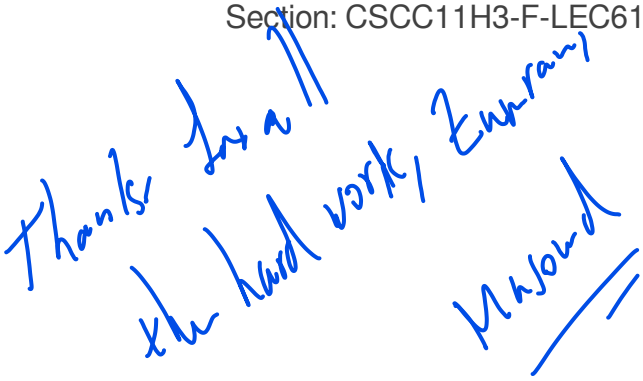
(2021 Fall) CSCC11H3 F 20219:Introduction to Machine Learning and Data Mining
Untimed Take-home Final Exam

Score: 0 / 100

Email: zumran.nain@mail.utoronto.ca

Name: Zumran Nain

Section: CSCC11H3-F-LEC61-20219, CSCC11H3-F-TUT0001-20219



Report

/100 points

Please submit a pdf version of your report here (HTML extension is also acceptable if you did not have any means to generate the pdf version).

Welcome to Jupyter!

Report

Analysis of the Stock Market Fluctuations, Anomalies and Fear Index Using Data-driven Methodologies

In this report, analysis of Stock Market Fluctuations Anomalies and Fear Index using Data-driven tecnologies and models will be used on two datasets from a stock company VIX. On the first dataset containing the monthly values of VIX from 1990 to 2019, the Gaussian Mixtures Model will be used with the genetic algorithm to analyze the regimes of the mixture distribution, and to analyze why the regime-switching phenomenon occurs in the stock market. In the second dataset containing same dataset as the first, but also with the additional column of monthly EMV trackers, the mechanisms underlying the sudden volatility changes in the stock market will be analyzed. In general, for the stock market, we can see the regime-switching phenomenon take place, since there are many states that can be achieved in a rapidly fluctutating stock market. There are also many mechanisms underlying sudden volality changes in stock markets such as political instability, Infectious Disease, Macroeconomics and so forth.

Introduction

There will be two datasets analyzed from stock company VIX in this report. The first dataset contains the monthly stock values of VIX from 1990 to 2019, and the second dataset contains the same data as the first but with the additional column of monthly EMV trackers(45 of them). We want to analyze the historical stock price of this company, and see if it is possible to forecast the future stock prices of this company using the given historical data, as well as various machine learning modelling and fitting techniques. We want to see if it is even possible to forecast the stock market fear index based on past and current trends, to see if its possible to use past data to construct a predictive model using machine learning techniques, and lastly to see if the behaviour in stock markets is entirely predictable, or if there are additional factors which might have a great influence on the behaviour that one observes in stock markets.

It is generally not possible to forecast the stock market fear index with perfect accuracy. Stock prices, are, by nature, uncertain, and due to the monetary policies and uncertain events that take place on our world, we cannot forecast the stock market tomorrow with certainty. There are some methods we can use to better predict stock prices, such as the earnings per share, price-to-earnings ratio, price-to-earnings growth ratio. Machine learning techniques can also be used to predict the stock price on certain hours of the stock market. In general, ratios can be used to tell an investor more about a stock price, than he/she can learn by simply researching financial statements. Financial ratios can help investors determine the strengths and weaknesses of a company's stock or an entire industry. Generally speaking companies with higher ratios are favourable. However, as stated before, we cannot predict the performance of with certainty even with these calculations. There will always be some sort of volatility due to the uncertainty of global events and major changes in the company.

It is possible to gauge some of the market movement using past data. We can construct a predictive model using past data to forecast future market movement based on supply and demand. With technical analysis and several technical tools to analyze the past data, we can construct a predictive model to forecast future stock market prices. However, no machine learning model is perfect, and depending on the model we choose, we will analyze some of the benefits and fallacies associated with each model.

As mentioned before, there are many external factors that are out of the reach of researchers that can greatly influence the behaviour of the volatility index seen in stock markets. Unpredictability, volatility and major global and industrial events are examples of some of those.

As you will see in this report, it is generally not possible to omit these external factors, and there will be major consequences by simplfiying the problem and omitting these factors

- Load the file MVIX.pkl containing monthly values of VIX from January 1990 - December 2019, and use the derivative-free optimization GA solver to learn the parameters of MoG. The optimal number will be one of k = 1,2,3,4,5.

```
pip install --upgrade matplotlib
```

```
Requirement already satisfied: matplotlib in
/srv/conda/envs/notebook/lib/python3.6/site-packages (3.3.4)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!
=2.1.6,>=2.0.3 in /srv/conda/envs/notebook/lib/python3.6/site-packages
(from matplotlib) (2.4.7)
Requirement already satisfied: pillow>=6.2.0 in
/srv/conda/envs/notebook/lib/python3.6/site-packages (from matplotlib)
(8.3.2)
Requirement already satisfied: python-dateutil>=2.1 in
/srv/conda/envs/notebook/lib/python3.6/site-packages (from matplotlib)
(2.8.2)
Requirement already satisfied: numpy>=1.15 in
/srv/conda/envs/notebook/lib/python3.6/site-packages (from matplotlib)
(1.19.5)
Requirement already satisfied: cycler>=0.10 in
/srv/conda/envs/notebook/lib/python3.6/site-packages (from matplotlib)
(0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/srv/conda/envs/notebook/lib/python3.6/site-packages (from matplotlib)
(1.3.1)
Requirement already satisfied: six>=1.5 in
/srv/conda/envs/notebook/lib/python3.6/site-packages (from python-
dateutil>=2.1->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

```
import sys
print (sys.version)
```

```
3.6.13 | packaged by conda-forge | (default, Sep 23 2021, 07:56:31)
[GCC 9.4.0]
```

```
pip install func-timeout
```

```
Collecting func-timeout
  Downloading func_timeout-4.3.5.tar.gz (44 kB)
eout
  Building wheel for func-timeout (setup.py) ... eout:
filename=func_timeout-4.3.5-py3-none-any.whl size=15095
sha256=7df2eb7f428106cd10589855dc81bf24a0f54233f8d68a84daefe863248411b
9
  Stored in directory:
```

```
/home/jovyan/.cache/pip/wheels/f5/f7/77/59440b4bf3fb41755539a8891b0e01
164c4ff597ed2c1742fb
Successfully built func-timeout
Installing collected packages: func-timeout
Successfully installed func-timeout-4.3.5
Note: you may need to restart the kernel to use updated packages.
```

```
'''
An easy implementation of genetic-algorithm (GA) to solve continuous
and combinatorial optimization problems with real, integer, and mixed
variables in Python

'''

#########################################################################
#########
#########################################################################
#########
#########################################################################
#########

import numpy as np
import sys
import time
from func_timeout import func_timeout, FunctionTimedOut
#import matplotlib.pyplot as plt

#########################################################################
#########
#########################################################################
#########
#########################################################################
#########

class GeneticAlgorithm():

    '''  Genetic Algorithm (Elitist version) for Python

    An implementation of elitist genetic algorithm for solving
problems with
    continuous, integers, or mixed variables.



    Implementation and output:

        methods:
                run(): implements the genetic algorithm

        outputs:
```

```
                output_dict:  a dictionary including the best set of
variables
            found and the value of the given function associated to
it.
            {'variable': , 'function': }

                report: a list including the record of the progress of
the
                algorithm over iterations

    '''
    ###########################################################

    def __init__(self, function, dimension, variable_type='bool', \
                 variable_boundaries=None,\
                 variable_type_mixed=None, \
                 function_timeout=10,\
                 algorithm_parameters={'max_num_iteration': None,\
                                       'population_size':100,\
                                       'mutation_probability':0.1,\
                                       'elit_ratio': 0.01,\
                                       'crossover_probability': 0.5,\
                                       'parents_portion': 0.3,\
                                       'crossover_type':'uniform',\

'max_iteration_without_improv':None},\
                     convergence_curve=True,\
                         progress_bar=True):


        '''
        @param function <Callable> - the given objective function to
be minimized
        NOTE: This implementation minimizes the given objective
function.
        (For maximization multiply function by a negative sign: the
absolute
        value of the output would be the actual objective function)

        @param dimension <integer> - the number of decision variables

        @param variable_type <string> - 'bool' if all variables are
Boolean;
        'int' if all variables are integer; and 'real' if all
variables are
        real value or continuous (for mixed type see @param
variable_type_mixed)

        @param variable_boundaries <numpy array/None> - Default None;
leave it
```

```
        None if variable_type is 'bool'; otherwise provide an array of
tuples
        of length two as boundaries for each variable;
        the length of the array must be equal dimension. For example,
        np.array([0,100],[0,200]) determines lower boundary 0 and
upper boundary 100 for first
        and upper boundary 200 for second variable where dimension is
2.

        @param variable_type_mixed <numpy array/None> - Default None;
leave it
        None if all variables have the same type; otherwise this can
be used to
        specify the type of each variable separately. For example if
the first
        variable is integer but the second one is real the input is:
        np.array(['int'],['real']). NOTE: it does not accept 'bool'.
If variable
        type is Boolean use 'int' and provide a boundary as [0,1]
        in variable_boundaries. Also if variable_type_mixed is
applied,
        variable_boundaries has to be defined.

        @param function_timeout <float> - if the given function does
not provide
        output before function_timeout (unit is seconds) the algorithm
raise error.
        For example, when there is an infinite loop in the given
function.

        @param algorithm_parameters:
            @ max_num_iteration <int> - stoping criteria of the
genetic algorithm (GA)
            @ population_size <int>
            @ mutation_probability <float in [0,1]>
            @ elit_ration <float in [0,1]>
            @ crossover_probability <float in [0,1]>
            @ parents_portion <float in [0,1]>
            @ crossover_type <string> - Default is 'uniform';
'one_point' or
            'two_point' are other options
            @ max_iteration_without_improv <int> - maximum number of
            successive iterations without improvement. If None it is
ineffective

        @param convergence_curve <True/False> - Plot the convergence
curve or not
        Default is True.
        @progress_bar <True/False> - Show progress bar or not. Default
is True.
```

```
        '''
        self.__name__ = GeneticAlgorithm
        #############################################################
        # input function
        assert (callable(function)),"function must be callable"

        self.f = function
        #############################################################
        #dimension

        self.dim = int(dimension)

        #############################################################
        # input variable type

        assert(variable_type=='bool' or variable_type=='int' or\
               variable_type=='real'), \
               "\n variable_type must be 'bool', 'int', or 'real'"

       #############################################################
        # input variables' type (MIXED)

        if variable_type_mixed is None:

            if variable_type=='real':
                self.var_type=np.array([['real']]*self.dim)
            else:
                self.var_type=np.array([['int']]*self.dim)


        else:
            assert (type(variable_type_mixed).__module__=='numpy'),\
            "\n variable_type must be numpy array"
            assert (len(variable_type_mixed) == self.dim), \
            "\n variable_type must have a length equal dimension."


            for i in variable_type_mixed:
                assert (i=='real' or i=='int'),\
                "\n variable_type_mixed is either 'int' or 'real' "+\
                "ex:['int','real','real']"+\
                "\n for 'boolean' use 'int' and specify boundary as
[0,1]"


            self.var_type=variable_type_mixed

        #############################################################
```

```
        # input variables' boundaries


        if variable_type!='bool' or
type(variable_type_mixed).__module__=='numpy':

            assert (type(variable_boundaries).__module__=='numpy'),\
            "\n variable_boundaries must be numpy array"

            assert (len(variable_boundaries)==self.dim),\
            "\n variable_boundaries must have a length equal
dimension"


            for i in variable_boundaries:
                assert (len(i) == 2), \
                "\n boundary for each variable must be a tuple of
length two."
                assert(i[0]<=i[1]),\
                "\n lower_boundaries must be smaller than
upper_boundaries [lower,upper]"
            self.var_bound = variable_boundaries
        else:
            self.var_bound = np.array([[0,1]]*self.dim)

        ##################################################################

        #Timeout
        self.funtimeout = float(function_timeout)

        ##################################################################

        #convergence_curve
        if convergence_curve==True:
            self.convergence_curve=True
        else:
            self.convergence_curve=False
        ##################################################################

        #progress_bar
        if progress_bar==True:
            self.progress_bar=True
        else:
            self.progress_bar=False

        ##################################################################
        ##################################################################

        # input algorithm's parameters
```

```
        self.param=algorithm_parameters

        self.pop_s=int(self.param['population_size'])

        assert (self.param['parents_portion']<=1\
                and self.param['parents_portion']>=0),\
        "parents_portion must be in range [0,1]"

        self.par_s = int(self.param['parents_portion']*self.pop_s)
        trl = self.pop_s-self.par_s
        if trl % 2 != 0:
            self.par_s+=1

        self.prob_mut=self.param['mutation_probability']

        assert (self.prob_mut<=1 and self.prob_mut>=0), \
        "mutation_probability must be in range [0,1]"


        self.prob_cross = self.param['crossover_probability']
        assert (self.prob_cross<=1 and self.prob_cross>=0), \
        "mutation_probability must be in range [0,1]"

        assert (self.param['elit_ratio']<=1 and
self.param['elit_ratio']>=0),\
        "elit_ratio must be in range [0,1]"

        trl=self.pop_s*self.param['elit_ratio']
        if trl<1 and self.param['elit_ratio']>0:
            self.num_elit=1
        else:
            self.num_elit=int(trl)

        assert(self.par_s>=self.num_elit), \
        "\n number of parents must be greater than number of elits"

        if self.param['max_num_iteration']==None:
            self.iterate=0
            for i in range (0,self.dim):
                if self.var_type[i]=='int':
                    self.iterate+=(self.var_bound[i][1]-
self.var_bound[i][0])*self.dim*(100/self.pop_s)
                else:
                    self.iterate+=(self.var_bound[i][1]-
self.var_bound[i][0])*50*(100/self.pop_s)
            self.iterate=int(self.iterate)
            if (self.iterate*self.pop_s)>10000000:
                self.iterate=10000000/self.pop_s
        else:
```

```
            self.iterate=int(self.param['max_num_iteration'])

        self.c_type=self.param['crossover_type']
        assert (self.c_type=='uniform' or self.c_type=='one_point' or\
                self.c_type=='two_point'),\
        "\n crossover_type must 'uniform', 'one_point', or 'two_point'
Enter string"


        self.stop_mniwi=False
        if self.param['max_iteration_without_improv']==None:
            self.mniwi=self.iterate+1
        else:
            self.mniwi=int(self.param['max_iteration_without_improv'])


        ###################################################################

    def run(self):


        ###################################################################
        # Initial Population

        self.integers = np.where(self.var_type=='int')
        self.reals = np.where(self.var_type=='real')



        pop = np.array([np.zeros(self.dim+1)]*self.pop_s)
        solo = np.zeros(self.dim+1)
        var = np.zeros(self.dim)

        for p in range(0,self.pop_s):

            for i in self.integers[0]:
                var[i] = np.random.randint(self.var_bound[i][0],\
                        self.var_bound[i][1]+1)
                solo[i] = var[i].copy()
            for i in self.reals[0]:
                var[i]=self.var_bound[i][0]+np.random.random()*\
                (self.var_bound[i][1]-self.var_bound[i][0])
                solo[i]=var[i].copy()


            obj = self.sim(var)
            solo[self.dim] = obj
            pop[p] = solo.copy()

        ###################################################################
```

```
        ###################################################################
        # Report
        self.report=[]
        self.test_obj=obj
        self.best_variable=var.copy()
        self.best_function=obj
        ###################################################################


        t=1
        counter=0
        while t<=self.iterate:

            if self.progress_bar==True:
                self.progress(t,self.iterate,status="GA is
running...")

##################################################################
            #Sort
            pop = pop[pop[:,self.dim].argsort()]



            if pop[0,self.dim]<self.best_function:
                counter=0
                self.best_function=pop[0,self.dim].copy()
                self.best_variable=pop[0,: self.dim].copy()
            else:
                counter+=1

##################################################################
            # Report

            self.report.append(pop[0,self.dim])


##################################################################

            # Normalizing objective function

            normobj=np.zeros(self.pop_s)

            minobj=pop[0,self.dim]
            if minobj<0:
                normobj=pop[:,self.dim]+abs(minobj)

            else:
                normobj=pop[:,self.dim].copy()
```

```
            maxnorm=np.amax(normobj)
            normobj=maxnorm-normobj+1


##############################################################
            # Calculate probability

            sum_normobj=np.sum(normobj)
            prob=np.zeros(self.pop_s)
            prob=normobj/sum_normobj
            cumprob=np.cumsum(prob)


##############################################################
            # Select parents
            par=np.array([np.zeros(self.dim+1)]*self.par_s)

            for k in range(0,self.num_elit):
                par[k]=pop[k].copy()
            for k in range(self.num_elit,self.par_s):
                index=np.searchsorted(cumprob,np.random.random())
                par[k]=pop[index].copy()

            ef_par_list=np.array([False]*self.par_s)
            par_count=0
            while par_count==0:
                for k in range(0,self.par_s):
                    if np.random.random()<=self.prob_cross:
                        ef_par_list[k]=True
                        par_count+=1

            ef_par=par[ef_par_list].copy()


##############################################################
            #New generation
            pop=np.array([np.zeros(self.dim+1)]*self.pop_s)

            for k in range(0,self.par_s):
                pop[k]=par[k].copy()

            for k in range(self.par_s, self.pop_s, 2):
                r1=np.random.randint(0,par_count)
                r2=np.random.randint(0,par_count)
                pvar1=ef_par[r1,: self.dim].copy()
                pvar2=ef_par[r2,: self.dim].copy()

                ch=self.cross(pvar1,pvar2,self.c_type)
                ch1=ch[0].copy()
```

```
                ch2=ch[1].copy()

                ch1=self.mut(ch1)
                ch2=self.mutmidle(ch2,pvar1,pvar2)
                solo[: self.dim]=ch1.copy()
                obj=self.sim(ch1)
                solo[self.dim]=obj
                pop[k]=solo.copy()
                solo[: self.dim]=ch2.copy()
                obj=self.sim(ch2)
                solo[self.dim]=obj
                pop[k+1]=solo.copy()
        ############################################################

            t+=1
            if counter > self.mniwi:
                pop = pop[pop[:,self.dim].argsort()]
                if pop[0,self.dim]>=self.best_function:
                    t=self.iterate
                    if self.progress_bar==True:
                        self.progress(t,self.iterate,status="GA is
running...")
                    time.sleep(2)
                    t+=1
                    self.stop_mniwi=True

        ############################################################
        #Sort
        pop = pop[pop[:,self.dim].argsort()]

        if pop[0,self.dim]<self.best_function:

            self.best_function=pop[0,self.dim].copy()
            self.best_variable=pop[0,: self.dim].copy()
        ############################################################
        # Report

        self.report.append(pop[0,self.dim])



        self.output_dict={'variable': self.best_variable, 'function':\
                          self.best_function}
        if self.progress_bar==True:
            show=' '*100
            sys.stdout.write('\r%s' % (show))
        sys.stdout.write('\r The best solution found:\n %s' %
(self.best_variable))
        sys.stdout.write('\n\n Objective function:\n %s\n' %
```

```
(self.best_function))
        sys.stdout.flush()
        re=np.array(self.report)
        if self.convergence_curve==True:
            plt.plot(re)
            plt.xlabel('Iteration')
            plt.ylabel('Objective function')
            plt.title('Genetic Algorithm')
            plt.show()

        if self.stop_mniwi==True:
            sys.stdout.write('\nWarning: GA is terminated due to
the'+\
                             ' maximum number of iterations without
improvement was met!')
#############################################################################
########
#############################################################################
########
    def cross(self,x,y,c_type):

        ofs1=x.copy()
        ofs2=y.copy()


        if c_type=='one_point':
            ran=np.random.randint(0,self.dim)
            for i in range(0,ran):
                ofs1[i]=y[i].copy()
                ofs2[i]=x[i].copy()

        if c_type=='two_point':

            ran1=np.random.randint(0,self.dim)
            ran2=np.random.randint(ran1,self.dim)

            for i in range(ran1,ran2):
                ofs1[i]=y[i].copy()
                ofs2[i]=x[i].copy()

        if c_type=='uniform':

            for i in range(0, self.dim):
                ran=np.random.random()
                if ran <0.5:
                    ofs1[i]=y[i].copy()
                    ofs2[i]=x[i].copy()

        return np.array([ofs1,ofs2])
#############################################################################
```

```
#########

    def mut(self,x):

        for i in self.integers[0]:
            ran=np.random.random()
            if ran < self.prob_mut:

                x[i]=np.random.randint(self.var_bound[i][0],\
                 self.var_bound[i][1]+1)



        for i in self.reals[0]:
            ran=np.random.random()
            if ran < self.prob_mut:

               x[i]=self.var_bound[i][0]+np.random.random()*\
                (self.var_bound[i][1]-self.var_bound[i][0])

        return x
#####################################################################
#########

    def mutmidle(self, x, p1, p2):
        for i in self.integers[0]:
            ran=np.random.random()
            if ran < self.prob_mut:
                if p1[i]<p2[i]:
                    x[i]=np.random.randint(p1[i],p2[i])
                elif p1[i]>p2[i]:
                    x[i]=np.random.randint(p2[i],p1[i])
                else:
                    x[i]=np.random.randint(self.var_bound[i][0],\
                     self.var_bound[i][1]+1)

        for i in self.reals[0]:
            ran=np.random.random()
            if ran < self.prob_mut:
                if p1[i]<p2[i]:
                    x[i]=p1[i]+np.random.random()*(p2[i]-p1[i])
                elif p1[i]>p2[i]:
                    x[i]=p2[i]+np.random.random()*(p1[i]-p2[i])
                else:
                    x[i]=self.var_bound[i][0]+np.random.random()*\
                    (self.var_bound[i][1]-self.var_bound[i][0])
        return x
#####################################################################
#########
```

```
    def evaluate(self):
        return self.f(self.temp)

#########################################################################
#########

    def sim(self,X):
        self.temp=X.copy()
        obj=None
        try:
            obj=func_timeout(self.funtimeout,self.evaluate)
        except FunctionTimedOut:
            print("given function is not applicable")
        assert (obj!=None), "After "+str(self.funtimeout)+" seconds
delay "+\
                "func_timeout: the given function does not provide any
output"
        return obj

#########################################################################
#########

    def progress(self, count, total, status=''):
        bar_len = 50
        filled_len = int(round(bar_len * count / float(total)))

        percents = round(100.0 * count / float(total), 1)
        bar = '|' * filled_len + '_' * (bar_len - filled_len)

        sys.stdout.write('\r%s %s%s %s' % (bar, percents, '%',
status))
        sys.stdout.flush()

#########################################################################
#########
#########################################################################
#########
```

```
import statistics
print('NormalDist' in dir(statistics))
```

```
False
```

```
import _pickle as pickle

import pandas as pd
import numpy as np
import random

import statistics


#import matplotlib.pyplot as plt
```

```
pwd
```

```
'/home/jovyan/binder'
```

```
with open('/home/jovyan/binder/MVIX.csv', "rb") as f:
```

```
    data=pd.read_csv(f)
    print(data)
```

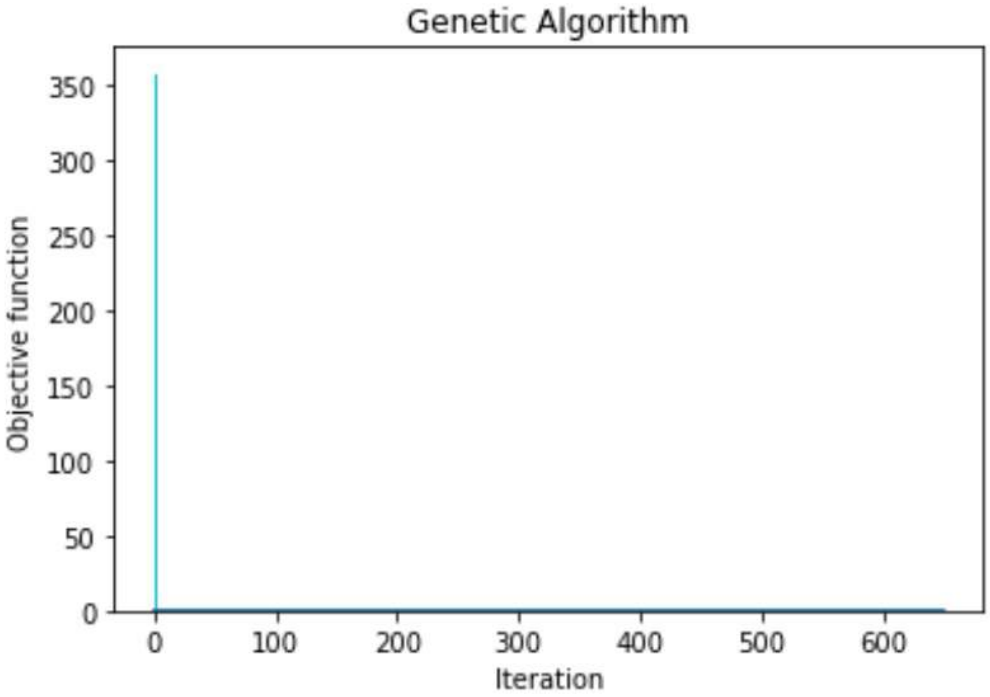
```
      25.360001
0     21.990000
1     19.730000
2     19.520000
3     17.370001
4     15.500000
..          ...
354   18.980000
355   16.240000
356   13.220000
357   12.620000
358   13.780000

[359 rows x 1 columns]
```

```
import matplotlib.pyplot as plt

from scipy.stats import norm



#import matplotlib

import numpy as np
import sys
import time
#from func_timeout import func_timeout, FunctionTimedOut

def obj_func(theta):

    pj_theta=[0 for i in range(20)]

    summ=0

    X_t=np.log(data[1:]/data[:-1])

    histo=plt.hist(X_t,20, histtype='step')

    count=histo[0]

    edges=histo[1]


    pj_hat=count/X_t.shape[0]
```

```
    for j in range(0, edges.size-1):
        x_min=edges[j]
        x_max=edges[j+1]

        for i in range(0, k):

                pj_theta[j]=0


                cdf2=norm.cdf(x_max, loc=theta[i][0],scale=theta[i][1])

                cdf1=norm.cdf(x_min,loc=theta[i][0],scale=theta[i][1])


                cdf=cdf2-cdf1


                pj_theta[j]+=theta[i][2]*cdf



        if pj_hat[j]>0 and pj_theta[i]>0:
            summ += pj_hat[j]*np.log(pj_theta[i]/pj_hat[j])

    z= -2 * X_t.shape[0] *  summ

    lamda=random.random()



    z+= lamda* abs(np.sum(theta[i][2])-1)

    theta=np.append(theta, lamda)


    return z

   # print("THIS IS Edges" ,edges)



k=3
model=GeneticAlgorithm(function=obj_func, dimension=3*k+1,
variable_type='real', variable_boundaries= np.array([[-5,5],[0,2],
[0,1]]))
model.run()

results=model.report
```

```
print(results)
```

```
 The best solution found:

 [2.57266513 0.01552495 0.94278404]

 Objective function:
 5.192288933022168e-06
```




```
[0.000996241518350457, 0.000996241518350457, 0.000996241518350457,
0.000996241518350457, 0.000996241518350457, 0.000996241518350457,
0.000996241518350457, 0.000996241518350457, 0.00048068634677367664,
0.00048068634677367664, 0.00048068634677367664,
0.00048068634677367664, 0.00048068634677367664,
0.00048068634677367664, 0.00048068634677367664,
0.00048068634677367664, 0.00048068634677367664, 0.0002194865680631799,
0.0002194865680631799, 0.0002194865680631799, 0.0002194865680631799,
0.0002194865680631799, 3.6605639243042756e-05, 3.6605639243042756e-05,
3.6605639243042756e-05, 3.6605639243042756e-05, 3.6605639243042756e-
05, 3.6605639243042756e-05, 3.6605639243042756e-05,
3.6605639243042756e-05, 3.6605639243042756e-05, 3.6605639243042756e-
05, 3.6605639243042756e-05, 3.6605639243042756e-05,
3.6605639243042756e-05, 3.6605639243042756e-05, 3.6605639243042756e-
05, 3.6605639243042756e-05, 3.6605639243042756e-05,
3.6605639243042756e-05, 3.6605639243042756e-05, 3.6605639243042756e-
05, 3.6605639243042756e-05, 3.6605639243042756e-05,
3.6605639243042756e-05, 3.6605639243042756e-05, 3.6605639243042756e-
```

05, 3.6605639243042756e-05, 3.6605639243042756e-05,
3.6605639243042756e-05, 3.6605639243042756e-05, 3.6605639243042756e-
05, 3.6605639243042756e-05, 3.6605639243042756e-05,
3.6605639243042756e-05, 3.6605639243042756e-05, 3.6605639243042756e-
05, 3.6605639243042756e-05, 3.6605639243042756e-05,
3.6605639243042756e-05, 2.4382246766674848e-05, 2.4382246766674848e-
05, 2.4382246766674848e-05, 2.4382246766674848e-05,
2.4382246766674848e-05, 2.4382246766674848e-05, 2.4382246766674848e-
05, 2.4382246766674848e-05, 2.4382246766674848e-05,
2.4382246766674848e-05, 2.4382246766674848e-05, 2.4382246766674848e-
05, 2.4382246766674848e-05, 2.4382246766674848e-05,
2.4382246766674848e-05, 2.4382246766674848e-05, 2.4382246766674848e-
05, 2.4382246766674848e-05, 2.4382246766674848e-05,
2.4382246766674848e-05, 2.4382246766674848e-05, 2.4382246766674848e-
05, 2.4382246766674848e-05, 2.4382246766674848e-05,
2.4382246766674848e-05, 2.4382246766674848e-05, 2.4382246766674848e-
05, 2.4382246766674848e-05, 2.4382246766674848e-05,
2.4382246766674848e-05, 2.4382246766674848e-05, 2.4382246766674848e-
05, 2.4382246766674848e-05, 2.4382246766674848e-05,
2.4382246766674848e-05, 2.4382246766674848e-05, 2.4382246766674848e-
05, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-

```
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
```

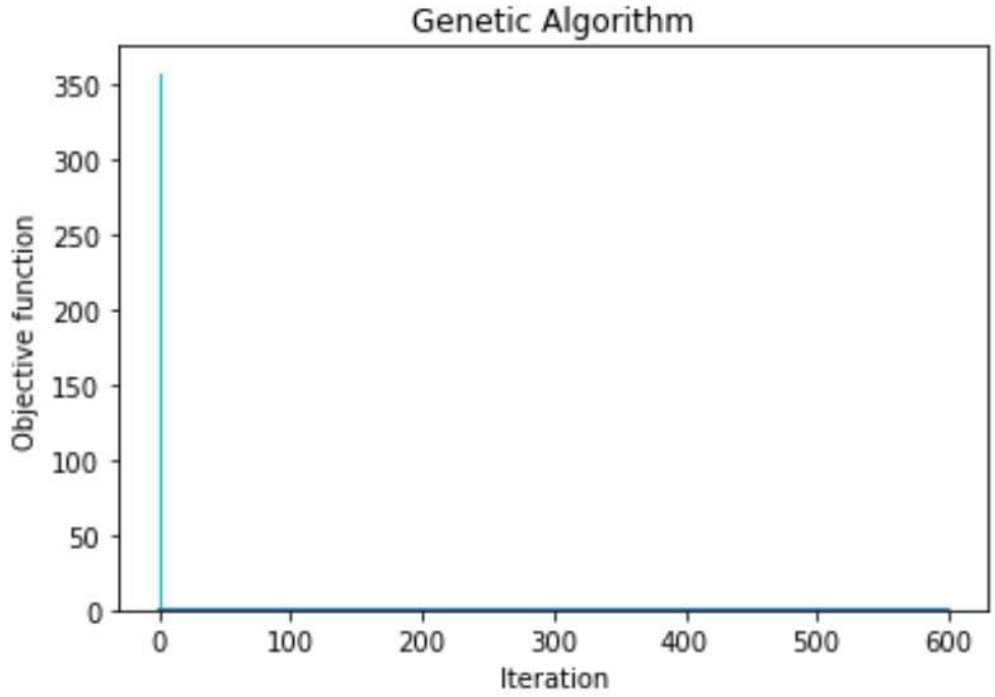
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-

06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-

```
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06, 5.192288933022168e-
06, 5.192288933022168e-06, 5.192288933022168e-06]
```

```
k=2
model=GeneticAlgorithm(function=obj_func, dimension=3*k+1,
variable_type='real', variable_boundaries= np.array([[-5,5],[0,2]]))
model.run()

results=model.report

print(results)
```

```
 The best solution found:

 [3.40907512 0.05988934]

 Objective function:
 2.979759533899338e-05
```




```
[0.11079234195883703, 0.08419138590168165, 0.08419138590168165,
0.08419138590168165, 0.008248787953246394, 0.008248787953246394,
0.008248787953246394, 0.008248787953246394, 0.008248787953246394,
0.008248787953246394, 0.008248787953246394, 0.008248787953246394,
0.008248787953246394, 0.008248787953246394, 0.0031033702350126186,
```

```
0.0031033702350126186, 0.0031033702350126186, 0.0031033702350126186,
0.0031033702350126186, 0.0031033702350126186, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 0.000233012305997104, 0.000233012305997104,
0.000233012305997104, 4.312211107042519e-05, 4.312211107042519e-05,
4.312211107042519e-05, 4.312211107042519e-05, 4.312211107042519e-05,
4.312211107042519e-05, 4.312211107042519e-05, 4.312211107042519e-05,
4.312211107042519e-05, 4.312211107042519e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
```

2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,

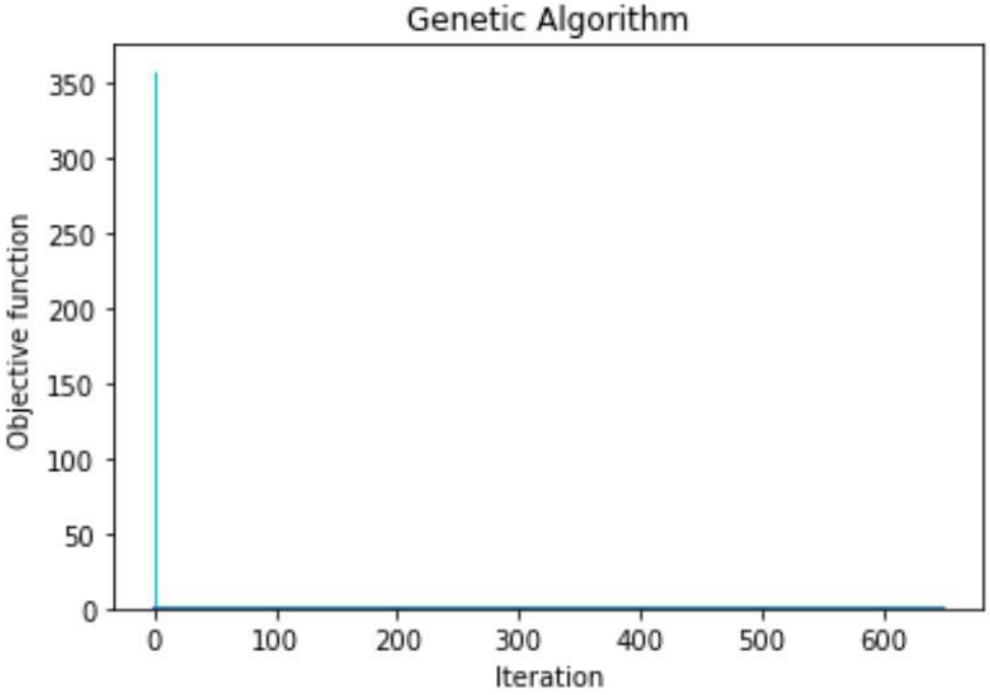
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,

```
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05, 2.979759533899338e-05, 2.979759533899338e-05,
2.979759533899338e-05]
```

```python
k=1
model=GeneticAlgorithm(function=obj_func, dimension=3*k+1,
variable_type='real', variable_boundaries= np.array([[-5,5],[0,2],
[0,1]]))
```

```
model.run()

results=model.report

print(results)
```

```
 The best solution found:

 [-3.19955169  0.25226691  0.99003637]

 Objective function:
 1.5633985098495285e-07
```




```
[0.008698219143746402, 0.008698219143746402, 0.0009083154130223656,
0.0009083154130223656, 0.0009083154130223656, 0.0009083154130223656,
0.0009083154130223656, 0.0009083154130223656, 0.0005616490853088584,
0.0005616490853088584, 0.0005616490853088584, 0.0005616490853088584,
0.0005616490853088584, 0.0005616490853088584, 0.0005616490853088584,
0.0005616490853088584, 0.0005616490853088584, 0.0005616490853088584,
0.0005616490853088584, 0.0005616490853088584, 0.0005616490853088584,
0.0005616490853088584, 0.0005616490853088584, 0.0005616490853088584,
0.0005616490853088584, 0.0005616490853088584, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
```

```
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 5.3792519237249404e-05, 5.3792519237249404e-
05, 5.3792519237249404e-05, 5.3792519237249404e-05,
5.3792519237249404e-05, 4.086226170348871e-05, 4.086226170348871e-05,
4.086226170348871e-05, 4.086226170348871e-05, 4.086226170348871e-05,
4.086226170348871e-05, 4.086226170348871e-05, 4.086226170348871e-05,
4.086226170348871e-05, 4.086226170348871e-05, 4.086226170348871e-05,
4.086226170348871e-05, 4.086226170348871e-05, 4.086226170348871e-05,
4.086226170348871e-05, 4.086226170348871e-05, 4.086226170348871e-05,
4.086226170348871e-05, 4.086226170348871e-05, 4.086226170348871e-05,
4.086226170348871e-05, 4.086226170348871e-05, 4.086226170348871e-05,
4.086226170348871e-05, 4.086226170348871e-05, 4.086226170348871e-05,
4.086226170348871e-05, 4.086226170348871e-05, 4.086226170348871e-05,
4.086226170348871e-05, 4.086226170348871e-05, 4.086226170348871e-05,
4.086226170348871e-05, 4.086226170348871e-05, 4.086226170348871e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
```

2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,

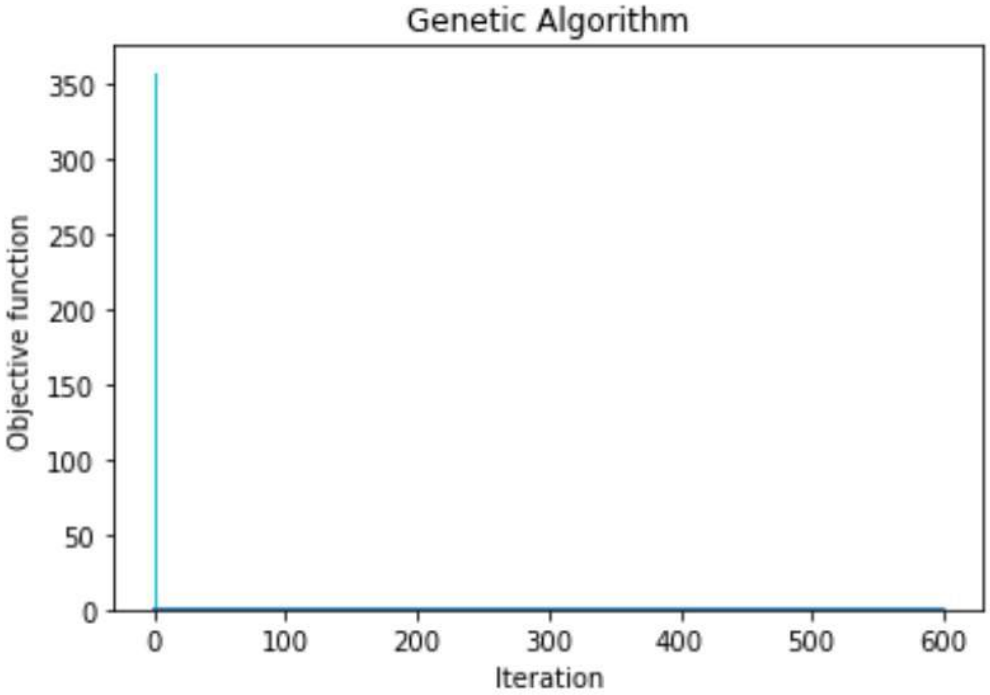
```
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
2.742962950328323e-05, 2.742962950328323e-05, 2.742962950328323e-05,
1.934772604221301e-05, 1.934772604221301e-05, 1.934772604221301e-05,
1.934772604221301e-05, 1.934772604221301e-05, 1.934772604221301e-05,
1.934772604221301e-05, 1.934772604221301e-05, 1.934772604221301e-05,
1.934772604221301e-05, 1.934772604221301e-05, 1.934772604221301e-05,
1.934772604221301e-05, 1.934772604221301e-05, 1.6499508957010782e-05,
1.6499508957010782e-05, 1.6499508957010782e-05, 1.6499508957010782e-
05, 1.6499508957010782e-05, 1.6499508957010782e-05,
1.6499508957010782e-05, 1.6499508957010782e-05, 1.6499508957010782e-
05, 1.6499508957010782e-05, 1.6499508957010782e-05,
1.6499508957010782e-05, 1.6499508957010782e-05, 1.6499508957010782e-
05, 1.6499508957010782e-05, 1.6499508957010782e-05,
1.6499508957010782e-05, 1.6499508957010782e-05, 1.6499508957010782e-
05, 1.6499508957010782e-05, 1.6499508957010782e-05,
1.6499508957010782e-05, 1.6499508957010782e-05, 1.6499508957010782e-
05, 1.6499508957010782e-05, 1.6499508957010782e-05,
1.6499508957010782e-05, 1.6499508957010782e-05, 1.6499508957010782e-
05, 1.6499508957010782e-05, 1.6499508957010782e-05,
1.6499508957010782e-05, 1.6499508957010782e-05, 1.6499508957010782e-
05, 1.6499508957010782e-05, 1.6499508957010782e-05,
1.6499508957010782e-05, 1.6499508957010782e-05, 1.6499508957010782e-
05, 1.6499508957010782e-05, 1.6499508957010782e-05,
1.6499508957010782e-05, 1.6499508957010782e-05, 1.6499508957010782e-
05, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
```

1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.1875184956415922e-06,
1.1875184956415922e-06, 1.1875184956415922e-06, 1.1875184956415922e-
06, 1.1875184956415922e-06, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,

```
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07, 1.5633985098495285e-
07, 1.5633985098495285e-07, 1.5633985098495285e-07,
1.5633985098495285e-07, 1.5633985098495285e-07]
```

```
k=4
model=GeneticAlgorithm(function=obj_func, dimension=3*k+1,
variable_type='real', variable_boundaries= np.array([[-5,5],[0,2]]))
model.run()

results=model.report

print(results)
```

```
 The best solution found:

 [-4.80164716  0.50775165]

 Objective function:
 1.7831174600824705e-06
```



```
[0.03896952722834673, 0.03896952722834673, 0.028704188750366774,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
```

06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-

06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-

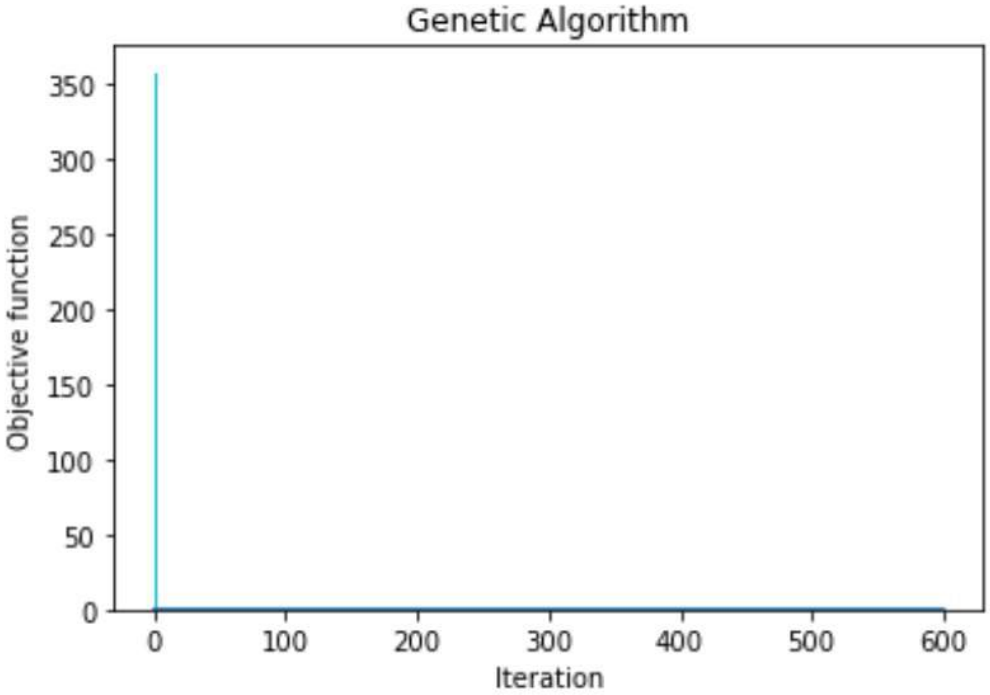
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-

06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-

```
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06, 1.7831174600824705e-06, 1.7831174600824705e-06,
1.7831174600824705e-06, 1.7831174600824705e-06, 1.7831174600824705e-
06]
```

```python
k=5
model=GeneticAlgorithm(function=obj_func, dimension=3*k+1,
variable_type='real', variable_boundaries= np.array([[-5,5],[0,2]]))
model.run()

results=model.report

print(results)
```

```
 The best solution found:

 [3.53789554 0.05826817]

 Objective function:
 6.94047113526585e-05
```



[0.004685066586257053, 0.004685066586257053, 0.004685066586257053,
0.004685066586257053, 0.004685066586257053, 0.004685066586257053,
0.004605348239597769, 0.004605348239597769, 0.004605348239597769,
0.004605348239597769, 0.004605348239597769, 0.004346847476084951,
0.004346847476084951, 0.004346847476084951, 0.004346847476084951,
0.004346847476084951, 0.004346847476084951, 0.004346847476084951,
0.004346847476084951, 0.004346847476084951, 0.004346847476084951,
0.004346847476084951, 0.004346847476084951, 0.004346847476084951,
0.004346847476084951, 0.004346847476084951, 0.004346847476084951,
0.004346847476084951, 0.004346847476084951, 0.004346847476084951,
0.004346847476084951, 0.002828344817163606 7, 0.002828344817163606 7,
0.002828344817163606 7, 0.002828344817163606 7, 0.000682418012397694 5,
0.000682418012397694 5, 0.000566945993838768, 0.000566945993838768,
0.000566945993838768, 0.000566945993838768, 0.000566945993838768,
0.000566945993838768, 0.000566945993838768, 0.000566945993838768,
0.000566945993838768, 0.000566945993838768, 0.000566945993838768,
0.000566945993838768, 0.000566945993838768, 0.000301879455585023 1,
0.000301879455585023 1, 0.000301879455585023 1, 0.000301879455585023 1,
0.000301879455585023 1, 0.000301879455585023 1, 0.000301879455585023 1,
0.000301879455585023 1, 0.000301879455585023 1, 0.000301879455585023 1,
0.000301879455585023 1, 0.000301879455585023 1, 0.000301879455585023 1,
0.000301879455585023 1, 0.000301879455585023 1, 0.000301879455585023 1,

```
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.0003018794555850231, 0.0003018794555850231, 0.0003018794555850231,
0.00016947123584598207, 0.00016947123584598207,
0.00016947123584598207, 0.00016947123584598207,
0.00016947123584598207, 0.00016947123584598207,
0.00016947123584598207, 0.00016947123584598207,
0.00016947123584598207, 0.00016947123584598207,
0.00016947123584598207, 0.00016947123584598207,
0.00016947123584598207, 0.00016947123584598207,
0.00016947123584598207, 0.00016947123584598207,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
```

0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,

0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823,
0.00012945738769005823, 0.00012945738769005823, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,

9.414144630194498e-05, 9.414144630194498e-05, 9.414144630194498e-05,
9.414144630194498e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,

```
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05,
6.94047113526585e-05, 6.94047113526585e-05, 6.94047113526585e-05]
```

Model Specification:

Load the file MVIX.pkl containing monthly values of VIX from January 1990 - December 2019, and use the derivative-free optimization GA solver to learn the parameters of MoG. The optimal number will be one of k = 1, 2, 3, 4, 5.

As seen the optimal value that minimizes the objective value k is k=3 and optimal objective value is 35-40. If the k value is too high(k=4 or 5), then overfitting occurs and we get an objective value of 50-60. We explored the issue of overfitting earlier in the course, when the model is too complex(i.e. has a high degree), then the function is too closely aligned to the data points, and so is unable to generalize well for new training data. Training error is low but test error is high. Similarly if the k value is too low(k=1 or 2), then underfitting occurs and then we get an objective value of 1000. Underfitting is the complete opposite, and it occurs when the model is too simple(i.e has a low degree), then function does not align at all with the data, leading to high training and test errors.

For each component of the mixture distribution, compute and report the estimates of its parameters (you may report them in a table).

For each regime of the mixture distribution we get the following estimate for the parameters.

For k=1: The best solution found:
[-3.19955169 0.25226691 0.99003637]

Objective function: 1.5633985098495285e-07

For k=2 The best solution found:
[3.40907512 0.05988934]

Objective function: 2.979759533899338e-05

For k=3, The best solution found:
[2.57266513 0.01552495 0.94278404]

Objective function: 5.192288933022168e-06

For k=4, The best solution found:
[-4.80164716 0.50775165]

Objective function: 1.7831174600824705e-06

For k=5: The best solution found:
[3.53789554 0.05826817]

Objective function: 6.94047113526585e-05

How did you decide the correct number of the components (regimes) of the mixture distribution, say denoted by k ?

In order to determine k, or the correct number of regimes of a mixture, we want to find the optimal k value that causes the training and test errors to be as low as possible without comprimising one over the other. It usually occurs at the midpoint of the degrees. For our problem for k=1,2,3,4,5, the midpoint occurs at k=3, and thus seemed like to be the most likely candidate for being the correct number of components of the mixture distribution. After determining the training and test errors, and comparing the values obtained by reach of the ks to the optimal value, the results showed that k=3 was the correct number of components of the mixture distribution.

Find the mcp at each observation and then compute a k ? × k ? probability transition matrix, where each element will be the probability by which the market could switch from its current regime to another one, or else just remain in the same regime.

where is it?

The probabilities of all the regimes should add up to one. Generally speaking if the regimes are very different, then the probability is lower. The more similar the regimes are, the higher the probability will swtich from one regime to the next.

Why do you believe the regime-switching phenomenon occurs in stock market?

Within the stock market, we can see that the characteristics of time series data, including means, variances, and model parameters change across regimes as observed in our data. Many of the regimes are temporary and recurring. We can see that there does seem to be cyclically occurring changes in the stock market, due to the periodic growth and recession in the economy, hence the explanation for the regime-switching phenomenon. It can explain periods of rapid growth in the stock markets or financial crises, economic downturns, hyperinflation because they follow the economic cycle.

The objective values for the regimes reaches the optimal value at k=3 after applying Lagrangian relaxation on the model. This is due to the underfitting-overfitting problem we have discussed earlier.

In general Regime switching models can match the tendency of stock markets to often change their behavior and the phenomenon that the new behavior of financial variables often persists for several periods after such a change.

