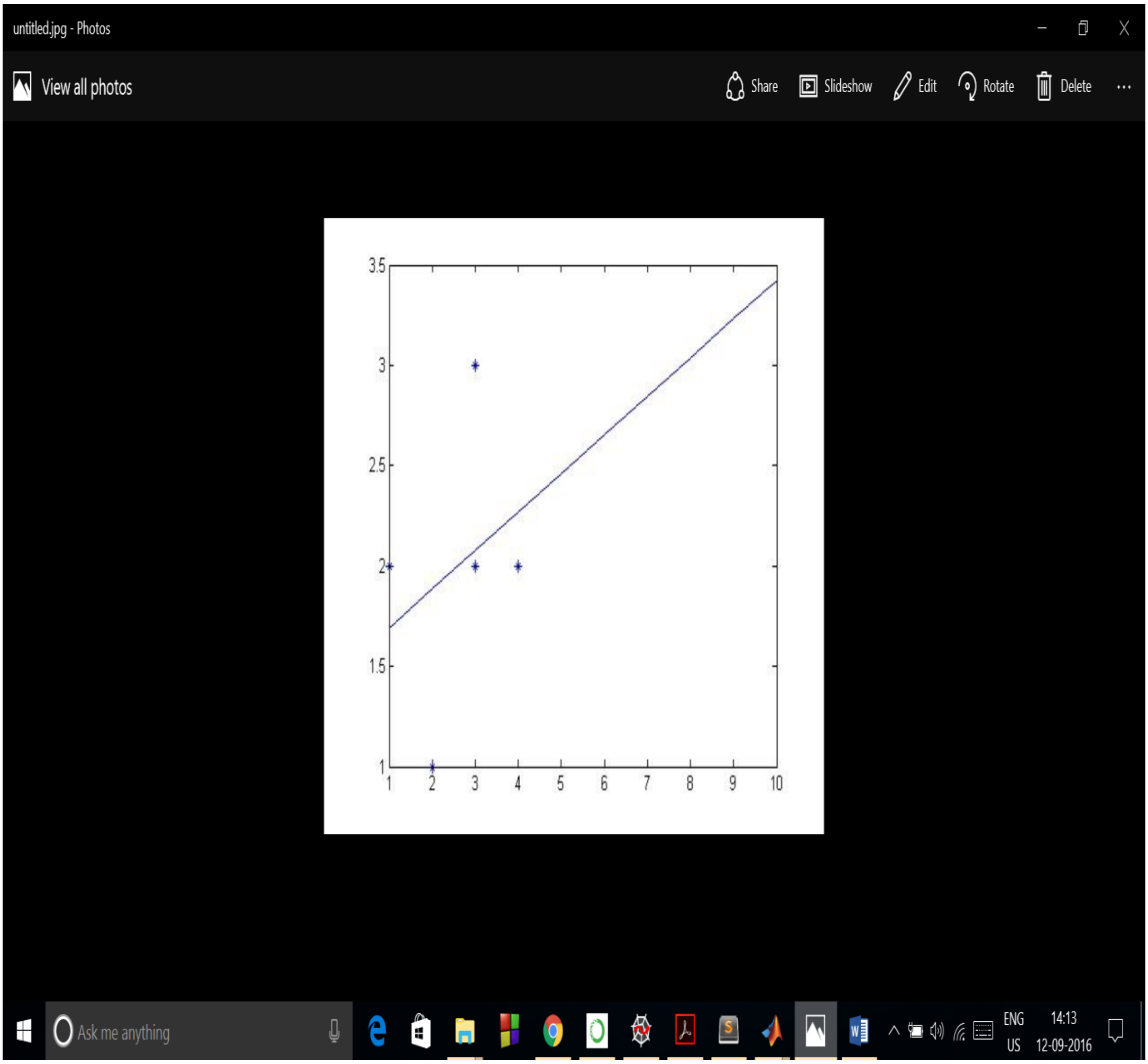
Assignment 2

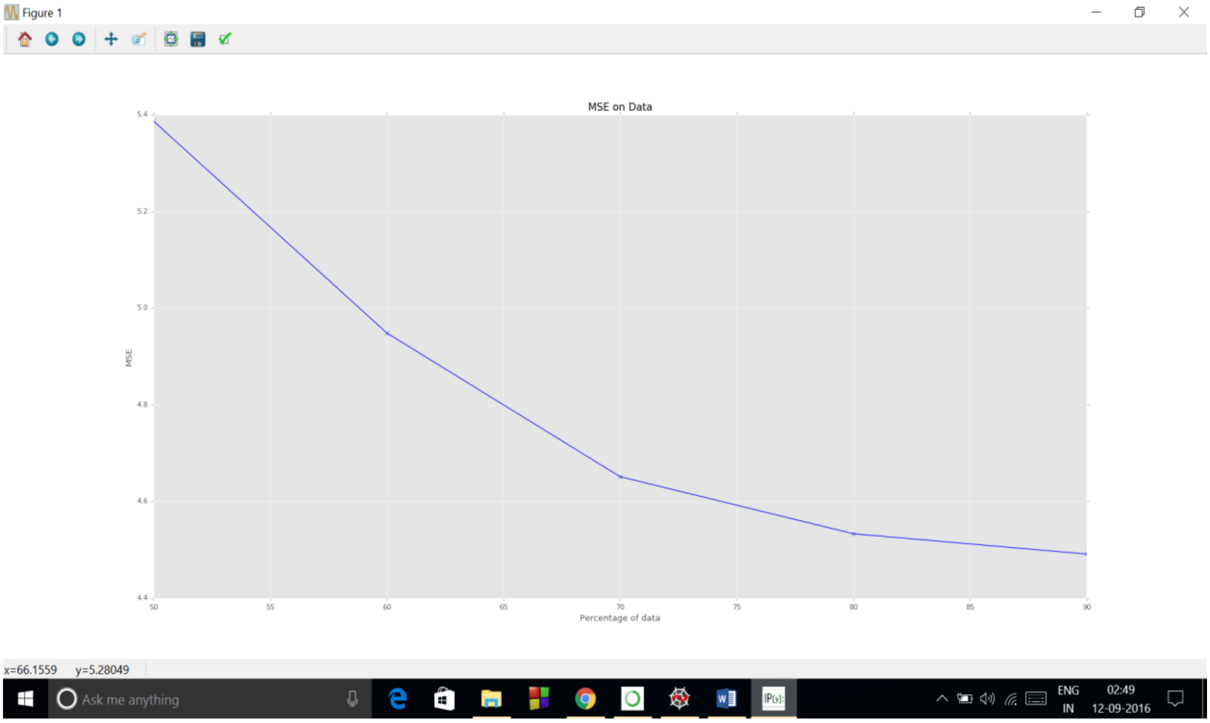
Ans 3).



Ans 4).

