|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Academic Year: | 2022/2023 | Term: | Spring 2023 |  |
| Course Code: | ELC 4028 | Course Title: | Artificial Neural Networks and its Applications |

Cairo University

Faculty of Engineering

Electronics and Communications Engineering Department – 4th Year

Neural Networks Applications

- Assignment Report: Classical Machine Learning Methods -

*Submitted to: Dr. Mohsen Rashwan*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | BN | Sec | ID | رقم الجلوس |
| احمد محمود حسيني عطية | 22 | 1 | 9180178 | 34022 |
| علي ماهر عبدالسلام نبيه | 5 | 3 | 9190067 | 34117 |
| محمد احمد طه السيد | 30 | 3 | 9191043 | 34142 |
| محمد حسام عثمان يسن | 35 | 3 | 9191083 | 34147 |
| محمد عاطف ربيع | 43 | 3 | 9190924 | 34155 |

Table of Contents

[1. Part 1 1](#_Toc131703253)

[1.1. The numbers of insured persons with an insurance company 1](#_Toc131703254)

[1.2. Throwing a Rubber Ball 11](#_Toc131703255)

[1.2.1. Results 11](#_Toc131703256)

[1.2.2. Comments 13](#_Toc131703257)

[1.2.3. Prediction 14](#_Toc131703258)

[1.3. Heating Oil Consumption Model 15](#_Toc131703259)

[1.3.1. Linear Model Results 15](#_Toc131703260)

[1.3.2. Quadratic Model Results 16](#_Toc131703261)

[1.3.2. Comments 16](#_Toc131703262)

[1.3.3. Removing Unreasonable Data 17](#_Toc131703263)

[2. Part 2 18](#_Toc131703264)

[2.1 Results 18](#_Toc131703265)

[2.2 Notes 18](#_Toc131703266)

[2.3 Conclusions 19](#_Toc131703267)

[3. Appendix A: problem 1 codes 19](#_Toc131703268)

[4. Appendix B: problem 2 codes 21](#_Toc131703269)

[5. Appendix C: problem 3 codes 24](#_Toc131703270)

[6. Appendix D: part 2 code 26](#_Toc131703271)

# 1. Part 1

## 1.1. The numbers of insured persons with an insurance company

The numbers of insured persons y with an insurance company for the years 1987 to

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Year | 1987 | 1988 | 1989 | 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | 1996 |
| y | 14000 | 13000 | 12000 | 11000 | 1050 | 10000 | 9500 | 9000 | 8700 | 8000 |

1996 are shown in the table.

**Note :** The possible outlier here is at year 1991 with y=1050 as this value is far away from the other points so it may be an error or high noise.

Make a scatterplot of the data, letting x represent the number of years since 1987.

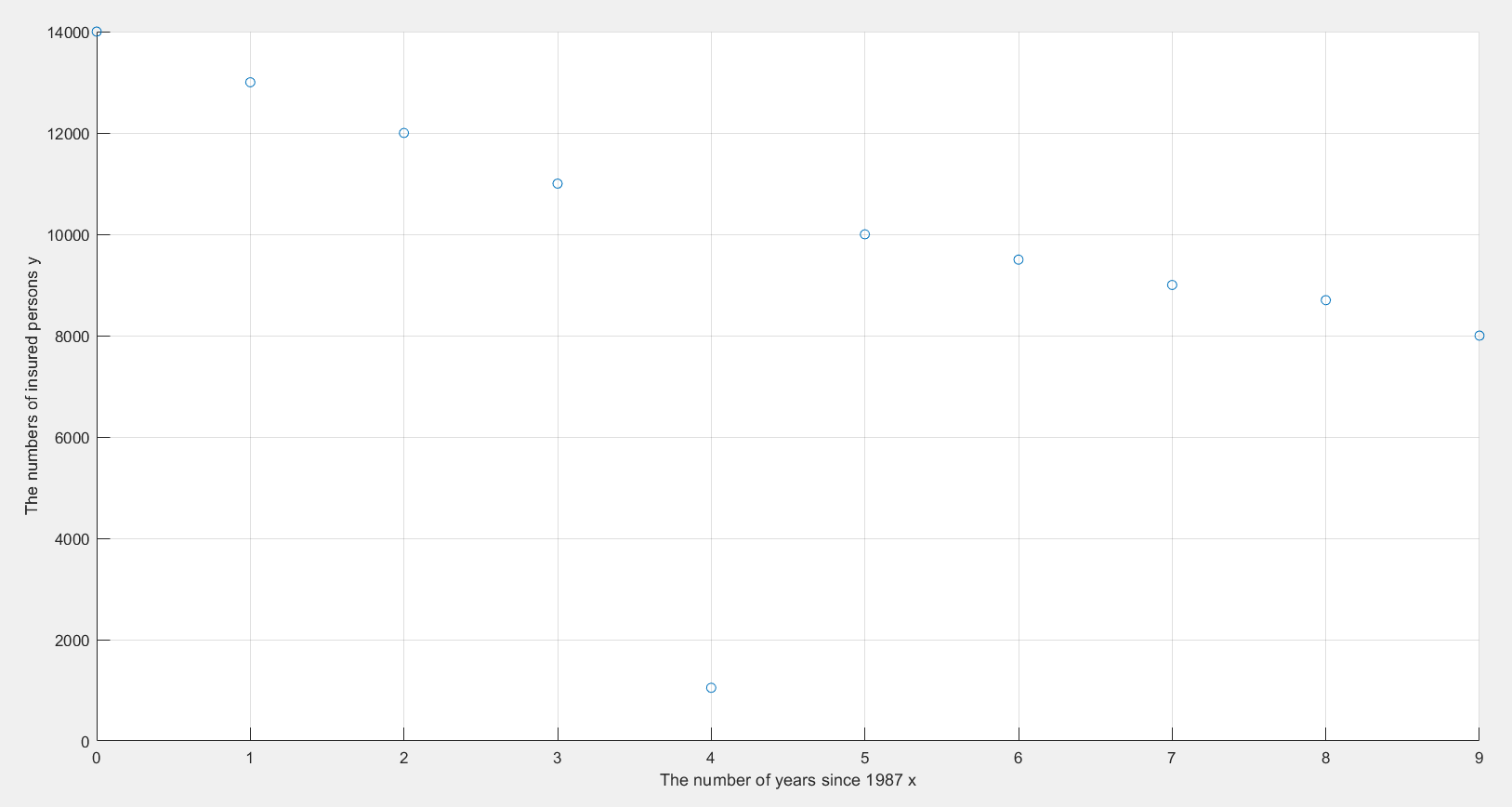
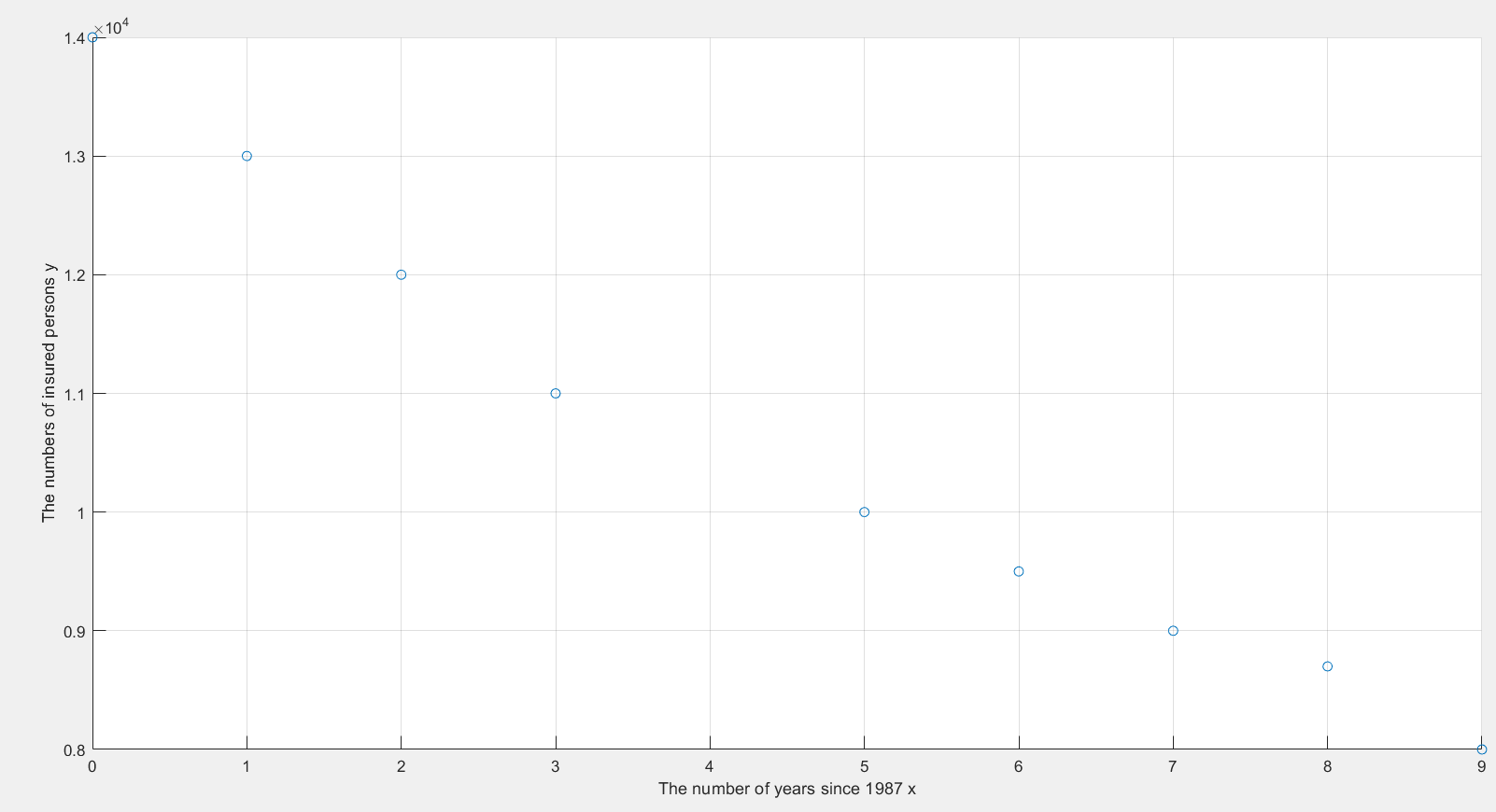
**Ans:**

Fig.1 scatterplot of data (with the outlier) against number of years since 1987

Fig.2 scatterplot of data (without the outlier) against number of years since 1987

a) Fit linear, quadratic, cubic, by comparing the values of R2 . Determine the function that best fits the data. (Hint: take care of note 4 above)

b) In all your answers in each model, write down the equations of your solutions

(after calculating its parameters).

c) Graph the function of best fit with the scatterplot of the data.

d) With the best function found in part (b), predict the average number of insured

persons in 1997.