```
#pip install -U scikit-learn scipy matplotlib

#pip install scikit-learn


import _pickle as pickle

import pandas as pd
import numpy as np

from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt
```

Fitting and Diagnostics

The results for each regression can be found in the appendices section.

Some fitting techniques used were the OLS, LASSO, Ridge Regression and Elastic Net Regression.

For ordinary least squares, some advantages include it is relatively fast and easy to compute, it is applicable in most machine learning problems. It can be effective if its assumptions are met(that is little to no outliers and linearity in data). For OLS, disadvantages include poor performance in the presence of outliers and so the weights may have to reduced for outliers in order to retain performance. It also performs poorly on non-linear datasets, becuase OLS assumes linearlity in data. It also had a strong tendency to overfit data, especially when the degree of the polynomial is too high. Lasso and Ridge regression will be a better choice in higher dimensional situtations since they are designed to avoid overfitting.

OLS did not perform well on this dataset, as it was not a linear set.

For Lasso, some advantages include the fact that it selects features through the process of feature selection and regularization of data models, thereby shrinking the coefficient to 0. Regularization basically consists of adding a penalty to the different parameters of the machine learning model to reduce the freedom of the model. Lasso tries to minimize the cost of regression, by choosing only those features that are useful and discarding redundant ones. As a result, it also tends to avoid overfitting.

Some disadvantages include the fact that selected features will be highly biased, and that prediction performance can be worse than Ridge Regression. Additionally if the dimension

p is large to the extent that n<<p for n features, LASSO can only select at most n features. After this point, the model will saturate, which is the state that the neuron predominantly outputs values close to the asymptotic ends of the bounded activation function.

Interestingly, Lasso can be applied even when the number of features is larger than the number of data points.

Lasso regression did a decent job at avoiding overfitting, however, it still had saturation issues since it did not match the general trend of the data. This was because we had many columns in our dataset.

Smilarly for Ridge regession, it avoid overfitting of the model through feature selection and regularization, but it does not have the problem of selecting more than n predictors when n<<p, whereas LASSO saturates when n<<p.

Disadvatanges of Ridge: It includes all the predictors in the final model, is unable to perform feature selection and it trades variance for bias.

Ridge regression did a much better job than Lasso regression, but it still had issues with feature selection

For Elastic net regression,it doesn't have the problem of selecting more than n predictors when n<<p, whereas LASSO saturates when n<<p. It essentially takes the convex combination of Lasso and Ridge to get the benefits of both models.

However, it is computationally much more expensive than both LASSO and Ridge and should only be used for much higher dimensional models.

Overall the best regressor for this model was the Elastic Net, as the model had a large number of columns(46), however it is computationally expensive. It also had problems with regressing the dataset due to the nature of the volatility index dataset.

Can you provide a coherent picture of mechanisms underlying the sudden volatility changes in stock markets?

Volatility is defined as the range of a price change over a given period of time. A stock has low volatility when its price stays relatively stable, whereas one with a high volatility has a price that move erratically for a given time period, and hits highs and lows rather quickly.

As seen in our graphs from the machine learning models, the computed EMV values varied a lot through the 382 various time periods from 1990 to 2019. It depends a lot on which 45 trackers that were influencing the volatility index at that time, and the movement in the stock market can be quite erratic at times. As a result, our interpolated line(in black) using the predicted values seemed to underfit for every single model, even the Elastic regression. There are overall many factors that can cause volaltility changes in stock markets(in our case 45), and it can be difficult to model them in a machine learning problem.

Forecasting

In matplotlib and sklearn, forecasting can be done by calling the prediction function on the model and test x data, after calling train_split to retrieve train_X, train_Y, test_X, and test_y. Afterwards I plotted the line that predicted our data. We need forecasting to explore the relationship between variables in the current dataset, and in order to be able to predict new values in the dataset. Forecasting is valuable in the context of VIX because it gives investors the ability to make informed investment decisions and develop data-driven strategies based on the forecasted volatility in the near future.

Forecasting the VIX can impact how the investors buy and trade options since the determined level of the VIX effects options premiums and prices

Discussion

In general, it was quite difficult to regress the model perfectly. Due to the nature of the stock market volatility index and the erratic nature of stock prices in general, the graphs were quite difficult to be fitted accurately. Even on the Elastic Regression algorithm, the line seemed to be underfitting, indicated that the presence of many trackers and external influences on the EMV were making it difficult for the machine learning algorithm to predict.

Bibliography:

Boyte-White, C. (2021, December 7). Volatility from the investor's point of View. Investopedia. Retrieved December 22, 2021, from https://www.investopedia.com/ask/answers/010915/volatility-good-thing-or-bad-thing-investors-point-view-and-why.asp

Gupta, S. (2020, June 23). Pros and cons of various Classification ML algorithms. Medium. Retrieved December 22, 2021, from https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6

Appendices

```
A=0.001


def OLS_Regression(train_X, train_Y, test_X, test_Y):

    reg=LinearRegression()

    reg.fit(train_X, train_Y)
```

```
    pred_Y=reg.predict(test_X)


    for tracker in range(45):

        plt.scatter(test_X, [y for y in test_Y])

        plt.plot(test_X,  [y for y in  pred_Y], color='black')

        plt.title("OLS Rgession" )

        plt.xlabel("Time ")

        plt.ylabel("EMV value")



        plt.show()
with open(r'/home/jovyan/binder/MEMV.csv', "rb") as f:


    df=pd.read_csv(r'/home/jovyan/binder/MEMV.csv')
    df
```

```
df
```

```
        Date      EMV  Political Uncertainty Tracker  \
0    1990-01  17.8882                          7.9697
1    1990-02  22.1500                          9.6857
2    1990-03  17.6410                          8.7400
3    1990-04  15.9283                          6.8080
4    1990-05  12.1001                          6.0545
..       ...      ...                             ...
377  2021-06  21.9521                         10.0916
378  2021-07  16.6235                          7.2485
379  2021-08  19.6411                          9.0514
380  2021-09  23.4802                         12.5473
381  2021-10  19.0222                          9.3300

     Infectious Disease EMV Tracker  \
0                            0.2411
1                            0.7699
2                            0.2673
3                            0.1493
4                            0.2521
..                              ...
377                         12.5927
378                         10.0712
379                         11.0266
380                         30.9127
```

```
381                         11.0963

     Macroeconomic News and Outlook EMV Tracker  \
0                                       13.7416
1                                       16.4644
2                                       12.0279
3                                       11.3987
4                                        9.2011
..                                          ...
377                                     14.2944
378                                     11.6486
379                                     12.4049
380                                     18.0747
381                                     13.7911

     Macro – Broad Quantity Indicators EMV Tracker  \
0                                          4.5323
1                                          3.7312
2                                          3.1184
3                                          2.8870
4                                          2.6469
..                                            ...
377                                        3.4034
378                                        3.5188
379                                        3.2735
380                                        7.2636
381                                        4.4385

     Macro – Inflation EMV Indicator  Macro – Interest Rates EMV
Tracker  \
0                              6.7020
6.5574
1                              8.5876
8.5283
2                              5.6576
5.2121
3                              4.0816
4.5794
4                              4.1594
4.4955
..                                ...                                .
..
377                            6.1262
4.4245
378                            4.6109
2.7908
379                            4.3073
2.2398
380                            4.3920
5.7433
```

```
381                          6.3407
4.4385

     Macro — Other Financial Indicators EMV Tracker  \
0                                                0.6268
1                                                0.9476
2                                                0.3118
3                                                0.4978
4                                                0.6302
..                                                  ...
377                                              0.8509
378                                              0.2427
379                                              0.6892
380                                              0.1689
381                                              0.0000

     Macro — Labor Markets EMV Tracker  ...  \
0                                3.1823  ...
1                                3.6127  ...
2                                2.3610  ...
3                                2.9866  ...
4                                2.7309  ...
..                                  ...  ...
377                              6.2963  ...
378                              4.6109  ...
379                              5.5133  ...
380                             10.3042  ...
381                              5.0726  ...

     National Security Policy EMV Tracker   \
0                                   2.0251
1                                   1.7175
2                                   1.6928
3                                   1.0453
4                                   0.8823
..                                     ...
377                                 2.2122
378                                 1.8201
379                                 1.2060
380                                 2.5338
381                                 2.2193

     Government-Sponsored Enterprises EMV Tracker  Trade Policy EMV
Tracker  \
0                                         0.6268
0.4339
1                                         0.8291
0.3553
2                                         0.6682
0.4009
```

```
3                                          0.7466
0.3484
4                                          0.6722
0.3781
..                                            ...
...
377                                        0.6807
0.1702
378                                        0.2427
0.0000
379                                        0.1723
0.3446
380                                        0.5068
0.5068
381                                        0.4756
0.3170
```

	Healthcare Policy EMV Tracker	Food and Drug Policy EMV Tracker	\
0	0.3857	0.1929	
1	0.3553	0.1777	
2	0.2227	0.0891	
3	0.2489	0.1493	
4	0.2941	0.3361	
..	...	...	
377	2.3824	0.5105	
378	0.7280	0.0000	
379	1.2060	0.5169	
380	0.6757	0.0000	
381	0.6341	0.0000	

	Transportation, Infrastructure, and Public Utilities EMV Tracker	\
0	0.3375	
1	0.1777	
2	0.2227	

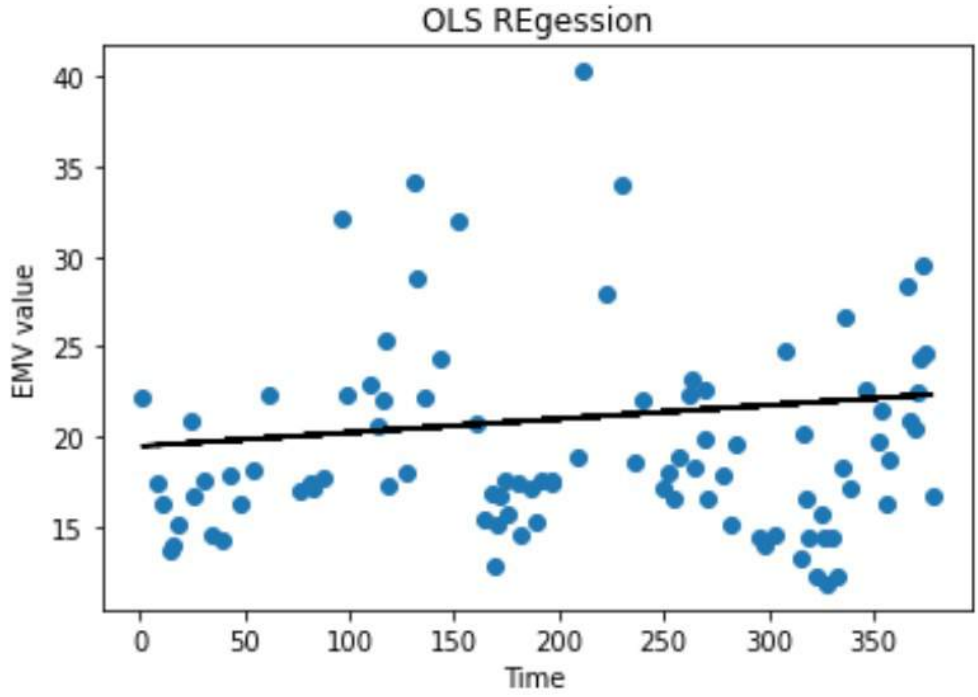
```
3                                                0.0996

4                                                0.0420

..                                                  ...

377                                              0.3403

378                                              0.4854

379                                              0.1723

380                                              0.3378

381                                              0.0000


     Elections and Political Governance EMV Tracker  \
0                                                0.5304
1                                                0.4146
2                                                0.3564
3                                                0.2987
4                                                0.2521
..                                                  ...
377                                              1.1912
378                                              0.2427
379                                              0.5169
380                                              0.0000
381                                              0.6341

     Agricultural Policy EMV Tracker  Petroleum Markets EMV Tracker
VIX
0                                0.0482                        3.567993
21.990000
1                                0.0000                        4.441847
19.730000
2                                0.0000                        3.073809
19.520000
3                                0.0000                        2.837237
17.370001
4                                0.0420                        1.554521
15.500000
..                                  ...                             ...
...
377                              0.0000                        4.254290
18.240000
378                              0.0000                        2.669466
16.480000
```

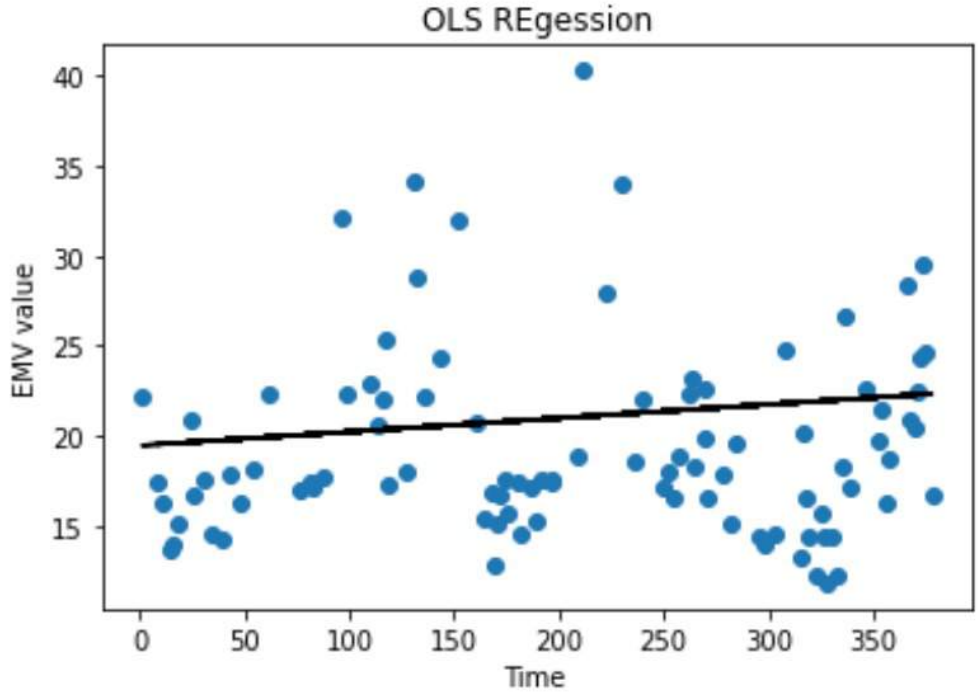
```
379                             0.0000                          4.479545
23.139999
380                             0.0000                          6.587960
16.260000
381                             0.0000                          4.914065
27.190001

[382 rows x 47 columns]
```

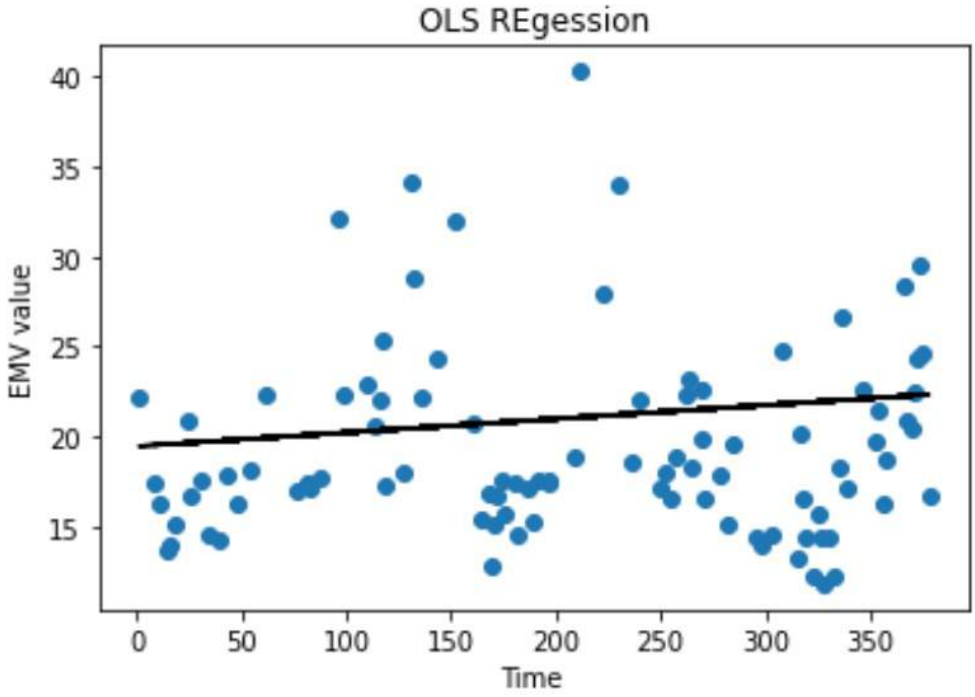
```
y=df["EMV"]

a_list = list(range(0, 382))

x=np.array(a_list)



train_X, test_X, train_Y, test_Y = train_test_split(x.reshape(-1,1),y)

