easily rent a bike from a particular position and return back at another position. Currently, there are about over 500 bike-sharing programs around the world which is composed of over 500 thousand's bicycles. Today, there exists great interest in these systems due to their important role in traffic, environmental and health issues. Apart from interesting real-world applications of bike sharing systems, the characteristics of data being generated by these systems make them attractive for the research. Opposed to other transport services such as bus or subway, the duration of travel, departure and arrival position is explicitly recorded in these systems. This feature turns bike sharing system into a virtual sensor network that can be used for sensing mobility in the city. Hence, it is expected that most of important events in the city could be detected via monitoring these data. Bike-sharing rental process is highly correlated to the environmental and seasonal settings. For instance, weather conditions, precipitation, day of week, season, hour of the day, etc. can affect the rental behaviour's The core data set is related to the two-year historical log corresponding to years 2011 and 2012 from Capital Bikeshare system, Washington D.C., USA. Bike sharing systems allow the users to take one way bicycle trips over short distances. Generally, these systems are operated via automated kiosks to save manpower and reduce waiting time for the users. Bike Sharing System ensures that pollution is reduced as with use of bicycles there is reduction in use of motor vehicles which leads to reduction in emission of pollutants in the air. This practice of Bike Sharing Systems is common in Western Countries while the same is not seen yet in countries like India. In India most of the bike sharing systems could not achieve their maximum potential as data analysis was not used properly. The advantage of this system is that we can have public bike stations without any human involvement.

1.1. EXISTING SYSTEM

Bike sharing is an emerging industry and it is very popular in western countries, while people have tried to start the same in India, we will look into some of the stats regarding how many people use bike sharing systems. According to Wikipedia by August 2014 only 600 cities in the world had bike sharing systems and most of them were in western countries with a fleet of about 500000 bicycles with them. There is a sharp increase in Next Bike, Cogo BikeShare are some of the leading Bike Sharing systems that are currently in operation in the world. While considering Indian perspective in the Bike Share industry, India has not yet adapted the application of this emerging industry. Currently there are a few bike share systems in India and still are running on test basis, some of them are:

a. Namma Cycles:

This was started in Bengaluru in August 2012, as a IISC project and is still working efficiently. This system has 5
cycle stations and about 50 bicycles with them. Limitation with Namma Cycles is that it is practiced in a closed area i.e. a Campus area and Government is also participating in this initiative.

B. Current use of Analytics in Bike Share Systems.

The current use of analytics in Bike Share Industry is encouraging, most of the companies are having their own analytics divisions. Before starting any new Bike Station, the companies analyse how much the station will be useful and will it generate appropriate revenue and whether setting up the station is feasible or not. With enhancements in analytic techniques in current era, with some simple surveys Bike Share companies can forecast the use of the system and then plan accordingly.

1.2. PROBLEM DEFINATION:

Generally, in bicycle sharing systems it is very important that the administrators should know how many cycles will be needed in each bicycle station, knowing this count enables them the arrange proper number of cycles at the stations and decide whether a particular station needs to have extra number of bicycles stands So in this research work we study various prediction algorithms i.e., random forests, decision trees, gradient boosting machines. This research work focuses on which algorithm can work better for the real-world problem of bicycle sharing demand prediction.

2. LITERATURE SURVEY

2.1 RELATED WORK

a. **PYTHON -** Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed AND supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

b. **GOOGLE COLLAB –** Google Collab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Collab supports many popular machine learning libraries which can be easily loaded in your notebook.

2.2. SYSTEM STUDY

In this project, we will be investigating into the bike share rental data from "Capital Bikeshare" servicing Washington D.C. and surrounding areas beginning 2010. Capital Bikeshare was the largest bike sharing service in the United States when they started, until Citi Bike for New York City started operations in 2013. Capital Bikeshare started from 10 stations and 120 bicycles in Washington D.C. and expanding into a bike share system that owns more than 429 stations and 2500 bicycles and also services.

3. DESIGNS

3.1 Requirement specifications

3.1.1 software requirements

Python

Python 3

Libraries

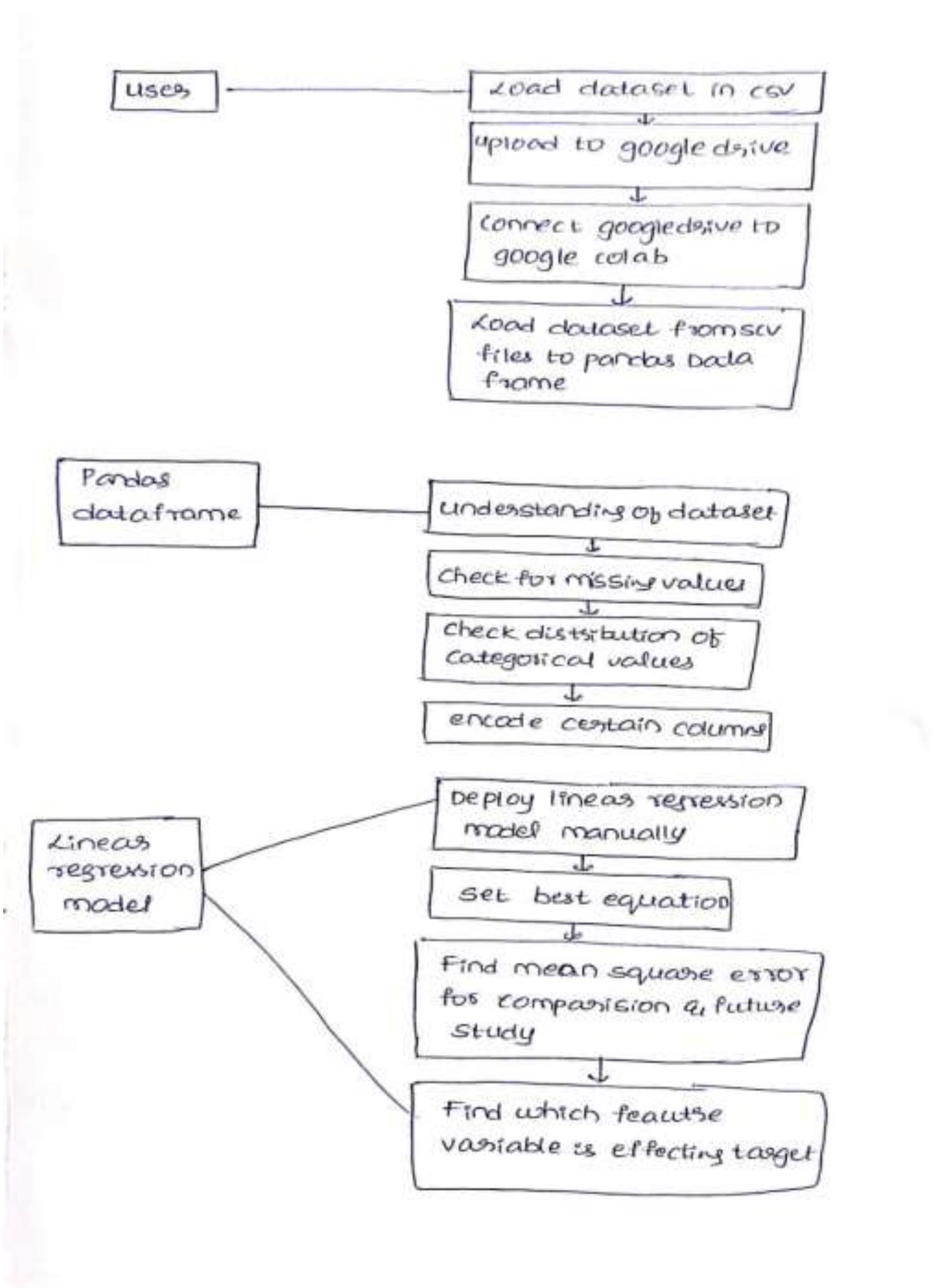
- Numpy
- OpenCV
- Win sound
- Operating system, etc…

3.1.2. Hardware requirements

1. laptop with basic hardware

3.2 UML Diagram

Use case diagram



user
Load dataset in csv
upload to google drive
connect googledrive to google colab
Load dataset from scv files to pandas Data frame
Pandas dataframe
understanding of dataset
check for missing values
check distribution of categorical values
encode certain columns
Linear regression model
Deploy linear regression model manually
set best equation
Find mean square error for comparision & future study
Find which feature variable is effecting target

4. IMPLEMENTATION

```
import pandas as pd
d=pd.read_csv('/content/day.csv')
print(d)
```

```
     instant      dteday  season  yr  ...  windspeed  casual  registered  c
ount
0          1  01-01-2011       1   0  ...   0.160446     331         654   
985
1          2  02-01-2011       1   0  ...   0.248539     131         670   
801
2          3  03-01-2011       1   0  ...   0.248309     120        1229   
1349
3          4  04-01-2011       1   0  ...   0.160296     108        1454   
1562
4          5  05-01-2011       1   0  ...   0.186900      82        1518   
1600
..       ...         ...     ...  ..  ...        ...     ...         ...   
...
726      727  27-12-2012       1   1  ...   0.350133     247        1867   
2114
727      728  28-12-2012       1   1  ...   0.155471     644        2451   
3095
728      729  29-12-2012       1   1  ...   0.124383     159        1182   
1341
729      730  30-12-2012       1   1  ...   0.350754     364        1432   
1796
730      731  31-12-2012       1   1  ...   0.154846     439        2290   
2729

[731 rows x 16 columns]
```

```
x1=d['season']
y=d['count']
print(x1,y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
726    1
727    1
728    1
729    1
730    1
Name: season, Length: 731, dtype: int64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
```

```
729    1796
730    2729
Name: count, Length: 731, dtype: int64
```

In [ ]:

```
ee=[]
import numpy as np
mm=[]
cc=[]
def liner(m1,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):
    yp.append(m1*x1[i]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[42])
```

```
19
42
52 62
50
lll 19
```

In [ ]:

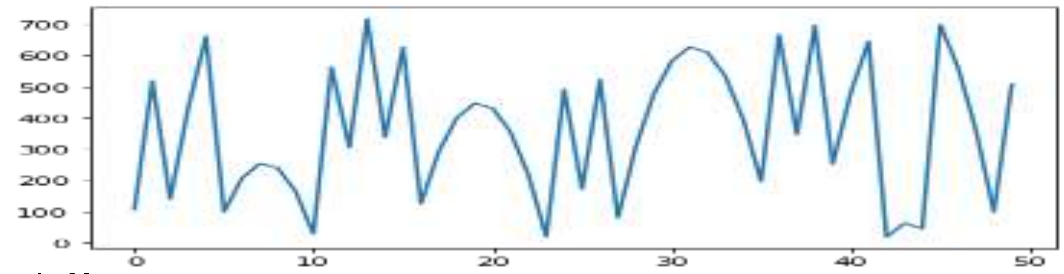
```
for i in range(0,50):
  liner(i+10,i+20)
```

```
106
519
140
431
661
99
207
254
240
165
29
563
305
717
337
627
125
```

```
293
400
446
431
355
218
20
492
172
522
80
308
475
581
626
610
533
395
196
667
346
695
252
479
645
19
63
46
699
560
360
99
508
```

```python
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,50):
  x.append(i)
pt.plot(x,ee)
```

