Lab_7_Regression_Part_2

September 27, 2021

1 LAB 7 : Regression Part 2

In this Lab we will look into the shortcomings of Linear Regression and see how those problems can be solved using Logistic Regression. We will also explore Polynomian Regression

1. Polynomial Regression
2. Linear Regression on a specific pattern of data to observe shortcomings
3. Logistic Regression to solve those problems

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

2 Polynomial Regression

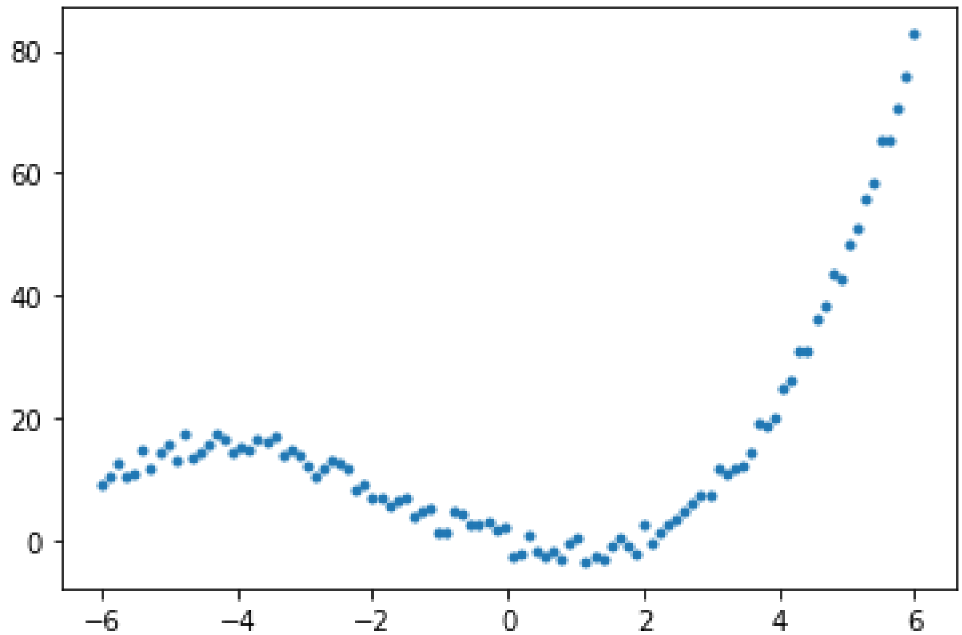
1. Generate data using relation $y = 0.25x^3 + 1.25x^2 - 3x - 3$
2. Corrupt y by adding random noise (uniformly sampled)
3. Fit the generated curve using different polynomial order. (Using matrix inversion and gradient descent)

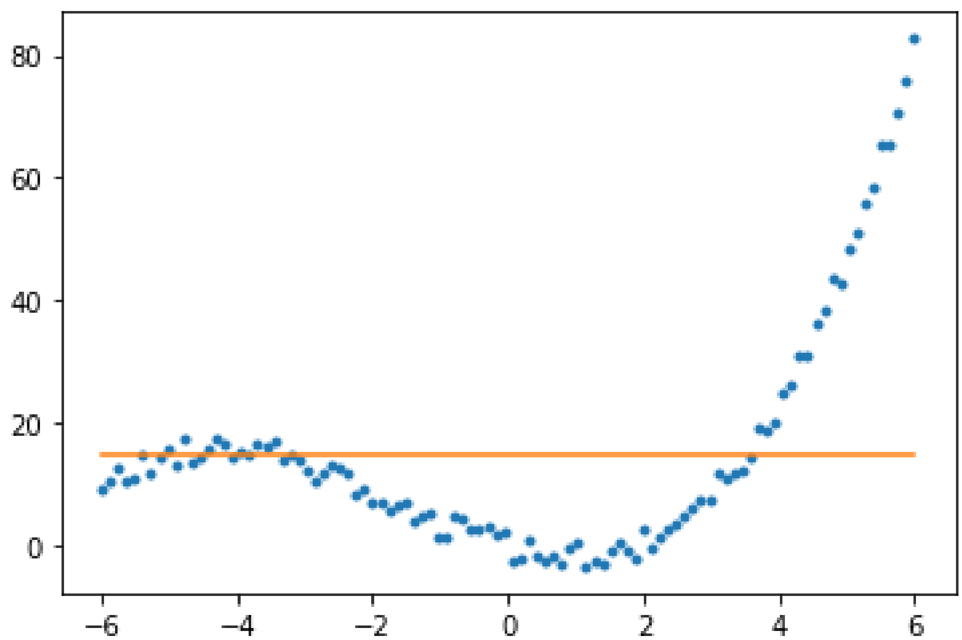
[1]:
```
## Use the Regression class defined in the previous lab
```

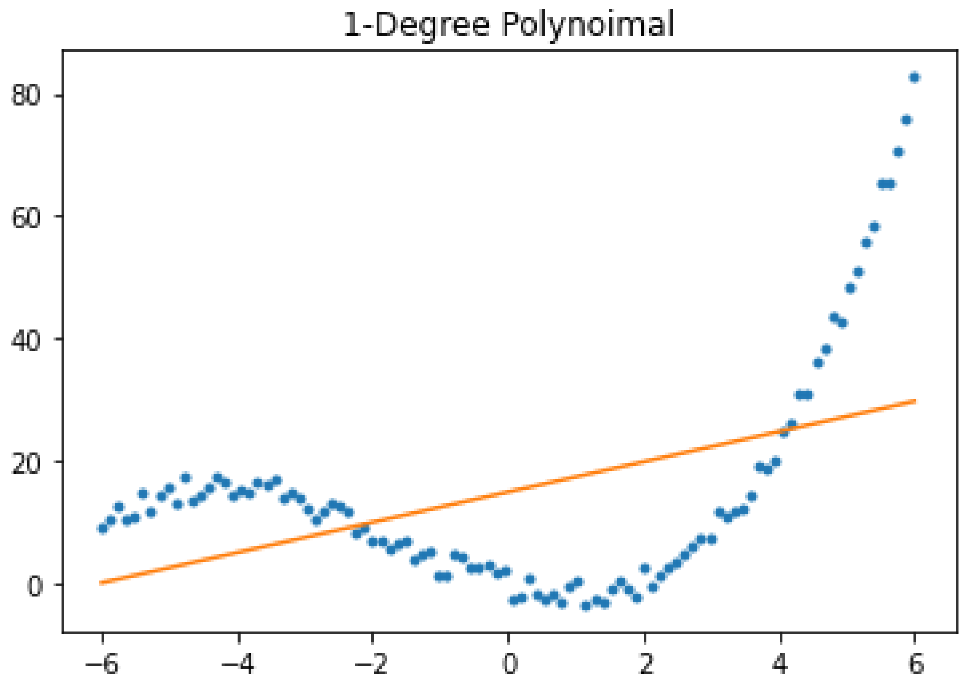
[3]:
```
## Data generation

x=np.linspace(-6,6,100)
x=x[np.newaxis,:]

w = ## Define Weights as per the given equation

## Function to transform the data into polynomial

def data_transform(X,degree):

  ## Write your code here

  return X_new
```

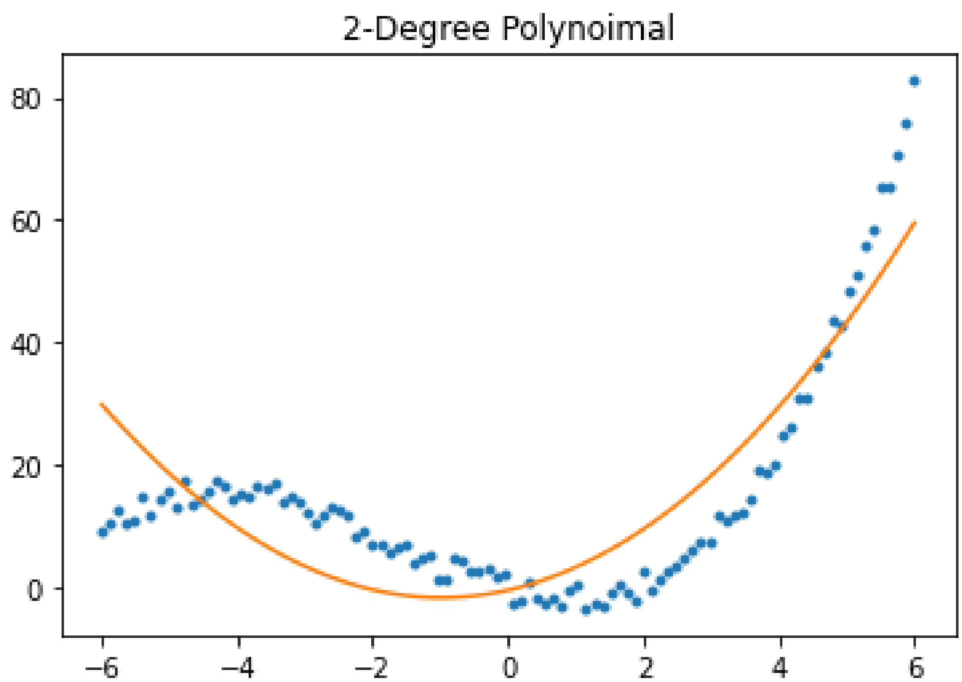
```
X = data_transform(x,3)

y = X.T @ w

y = y+5*np.random.uniform(0,1,y.shape)

plt.plot(x.T,y,'.')

reg=regression()

# By computation

# Code for degree 0 polynomial fitting

degree = 0
X_1 = data_transform(x,degree)
w_mat=reg.mat_inv(y,X_1)
y_pred=X_1.T @ w_mat
plt.figure()
plt.plot(x.T,y,'.')
plt.plot(x.T,y_pred)
plt.title('0-Degree Polynoimal')


# Write the code for degree 1 polynomial fitting

# Write the code for degree 2 polynomial fitting

# Write the code for degree 3 polynomial fitting

# Write the code for degree 4 polynomial fitting
```