test_Y
```

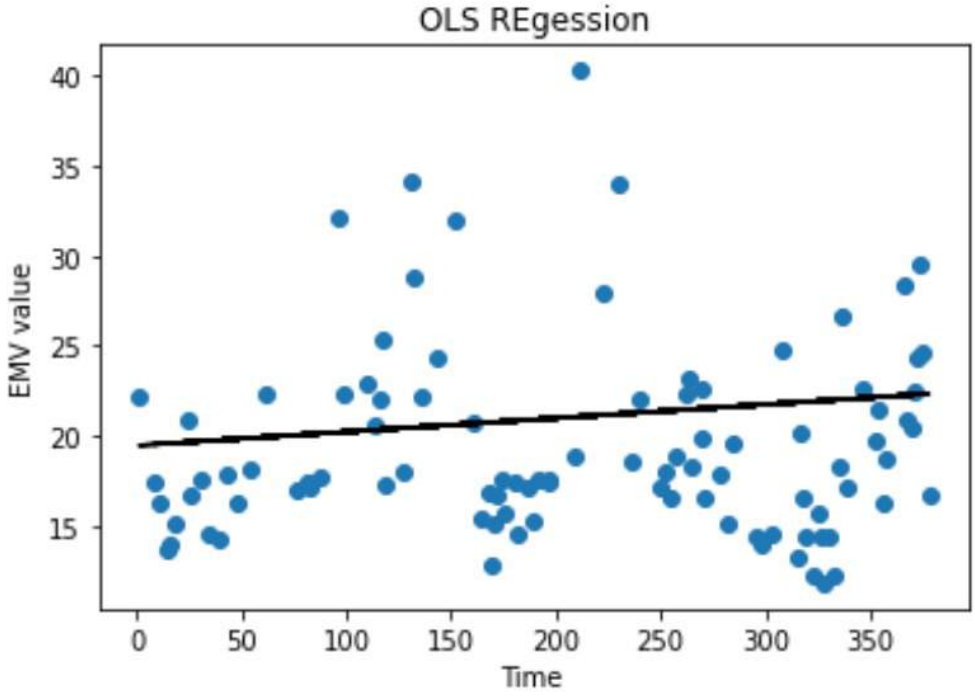
```
373    24.2737
230    34.0597
240    22.0811
278    17.9135
222    27.8947
       ...
271    16.6146
81     17.3695
8      17.4587
197    17.3766
370    20.4961
Name: EMV, Length: 96, dtype: float64
```

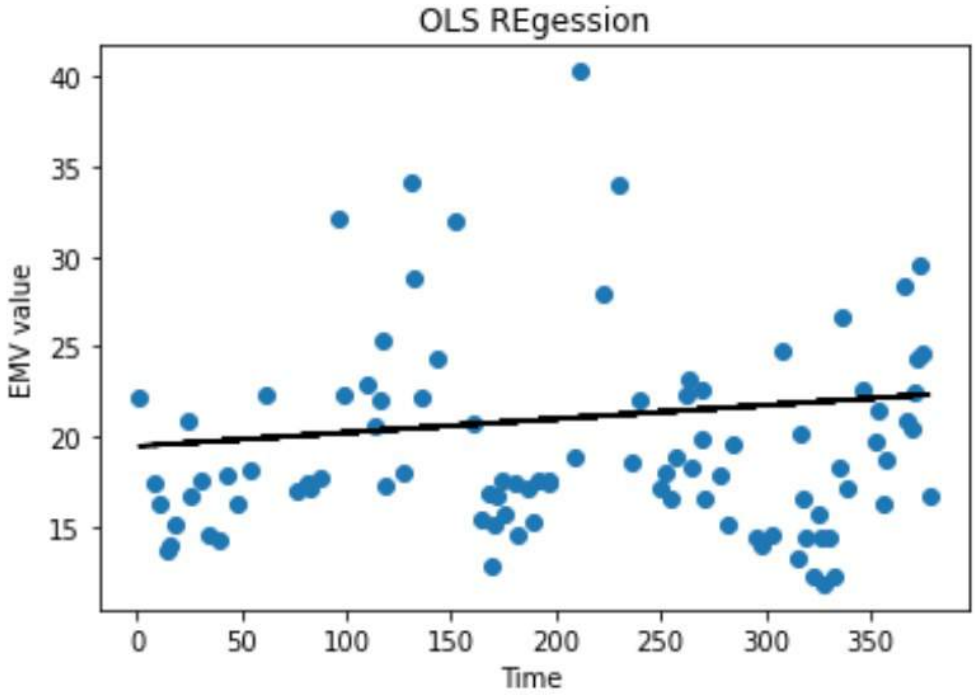
```
OLS_Regression(train_X, train_Y, test_X, test_Y)
```

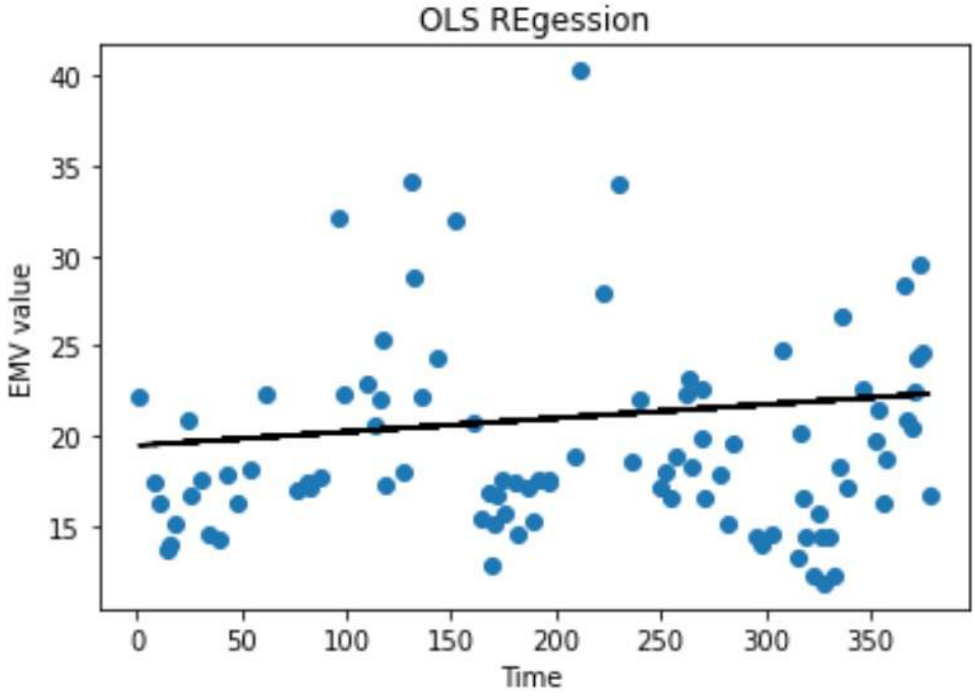

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

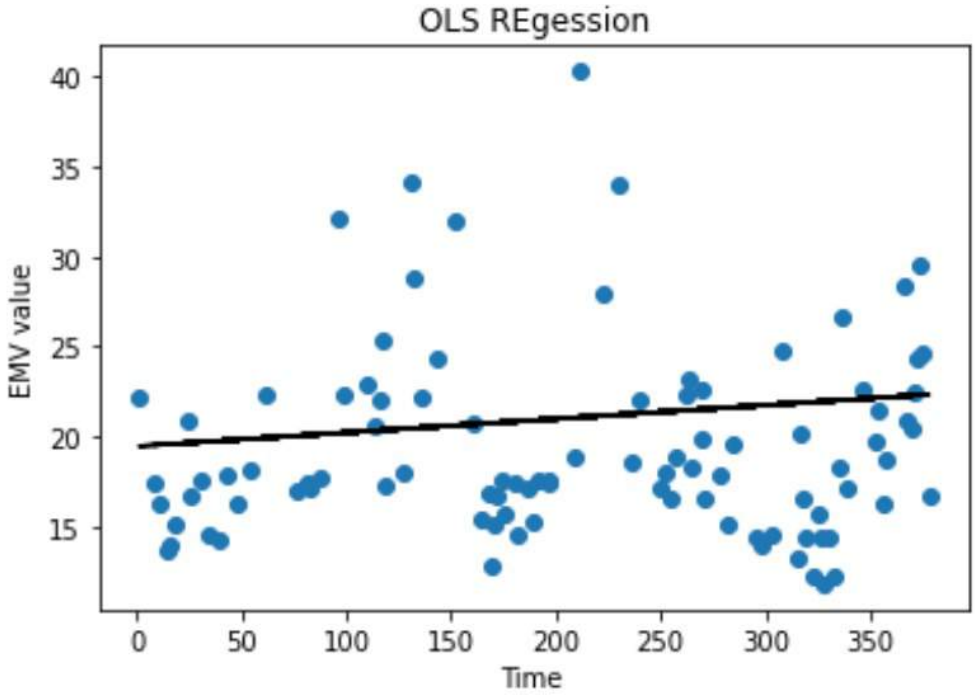

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

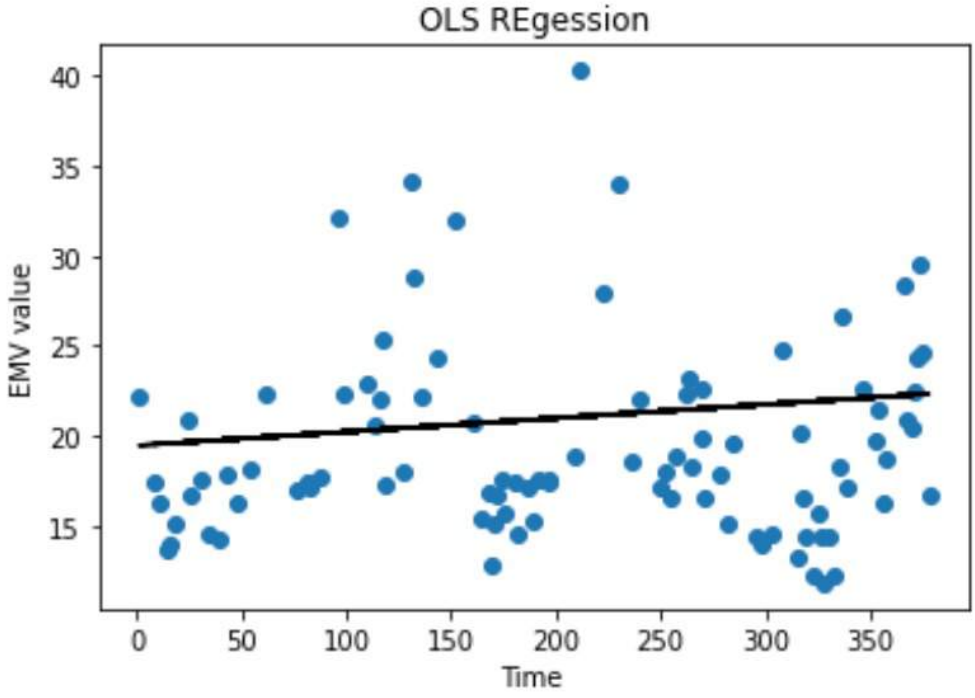

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

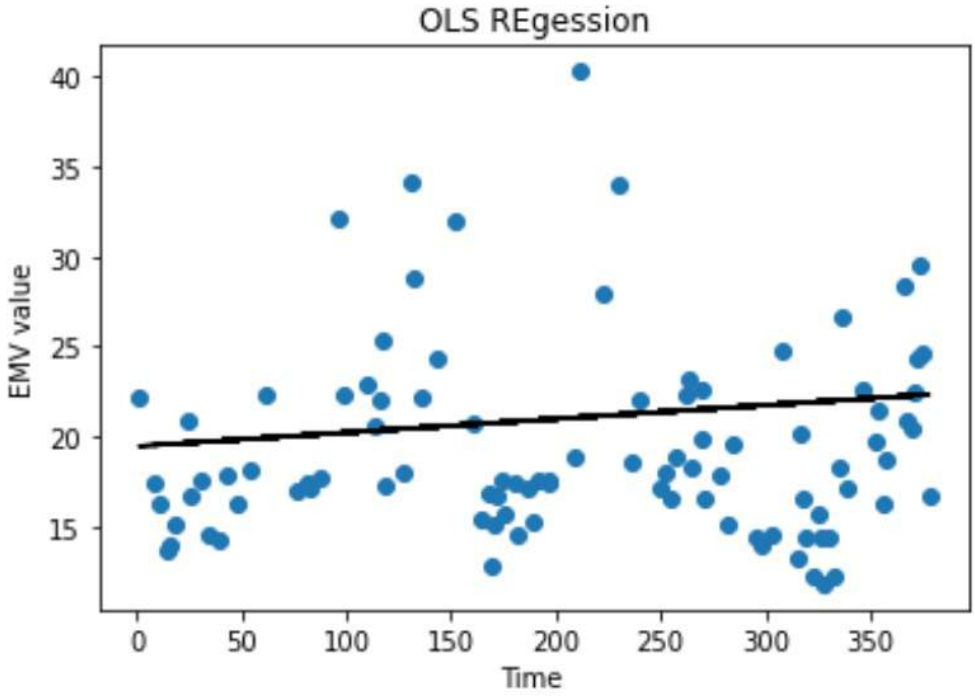

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

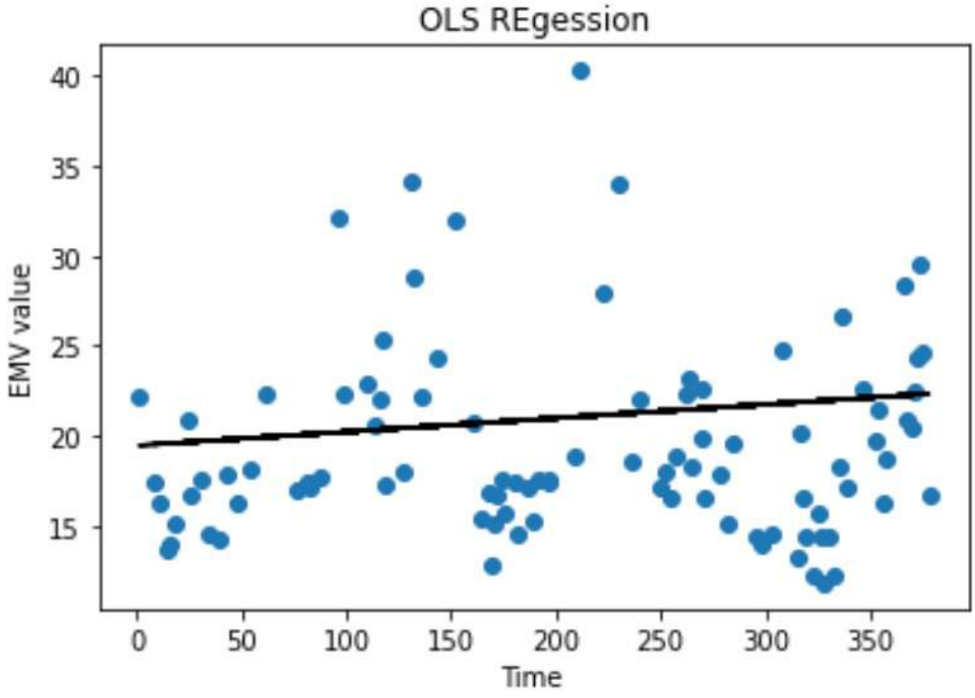

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

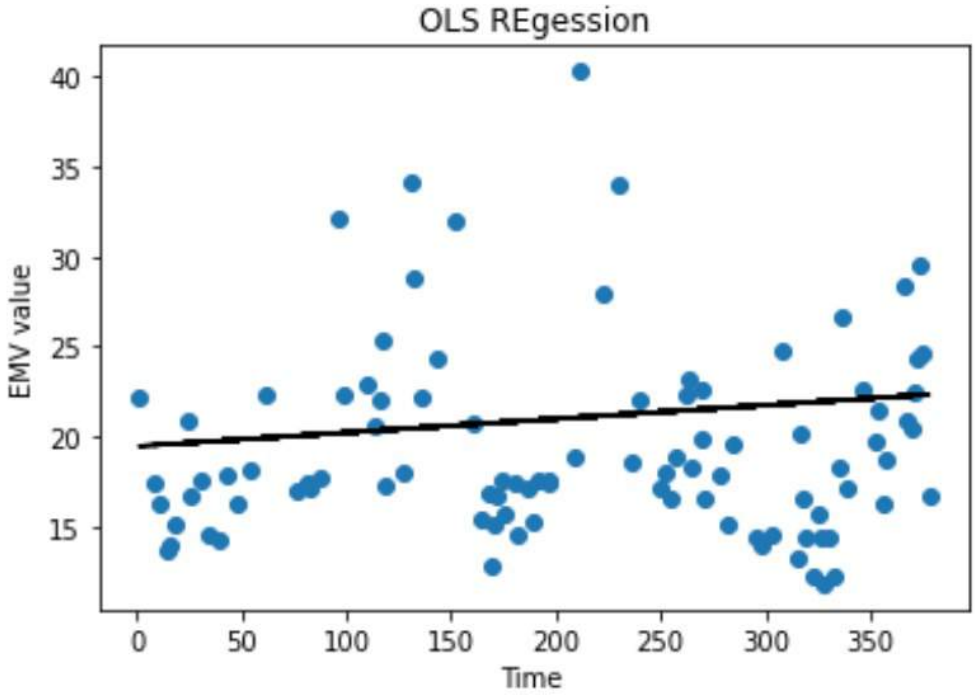

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

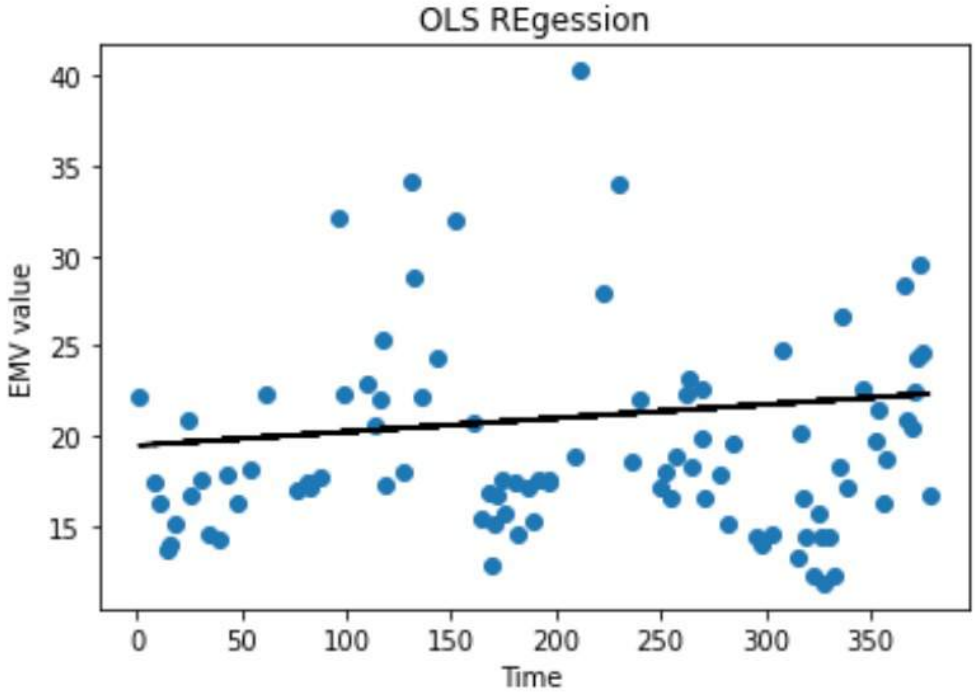

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

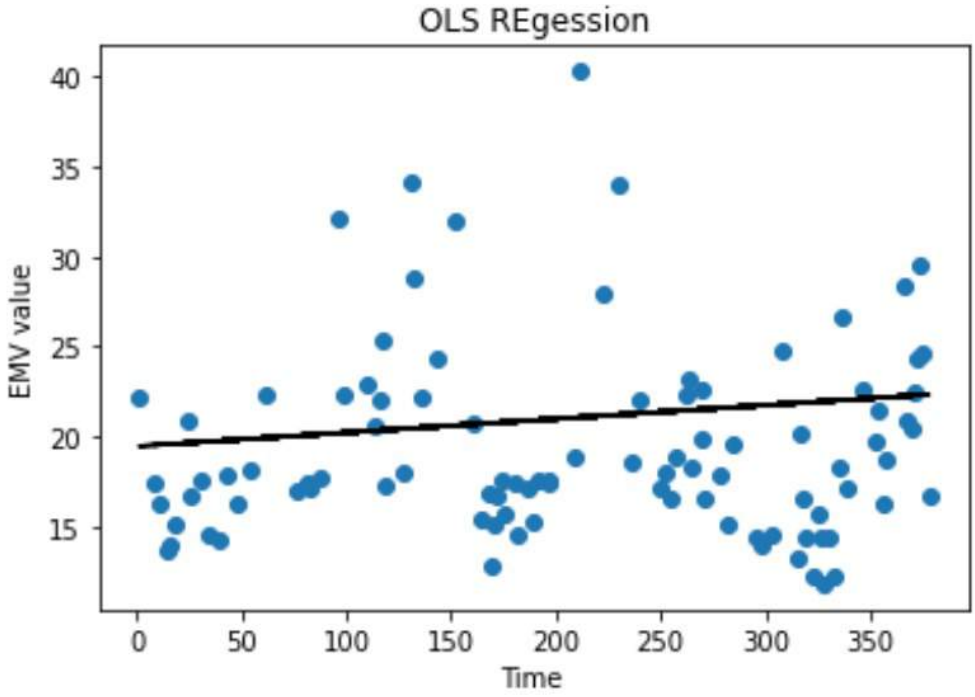

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

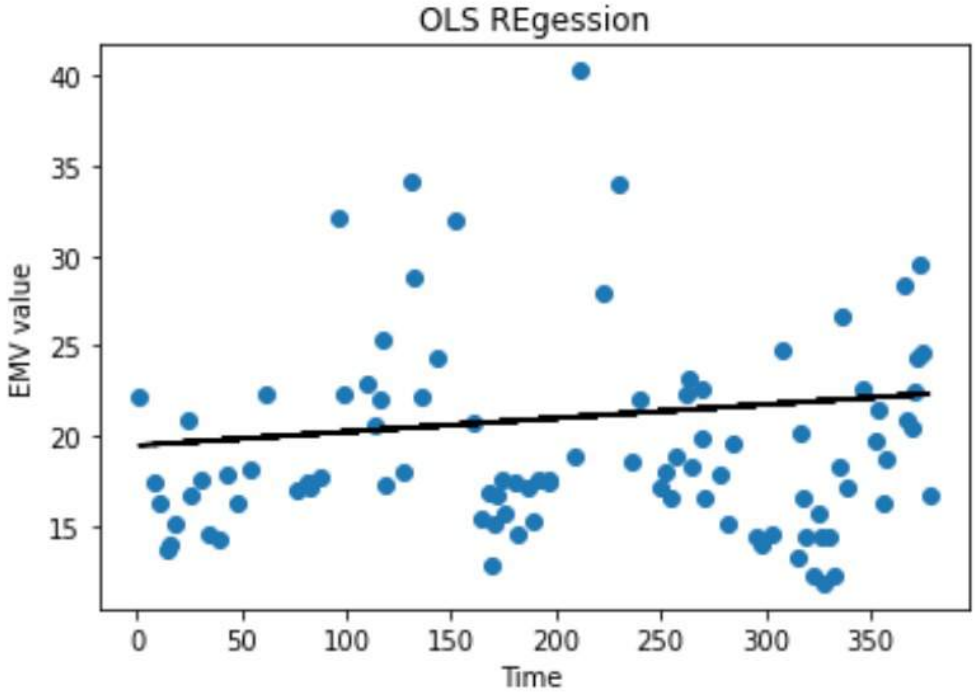

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

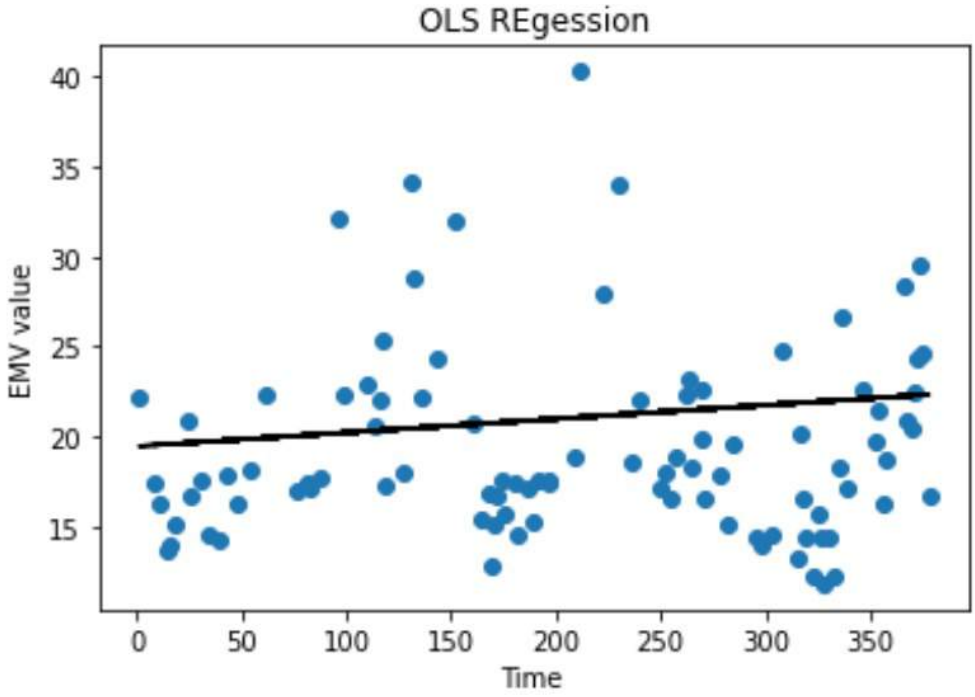

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

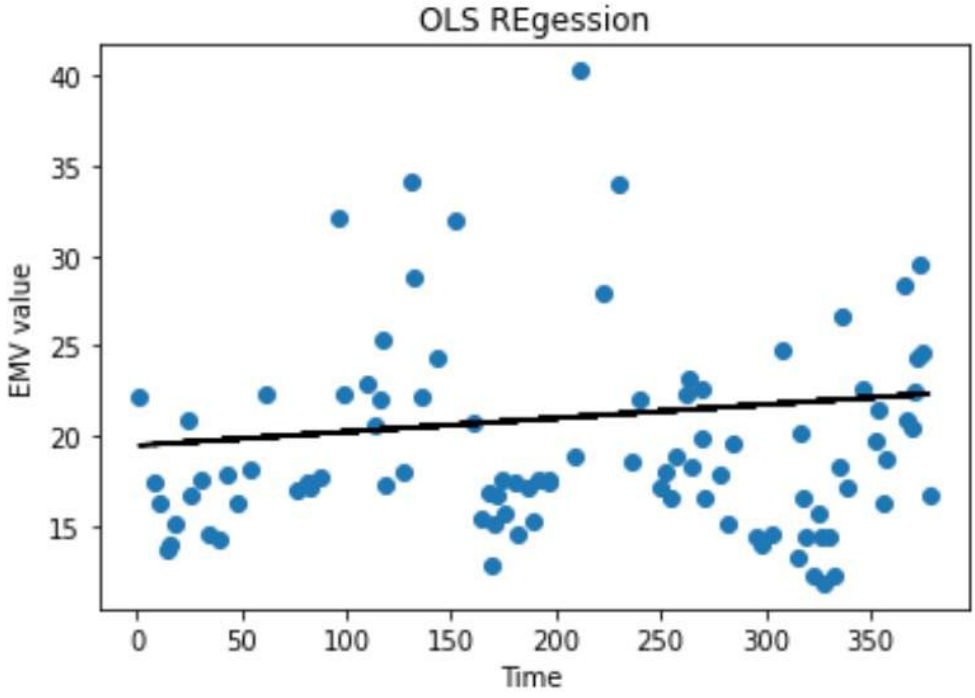

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

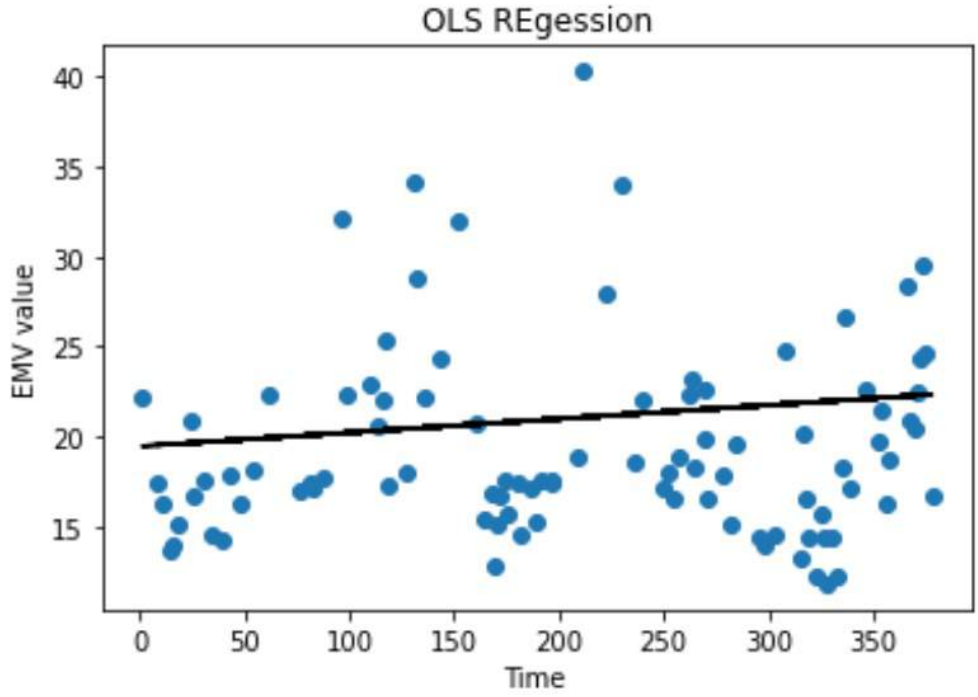

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

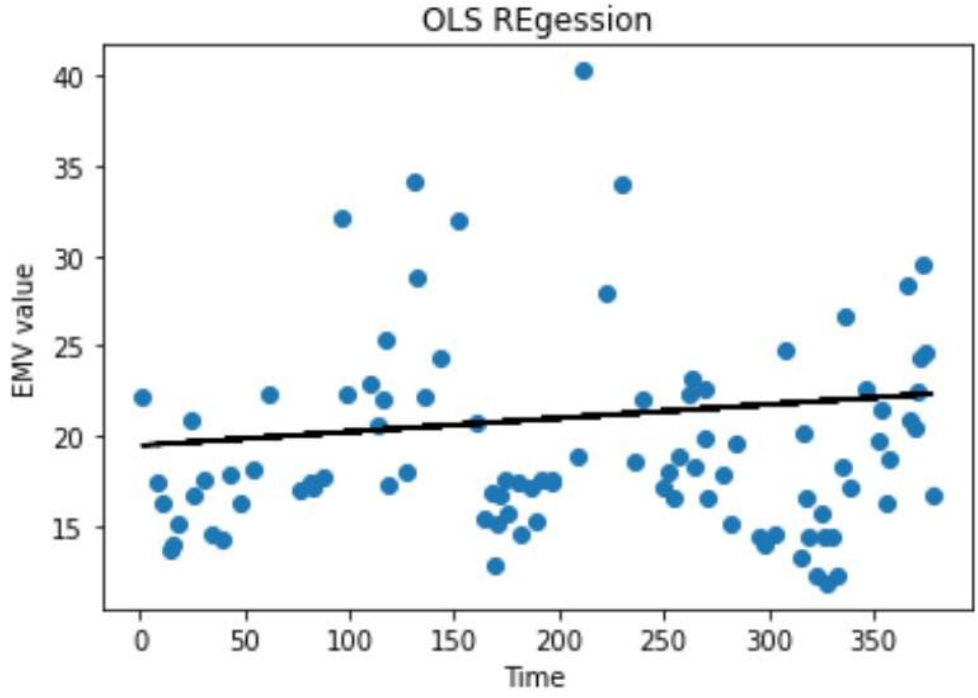

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

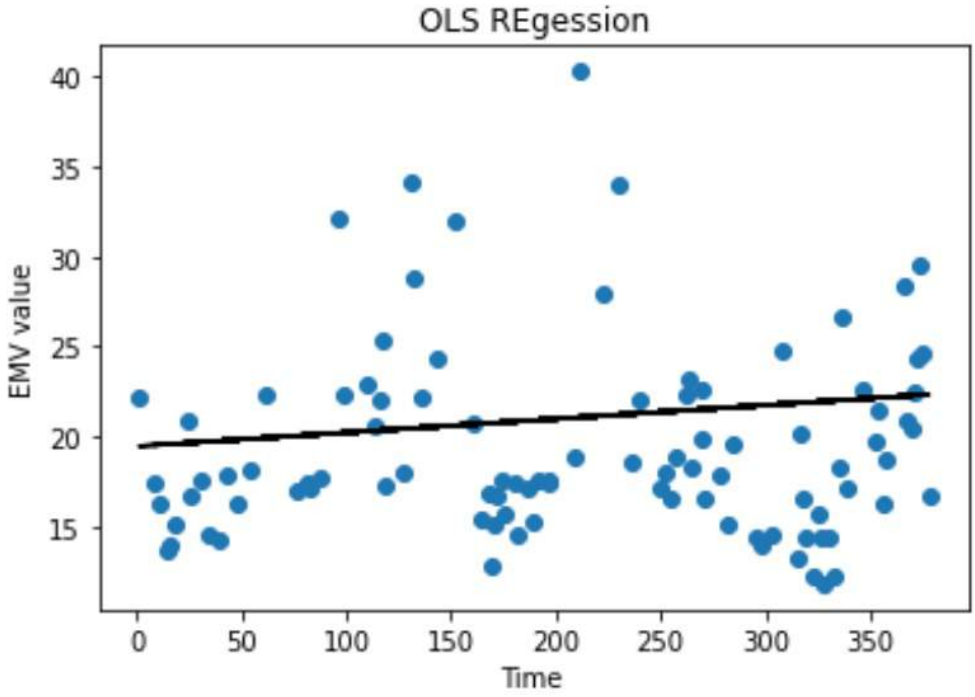

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

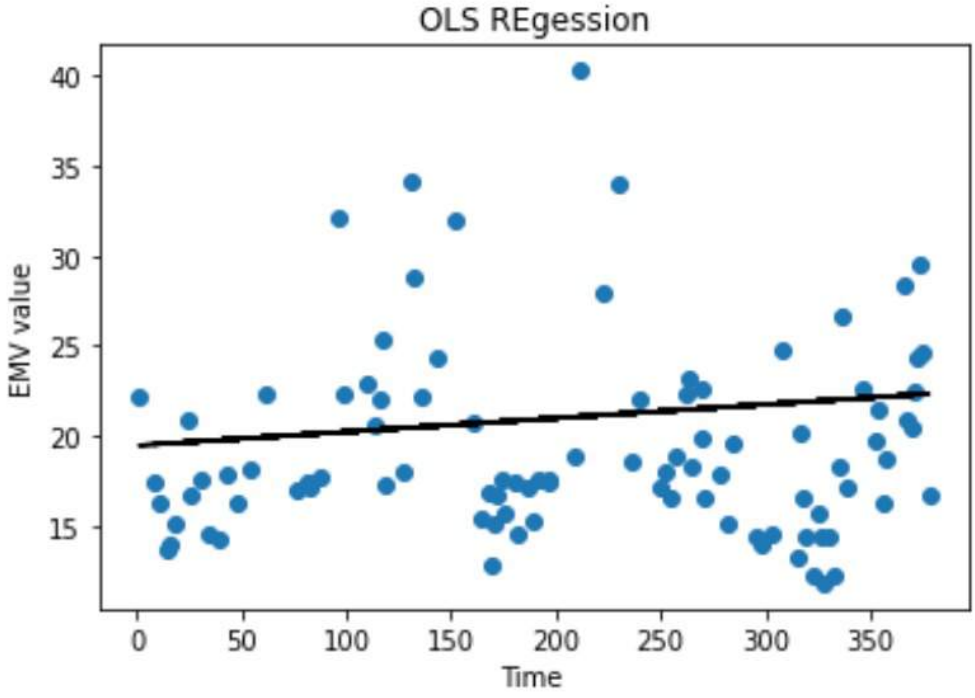

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

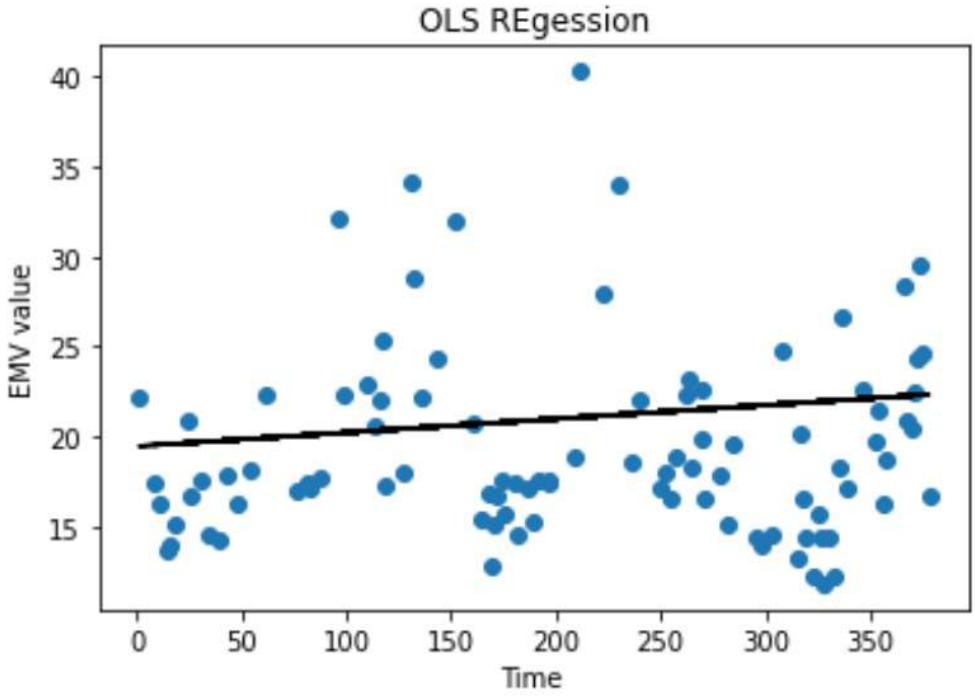

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

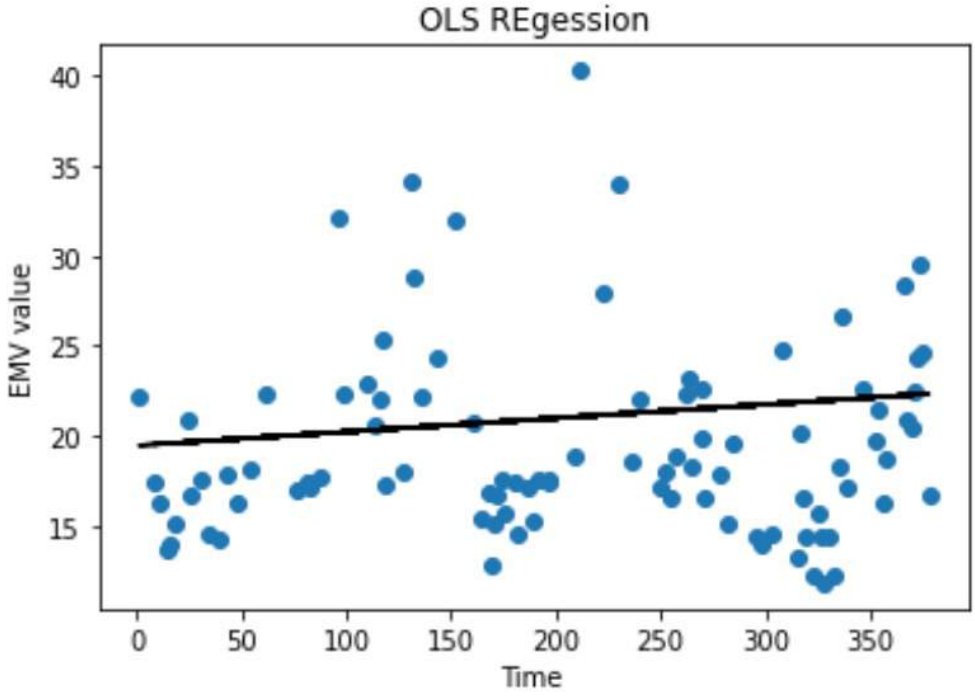

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

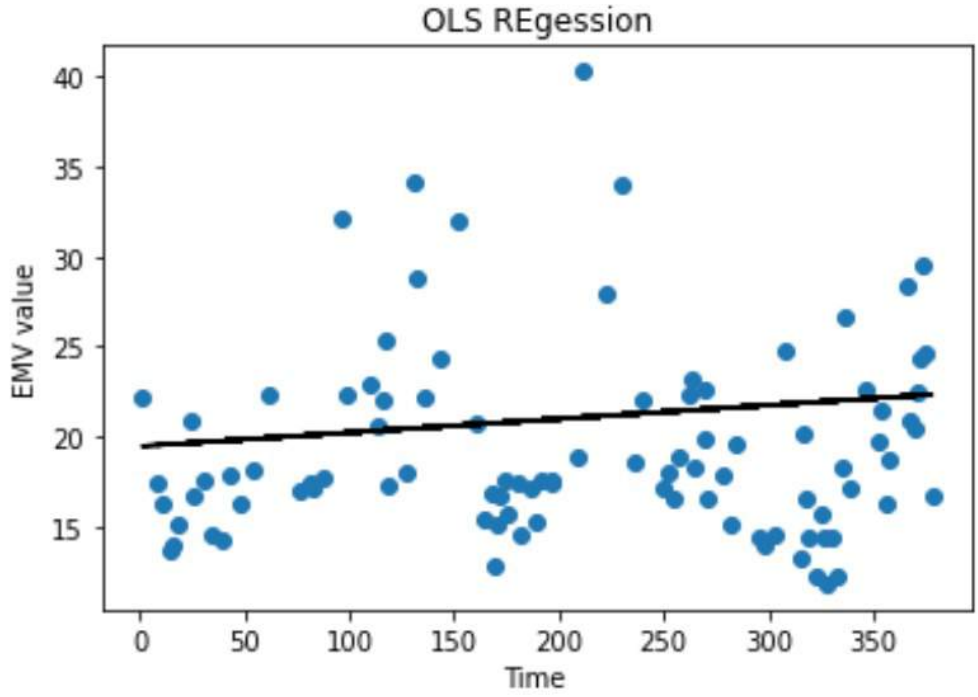

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

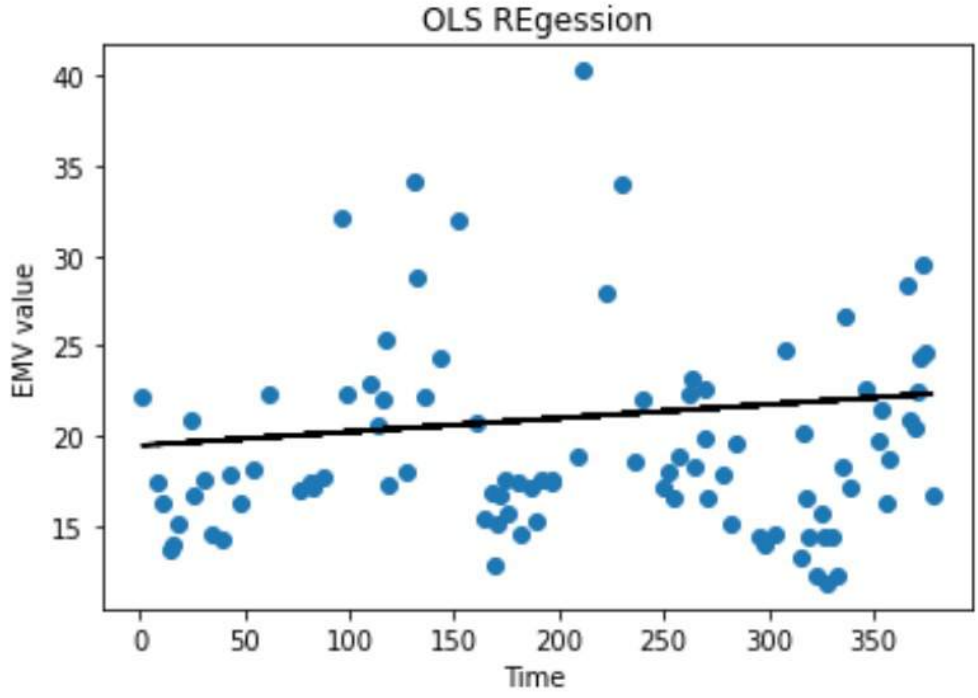

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

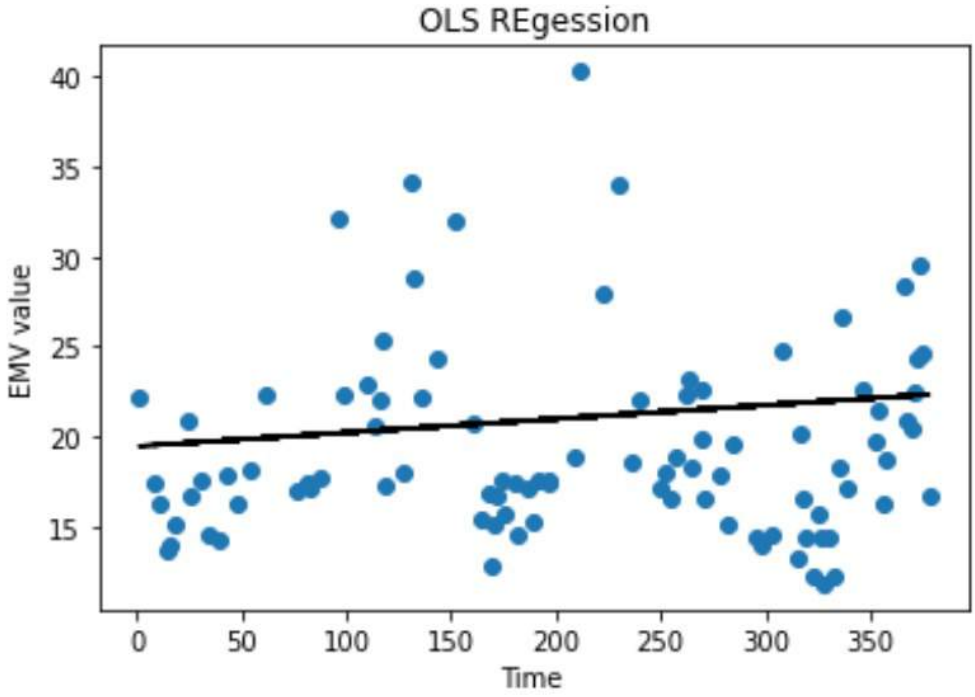

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

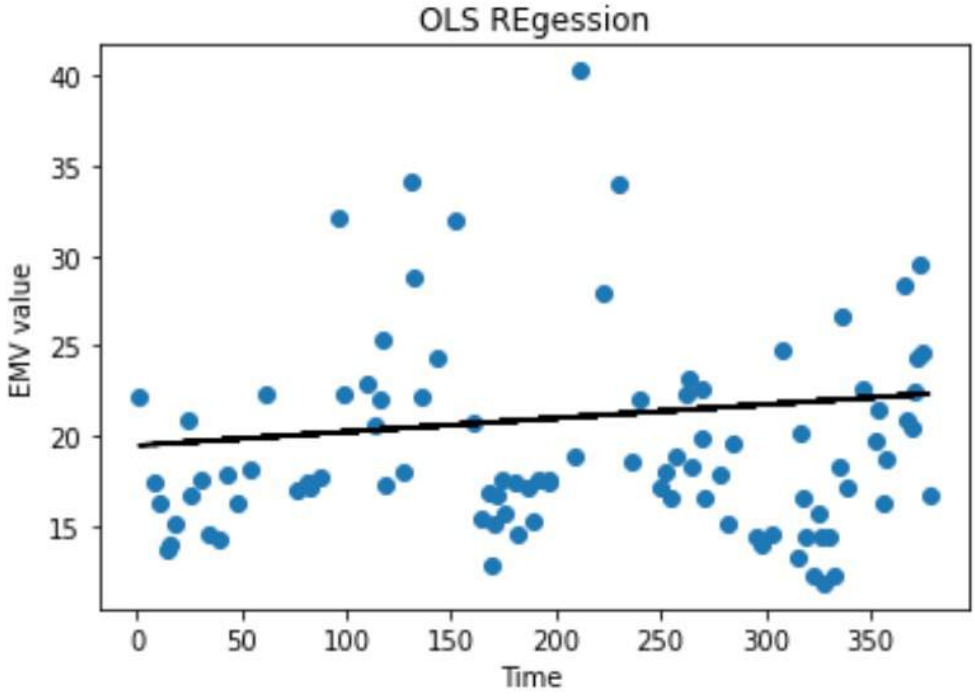