A).

lin.txt:

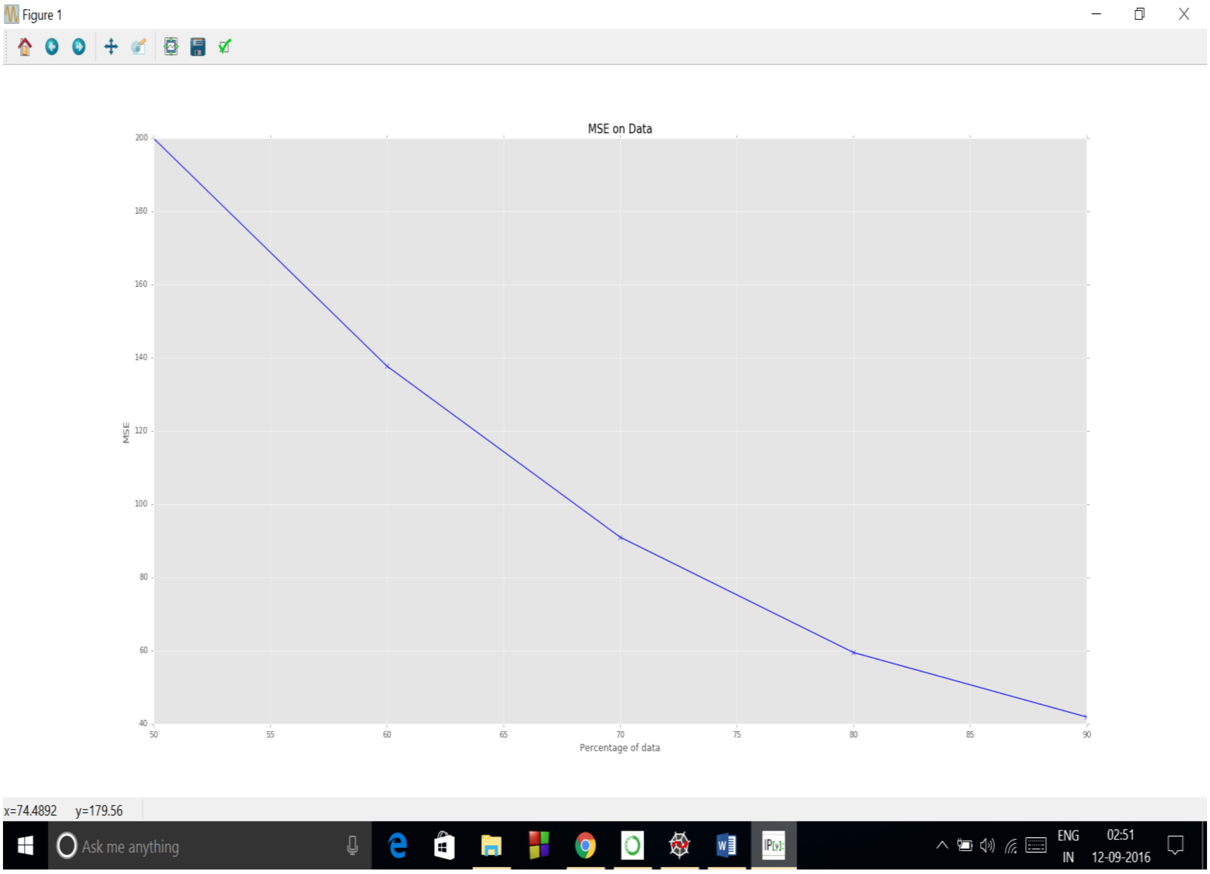


Final_parameters:

[[-3.83246815]

 [1.20054998]]

Sph.txt:

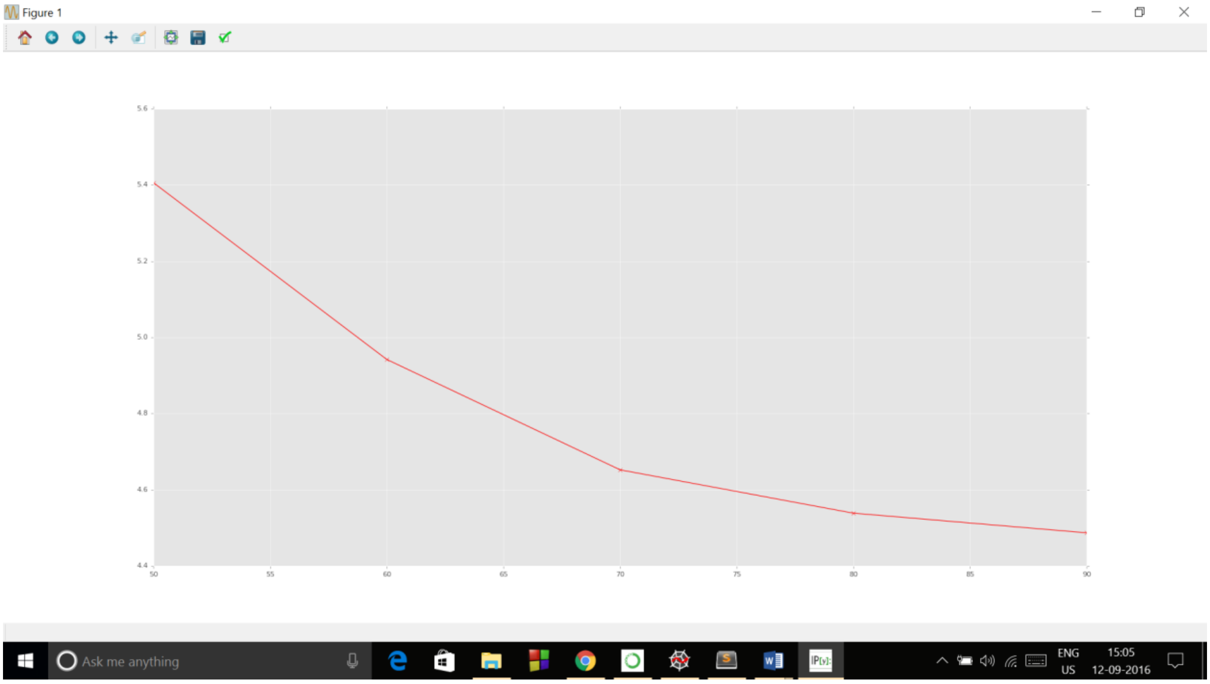


Final_parameters:

```
[[-12.28637391]
 [  0.95160029]]
```

B). lin.txt

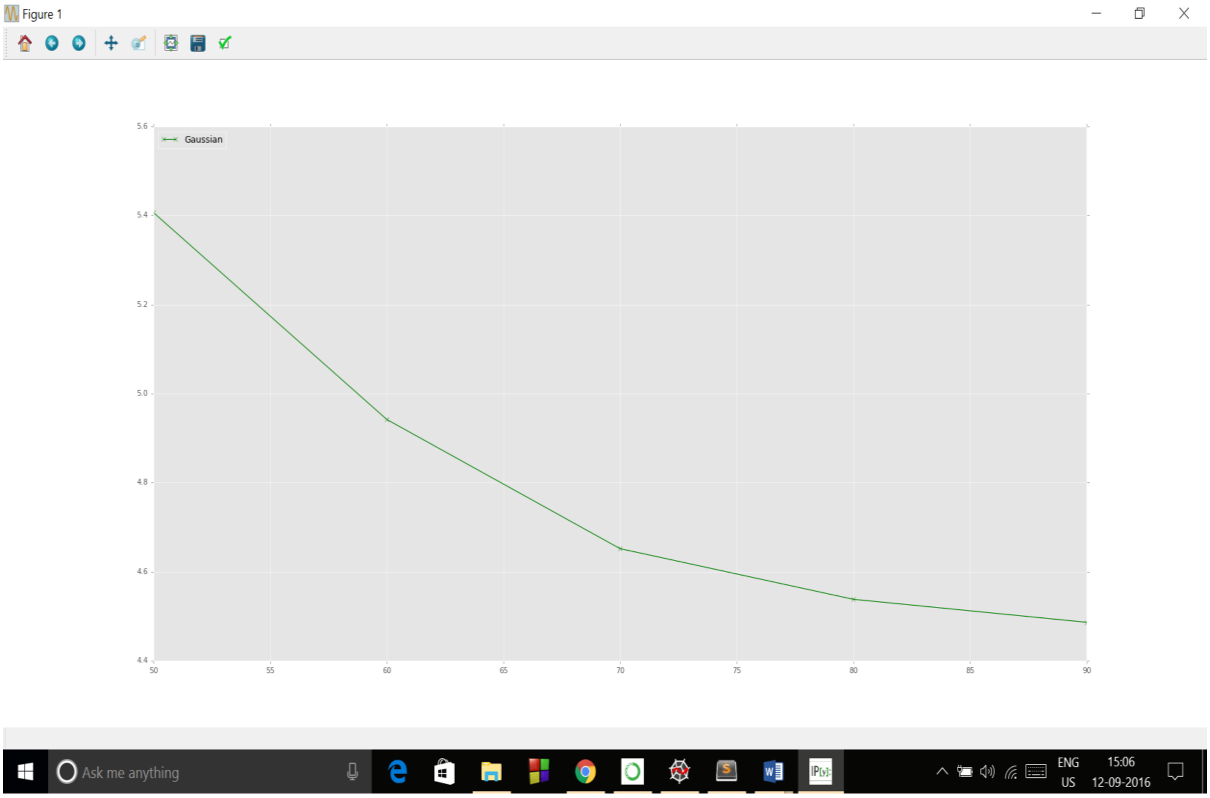
Polynomial:



Final_parameters:

```
[[ -6.64533714e+00]
 [  2.05992598e+00]
 [ -7.60642322e-02]
 [  1.98547372e-03]]
```

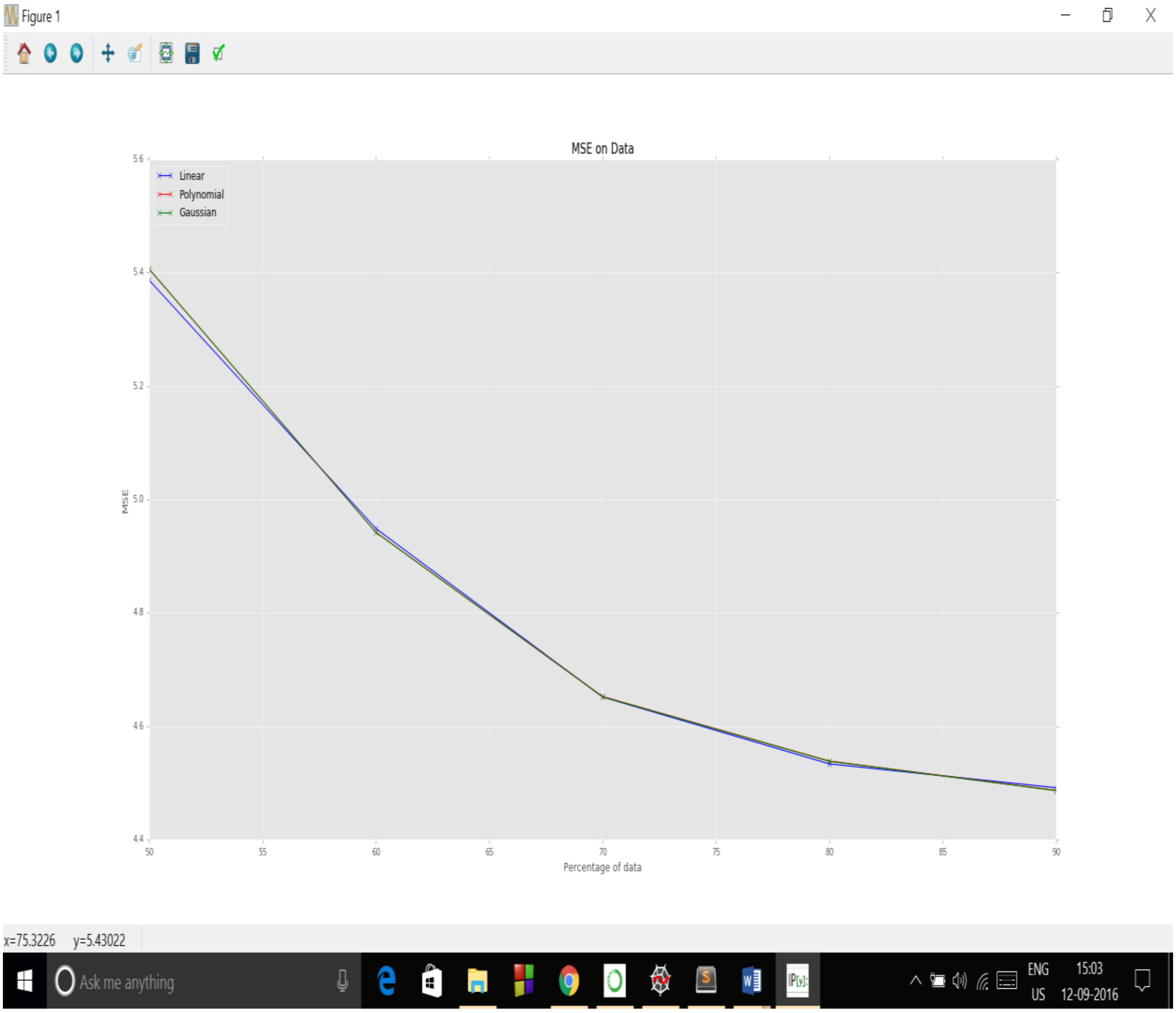
Gaussian:



Final_parameters:

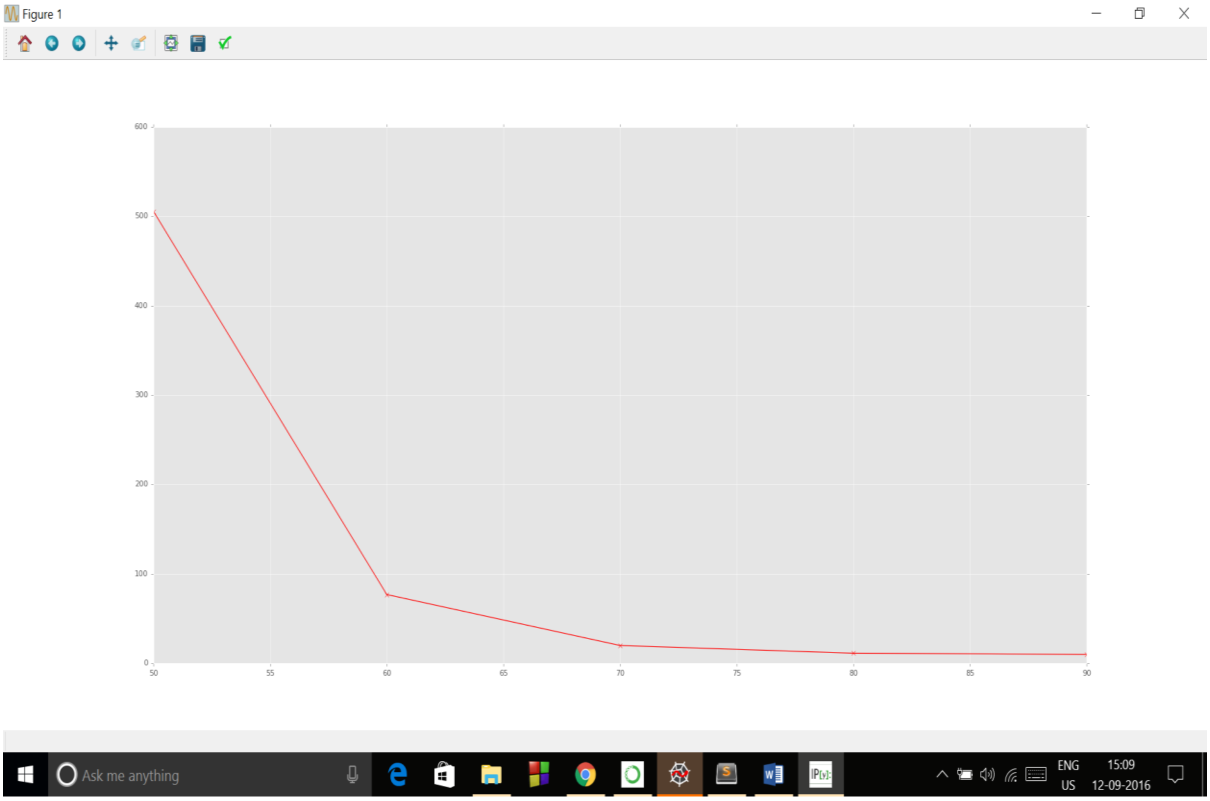
```
[[   -6.76938133]
 [  148.45639756]
 [ -567.14117141]
 [ 1915.29457624]]
```