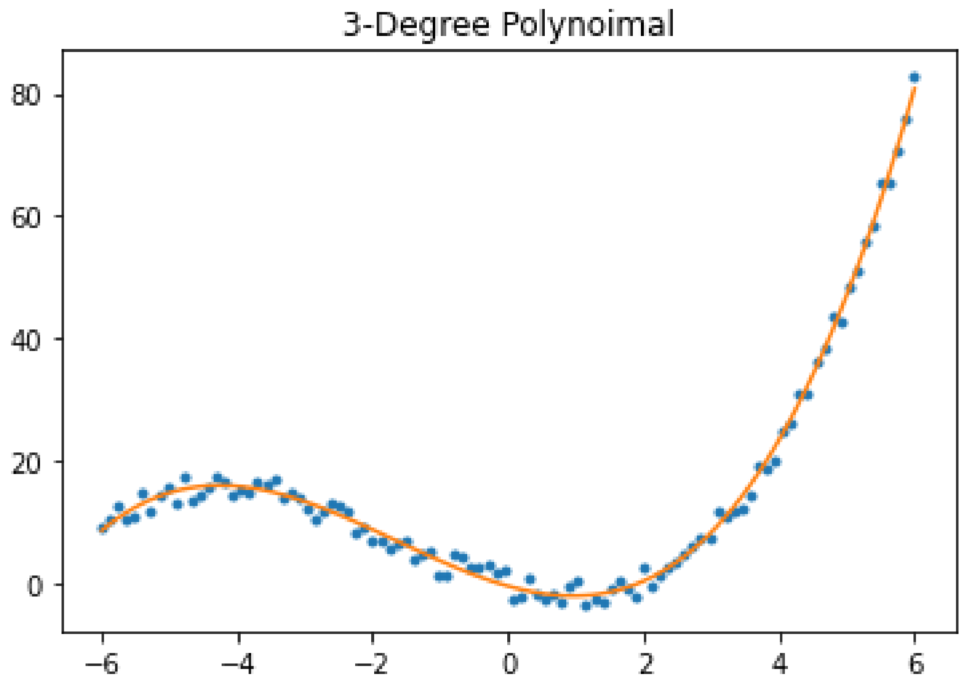
[3]:
```
Text(0.5, 1.0, '4-Degree Polynoimal')
```