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

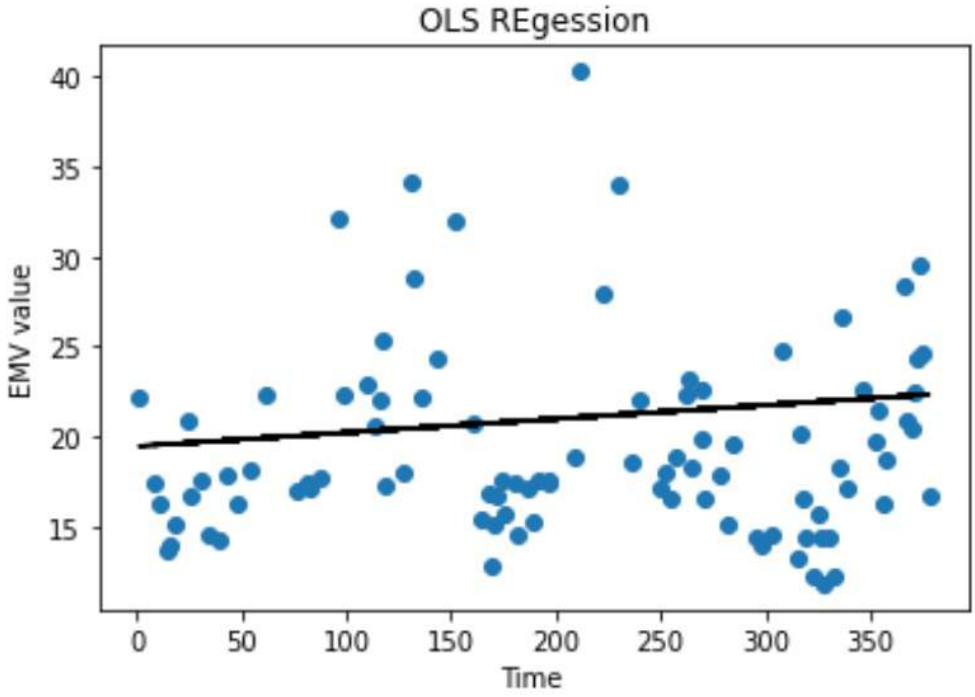

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

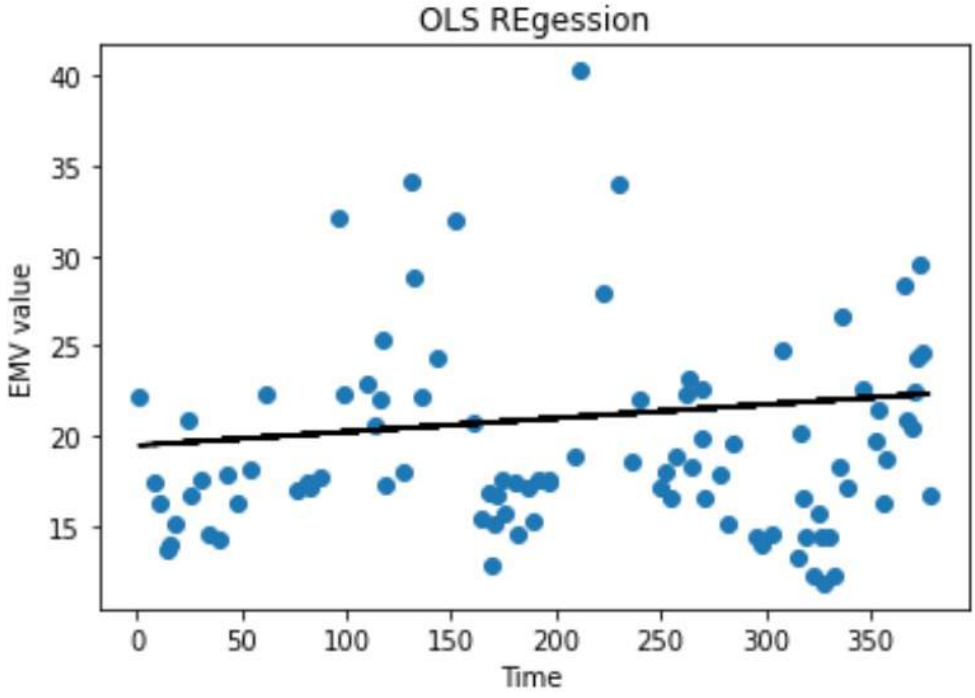

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

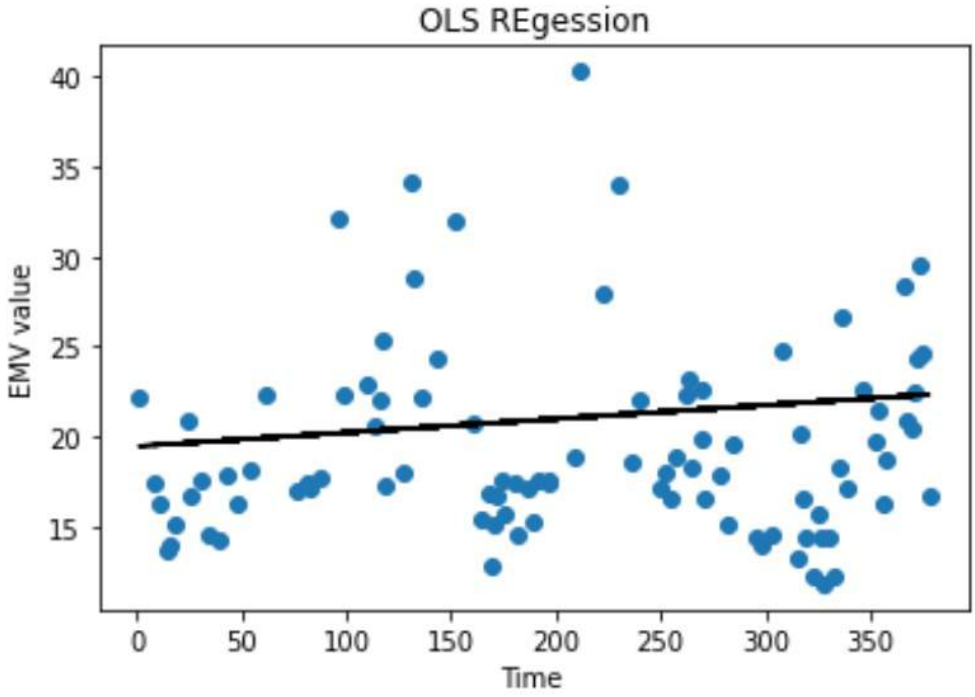

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

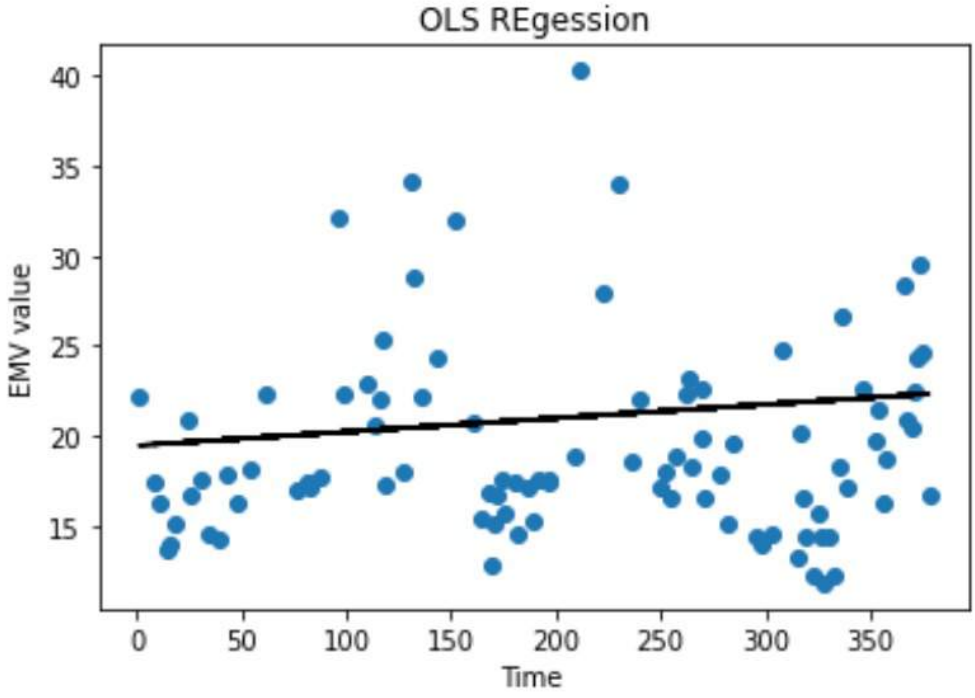

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

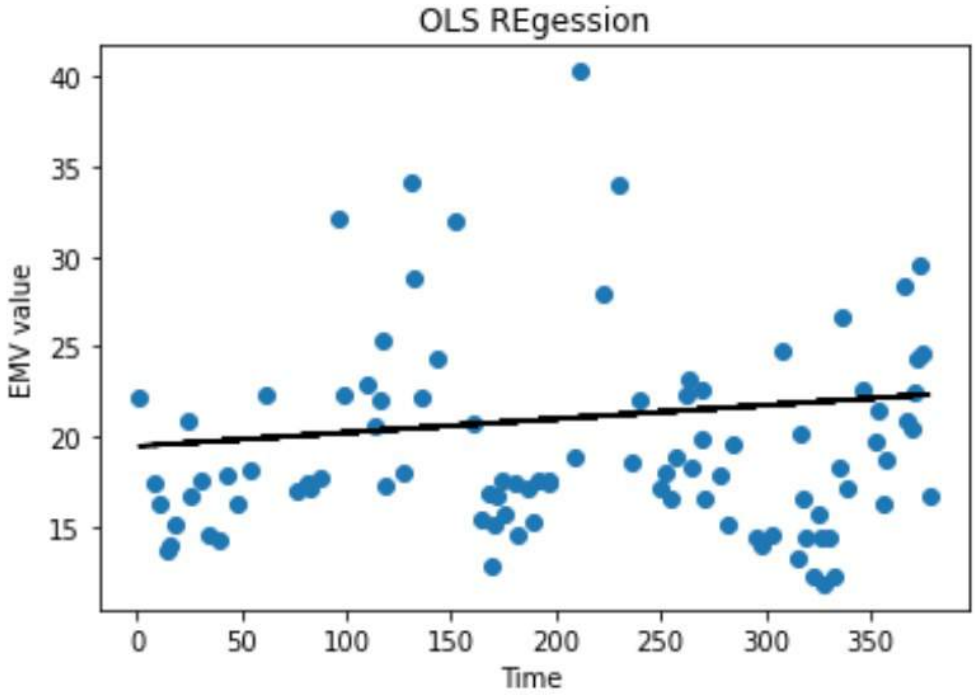

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

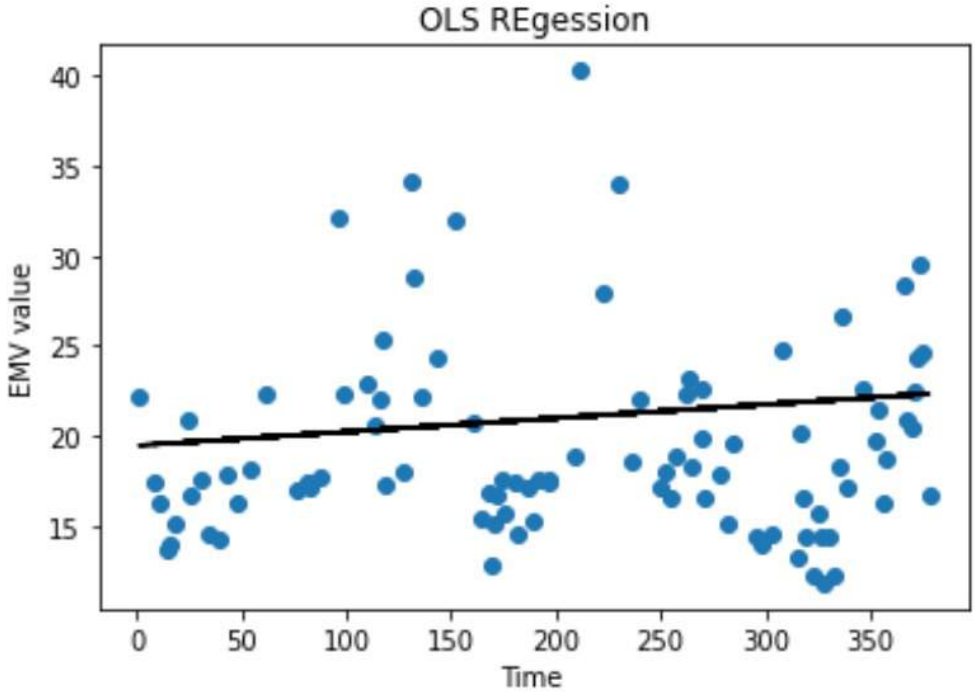

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

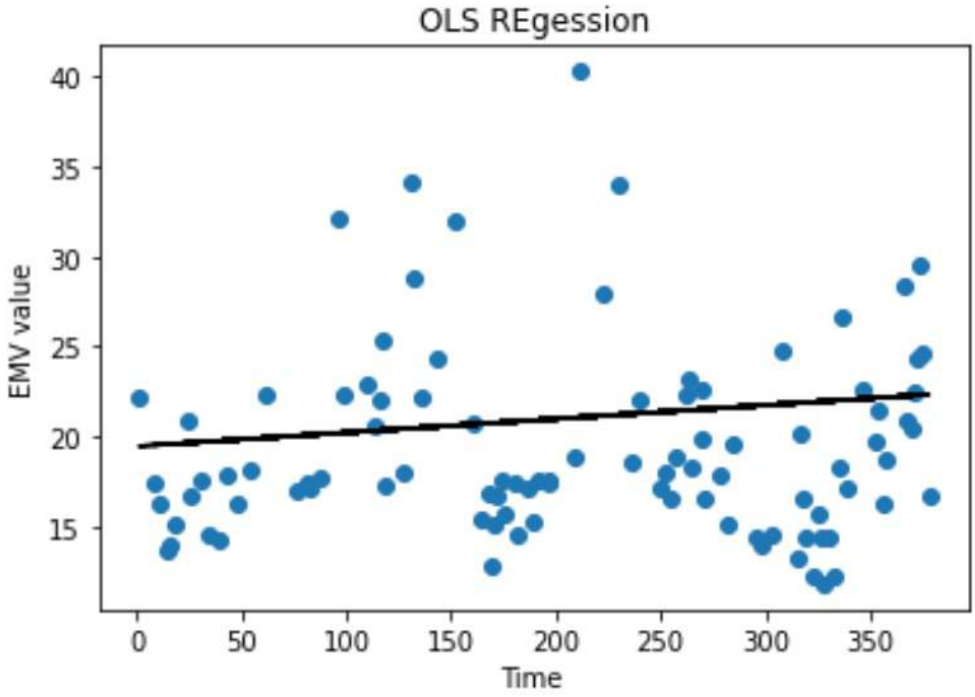

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

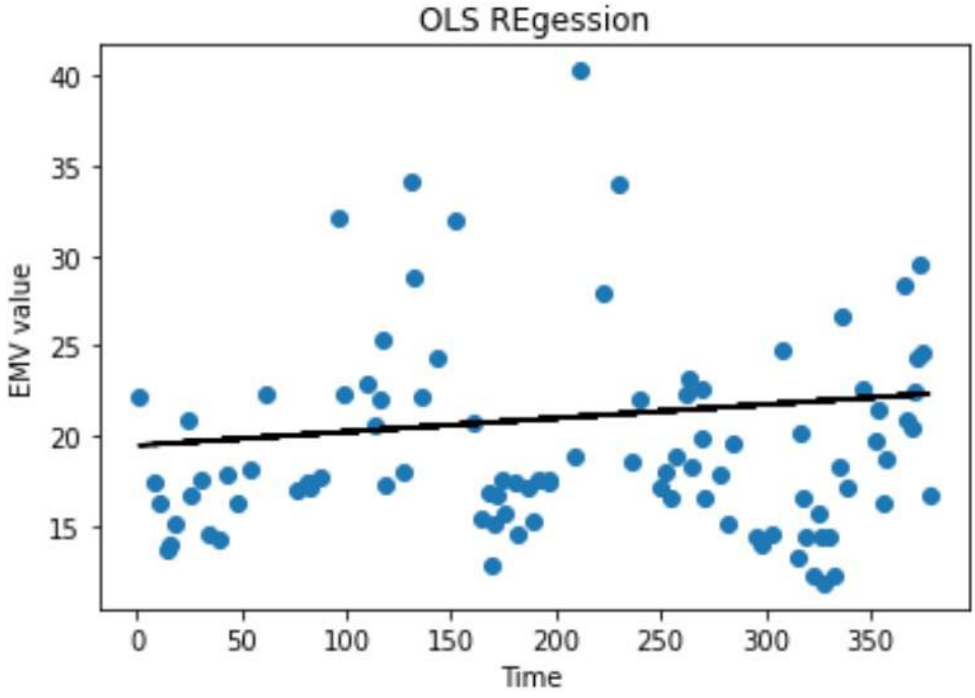

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

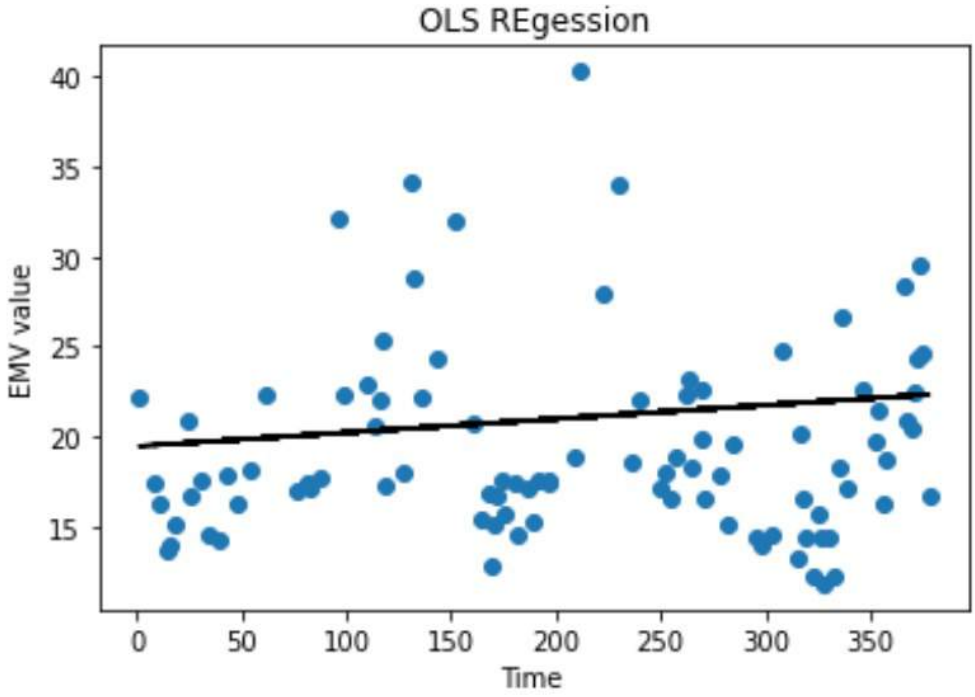

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

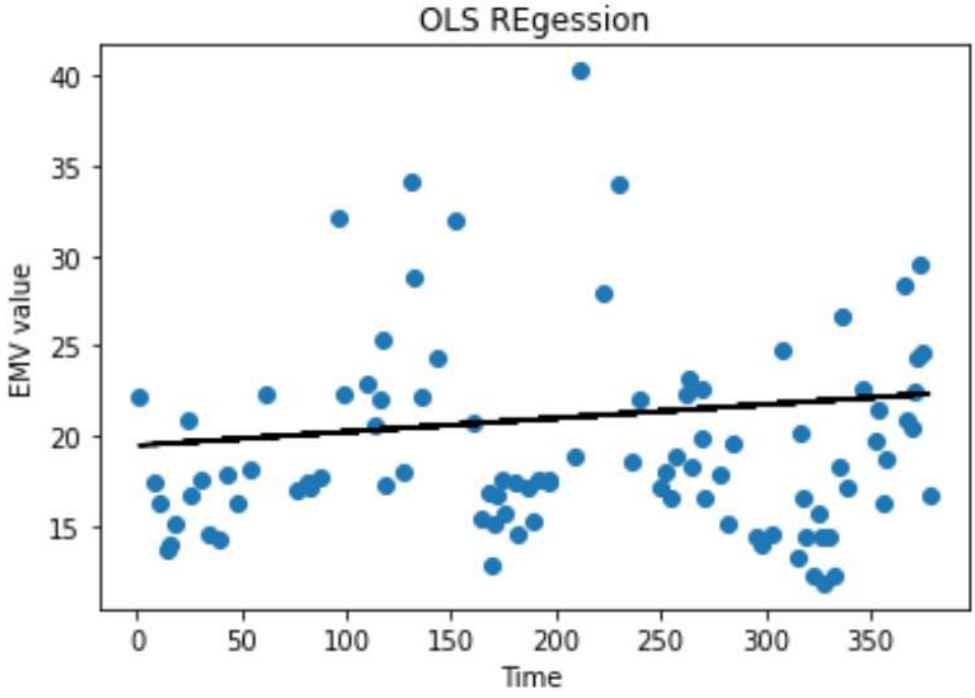

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

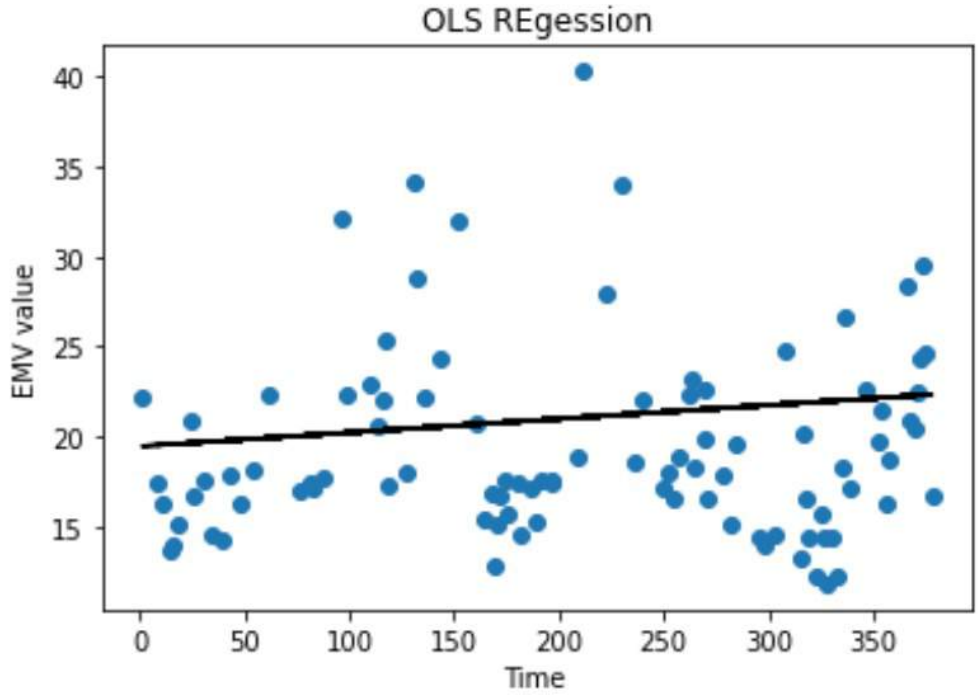

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

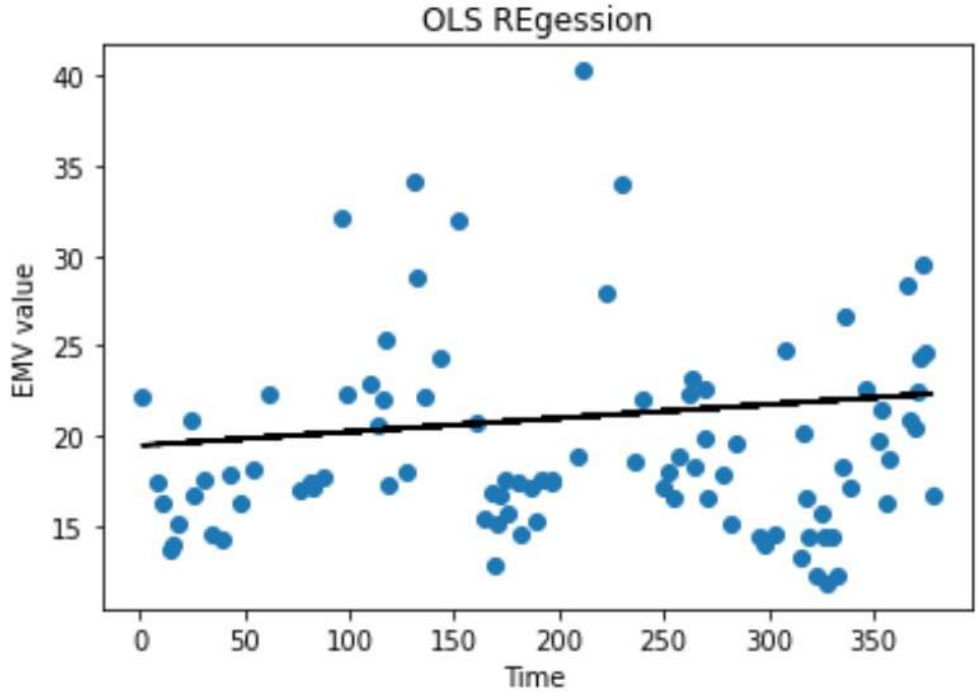

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

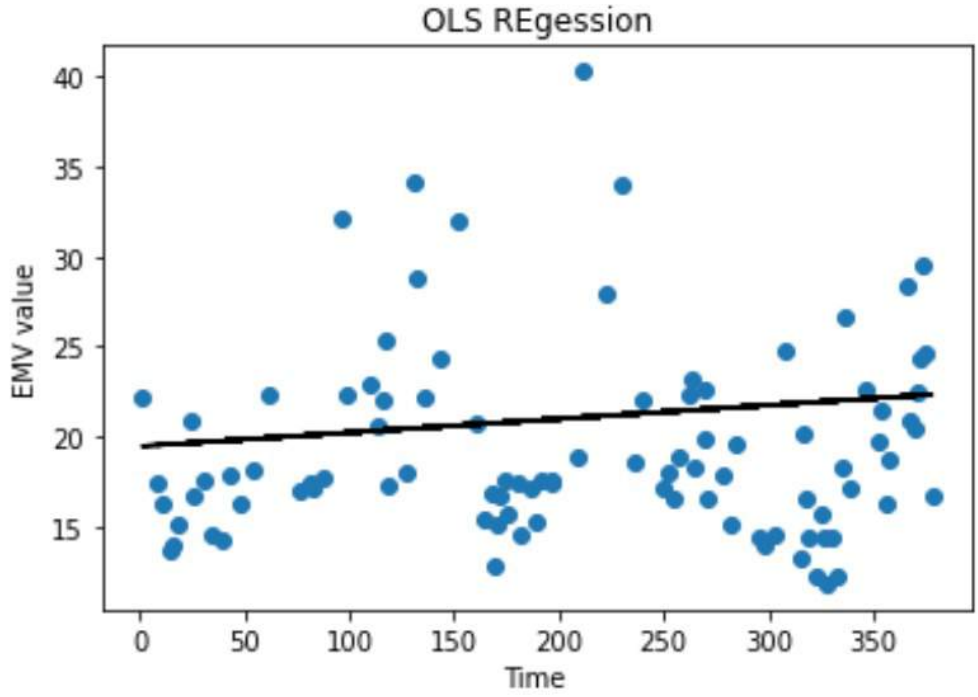

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

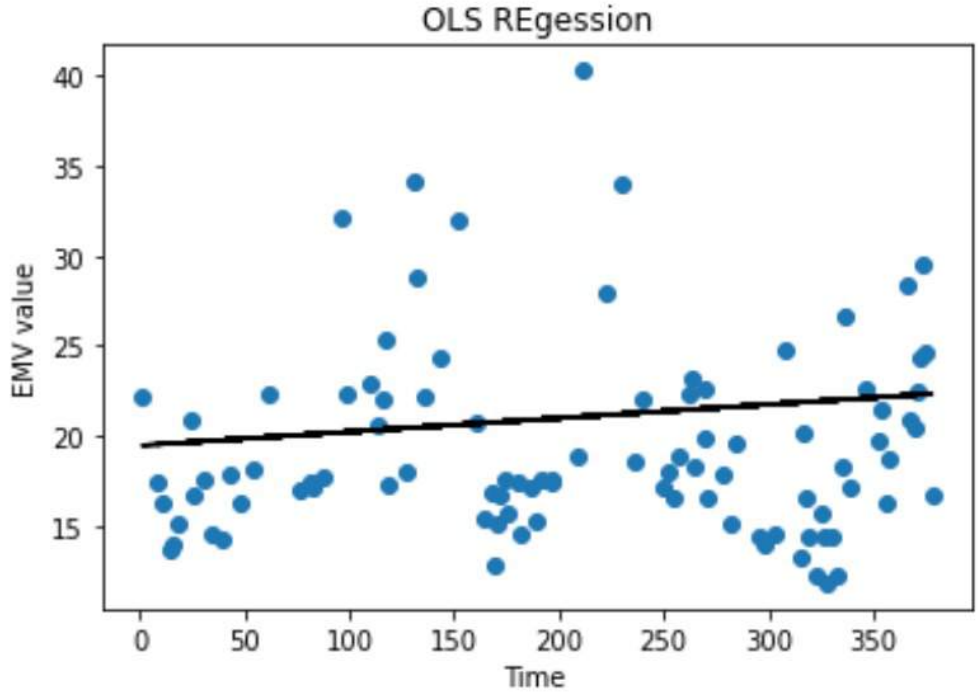

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

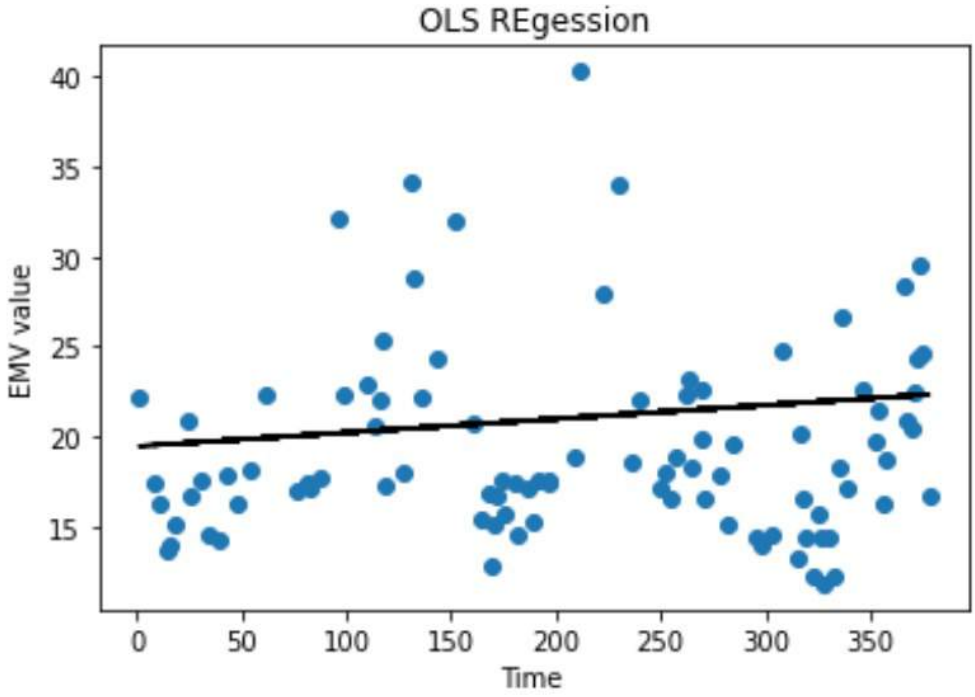

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

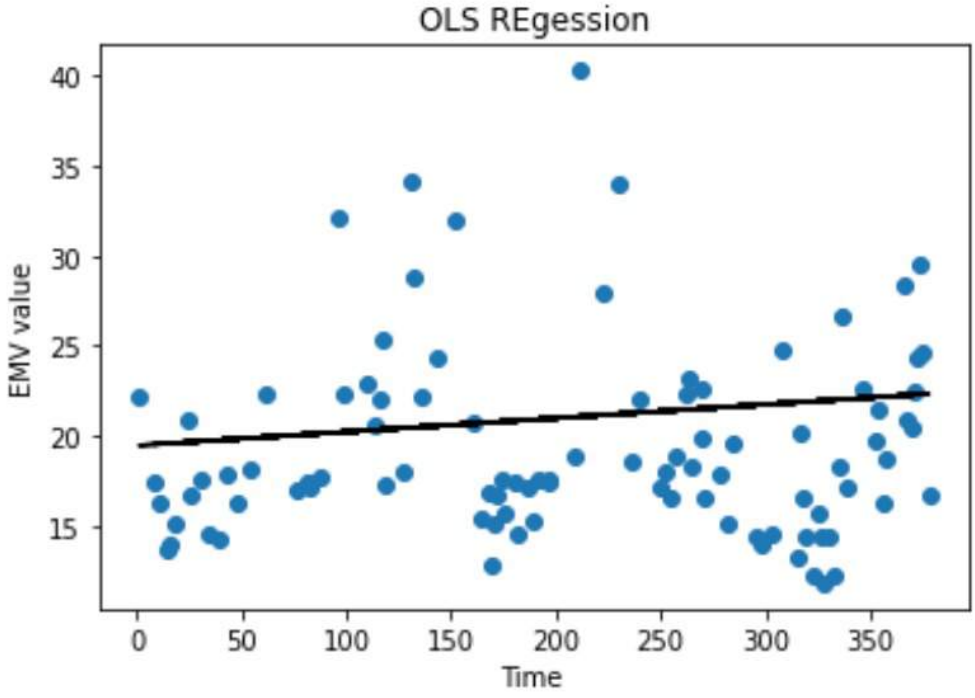

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

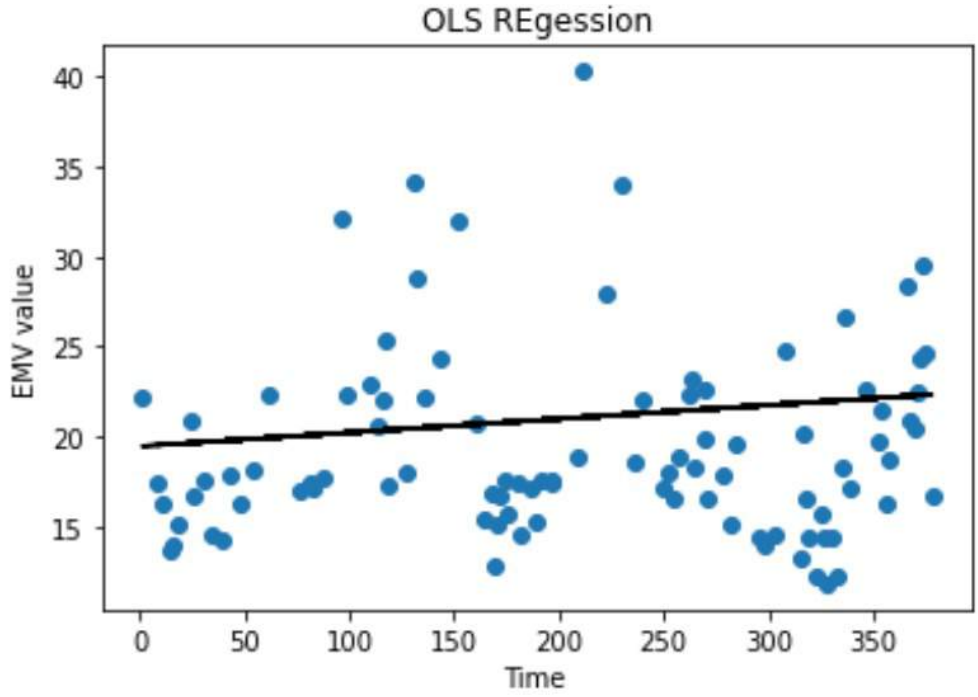

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

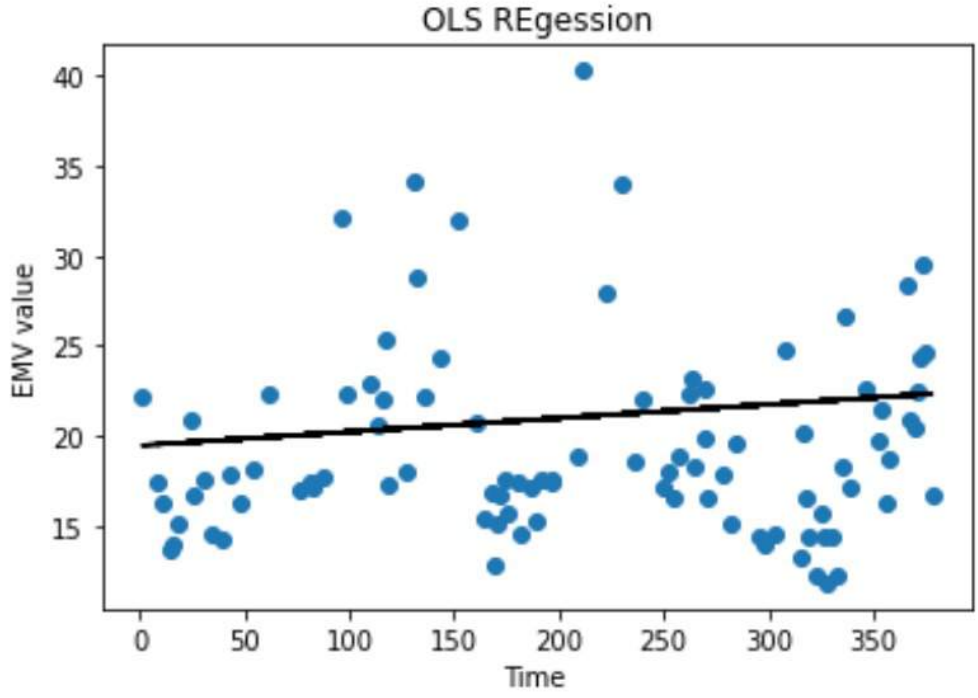

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

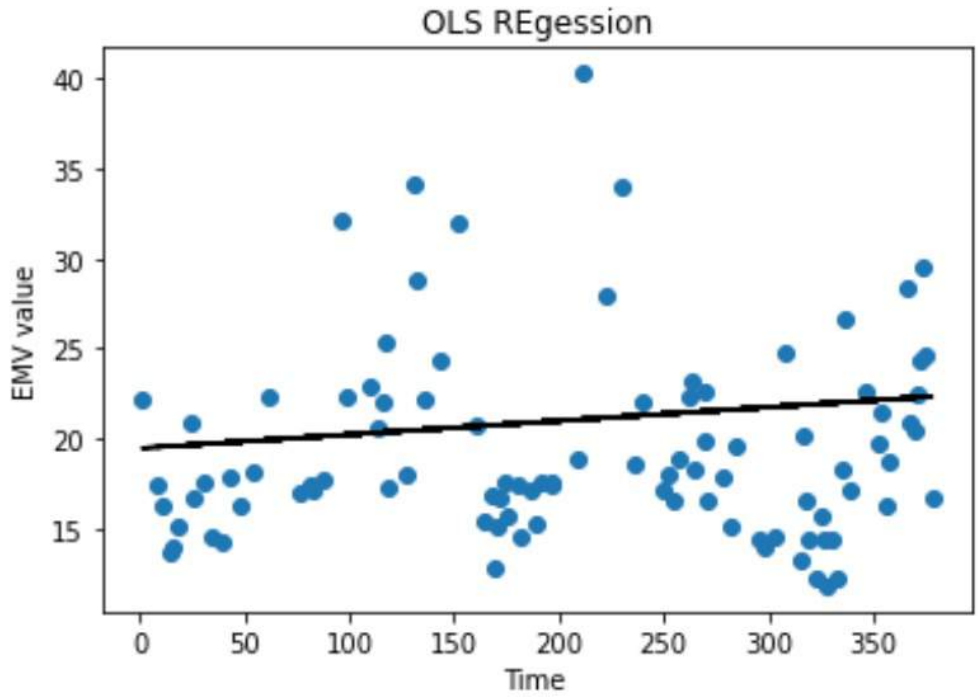

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350


OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

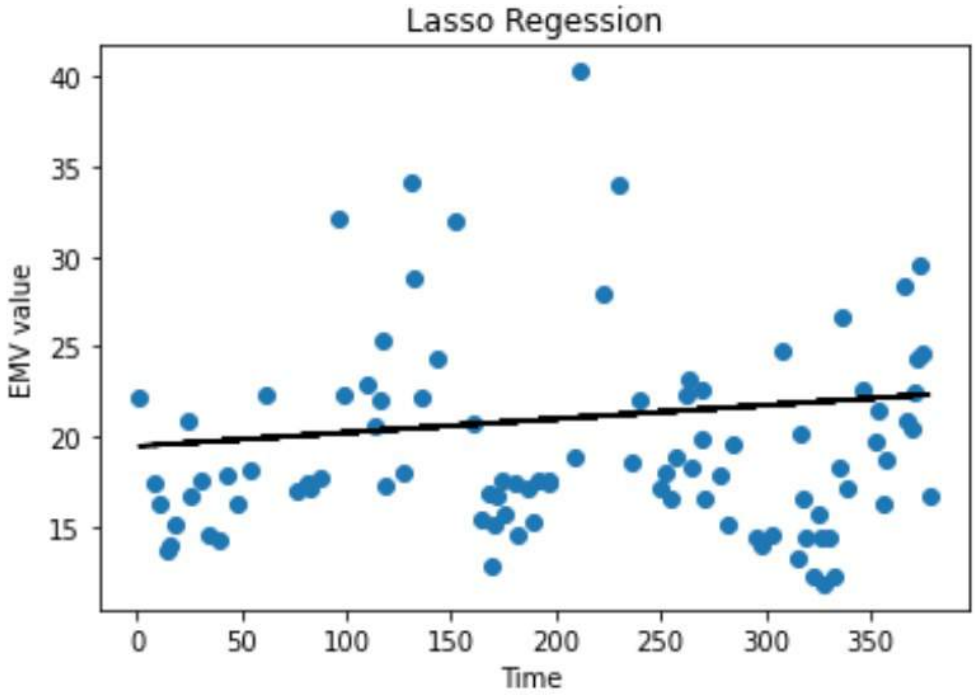

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

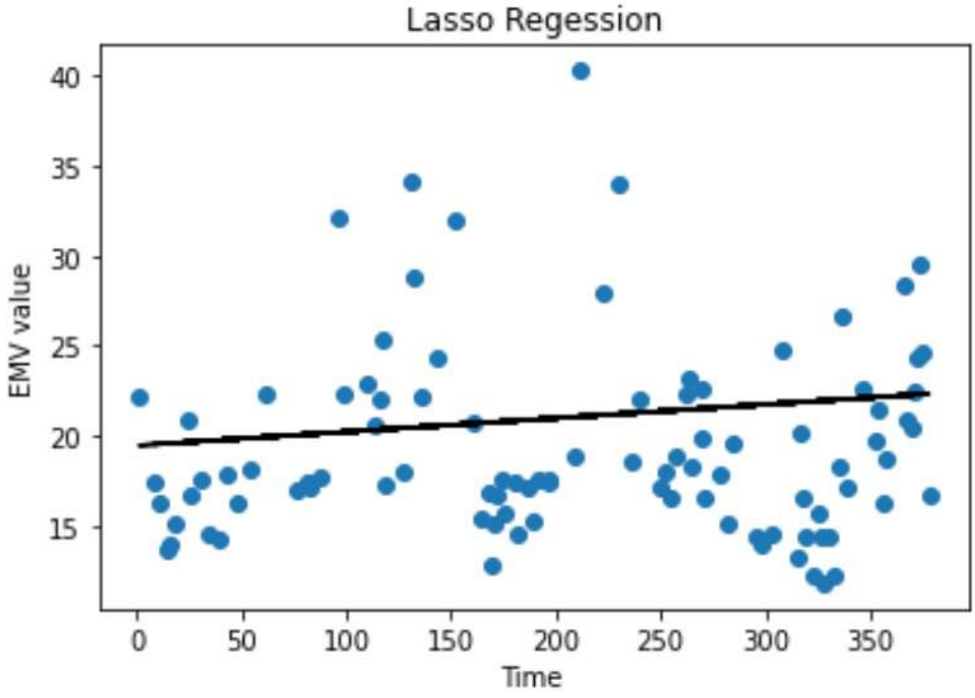

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

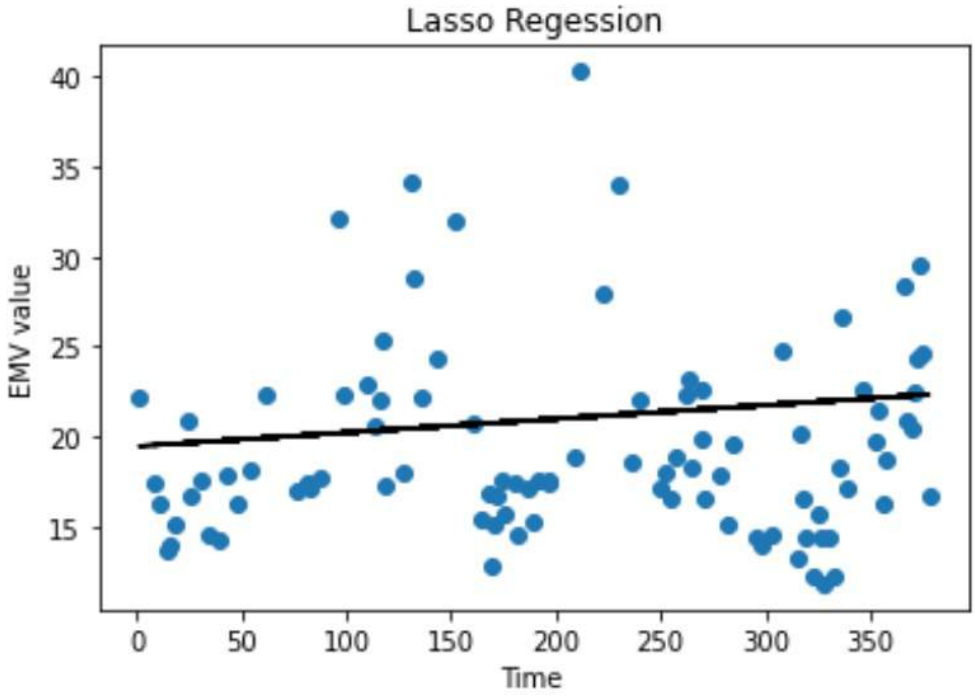

OLS REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

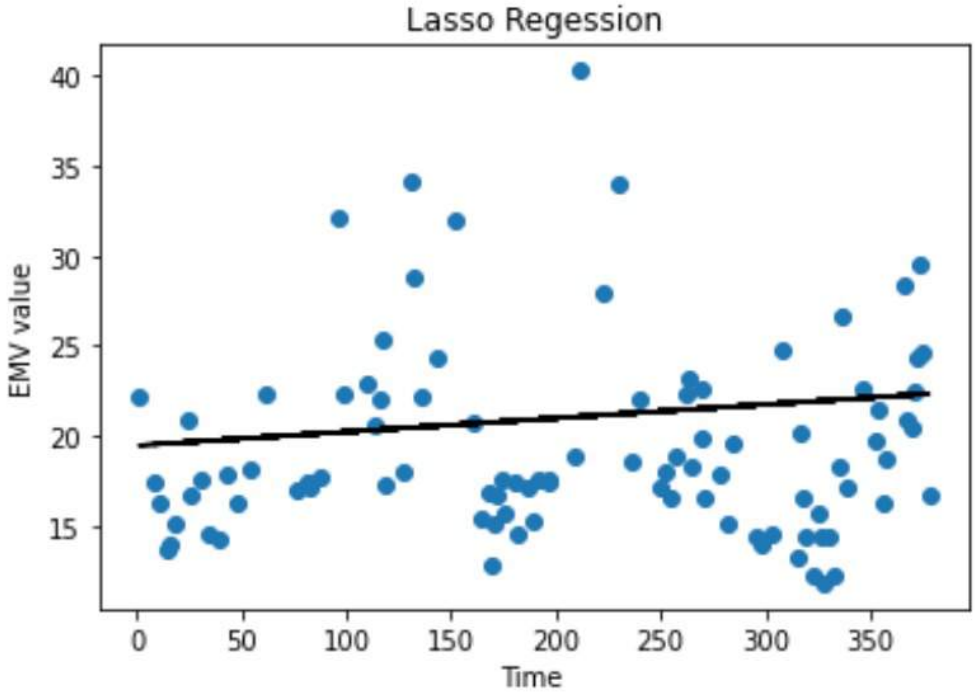


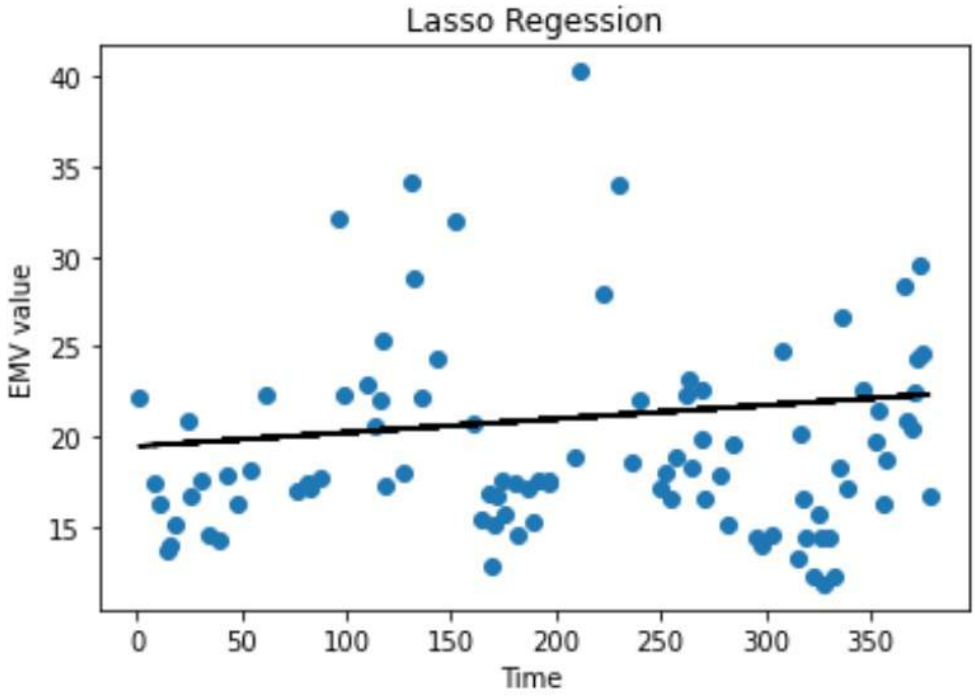