**Ans :**

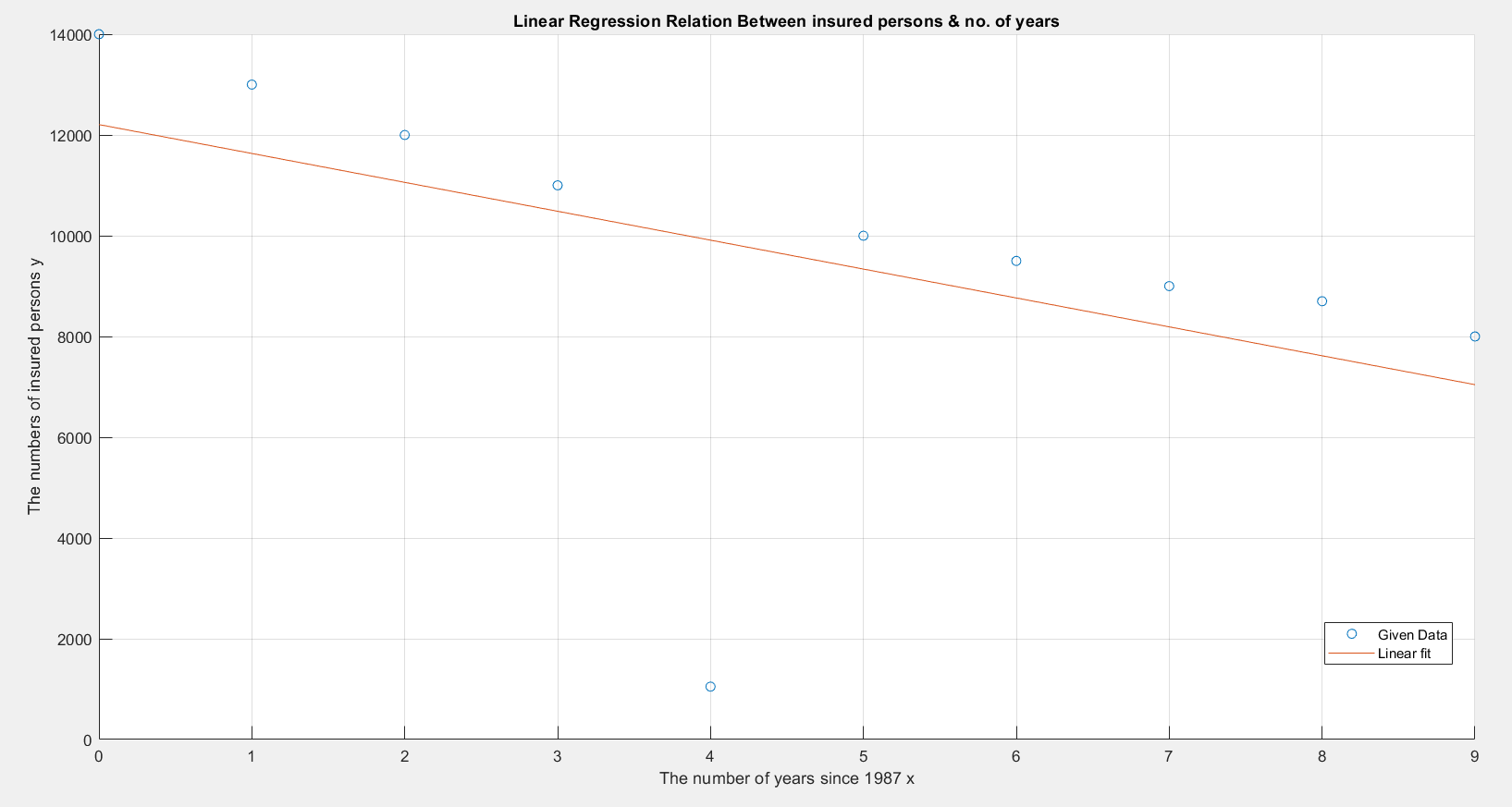
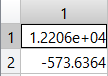
- Linear Regression : equation : y = a+bx

Fig.3 Linear Regression Relation Between insured persons & number of years (with the outlier)



a =>

b =>

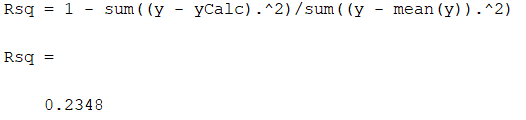
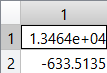
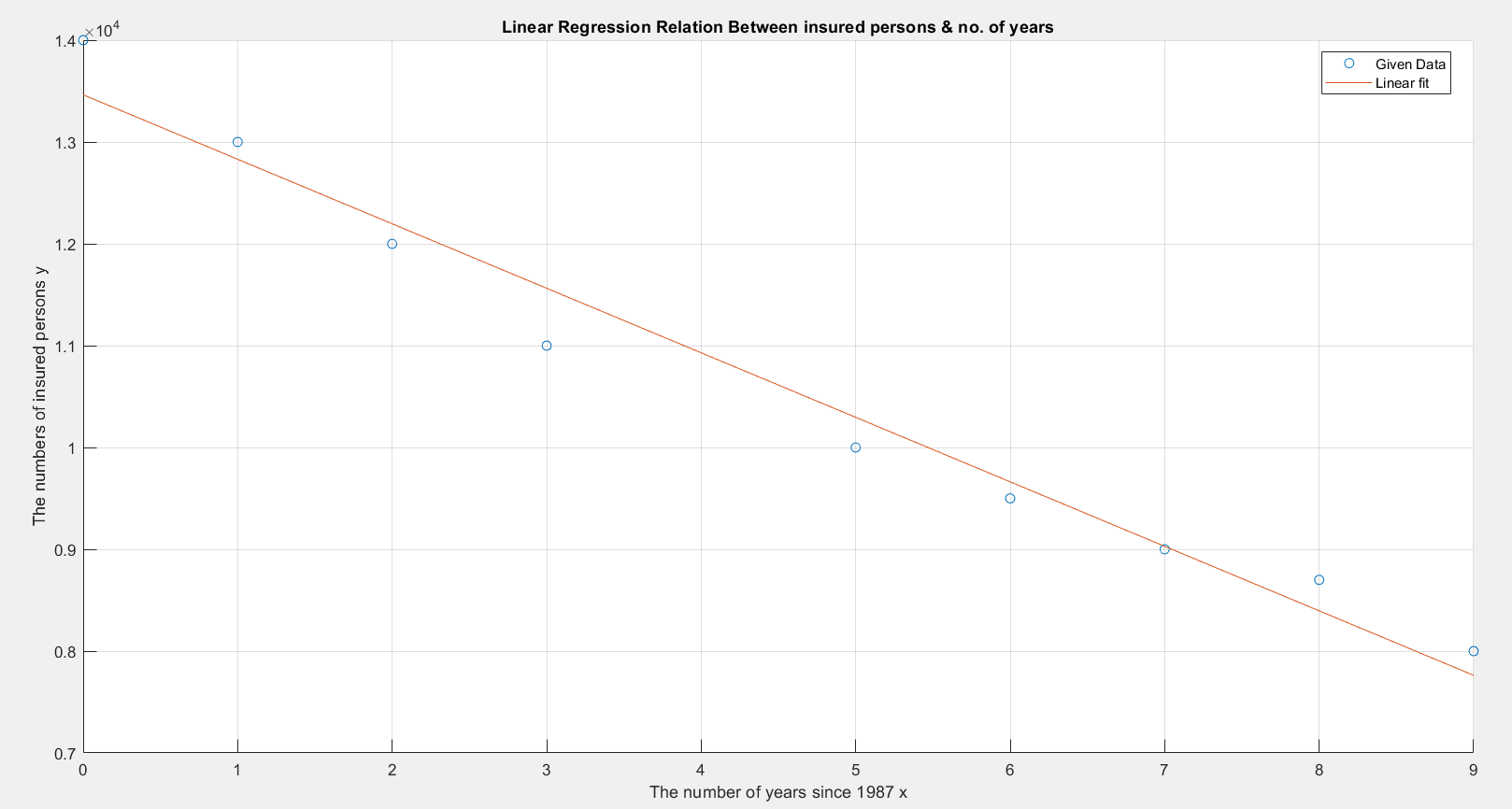
Fig.4 Linear Regression Parameters (with the outlier)

Fig.5 Linear Regression R2 (with the outlier)

Fig.6 Linear Regression Relation Between insured persons & number of years (with no outlier)

a =>

b =>

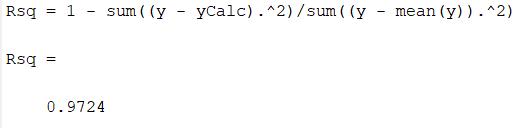
Fig.7 Linear Regression Parameters (with no outlier)

Fig.8 Linear Regression R2 (with no outlier)

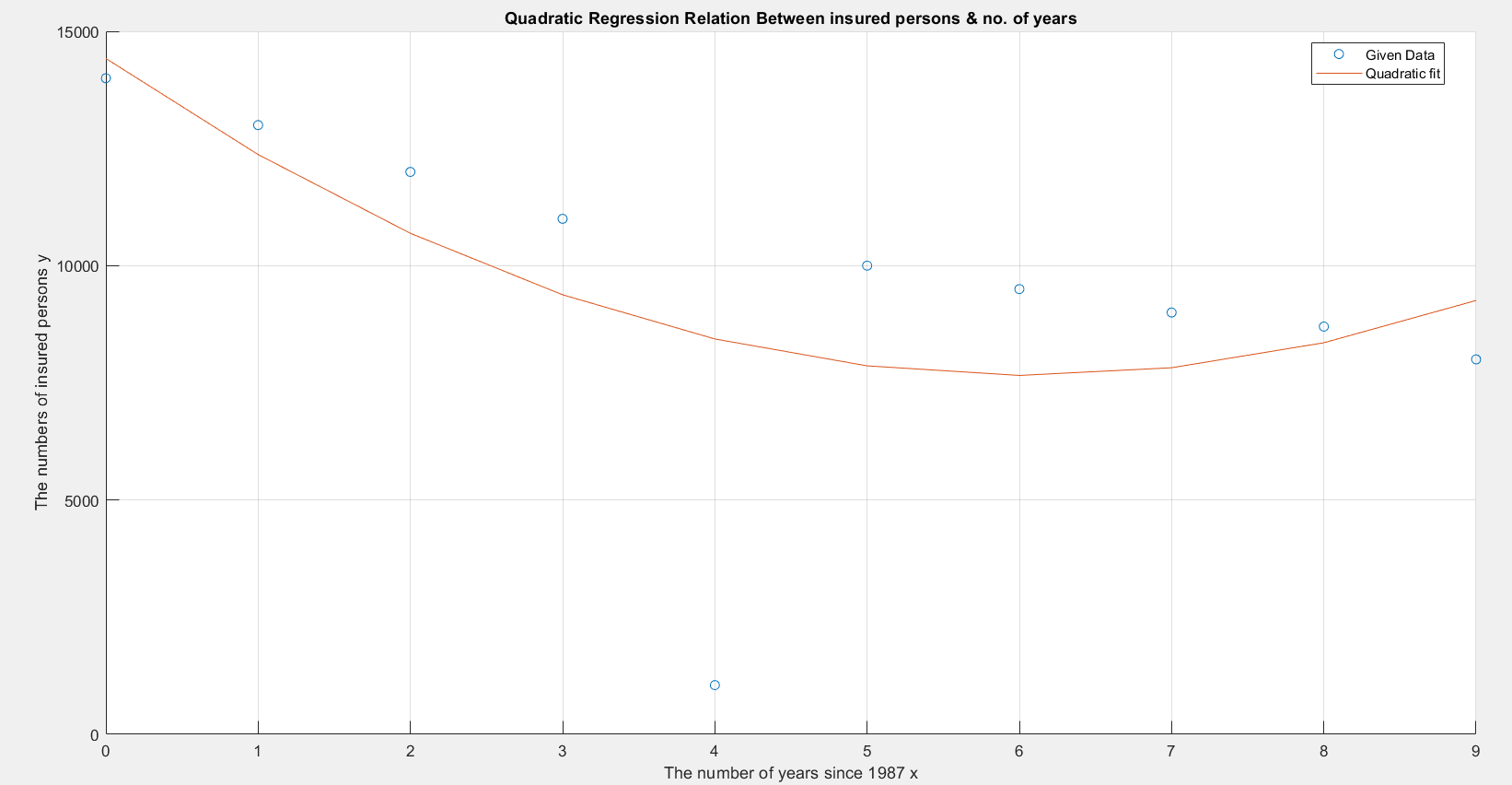
-Quadratic Regression : equation : y = c+bx+ax2

Fig.9 Quadratic Regression Relation Between insured persons & number of years (with the outlier)

Fig.10 Quadratic Regression Parameters (with the outlier)

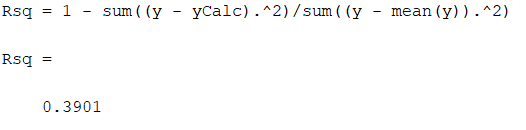
a = 184.4697 , b = -2233.9 , c = 14420

Fig.11 Quadratic Regression R2 (with the outlier)