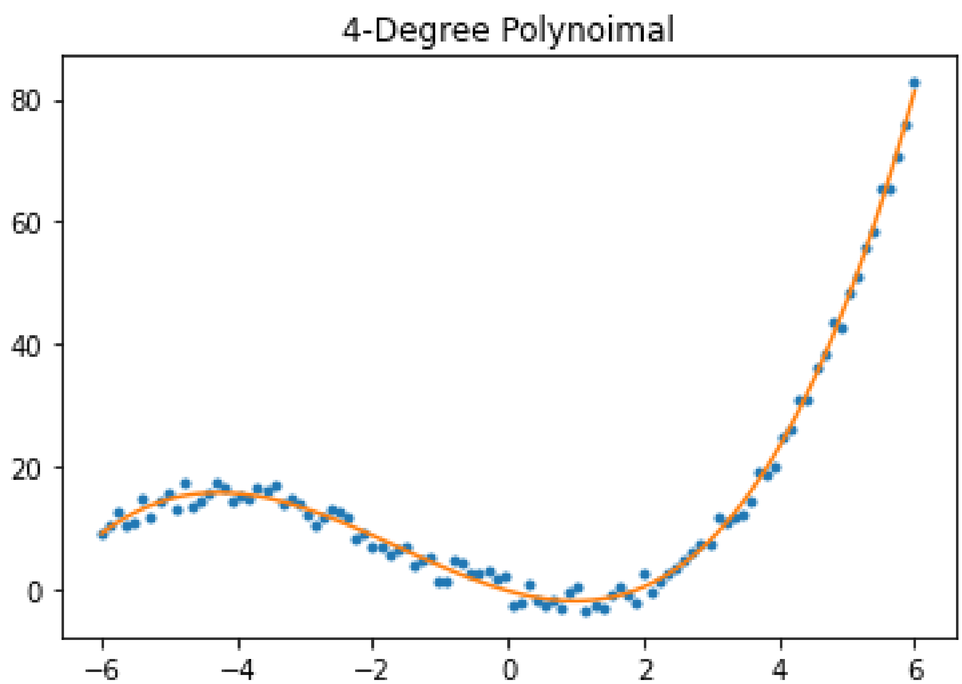

80
60
40
20
0
−6
−4
−2
0
2
4
6


80
60
40
20
0
−6
−4
−2
0
2
4
6


1-Degree Polynoimal
80
60
40
20
0
−6
−4
−2
0
2
4
6


2-Degree Polynoimal
80
60
40
20
0
−6
−4
−2
0
2
4
6





[ ]:
```
# By Gradient Descent
```

```
## Write your code here
```

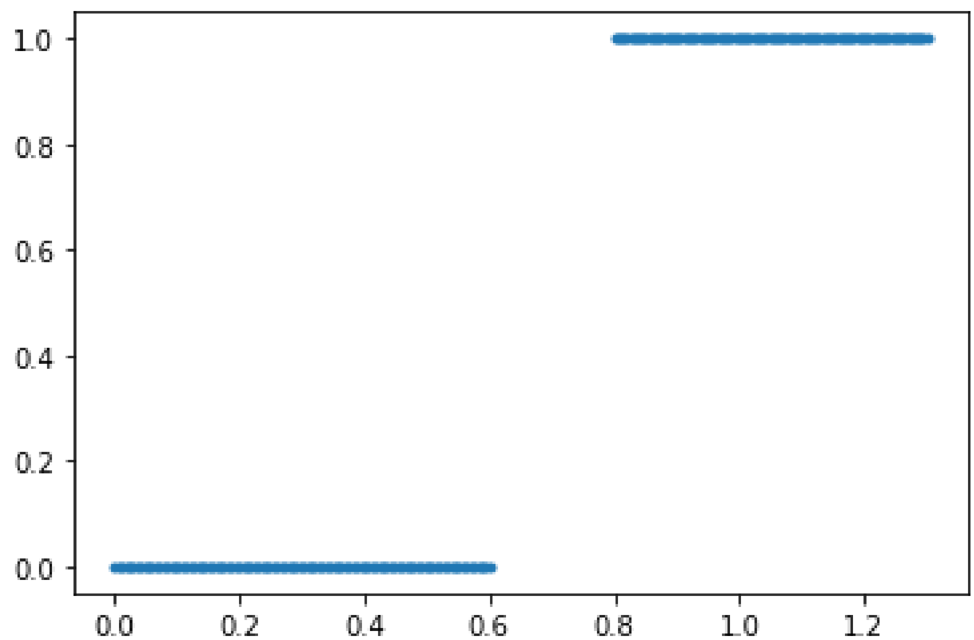
3 Linear Regression

Generate the data as shown in the figure below

[9]:
```
## Write your code here
```

```
(200,)
```

[9]:
```
[<matplotlib.lines.Line2D at 0x7f96a88ea110>]
```



Use the Regression class defined in the previous lab to fit the curve

[5]:
```
## Write your Code here
```

Augment the Data and generate optimal weights

[10]:
```
## Write your Code here
```

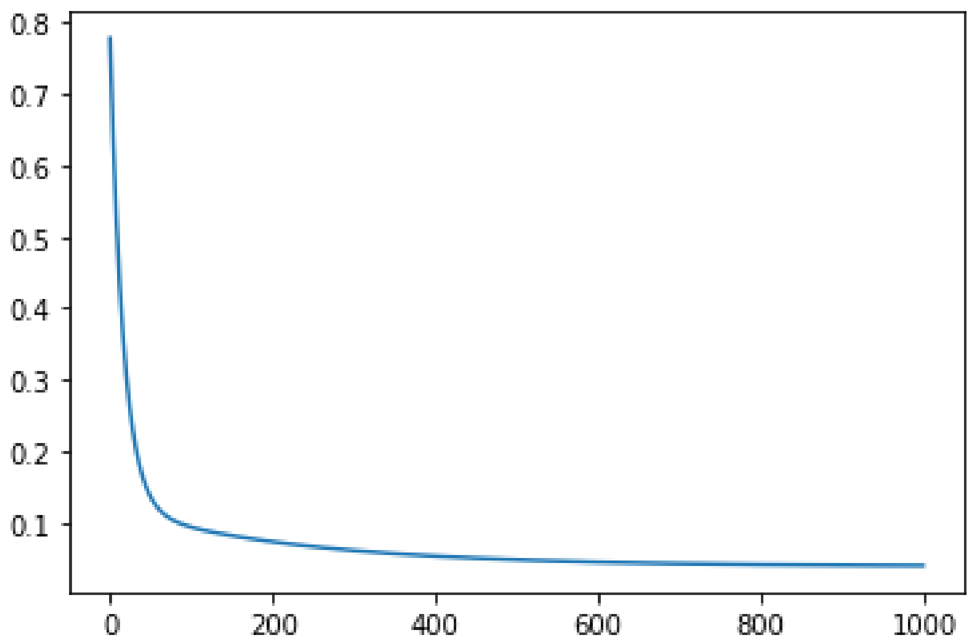
```
Shape of x :  (1, 200)
Shape of Augmented x :  (2, 200)
```

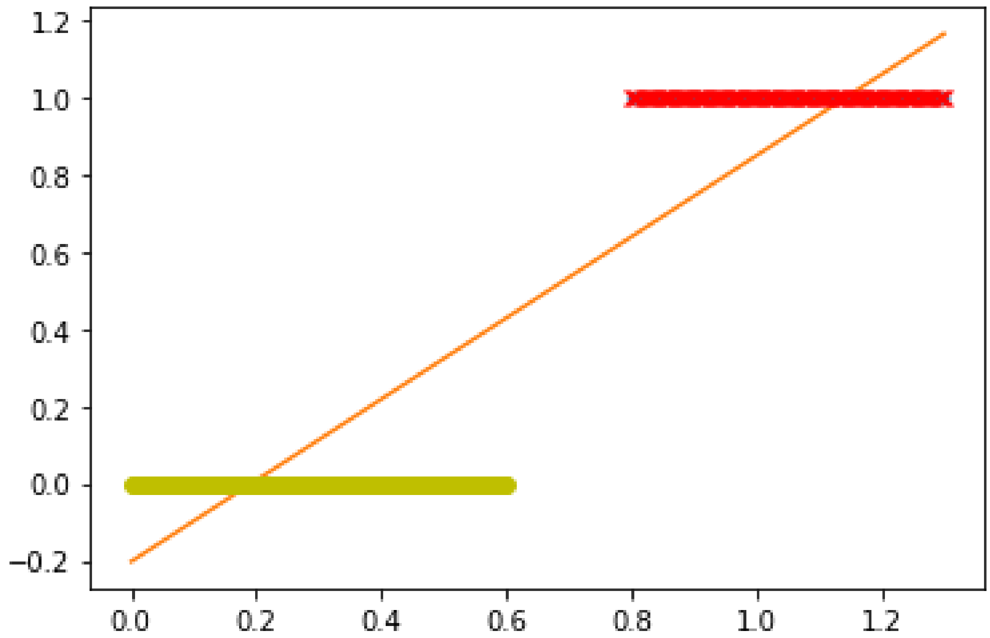
Using the optimal weights, fit the curve

[7]:
```
## Write your Code here
```

```
[[-0.25988351]
 [ 1.12575335]]
[[-0.20192789]
 [ 1.04935097]]
(2, 1)
(200, 1)
```

[7]: `[<matplotlib.lines.Line2D at 0x7f96a8a5b190>]`



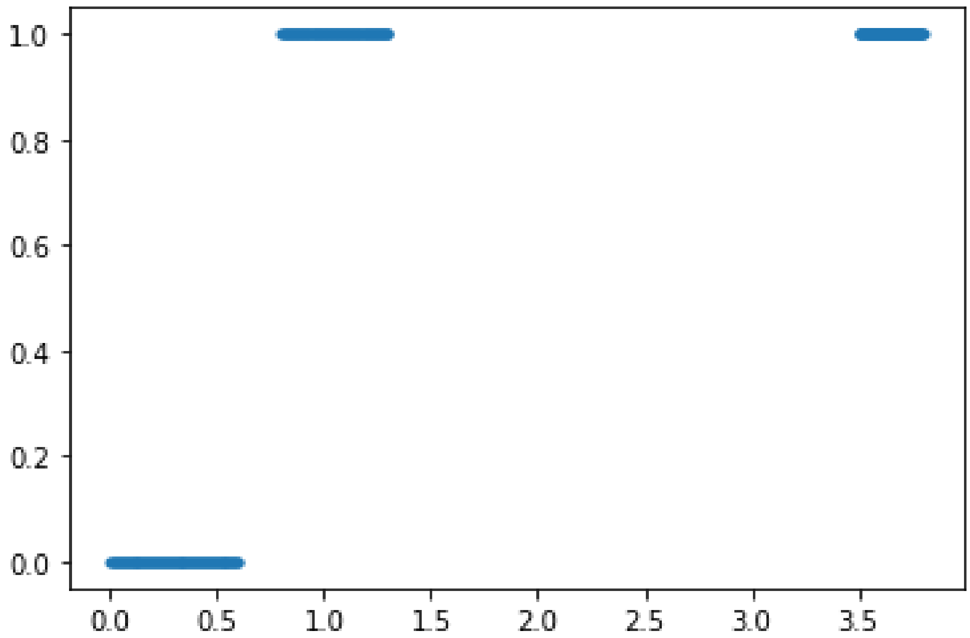


4 Drawback of Linear regression based Classificaton

Generate the Data as shown in the figure and follow the same steps as above to fit a curve using regression class

[12]:
```
## Write your code here
```

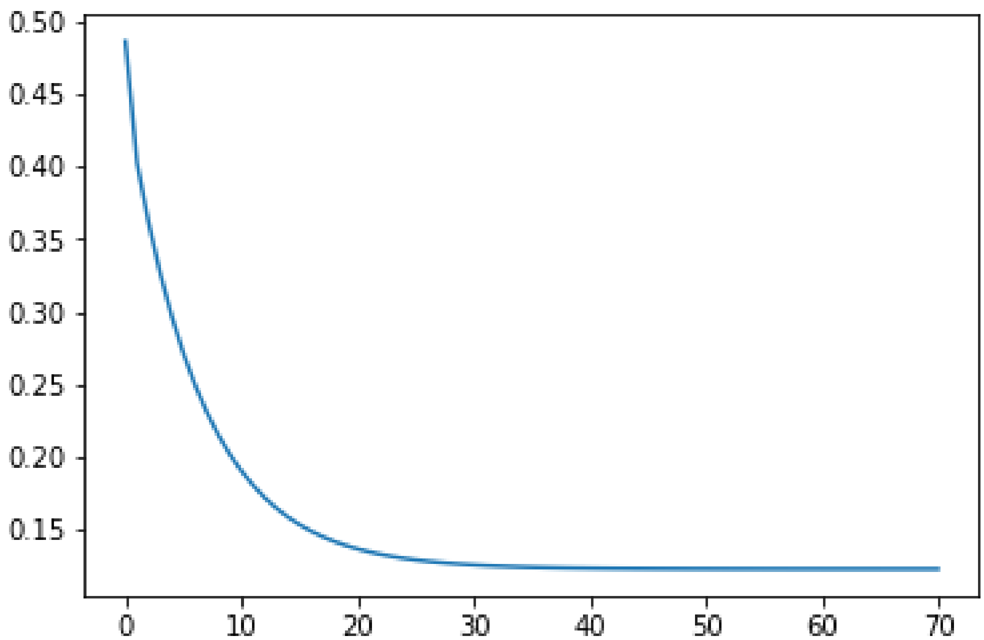
[12]:
```
[<matplotlib.lines.Line2D at 0x7f96a8bc0e10>]
```