All the three in same plot:



sph.txt:

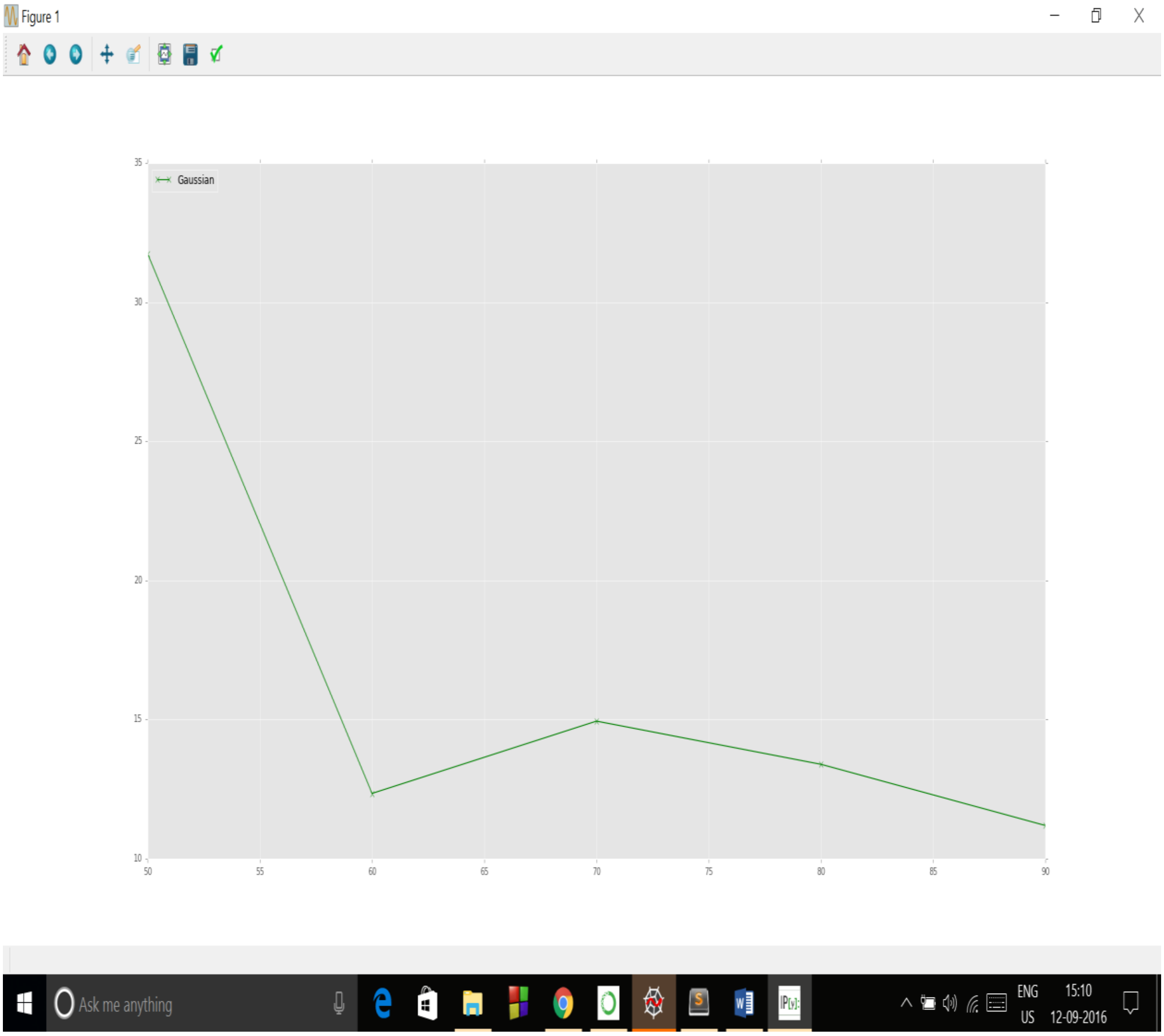
Polynomial:



Final_parameters:

```
[[  9.53078664e-01]
 [  1.56195472e-01]
 [  6.69055133e-03]
 [  2.50877482e-05]]
```

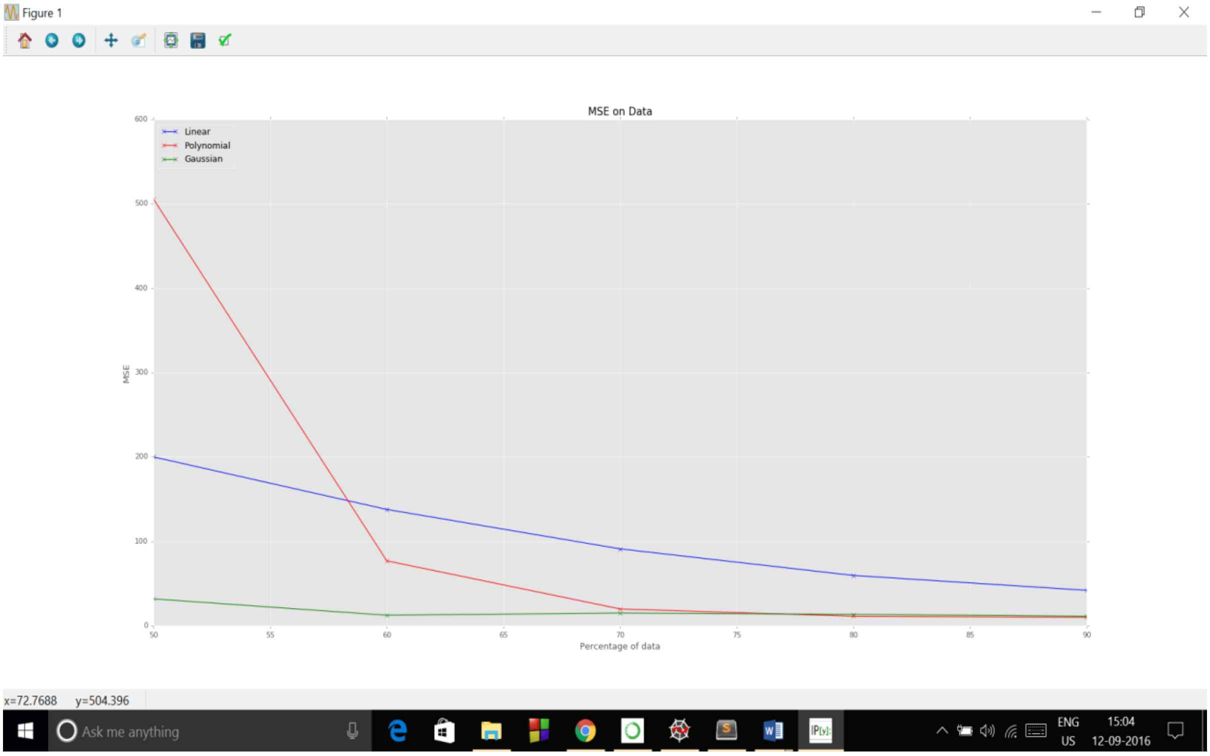
Gaussian:



Final_parameters:

```
[[  -1.61926295]
 [  48.13812614]
 [-120.94531105]
 [ 319.61259127]]
```

All the three in same plot:



C).

lin.txt:

Using linear kernel:

Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:

```
[[ 5.38669943]
 [ 4.94857837]
 [ 4.65143337]
 [ 4.53341478]
 [ 4.49135934]]
```

Final_parameters:

```
[[-3.77399579]
 [ 1.19872364]]
```

For polynomial kernel I have used 3 degree of polynomial for polynomial kernel. I have used 3 as degree of polynomial.

Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:

```
[[ 5.40603452]
 [ 4.94200299]
 [ 4.65225806]
 [ 4.53839634]
 [ 4.48682286]]
```

Final_parameters:

```
[[ -6.64533714e+00]
 [  2.05992598e+00]
 [ -7.60642322e-02]
 [  1.98547372e-03]]
```

For Gaussian kernel:

```
Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:
[[ 5.40655131]
 [ 4.94186324]
 [ 4.65169658]
 [ 4.53792331]
 [ 4.48622959]]
Final_parameters:
[[   -6.76938133]
 [  148.45639756]
 [ -567.14117141]
 [ 1915.29457624]]
```

So, we observe that Gaussian and Polynomial gives approx the same result but Gaussian kernel is little better.

sph.txt:

Using linear kernel:

```
Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:
[[ 199.95745047]
 [ 137.75402046]
 [  91.0131034 ]
 [  59.49966616]
 [  41.87986043]]
Final_parameters:
[[-12.28637391]
 [  0.95160029]]
```

Polynomial kernel:

For polynomial kernel I have used 3 degree of polynomial for polynomial kernel. I have used 3 as degree of polynomial.

Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:

```
[[ 505.53585954]
 [  76.83831084]
 [  19.80531261]
 [  11.15582704]
 [   9.79429605]]
```

Final_parameters:

```
[[  9.53078664e-01]
 [  1.56195472e-01]
 [  6.69055133e-03]
 [  2.50877482e-05]]
```

Gaussian kernel:

Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:

```
[[ 31.73330215]
 [ 12.34528619]
 [ 14.94332305]
 [ 13.39268413]
 [ 11.19358357]]
```

Final_parameters:

```
[[  -1.61926295]
 [  48.13812614]
 [-120.94531105]
 [ 319.61259127]]
```

So, we observe that polynomial kernel gives best result.

D).

lin.txt

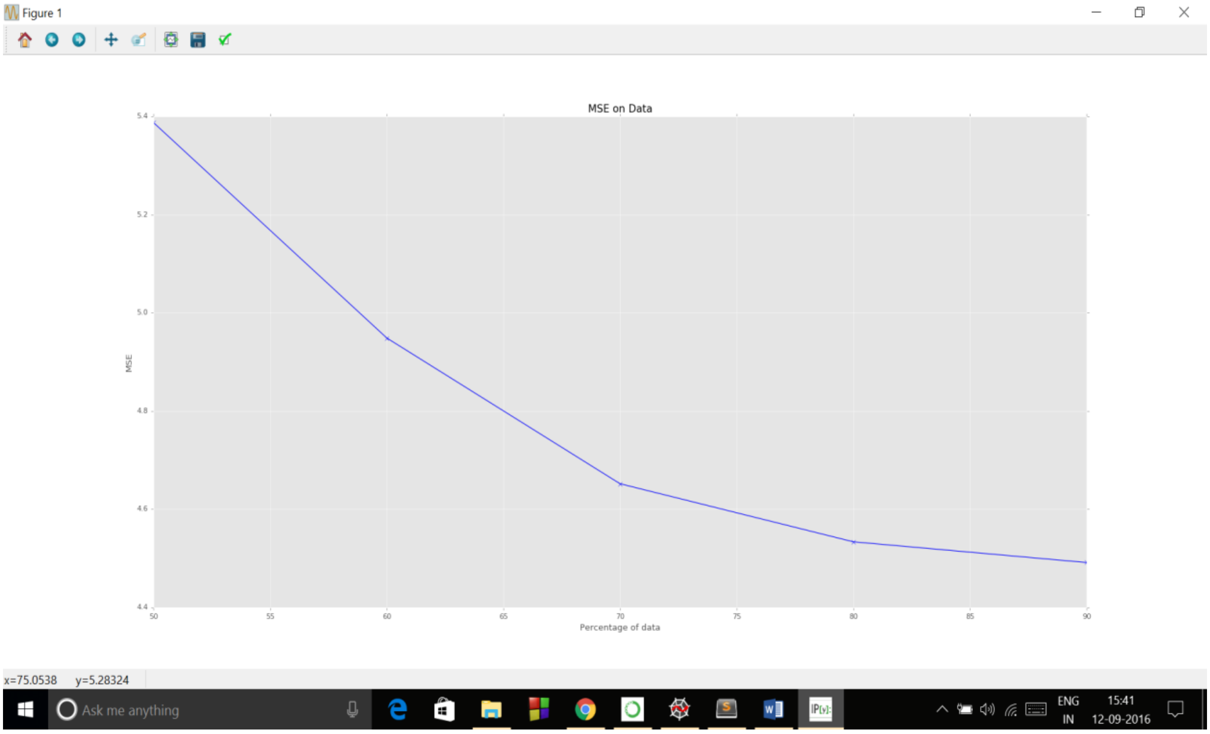
So I have chosen delta value = 0.1 which best fit the data.

Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:

```
[[ 5.39196127]
 [ 4.94951354]
 [ 4.65529283]
 [ 4.53530362]
 [ 4.49195672]]
```

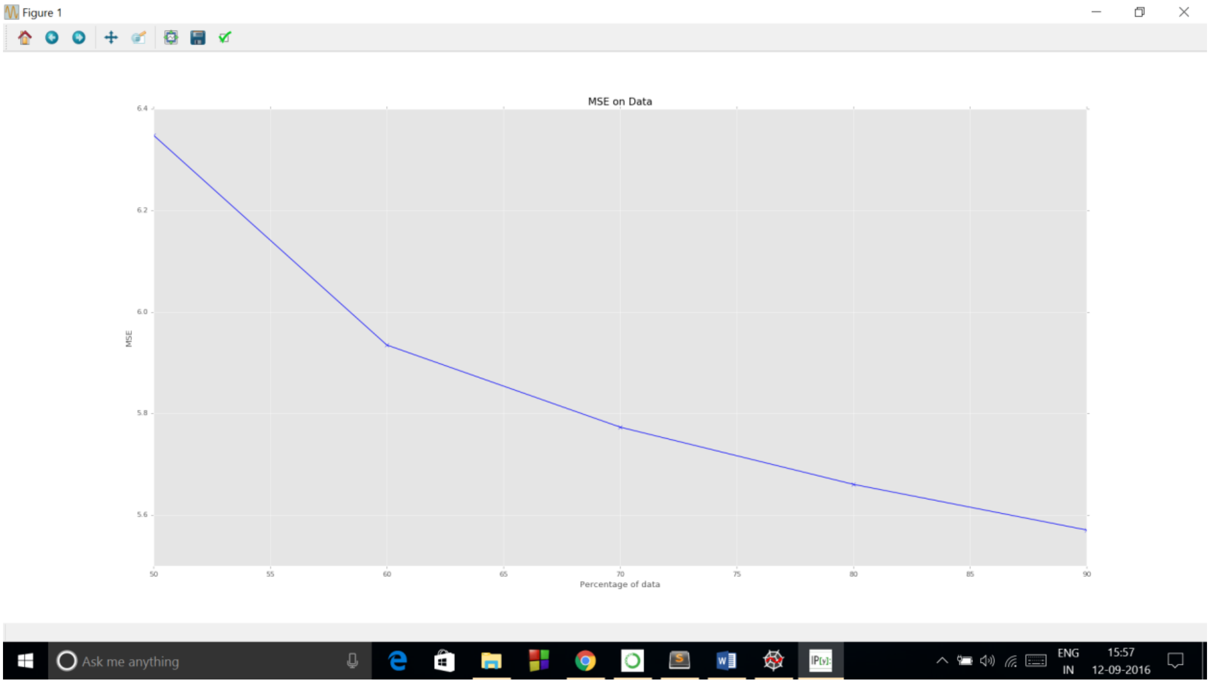
Final_parameters:

```
[[-3.75206566]
 [ 1.19657154]]
```