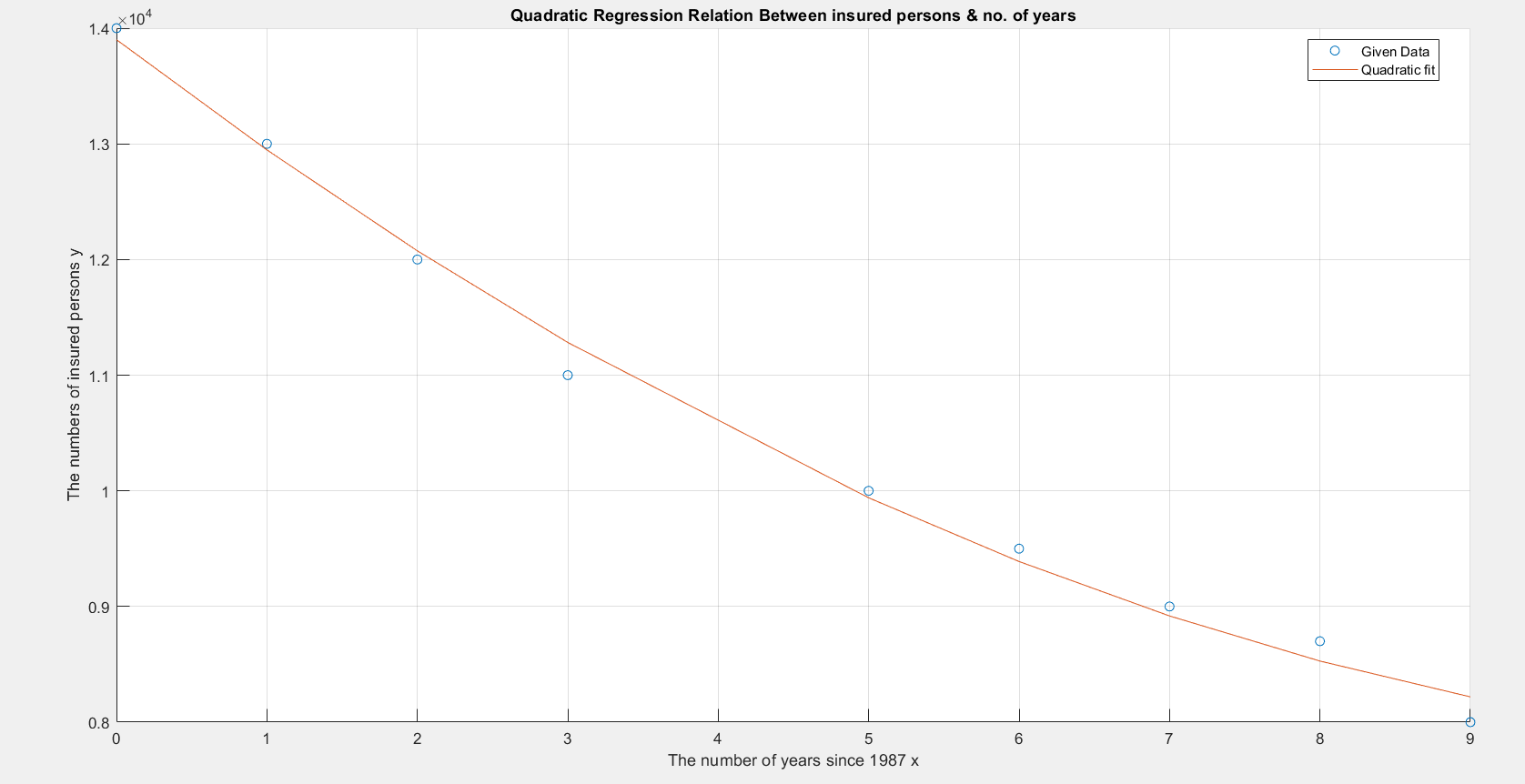
Fig.12 Quadratic Regression Relation Between insured persons & number of years (with no outlier)

Fig.13 Quadratic Regression Parameters (with no outlier)

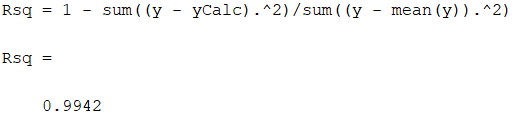
a = 40.2107 , b = -993.2363 , c = 13901

Fig.14 Quadratic Regression R2 (with no outlier)

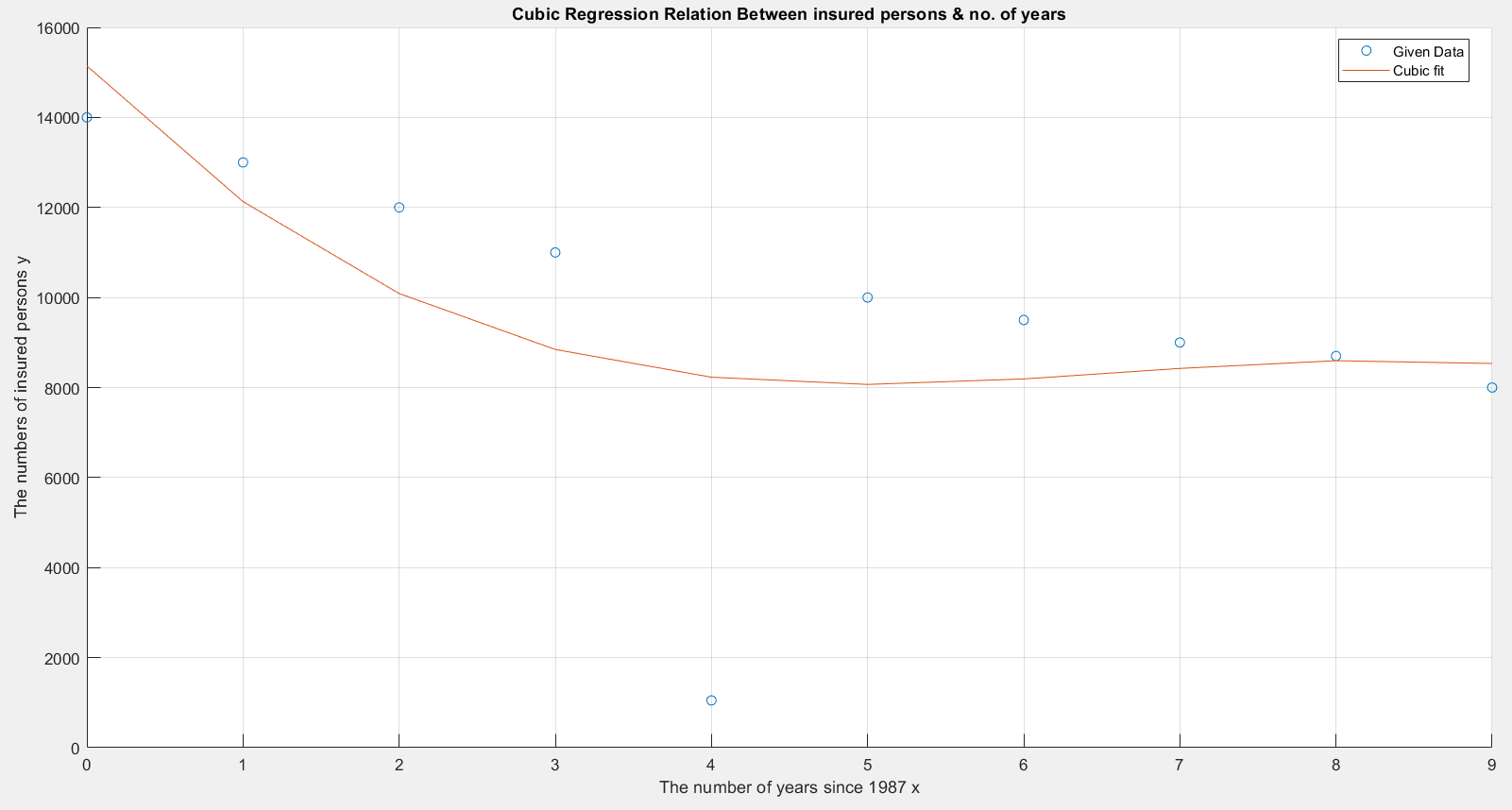
-Cubic Regression : equation : y = d+cx+bx2+ax3

Fig.15 Cubic Regression Relation Between insured persons & number of years (with the outlier)

Fig.16 Cubic Regression Parameters (with the outlier)

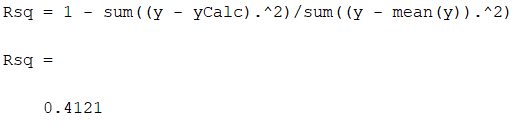
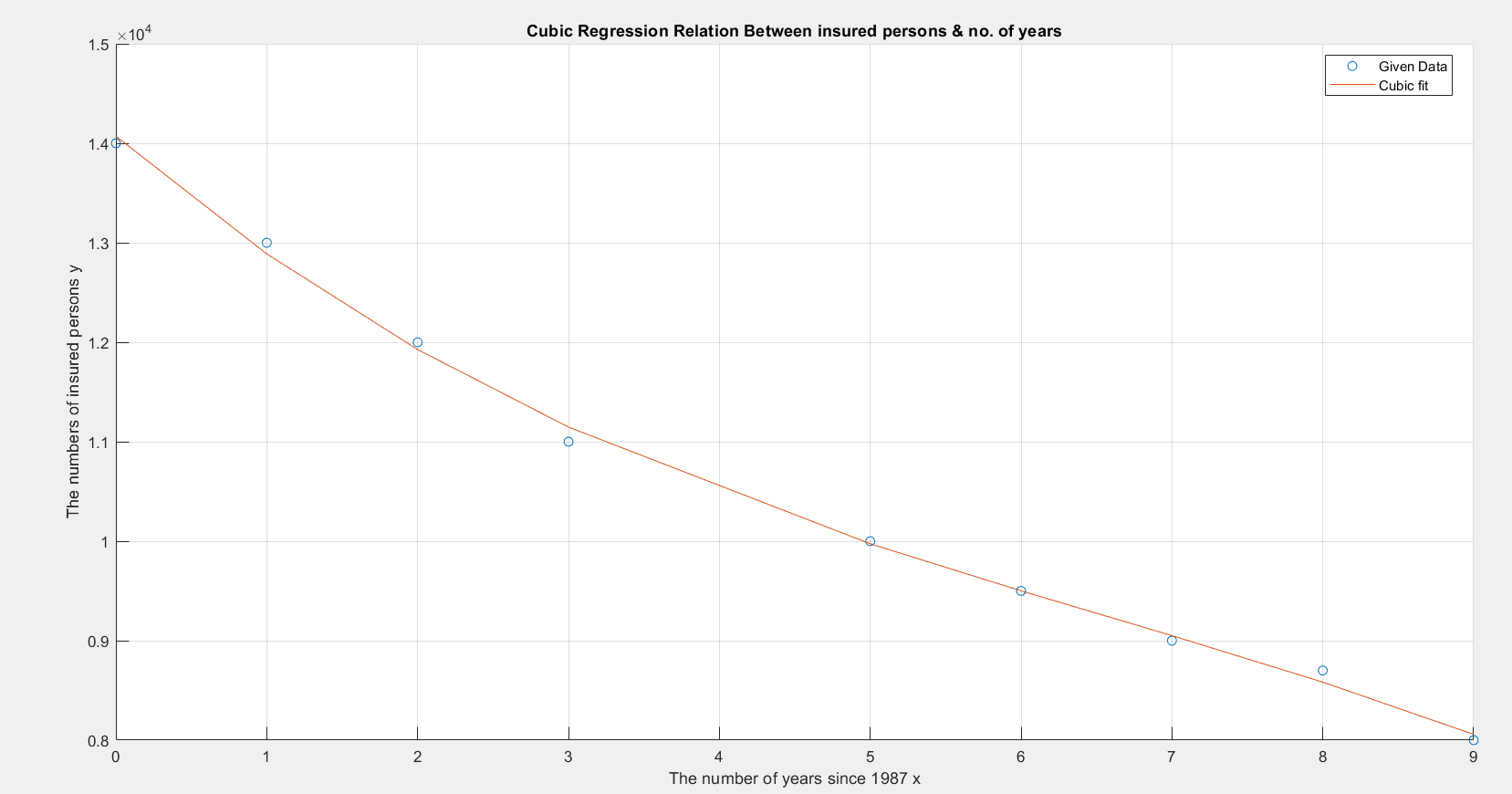
a = -28.6908 , b = 571.7949 , c = -3556.5 , d = 15143

Fig.17 Cubic Regression R2 (with the outlier)

Fig.18 Cubic Regression Relation Between insured persons & number of years (with no outlier)

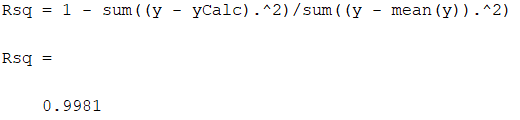
Fig.19 Cubic Regression Parameters (with no outlier)

Fig.20 Cubic Regression R2 (with no outlier)

a) The function that best fits the data is the Linear Regression model with no outlier as it has very high R2 with lower number of parameters, although Quadratic and Cubic Regressions have higher R2 than Linear Regression but the priority is to the Regression model with high enough R2 and lower parameters and as the difference between the values of R2 of all of them can be neglected.