```
def Lasso_Regression(train_X, train_Y, test_X, test_Y):
    reg=Lasso(alpha=A)
    reg.fit(train_X, train_Y)
    pred_Y=reg.predict(test_X)

    for tracker in range(45):
        plt.scatter(test_X, [y for y in test_Y])
        plt.plot(test_X,  [y for y in  pred_Y], color='black')
        plt.title("Lasso Regession")
        plt.xlabel("Time ")
        plt.ylabel("EMV value")

        plt.show()

Lasso_Regression(train_X, train_Y, test_X, test_Y)
```

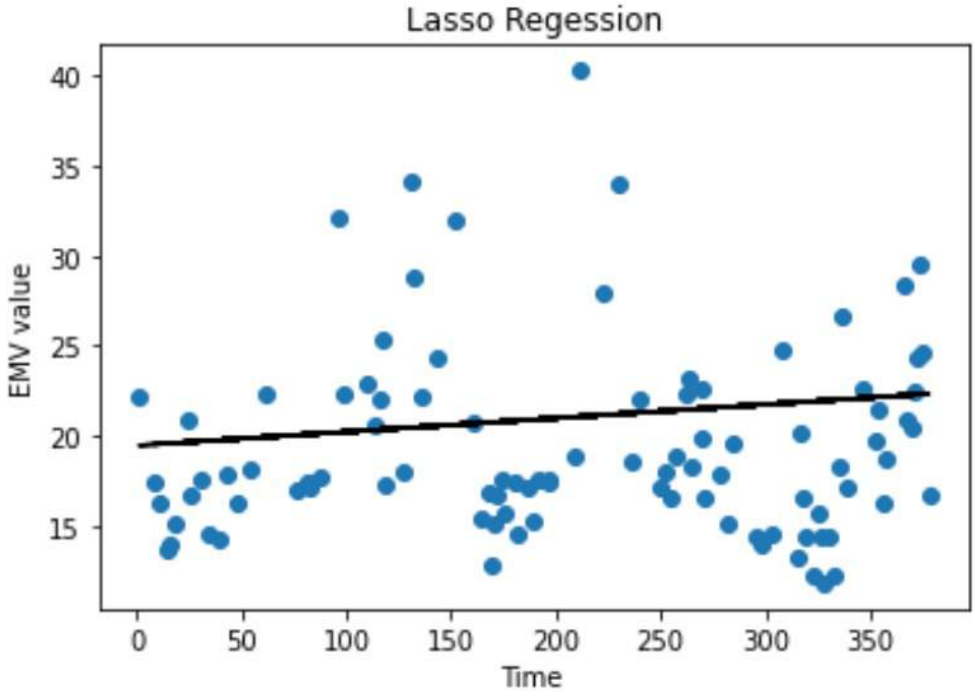

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

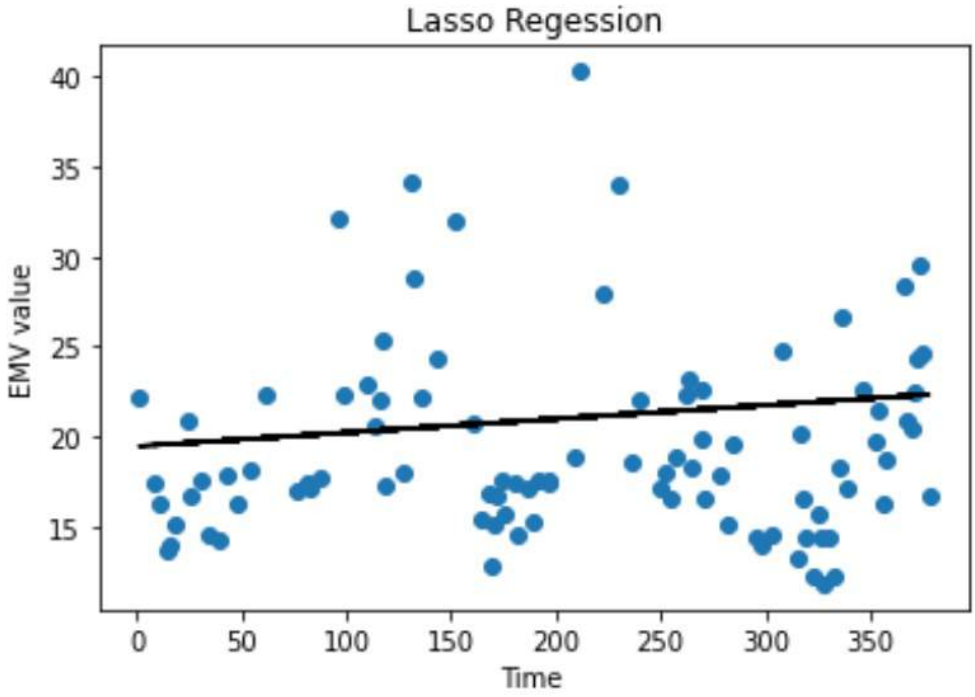

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

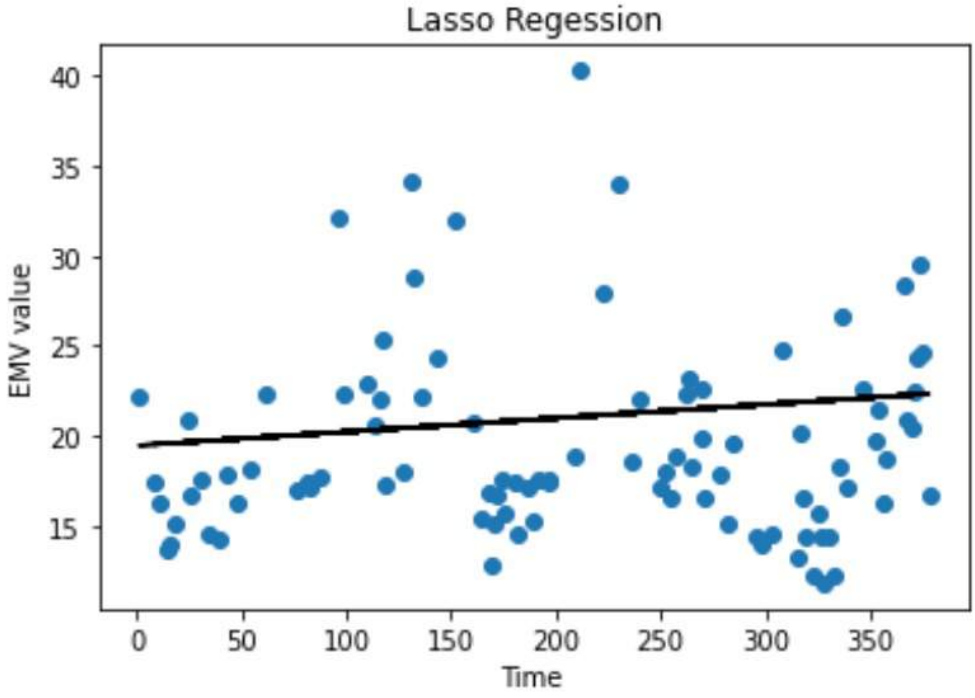

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

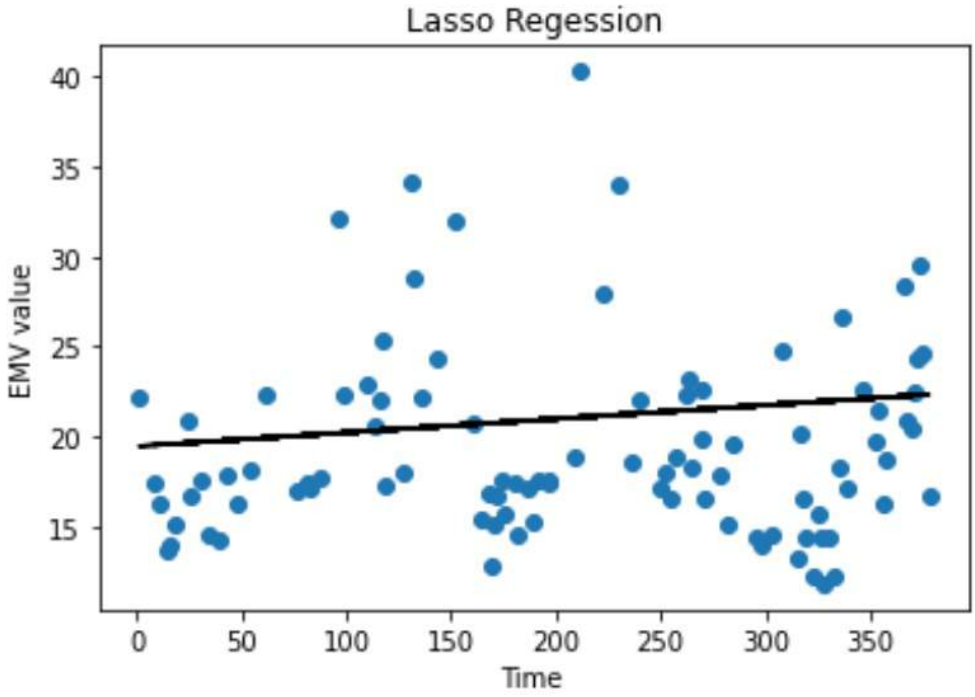

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

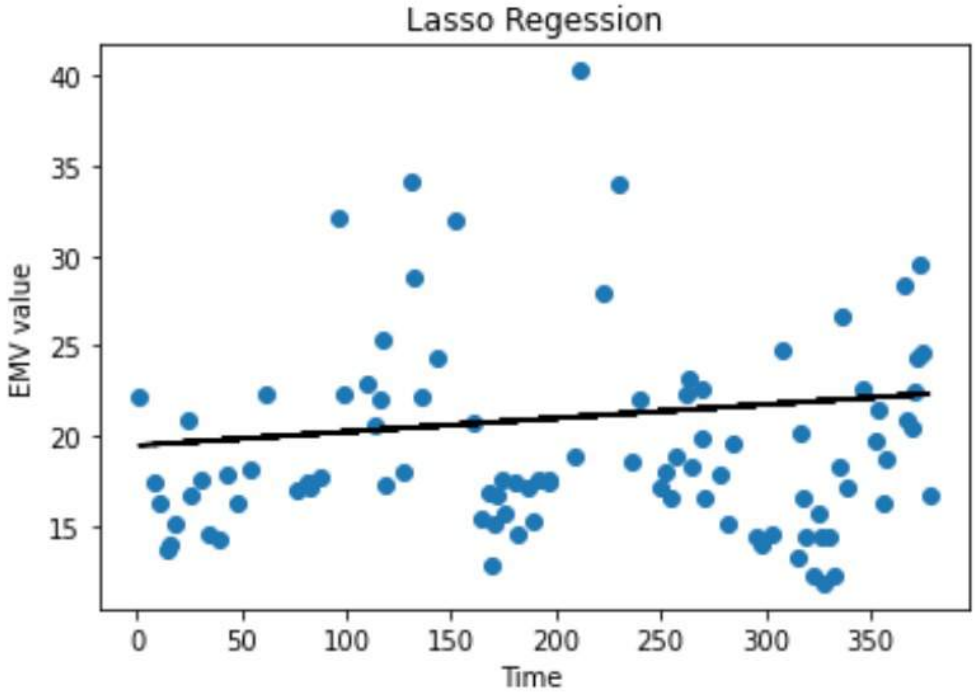

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

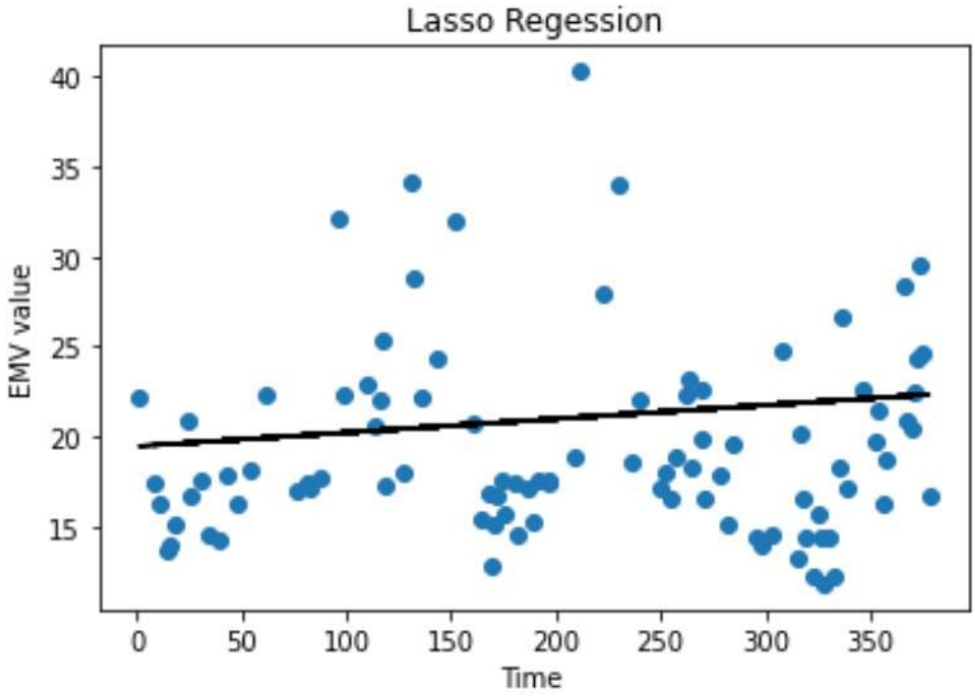

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

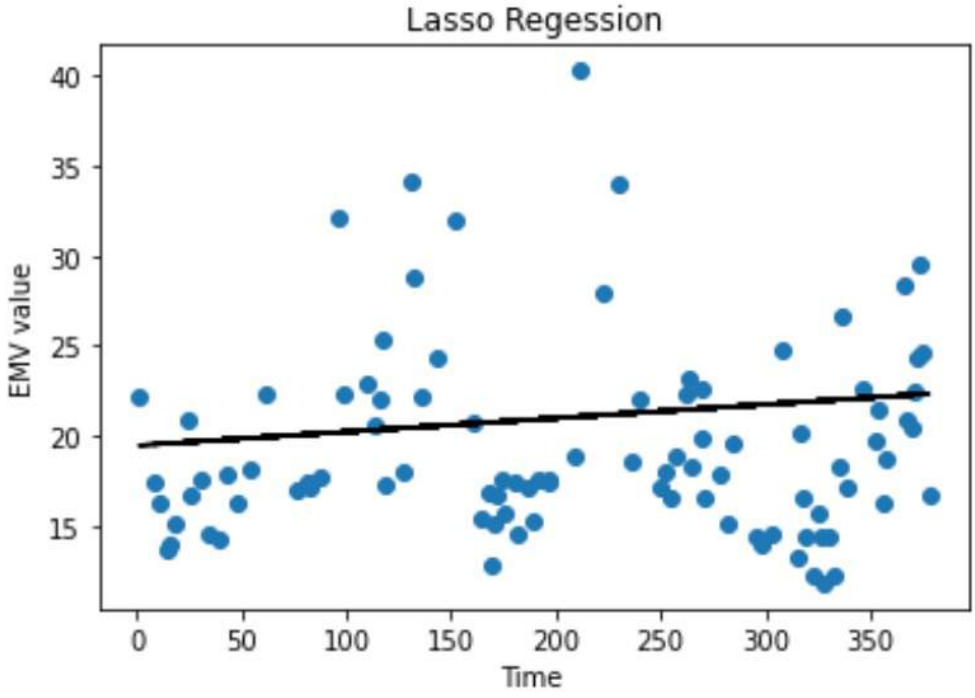

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

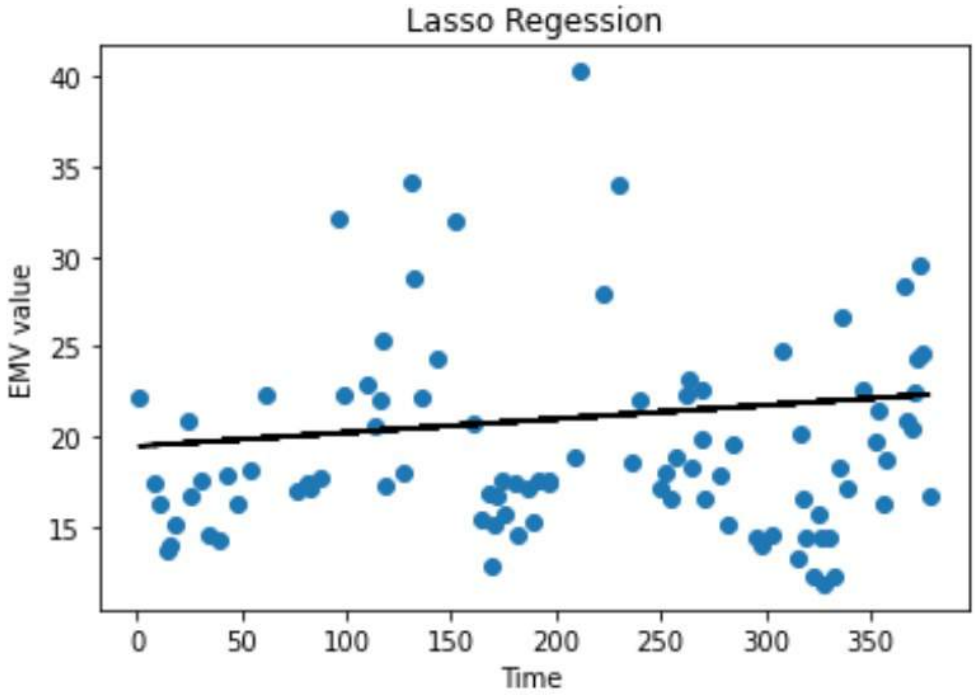

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

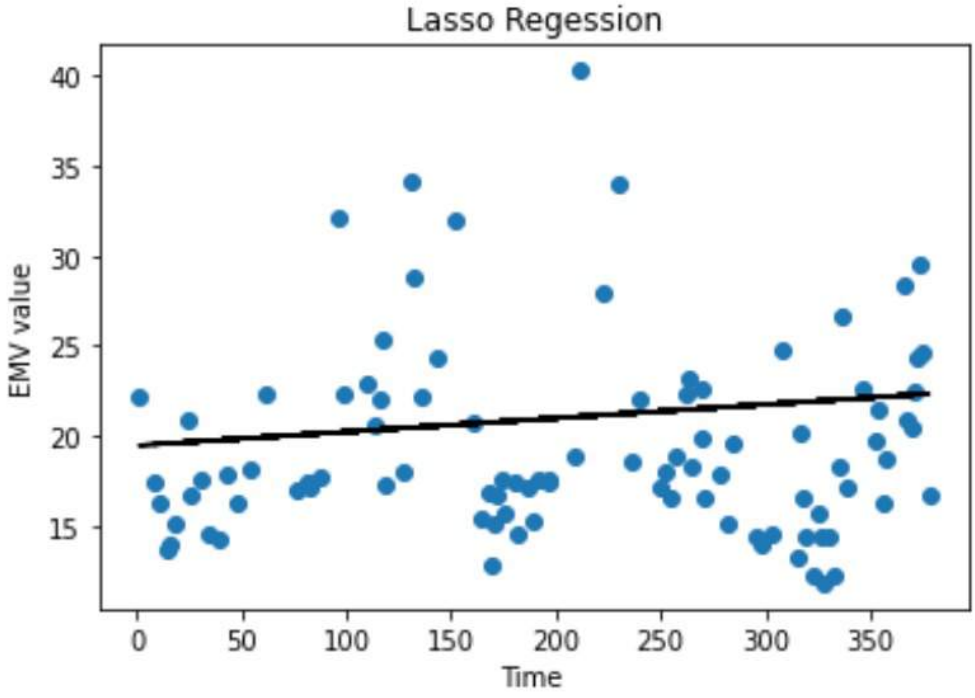

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

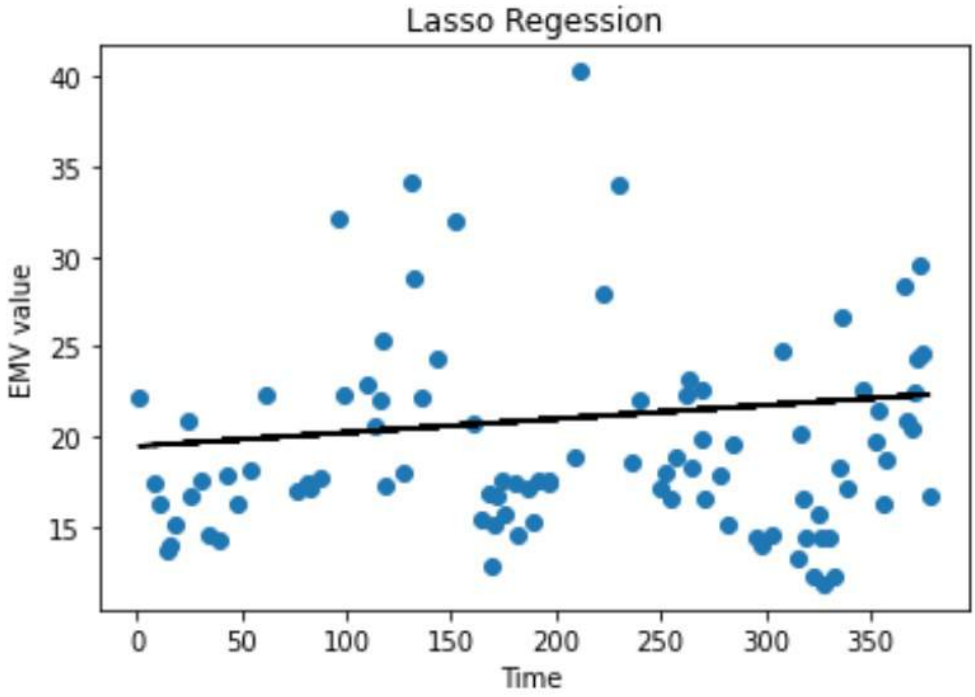

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

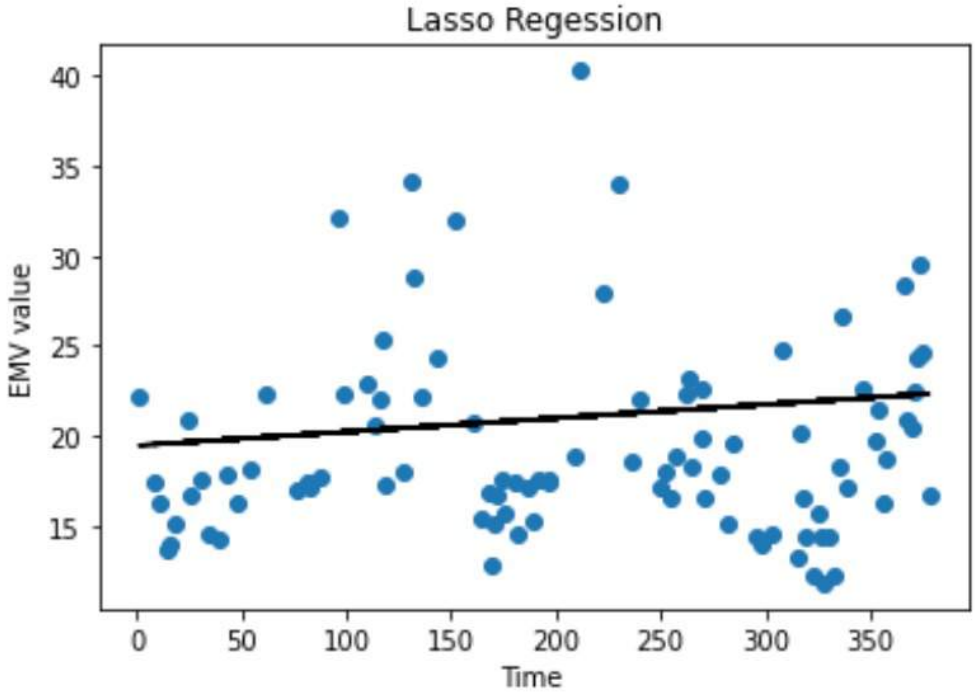

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

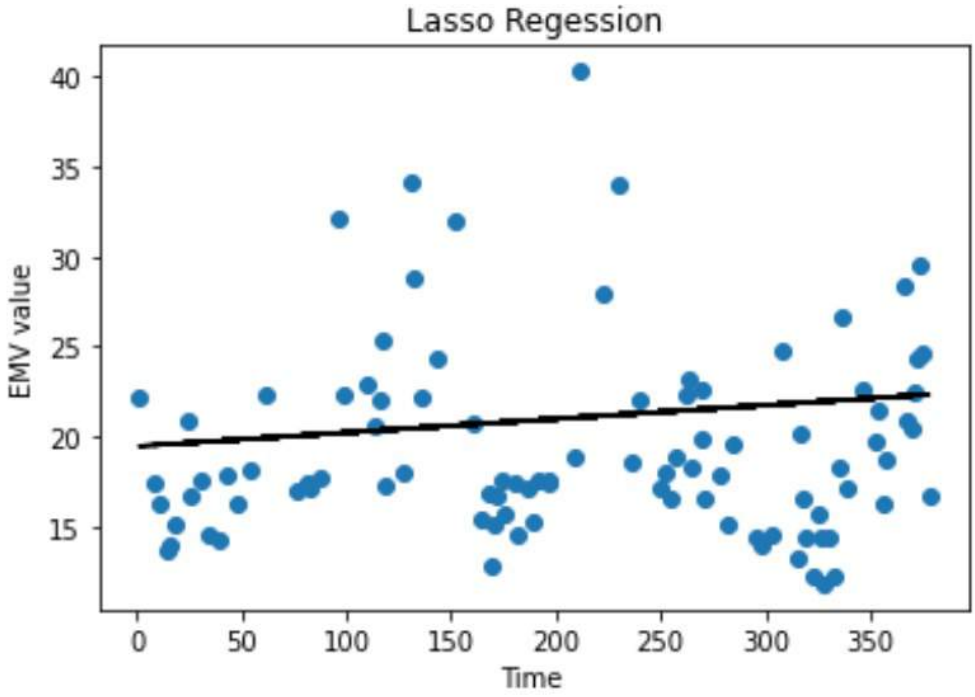

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

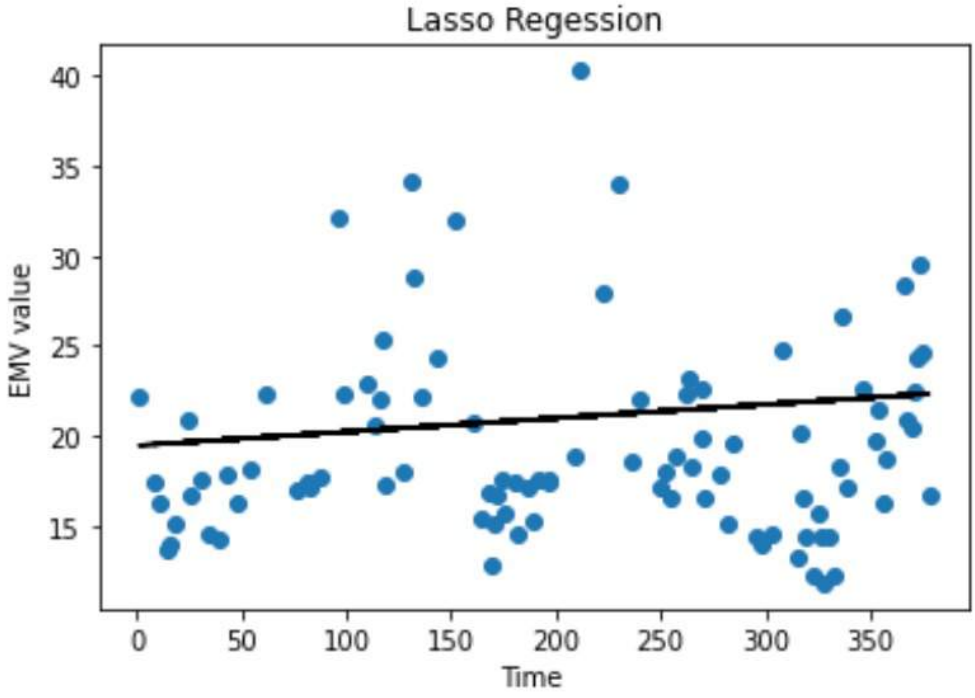

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

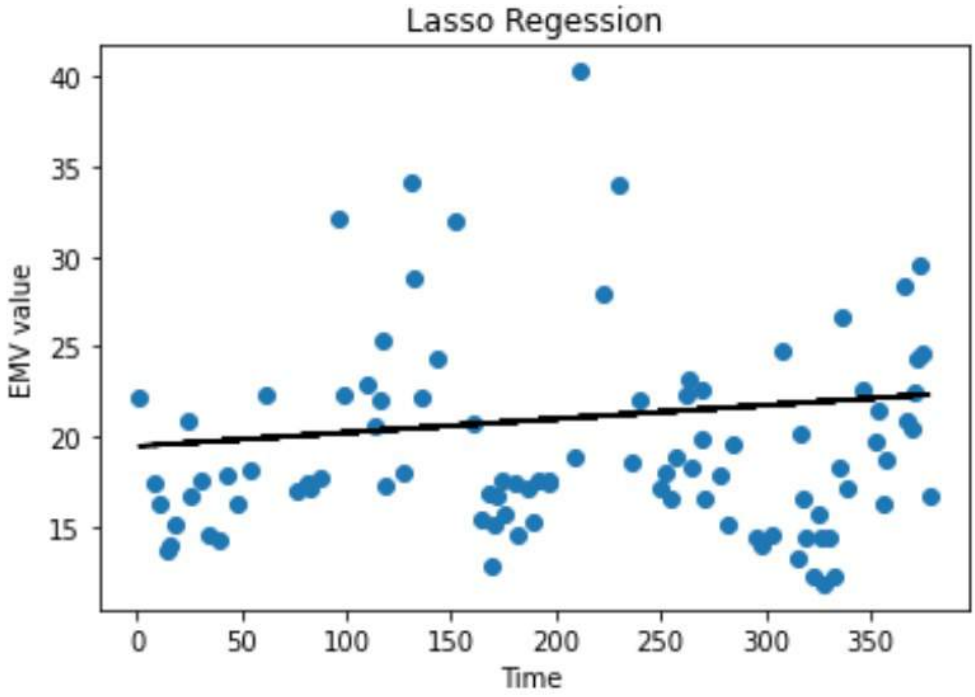

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

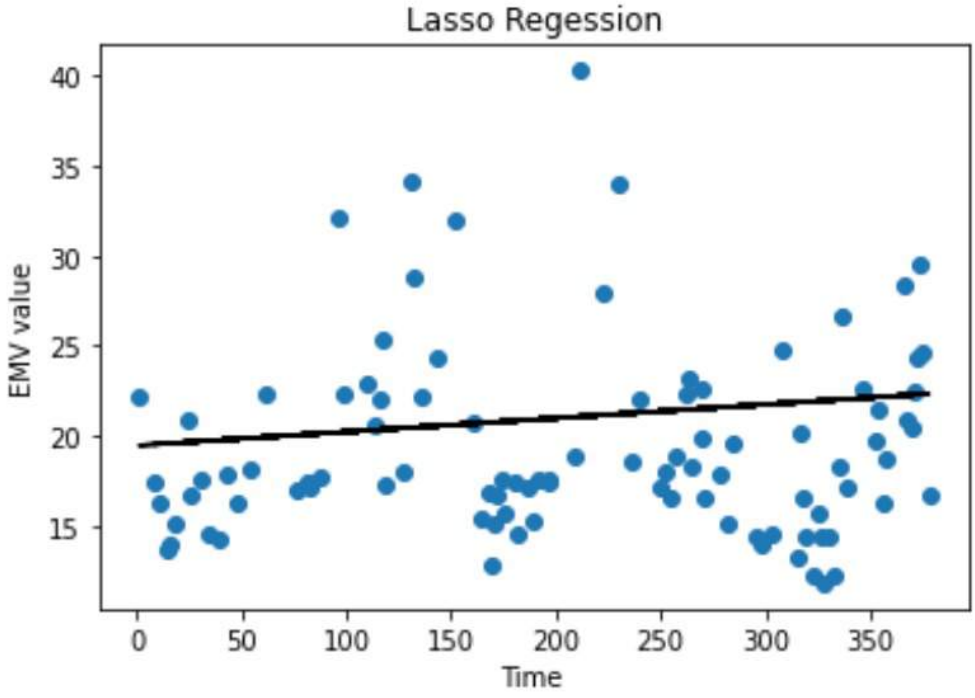

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

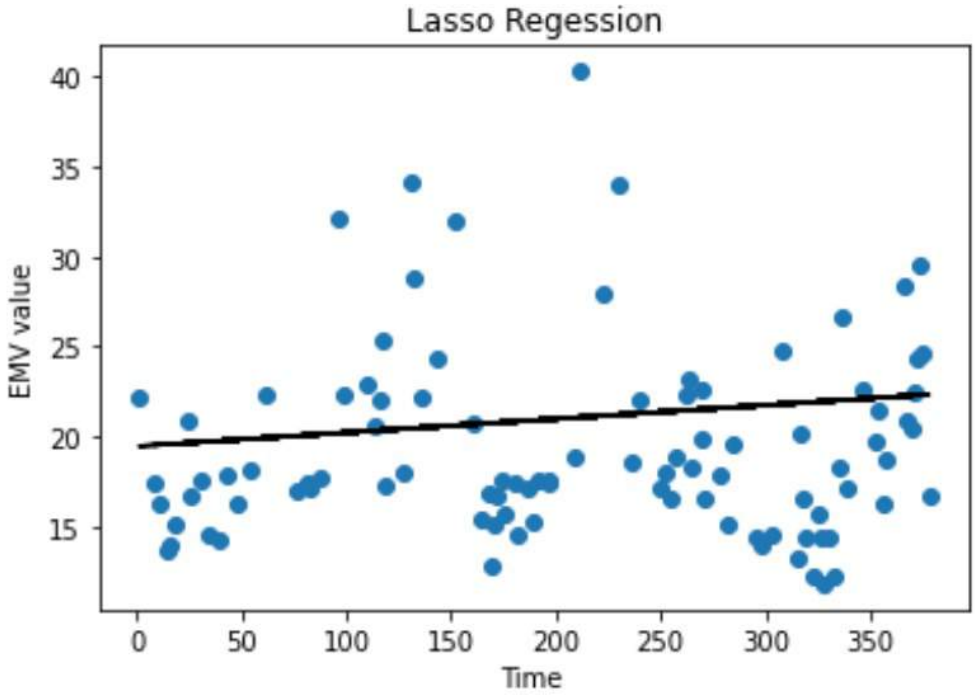

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

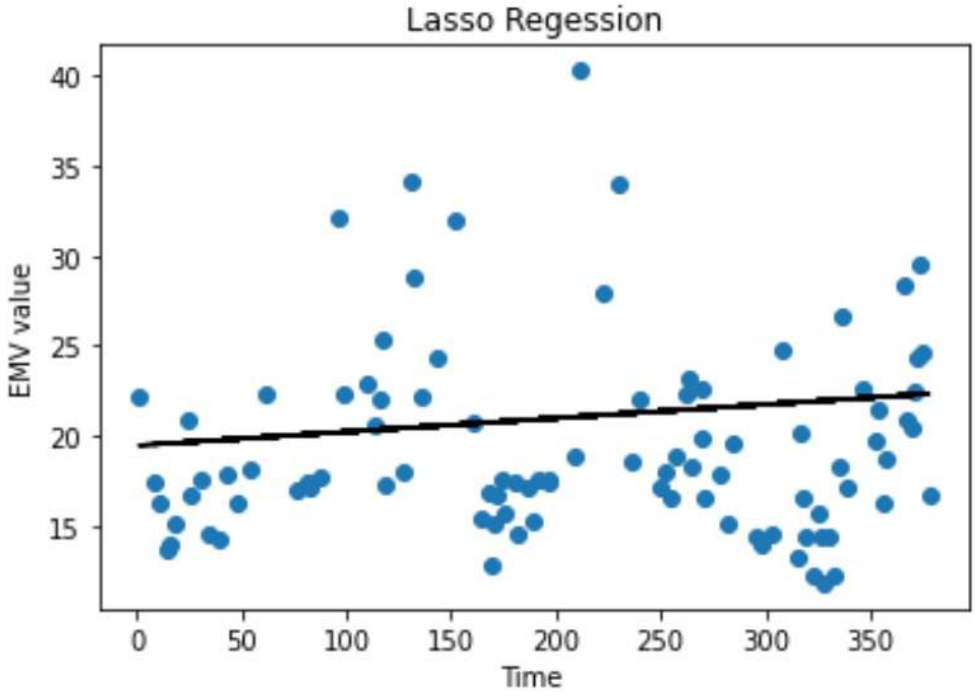

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

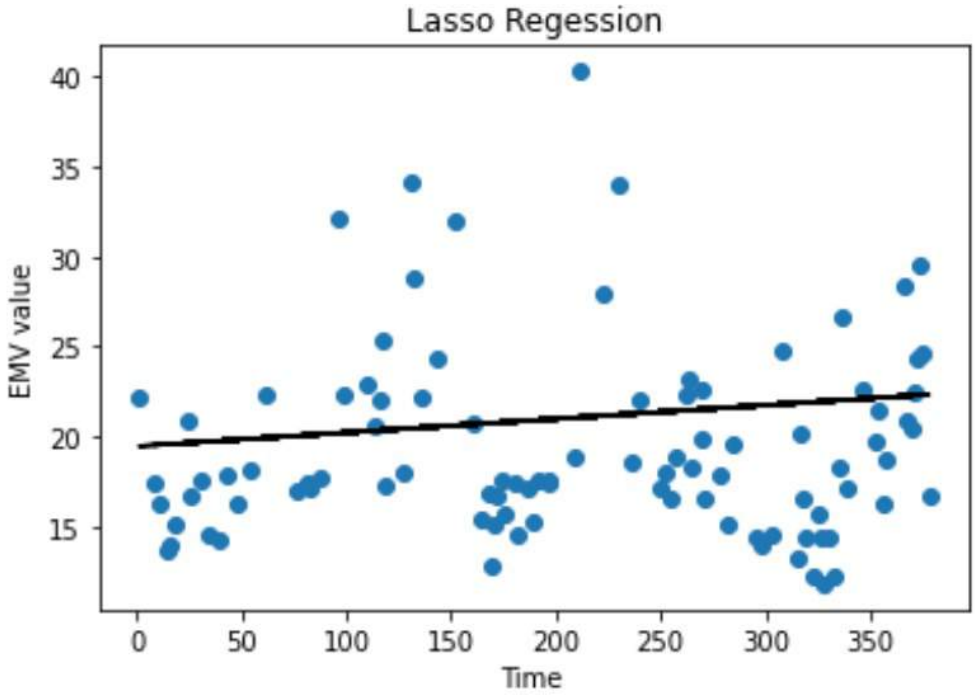

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

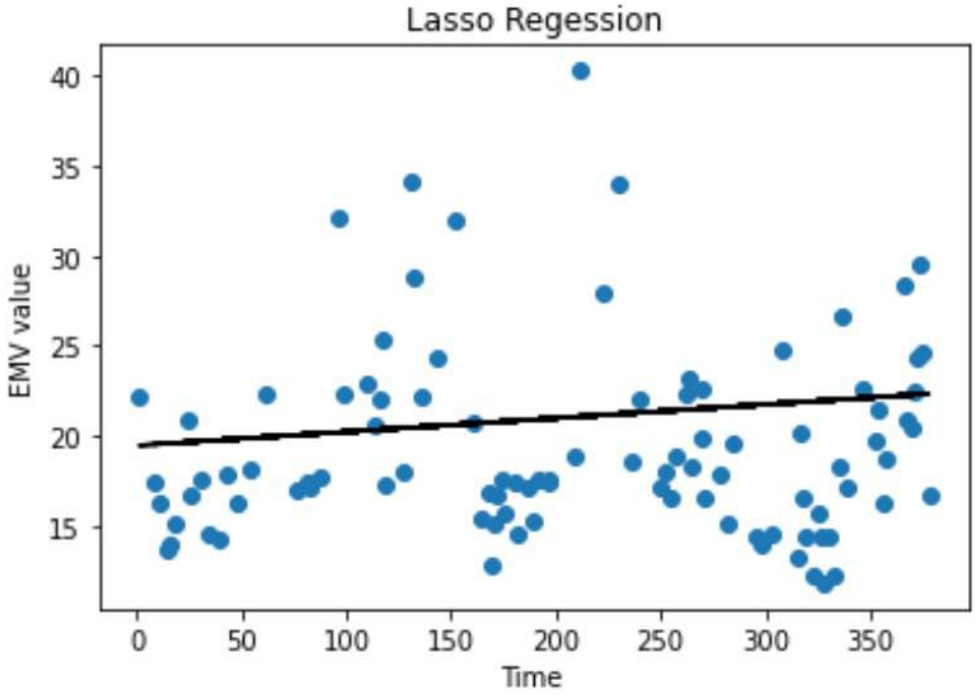

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

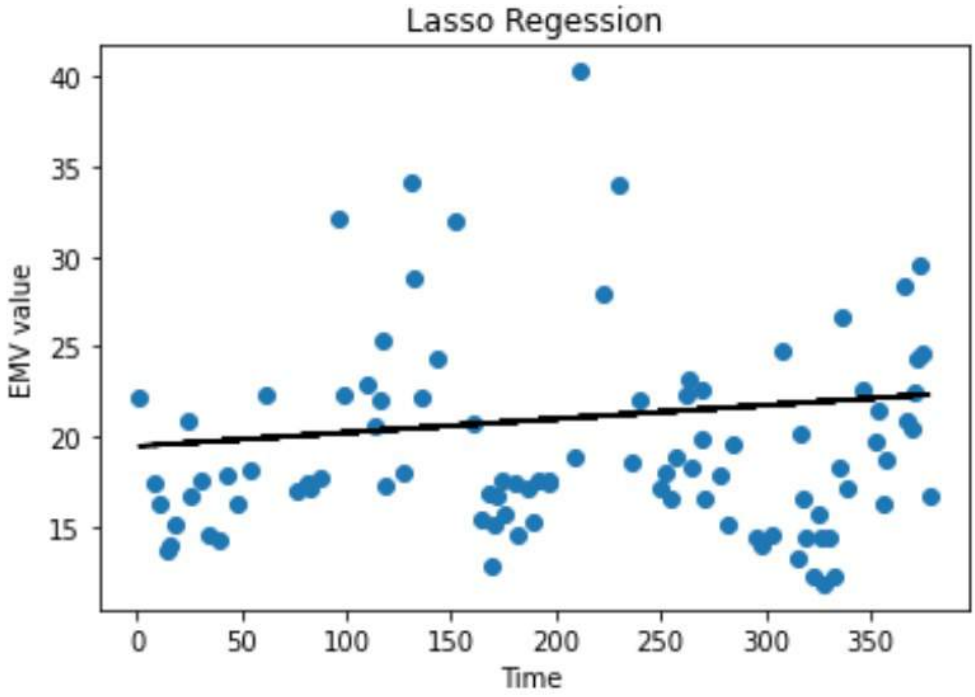

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

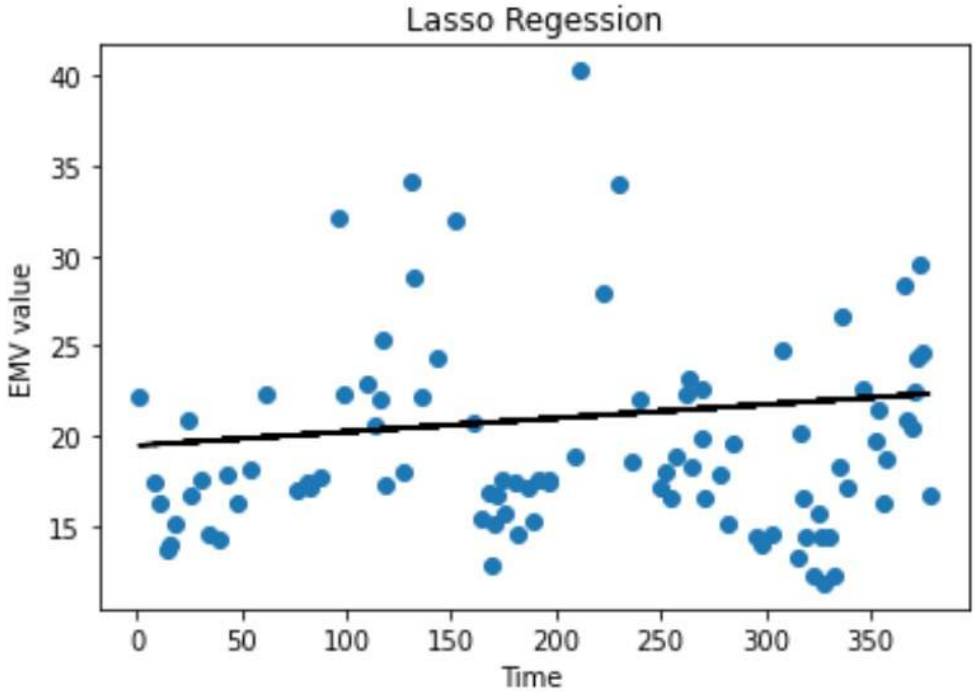

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

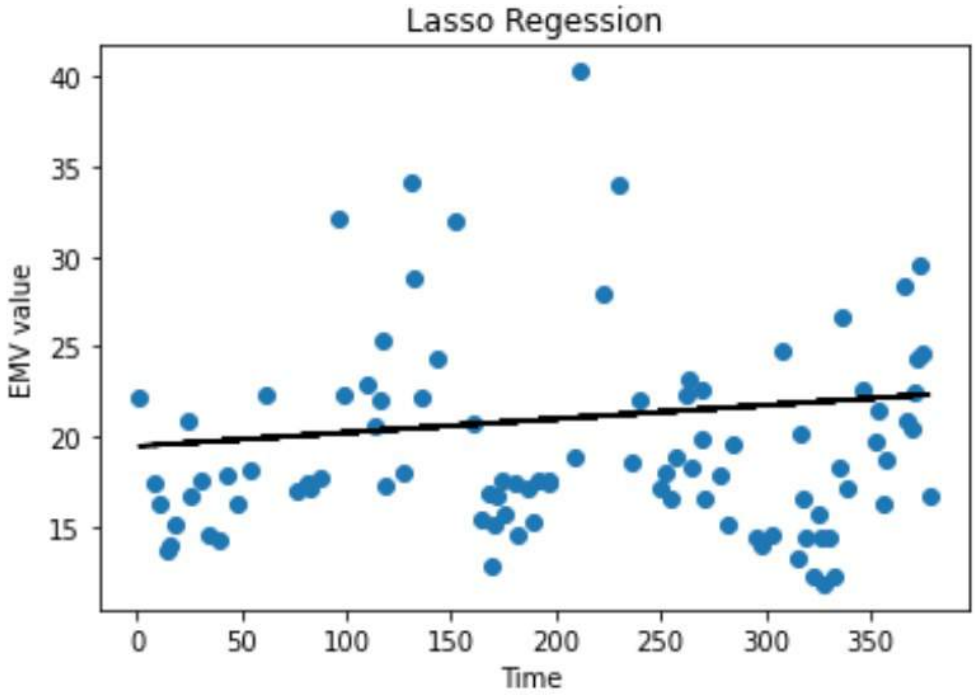

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

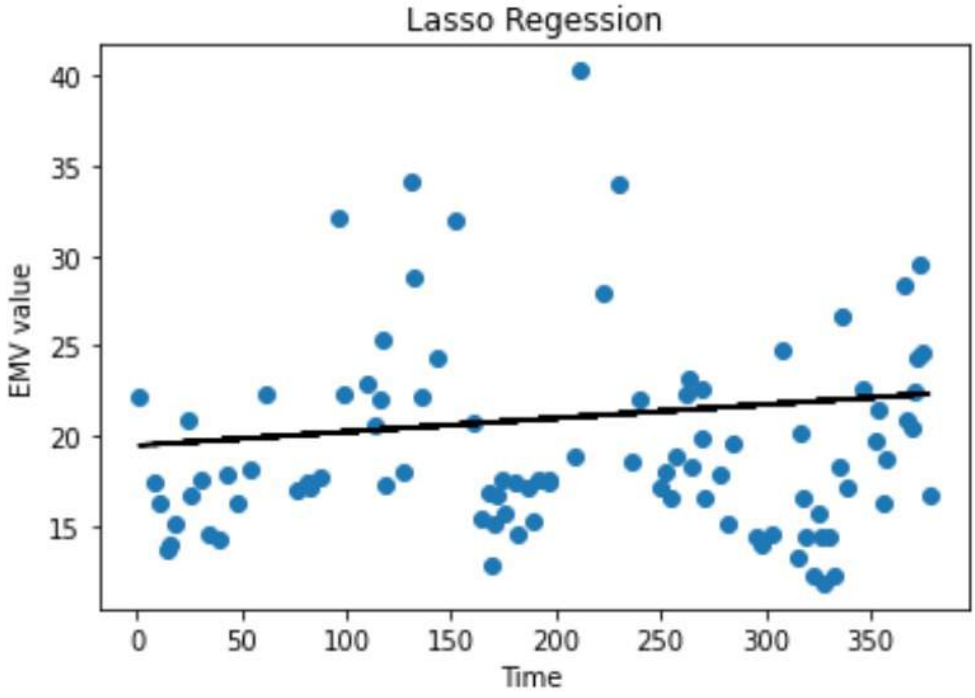

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

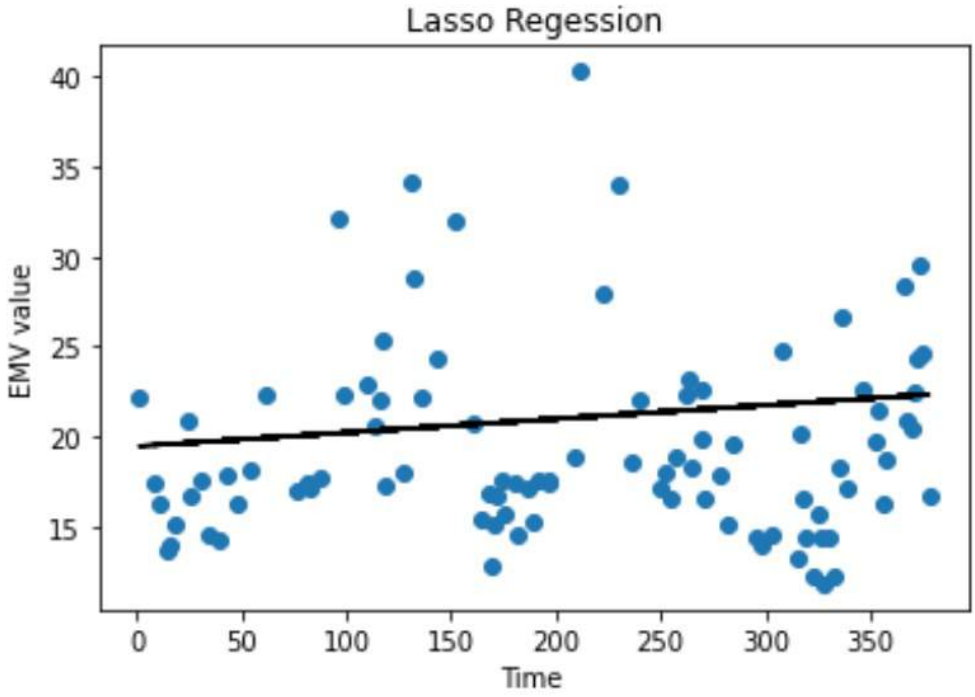

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

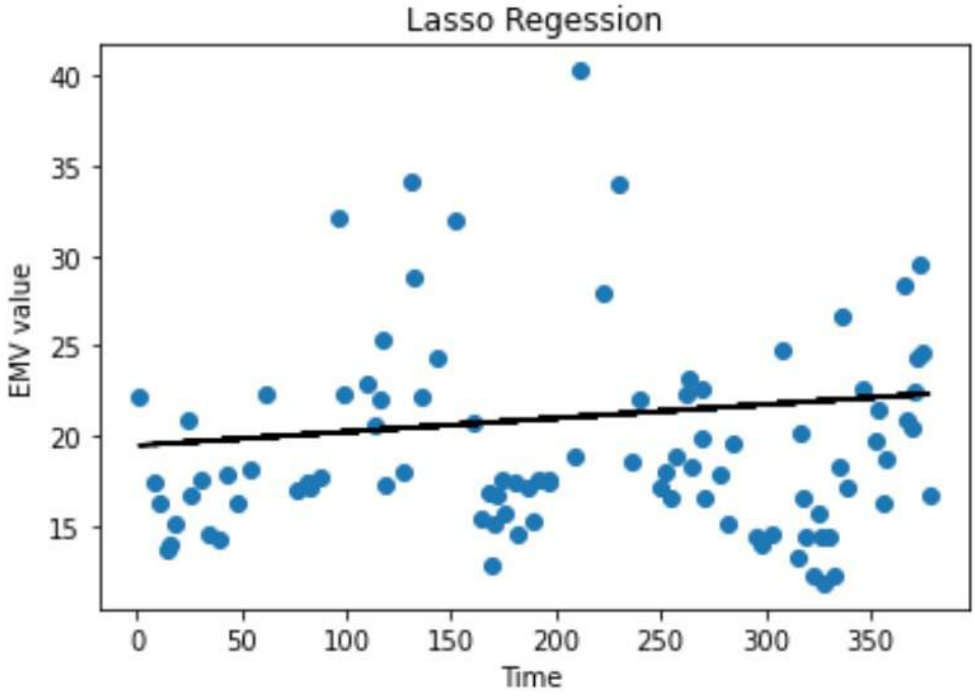

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

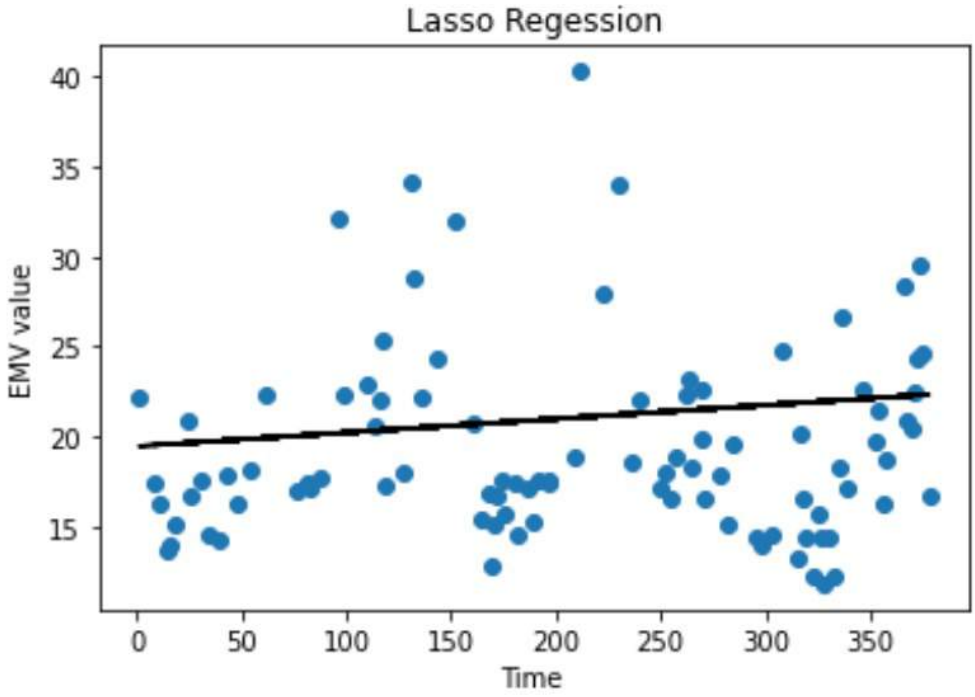

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

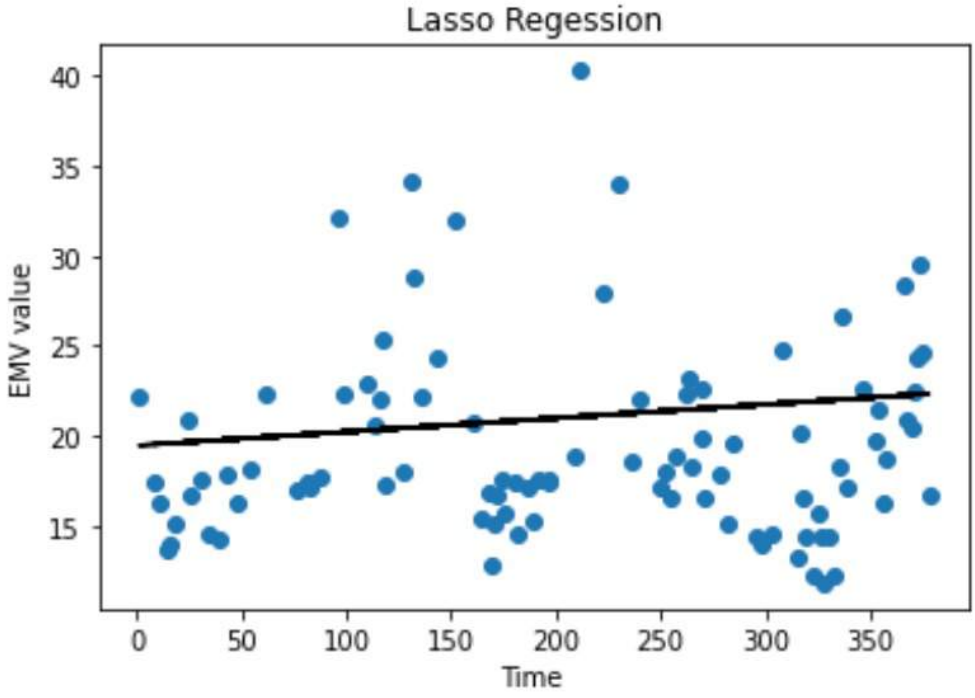

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

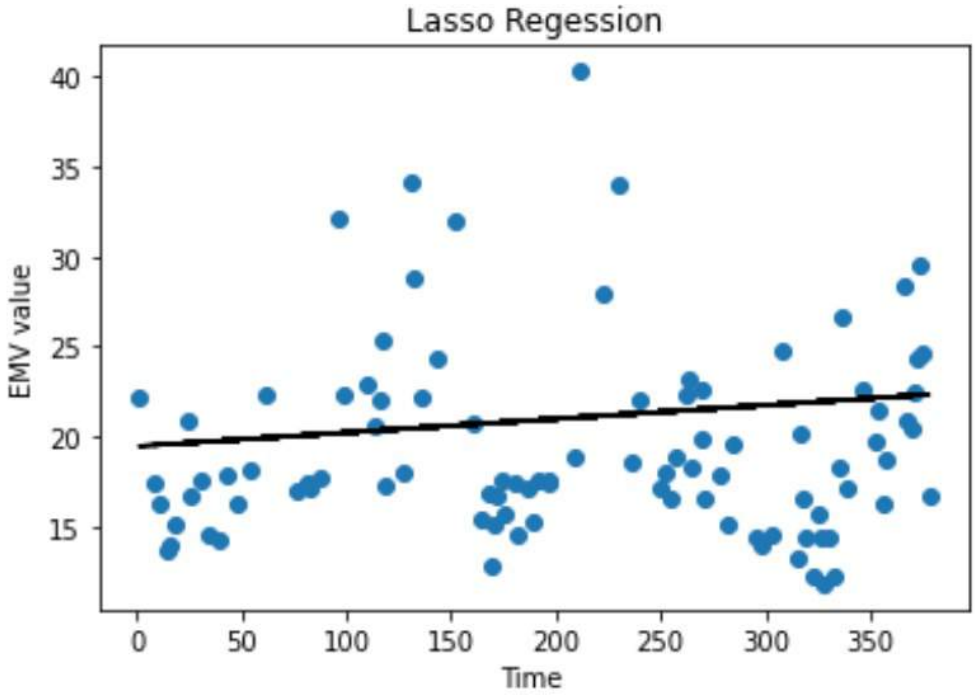

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

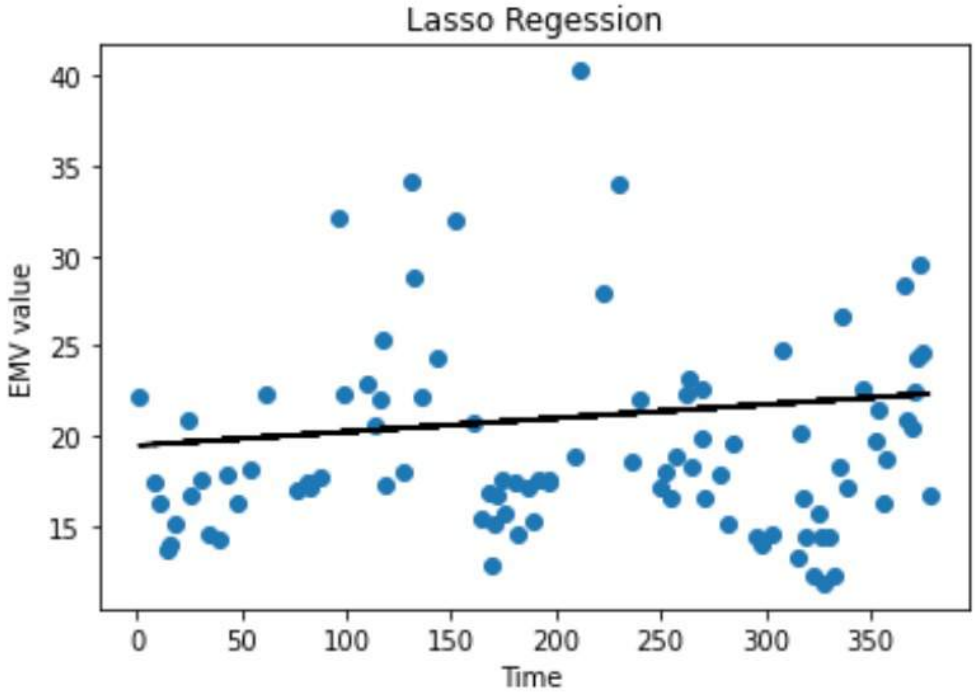

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

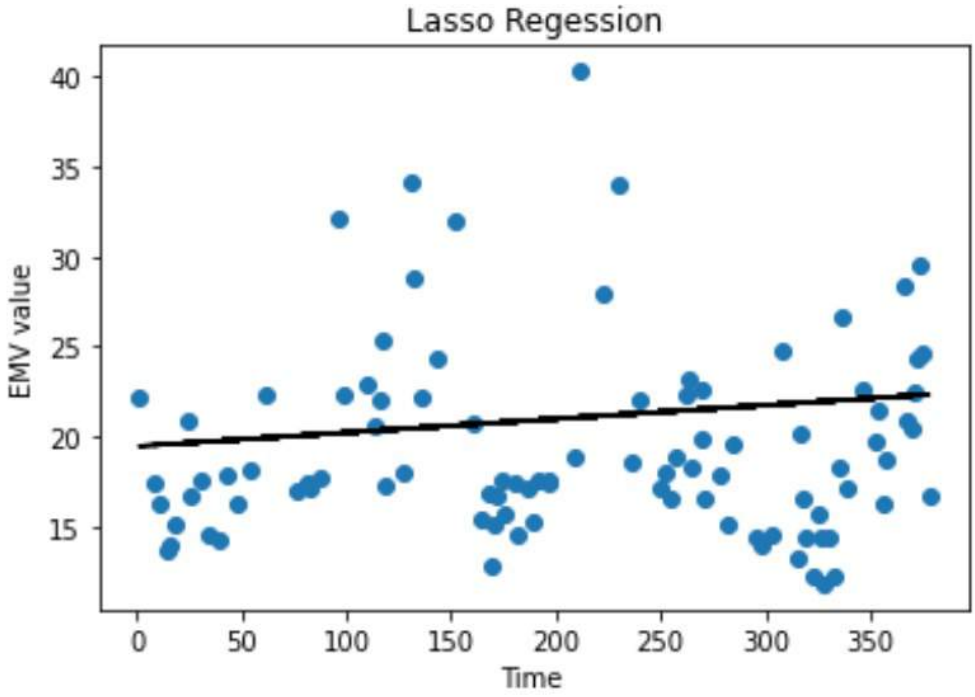

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

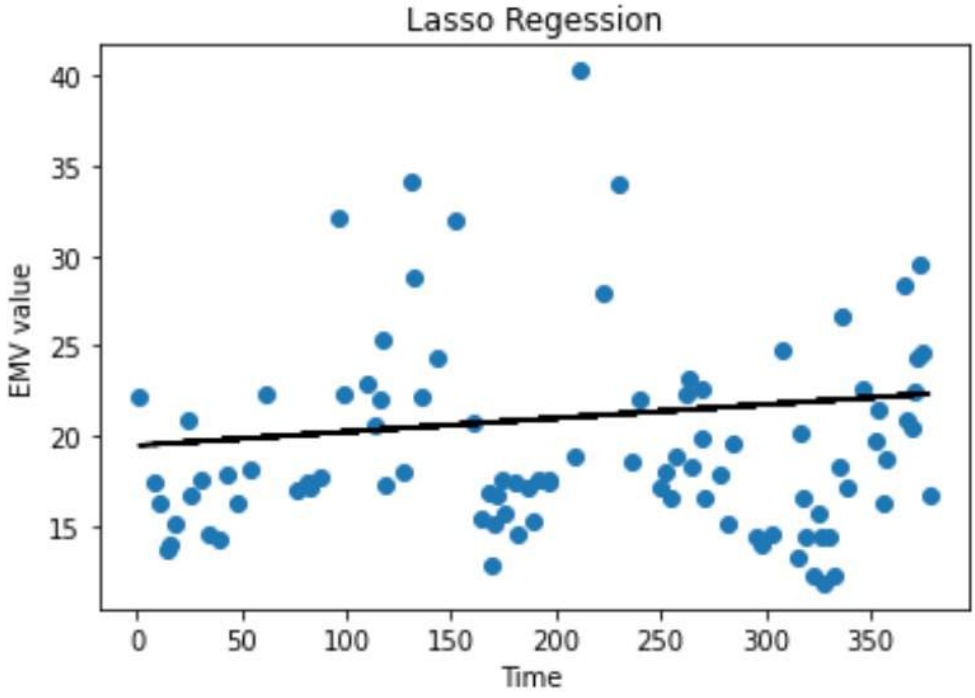

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

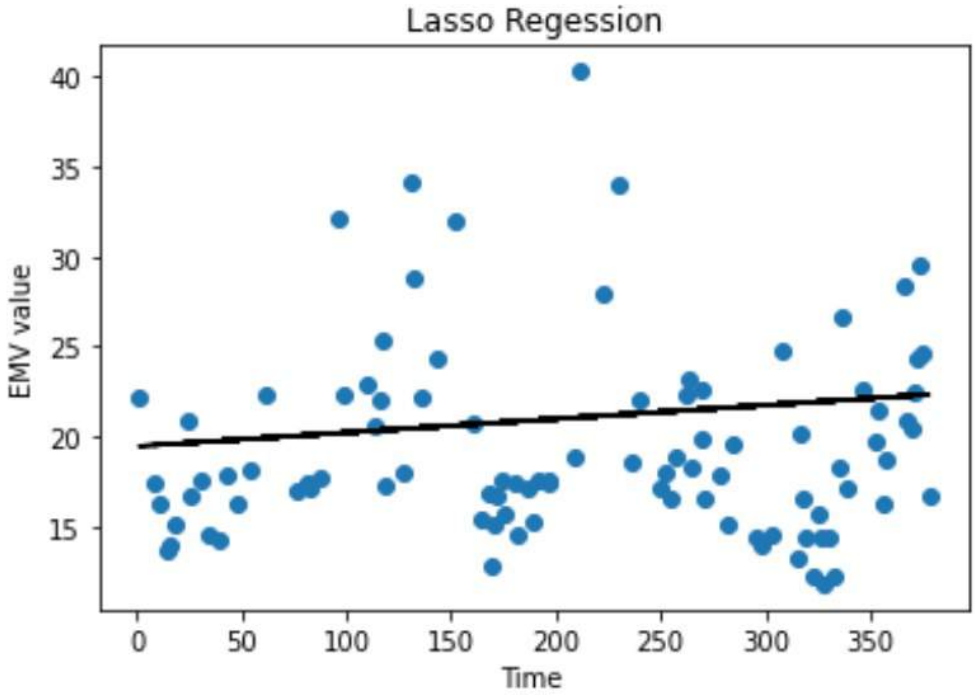

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

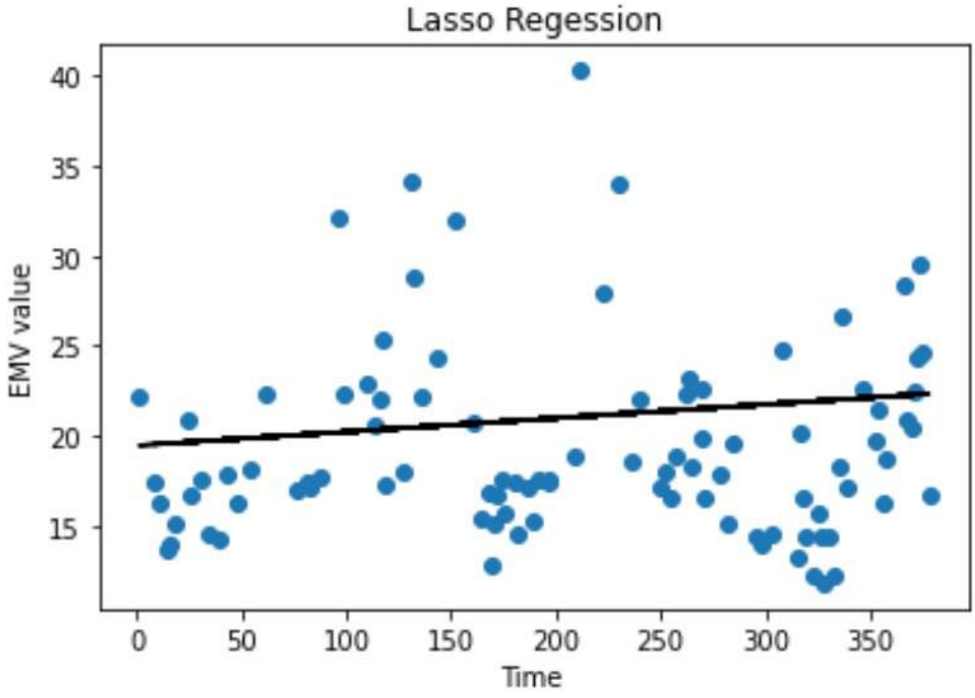

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

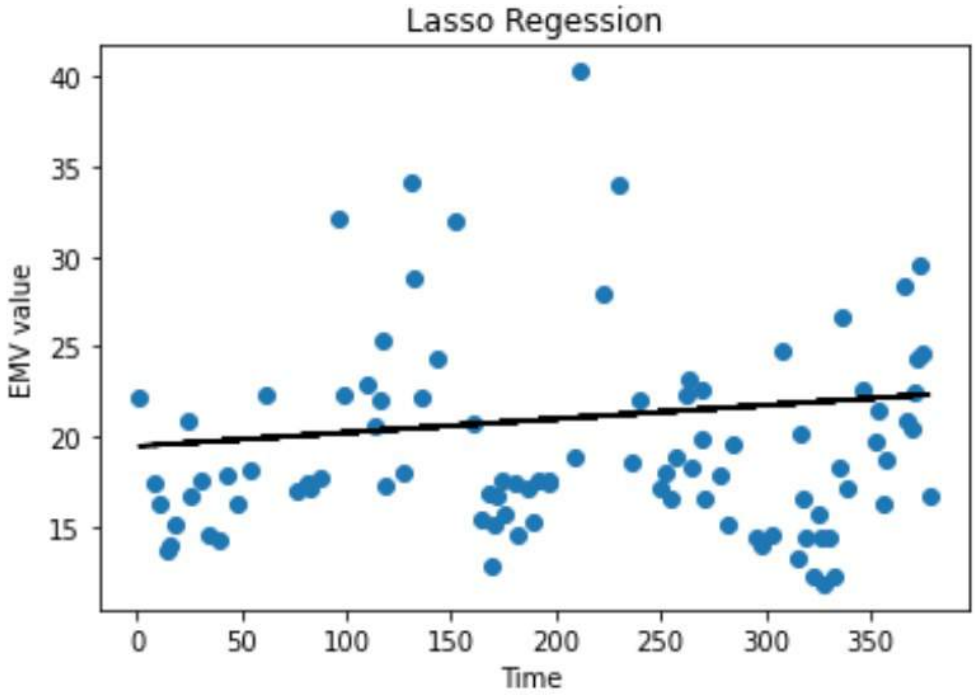

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

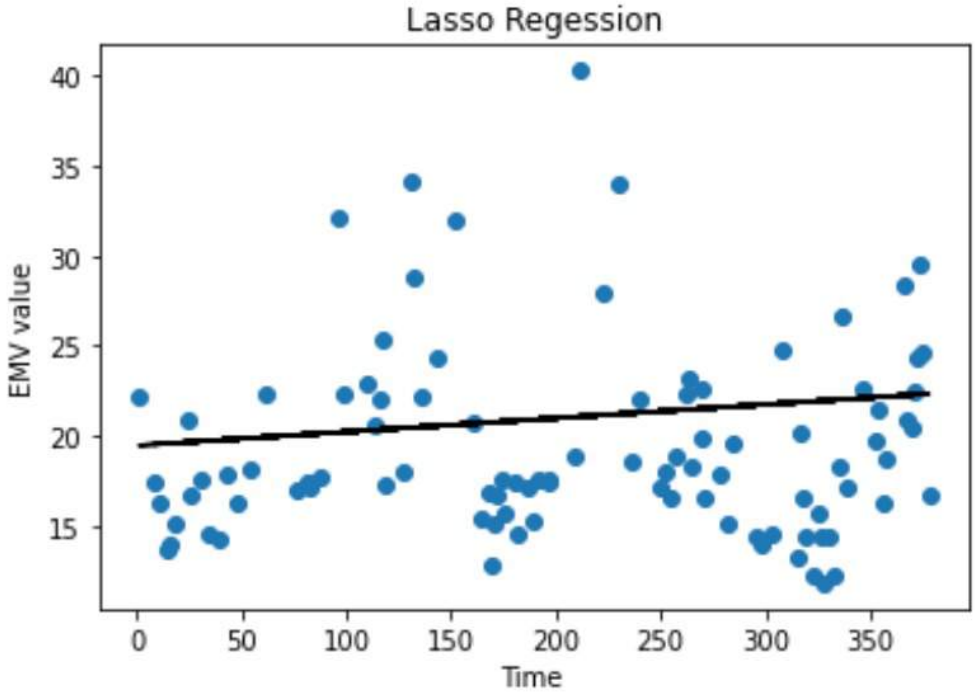

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

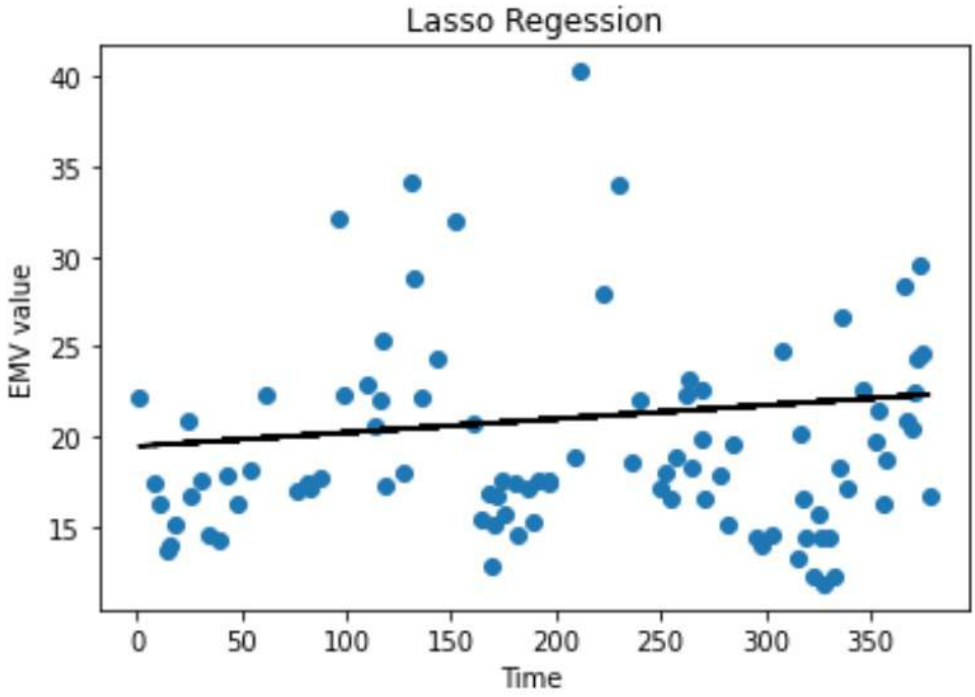

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

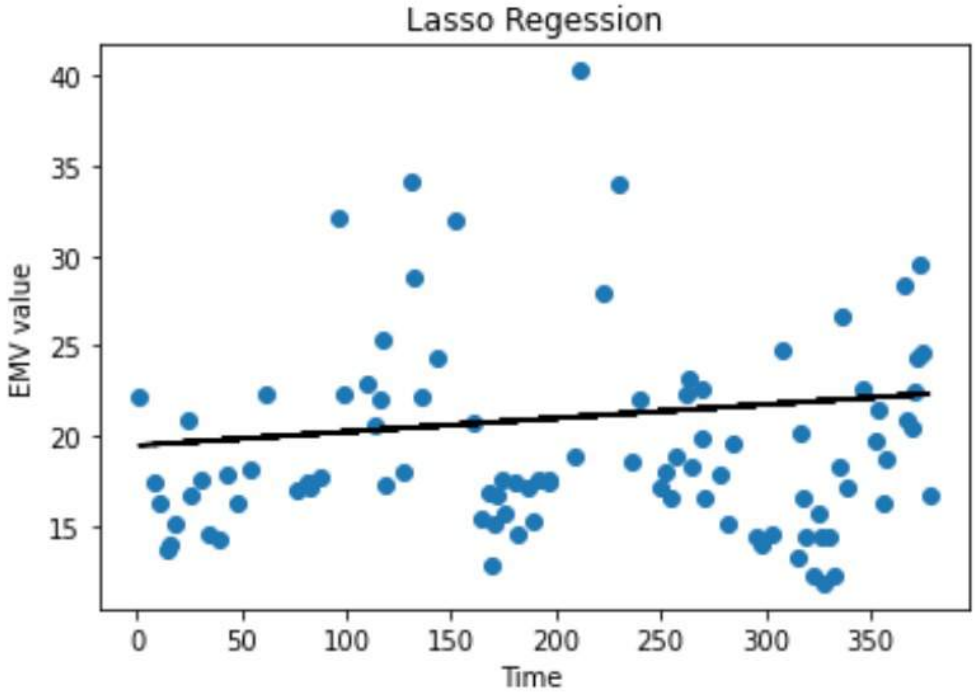

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

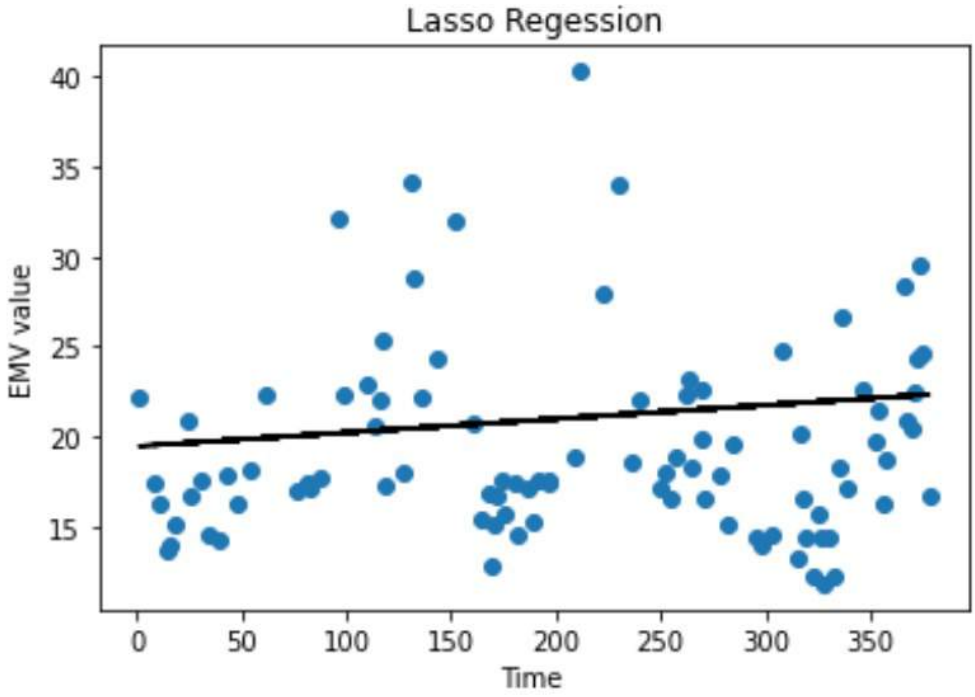

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

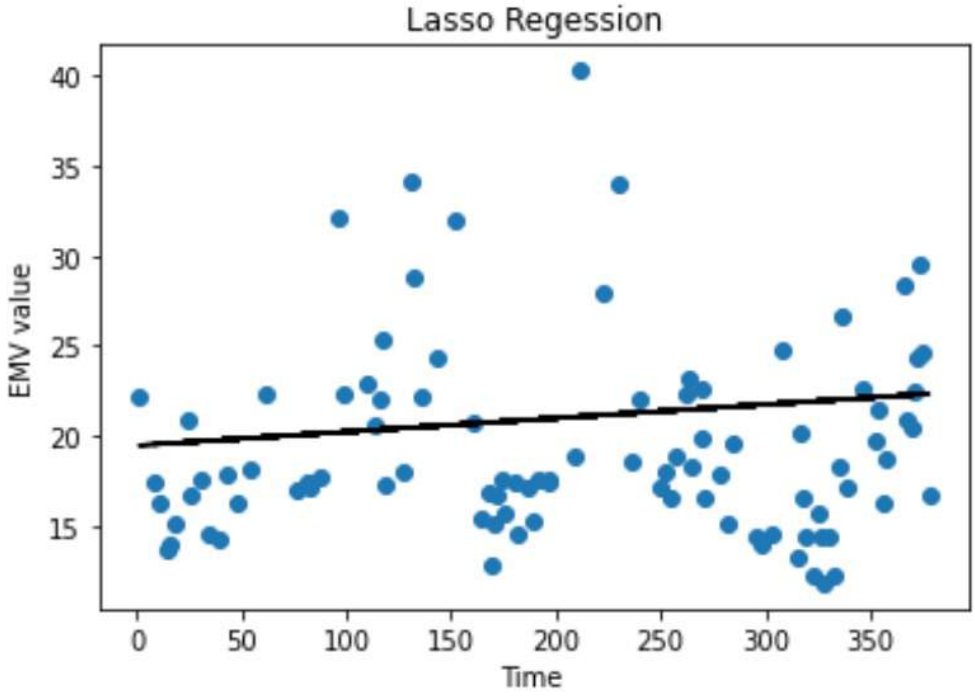

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

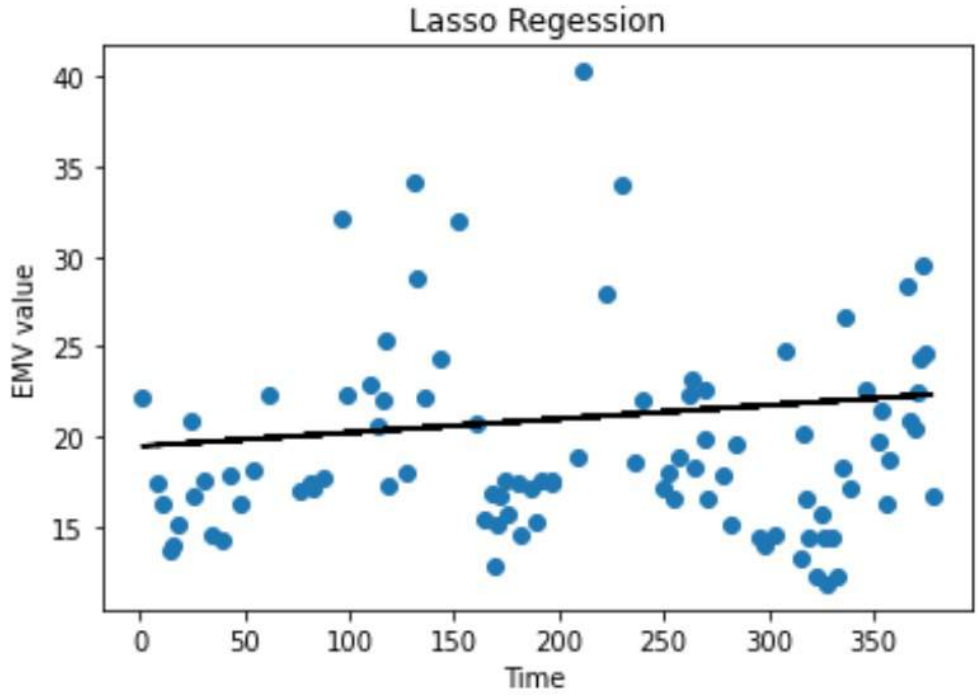

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350


Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

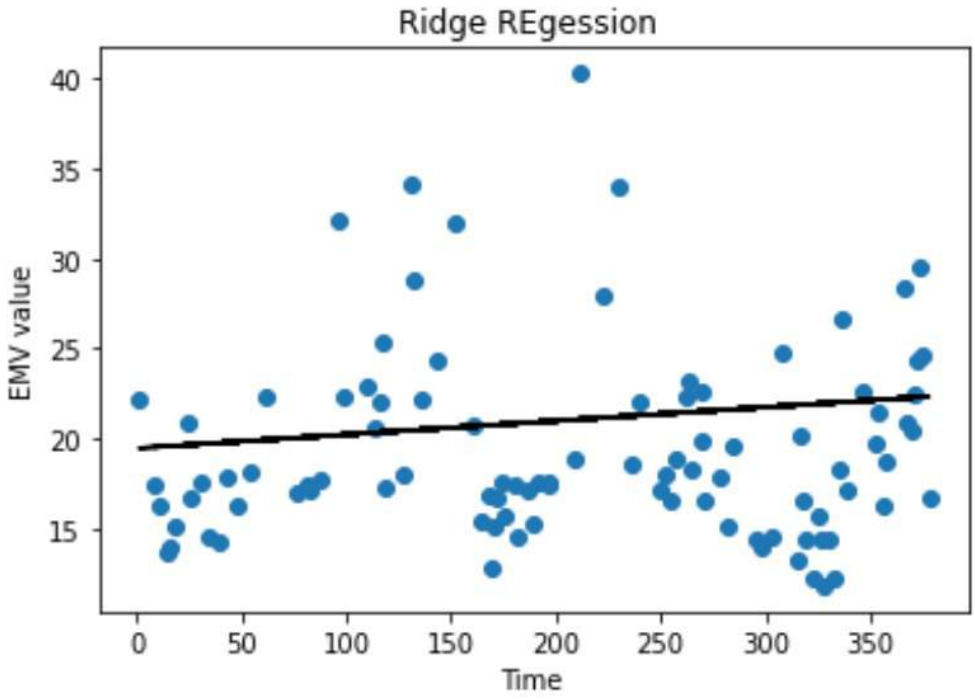

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

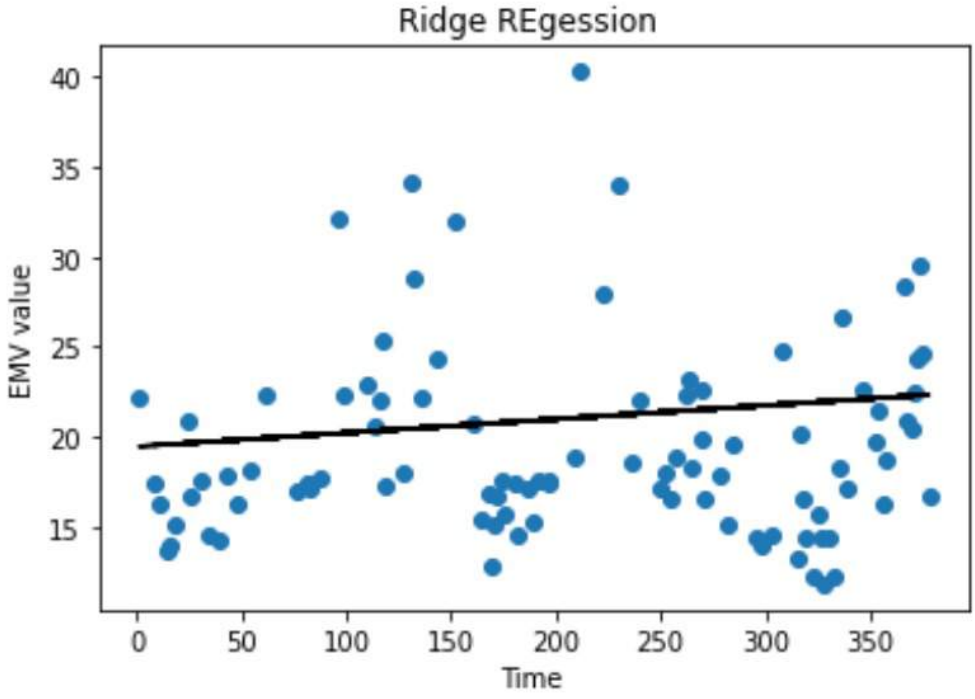

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

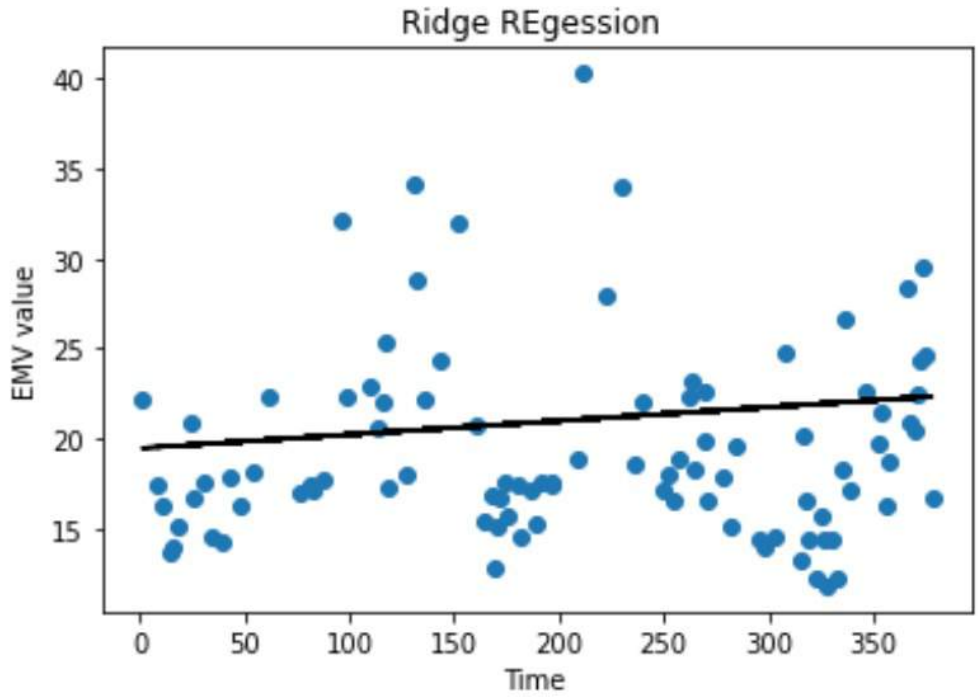

Lasso Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

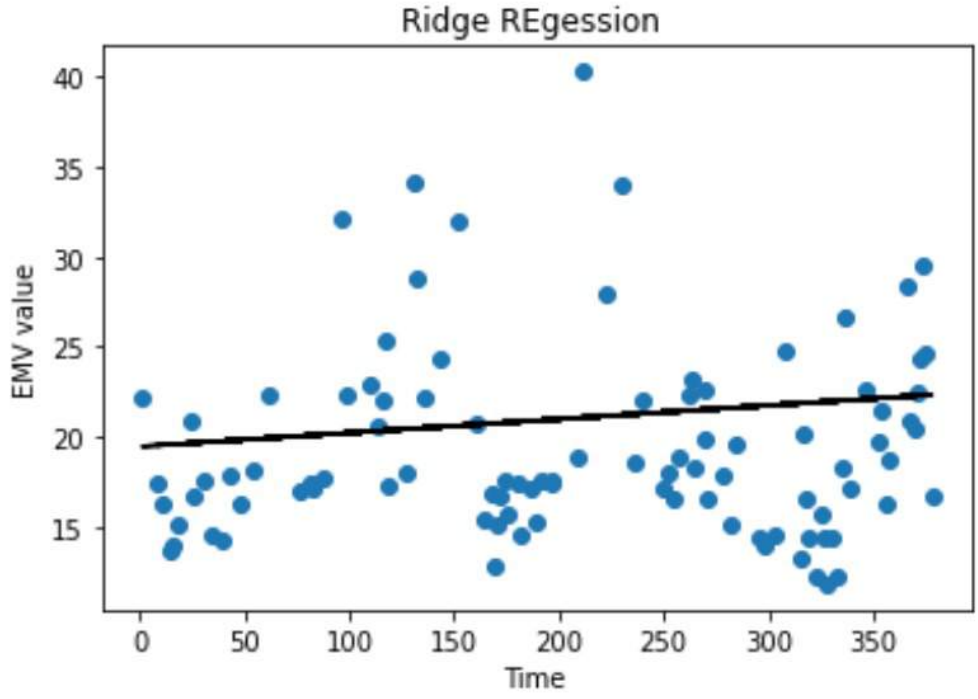


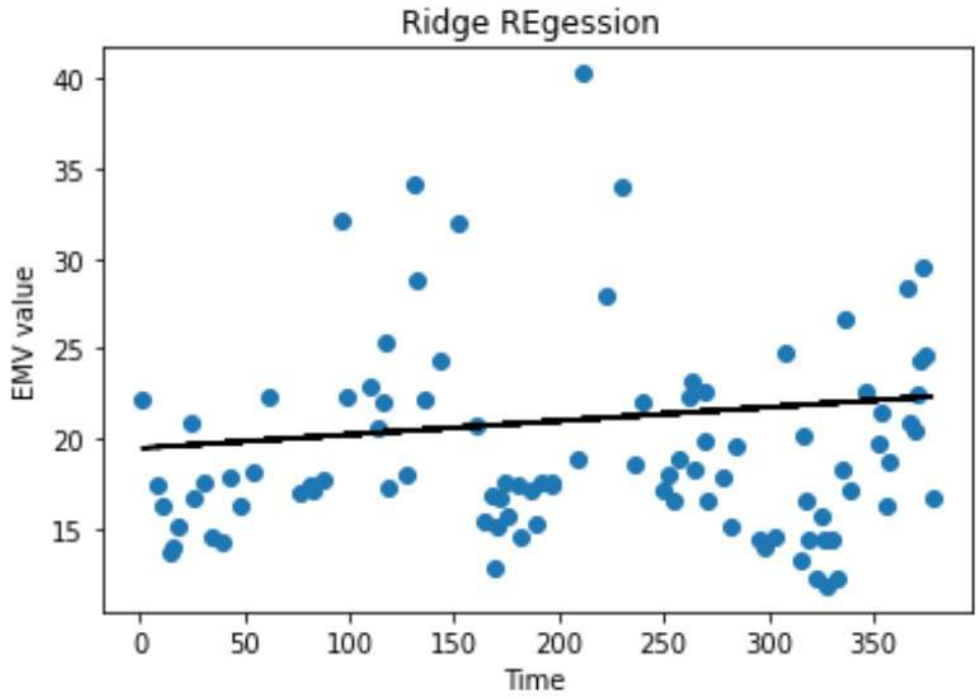