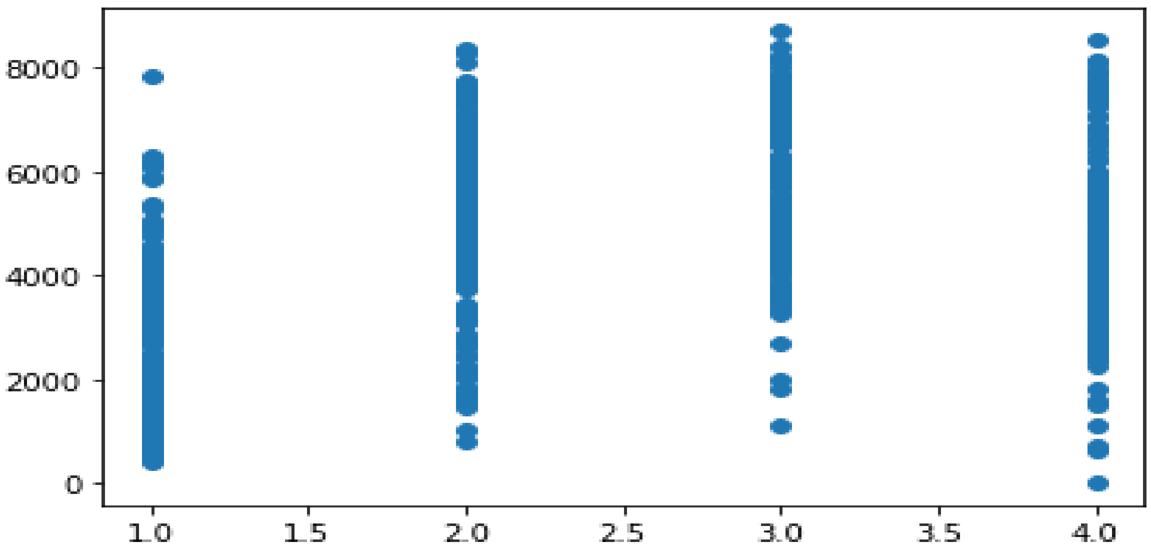
```
[<matplotlib.lines.Line2D at 0x7fc9cc81c790>]
```



In [ ]:

```python
from matplotlib import pyplot as plt
```

```
plt.scatter(x1,y)
```



```
<matplotlib.collections.PathCollection at 0x7fc9cc784c90>
```

```
x2=d['yr']
y=d['count']
print(x2,y)
```

```
0      0
1      0
2      0
3      0
4      0
      ..
726    1
727    1
728    1
729    1
730    1
Name: yr, Length: 731, dtype: int64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
729    1796
730    2729
Name: count, Length: 731, dtype: int64
```

In [ ]:

```
ee=[]
import numpy as np
mm=[]
cc=[]
def liner(m1,m2,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):
    yp.append(m1*x1[i]+m2*x2[i]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[25])
```

```
2
25
35 50
50
lll 2
```

In [ ]:

```
for i in range(0,50):
  liner(i+10,i+20,i+i)
```

```
313
573
33
155
208
192
107
684
461
169
539
109
341
504
598
623
579
466
284
33
444
55
328
532
667
2
730
658
517
```

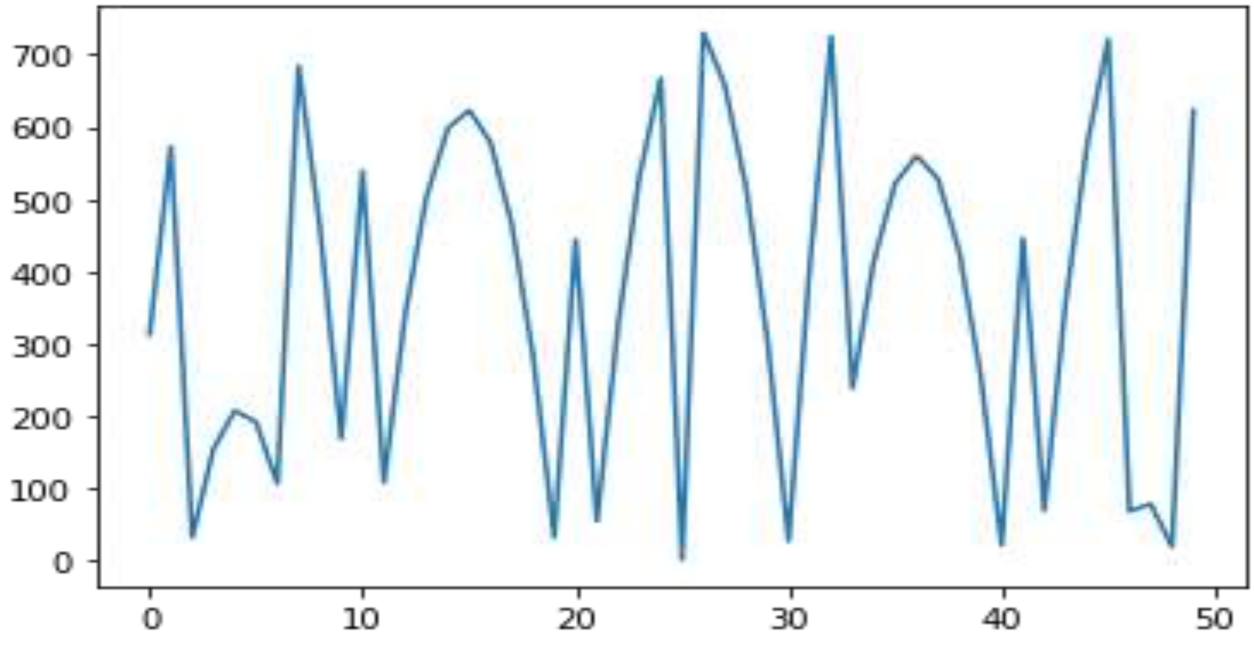
```
307
28
411
725
239
415
522
560
529
429
260
22
446
70
356
573
721
69
79
20
623
```

In [ ]:

```
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,50):
  x.append(i)
pt.plot(x,ee)
```

Out[ ]:

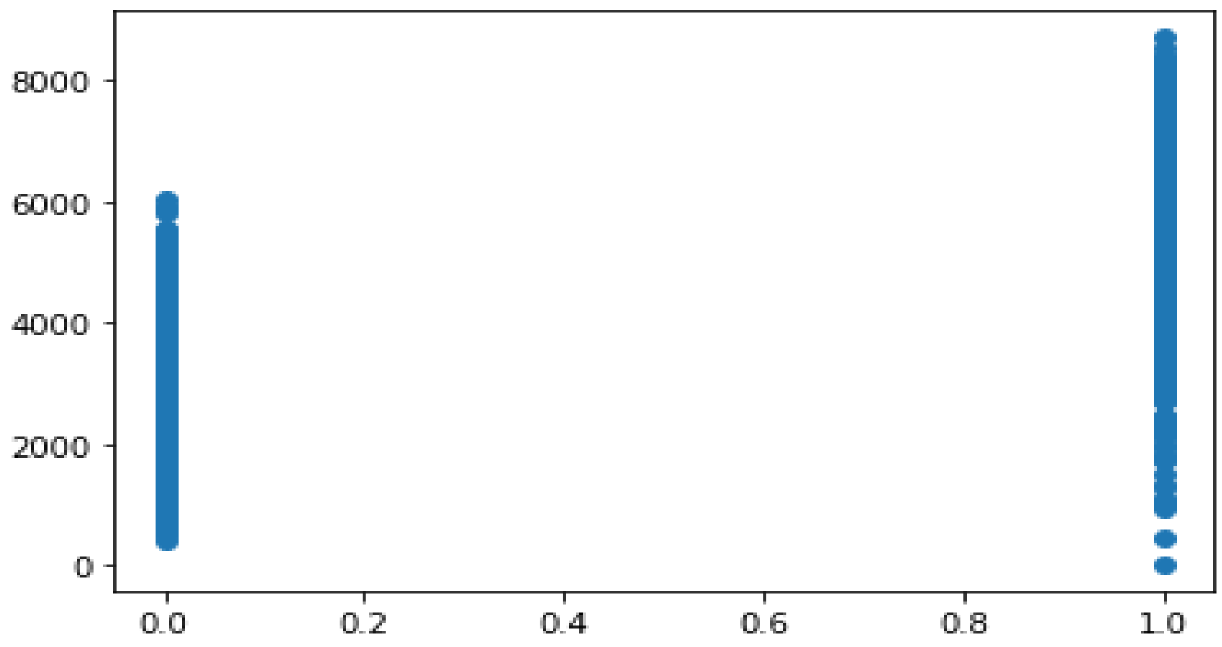
```
[<matplotlib.lines.Line2D at 0x7fc9ccd13d90>]
```



In [ ]:

```
from matplotlib import pyplot as plt
plt.scatter(x2,y)
```

Out[ ]:



```
<matplotlib.collections.PathCollection at 0x7fc9cc83b450>
```

In [ ]:

```
x3=d['mnth']
y=d['count']
print(x3,y)
```

In [ ]:

```
ee=[]
import numpy as np
mm=[]
cc=[]
def liner(m1,m2,m3,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):
    yp.append(m1*x1[i]+m2*x2[i]+m3*x3[i]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[21])
```

```
6
```

```
21
31 22
50
lll 6
```

In [ ]:

```
for i in range(0,50):
  liner(i+10,i+20,i+13,i+1)
```

```
317
334
132
442
533
405
58
223
169
627
135
155
687
269
363
238
625
62
11
472
714
6
541
126
223
101
491
662
614
347
592
618
425
13
113
725
387
561
516
252
500
529
339
661
33
648
313
490
448
```