**Equation :-** y = 13464 – 633.5135 \* x

b) - Linear Regression equation :

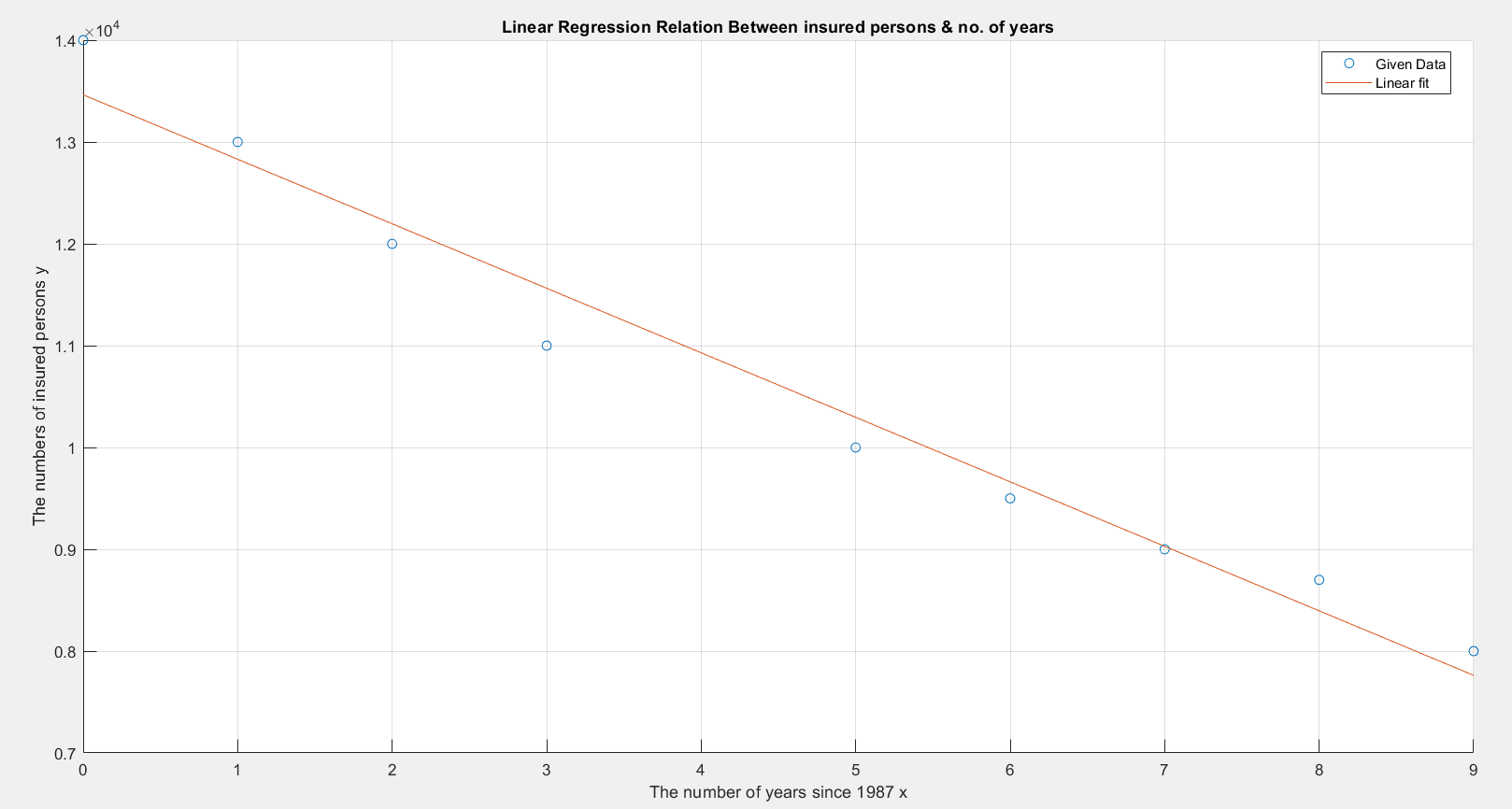
* With outlier : y = 12206 - 573.6364 \* x
* With no outlier : y = 13464 – 633.5135 \* x

- Quadratic Regression equation :

* With outlier : y = 14420 – 2233.9 \* x + 184.4697 \* x2
* With no outlier : y = 13901 – 993.2363 \* x + 40.2107 \* x2

- Cubic Regression equation :

* With outlier : y = 15143 – 3556.5 \* x + 571.7949 \* x2 – 28.6908 \* x3
* With no outlier : y = 14071 – 1307.4 \* x + 130.7944 \* x2 – 6.6407 \* x3

c) Graph of the function of best fit with the scatterplot of the data :

d) With the best function found in part (b), the predicted average number of insured persons in 1997 (after 10 years from 1987) = 13464 – 633.5135 \* (10) ≈ 7129 persons

**Note:**

1. Solve all the problems using MATLAB or any other computer language.

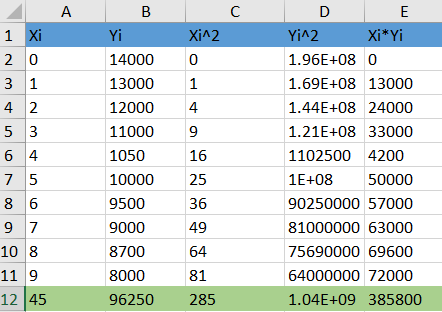
2. After calculating the best model given your data, review the given data and you may remove any possible outliers (a value with a possible error or high noise), then recalculate the model for cleaned

data. Compare the two models. This exists only in problems 1 and 5.

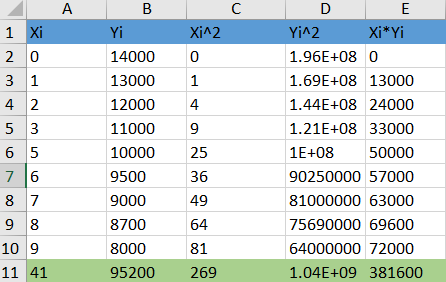
**Ans:**

As we see the model without the possible outlier has much better R2 than the model with the possible outlier. So, we should remove the possible outlier.

3. Try by hand the linear and the logistic problems as well. You may use an excel sheet for this calculation.

**Ans:** -With the outlier : equation : y = a + bx

= -573.636 , 9625- (-573.636)\*(4.5) = 12206.36

-With no outlier :

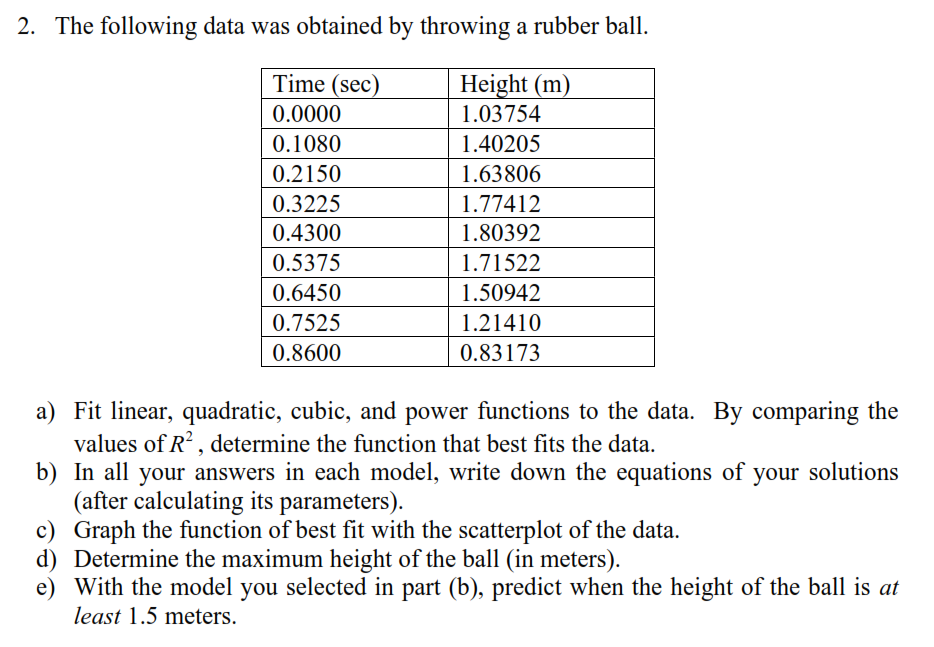
= -633.514 , 10577.78 – (-633.514)\*(4.56) = 13466.6

As we see, the hand analysis results are almost the same as the simulation results above.

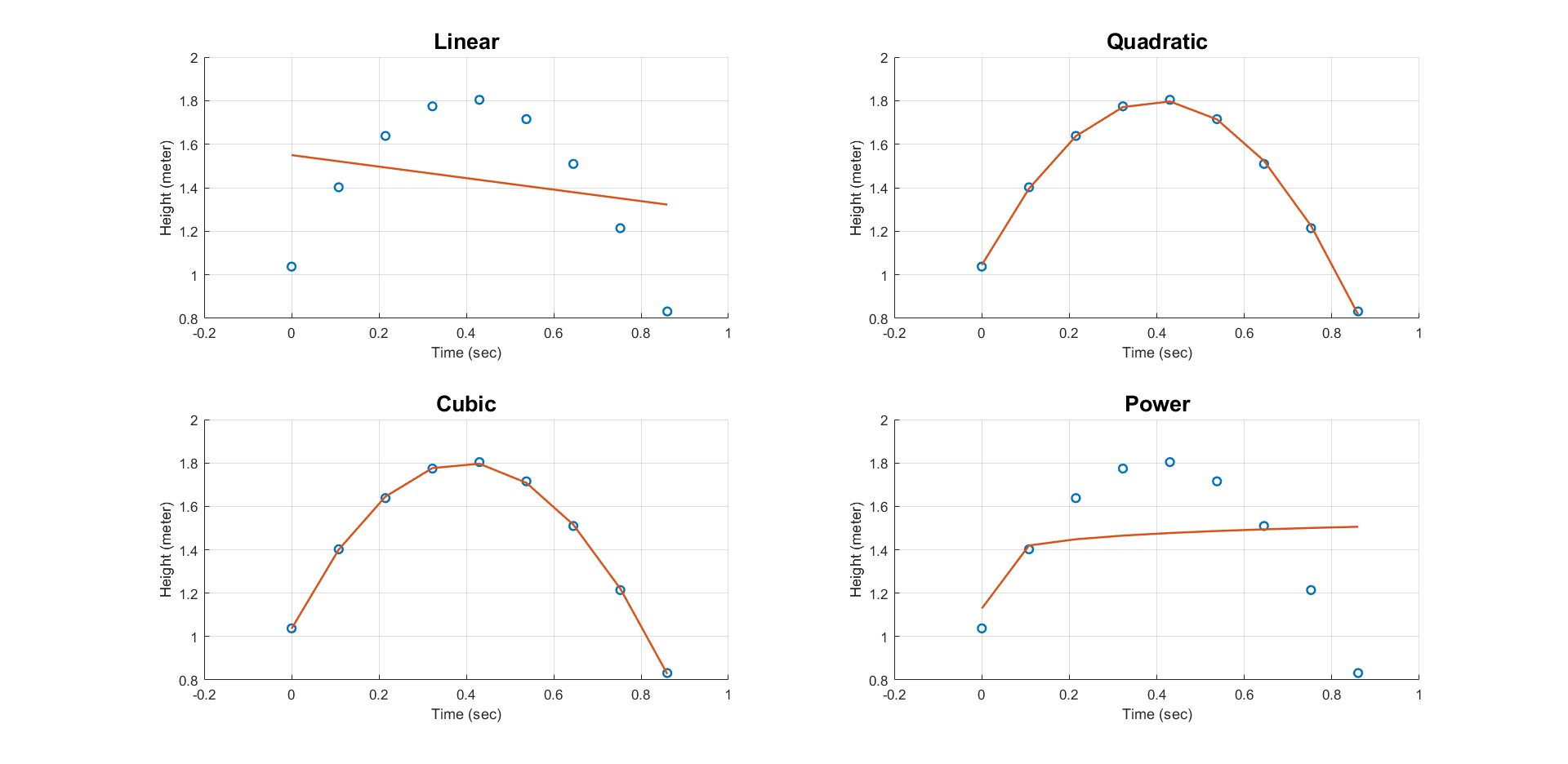
4. In all your answers to each question, write down the equations of your solutions (after calculating their parameters).