```
def Ridge_reg(train_X,train_Y,test_X,test_Y):
    reg=Ridge(alpha=A)
    reg.fit(train_X, train_Y)
    pred_Y=reg.predict(test_X)

    for tracker in range(45):
        plt.scatter(test_X, [y for y in test_Y])
        plt.plot(test_X,  [y for y in  pred_Y], color='black')
        plt.title("Ridge Regession ")
        plt.xlabel("Time ")
        plt.ylabel("EMV value")


        plt.show()
Ridge_reg(train_X, train_Y, test_X, test_Y)
```

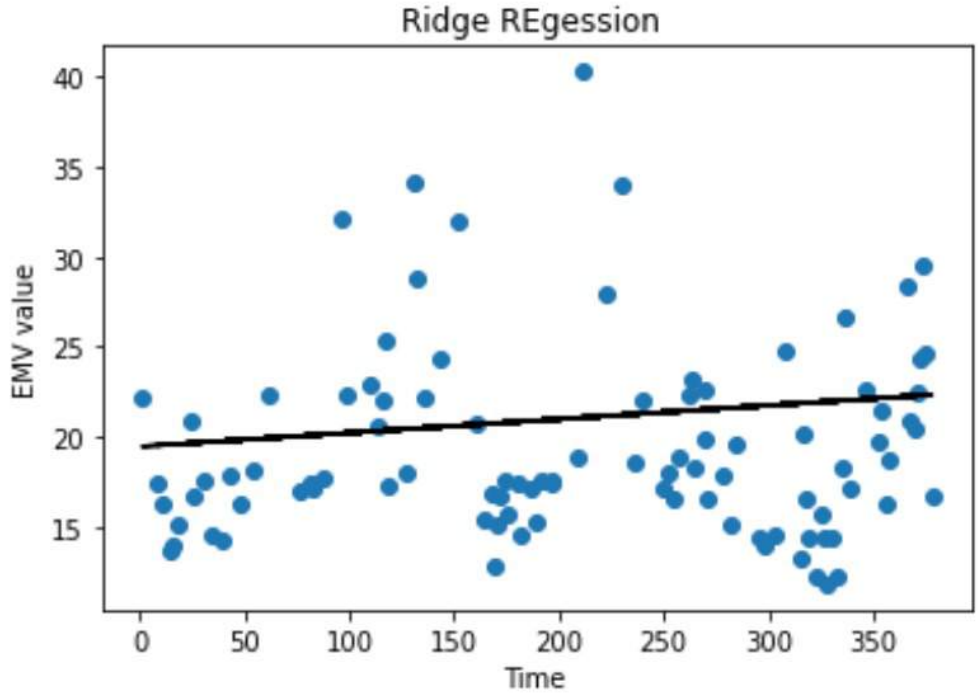

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

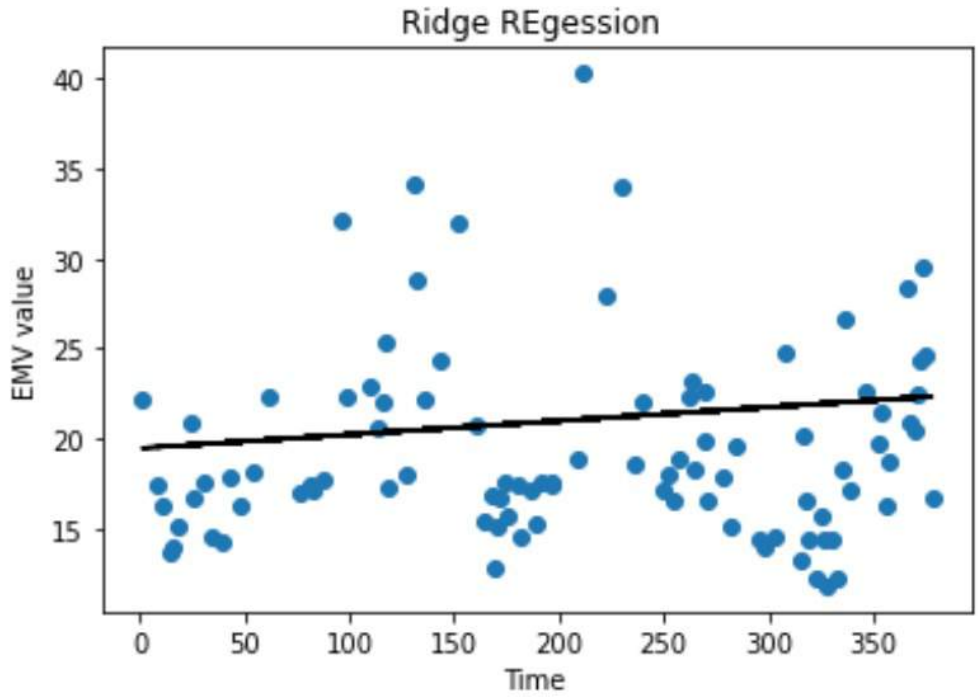

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

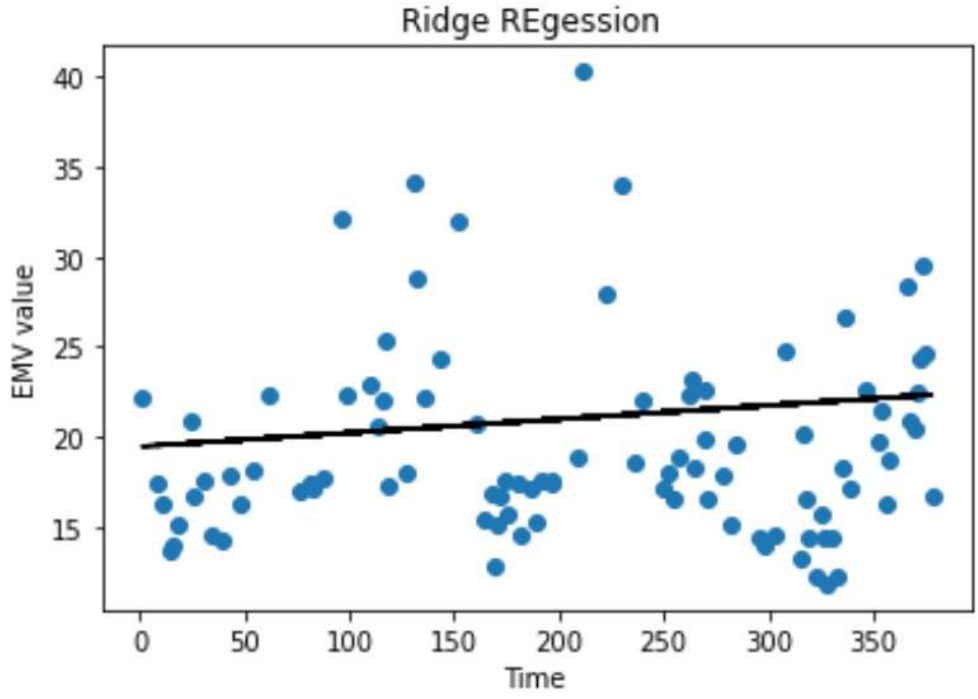

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

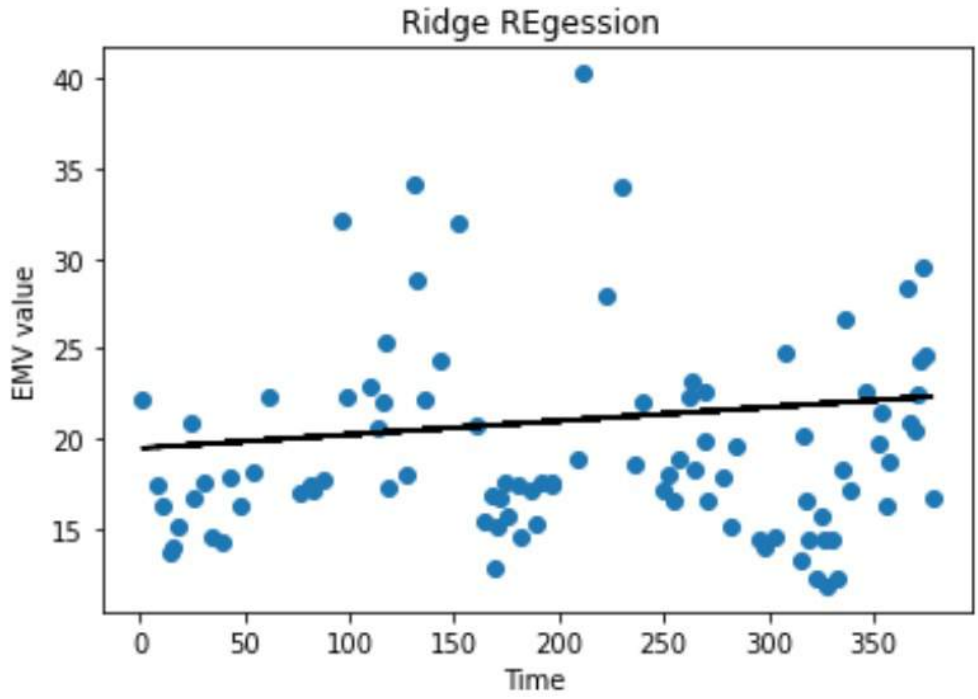

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

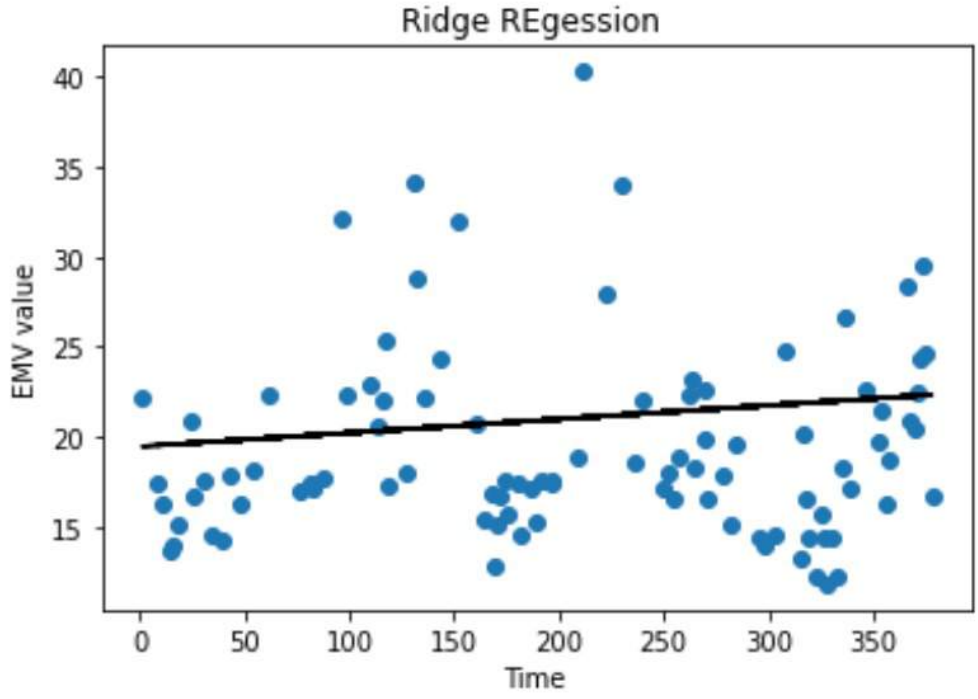

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

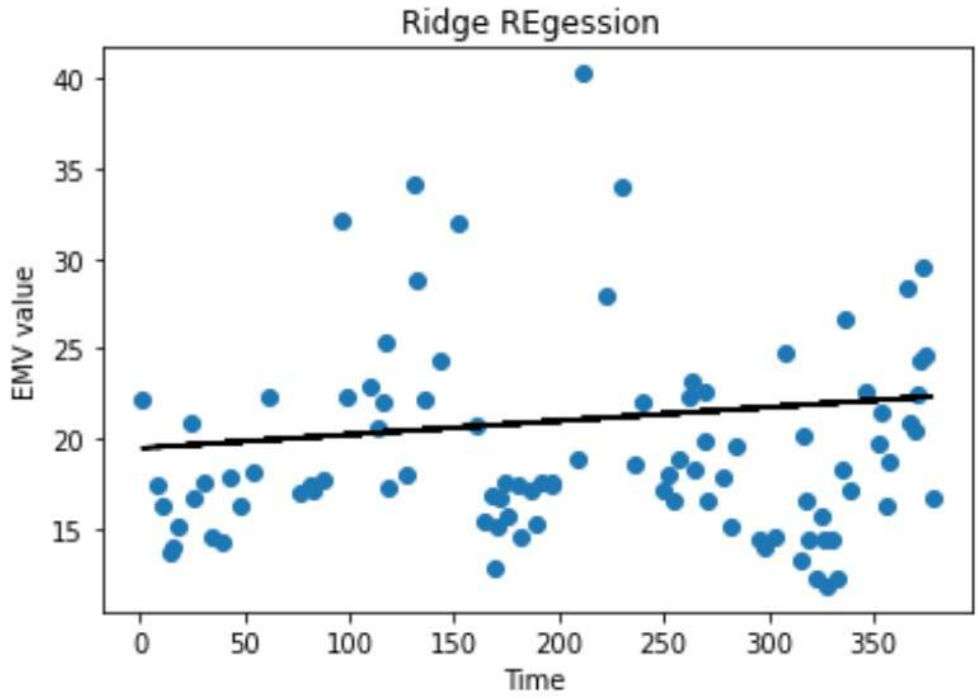

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

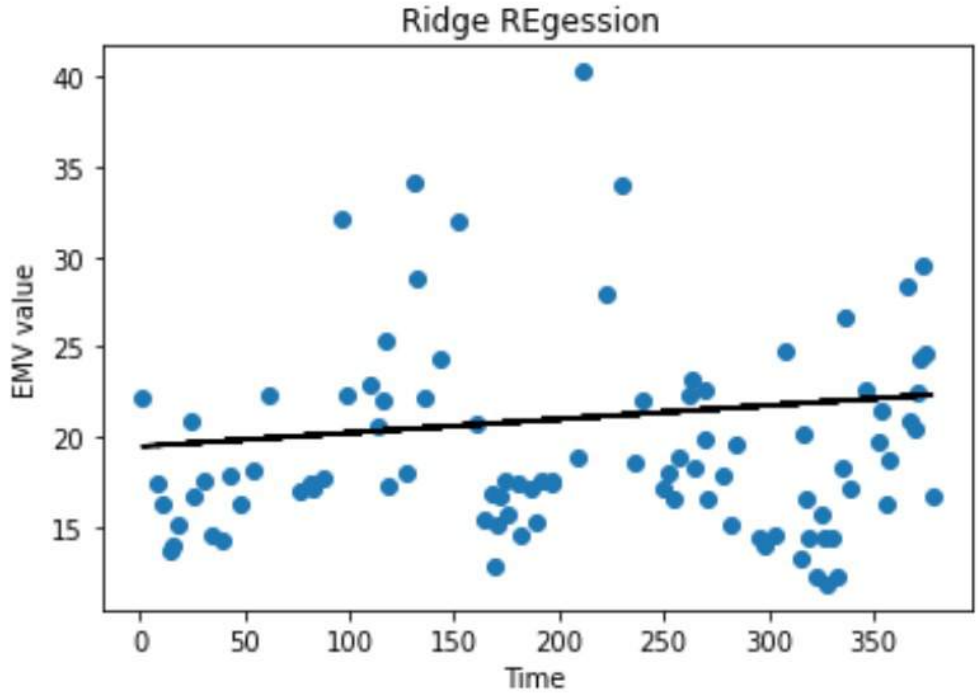

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

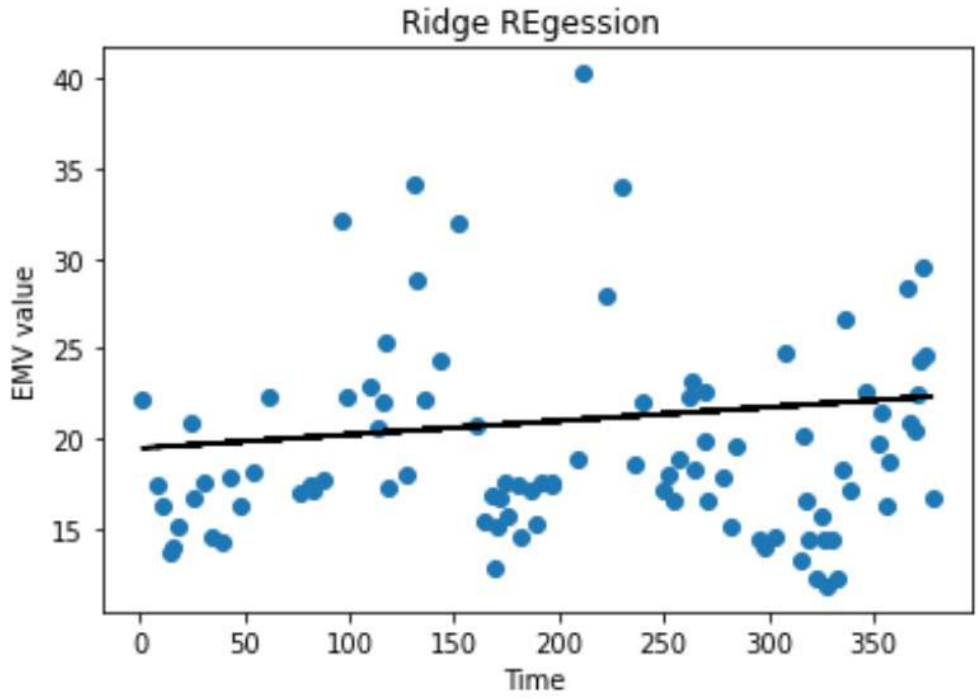

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

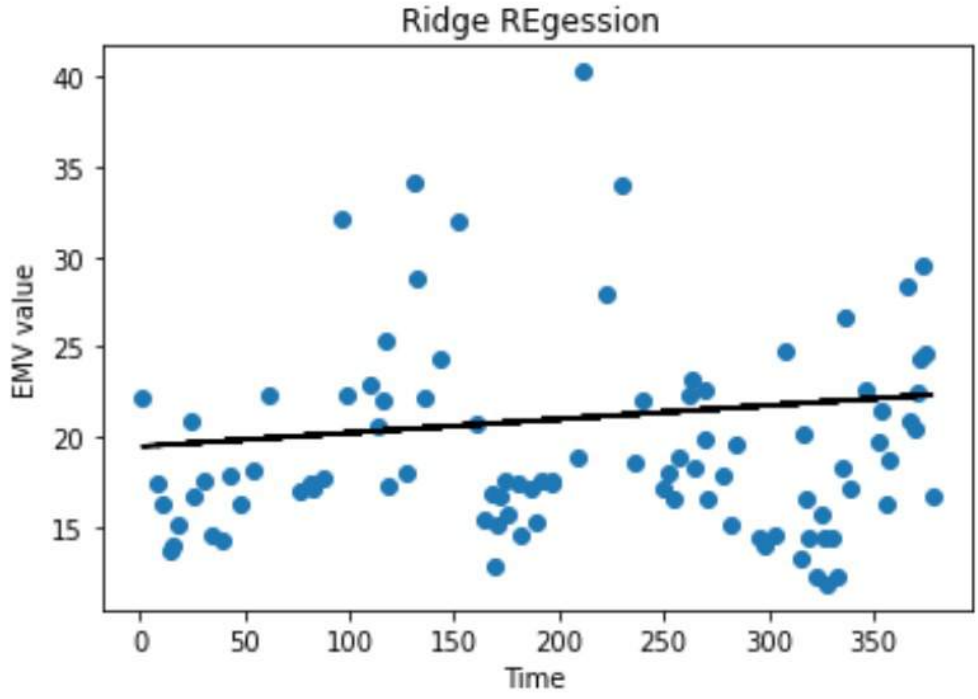

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

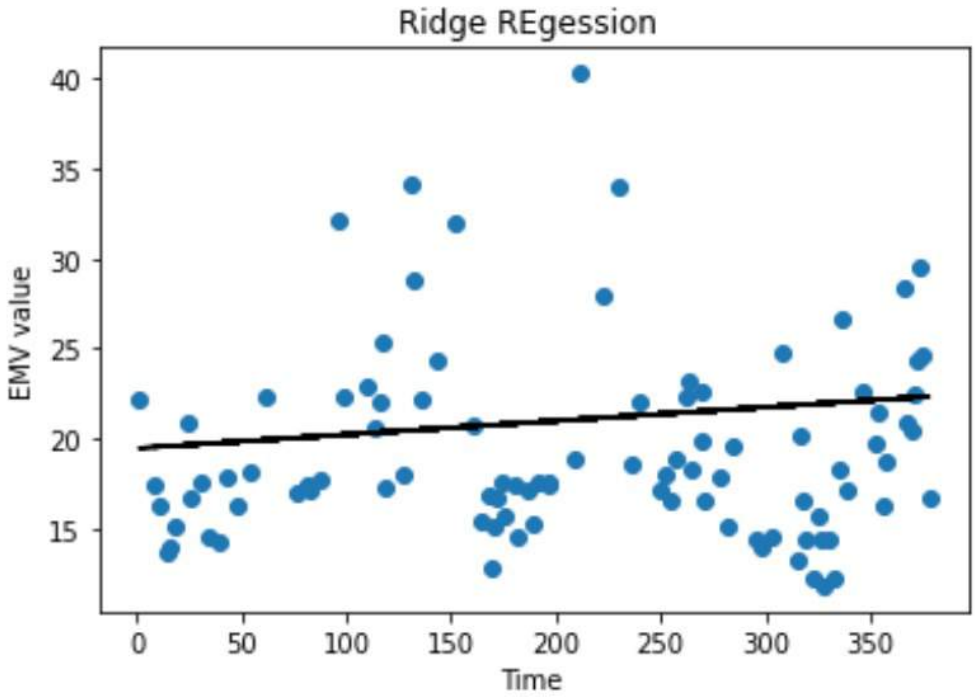

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

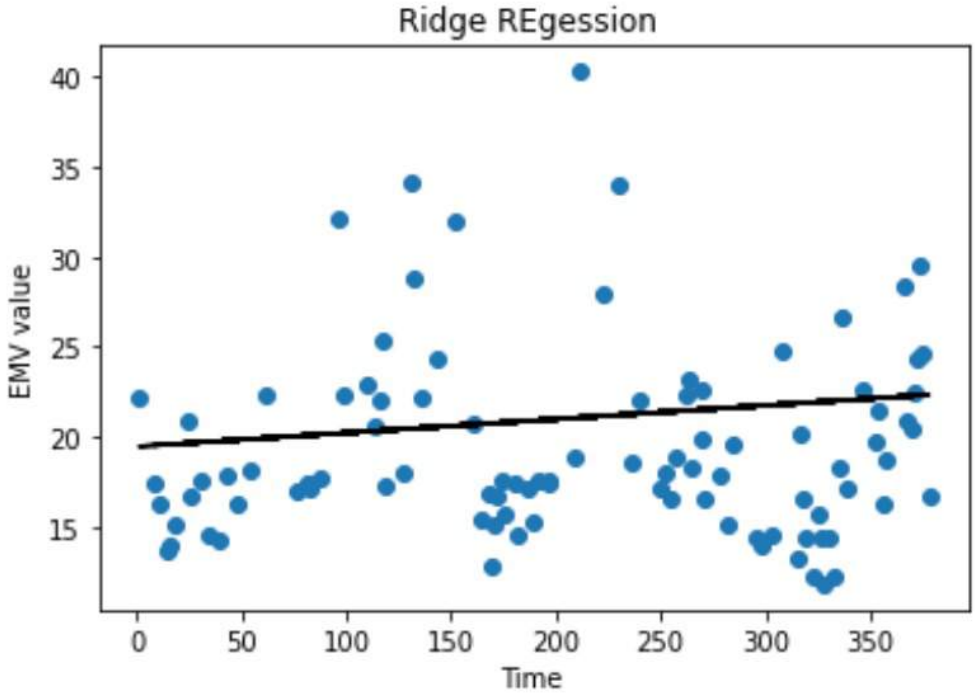

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

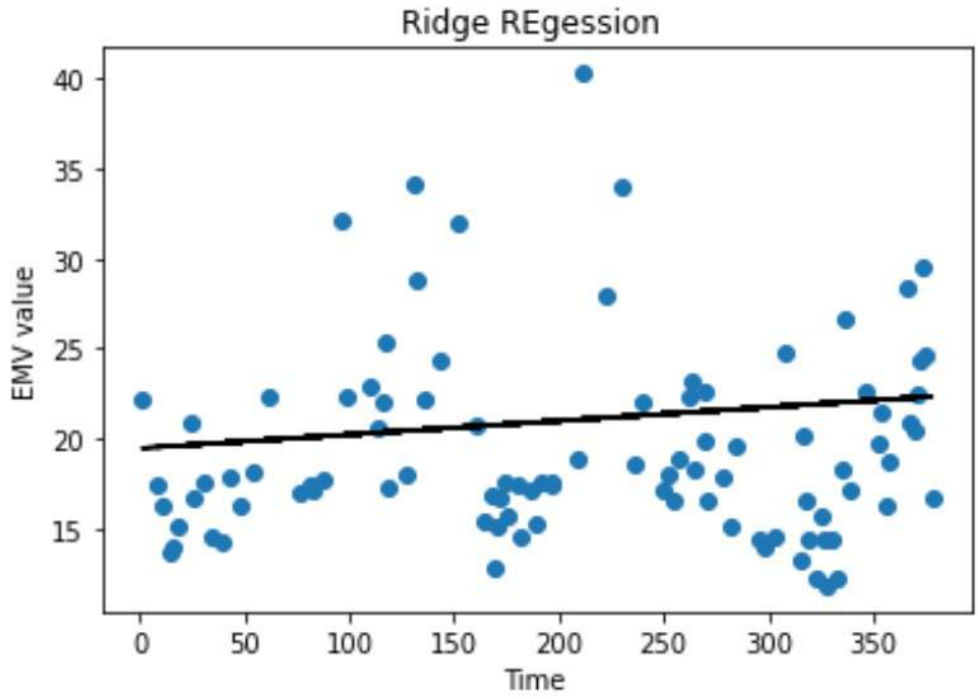

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

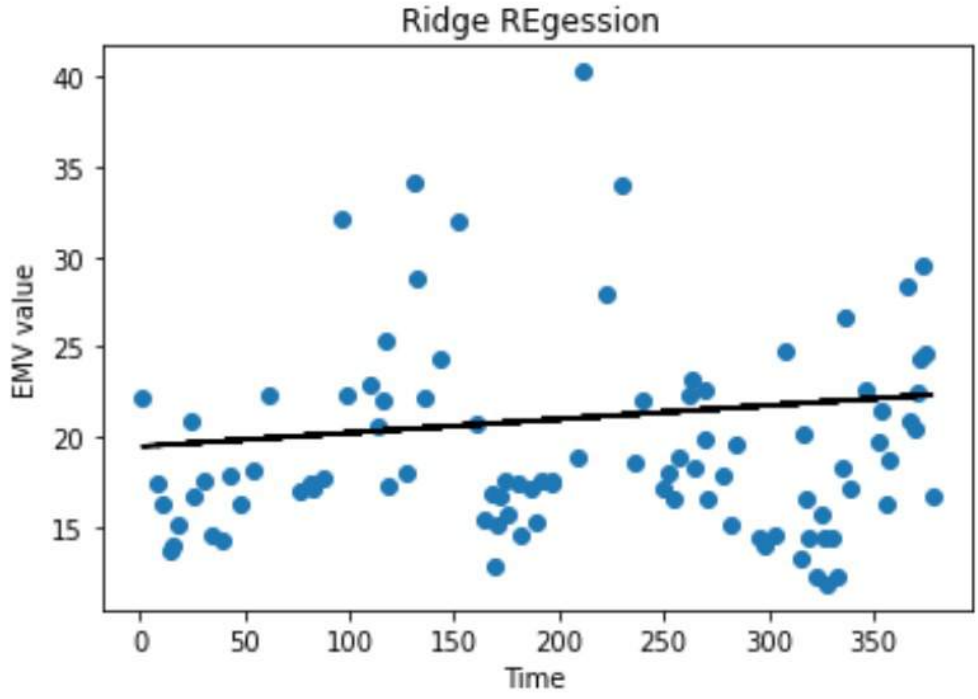

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

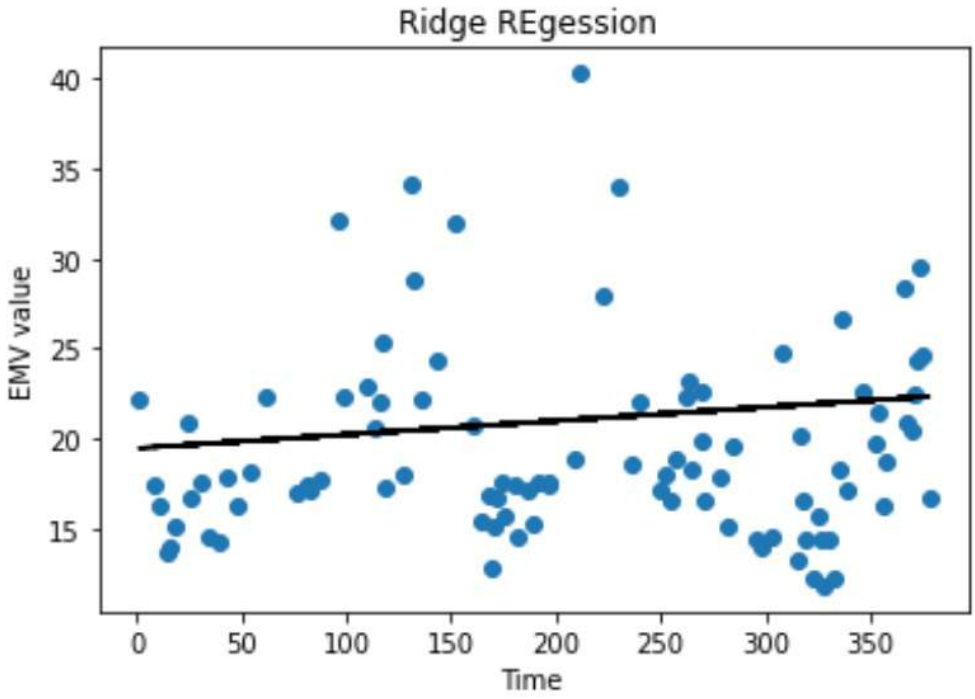

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

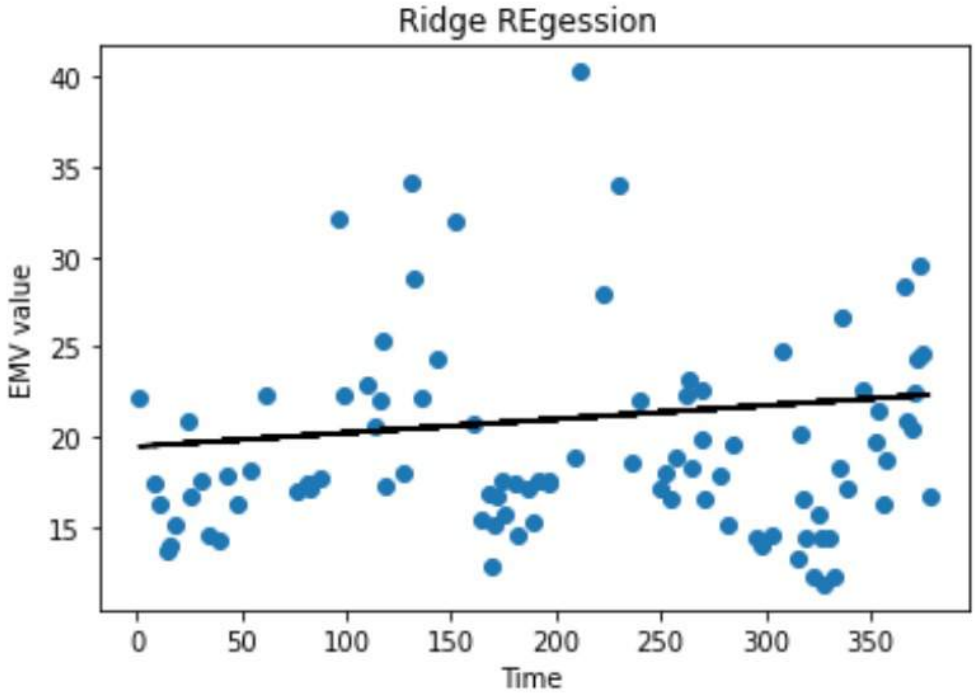

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

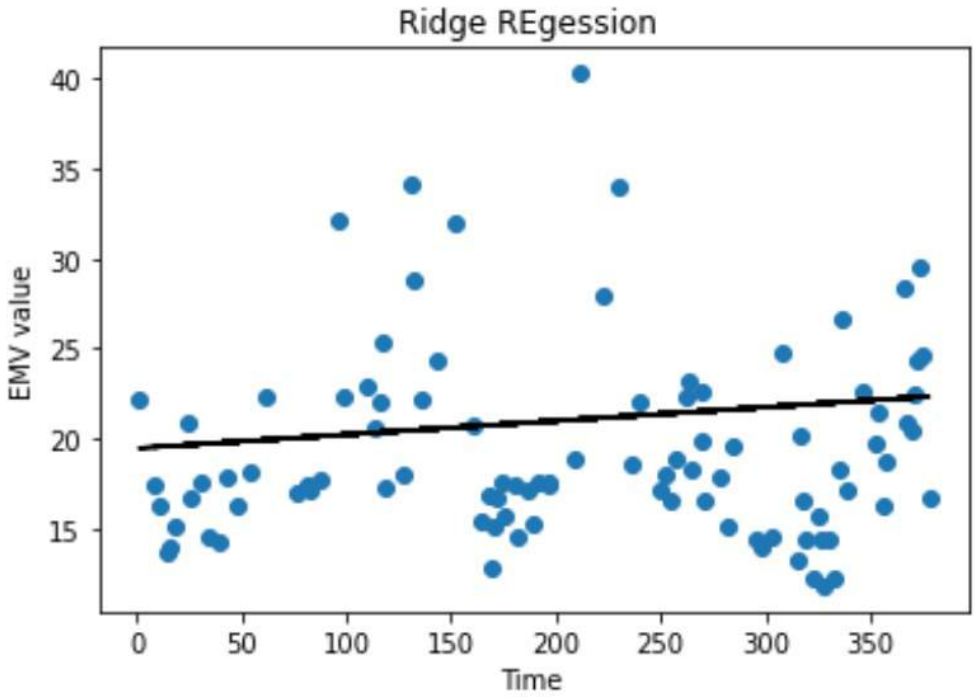

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

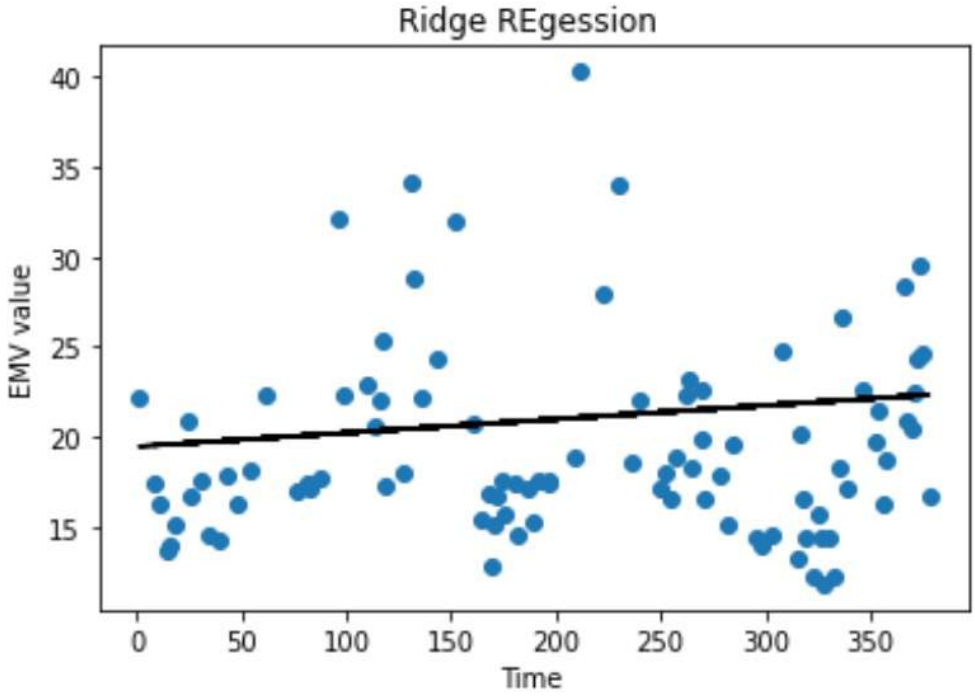

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

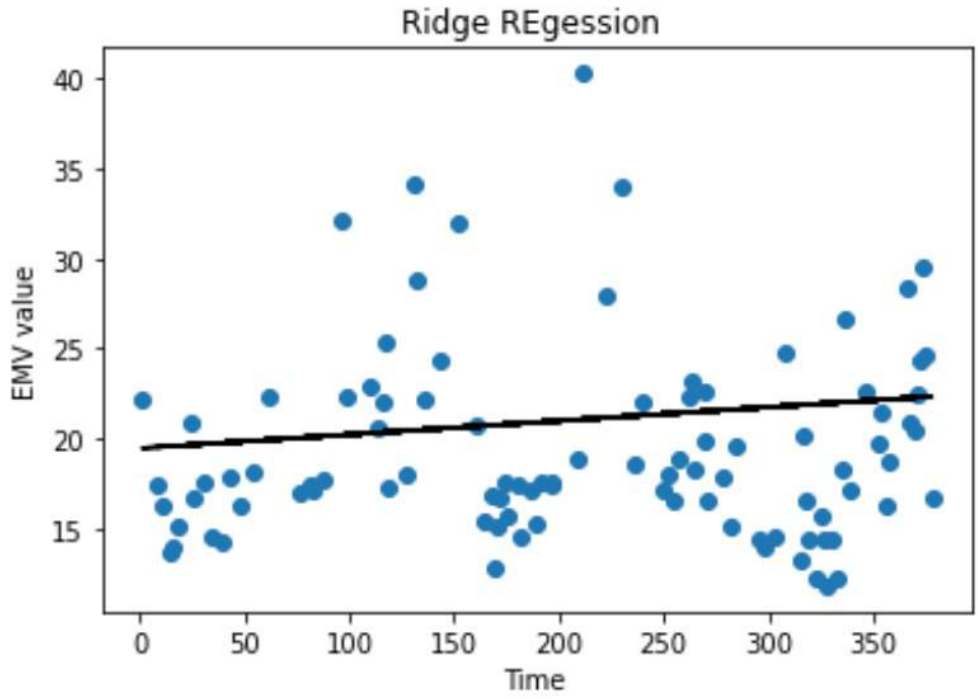

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

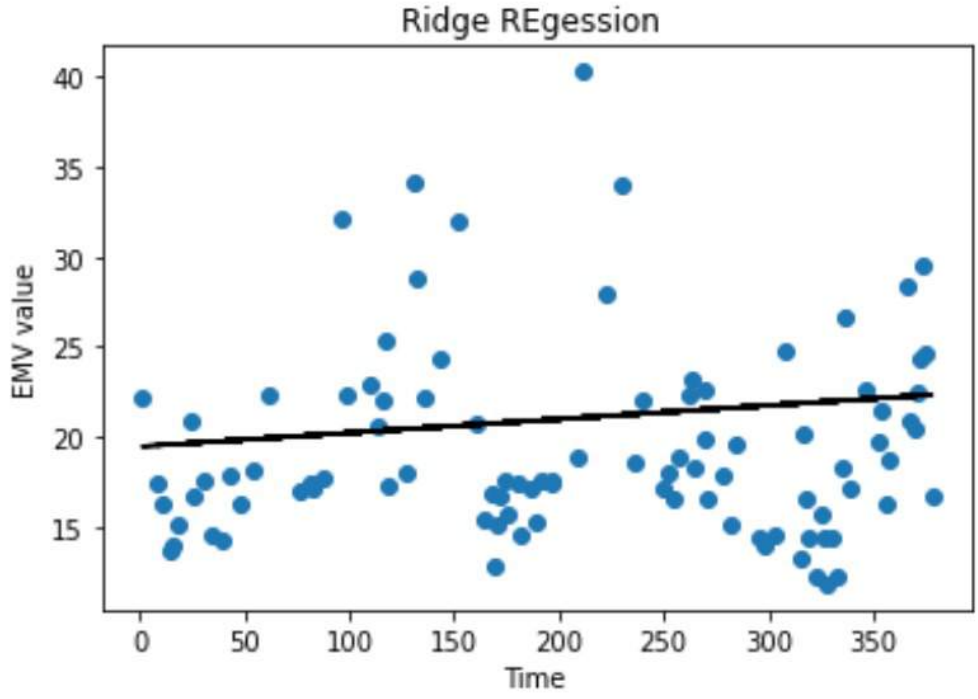

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

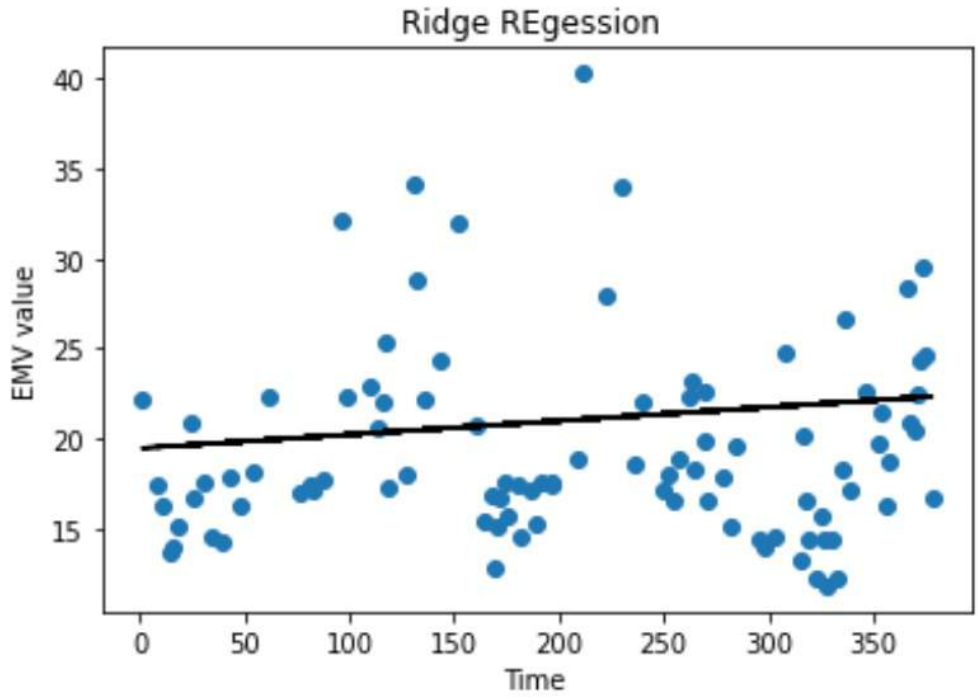

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