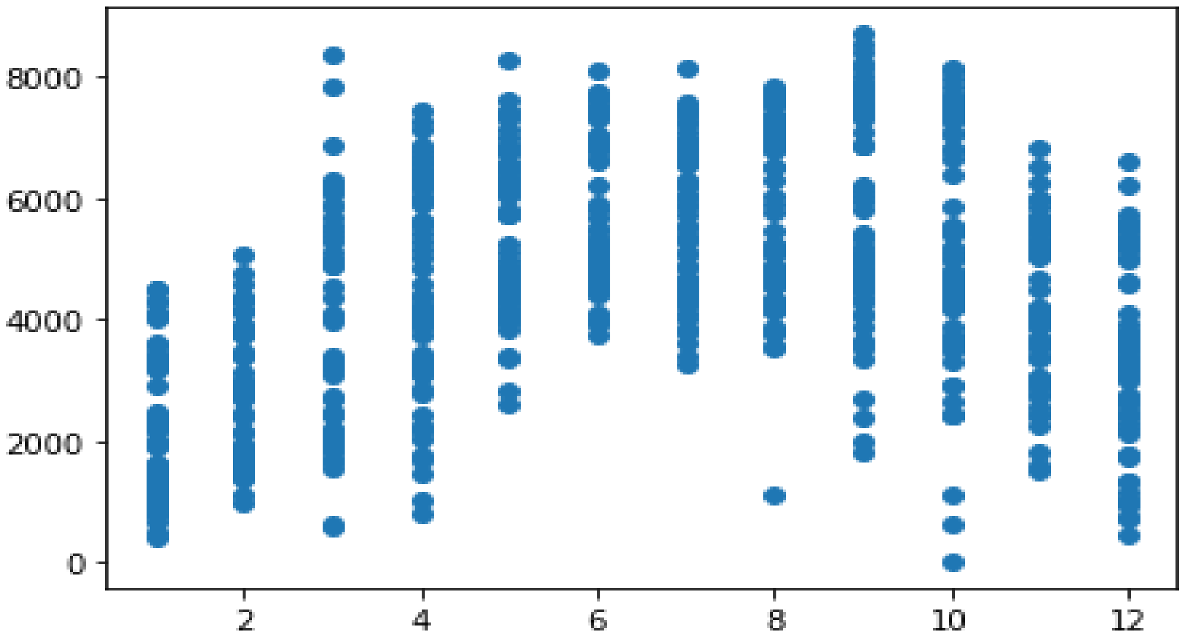
```
187
```

In [ ]:

```
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,50):
  x.append(i)
pt.plot(x,ee)
```

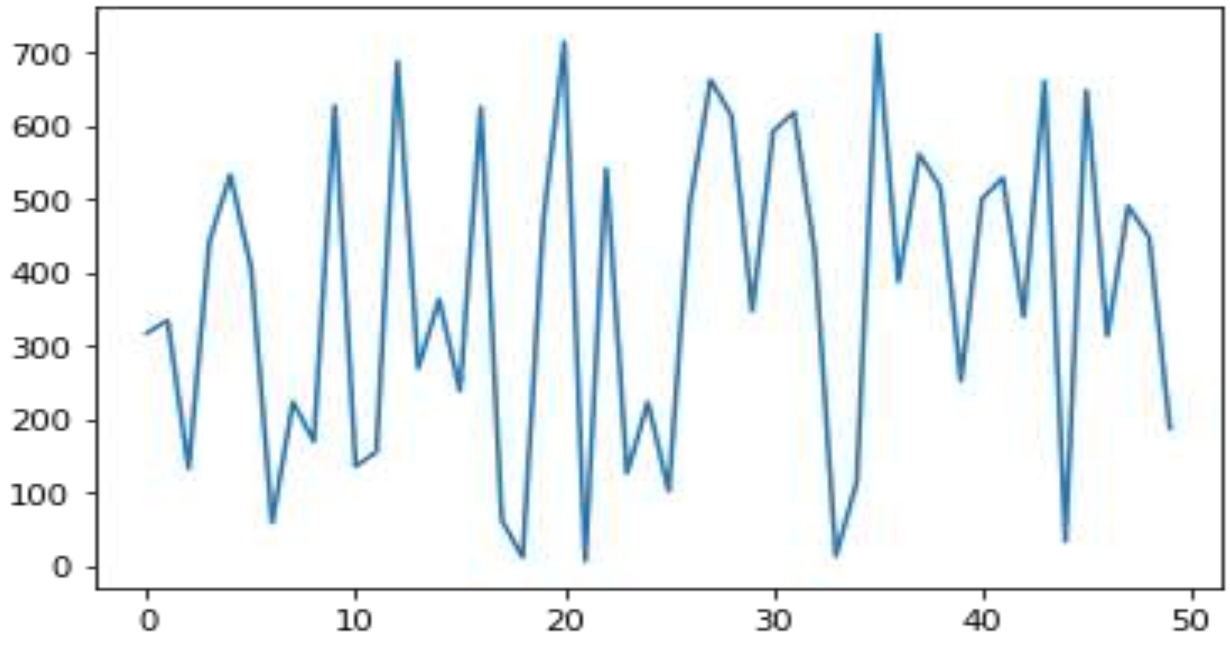
Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cc8ba510>]
```



```
from matplotlib import pyplot as plt
plt.scatter(x3,y)
```

```
<matplotlib.collections.PathCollection at 0x7fc9cce3b250>
```



In [ ]:

```
x4=d['holiday']
y=d['count']
print(x4,y)
```

```
0      0
1      0
2      0
3      0
4      0
      ..
726    0
727    0
728    0
729    0
730    0
Name: holiday, Length: 731, dtype: int64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
729    1796
730    2729
Name: count, Length: 731, dtype: int64
```

In [ ]:

```
ee=[]
import numpy as np
mm=[]
cc=[]
def liner(m1,m2,m3,m4,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):
    yp.append(m1*x1[i]+m2*x2[i]+m3*x3[i]+m4*x4[i]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[70])
```

```
4
70
80 140
80
lll 4
```

In [ ]:

```
for i in range(0,80):
  liner(i+10,i+20,i+13,i+2,i+i)
```

```
649
317
123
67
149
369
727
492
395
436
615
201
656
518
518
656
201
615
436
395
492
727
369
149
67
123
317
649
388
265
280
433
724
422
258
232
344
594
251
46
710
50
259
606
360
252
282
450
```

```
25
469
320
309
436
701
373
183
131
217
441
72
572
479
524
707
297
25
622
626
37
317
4
560
523
624
132
509
293
215
275
473
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,80):
  x.append(i)
pt.plot(x,ee)
```

```
[<matplotlib.lines.Line2D at 0x7fc9cc6d6d90>]
```

```
from matplotlib import pyplot as plt
plt.scatter(x4,y)
```

```
<matplotlib.collections.PathCollection at 0x7fc9cc644590>
```

```
x5=d['weekday']
y=d['count']
print(x5,y)
```

```
0      6
1      0
2      1
3      2
4      3
      ..
726    4
727    5
```

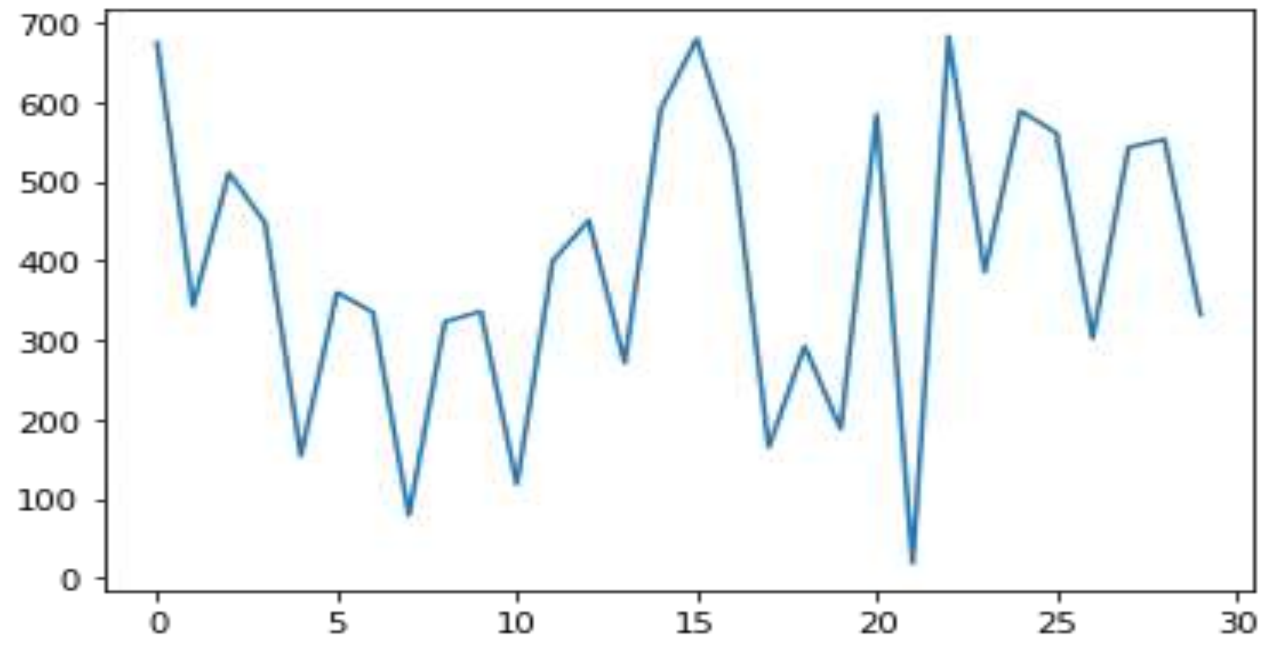
```
728    6
729    0
730    1
Name: weekday, Length: 731, dtype: int64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
729    1796
730    2729
Name: count, Length: 731, dtype: int64
ee=[]
import numpy as np
mm=[]
cc=[]
def liner(m1,m2,m3,m4,m5,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):
    yp.append(m1*x1[i]+m2*x2[i]+m3*x3[i]+m4*x4[i]+m5*x5[i]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)

a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[21])

18
21
41 42
30
lll 18

for i in range(0,30):
  liner(i+20,i+10,i+30,i+5,i+2,i+i)

675
343
511
448
154
360
335
79
```

```
323
336
118
400
451
271
591
680
538
165
292
188
584
18
683
386
589
561
302
543
553
332
```

In [ ]:

```
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,30):
  x.append(i)
pt.plot(x,ee)
```

Out[ ]:



```
[<matplotlib.lines.Line2D at 0x7fc9cc5b7090>]
```

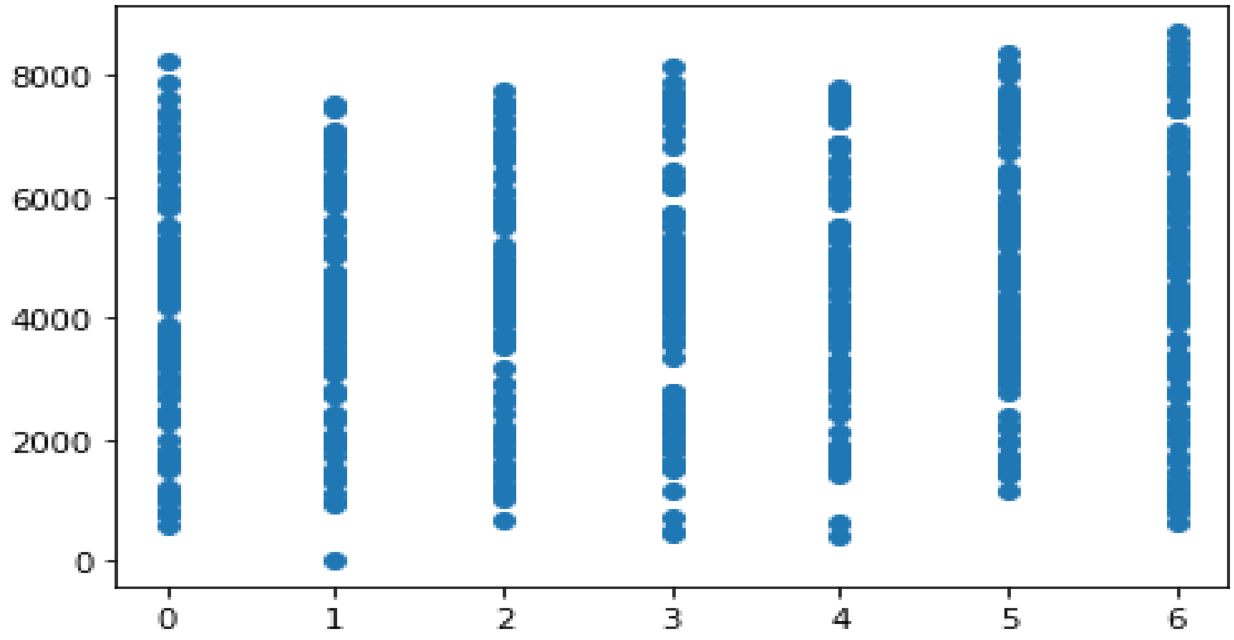
In [ ]:

```
from matplotlib import pyplot as plt
```

```
plt.scatter(x5,y)
```

Out[ ]:

```
<matplotlib.collections.PathCollection at 0x7fc9cc59d910>
```



In [ ]:

```
x6=d['workingday']
y=d['count']
print(x6,y)
```

```
0      0
1      0
2      1
3      1
4      1
      ..
726    1
727    1
728    0
729    0
730    1
Name: workingday, Length: 731, dtype: int64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
729    1796
730    2729
Name: count, Length: 731, dtype: int64
```

In [ ]:

```
ee=[]
import numpy as np
mm=[]
```

```
cc=[]
def liner(m1,m2,m3,m4,m5,m6,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):
    yp.append(m1*x1[i]+m2*x2[i]+m3*x3[i]+m4*x4[i]+m5*x5[i]+m6*x6[i]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[27])
```

```
8
27
47 28
50
lll 8
```

In [ ]:

```
for i in range(0,50):
  liner(i+20,i+10,i+3,i+5,i+2,i+4,i+1)
```

```
532
543
535
508
462
397
313
210
88
678
518
339
141
655
419
164
621
328
16
416
66
428
40
364
```

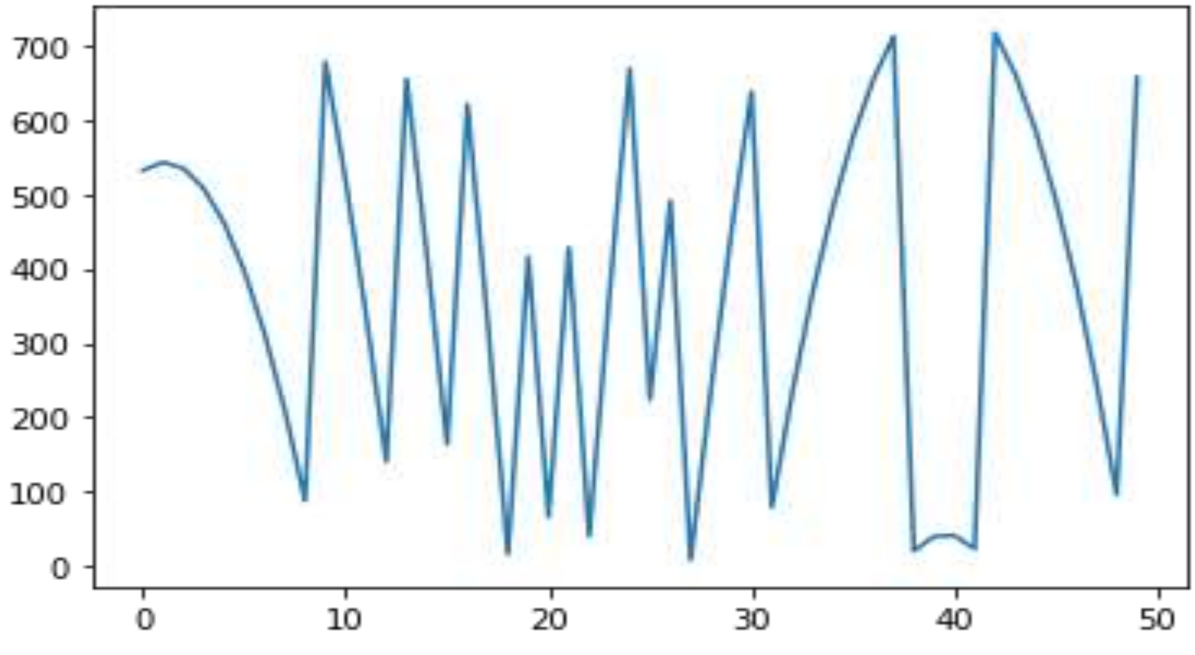
```
669
224
491
8
237
447
638
79
232
366
481
577
654
712
20
40
41
23
717
661
586
492
379
247
96
657
```

In [ ]:

```
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,50):
  x.append(i)
pt.plot(x,ee)
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cc514890>]
```

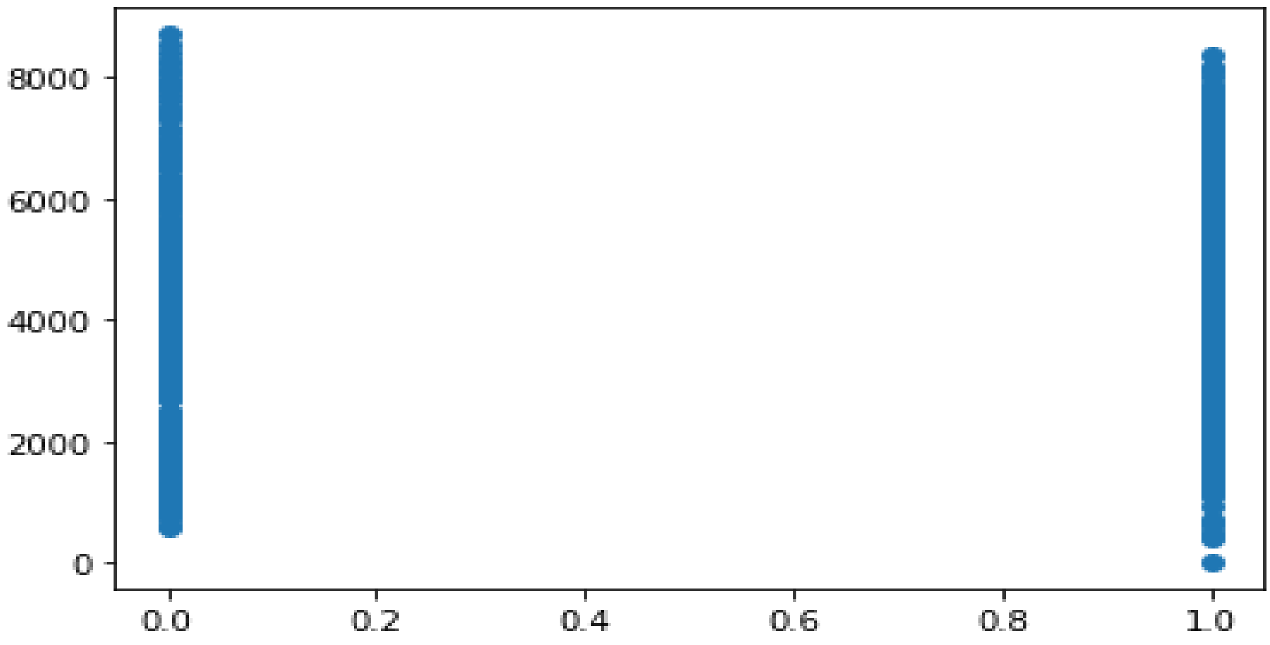


In [ ]:

```
from matplotlib import pyplot as plt
plt.scatter(x6,y)
```

Out[ ]:

```
<matplotlib.collections.PathCollection at 0x7fc9cc47b9d0>
```



In [ ]:

```
x7=d['weathersit']
y=d['count']
print(x7,y)
```

```
0      2
1      2
2      1
3      1
4      1
      ..
726    2
727    2
728    2
729    1
730    2
Name: weathersit, Length: 731, dtype: int64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
729    1796
730    2729
Name: count, Length: 731, dtype: int64
```

In [ ]:

```
ee=[]
```

```
import numpy as np
mm=[]
cc=[]
def liner(m1,m2,m3,m4,m5,m6,m7,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):

yp.append(m1*x1[i]+m2*x2[i]+m3*x3[i]+m4*x4[i]+m5*x5[i]+m6*x6[i]+m7*x7[i]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[40])
```

```
15
44
64 88
50
lll 319
```

In [ ]:

```
for i in range(0,50):
  liner(i+20,i+10,i+3,i+5,i+2,i+4,i+1,i+i)
```

```
155
574
162
381
500
519
438
257
707
326
576
726
45
726
576
326
707
257
438
519
500
```

```
381
162
574
155
367
479
491
403
215
658
270
513
656
699
642
485
228
602
145
319
393
367
241
15
420
725
199
304
309
```

In [ ]:

```
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,50):
  x.append(i)
pt.plot(x,ee)
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cc3ef790>]
```

In [ ]:

```
from matplotlib import pyplot as plt
plt.plot(x7,y,'r')
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cc354bd0>]
```

In [ ]:

```
x8=d['temp']
y=d['count']
print(x8,y)
```

```
0      0.344167
1      0.363478
2      0.196364
3      0.200000
4      0.226957
          ...
```

```
726    0.254167
727    0.253333
728    0.253333
729    0.255833
730    0.215833
Name: temp, Length: 731, dtype: float64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
729    1796
730    2729
Name: count, Length: 731, dtype: int64
```

In [ ]:

```
ee=[]
import numpy as np
mm=[]
cc=[]
def liner(m1,m2,m3,m4,m5,m6,m7,m8,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):

yp.append(m1*x1[i]+m2*x2[i]+m3*x3[i]+m4*x4[i]+m5*x5[i]+m6*x6[i]+m7*x7[i]+m8
*x8[i]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[34])
```