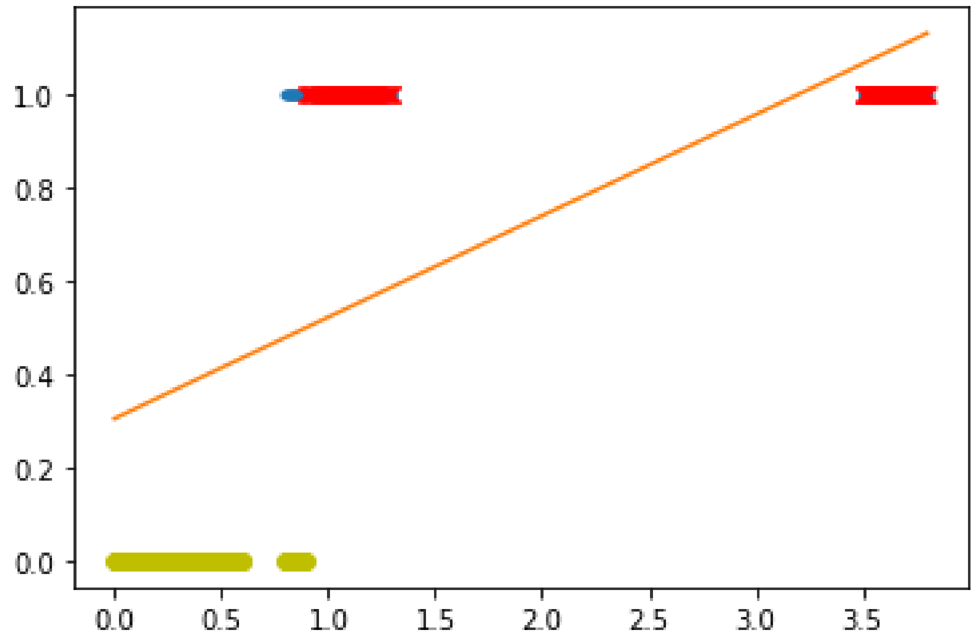


[13]:
```
## Write your code here
```

```
[[0.30516319]
 [0.21768954]]
(119,)
```

[13]: `[<matplotlib.lines.Line2D at 0x7f96a8b59990>]`



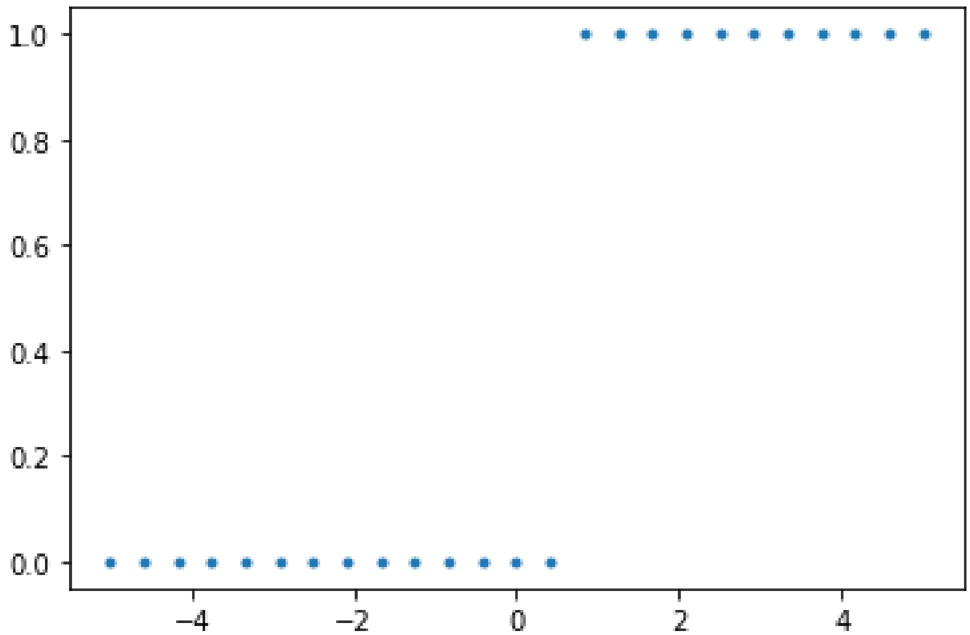


5 Logistic regression

Error Surface (Comparison between Logistic Loss and Mean Squared Error)

[14]:
```
import numpy as np
import matplotlib.pyplot as plt


x=np.linspace(-5,5,25)
y=np.zeros(x.shape)
y[np.where(x>0.7314)]=1

plt.plot(x,y,'.')
```

[14]:
```
[<matplotlib.lines.Line2D at 0x7f96a967aa90>]
```

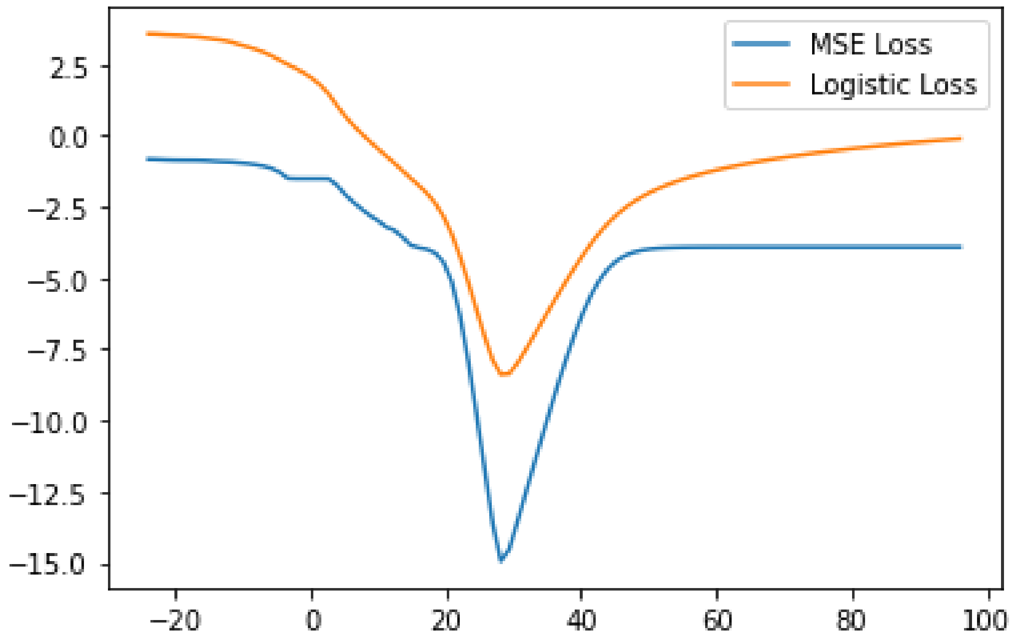


1. MSE=$\frac{1}{2N}\sum_{i=1}^{N}(y_i^p - y_i)^2$, where $y^p = \frac{1}{1+e^{-w^T x}}$
2. Logistic loss=$-\frac{1}{N}\sum_{i=1}^{N} y_i log(y_i^p) + (1-y_i) log(1-y_i^p)$

[15]:
```
# search space (only w1 is searched, where as w0 is fixed)
w1_in=10/(x[1]-x[0])
w0=-w1_in*0.7314
w1=np.linspace(-w1_in,4*w1_in,100)


cost_fn_mse=[]
cost_fn_logis=[]
for i in range(w1.shape[0]):

  # Compute Mean square error and logistic loss using cost function
  # Write your code here
```

[16]:
```
# Ploting of error surface
plt.figure()
plt.plot(w1,np.log(cost_fn_mse),label='MSE Loss')
plt.plot(w1,np.log(cost_fn_logis),label = 'Logistic Loss')
plt.legend()
```

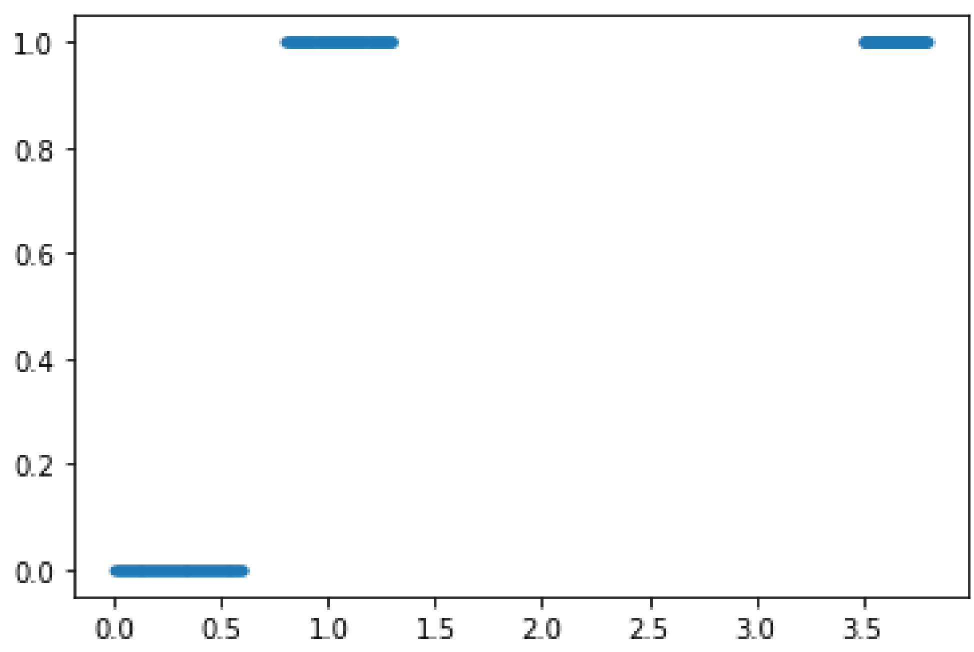
[16]: `<matplotlib.legend.Legend at 0x7f96a970e4d0>`