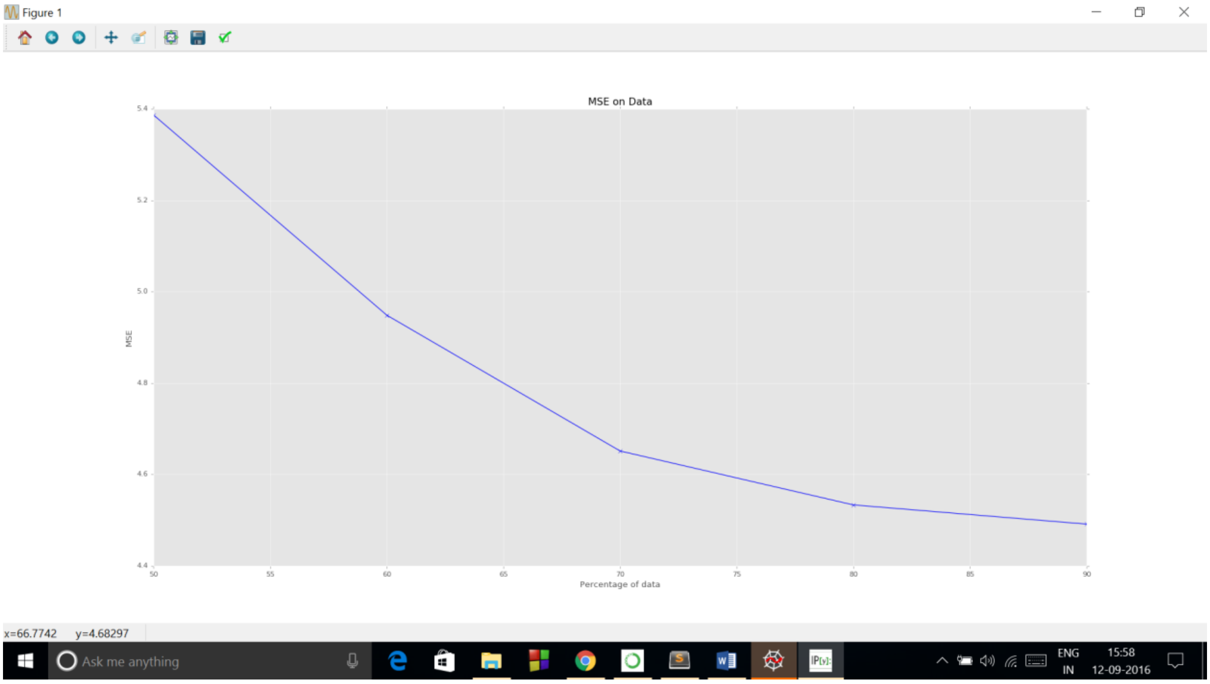


For higher delta we see there is underfit and cost tends to increase. For smaller delta we see there is overfit. So best fit is when delta = 0.1.

Higher delta:



Lower delta:



sph.txt

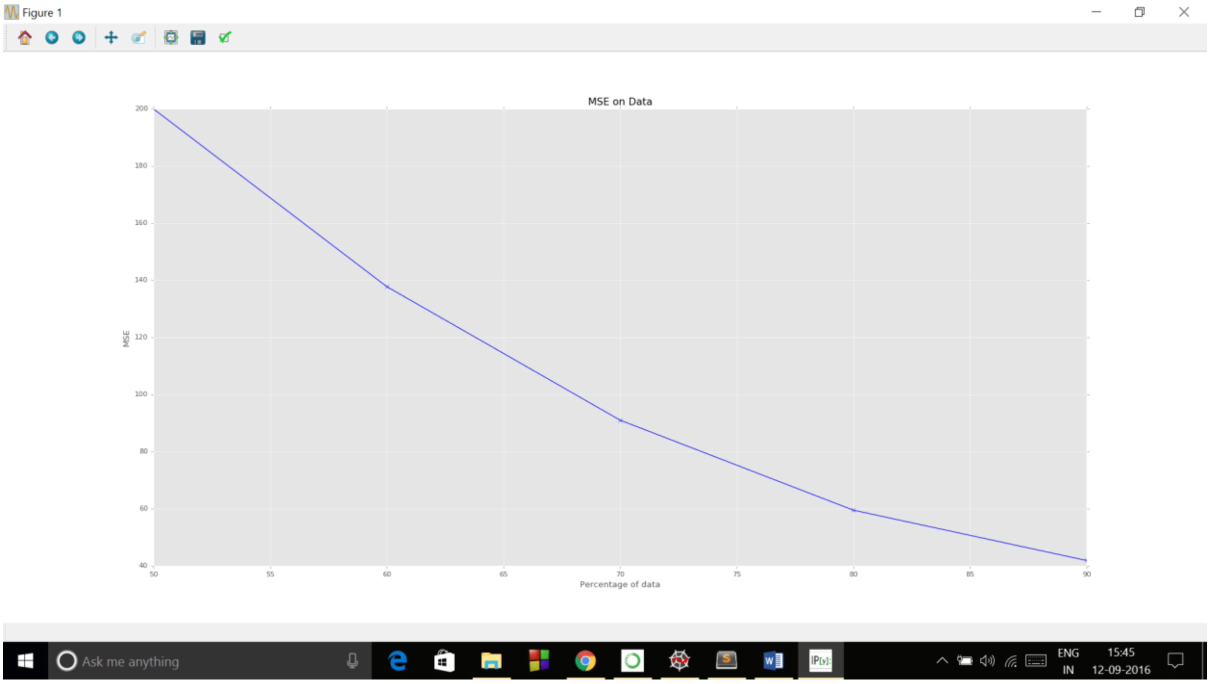
So I have chosen delta value = 0.1 which best fit the data.

Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:

```
[[ 199.97210943]
 [ 137.76523603]
 [  91.02075088]
 [  59.50416632]
 [  41.88180253]]
```

Final_parameters:

```
[[-12.28450012]
 [  0.95156913]]
```



For higher delta we see there is underfit and cost tends to increase. For smaller delta we see there is overfit. So best fit is when delta = 0.1.

Higher delta:

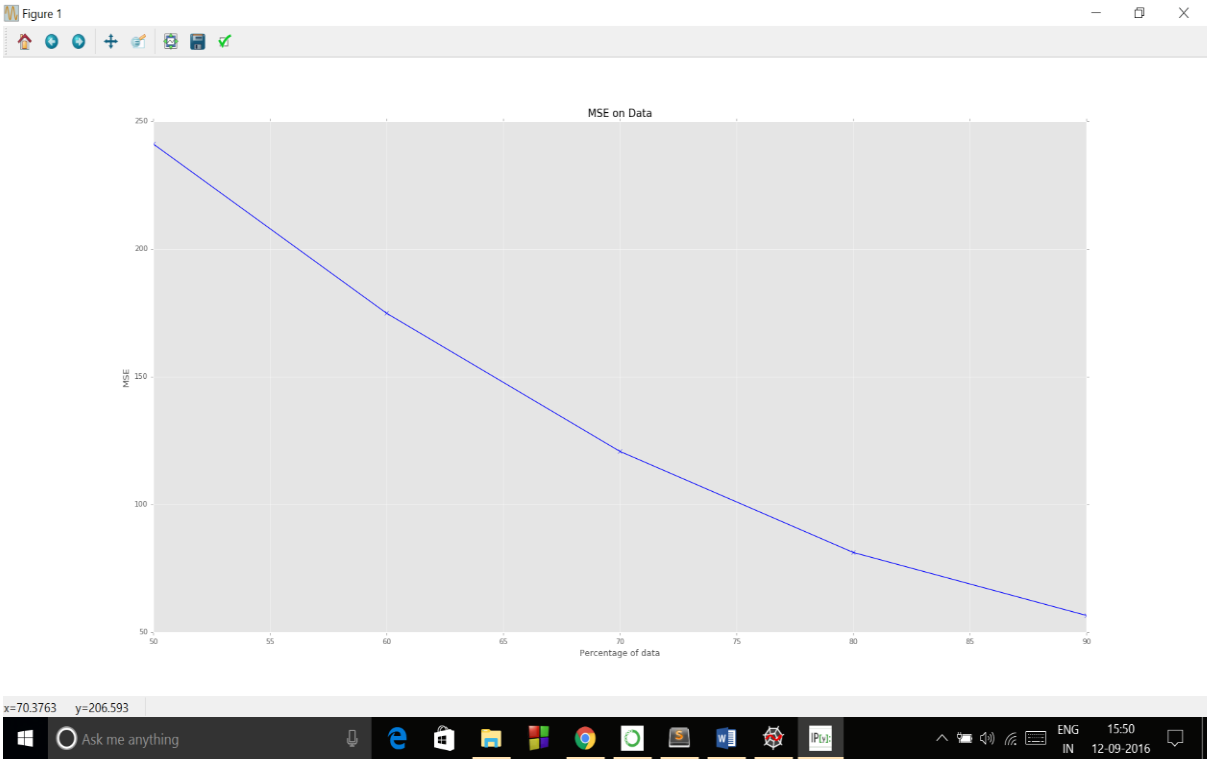


Figure 1
MSE on Data
250
200
150
100
50
MSE
50
55
60
65
70
75
80
85
90
Percentage of data
x=70.3763 y=206.593
Ask me anything
ENG
IN
15:50
12-09-2016