```
16.357688903808594
34
54 40
50
lll 16.357688903808594
```

In [ ]:

```
for i in range(0,50):
  liner(i+20,i+10,i+7,i+5,i+4,i+3,i+2,i+1,i+6)
```

```
180.95965385437012
```

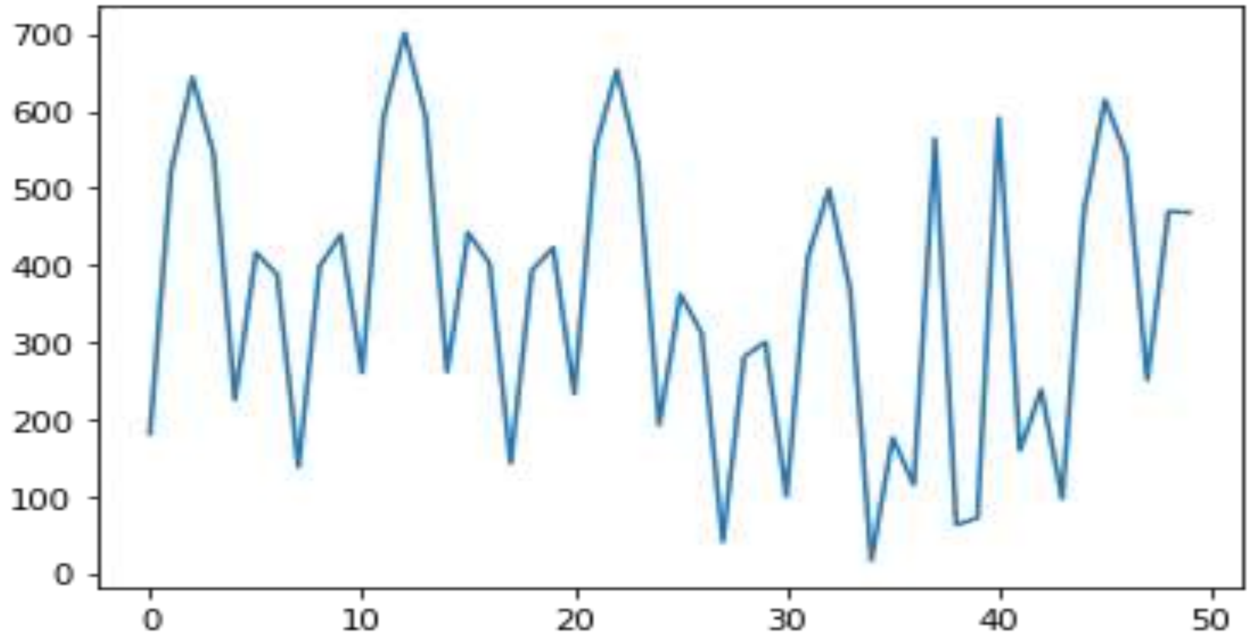
```
522.5592231750488
643.7987670898438
544.6782550811768
225.19769477844238
416.35709381103516
387.15642166137695
137.59571266174316
398.6749382019043
439.3941535949707
259.7532768249512
590.7523670196533
701.3914127349854
591.6704063415527
261.58934783935547
442.148229598999
402.3470630645752
142.18588256835938
392.66461753845215
422.78330421447754
232.54194450378418
552.9405250549316
652.9790649414062
532.6575527191162
191.9760036468506
361.9344005584717
311.5327396392822
40.77103233337402
280.6492691040039
300.16745948791504
99.32561302185059
409.1236991882324
498.56174087524414
367.63973236083984
16.357688903808594
175.71557235717773
114.7134017944336
564.3512191772461
62.628950119018555
71.5466423034668
591.1042938232422
159.30188751220703
238.13942527770996
96.61692810058594
465.73437118530273
614.4917621612549
542.8891105651855
250.92639350891113
469.6036434173584
467.9208564758301
```

In [ ]:

```
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,50):
  x.append(i)
pt.plot(x,ee)
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cc2e75d0>]
```

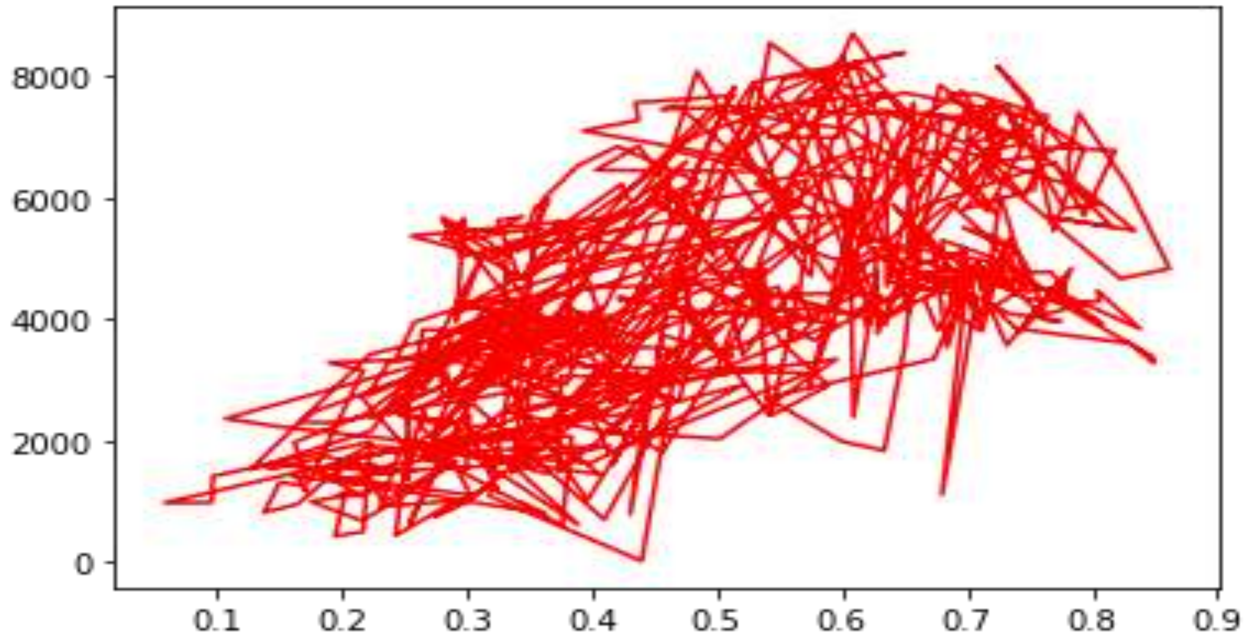


In [ ]:

```
from matplotlib import pyplot as plt
plt.plot(x8,y,'r')
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cc24bdd0>]
```



In [ ]:

```
x9=d['atemp']
y=d['count']
print(x9,y)
```

```
0       0.363625
```

```
1      0.353739
2      0.189405
3      0.212122
4      0.229270
         ...
726    0.226642
727    0.255046
728    0.242400
729    0.231700
730    0.223487
Name: atemp, Length: 731, dtype: float64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
729    1796
730    2729
Name: count, Length: 731, dtype: int64
```

In [ ]:

```
ee=[]
import numpy as np
mm=[]
cc=[]
def liner(m1,m2,m3,m4,m5,m6,m7,m8,m9,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):

yp.append(m1*x1[i]+m2*x2[i]+m3*x3[i]+m4*x4[i]+m5*x5[i]+m6*x6[i]+m7*x7[i]+m8
*x8[i]+m9*x9[i]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[49])
```

```
2.771759033203125
49
69 98
50
lll 2.771759033203125
```

In [ ]:

```
for i in range(0,50):
  liner(i+20,i+10,i+9,i+8,i+3,i+5,i+2,i+4,i+1,i+i)
```

```
226.37081336975098
195.6606903076172
167.90479278564453
143.10313987731934
121.25575828552246
102.36262321472168
86.4237174987793
73.43909072875977
63.4087028503418
56.33255577087402
52.210655212402344
51.04303169250488
52.82964897155762
57.57048988342285
65.2656078338623
75.91497421264648
89.51856994628906
106.07642364501953
125.58853340148926
148.05487823486328
173.47548484802246
201.85034942626953
233.17945671081543
267.46281242370605
304.70041275024414
344.89227294921875
388.03838539123535
434.13872718811035
483.19333839416504
535.2021999359131
590.1652908325195
648.082649230957
708.954252243042
41.780107498168945
108.56022834777832
178.2945728302002
250.9831657409668
326.62603187561035
405.22313499450684
486.77448654174805
571.280086517334
658.7399444580078
18.154050827026367
111.52239227294922
207.84500122070312
307.1218433380127
409.3529510498047
514.5383052825928
622.6779041290283
2.771759033203125
```

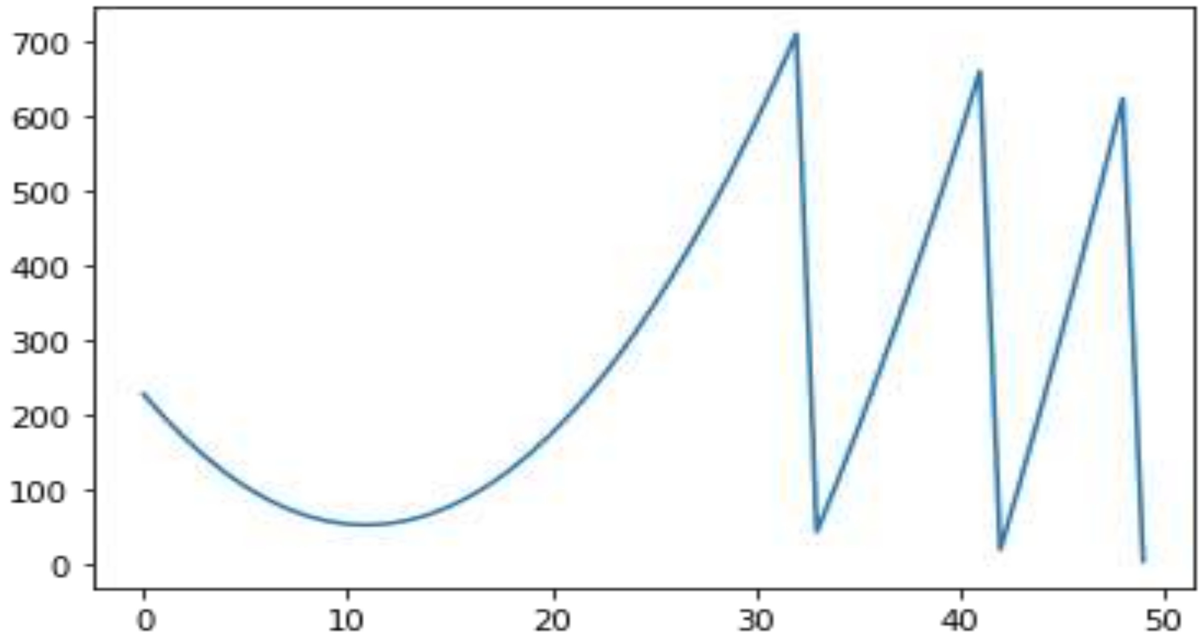
In [ ]:

```
from matplotlib import pyplot as pt
import numpy as np
```

```
x=[]
for i in range(0,50):
  x.append(i)
pt.plot(x,ee)
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cc1b4450>]
```

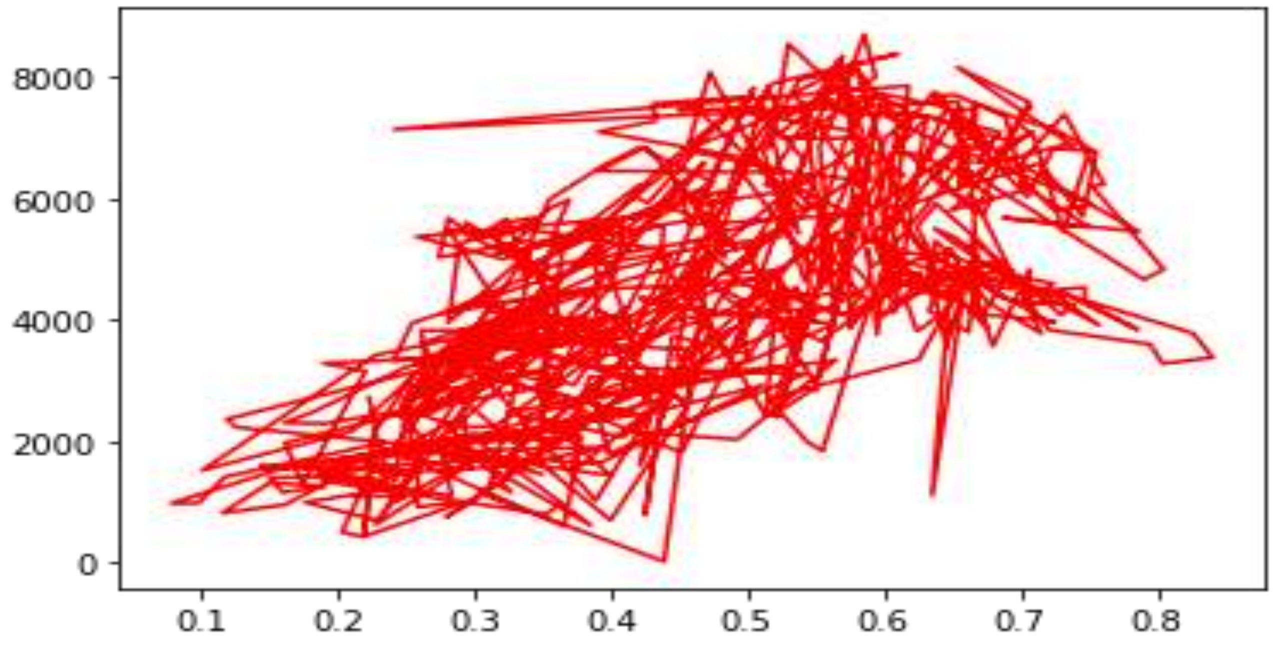


In [ ]:

```
from matplotlib import pyplot as plt
plt.plot(x9,y,'r')
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cc11c9d0>]
```



In [ ]:

```
x10=d['hum']
y=d['count']
print(x10,y)
```

```
0      0.805833
1      0.696087
2      0.437273
3      0.590435
4      0.436957
         ...
726    0.652917
727    0.590000
728    0.752917
729    0.483333
730    0.577500
Name: hum, Length: 731, dtype: float64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
729    1796
730    2729
Name: count, Length: 731, dtype: int64
```

In [ ]:

```
ee=[]
import numpy as np
mm=[]
cc=[]
def liner(m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):

yp.append(m1*x1[i]+m2*x2[i]+m3*x3[i]+m4*x4[i]+m5*x5[i]+m6*x6[i]+m7*x7[i]+m8
*x8[i]+m9*x9[i]+m10*x[10]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[72])
```

```
4.441112518310547
72
82 144
100
lll 4.441112518310547
```

In [ ]:

```
for i in range(0,100):
  liner(i+10,i+20,i+30,i+6,i+5,i+4,i+3,i+2,i+1,i+9,i+i)
```

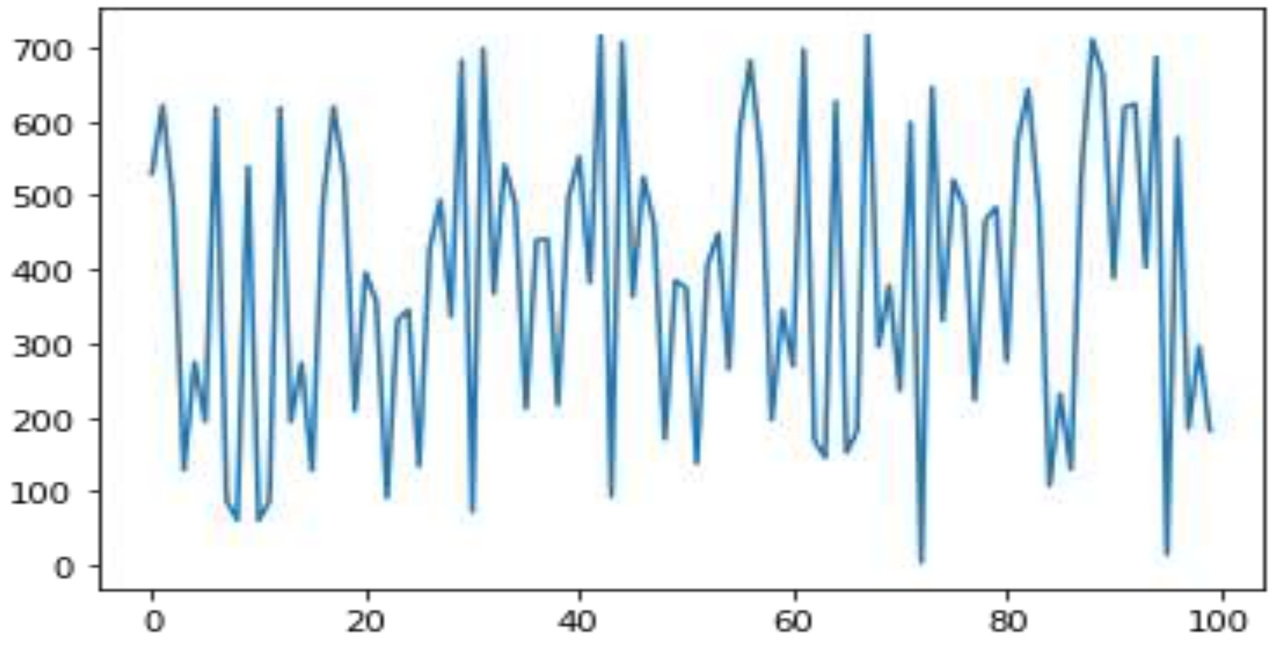
```
529.3412227630615
621.7629203796387
488.3007583618164
128.95466995239258
274.7246723175049
194.61078453063965
619.6129722595215
87.73126983642578
60.96563148498535
539.3160934448242
60.782676696777344
87.36535263061523
619.0641021728516
193.8789520263672
273.8098907470703
127.85692405700684
487.02005767822266
620.2992973327637
527.6945934295654
209.20601081848145
395.8335132598877
356.5771255493164
91.43682670593262
331.41261100769043
345.50449562072754
133.71245193481445
427.036527633667
494.47669410705566
336.03294944763184
682.7052879333496
72.49374580383301
698.3982906341553
367.41889572143555
541.5556354522705
489.8084602355957
212.1773738861084
439.662389755249
441.2634754180908
216.98067665100098
497.81394386291504
552.7633419036865
381.828821182251
716.0103912353516
93.30805397033691
706.7218170166016
363.2516670227051
524.8976125717163
460.65965366363525
```

```
170.53778076171875
385.53201484680176
374.6423349380493
137.86874771118164
406.211256980896
448.66985034942627
265.24454975128174
586.9353332519531
682.7422199249268
552.6652011871338
196.70427131652832
345.85943603515625
269.130690574646
697.5180463790894
169.02149772644043
145.64103412628174
627.3766717910767
152.22839736938477
182.19621467590332
717.2801332473755
295.48014068603516
378.7962427139282
236.2284393310547
598.7767333984375
4.441112518310547
646.2215929031372
331.11816596984863
521.1308269500732
485.25959396362305
223.5044355392456
466.8653898239136
484.34243392944336
275.93556785583496
572.6447954177856
643.4701137542725
488.4115324020386
107.4690465927124
231.64264678955078
129.9323387145996
533.3381395339966
710.8600215911865
662.4979982376099
388.2520694732666
619.1222352981567
624.1085028648376
403.2108540534973
687.4293007850647
14.76383924484253
578.2144756317139
184.7812089920044
296.4640336036682
182.26294565200806
```

In [ ]:

```
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,100):
  x.append(i)
pt.plot(x,ee)
```

Out[ ]:

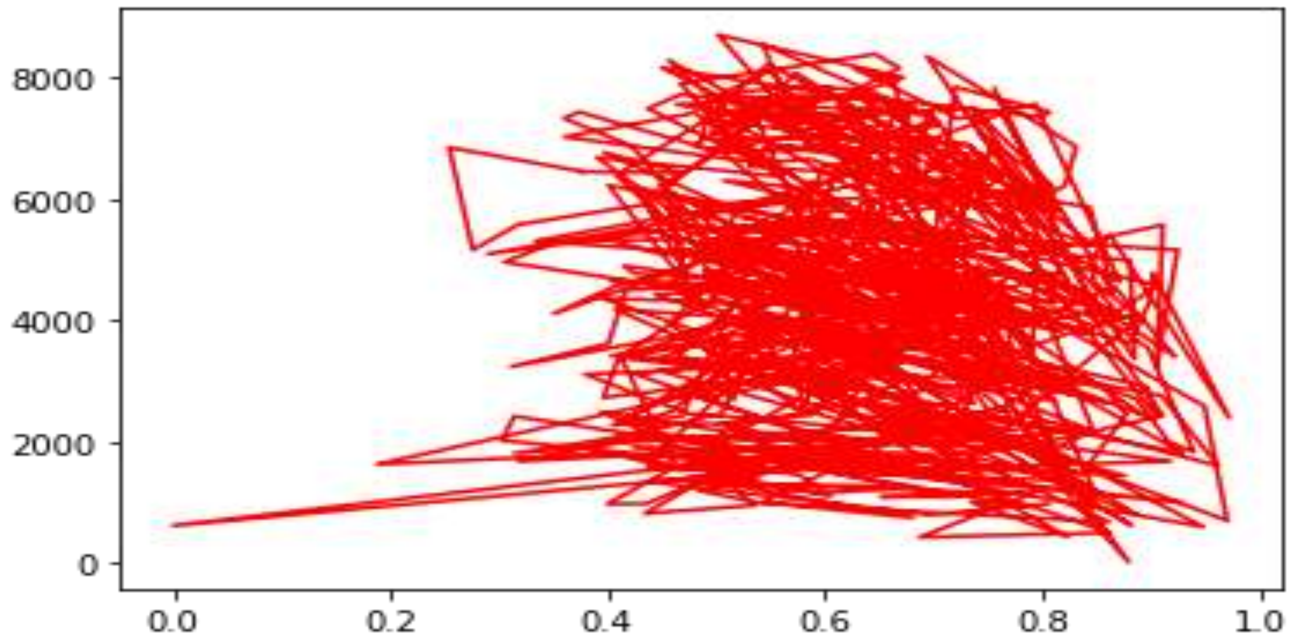