Solving the Outlier Issue
Generate the Data as shown in the figure

[17]:
```
## Write your Code here
```

[17]: [<matplotlib.lines.Line2D at 0x7f96b17b7250>]



Define a Logistic Regression class

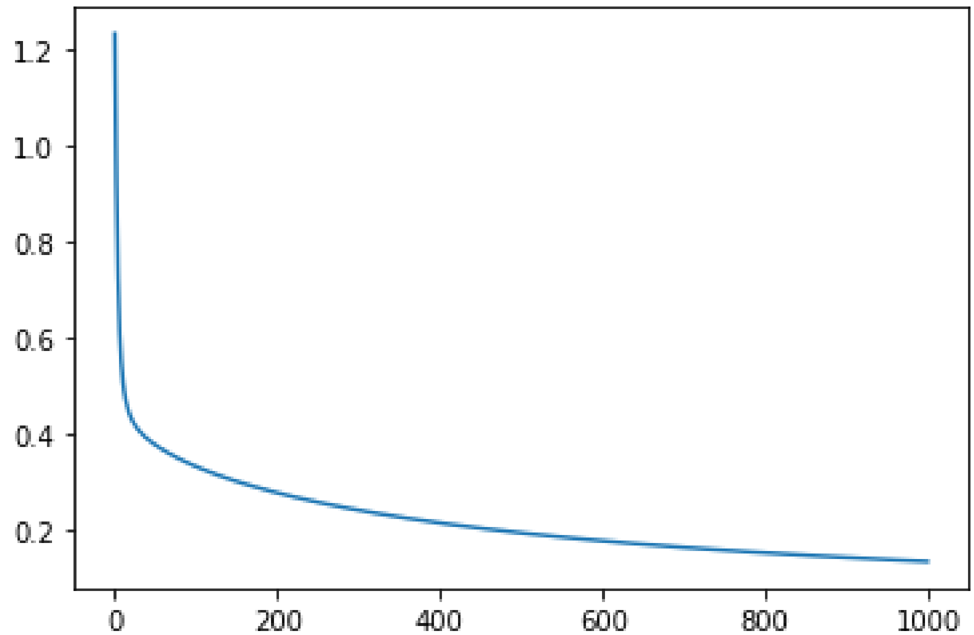
[18]:
```
class logis_regression:
  # Constructor
  def __init__(self, name='reg'):
    self.name = name  # Create an instance variable

  def logis(self,x,w_old):
    # write code here
    return op

  def grad_update(self,w_old,lr,y,x):
    # write code here
    return w

  def error(self,w,y,x):
    return # write code here

  def Regression_grad_des(self,x,y,lr):

    for i in range(1000):
      # write code here

      dev=np.abs(# write code here)

      if dev<=10**(-20):
        break

    return w_pred,err
```

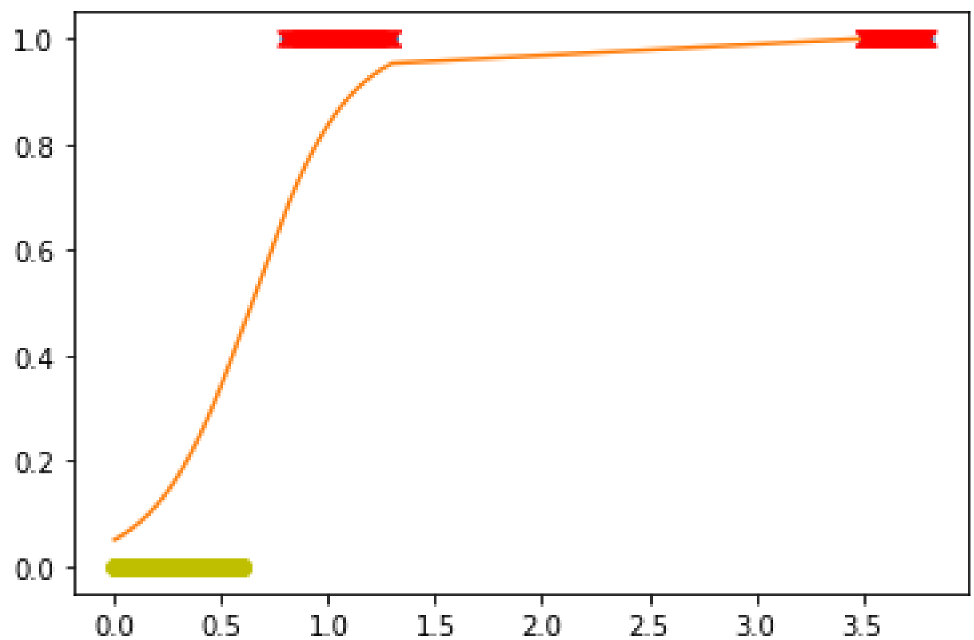
Augment the data and fit the curve by obtaining optimal weights (Using Gradient Descent)

[19]:
```
## Write your code here
```

```
[[-2.93565255]
 [ 4.57083202]]
(100,)
```

[19]: `[<matplotlib.lines.Line2D at 0x7f96a8d246d0>]`

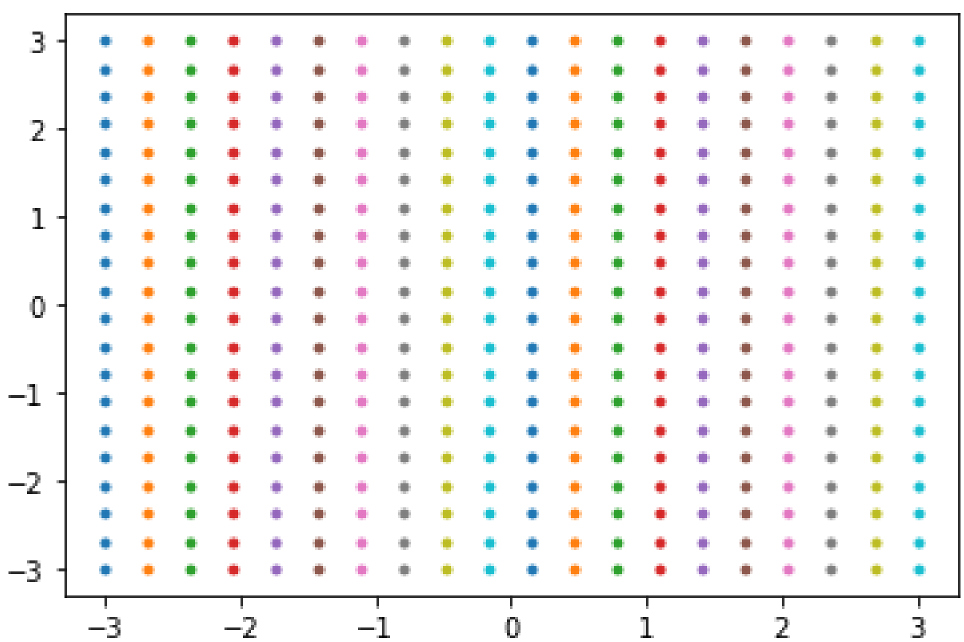

1.2
1.0
0.8
0.6
0.4
0.2
0
200
400
600
800
1000


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

6 Classification of circularly separated data using logistic regression

[20]:
```
x1=np.linspace(-3,3,20)
x2=np.linspace(-3,3,20)

x11,x22=np.meshgrid(x1,x2)
plt.plot(x11,x22,'.')
```