Lower delta:

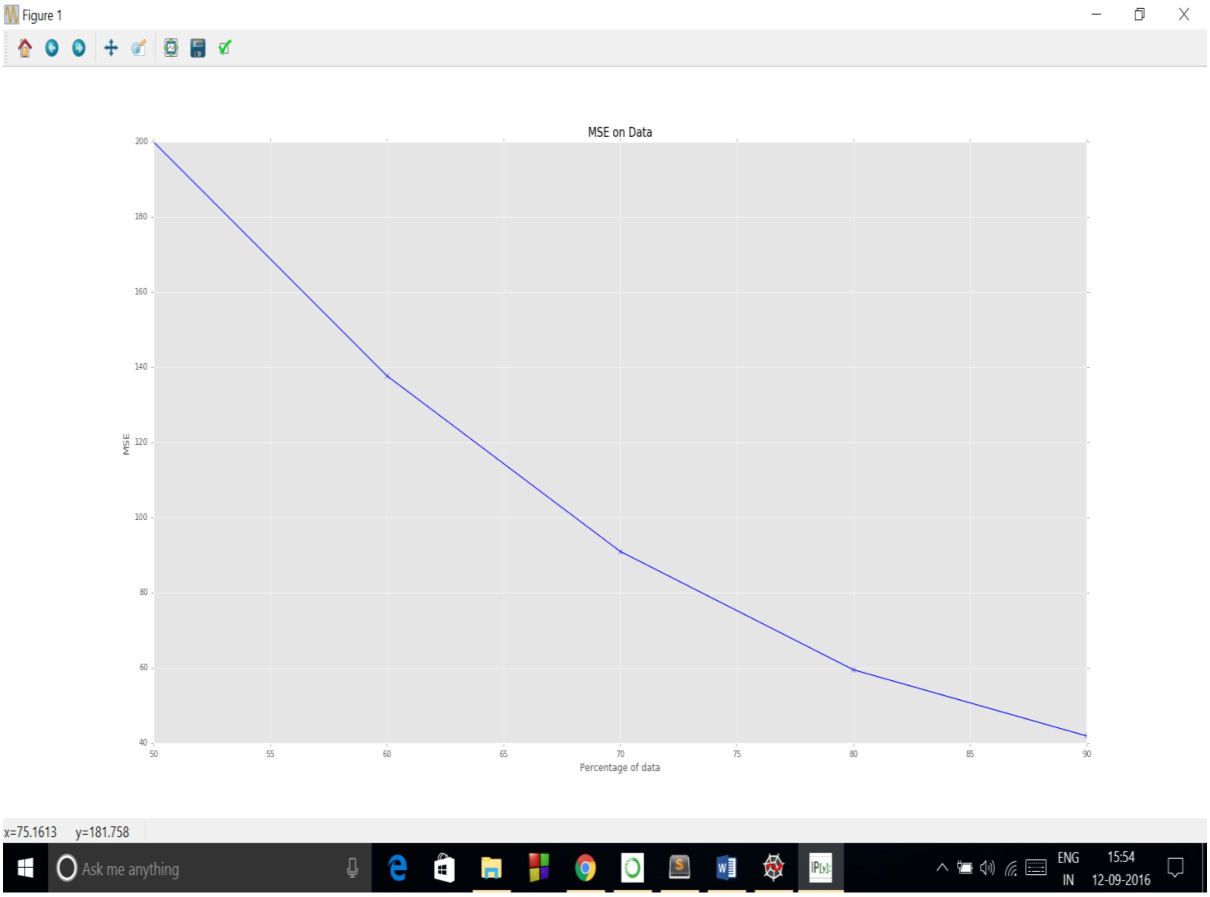


Figure 1
MSE on Data
200
180
160
140
120
100
80
60
40
MSE
50
55
60
65
70
75
80
85
90
Percentage of data
x=75.1613 y=181.758
Ask me anything
ENG
IN
15:54
12-09-2016

E).

lin.txt:

Here Gaussian kernel best fits the data.

Mean of test error: 5.1995724247

Std of test error: 7.04496308348

Final_parameters:

```
[[-3.83246815]
 [ 1.20054998]]
```

In polynomial kernel:

Mean of test error: 5.39951571657

Std of test error: 7.1130080247

Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:

```
[[ 5.40603452]
 [ 4.94200299]
 [ 4.65225806]
 [ 4.53839634]
 [ 4.48682286]]
```

Final_parameters:

```
[[ -6.14572078e+00]
 [  1.90253007e+00]
 [ -6.29580484e-02]
 [  1.66455547e-03]]
```

In gaussian kernel:

Mean of test error: 5.1995724247

Std of test error: 7.04496308348

Final_parameters:

```
[[-3.83246815]
 [ 1.20054998]]
Gaussian kernel:
Mean of test error:  5.39951572216
Std of test error:  7.11300801726
Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:
[[ 5.40603453]
 [ 4.94200298]
 [ 4.65225806]
 [ 4.53839634]
 [ 4.48682286]]
Final_parameters:
[[ -6.14572186e+00]
 [  4.25418732e+04]
 [ -4.45180885e+07]
 [  4.55857861e+10]]


sph.txt:
Linear kernel:
Mean of test error:  59.3670985994
Std of test error:  60.0409736531

Final_parameters:
[[-12.28637391]
 [  0.95160029]]
```

```
Polynomial kernel:
Mean of test error:  9.9456586336
Std of test error:  2.25962634043
Final_parameters:
[[  9.53078664e-01]
 [  1.56195472e-01]
 [  6.69055133e-03]
 [  2.50877482e-05]]
```

```
Gaussian kernel:
Mean of test error:  9.51229639628
Std of test error:  2.85368891888
Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:
[[ 1191.53659013]
 [ 1191.53659013]
 [ 1191.53659013]
 [ 1191.53659013]
 [ 1191.53659013]]
Final_parameters:
[[-11.56634391]
 [ 33.73033725]
 [-42.4021767 ]
 [ 20.44410388]]
```

Gaussian best fits the data.

seeds_dataset:

Linear kernel:

```
Mean of test error:  0.105967843521

Std of test error:  0.0936895203577

Final_parameters:

[[ 58.15571646]
 [  1.5159037 ]
 [ -3.24367246]
 [-37.0137699 ]
 [ -2.22050422]
 [  0.53613228]
 [  0.09044132]
 [  2.06137804]]
```