```
[<matplotlib.lines.Line2D at 0x7fc9cc03b690>]
```

In [ ]:

```
from matplotlib import pyplot as plt
plt.plot(x10,y,'r')
```

Out[ ]:



```
[<matplotlib.lines.Line2D at 0x7fc9cc01ad10>]
```

In [ ]:

```
x11=d['windspeed']
y=d['count']
print(x11,y)
```

```
0      0.160446
1      0.248539
2      0.248309
3      0.160296
4      0.186900
         ...
726    0.350133
727    0.155471
728    0.124383
729    0.350754
730    0.154846
Name: windspeed, Length: 731, dtype: float64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
729    1796
730    2729
Name: count, Length: 731, dtype: int64
```

In [ ]:

```
ee=[]
import numpy as np
mm=[]
cc=[]
def liner(m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):

yp.append(m1*x1[i]+m2*x2[i]+m3*x3[i]+m4*x4[i]+m5*x5[i]+m6*x6[i]+m7*x7[i]+m8
*x8[i]+m9*x9[i]+m10*x[10]+m11*x11[11]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[17])
```

```
13.493705749511719
17
27 34
50
lll 13.493705749511719
```

In [ ]:

```
for i in range(0,50):
  liner(i+10,i+20,i+10,i+3,i+5,i+4,i+3,i+2,i+1,i+9,i+6,i+i)
```

```
659.4980621337891
332.9363956451416
370.3199234008789
40.64860725402832
74.9224853515625
473.1415386199951
504.3057441711426
168.41513633728027
196.46969604492188
588.4694423675537
613.4143619537354
271.3044605255127
293.1397361755371
678.9201850891113
697.6457958221436
349.3165817260742
364.9325542449951
13.493705749511719
26.000011444091797
402.4515075683594
411.84818267822266
54.19002914428711
60.47704315185547
430.7092475891113
433.88661193847656
70.0091381072998
70.07687187194824
434.08977127075195
431.0478324890137
60.95109558105469
54.79950141906738
412.59309577941895
403.33188247680664
27.015810012817383
14.644937515258789
366.21923065185547
350.7386951446533
699.2033424377441
680.6131801605225
294.96815490722656
273.2683391571045
615.5136699676514
590.7041969299316
198.83990287780762
170.92076683044434
506.946813583374
475.9180393218994
77.83442115783691
```

```
43.69599914550781
373.502742767334
```

In [ ]:

```
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,50):
  x.append(i)
pt.plot(x,ee)
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cbf84fd0>]
```

In [ ]:

```
from matplotlib import pyplot as plt
plt.plot(x11,y,'r')
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cbefa990>]
```

In [ ]:

```
x12=d['casual']
y=d['count']
print(x12,y)
```

```
0      331
1      131
2      120
3      108
4       82
      ...
726    247
727    644
728    159
729    364
730    439
Name: casual, Length: 731, dtype: int64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
729    1796
730    2729
Name: count, Length: 731, dtype: int64
```

In [ ]:

```
ee=[]
import numpy as np
mm=[]
cc=[]
def liner(m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):

yp.append(m1*x1[i]+m2*x2[i]+m3*x3[i]+m4*x4[i]+m5*x5[i]+m6*x6[i]+m7*x7[i]+m8
*x8[i]+m9*x9[i]+m10*x[10]+m11*x11[11]+m12*x12[i]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[17])
```

```
17.428466796875
17
27 34
50
lll 17.428466796875
```

In [ ]:

```
for i in range(0,50):
  liner(i+10,i+20,i+10,i+3,i+5,i+4,i+3,i+2,i+1,i+9,i+6,i+30,i+i)
```

```
397.8577880859375
30.87353515625
384.2149658203125
726.882568359375
327.87548828125
649.1944580078125
228.83935546875
528.80908203125
87.105712890625
365.726806640625
633.67578125
159.946533203125
406.548583984375
642.470458984375
136.721923828125
351.2978515625
555.199951171875
17.428466796875
199.980712890625
371.8603515625
533.063720703125
683.59619140625
92.451904296875
221.63623046875
340.140625
447.974609375
545.13623046875
```

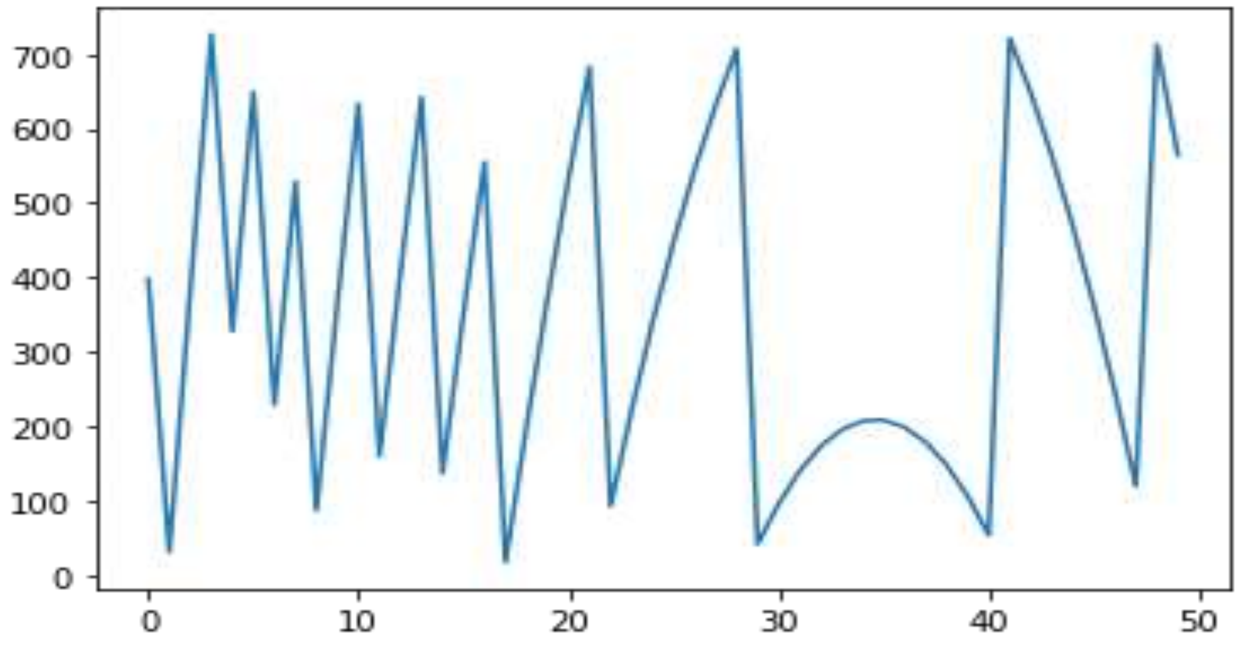
```
631.62158203125
707.43310546875
41.5673828125
96.02783203125
139.81591796875
172.93359375
195.3759765625
207.134765625
208.22607421875
198.640625
178.38818359375
147.45556640625
105.85205078125
53.56787109375
721.61572265625
647.98779296875
563.68603515625
468.7060546875
363.0498046875
246.7216796875
119.720703125
713.0546875
564.7041015625
```

In [ ]:

```
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,50):
  x.append(i)
pt.plot(x,ee)
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cc032410>]
```



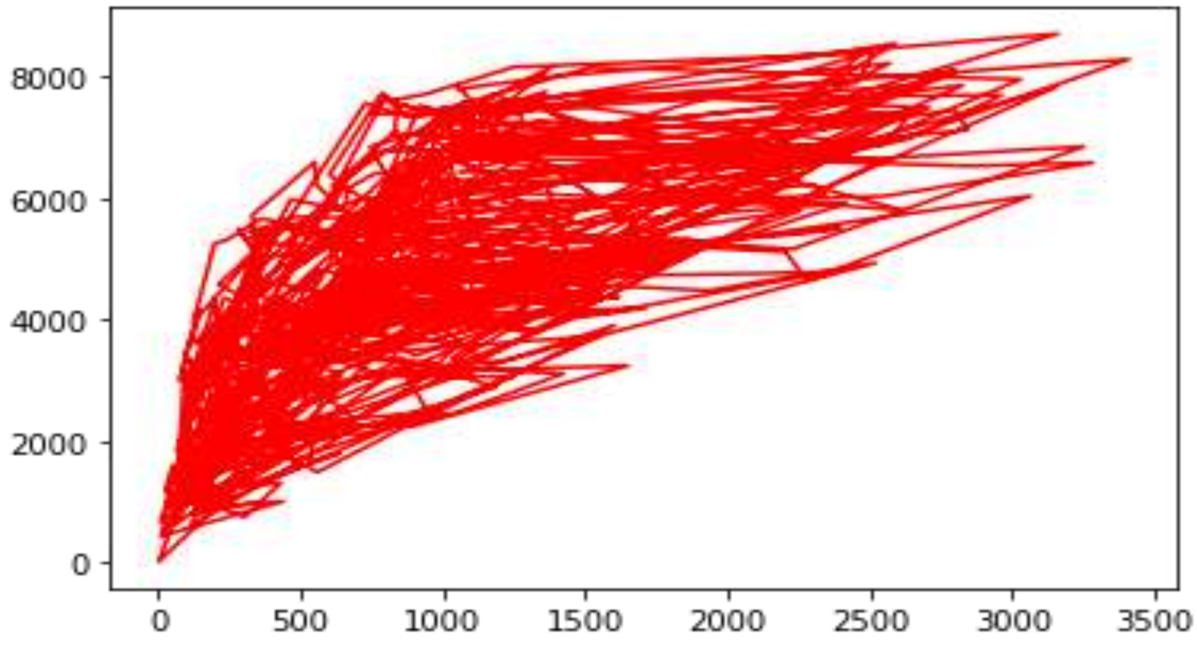
In [ ]:

```
from matplotlib import pyplot as plt
```

```
plt.plot(x12,y,'r')
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cc5c91d0>]
```



In [ ]:

```
x13=d['registered']
y=d['count']
print(x13,y)
```

```
0       654
1       670
2      1229
3      1454
4      1518
       ...
726    1867
727    2451
728    1182
729    1432
730    2290
Name: registered, Length: 731, dtype: int64 0       985
1       801
2      1349
3      1562
4      1600
       ...
726    2114
727    3095
728    1341
729    1796
730    2729
Name: count, Length: 731, dtype: int64
```

In [ ]:

```
ee=[]
import numpy as np
mm=[]
```

```
cc=[]
def liner(m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,c):
  ye=[]
  yp=[]
  sum=0
  mm.append(m1)
  cc.append(c)
  for i in range(0,len(x1)):

yp.append(m1*x1[i]+m2*x2[i]+m3*x3[i]+m4*x4[i]+m5*x5[i]+m6*x6[i]+m7*x7[i]+m8
*x8[i]+m9*x9[i]+m10*x[10]+m11*x11[11]+m12*x12[i]+m13*x[13]+c)
    ye.append((y[i]-yp[i])**2)
  for i in range(0,len(x1)):
    sum=sum+ye[i]
  ee1=np.mod(sum,len(x1))
  ee.append(ee1)
  print(ee1)
```

In [ ]:

```
a=np.array(ee).min()
print(a)
ind=ee.index(a)
print(ind)
print(mm[ind],cc[ind])
print(len(ee))
print("lll",ee[55])
```

```
6.0234375
55
65 110
60
lll 6.0234375
```

In [ ]:

```
for i in range(0,60):
  liner(i+10,i+20,i+10,i+3,i+5,i+4,i+3,i+2,i+1,i+9,i+6,i+30,i+2,i+i)
```

```
652.0244140625
569.410888671875
337.31591796875
686.7374267578125
155.6766357421875
206.134765625
107.1092529296875
589.601318359375
191.613037109375
375.1402587890625
409.1875
293.751220703125
28.833984375
345.43017578125
512.545166015625
530.17919921875
398.331787109375
117.0009765625
417.1865234375
567.892333984375
569.117919921875
420.856201171875
```

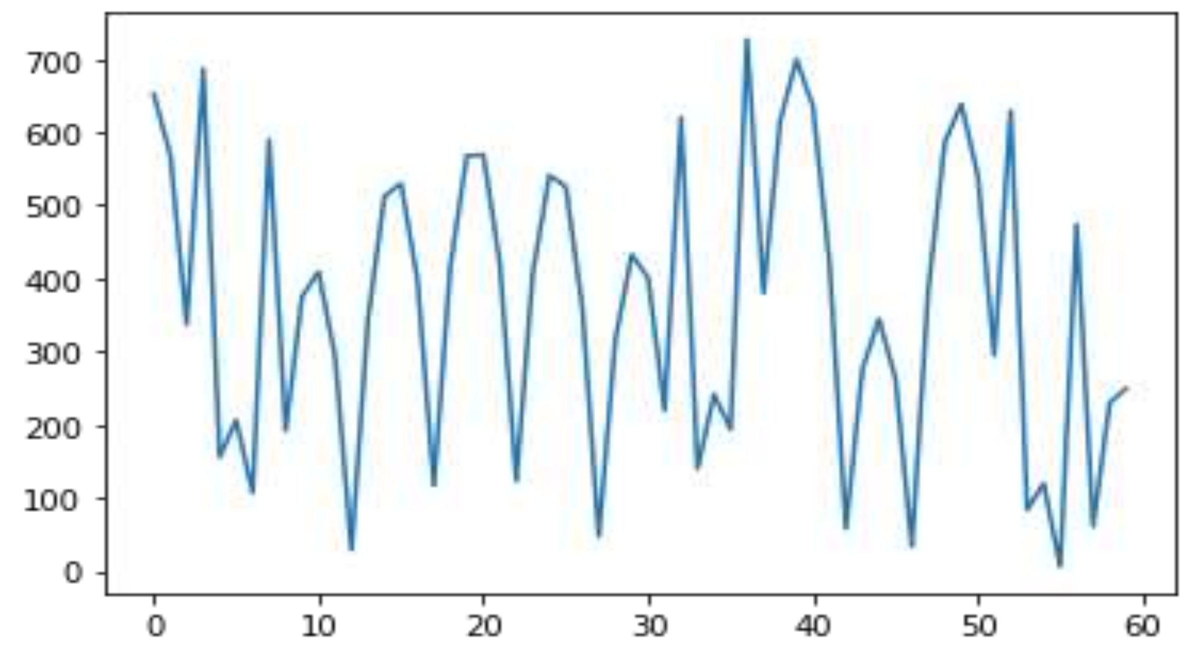
```
123.112060546875
406.88720703125
541.18017578125
525.99169921875
361.3173828125
47.16552734375
314.52880859375
432.41015625
400.80908203125
219.7275390625
620.16015625
140.1142578125
241.583984375
193.5693359375
727.07568359375
380.09521484375
614.63671875
699.69677734375
635.2724609375
421.36572265625
57.978515625
276.10693359375
344.7490234375
263.9140625
33.595703125
384.796875
586.5146484375
638.7490234375
541.498046875
294.7734375
629.5517578125
83.861328125
119.6875
6.0234375
473.8828125
61.26171875
230.1572265625
249.55859375
```

In [ ]:

```
from matplotlib import pyplot as pt
import numpy as np
x=[]
for i in range(0,60):
  x.append(i)
pt.plot(x,ee)
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cbdeced0>]
```

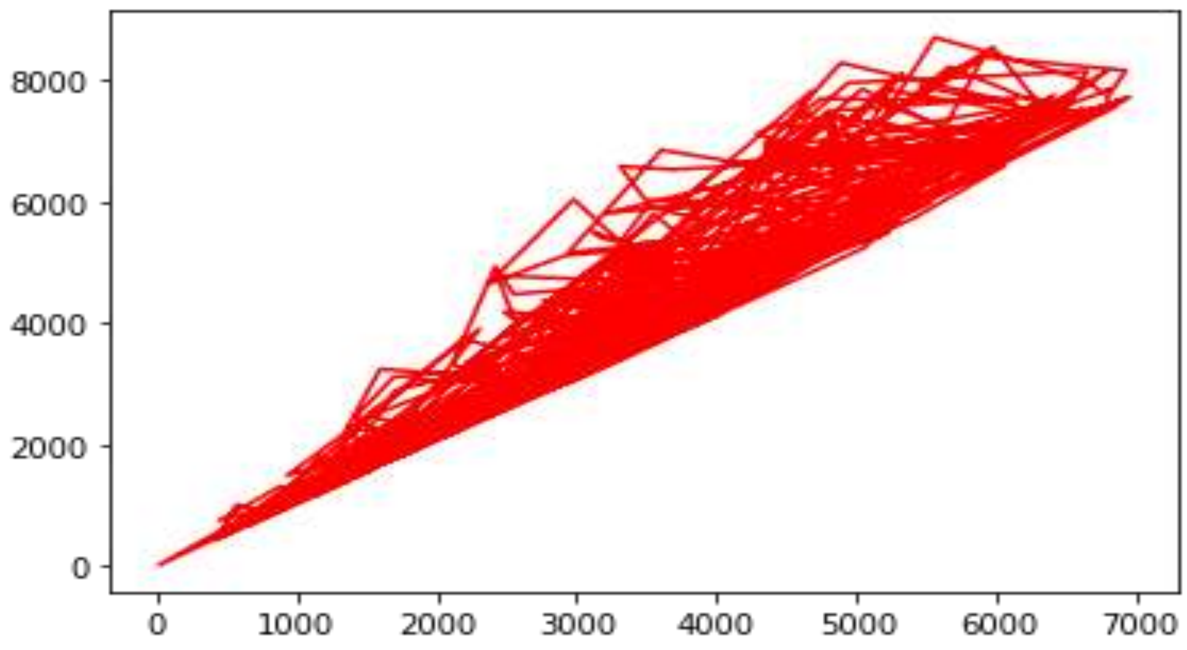


In [ ]:

```
from matplotlib import pyplot as plt
plt.plot(x13,y,'r')
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fc9cbd4da50>]
```



4.2. OVERVIEW TECHNOLOGY

In our program we used python 3 programming language. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

5. TESTING

We have applied various equations on our dataset and tried numerous times comparing its mean error.

And at last, we have achieved the best run out of all the other runs and have selected the same equation as the final one for our equation

6. RESULTS

Independent(x)	**Dependent(y)**	**Mean Square Error**
X1(season)	Y(count)	19
X2(yr)	y(count)	2
X3(mnth)	y(count)	6
X4(holiday)	y(count)	4
X5(weekday)	y(count)	18
X6(workingday)	y(count)	8
X7(weathersit)	y(count)	15
X8(temp)	y(count)	16.357688903808594
X9(atemp)	y(count)	2.771759033203125
X10(hum)	y(count)	4.441112518310547
X11(windspeed)	y(count)	13.493705749511719
X12(casual)	y(count)	17.428466796875
X13(registered)	y(count)	6.0234375

7. CONCLUSIONS

Bicycle sharing systems can be the new boom in India, with use of various prediction models the ease of operations will be increased.

- Maintenance activities for bikes should be done at night due to low usage of bikes during the night time. Removing some bikes from the streets at night time will not cause trouble for the customers.
- Offer dynamic pricing depending upon the seasonal variations to promote bike usage during fall and winter seasons.
- Rental hours should be fixed (i.e., 7-9am and 5-7pm) while planning for extra bikes to stations.
- Most of the rentals are for commuting (travel some distance between one's home and place of work on a regular basis) to workplaces and colleges on a daily basis. So, Cabs should launch more stations near these landmarks to reach out to their main customers.

8. FUTURE SCOPE

We can use this testing for future

- We can easily predict on which travelling conditions the bike sharing is on demand.
- We can also find out which feature variable is affecting the bike sharing.
- Just by collecting or upload the data we can find the result using this program.

BIBLIOGRAPHY

[1] Fanaee-T, Hadi, and Gama, Joao, "Event labeling combining ensemble detectors and background knowledge", Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg, doi:10.1007/s13748-013-0040-3.

@article{
year={2013},
issn={2192-6352},
journal={Progress in Artificial Intelligence},
doi={10.1007/s13748-013-0040-3},
title={Event labeling combining ensemble detectors and background knowledge},
url={http://dx.doi.org/10.1007/s13748-013-0040-3},
publisher={Springer Berlin Heidelberg},
keywords={Event labeling; Event detection; Ensemble learning; Background knowledge},
author={Fanaee-T, Hadi and Gama, Joao},

```
	pages={1-15}
}
```