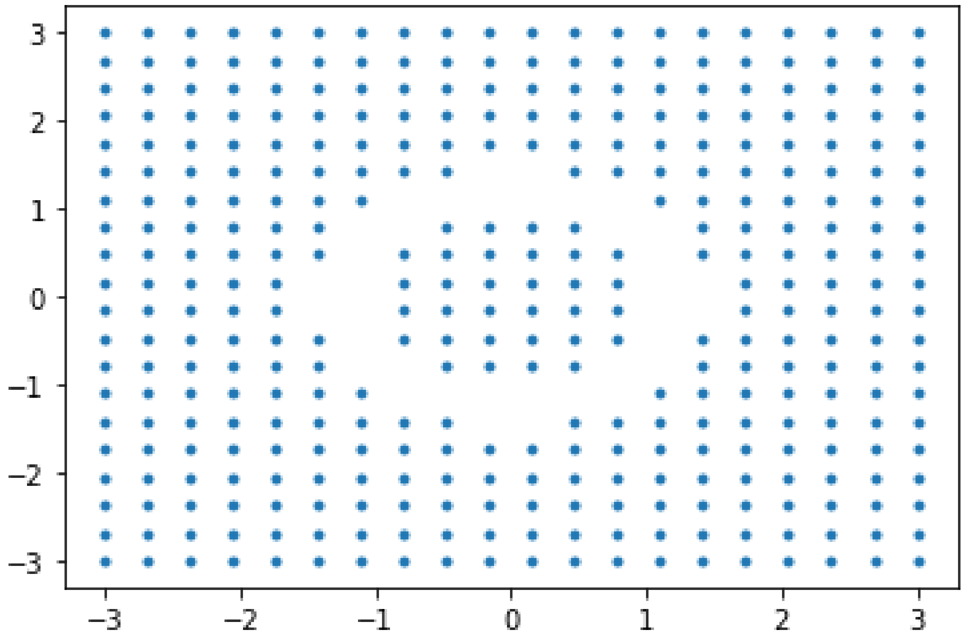
[20]:
```
[<matplotlib.lines.Line2D at 0x7f96b184b0d0>,
 <matplotlib.lines.Line2D at 0x7f96a8d32750>,
 <matplotlib.lines.Line2D at 0x7f96b184b490>,
 <matplotlib.lines.Line2D at 0x7f96b184b650>,
 <matplotlib.lines.Line2D at 0x7f96b184b810>,
 <matplotlib.lines.Line2D at 0x7f96b184b790>,
 <matplotlib.lines.Line2D at 0x7f96b184bc10>,
 <matplotlib.lines.Line2D at 0x7f96b184bdd0>,
 <matplotlib.lines.Line2D at 0x7f96b184bf90>,
 <matplotlib.lines.Line2D at 0x7f96b184ba10>,
 <matplotlib.lines.Line2D at 0x7f96a8d0db10>,
 <matplotlib.lines.Line2D at 0x7f96b1840450>,
 <matplotlib.lines.Line2D at 0x7f96b1840610>,
 <matplotlib.lines.Line2D at 0x7f96b18407d0>,
 <matplotlib.lines.Line2D at 0x7f96b1840990>,
 <matplotlib.lines.Line2D at 0x7f96b1840b50>,
 <matplotlib.lines.Line2D at 0x7f96b1840d10>,
 <matplotlib.lines.Line2D at 0x7f96b1840ed0>,
 <matplotlib.lines.Line2D at 0x7f96b1840e50>,
 <matplotlib.lines.Line2D at 0x7f96b1835290>]
```



Using the above data generate circular data

[21]:
```
# Write code here
```

[21]:
```
[<matplotlib.lines.Line2D at 0x7f96a9663f50>]
```

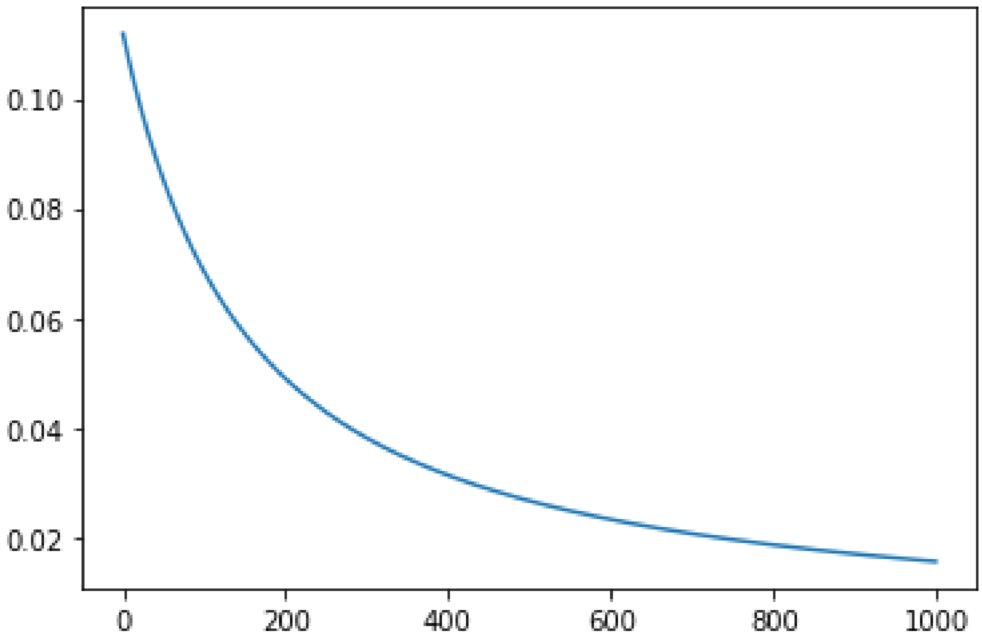


As in case of circularly separated data, the boundary is nonlinear, so squared feature is taken.

[ ]:
```
# perform logistic regression
```

```
(3, 364)
```

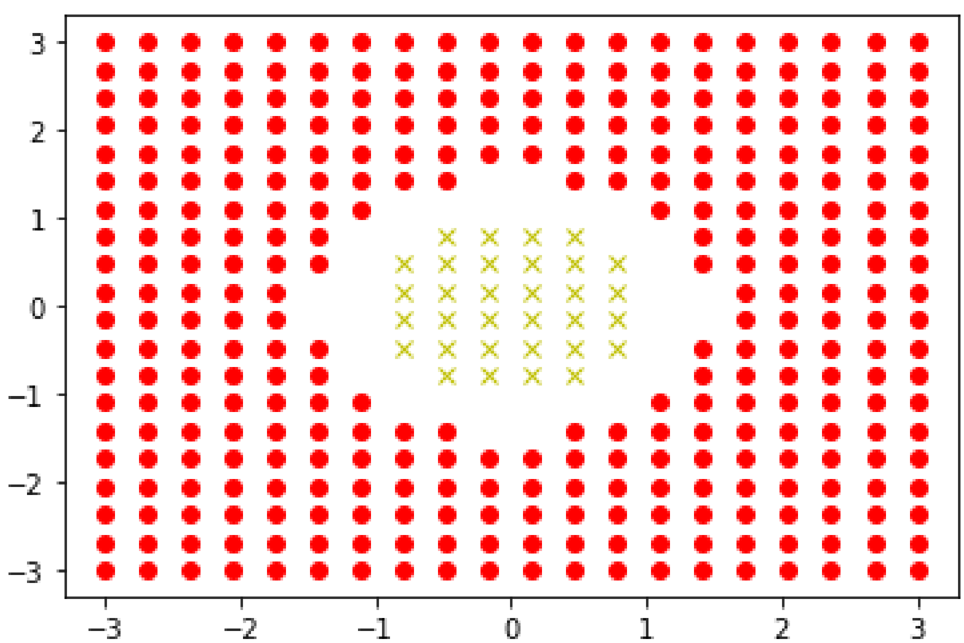
[ ]:
```
[<matplotlib.lines.Line2D at 0x7fbb1669c320>]
```



Plot classification using 0.5 as threshold

[ ]:
```
#write code here
```

[ ]: [<matplotlib.lines.Line2D at 0x7fbb164ad358>]



7 Multiclass logistic regression

1. Generate 1D data with 3 classes

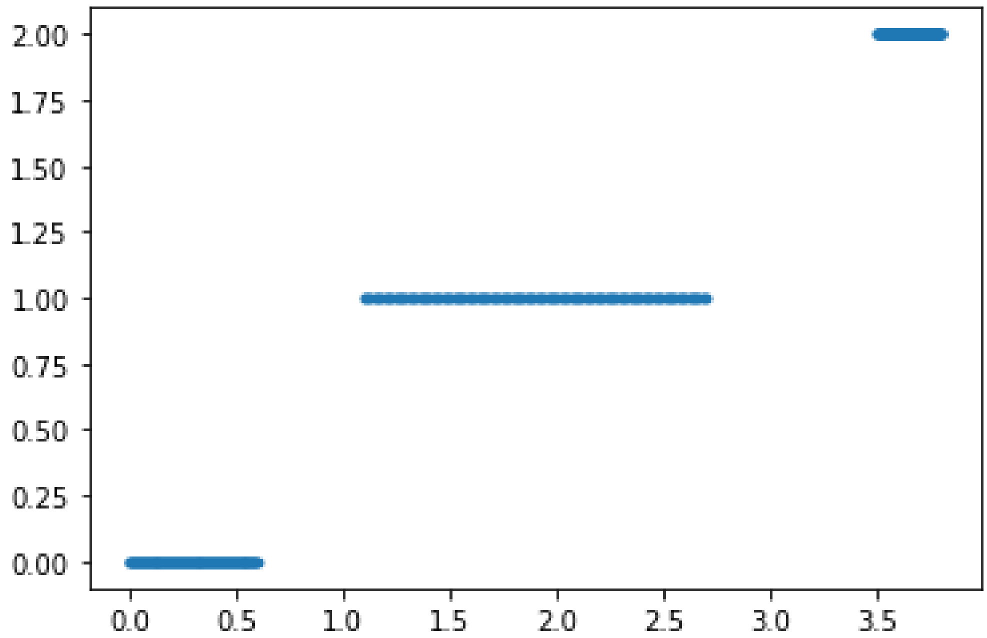
7.0.1 One vs rest classification

1. Lets take a polynomial of order 2 (by seeing the data distribution)

[22]:
```python
## Write your code here

import numpy as np
import matplotlib.pyplot as plt

x1=np.linspace(0,0.6,100)
x2=np.linspace(1.1,2.7,100)
x3=np.linspace(3.5,3.8,100)

x=np.concatenate((x1,x2,x3))
print(x.shape)

y1=np.zeros(x1.shape)
y2=np.ones(x2.shape)
y3=np.tile([2],x3.shape)

y=np.concatenate((y1,y2,y3))

plt.figure()
plt.plot(x,y,'.')
```

```
(300,)
```