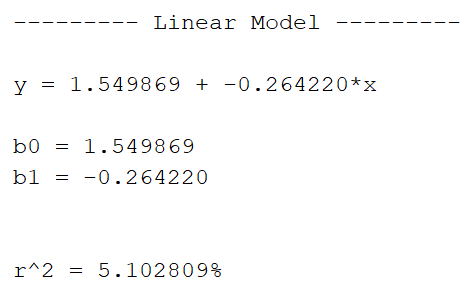
5. Hint: it is better to take the model of a lower number of parameters if the gain in R2 is not high enough.

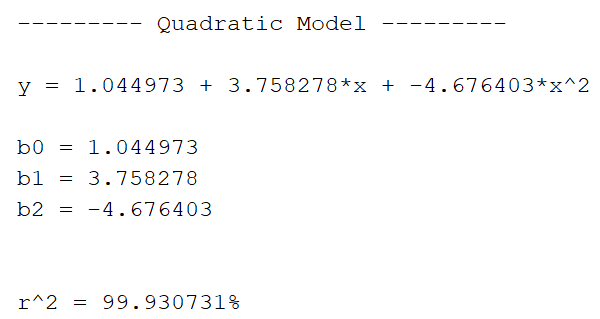
## 1.2. Throwing a Rubber Ball

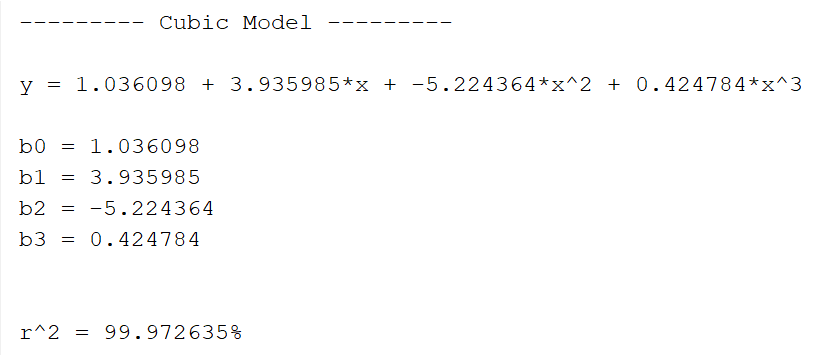


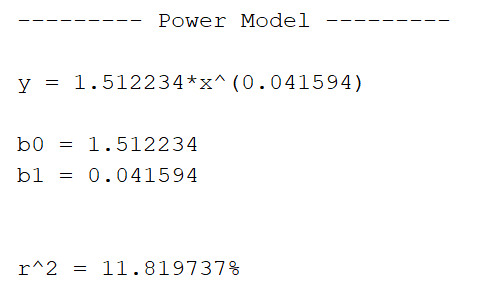
### 1.2.1. Results











### 1.2.2. Comments

|  |  |
| --- | --- |
| Linear: | Quadratic: |
| Cubic: | Power: |

Using the matrix method to obtain previous results as for example the linear parameters calculated by:

output matrix

independent variables matrix

parameters matrix

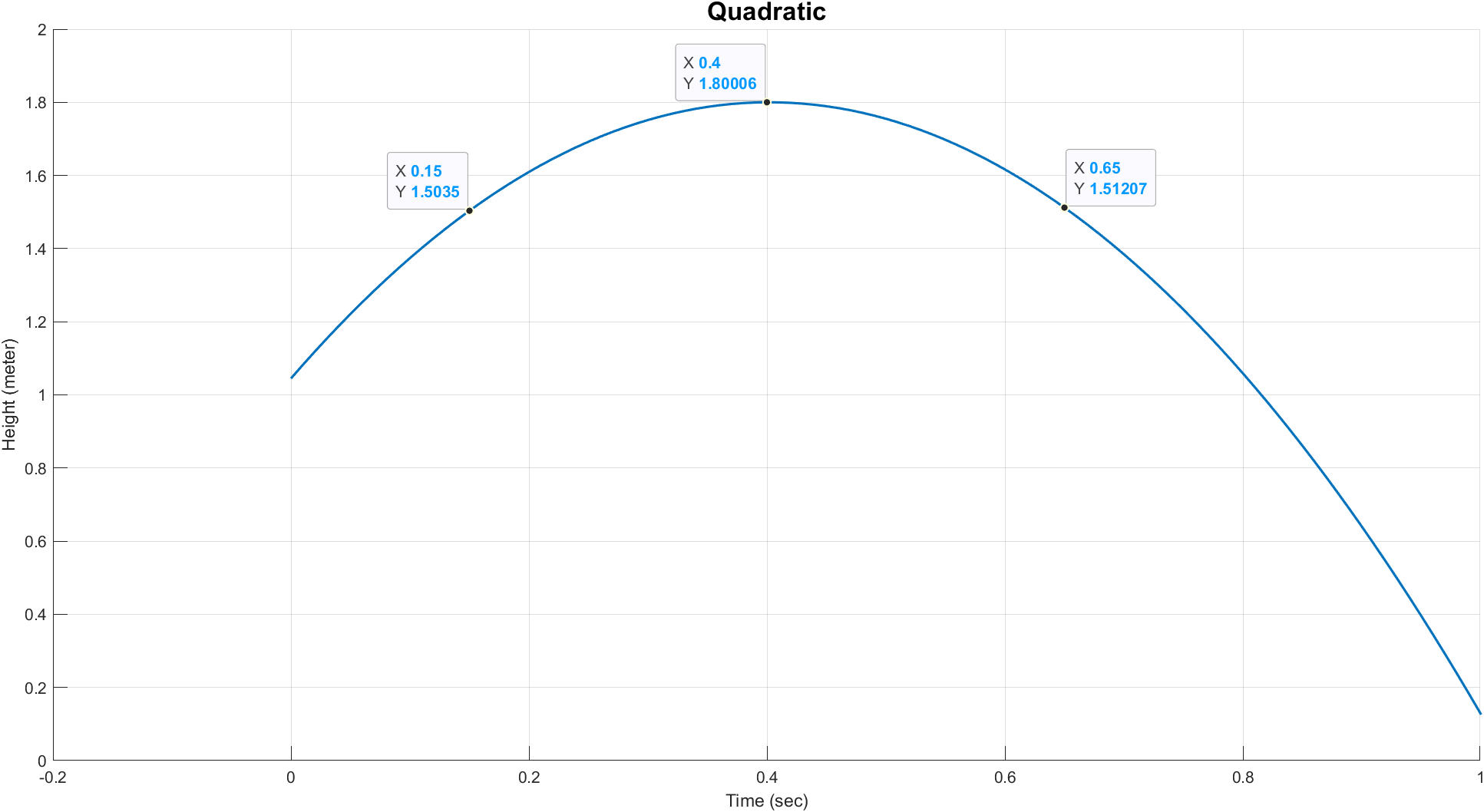
And for power model:

By comparing the of each model the best one is the quadratic even it is slightly lower than cubic but we always choose the one with lower number of parameters when the improvement from higher order not significant.

### 1.2.3. Prediction

When the height of the ball is at least 1.5 meters

From the quadratic model alone:

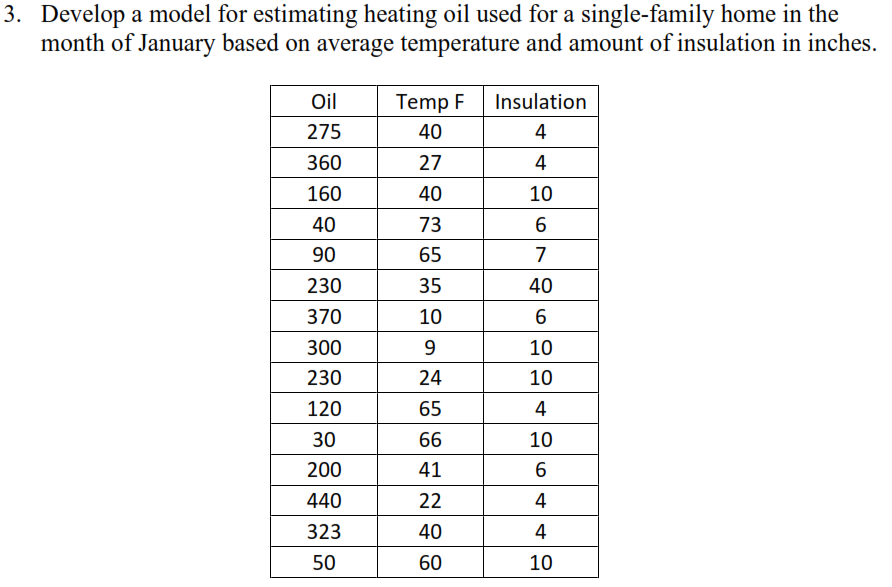


By sweeping over a range of time and calculate the height from the model

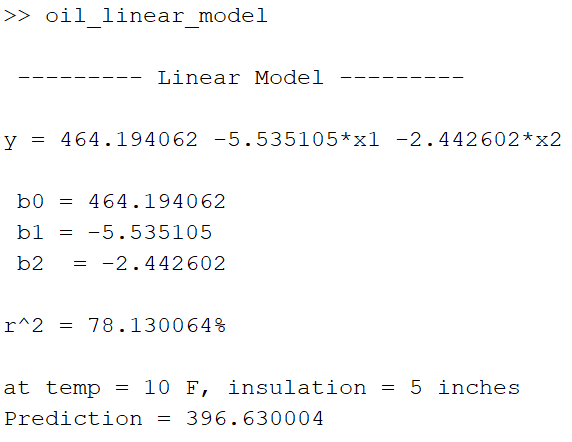
The height will we above 1.5 meters at time interval from 0.15 sec to 0.65 sec

And the maximum height is approximately 1.8 meters

## 1.3. Heating Oil Consumption Model



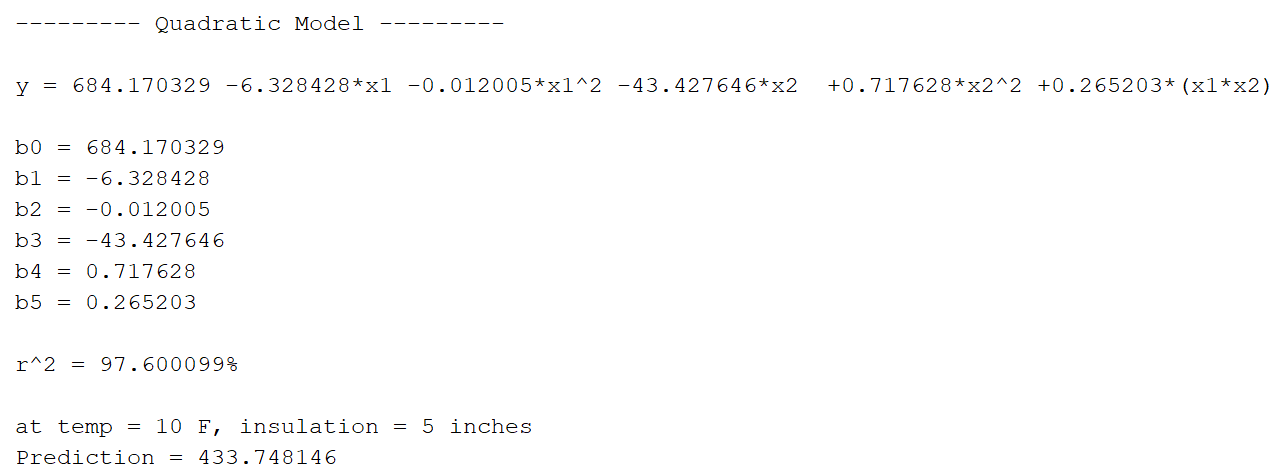
### 1.3.1. Linear Model Results



The final model equation is

### 1.3.2. Quadratic Model Results

only difference is the added parameters



The final model equation is

### 1.3.2. Comments

Used the matrix method to calculate the parameters. For linear model the regression equation is in the form

output matrix

independent variables matrix

parameters matrix

Usually, we would pick the model with the least number of parameters, but in this case