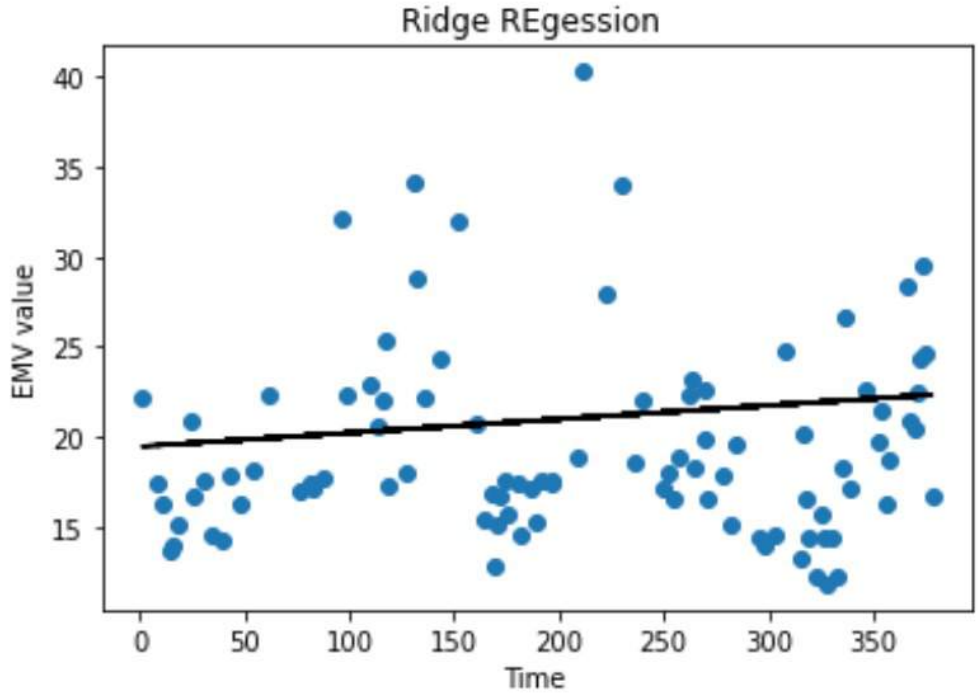

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

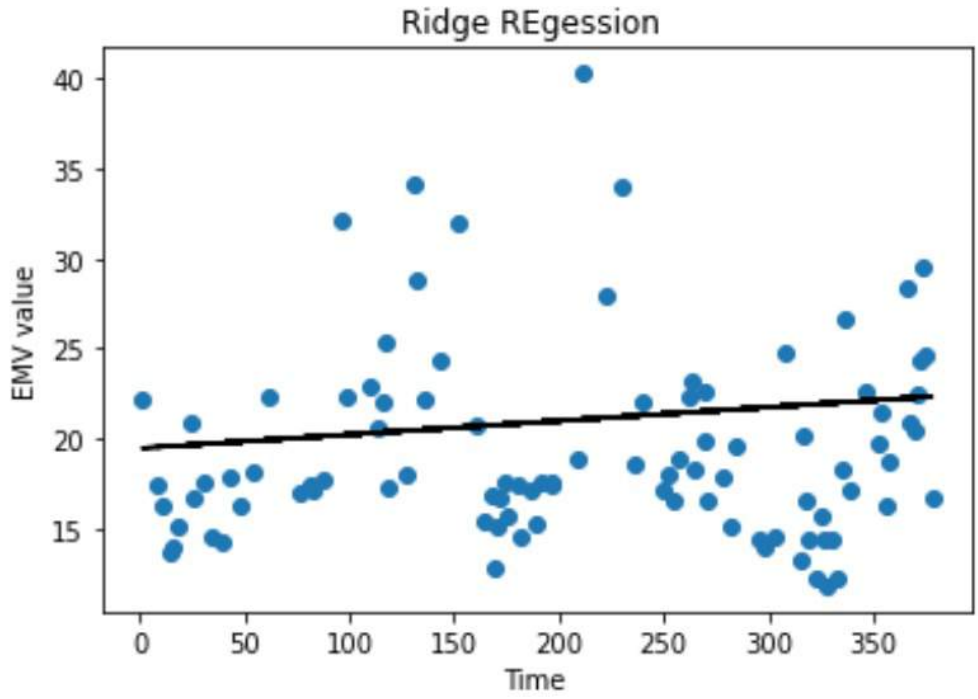

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

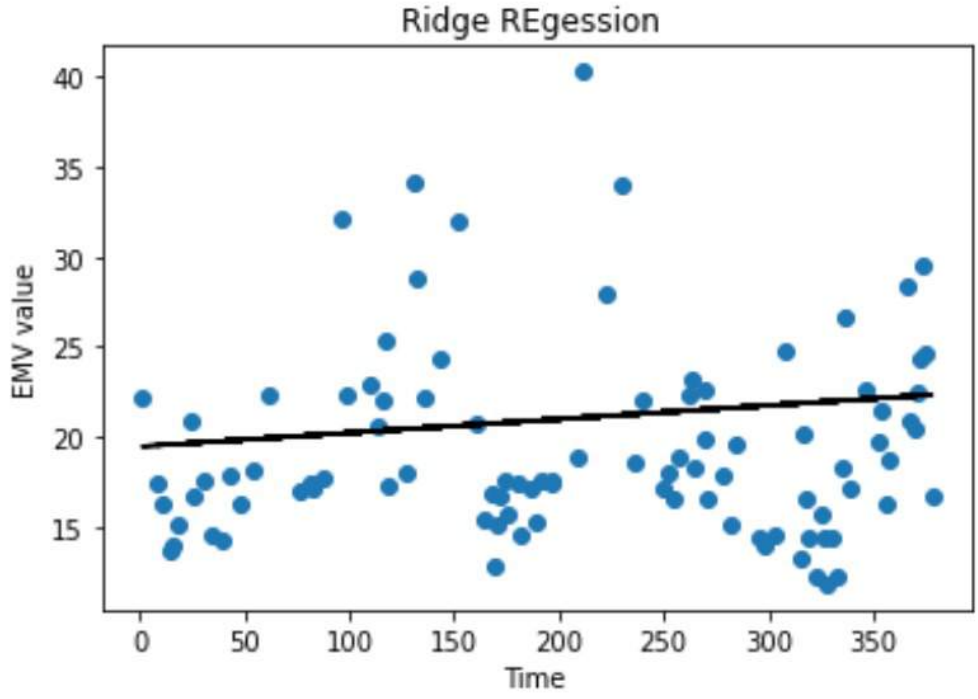

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

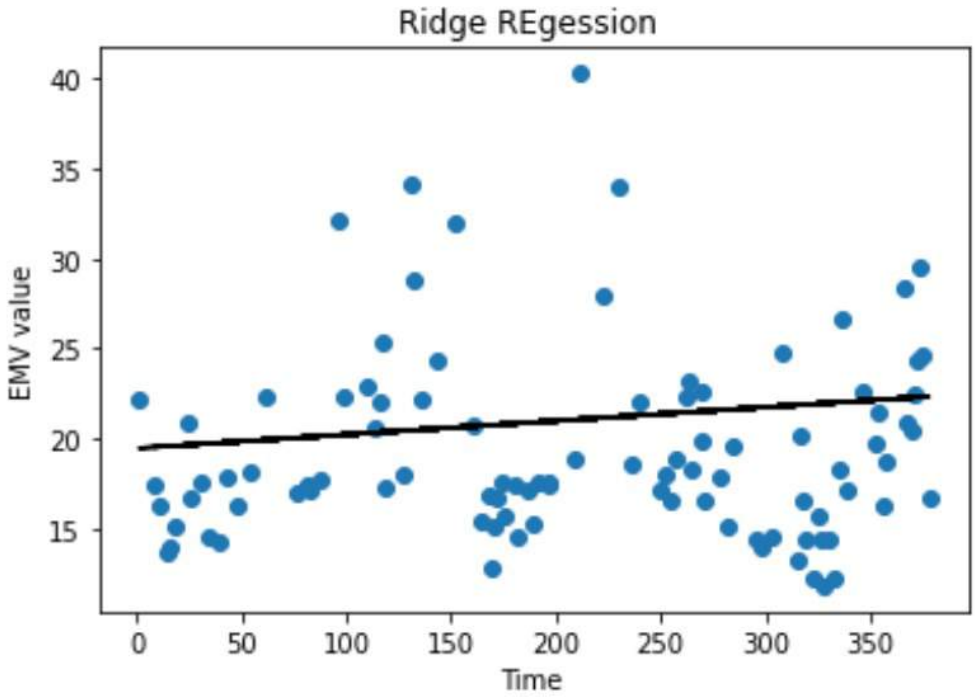

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

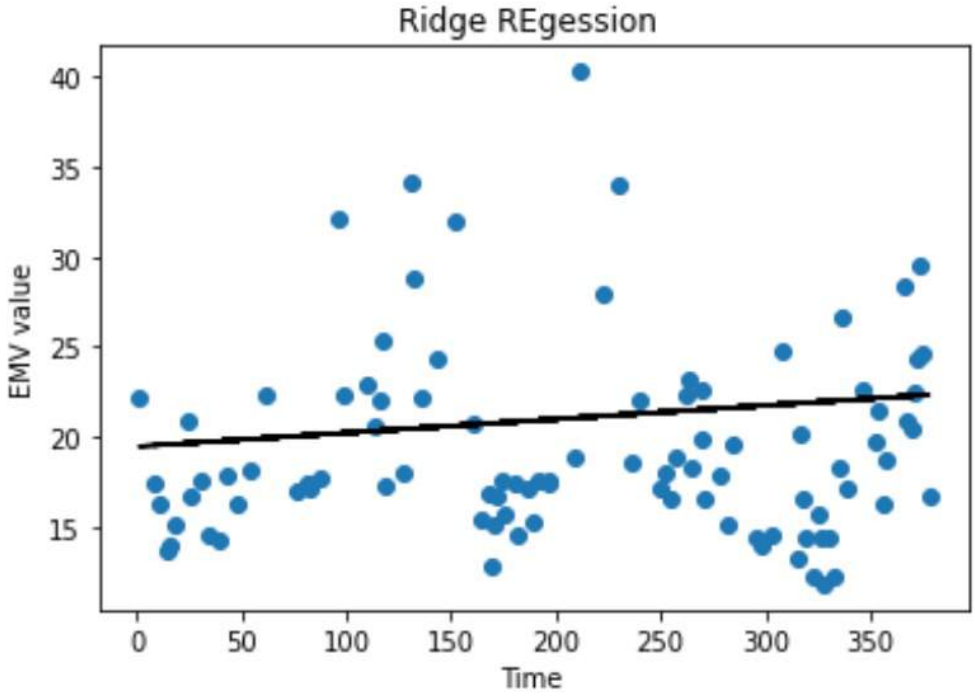

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

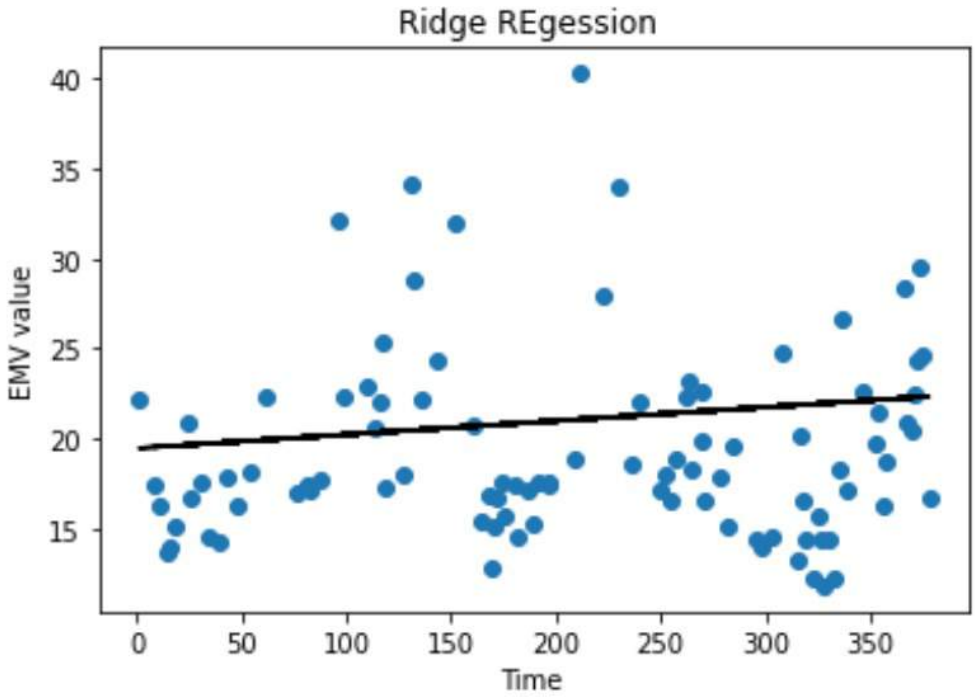

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

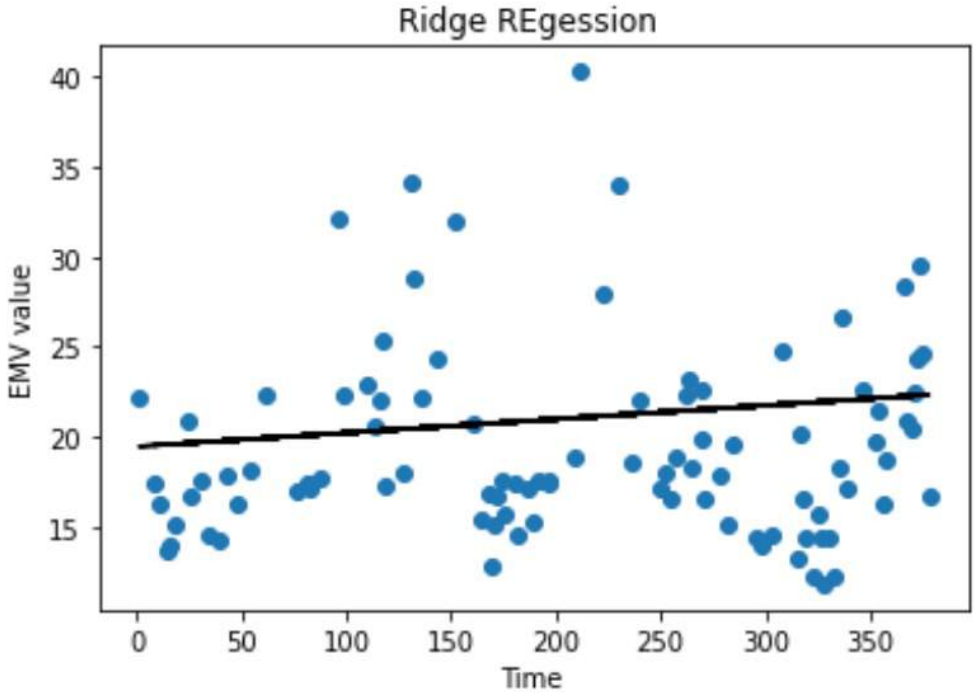

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

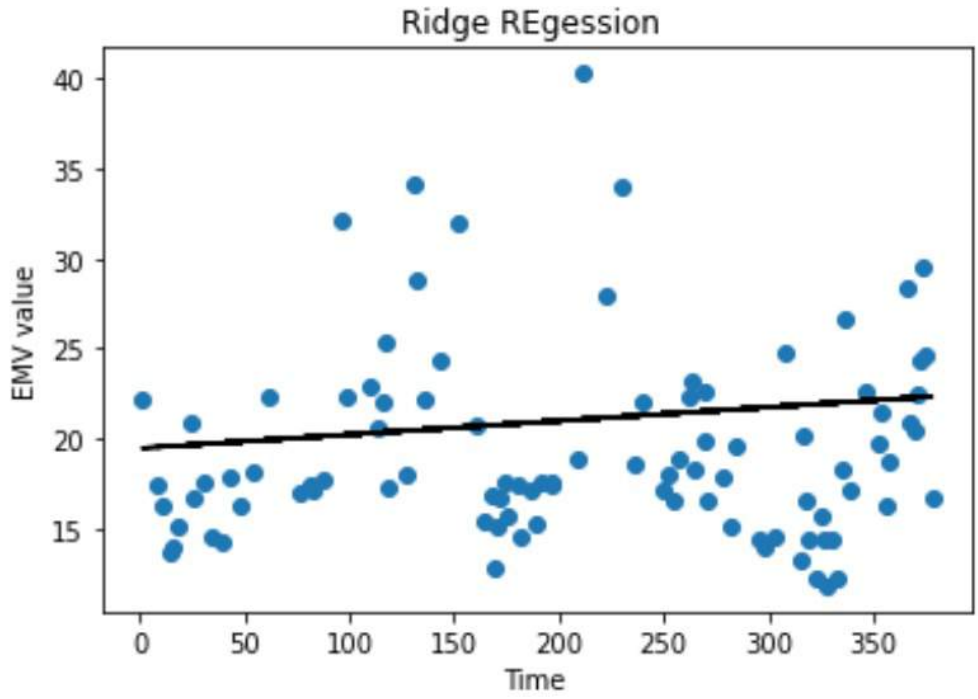

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

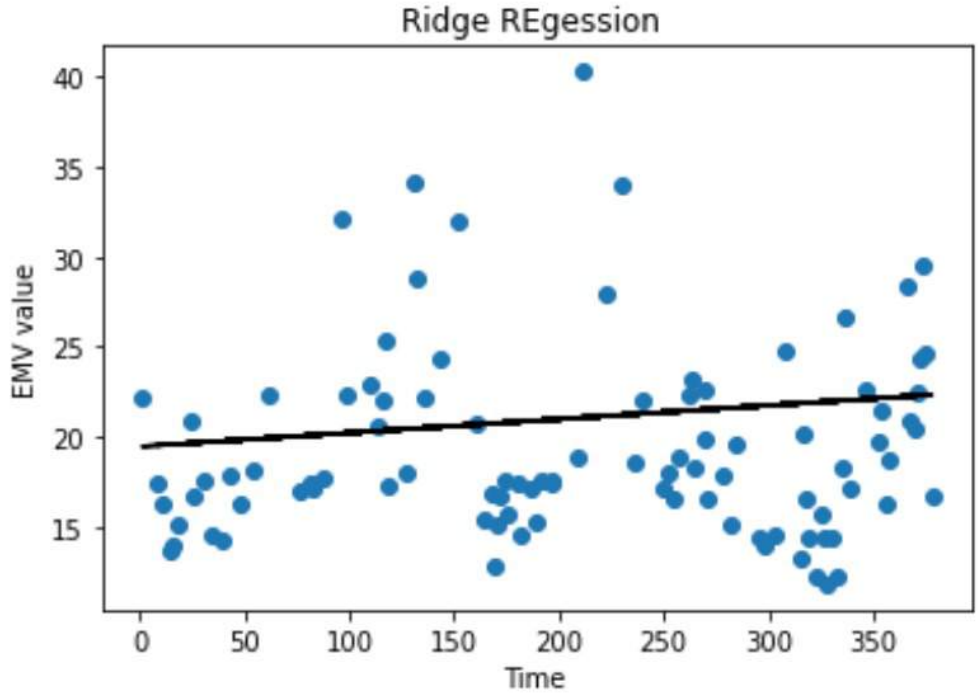

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

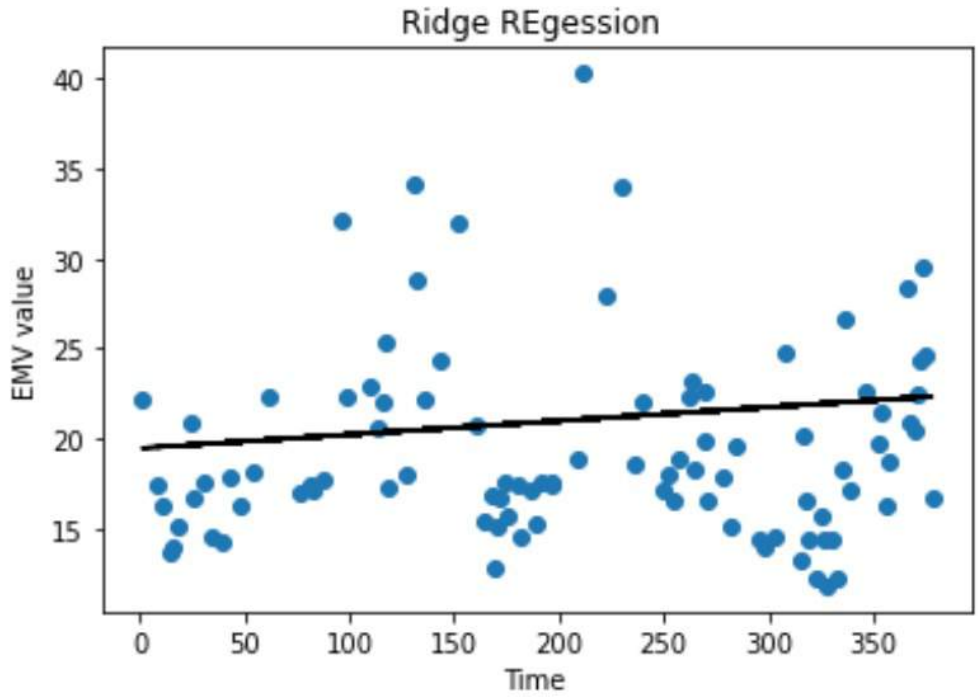

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

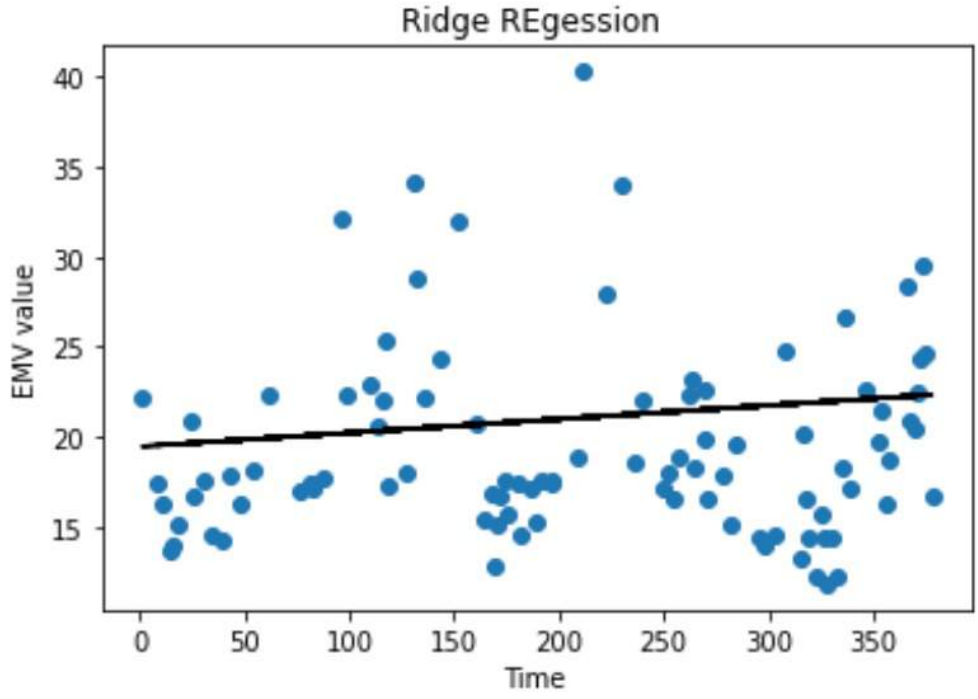

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

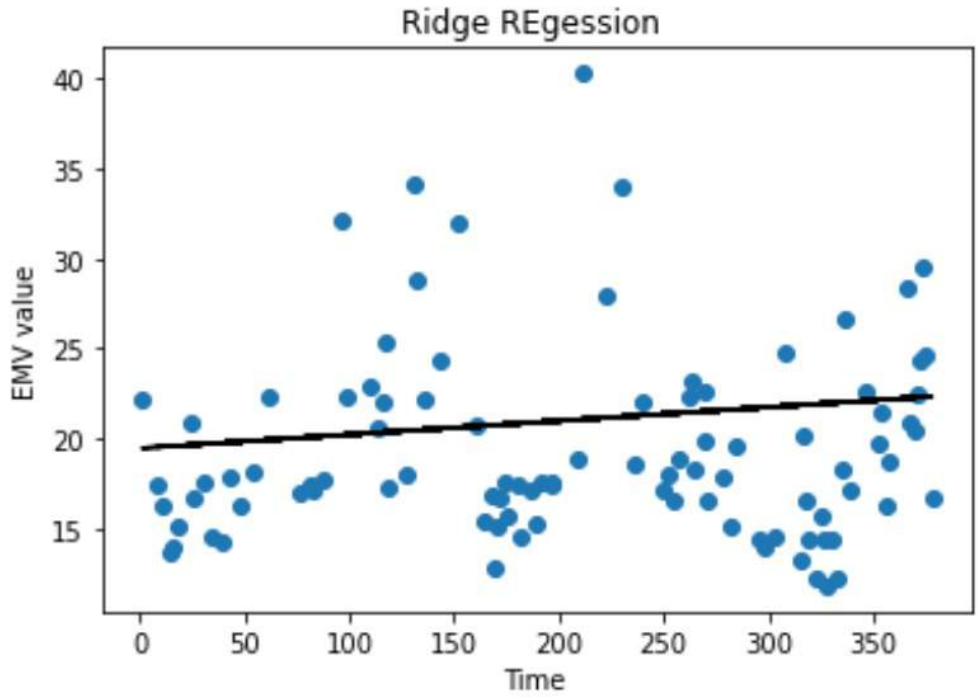

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

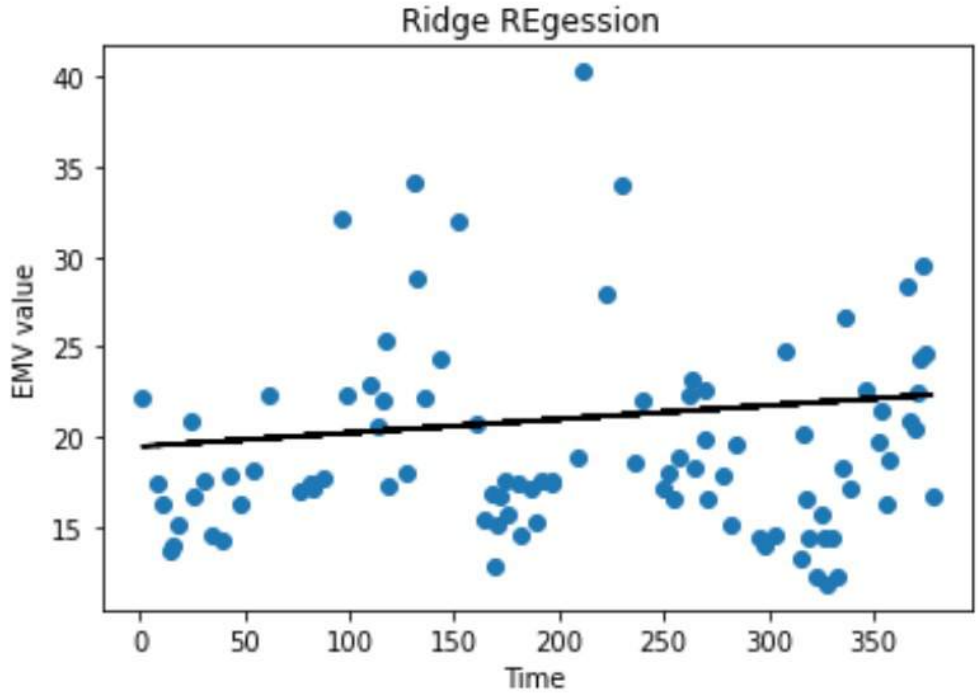

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

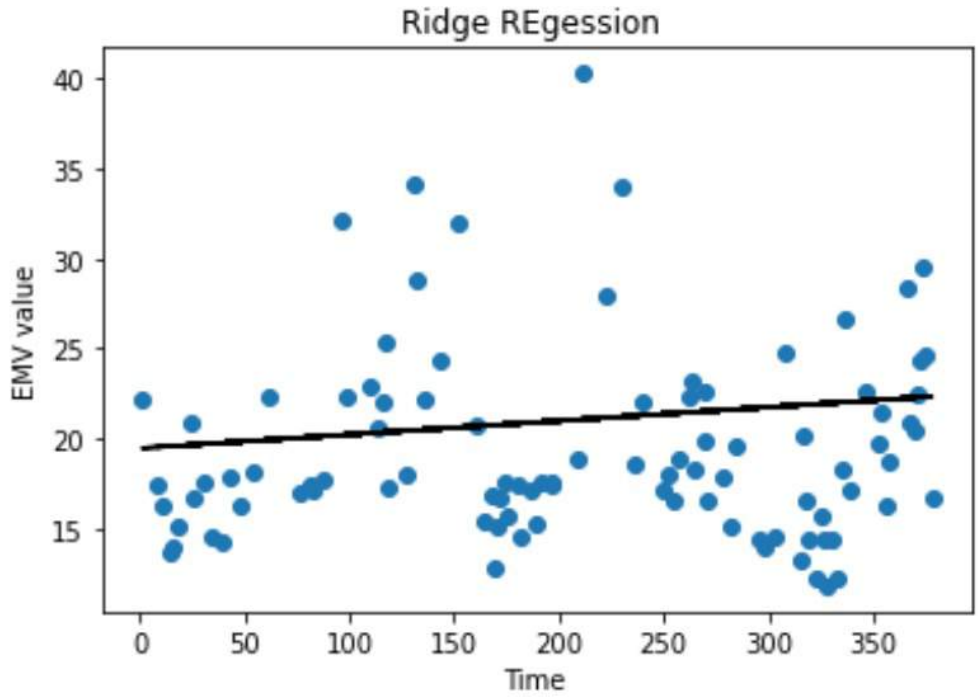

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

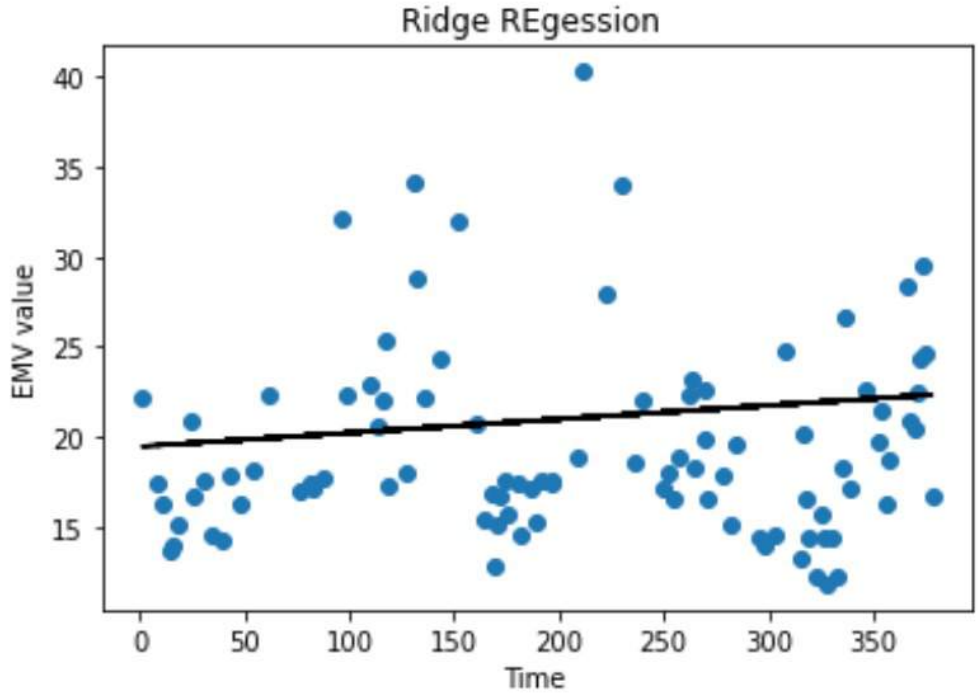

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

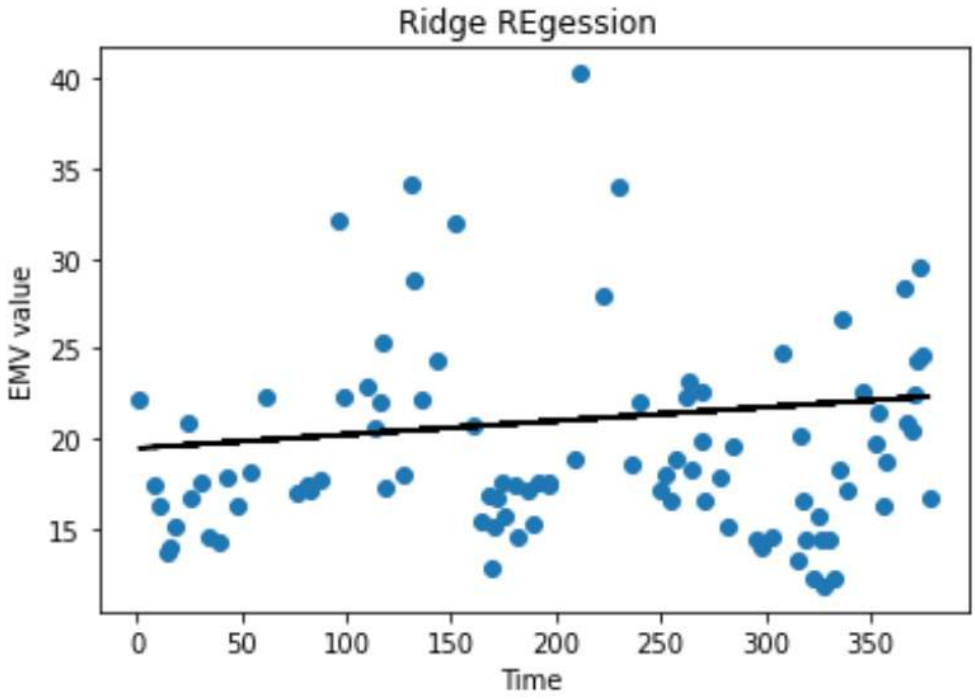

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

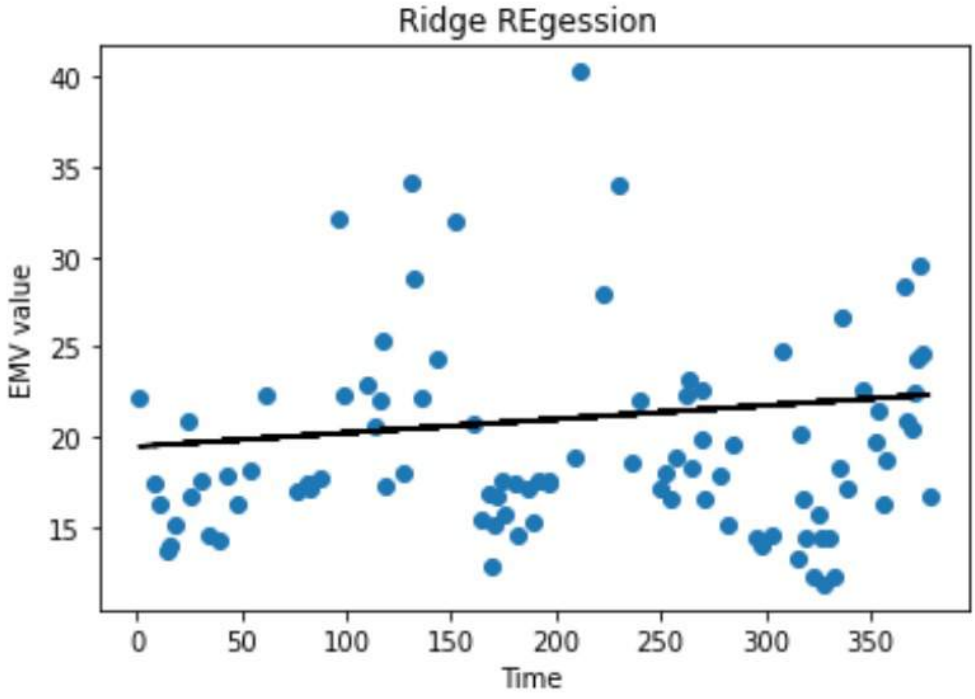

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

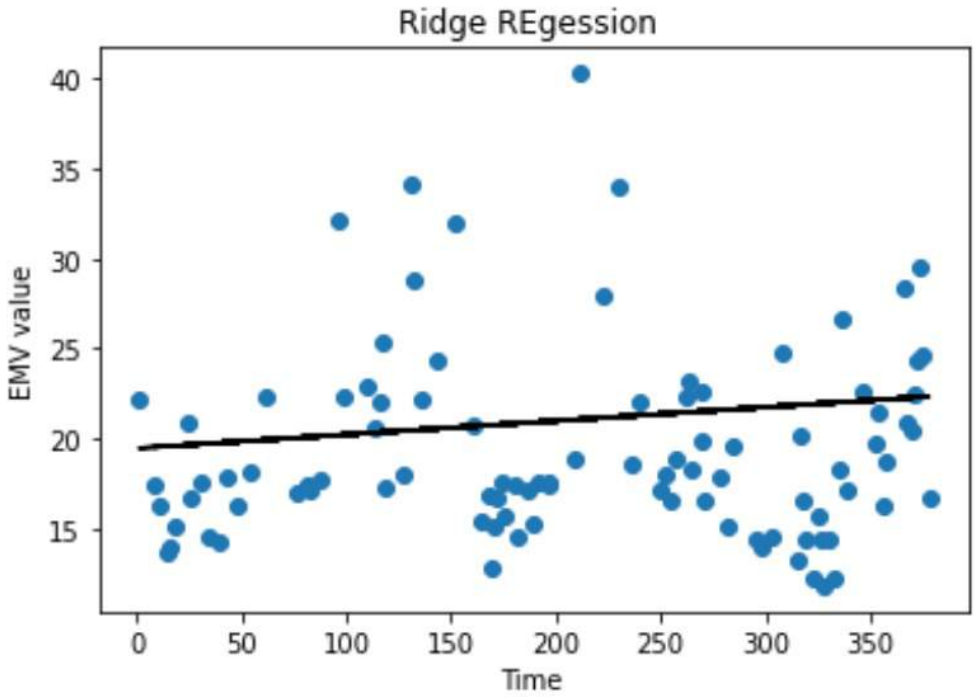

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

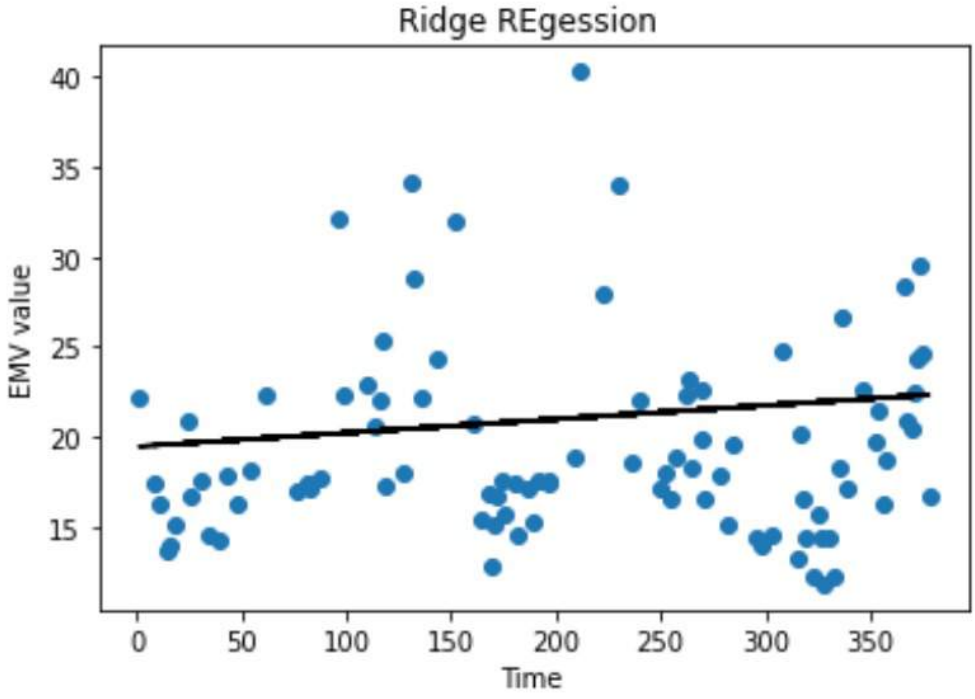

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

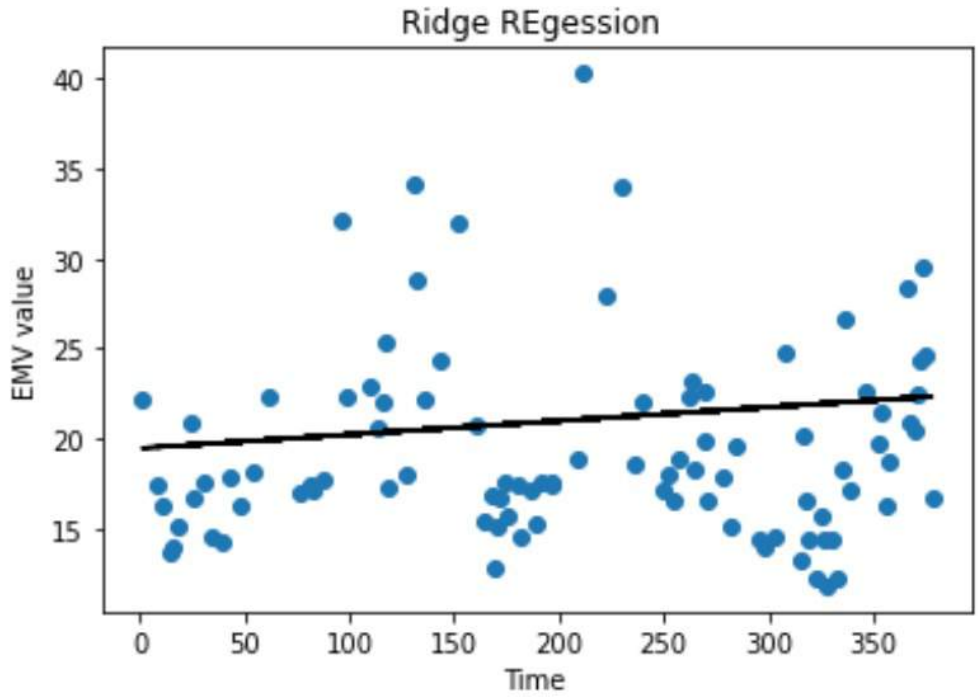

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350


Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

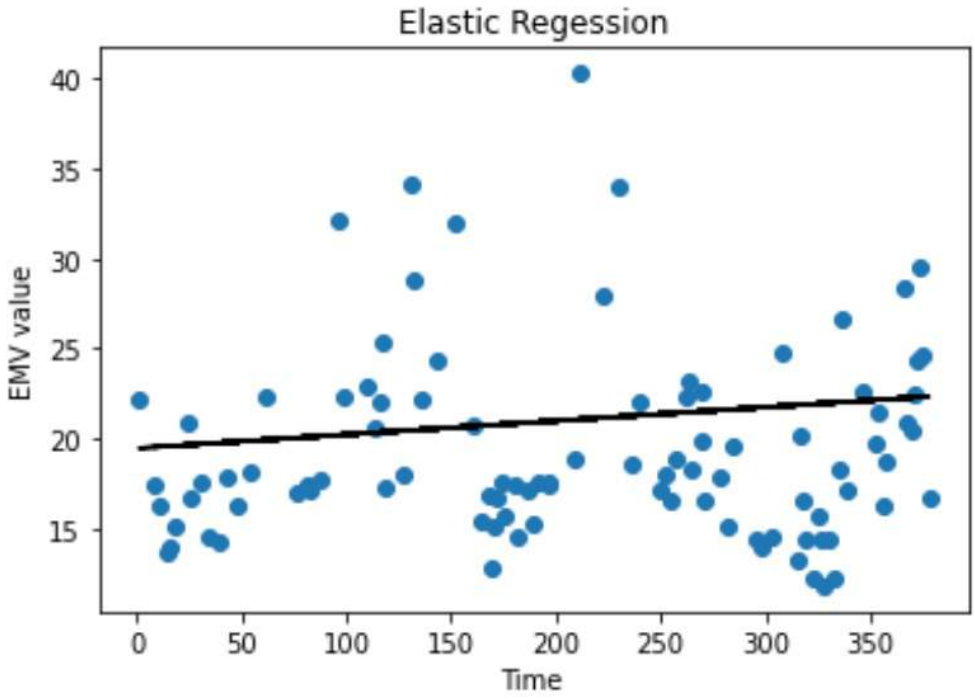

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

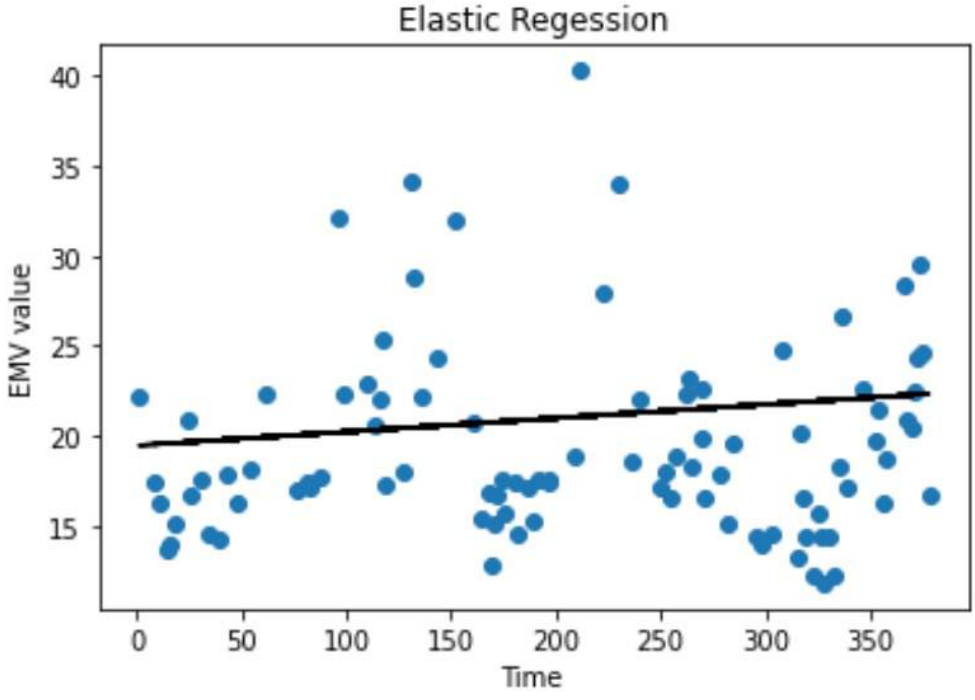

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

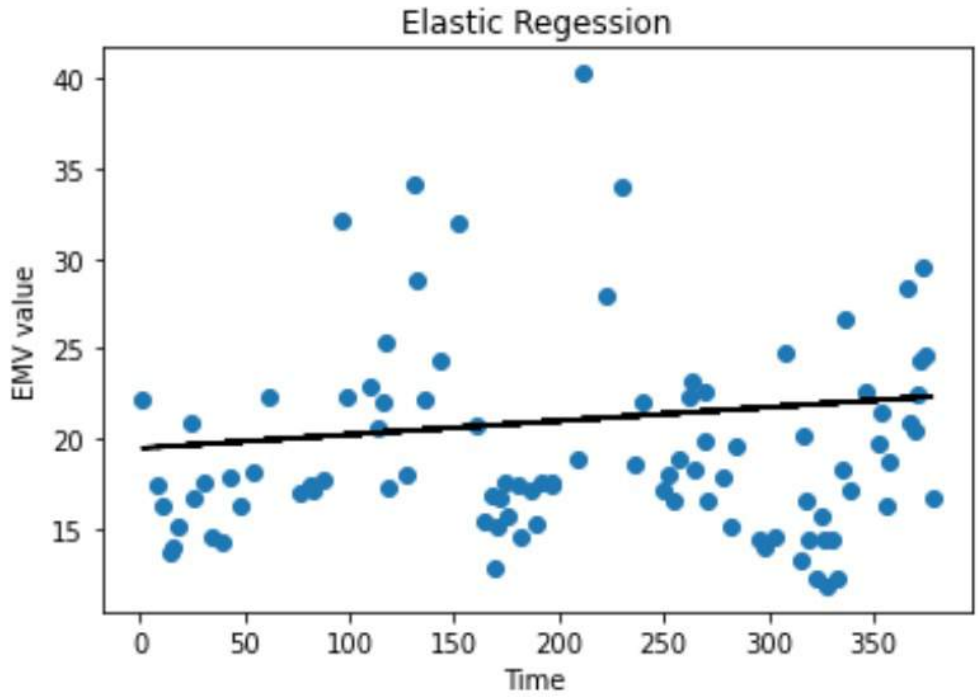

Ridge REgession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

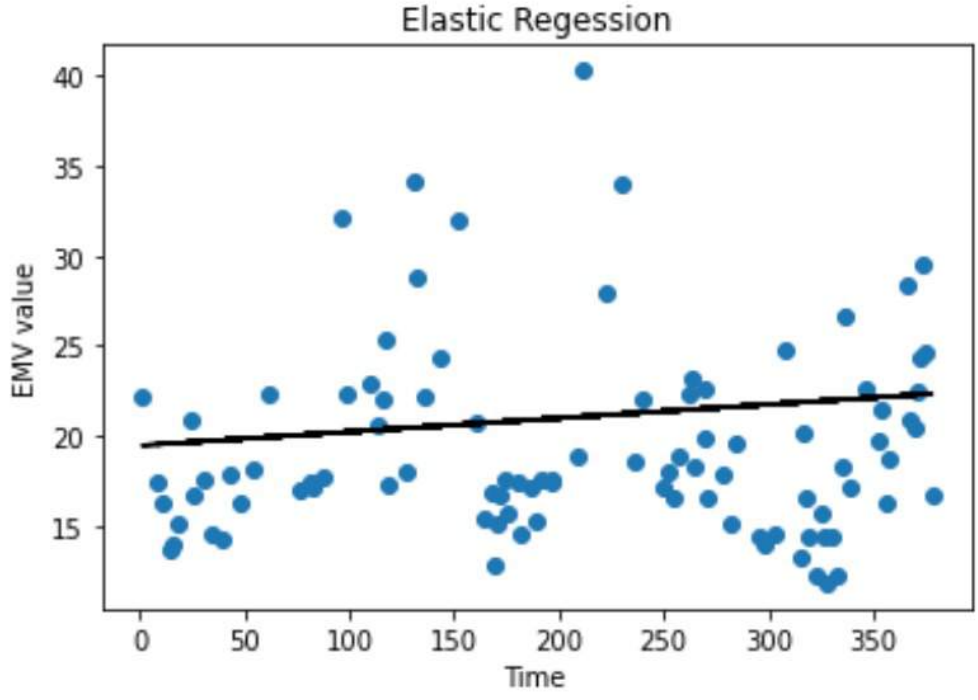


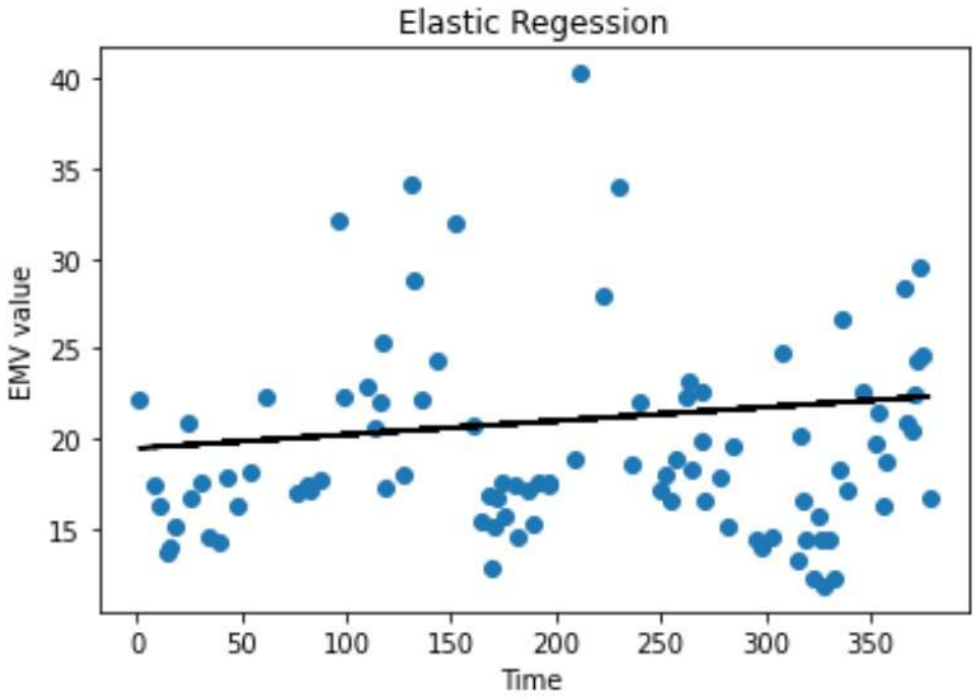