Polynomial kernel:

```
Mean of test error:  0.136326537046

Std of test error:  0.134521301659


Final_parameters:

[[ -2.74009149e+03]
 [ -1.74283436e+01]
 [  1.08236571e+02]
 [  7.35407329e+03]
 [  1.12032342e+02]
 [  2.32108330e+01]
 [ -3.49986287e-02]
 [ -5.14571746e+01]
 [  1.05563069e+00]
 [ -7.07170994e+00]
 [ -8.41305474e+03]
 [ -2.02615733e+01]
 [ -7.19929263e+00]
 [  2.94550796e-02]
 [  1.03051807e+01]
 [ -2.08170449e-02]
 [  1.53162730e-01]
 [  3.20492691e+03]
 [  1.19676985e+00]
 [  7.35165526e-01]
 [ -2.15122002e-03]
 [ -6.58783659e-01]]
```

```
Gaussian kernel:
Mean of test error:  0.783990542683
Std of test error:  1.20034810607
Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:
[[ 0.50007586]
 [ 3.64380383]
 [ 7.07654658]
 [ 0.39981445]
 [ 2.37352994]]
Final_parameters:
[[  5.58072774e+02]
 [  6.22697250e+03]
 [ -5.15041472e+03]
 [ -3.65435969e+03]
 [  7.76485260e+03]
 [  5.46122472e+02]
 [ -9.03337136e+03]
 [ -1.23236996e+05]
 [  7.65343937e+05]
 [  5.20006827e+07]
 [  7.92194599e+05]
 [  1.64125827e+05]
 [ -2.47475775e+02]
 [ -3.63857666e+05]
 [  7.46445905e+07]
 [ -5.00037572e+08]
 [ -5.94887370e+11]
 [ -1.43272299e+09]
 [ -5.09062877e+08]
 [  2.08316803e+06]
 [  7.28680722e+08]
 [ -1.80280545e+10]
 [  1.32641961e+11]
 [  2.77552244e+15]
 [  1.03644080e+12]
 [  6.36674049e+11]
 [ -1.86300256e+09]
 [ -5.70523958e+11]]
```

Gaussian best fits the data.

```
iris_data:
Linear kernel:
Mean of test error:  0.0293738811041
Std of test error:  0.0216304028656
Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:
[[ 0.06833199]
 [ 0.07141878]
 [ 0.04195914]
 [ 0.02690352]
 [ 0.02357895]]
Final_parameters:
[[ 1.84623117]
 [ 0.10719925]
 [ 0.0405839 ]
 [-0.24900758]
 [-0.53588614]]

Polynomial kernel:
Mean of test error:  0.0259139955207
Std of test error:  0.0210187146341
Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:
[[ 0.59421357]
 [ 0.45453488]
 [ 0.0361531 ]
 [ 0.02104076]
 [ 0.01763312]]
Final_parameters:
[[ 4.41174419]
 [-2.08477766]
 [ 0.49892123]
 [ 0.37373   ]
 [ 0.45208169]
 [ 0.32005398]
 [ 0.07679573]
 [-0.13660148]
 [-0.88507123]
 [-0.01453659]
 [-0.02727833]
 [ 0.007773  ]
 [ 0.21404562]]
```

```
Gaussian kernel:
Mean of test error:  0.28711981094
Std of test error:  0.221978202994
Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:
[[ 0.26105358]
 [ 2.11999115]
 [ 0.99831511]
 [ 4.20717824]
 [ 1.30839136]]
```

```
Final_parameters:
[[  3.49521822e+01]
 [  2.62084823e+01]
 [ -9.36280719e+01]
 [  4.40188608e+01]
 [ -1.01883828e+04]
 [ -3.00628285e+04]
 [ -1.39544375e+04]
 [ -1.86382009e+04]
 [  3.12879530e+07]
 [  7.80375423e+07]
 [  1.96034997e+07]
 [  1.94934873e+08]
 [ -3.36969610e+10]
 [ -7.79288458e+10]
 [  3.68764271e+08]
 [ -5.36742256e+11]]
```

Gaussian kernel best fits the data.

```
AirQualityUCI.txt
Linear kernel:
Mean of test error:  3.49144588364
Std of test error:  1.69059979269
Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:
[[ 5.25162289]
 [ 5.45141356]
 [ 3.80324406]
 [ 3.18256931]
 [ 2.6113028 ]]
Final_parameters:
[[  8.77141483e+00]
 [ -1.49533794e-01]
 [ -3.32211416e-04]
 [ -2.60470687e-03]
 [ -3.64992750e-01]
 [  1.21093288e-02]
 [  2.19197695e-03]
 [  1.44242206e-03]
 [ -2.59719811e-04]
 [  4.64696646e-03]
 [ -2.73366327e-03]
 [ -3.42181780e-01]
 [  1.40081888e+01]]
```

Polynomial kernel:

```
Mean of test error:  0.134104566275
Std of test error:  0.0749423458273
Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:
[[ 0.16947999]
 [ 0.16749253]
 [ 0.10308494]
 [ 0.09859191]
 [ 0.09095416]]
Final_parameters:
[[  1.26817448e+01]
 [ -1.29600690e-01]
 [  1.17822999e-02]
 [  1.03917328e-03]
 [  1.73619593e+00]
 [  3.47491062e-02]
 [ -2.49897205e-03]
 [ -6.94256775e-04]
 [  1.26387517e-02]
 [  3.54049109e-03]
 [ -1.03562716e-03]
 [ -1.34301069e+00]
 [  5.41226703e+01]
 [  3.57253268e-02]
 [ -8.22732747e-06]
 [ -2.37608422e-06]
 [ -5.32577981e-02]
 [ -8.06825338e-05]
 [  2.51132436e-06]
 [ -2.18352196e-08]
 [ -6.22135937e-05]
 [ -2.75120260e-06]
 [  7.32657181e-07]
 [  1.60755155e-02]
 [ -2.68262197e+01]
 [ -2.50745417e-03]
 [  1.82874532e-09]
 [  1.36408355e-09]
 [  1.29388862e-04]
 [  3.64173678e-08]
 [ -8.56695195e-10]
 [  1.00934075e-10]
 [  1.18861917e-07]
 [  5.76153087e-10]
 [ -1.40590917e-10]
 [ -7.72689071e-05]
 [  5.44496777e+00]]
```

```
Gaussian kernel:
Mean of test error:  1494.27159262
Std of test error:  3278.22616039
Cost after training from 50%, 60%, 70%, 80%, 90% of data respectively:
```

```
[[ 1422.35207022]
 [ 3694.72197059]
 [  857.90387968]
 [  244.80427764]
 [  511.50427454]]
Final_parameters:
[[  4.27067423e+05]
 [  2.54412939e+04]
 [ -3.17463621e+04]
 [  3.59795029e+06]
 [ -1.16895660e+05]
 [ -8.07879490e+03]
 [  2.62986238e+03]
 [ -4.10748809e+06]
 [  1.80125762e+04]
 [ -4.29467829e+03]
 [ -9.75134168e+05]
 [  1.17256407e+06]
 [ -8.30061876e+02]
 [  1.46987635e+02]
 [  5.59331808e+00]
 [  1.17477691e+04]
 [  1.31242327e+02]
 [ -2.26127121e+01]
 [  6.21320689e+00]
 [  9.22137683e+01]
 [  1.32412238e+02]
 [ -1.80362924e+01]
 [ -9.51712765e+03]
 [  3.80148366e+05]
 [  2.57682000e+06]
 [  1.65876855e+04]
 [ -2.25138760e+04]
 [ -8.64148562e+05]
 [ -8.62369468e+04]
 [ -5.39903819e+03]
 [  1.73303479e+03]
 [ -2.90824163e+06]
 [  1.14896058e+04]
 [ -2.84096807e+03]
 [  4.51964249e+05]
 [ -1.87188383e+09]
 [ -1.95883893e+09]
 [  7.17631232e+03]
 [ -1.28349859e+03]
 [  3.35486425e+07]
 [  1.19190893e+04]
 [ -2.15459078e+03]
 [  7.91645699e+02]
 [  2.03955453e+04]
 [  5.86504655e+03]
 [ -1.09965193e+03]
 [ -6.73351677e+07]
 [  4.63378844e+12]]
```

Gaussian kernel gives best result.

Reference:

For Gaussian kernel implementation I took help from:

http://www.csie.ntu.edu.tw/~cjlin/talks/kuleuven_svm.pdf