[22]:
```
[<matplotlib.lines.Line2D at 0x7f96a8d12e10>]
```



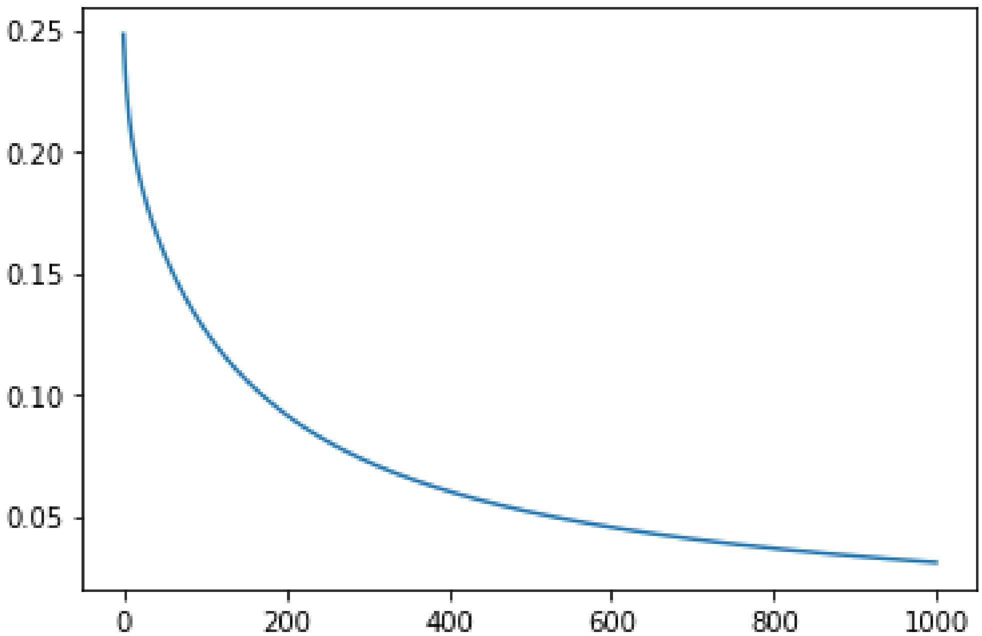
[23]:
```
# def data_transform(X,degree):
#   X_new=[]
#   for i in range(degree +1):
#     # write code here to generate a polynomial

def data_transform(X,degree):
  X_new=[]
  for i in range(degree +1):
    X_new.append(X**i)
  X_new = np.concatenate(X_new)
  return X_new

x_aug=data_transform(x[np.newaxis,:],2)
```

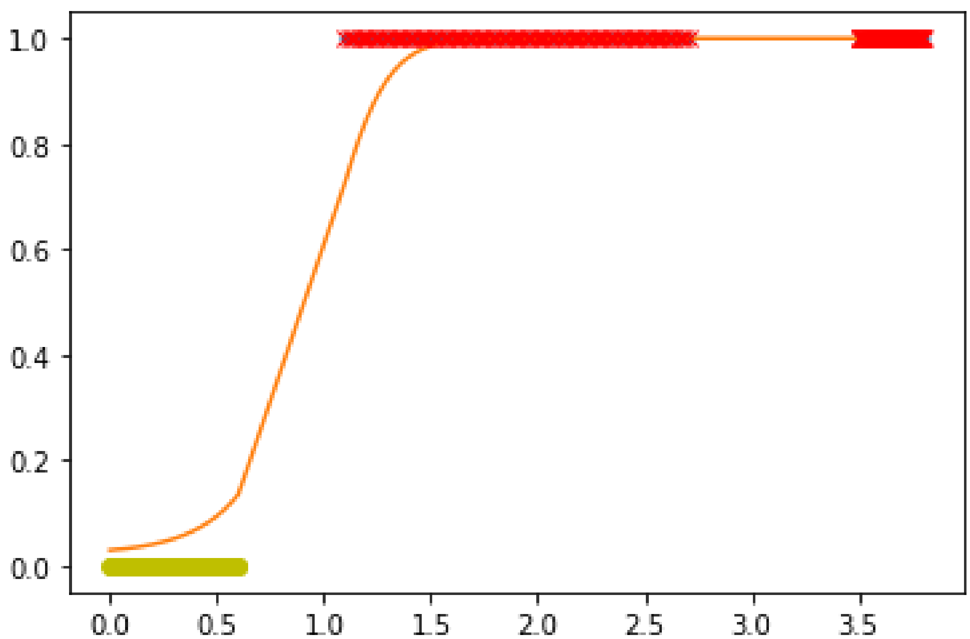
[ ]:
```
# plot for classification
def plot_op(x,y_pred):

  ind0,_=np.where(y_pred<0.5)
  ind1,_=np.where(y_pred>=0.5)
  x0=x[ind0,:]
  x1=x[ind1,:]
  plt.plot(x0,np.zeros((x0).shape),'o',color='y')
  plt.plot(x1,np.ones((x1).shape),'x',color='r')
```

Using the above function for plotting, plot the curve using different configurations

[25]:
```
# take class 0 as '0' and other to '1'
## Write your code here
```

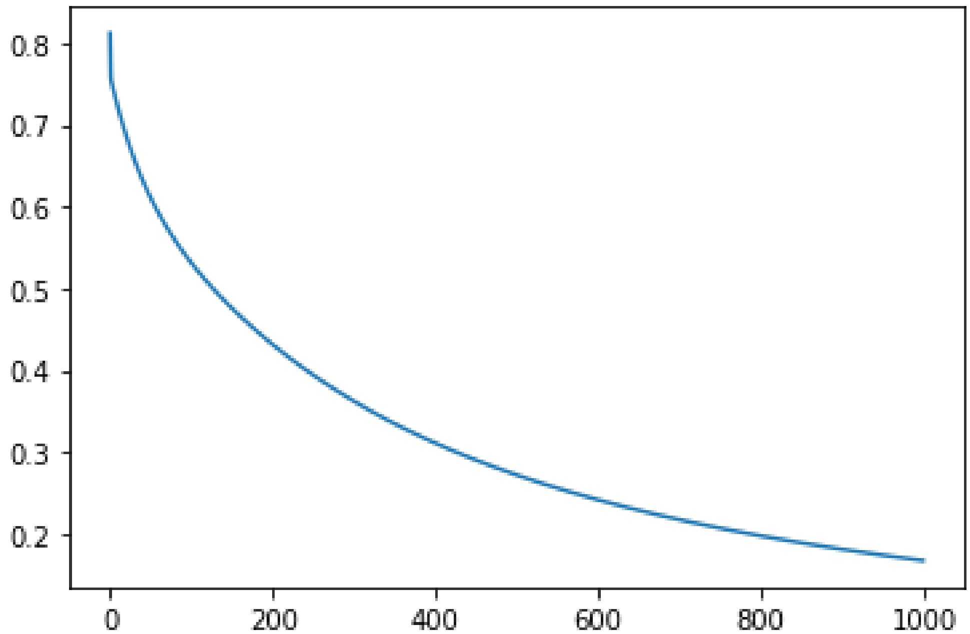
```
[[-3.50135734]
 [ 1.13204612]
 [ 2.67652528]]
```