```
def Elastic_Reg(train_X, train_Y, test_X, test_Y):
    reg=ElasticNet(alpha=A)
    reg.fit(train_X, train_Y)
    pred_Y=reg.predict(test_X)

    for tracker in range(45):
        plt.scatter(test_X, [y for y in test_Y])
        plt.plot(test_X,  [y for y in  pred_Y],color='black')
        plt.title("Elastic Regession")
        plt.xlabel("Time ")
        plt.ylabel("EMV value")

        plt.show()
Elastic_Reg(train_X, train_Y, test_X, test_Y)
```

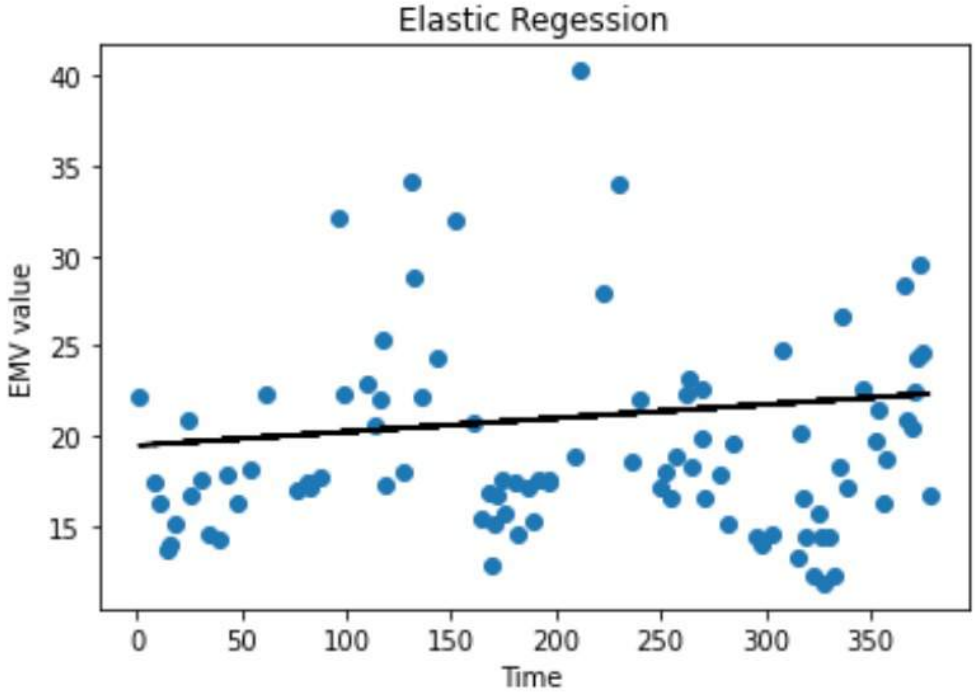

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

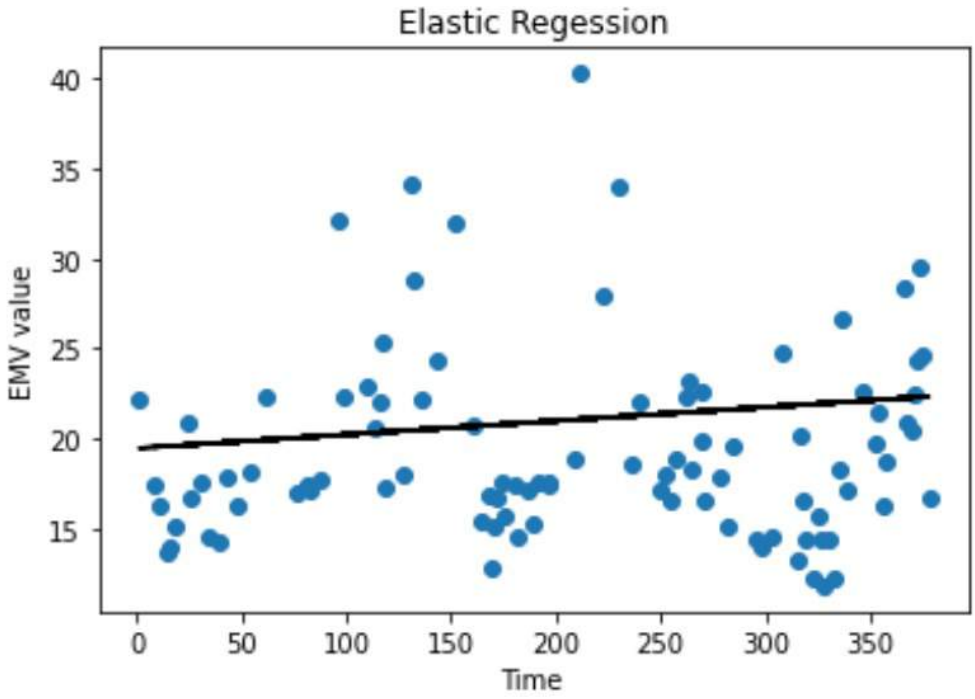

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

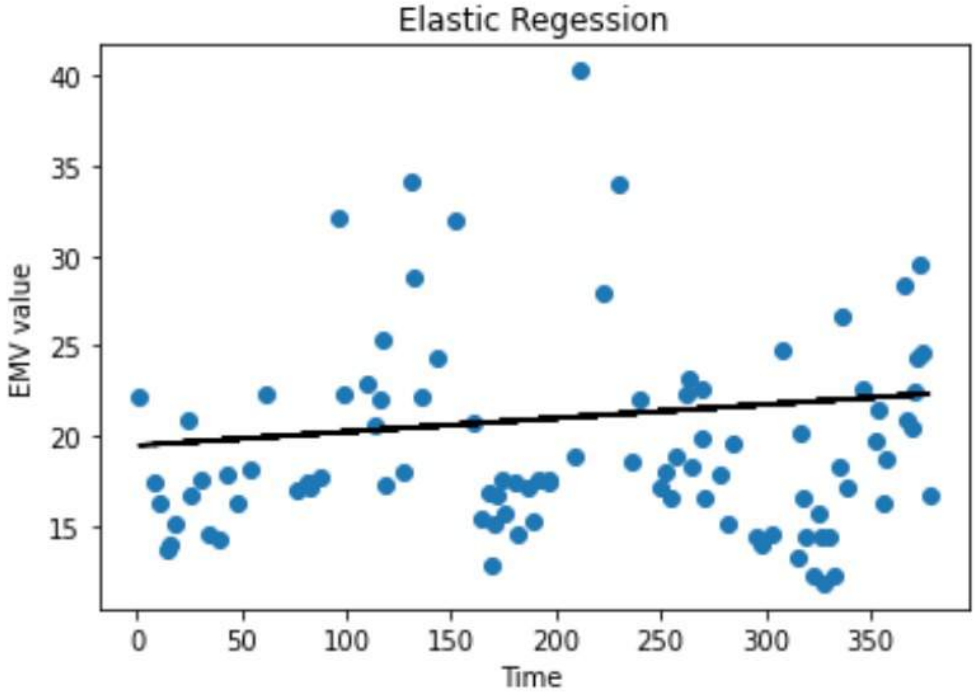

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

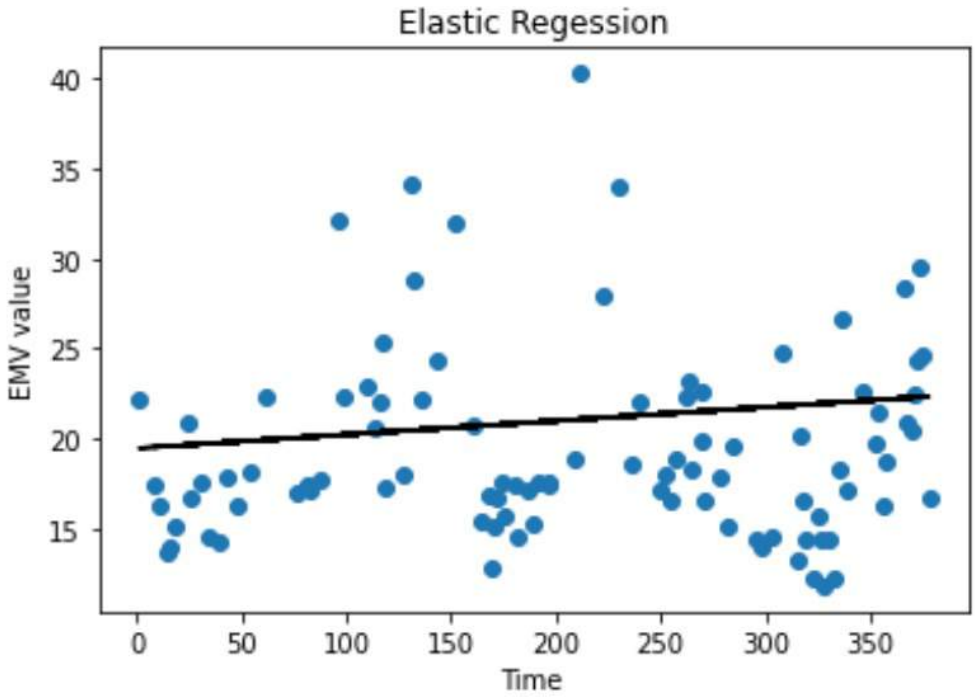

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

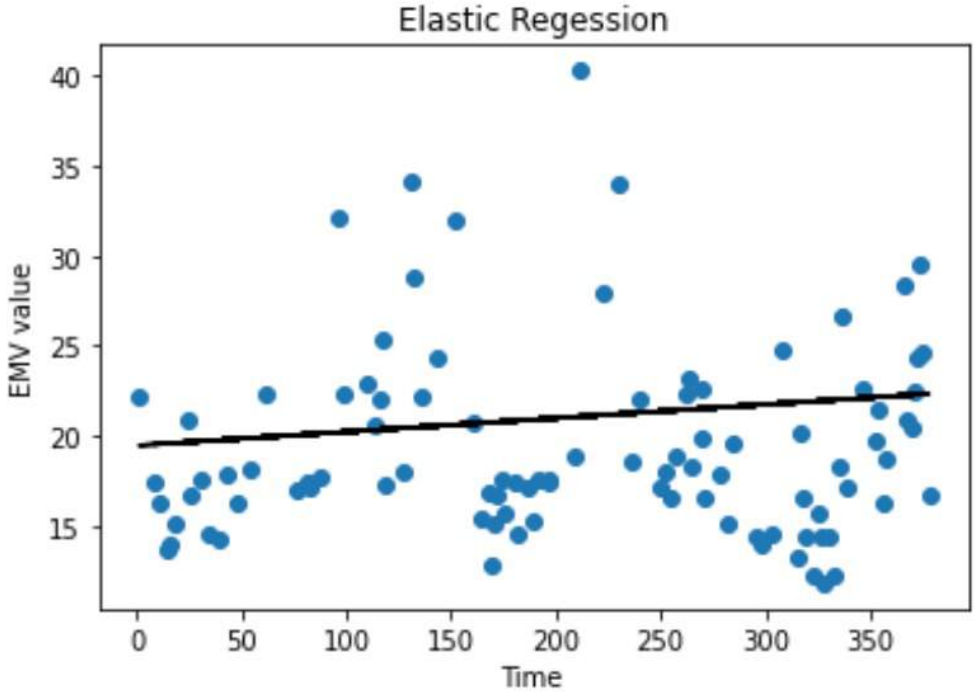

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

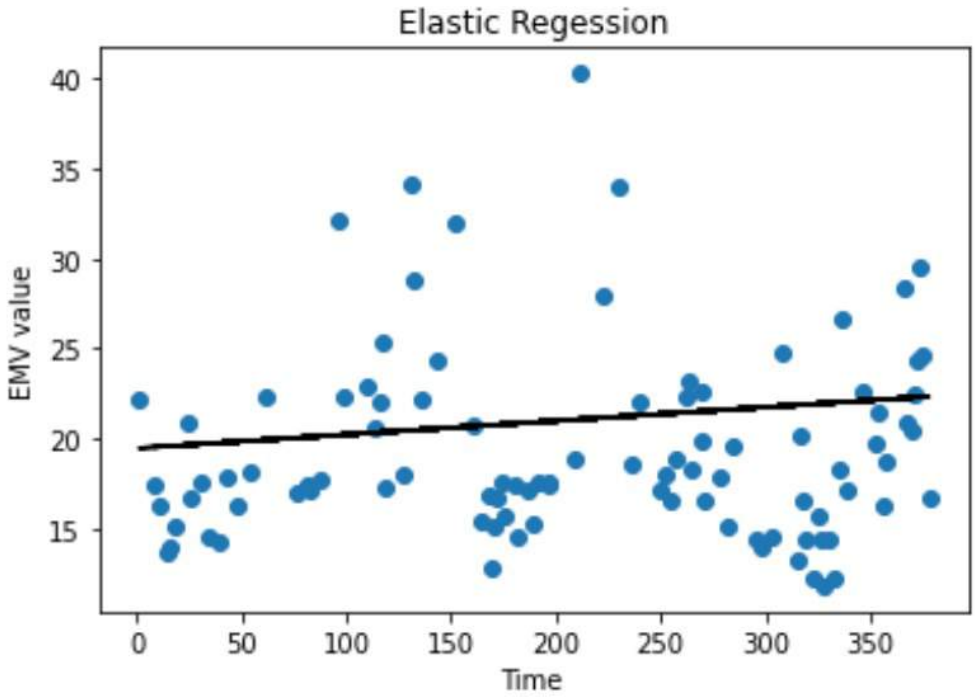

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

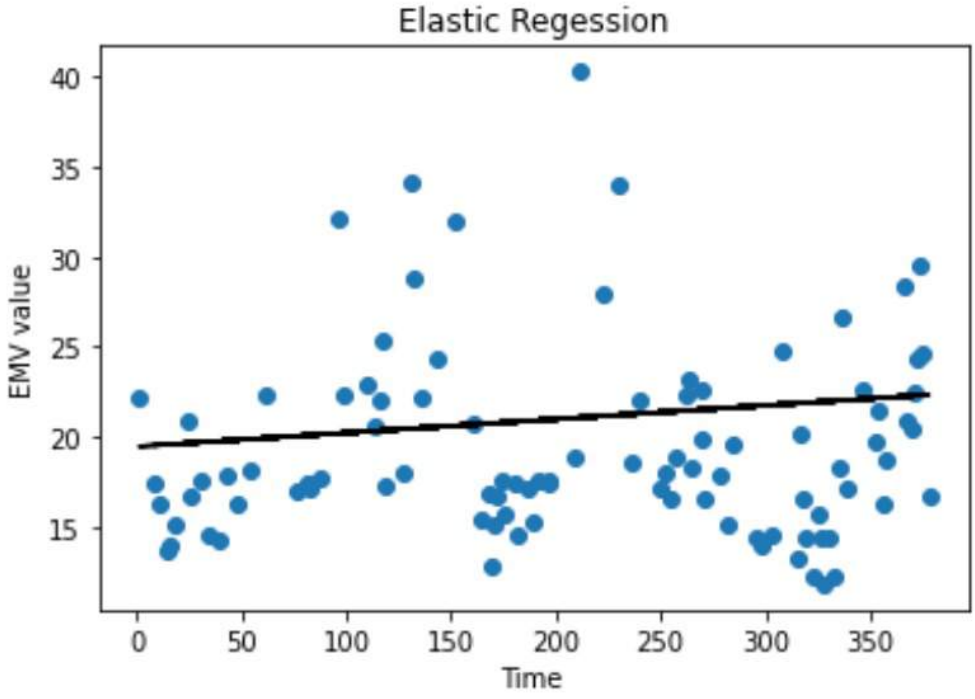

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

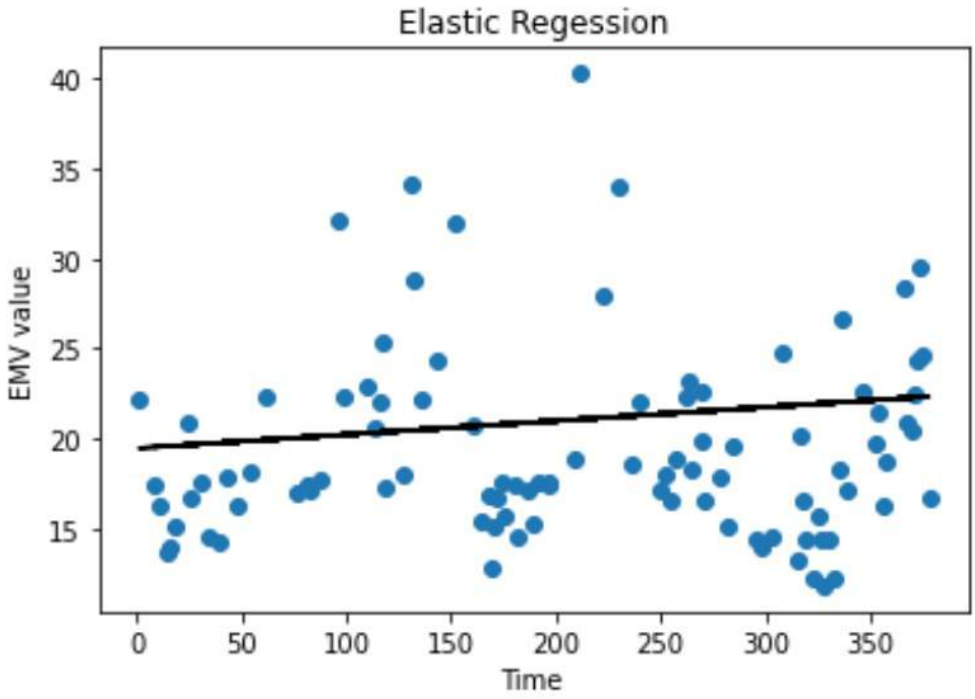

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

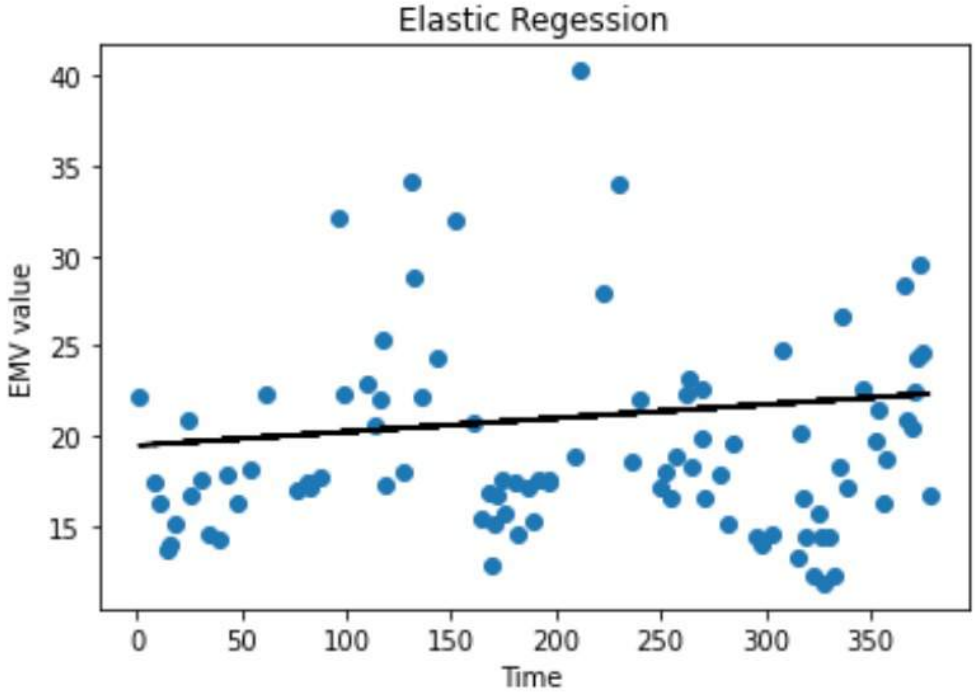

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

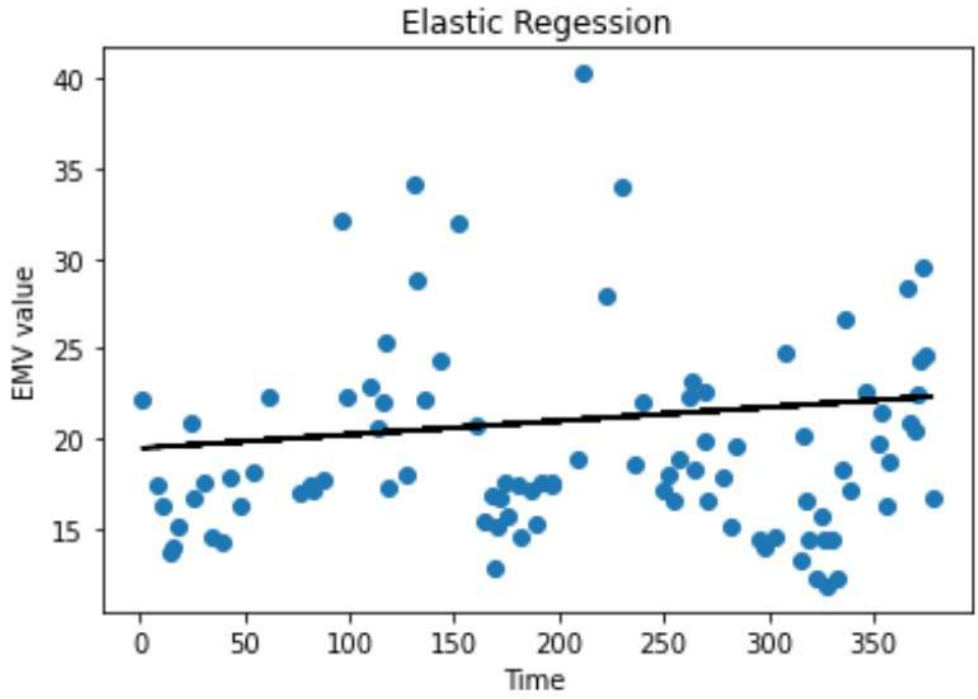

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

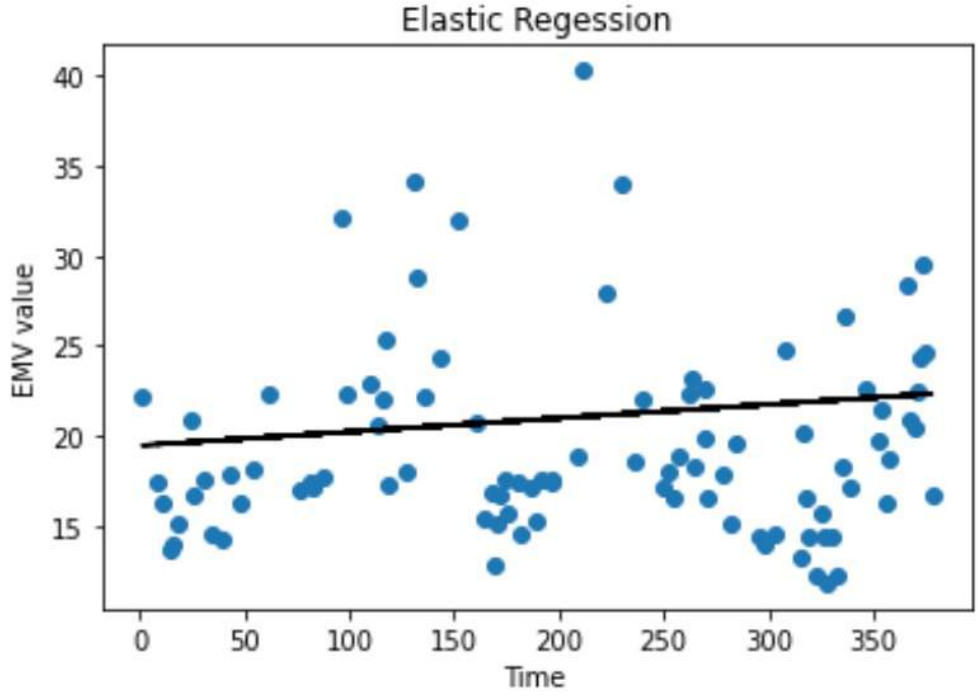

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

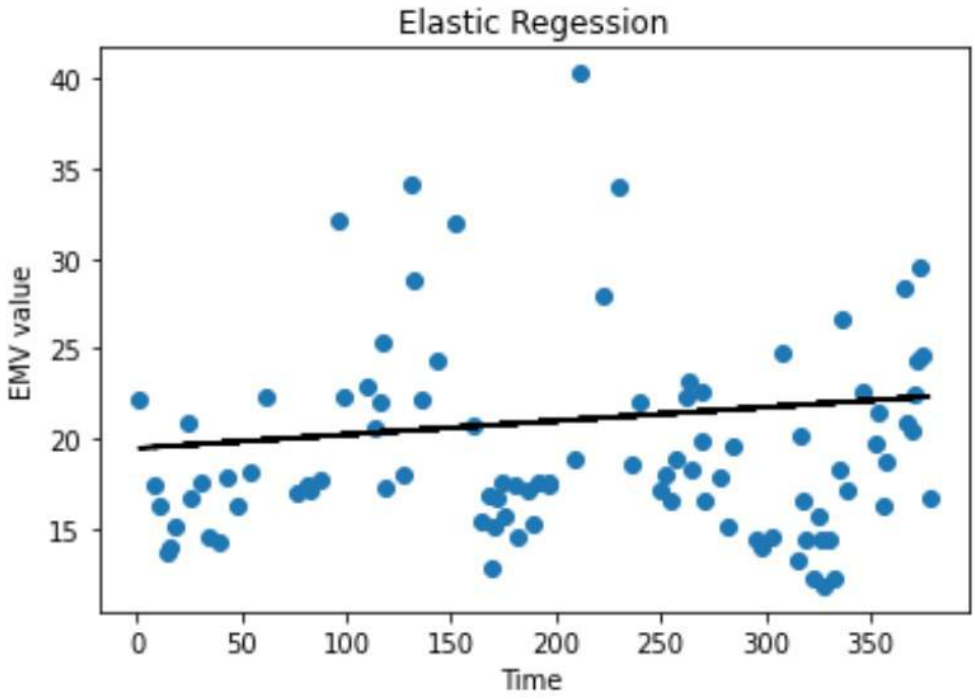

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

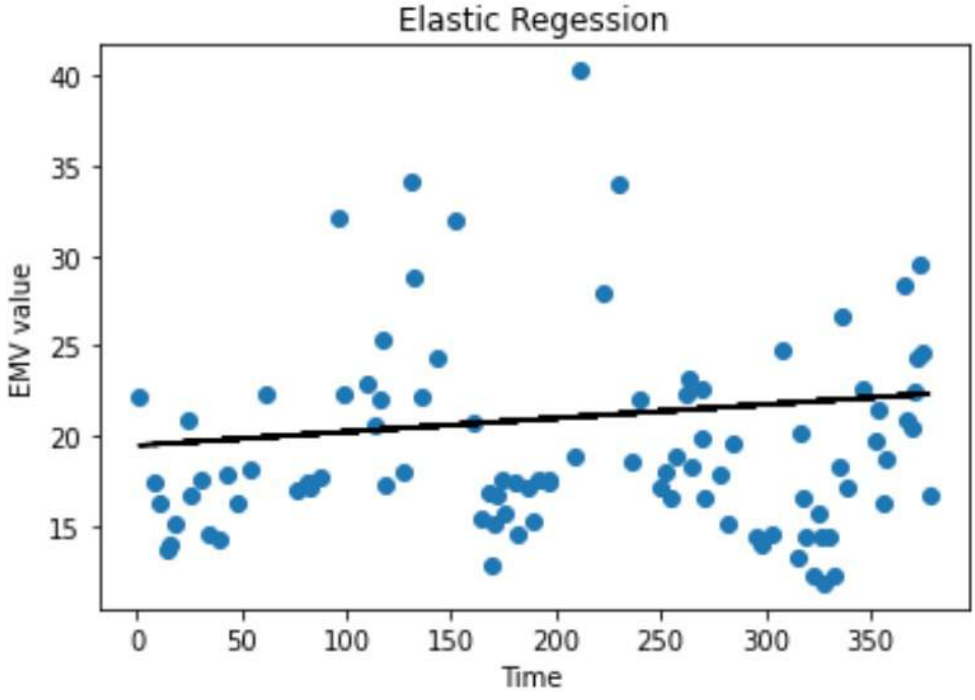

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

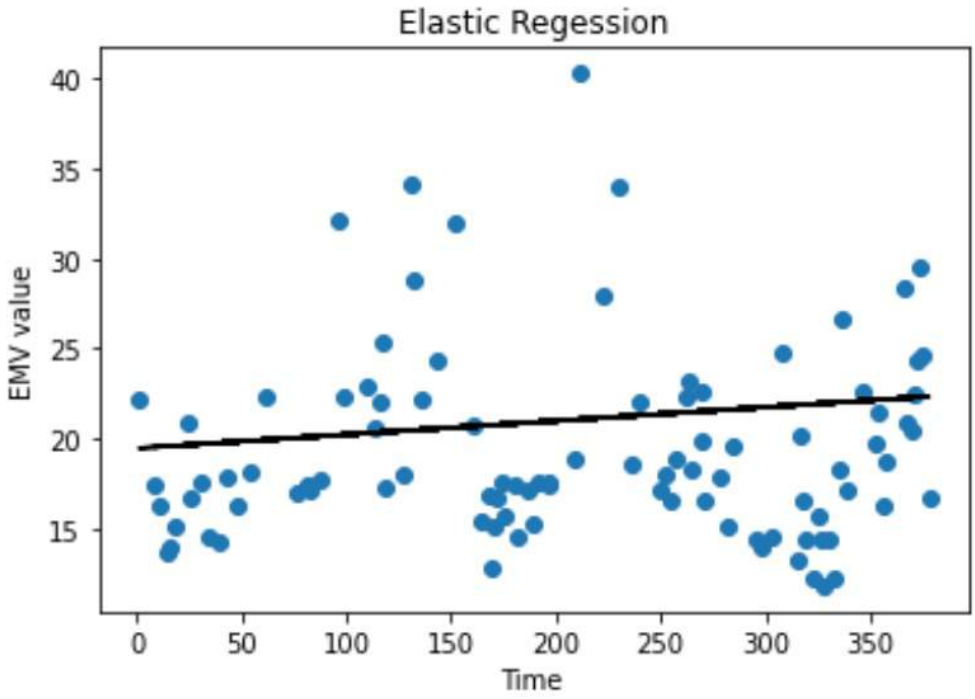

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

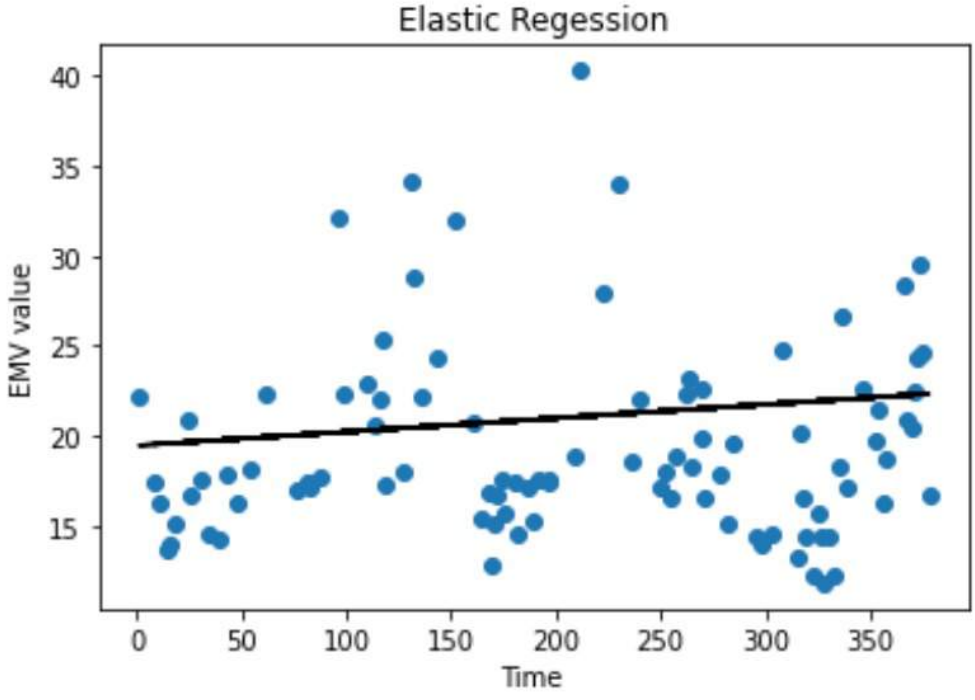

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

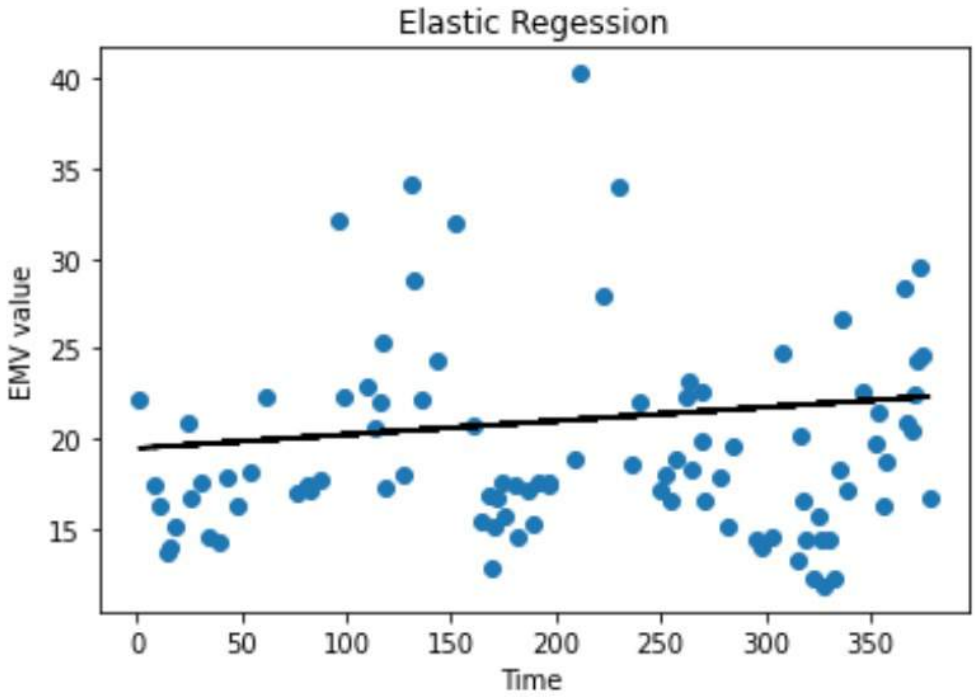

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

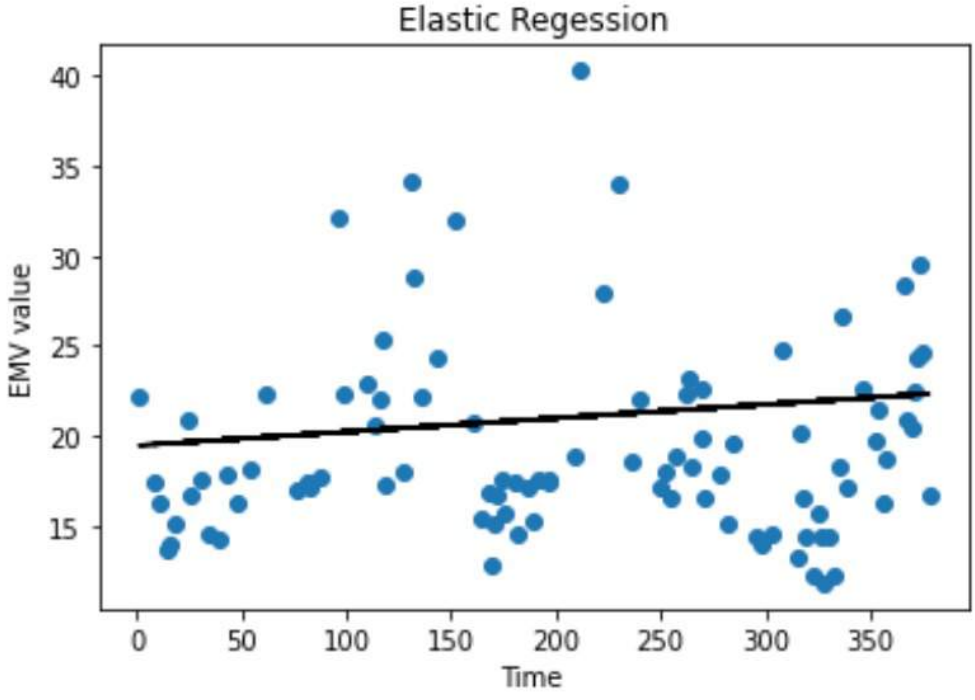

Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350


Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350


Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350


Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350


Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350


Elastic Regession
EMV value
Time
40
35
30
25
20
15
0
50
100
150
200
250
300
350

Discussion

From the report, we have analyzed two datasets. In the first dataset, we analyzed the regime-switching phenomenon in the stock market using Gaussian Mixture Models. In the second dataset, we built several regression models to predict the one-step-ahead value of the VIX.

For the first dataset, were able to observe the regime-swiching phenomenon in stock market, by computing a kxk probability matrix. We were able to determine k by finding the optimal value for k which did not result in any overfitting or underfitting in the dataset.

For the second dataset in general, it was quite difficult to regress the model perfectly. Due to the nature of the stock market volatility index and the erratic nature of stock prices in general, the graphs were quite difficult to be fitted accurately. Even on the Elastic Regression algorithm, the line seemed to be underfitting, indicated that the presence of many trackers and external influences on the EMV were making it difficult for the machine learning algorithm to predict.