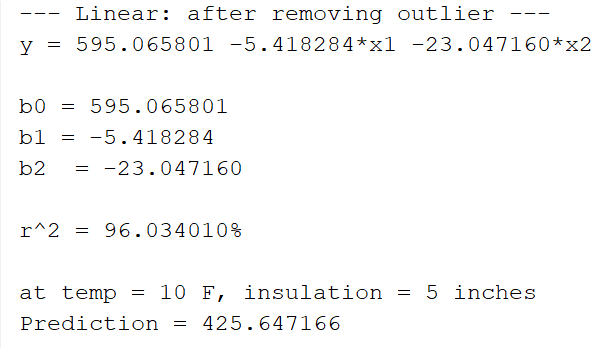
Linear:

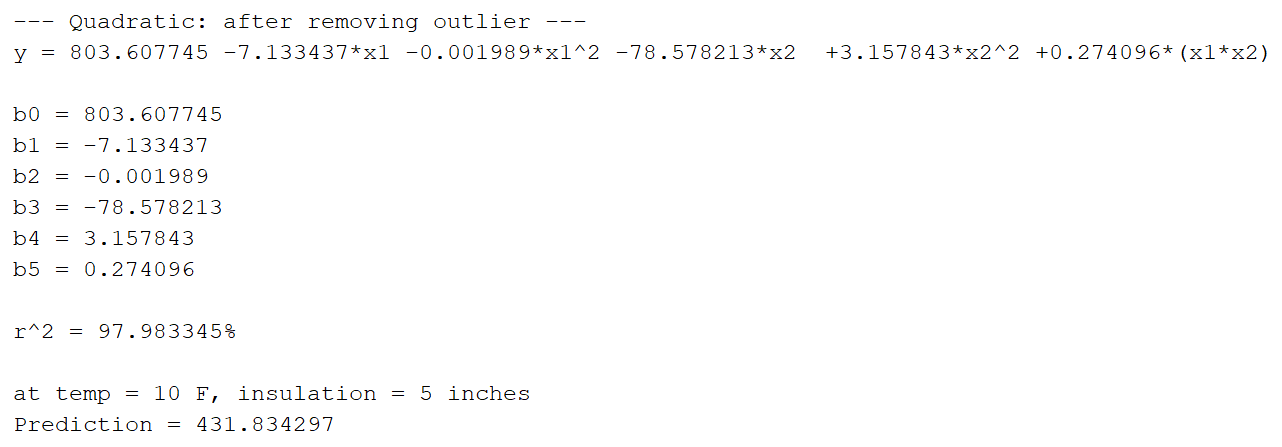
Quadratic:

There is a large gap of improvement in the coefficient of determinism of quadratic over linear. Hence, the quadratic model is more recommended in this case.

### 1.3.3. Removing Unreasonable Data

Sixth reading of 40 inches of insulation is too high for the usual range of data. Then recalculate all data of both models.





Now, clearly there is no much difference between both models. Hence, the linear model can be used.

# 2. Part 2

### 2.1 Results

Table : Final Results

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Features | | | | | |
| DCT | | PCA | | ICA | |
| Accuracy | Training Time | Accuracy | Training Time | Accuracy | Training Time |
| Classifier | |
| K-means Clustering | 1 | 62.65% | 0.619s | 63.15% | 0.903s | 64.4% | 0.166s |
| 4 | 89% | 1.221s | 88.65% | 1.812s | 81.5% | 0.542s |
| 16 | 93.15% | 3.504s | 93.25% | 4.783s | 89.35% | 1.424s |
| 32 | **95.4%** | 6.798s | 94.75% | 9.291s | 89.1% | 1.872s |
| SVM | Linear | 94.35% | 1.808s | 93.85% | 3.814s | 77.8% | 6.240s |
| Non-Linear (RBF) | 97.35% | 2.617s | **97.65%** | 7.158s | 93.8% | 0.783s |

|  |  |
| --- | --- |
| Figure 1: K-means, DCT, 32 Clusters | Figure 2: SVM, PCA, rbf (nonlinear) |

### 2.2 Notes

* We used Independent Component Analysis (ICA) as the third feature in this experiment. ICA is a technique that separates signals into additive subcomponents, assuming that at most one of those components is gaussian, and that all subcomponents are statistically independent.
* The non-linear SVM kernel we used is the Radial Basis-function kernel (RBF).

### 2.3 Conclusions

* Both DCT, and PCA features seem to be effective in capturing the most important features in the images while also reducing the computation cost.
* A supervised approach (SVM) as well as an unsupervised approach (K-means clustering) have both managed to classify the images with a very high accuracy when their parameters are fine-tuned for the application.
* Running the code on different machines can result in different training time for each classifier, but the algorithms can be considered relatively fast.

# 3. Appendix A: problem 1 codes

Linear model before removing outlier

x**=[**0**;** 1**;** 2**;** 3**;** 4**;** 5**;** 6**;** 7**;** 8**;** 9**]** **;**

y**=[**14000**;** 13000**;** 12000**;** 11000**;** 1050**;** 10000**;** 9500**;** 9000**;** 8700**;** 8000**]** **;**

scatter**(**x**,**y**)**

hold on

xlabel**(**'The number of years since 1987 x'**)**

ylabel**(**'The numbers of insured persons y'**)**

title**(**'Linear Regression Relation Between insured persons & no. of years'**)**

grid on

x**=[**ones**(**length**(**x**),**1**)** x**]** **;**

B**=**mldivide**(**x**,**y**)** **;**

yCalc **=** x**\***B**;**

x**(:,**1**)=[]** **;**

plot**(**x**,**yCalc**)**

legend**(**'Given Data'**,**'Linear fit'**,**'Location'**,**'best'**);**

Rsq **=** 1 **-** sum**((**y **-** yCalc**).^**2**)/**sum**((**y **-** mean**(**y**)).^**2**);**

Linear model after removing outlier

x**=[**0**;** 1**;** 2**;** 3**;** 5**;** 6**;** 7**;** 8**;** 9**]** **;**

y**=[**14000**;** 13000**;** 12000**;** 11000**;** 10000**;** 9500**;** 9000**;** 8700**;** 8000**]** **;**

scatter**(**x**,**y**)**

hold on

xlabel**(**'The number of years since 1987 x'**)**

ylabel**(**'The numbers of insured persons y'**)**

title**(**'Linear Regression Relation Between insured persons & no. of years'**)**

grid on

x**=[**ones**(**length**(**x**),**1**)** x**]** **;**

B**=**mldivide**(**x**,**y**)** **;**

yCalc **=** x**\***B**;**

x**(:,**1**)=[]** **;**

plot**(**x**,**yCalc**)**

legend**(**'Given Data'**,**'Linear fit'**,**'Location'**,**'best'**);**

Rsq **=** 1 **-** sum**((**y **-** yCalc**).^**2**)/**sum**((**y **-** mean**(**y**)).^**2**);**

Quadratic model before removing outlier

x**=[**0**;** 1**;** 2**;** 3**;** 4**;** 5**;** 6**;** 7**;** 8**;** 9**]** **;**

y**=[**14000**;** 13000**;** 12000**;** 11000**;** 1050**;** 10000**;** 9500**;** 9000**;** 8700**;** 8000**]** **;**

p **=** polyfit**(**x**,**y**,**2**);**

yCalc **=** polyval**(**p**,**x**);**

scatter**(**x**,**y**)**

hold on

xlabel**(**'The number of years since 1987 x'**)**

ylabel**(**'The numbers of insured persons y'**)**

title**(**'Quadratic Regression Relation Between insured persons & no. of years'**)**

grid on

plot**(**x**,**yCalc**)**

legend**(**'Given Data'**,**'Quadratic fit'**,**'Location'**,**'best'**);**

Rsq **=** 1 **-** sum**((**y **-** yCalc**).^**2**)/**sum**((**y **-** mean**(**y**)).^**2**);**

Quadratic model after removing outlier

x**=[**0**;** 1**;** 2**;** 3**;** 5**;** 6**;** 7**;** 8**;** 9**]** **;**

y**=[**14000**;** 13000**;** 12000**;** 11000**;** 10000**;** 9500**;** 9000**;** 8700**;** 8000**]** **;**

p **=** polyfit**(**x**,**y**,**2**);**

yCalc **=** polyval**(**p**,**x**);**

scatter**(**x**,**y**)**

hold on

xlabel**(**'The number of years since 1987 x'**)**

ylabel**(**'The numbers of insured persons y'**)**

title**(**'Quadratic Regression Relation Between insured persons & no. of years'**)**

grid on

plot**(**x**,**yCalc**)**

legend**(**'Given Data'**,**'Quadratic fit'**,**'Location'**,**'best'**);**

Rsq **=** 1 **-** sum**((**y **-** yCalc**).^**2**)/**sum**((**y **-** mean**(**y**)).^**2**)** **;**

Cubic model before removing outlier

x**=[**0**;** 1**;** 2**;** 3**;** 4**;** 5**;** 6**;** 7**;** 8**;** 9**]** **;**

y**=[**14000**;** 13000**;** 12000**;** 11000**;** 1050**;** 10000**;** 9500**;** 9000**;** 8700**;** 8000**]** **;**

p **=** polyfit**(**x**,**y**,**3**);**

yCalc **=** polyval**(**p**,**x**);**

scatter**(**x**,**y**)**

hold on

xlabel**(**'The number of years since 1987 x'**)**

ylabel**(**'The numbers of insured persons y'**)**

title**(**'Cubic Regression Relation Between insured persons & no. of years'**)**

grid on

plot**(**x**,**yCalc**)**

legend**(**'Given Data'**,**'Cubic fit'**,**'Location'**,**'best'**);**

Rsq **=** 1 **-** sum**((**y **-** yCalc**).^**2**)/**sum**((**y **-** mean**(**y**)).^**2**);**

Cubic model after removing outlier

x**=[**0**;** 1**;** 2**;** 3**;** 5**;** 6**;** 7**;** 8**;** 9**]** **;**

y**=[**14000**;** 13000**;** 12000**;** 11000**;** 10000**;** 9500**;** 9000**;** 8700**;** 8000**]** **;**

p **=** polyfit**(**x**,**y**,**3**);**

yCalc **=** polyval**(**p**,**x**);**

scatter**(**x**,**y**)**

hold on

xlabel**(**'The number of years since 1987 x'**)**

ylabel**(**'The numbers of insured persons y'**)**

title**(**'Cubic Regression Relation Between insured persons & no. of years'**)**

grid on

plot**(**x**,**yCalc**)**

legend**(**'Given Data'**,**'Cubic fit'**,**'Location'**,**'best'**);**

Rsq **=** 1 **-** sum**((**y **-** yCalc**).^**2**)/**sum**((**y **-** mean**(**y**)).^**2**)** **;**

# 4. Appendix B: problem 2 codes

Linear model

clear**;**

%% Given Data

y\_d **=** readmatrix**(**"height\_data.txt"**);**

x **=** readmatrix**(**"time\_data.txt"**);**

m **=** length**(**y\_d**);**

%% Matrices

X **=** **[**ones**(**m**,**1**),**x**];**

syms b0 b1

%% Estimating the Parameters

b **=** **(**X**'** **\*** X**)\** X**'** **\*** y\_d**;**

**[**b0**,**b1**]** **=** deal**(**b**(**1**),**b**(**2**));**

%% Scatter Plot

figure**(**1**)**

subplot**(**2**,**2**,**1**)**

scatter**(**x**,** y\_d**,** 'LineWidth'**,** 1.5**);**

hold on**;**

xlabel**(**'Time (sec)'**);** ylabel**(**'Height (meter)'**);**

title**(**"Linear"**,**'FontSize'**,**16**,**'FontWeight'**,**'bold'**);**

xlim**([-**0.2 1**]);**

grid on**;**

y **=** b0 **+** b1**\***x**;**

plot**(**x**,** y**,** 'LineWidth'**,** 1.5**);**

hold off**;**

%% Coeff of determinism

unexplained\_sum **=** sum**((**y\_d **-** y**).^**2**);**

total\_sum **=** sum**((**y\_d **-** mean**(**y\_d**)).^**2**);**

Rsqr **=** **(**total\_sum **-** unexplained\_sum**)** **/** total\_sum**;**

%% Print Results of Linear model

fprintf**(**"\n --------- Linear Model ---------\n"**);**

fprintf**(**"\n y = %f + %f\*x\n\n"**,**b0**,**b1**);**

fprintf**(**" b0 = %f\n b1 = %f\n\n"**,**b0**,**b1**);**

fprintf**(**"\n r^2 = %f%%\n\n"**,**Rsqr**\***100**);**

Cubic model

clear**;**

%% Given Data

y\_d **=** readmatrix**(**"height\_data.txt"**);**

x **=** readmatrix**(**"time\_data.txt"**);**

m **=** length**(**y\_d**);**

%% Matrices

X **=** **[**ones**(**m**,**1**),**x**,**x**.^**2**,**x**.^**3**];**

syms b0 b1 b2 b3

%% Estimating the Parameters

b **=** **(**X**'** **\*** X**)\** X**'** **\*** y\_d**;**

**[**b0**,**b1**,**b2**,**b3**]** **=** deal**(**b**(**1**),**b**(**2**),**b**(**3**),**b**(**4**));**

%% Scatter Plot

subplot**(**2**,**2**,**3**)**

scatter**(**x**,** y\_d**,** 'LineWidth'**,** 1.5**);**

hold on**;**

xlabel**(**'Time (sec)'**);** ylabel**(**'Height (meter)'**);**

title**(**"Cubic"**,**'FontSize'**,**16**,**'FontWeight'**,**'bold'**);**

xlim**([-**0.2 1**]);**

grid on**;**

y **=** b0 **+** b1**\***x **+** b2**\*(**x**.^**2**)** **+** b3**\*(**x**.^**3**);**

plot**(**x**,** y**,** 'LineWidth'**,** 1.5**);**

hold off**;**

%% Coeff of determinism

unexplained\_sum **=** sum**((**y\_d **-** y**).^**2**);**

total\_sum **=** sum**((**y\_d **-** mean**(**y\_d**)).^**2**);**

Rsqr **=** **(**total\_sum **-** unexplained\_sum**)** **/** total\_sum**;**

%% Print Results of Linear model

fprintf**(**"\n --------- Cubic Model ---------\n"**);**

fprintf**(**"\n y = %f + %f\*x + %f\*x^2 + %f\*x^3\n\n"**,**...