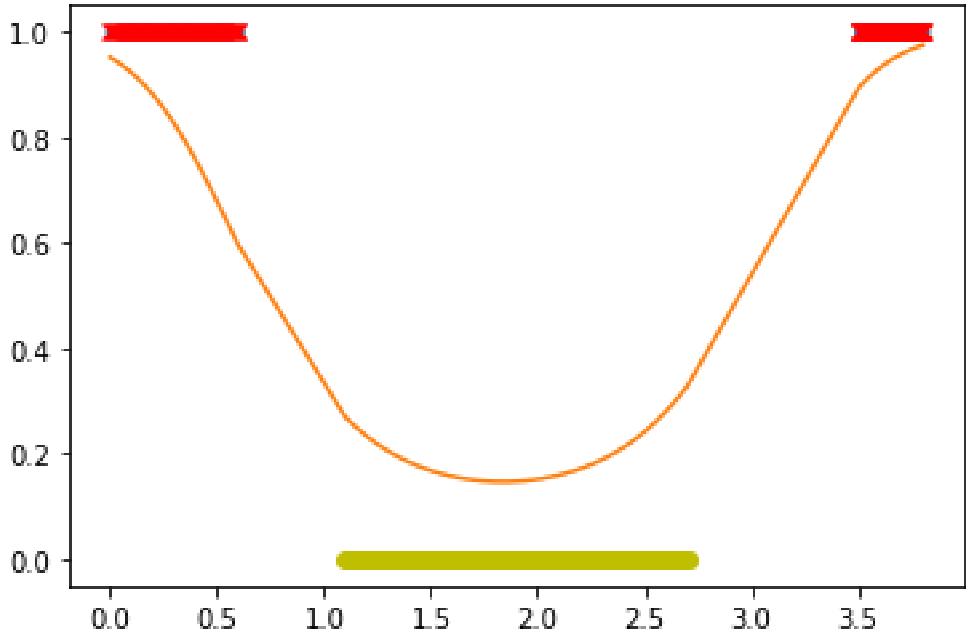




[26]:
```
# take class 1 as '0' and other to '1'
## Write your code here
```

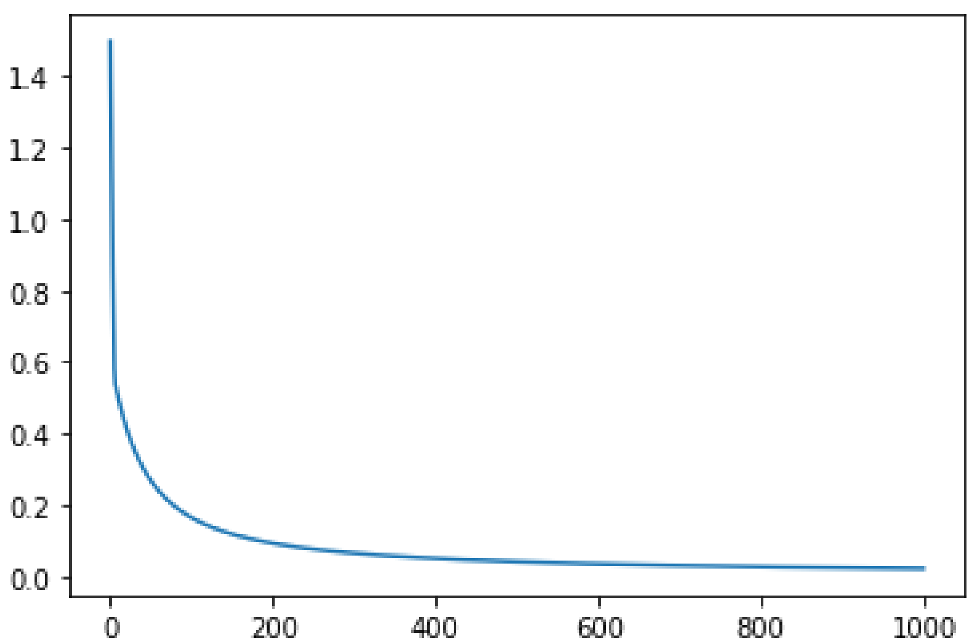
```
[[ 2.97520857]
 [-5.15587639]
 [ 1.40348805]]
```

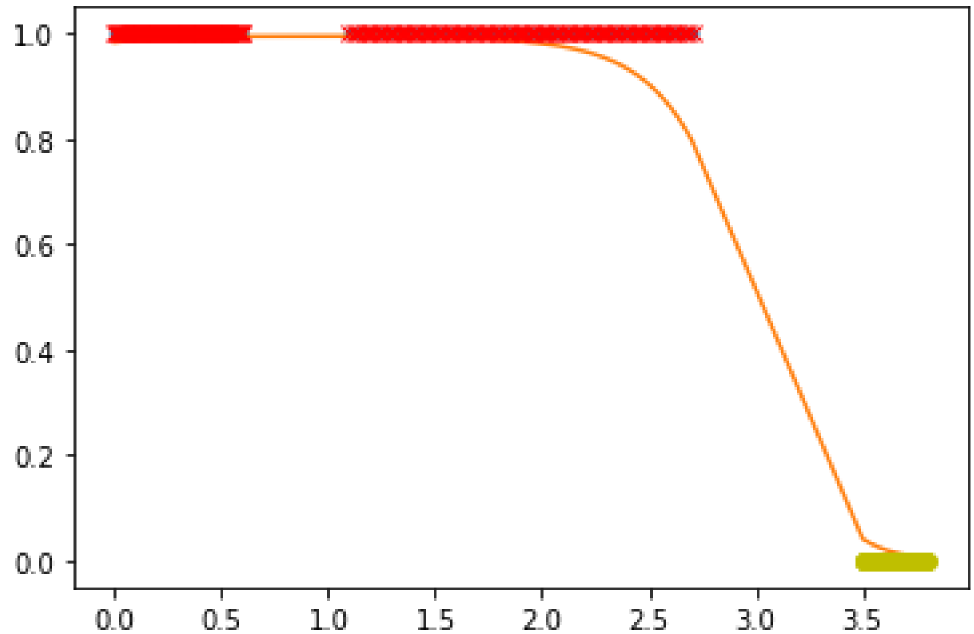




[27]:
```
# Take class 2 as '0' and other to '1'
## Write your code here
```

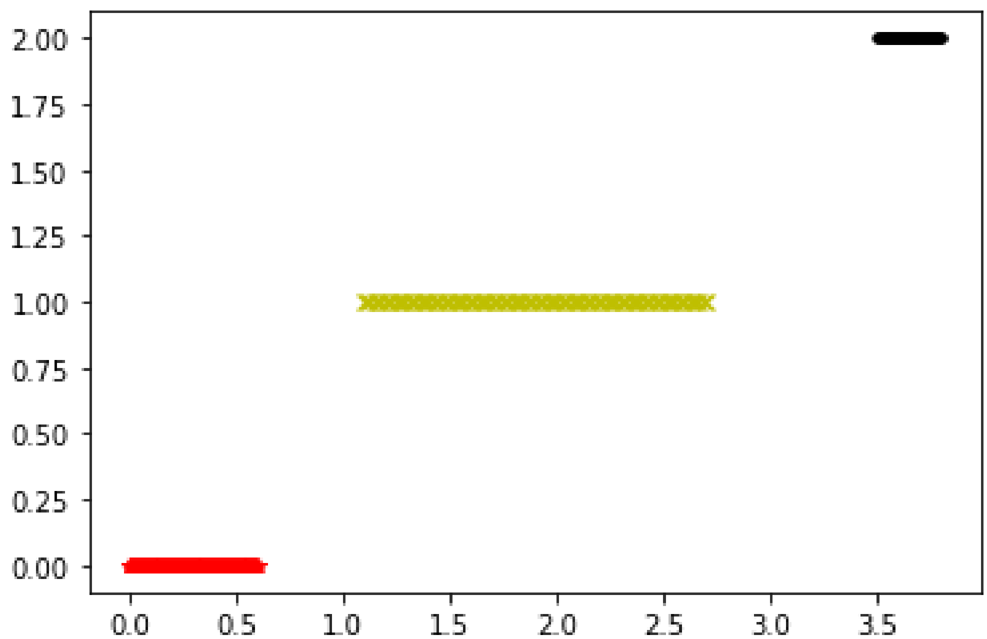
```
[[ 4.09124596]
 [ 2.57217703]
 [-1.3253653 ]]
```





[28]:
```
# final classification
## Write your code here
```

[28]: `[<matplotlib.lines.Line2D at 0x7f96a89dba90>]`



[30]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-generic-recommended

!jupyter nbconvert --to pdf '/content/Classification_V2.ipynb'
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
texlive-fonts-recommended is already the newest version (2017.20180305-1).
texlive-generic-recommended is already the newest version (2017.20180305-1).
texlive-xetex is already the newest version (2017.20180305-1).
0 upgraded, 0 newly installed, 0 to remove and 37 not upgraded.
[NbConvertApp] Converting notebook /content/Classification_V2.ipynb to pdf
[NbConvertApp] Support files will be in Classification_V2_files/
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Making directory ./Classification_V2_files
[NbConvertApp] Writing 52156 bytes to ./notebook.tex
[NbConvertApp] Building PDF
```

```
[NbConvertApp] Running xelatex 3 times: [u'xelatex', u'./notebook.tex',
'-quiet']
[NbConvertApp] Running bibtex 1 time: [u'bibtex', u'./notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 259313 bytes to /content/Classification_V2.pdf
```