b0**,**b1**,**b2**,**b3**);**

fprintf**(**" b0 = %f\n b1 = %f\n b2 = %f\n b3 = %f\n\n"**,**...

b0**,**b1**,**b2**,**b3**);**

fprintf**(**"\n r^2 = %f%%\n\n"**,**Rsqr**\***100**);**

Quadratic model

clear**;**

%% Given Data

y\_d **=** readmatrix**(**"height\_data.txt"**);**

x **=** readmatrix**(**"time\_data.txt"**);**

m **=** length**(**y\_d**);**

%% Matrices

X **=** **[**ones**(**m**,**1**),**x**,**x**.^**2**];**

syms b0 b1 b2

%% Estimating the Parameters

b **=** **(**X**'** **\*** X**)\** X**'** **\*** y\_d**;**

**[**b0**,**b1**,**b2**]** **=** deal**(**b**(**1**),**b**(**2**),**b**(**3**));**

%% Scatter Plot

subplot**(**2**,**2**,**2**)**

scatter**(**x**,** y\_d**,** 'LineWidth'**,** 1.5**);**

hold on**;**

xlabel**(**'Time (sec)'**);** ylabel**(**'Height (meter)'**);**

title**(**"Quadratic"**,**'FontSize'**,**16**,**'FontWeight'**,**'bold'**);**

xlim**([-**0.2 1**]);**

grid on**;**

y **=** b0 **+** b1**\***x **+** b2**\*(**x**.^**2**);**

plot**(**x**,** y**,** 'LineWidth'**,** 1.5**);**

hold off**;**

%% Coeff of determinism

unexplained\_sum **=** sum**((**y\_d **-** y**).^**2**);**

total\_sum **=** sum**((**y\_d **-** mean**(**y\_d**)).^**2**);**

Rsqr **=** **(**total\_sum **-** unexplained\_sum**)** **/** total\_sum**;**

%% Print Results of Linear model

fprintf**(**"\n --------- Quadratic Model ---------\n"**);**

fprintf**(**"\n y = %f + %f\*x + %f\*x^2\n\n"**,**b0**,**b1**,**b2**);**

fprintf**(**" b0 = %f\n b1 = %f\n b2 = %f\n\n"**,**b0**,**b1**,**b2**);**

fprintf**(**"\n r^2 = %f%%\n\n"**,**Rsqr**\***100**);**

%% Prediction

figure**(**2**)**

x **=** 0**:**0.01**:**2**;**

hold on**;**

xlabel**(**'Time (sec)'**);** ylabel**(**'Height (meter)'**);**

title**(**"Quadratic"**,**'FontSize'**,**16**,**'FontWeight'**,**'bold'**);**

xlim**([-**0.2 1**]);**

grid on**;**

y = b0 + b1\*x + b2\*(x.^2);

plot(x, y, 'LineWidth', 1.5);

hold off;

Power model

clear**;**

%% Given Data

y\_d **=** readmatrix**(**"height\_data.txt"**);**

x **=** readmatrix**(**"time\_data.txt"**);**

x**(**1**)** **=** 0.0001**;**

m **=** length**(**y\_d**);**

%% Matrices

X **=** **[**ones**(**m**,**1**),**log**(**x**)];**

syms b0 b1

%% Estimating the Parameters

b **=** **(**X**'** **\*** X**)\** X**'** **\*** y\_d**;**

**[**b0**,**b1**]** **=** deal**(**b**(**1**),**b**(**2**));**

%% Scatter Plot

subplot**(**2**,**2**,**4**)**

scatter**(**x**,** y\_d**,** 'LineWidth'**,** 1.5**);**

hold on**;**

xlabel**(**'Time (sec)'**);** ylabel**(**'Height (meter)'**);**

title**(**"Power"**,**'FontSize'**,**16**,**'FontWeight'**,**'bold'**);**

xlim**([-**0.2 1**]);**

grid on**;**

y **=** b0 **+** b1**\***log**(**x**);**

plot**(**x**,** y**,** 'LineWidth'**,** 1.5**);**

hold off**;**

%% Coeff of determinism

unexplained\_sum **=** sum**((**y\_d **-** y**).^**2**);**

total\_sum **=** sum**((**y\_d **-** mean**(**y\_d**)).^**2**);**

Rsqr **=** abs**(**total\_sum **-** unexplained\_sum**)** **/** total\_sum**;**

%% Print Results of Linear model

fprintf**(**"\n --------- Power Model ---------\n"**);**

fprintf**(**"\n y = %f\*x^(%f)\n\n"**,**b0**,**b1**);**

fprintf**(**" b0 = %f\n b1 = %f\n\n"**,**b0**,**b1**);**

fprintf**(**"\n r^2 = %f%%\n\n"**,**Rsqr**\***100**);**

# 5. Appendix C: problem 3 codes

Linear model

clear**;**

%% Given Data

% y\_d(m,1) is oil consumption

y\_d **=** readmatrix**(**"oil\_data.txt"**);**

x1 **=** readmatrix**(**"temp\_data.txt"**);**

x2 **=** readmatrix**(**"insulation\_data.txt"**);**

m **=** length**(**y\_d**);**

%% Matrices

X **=** **[**ones**(**m**,**1**),**x1**,**x2**];**

syms b0 b1 b2

%% Estimating the Parameters

b **=** **(**X**'** **\*** X**)\** X**'** **\*** y\_d**;**

**[**b0**,**b1**,**b2**]** **=** deal**(**b**(**1**),**b**(**2**),**b**(**3**));**

%% Regression equation

y **=** b0 **+** b1**\***x1 **+** b2**\***x2**;**

%% Coeff of determinism

unexplained\_sum **=** sum**((**y\_d **-** y**).^**2**);**

total\_sum **=** sum**((**y\_d **-** mean**(**y\_d**)).^**2**);**

Rsqr **=** **(**total\_sum **-** unexplained\_sum**)** **/** total\_sum**;**

%% Prediction

syms y x1 x2

y **=** b0 **+** b1**\***x1 **+** b2**\***x2**;**

x1 **=** 10**;** % temperation is 5 F

x2 **=** 5**;** % insulation is 5 inches

result **=** eval**(**y**);**

%% Print Results of Linear model

fprintf**(**"\n --------- Linear Model ---------\n"**);**

fprintf**(**"\n y = %f %f\*x1 %f\*x2\n\n"**,**b0**,**b1**,**b2**);**

fprintf**(**" b0 = %f\n b1 = %f\n b2 = %f\n"**,**b0**,**b1**,**b2**);**

fprintf**(**"\n r^2 = %f%%\n"**,**Rsqr**\***100**);**

fprintf**(**"\n at temp = 10 F, insulation = 5 inches"**);**

fprintf**(**"\n Prediction = %f\n\n"**,**result**);**

%% Removing unreasonable data

y\_d **=** **[**y\_d**(**1**:**5**);**y\_d**(**7**:**m**)];**

x1 **=** readmatrix**(**"temp\_data.txt"**);**

x1 **=** **[**x1**(**1**:**5**);**x1**(**7**:**m**)];**

x2 **=** readmatrix**(**"insulation\_data.txt"**);**

x2 **=** **[**x2**(**1**:**5**);**x2**(**7**:**m**)];**

m **=** length**(**y\_d**);**

%% Recalculations

X = [ones(m,1),x1,x2];

b = (X' \* X)\ X' \* y\_d;

[b0,b1,b2] = deal(b(1),b(2),b(3));

y = b0 + b1\*x1 + b2\*x2;

unexplained\_sum = sum((y\_d - y).^2);

total\_sum = sum((y\_d - mean(y\_d)).^2);

Rsqr = (total\_sum - unexplained\_sum) / total\_sum;

syms y x1 x2

y = b0 + b1\*x1 + b2\*x2;

x1 = 10; % temperation is 5 F

x2 = 5; % insulation is 5 inches

result = eval(y);

fprintf(" --- Linear: after removing outlier ---");

fprintf("\n y = %f %f\*x1 %f\*x2\n\n",b0,b1,b2);

fprintf(" b0 = %f\n b1 = %f\n b2 = %f\n",b0,b1,b2);

fprintf("\n r^2 = %f%%\n",Rsqr\*100);

fprintf("\n at temp = 10 F, insulation = 5 inches");

fprintf("\n Prediction = %f\n\n",result)

Quadratic model

clear**;**

%% Given Data

% y\_d(m,1) is oil consumption

y\_d **=** readmatrix**(**"oil\_data.txt"**);**

x1 **=** readmatrix**(**"temp\_data.txt"**);**

x2 **=** readmatrix**(**"insulation\_data.txt"**);**

m **=** length**(**y\_d**);**

%% Matrices

X **=** **[**ones**(**m**,**1**),**x1**,**x1**.^**2**,**x2**,**x2**.^**2**,**x1**.\***x2**];**

syms b0 b1 b2 b3 b4 b5

%% Estimating the Parameters

b **=** **(**X**'** **\*** X**)\** X**'** **\*** y\_d**;**

**[**b0**,**b1**,**b2**,**b3**,**b4**,**b5**]** **=** deal**(**b**(**1**),**b**(**2**),**b**(**3**),**b**(**4**),**b**(**5**),**b**(**6**));**

y **=** b0 **+** b1**\***x1 **+** b2**\*(**x1**.^**2**)** **+** b3**\***x2 **+** b4**\*(**x2**.^**2**)** **+** b5**\*(**x1**.\***x2**);**

%% Coeff of determinism

unexplained\_sum **=** sum**((**y\_d **-** y**).^**2**);**

total\_sum **=** sum**((**y\_d **-** mean**(**y\_d**)).^**2**);**

Rsqr **=** **(**total\_sum **-** unexplained\_sum**)** **/** total\_sum**;**

%% Prediction

syms y x1 x2

y **=** b0 **+** b1**\***x1 **+** b2**\*(**x1**.^**2**)** **+** b3**\***x2 **+** b4**\*(**x2**.^**2**)** **+** b5**\*(**x1**.\***x2**);**

x1 **=** 10**;** % temperation is 5 F

x2 **=** 5**;** % insulation is 5 inches

result **=** eval**(**y**);**

%% Print Results of Linear model

fprintf**(**"\n --------- Quadratic Model ---------\n"**);**

fprintf**(**"\n y = %f %f\*x1 %f\*x1^2 %f\*x2 +%f\*x2^2 +%f\*(x1\*x2)\n\n"**,**...

b0**,**b1**,**b2**,**b3**,**b4**,**b5**);**

fprintf**(**" b0 = %f\n b1 = %f\n b2 = %f\n b3 = %f\n b4 = %f\n b5 = %f\n"**,**...

b0**,**b1**,**b2**,**b3**,**b4**,**b5**);**

fprintf**(**"\n r^2 = %f%%\n"**,**Rsqr**\***100**);**

fprintf**(**"\n at temp = 10 F, insulation = 5 inches"**);**

fprintf**(**"\n Prediction = %f\n\n"**,**result**);**

%% Removing unreasonable data

y\_d **=** **[**y\_d**(**1**:**5**);**y\_d**(**7**:**m**)];**

x1 **=** readmatrix**(**"temp\_data.txt"**);**

x1 **=** **[**x1**(**1**:**5**);**x1**(**7**:**m**)];**

x2 **=** readmatrix**(**"insulation\_data.txt"**);**

x2 **=** **[**x2**(**1**:**5**);**x2**(**7**:**m**)];**

m = length(y\_d);

%% Recalculations

X = [ones(m,1),x1,x1.^2,x2,x2.^2,x1.\*x2];

b = (X' \* X)\ X' \* y\_d;

[b0,b1,b2,b3,b4,b5] = deal(b(1),b(2),b(3),b(4),b(5),b(6));

y = b0 + b1\*x1 + b2\*(x1.^2) + b3\*x2 + b4\*(x2.^2) + b5\*(x1.\*x2);

unexplained\_sum = sum((y\_d - y).^2);

total\_sum = sum((y\_d - mean(y\_d)).^2);

Rsqr = (total\_sum - unexplained\_sum) / total\_sum;

syms y x1 x2

y = b0 + b1\*x1 + b2\*(x1.^2) + b3\*x2 + b4\*(x2.^2) + b5\*(x1.\*x2);

x1 = 10; % temperation is 5 F

x2 = 5; % insulation is 5 inches

result = eval(y);

fprintf(" --- Quadratic: after removing outlier ---");

fprintf("\n y = %f %f\*x1 %f\*x1^2 %f\*x2 +%f\*x2^2 +%f\*(x1\*x2)\n\n",...

b0,b1,b2,b3,b4,b5);

fprintf(" b0 = %f\n b1 = %f\n b2 = %f\n b3 = %f\n b4 = %f\n b5 = %f\n",...

b0,b1,b2,b3,b4,b5);

fprintf("\n r^2 = %f%%\n",Rsqr\*100);

fprintf("\n at temp = 10 F, insulation = 5 inches");

fprintf("\n Prediction = %f\n\n",result);

# 6. Appendix D: part 2 code

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib**.**pyplot **as** plt

**import** seaborn **as** sns

**import** time **,** glob **,** re

**from** scipy**.**fftpack **import** dct **,**idct

**import** sklearn

**from** sklearn**.**decomposition **import** PCA**,** FastICA

**from** sklearn**.**cluster **import** KMeans

**from** sklearn**.**metrics **import** confusion\_matrix**,** classification\_report

**from** sklearn**.**metrics **import** accuracy\_score

**from** sklearn **import** svm

**from** sklearn**.**discriminant\_analysis **import** LinearDiscriminantAnalysis **as** LDA

**from** PIL **import** Image

# In[2]:

path\_train**=**'D:/input/reduced-mnist/Reduced\_MNIST\_Data/Reduced\_Trainging\_data'

path\_test**=**'D:/input/reduced-mnist/Reduced\_MNIST\_Data/Reduced\_Testing\_data'

images\_train**=[]**

images\_test**=[]**

**for** i **in** **range(**10**):**

# A list of all training,testing file names

list\_0 **=** glob**.**glob**(**'{}/{}/\*.jpg'**.format(**path\_train**,**i**))** #[:100]

images\_train**.**append**(**list\_0**)**

list\_1 **=** glob**.**glob**(**'{}/{}/\*.jpg'**.format(**path\_test**,**i**))** #[:20]

images\_test**.**append**(**list\_1**)**

# Expanding sublists into one list

images\_train **=** **[**item **for** sublist **in** images\_train **for** item **in** sublist**]**

images\_test **=** **[**item **for** sublist **in** images\_test **for** item **in** sublist**]**

#training , test data from lists

X\_train **=** np**.**array**([**np**.**array**(**Image**.open(**fname**))** **for** fname **in** images\_train**])**

X\_test **=** np**.**array**([**np**.**array**(**Image**.open(**fname**))** **for** fname **in** images\_test**])**

y\_train**=**np**.**array**([list(map(int,** re**.**findall**(**r'\b\d\b'**,** fname**)))[**0**]** **for** fname **in** images\_train**])**

y\_test**=**np**.**array**([list(map(int,** re**.**findall**(**r'\b\d\b'**,** fname**)))[**0**]** **for** fname **in** images\_test**])**

# Normalization for faster convergence

X\_train **=** X\_train**/**255

X\_test **=** X\_test**/**255

# DCT

# In[3]:

# Functions used to extract DCT features

**def** zigzag**(**a**):**

comp**=**np**.**concatenate**([**np**.**diagonal**(**a**[::-**1**,:],** i**)[::(**2**\*(**i **%** 2**)-**1**)]** **for** i **in** **range(**1**-**a**.**shape**[**0**],** a**.**shape**[**0**])])**

**return** comp**[:**200**]**

**def** dct\_extract**(**a**):**

features**=**np**.**zeros**((**a**.**shape**[**0**],**200**))**

**for** i **in** **range(**a**.**shape**[**0**]):**

z\_features**=**zigzag**(**dct**(**dct**(**a**[**i**].**T**,** norm**=**'ortho'**).**T**,** norm**=**'ortho'**))**

features**[**i**]=**z\_features

extracted**=**features**.**reshape**((**a**.**shape**[**0**],-**1**))**

**return** extracted

# In[4]:

X\_train\_DCT**=**dct\_extract**(**X\_train**)**

X\_test\_DCT**=**dct\_extract**(**X\_test**)**

# PCA

#

# In[5]:

pca\_model **=** PCA**(**.95**)** #we want a 95% variance

pca\_model**.**fit**(**X\_train**.**reshape**((**X\_train**.**shape**[**0**],**28**\***28**)))**

X\_train\_PCA **=** pca\_model**.**transform**(**X\_train**.**reshape**((**X\_train**.**shape**[**0**],**28**\***28**)))**

X\_test\_PCA **=** pca\_model**.**transform**(**X\_test**.**reshape**((**X\_test**.**shape**[**0**],**28**\***28**)))**

**print(**"For 95% varinace, there are {} components"**.format(**pca\_model**.**n\_components\_**))**

# ICA

# Independent Component Analysis is a computational method for separating a multivariate signal into additive subcomponents

# In[6]:

ica\_model **=** FastICA**(**n\_components**=**10**)**

X\_train\_ICA **=** ica\_model**.**fit\_transform**(**X\_train**.**reshape**((**X\_train**.**shape**[**0**],**784**)),** y\_train**)**

X\_test\_ICA **=** ica\_model**.**transform**(**X\_test**.**reshape**((**X\_test**.**shape**[**0**],**784**)))**

# ## K-Mean Classifiers

# In[7]:

#calculate the accuracies

**def** acc\_calc**(**y**,** y\_hat **,**c**):**

y\_cluster **=** np**.**zeros**(**y**.**shape**)**

y\_unique **=** np**.**unique**(**y**)**

y\_unique\_ord **=** np**.**arange**(**y\_unique**.**shape**[**0**])**

**for** ind **in** **range(**y\_unique**.**shape**[**0**]):**

y**[**y**==**y\_unique**[**ind**]]** **=** y\_unique\_ord**[**ind**]**

y\_unique **=** np**.**unique**(**y**)**

bins **=** np**.**concatenate**((**y\_unique**,** **[**np**.max(**y\_unique**)+**1**]),** axis**=**0**)**

**for** cluster **in** np**.**unique**(**y\_hat**):**

hist**,** \_ **=** np**.**histogram**(**y**[**y\_hat**==**cluster**],** bins**=**bins**)**

correct **=** np**.**argmax**(**hist**)**

y\_cluster**[**y\_hat**==**cluster**]** **=** correct

**if(**c**):**

**return** accuracy\_score**(**y**,** y\_cluster**)**

**else:**

**return** y\_cluster

# In[8]:

#calculating the k-mean clusters

**def** kmean\_cluster**(**X\_train**,**X\_test**,**y\_test**):**

no\_clusters **=** **[**10**,**40**,**160**,**320**]**

**for** i **in** no\_clusters**:**

kmeans **=** KMeans**(**n\_clusters **=** i**,**n\_init**=**5**,**max\_iter**=**10000**,**algorithm**=**'lloyd'**,**random\_state**=**0**)**

**print(**"Using {} clusters per class :"**.format(int(**i**/**10**)))**

start **=** time**.**time**()**

kmeans**.**fit**(**X\_train**)**

end**=**time**.**time**()**

**print(**"Training Time ="**,**end**-**start**,**"sec"**)**

y\_hat**=**kmeans**.**predict**(**X\_test**)**

accuracy**=**acc\_calc**(**y\_test**,** y\_hat **,**1**)**

**print(**"Accuracy : "**,**accuracy**,**"\n"**)**

# DCT Results

# In[9]:

kmean\_cluster**(**X\_train\_DCT**,**X\_test\_DCT**,**y\_test**)**

# PCA Results

# In[10]:

kmean\_cluster**(**X\_train\_PCA**,**X\_test\_PCA**,**y\_test**)**

# ICA Results

# In[11]:

kmean\_cluster**(**X\_train\_ICA**,**X\_test\_ICA**,**y\_test**)**

# # SVM Classifiers

# In[12]:

#using the linear kernel and the radial-basis function (rbf) non-linear kernel

**def** SVM\_classifier**(**X**,**y**,**X\_ts**,**y\_ts**):**

**for** kernel **in** **(**'linear'**,** 'rbf'**):**

model **=** svm**.**SVC**(**kernel**=**kernel**,** C**=**1**)**

start **=** time**.**time**()**

model**.**fit**(**X**,** y**)**

end **=** time**.**time**()**

**print(**'Using the {} kernel: '**.format(**kernel**))**

**print(**"Training Time ="**,**end**-**start**,**" sec"**)**

y\_hat **=** model**.**predict**(**X**)**

y\_hat\_ts**=** model**.**predict**(**X\_ts**)**

**print(**"Training Accuracy: {}"**.format(**accuracy\_score**(**y\_hat**,**y**))** **)**

**print(**"Testing Accuracy: {}\n"**.format(**accuracy\_score**(**y\_hat\_ts**,**y\_ts**))** **)**

# DCT Results

# In[13]:

SVM\_classifier**(**X\_train\_DCT**,**y\_train**,**X\_test\_DCT**,**y\_test**)**

# PCA Results

# In[14]:

SVM\_classifier**(**X\_train\_PCA**,**y\_train**,**X\_test\_PCA**,**y\_test**)**

# ICA Results

# In[15]:

SVM\_classifier**(**X\_train\_ICA**,**y\_train**,**X\_test\_ICA**,**y\_test**)**

# Confusion Matrix For the Best Classifiers

# In[16]:

**def** confusion\_matrix**(**y**,**y\_hat**):**

df **=** pd**.**DataFrame**({**'Labels'**:** y**,** 'predictions'**:** y\_hat**})**

ct **=** pd**.**crosstab**(**df**[**'Labels'**],** df**[**'predictions'**])**

sns**.**heatmap**(**ct**,** annot**=True,** cmap**=**'Reds'**,** fmt**=**'g'**)**

plt**.**xlabel**(**'Predictions'**)**

plt**.**ylabel**(**'Labels'**)**

plt**.**title**(**'Confusion Matrix'**)**

plt**.**show**()**

# DCT-based K-means with 32 clusters

# In[17]:

kmeans\_model **=** KMeans**(**n\_clusters **=**320**,**n\_init**=**5**,**max\_iter**=**10000**,**algorithm**=**'lloyd'**,**random\_state**=**0**)**

kmeans\_model**.**fit**(**X\_train\_DCT**)**

predicted**=**kmeans\_model**.**predict**(**X\_test\_DCT**)**

y\_hat**=**acc\_calc**(**y\_test**,** predicted **,** 0**)**

confusion\_matrix**(**y\_test**,**y\_hat**)**

# PCA-based SVM Classifier using the rbf kernel

# In[18]:

svm\_model **=** svm**.**SVC**(**kernel**=**'rbf'**)**

svm\_model**.**fit**(**X\_train\_PCA**,** y\_train**)**

predicted\_svm**=** svm\_model**.**predict**(**X\_test\_PCA**)**

confusion\_matrix**(**y\_test**,**predicted\_svm**)**

# In[ ]: