	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

1.1. The numbers of insured persons with an insurance company

The numbers of insured persons y with an insurance company for the years 1987 to 1996 are shown in the table.

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

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:

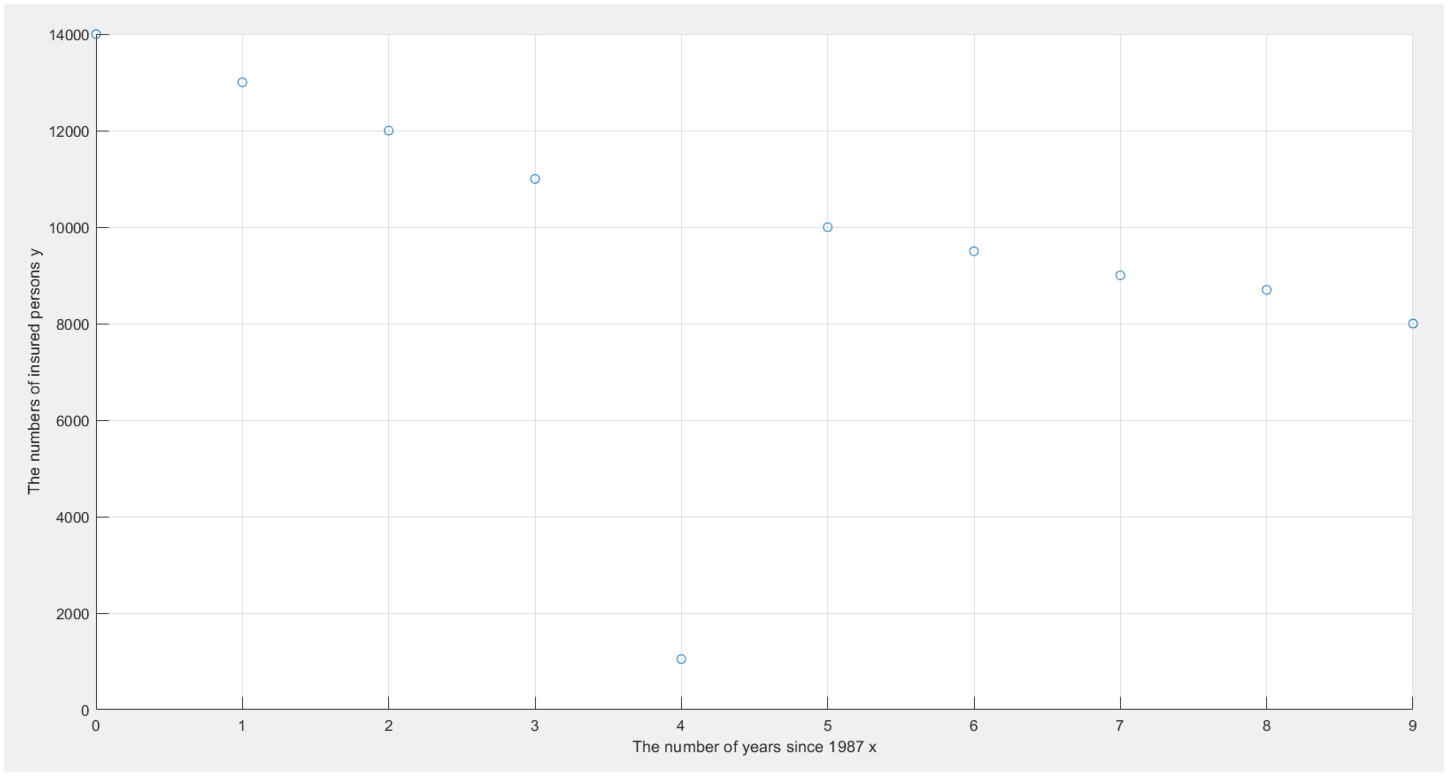


Fig.1 scatterplot of data (with the outlier) against number of years since 1987

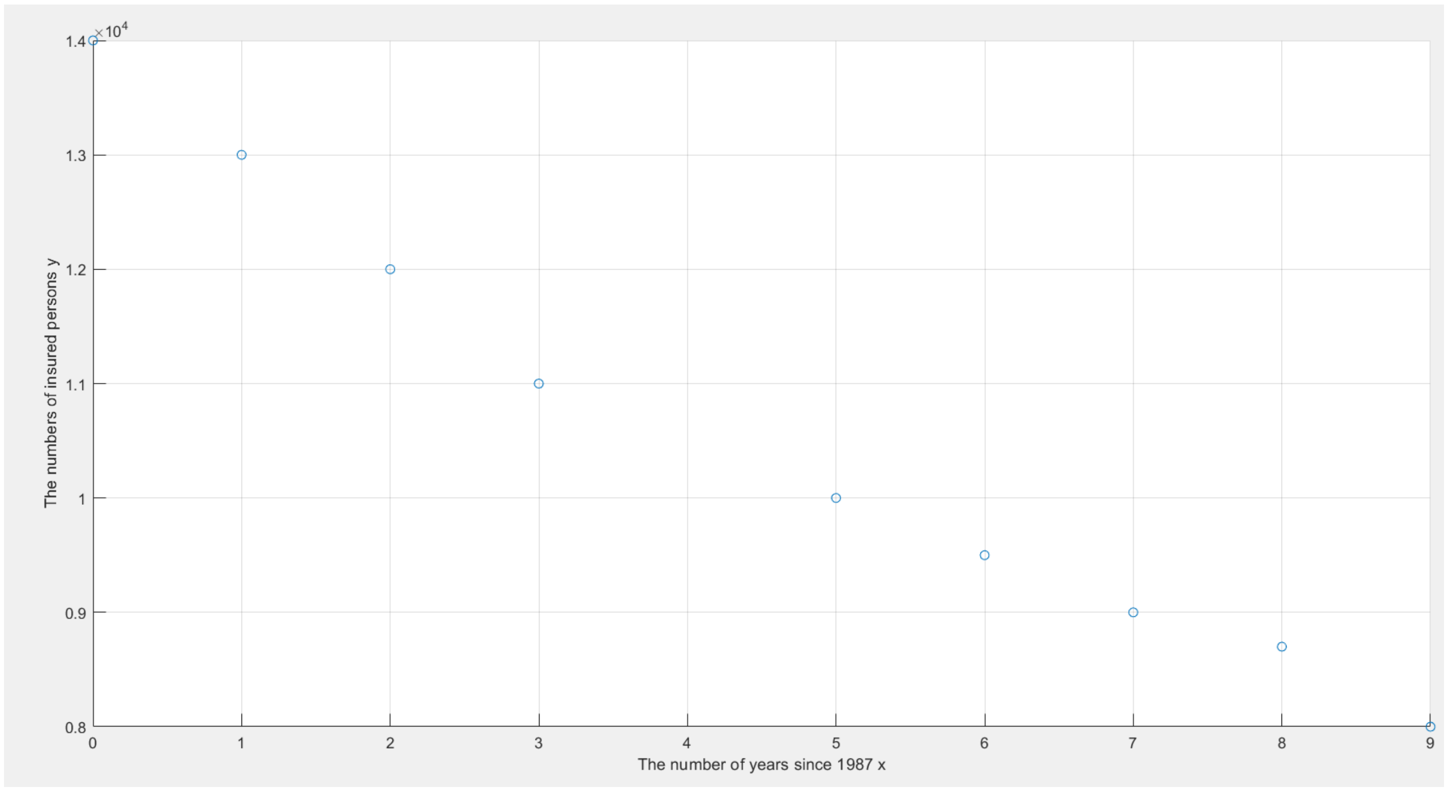


Fig.2 scatterplot of data (without the outlier) against number of years since 1987

a) Fit linear, quadratic, cubic, by comparing the values of R^2 . 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

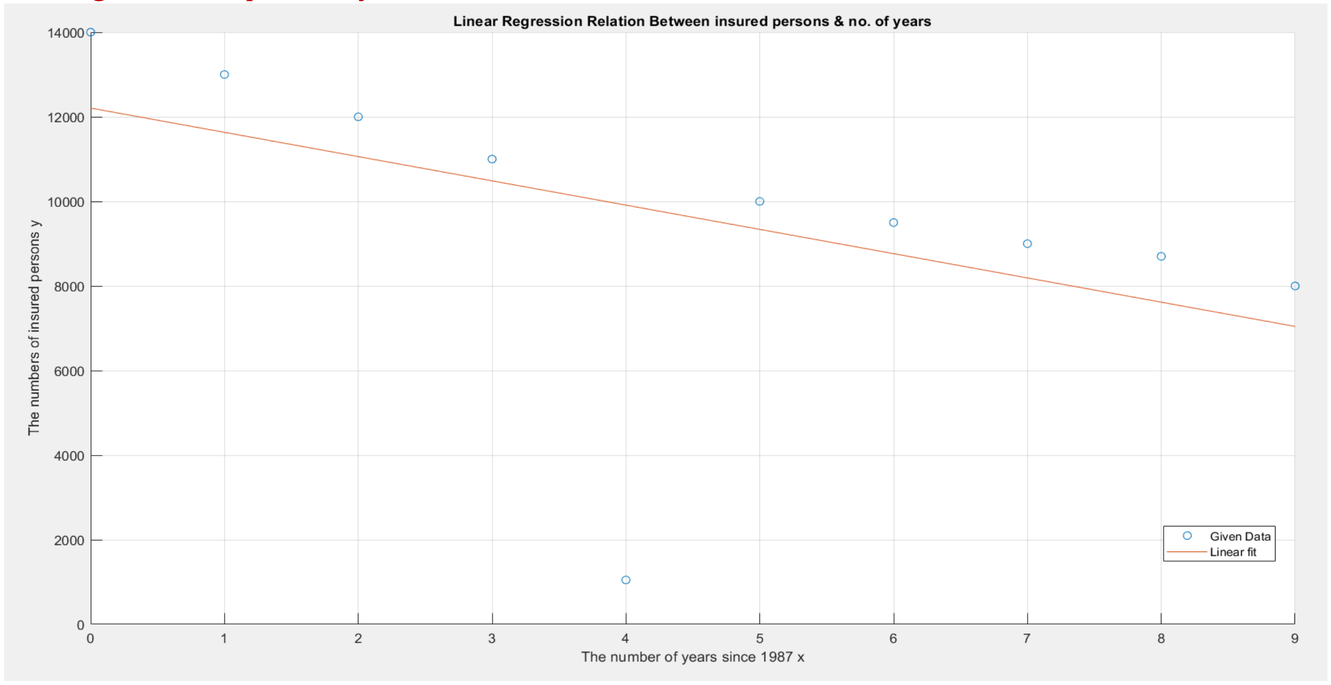
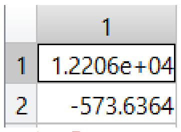


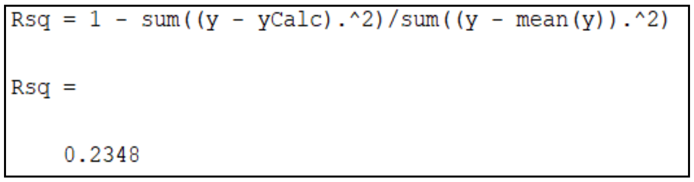
Fig.3 Linear Regression Relation Between insured persons & number of years (with the outlier)

a =>
b =>



	1
1	1.2206e+04
2	-573.6364

Fig.4 Linear Regression Parameters (with the outlier)



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

Rsq =

    0.2348
```

Fig.5 Linear Regression R^2 (with the outlier)

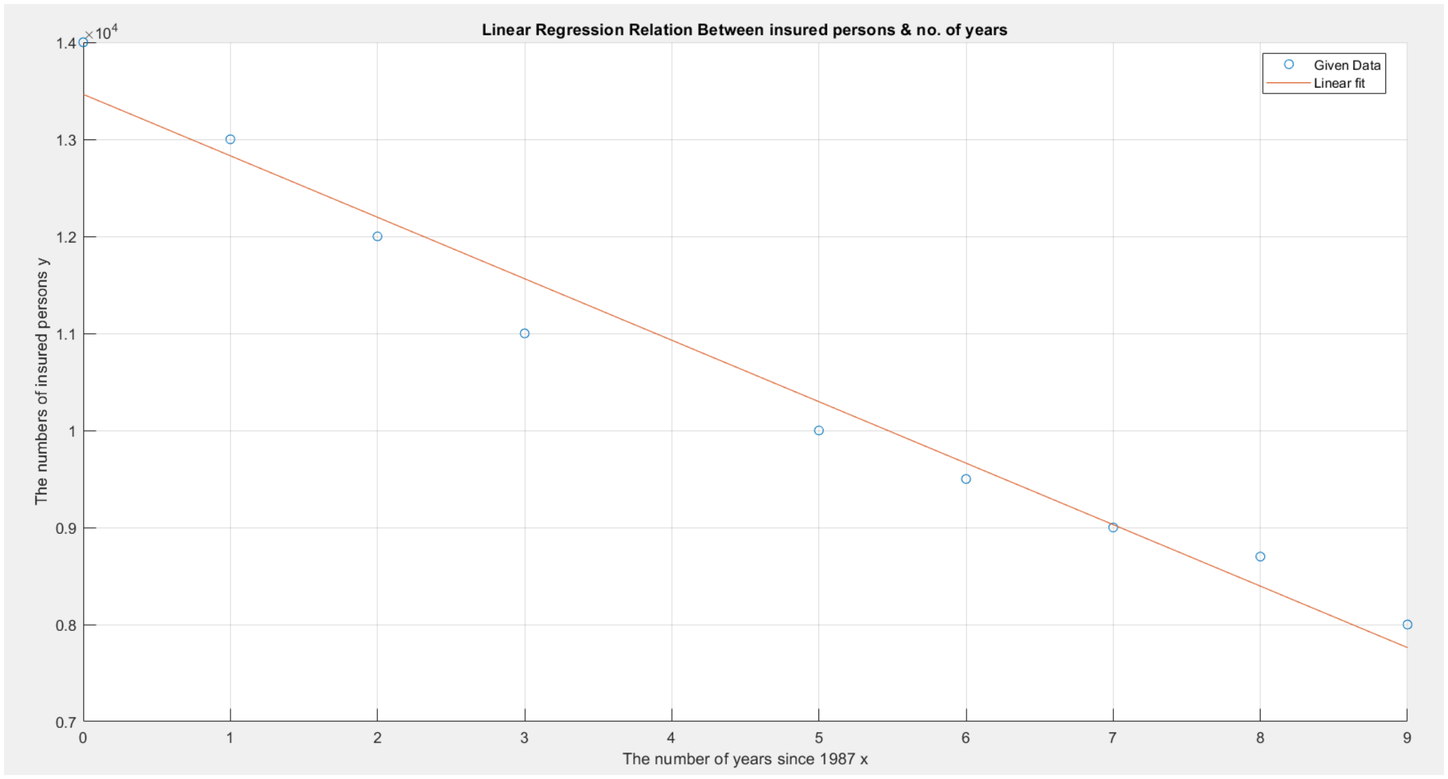
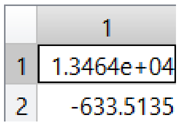
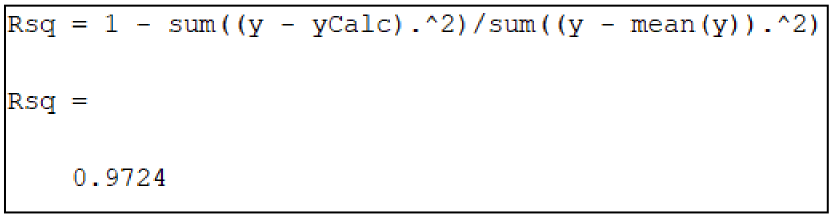


Fig.6 Linear Regression Relation Between insured persons & number of years (with no outlier)



		1
a =>	1	1.3464e+04
b =>	2	-633.5135

Fig.7 Linear Regression Parameters (with no outlier)



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

Rsq =

    0.9724
```

Fig.8 Linear Regression R^2 (with no outlier)

-Quadratic Regression : equation : $y = c+bx+ax^2$

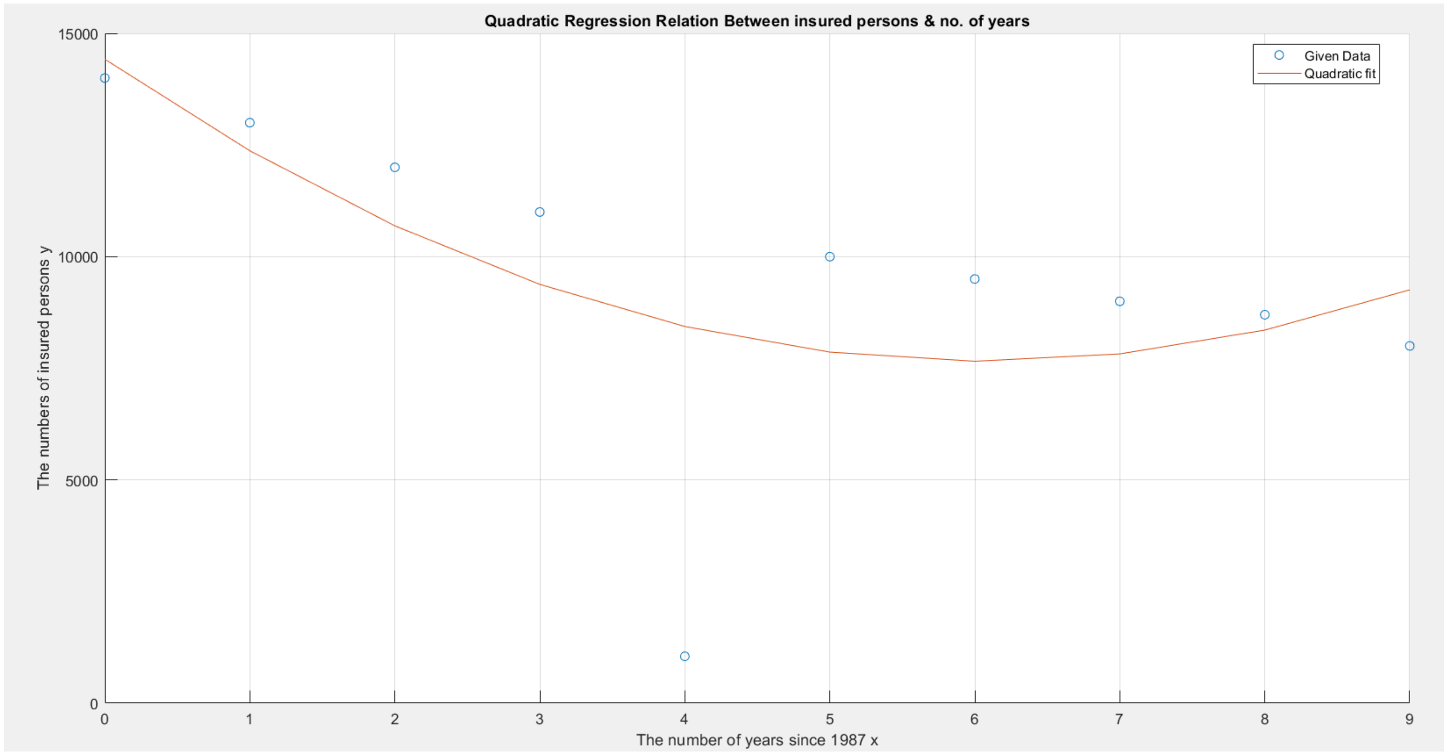
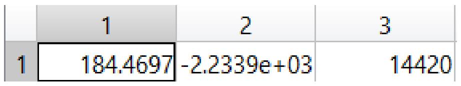


Fig.9 Quadratic Regression Relation Between insured persons & number of years (with the outlier)



	1	2	3
1	184.4697	-2.2339e+03	14420

Fig.10 Quadratic Regression Parameters (with the outlier)

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

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

Rsq =

    0.3901
```

Fig.11 Quadratic Regression R^2 (with the outlier)

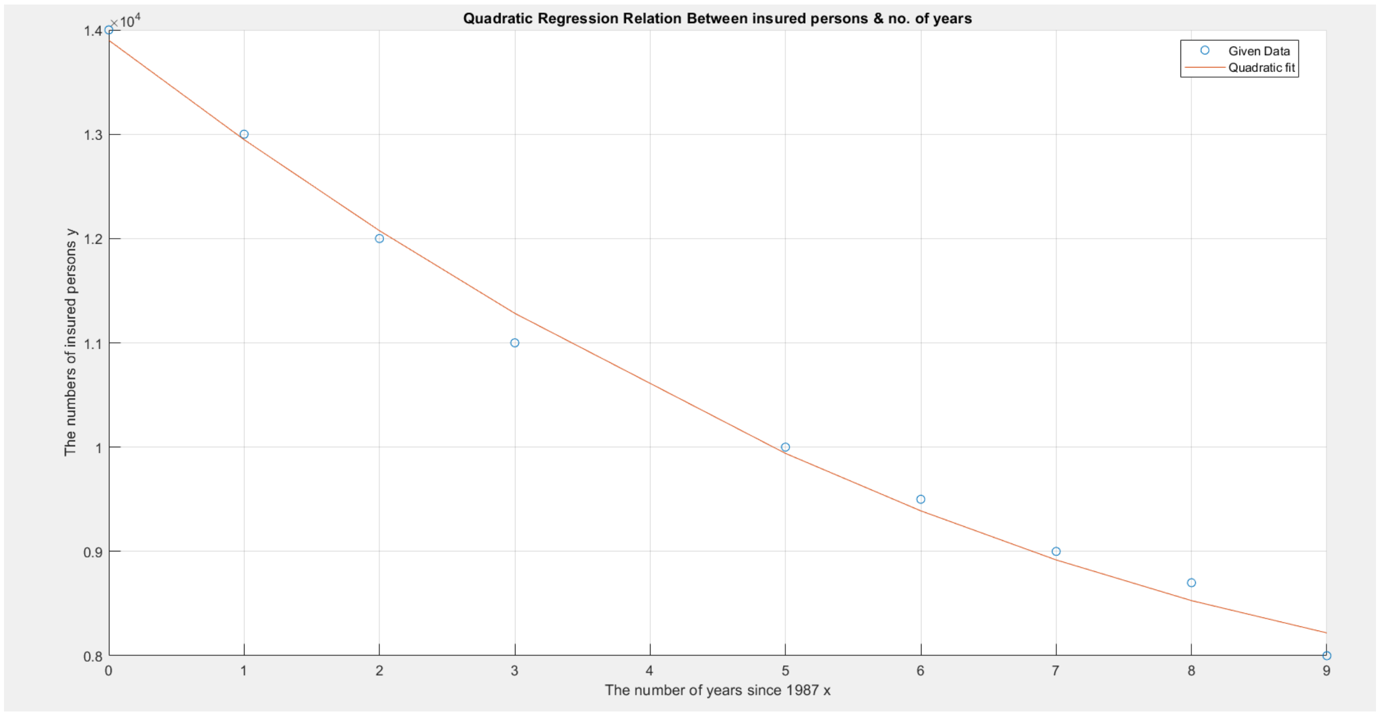
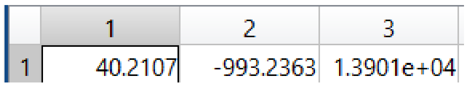


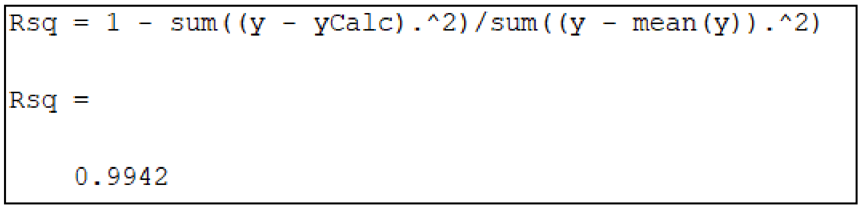
Fig.12 Quadratic Regression Relation Between insured persons & number of years (with no outlier)



	1	2	3
1	40.2107	-993.2363	1.3901e+04

Fig.13 Quadratic Regression Parameters (with no outlier)

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



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

Rsq =

    0.9942
```

Fig.14 Quadratic Regression R^2 (with no outlier)

-Cubic Regression : equation : $y = d+cx+bx^2+ax^3$

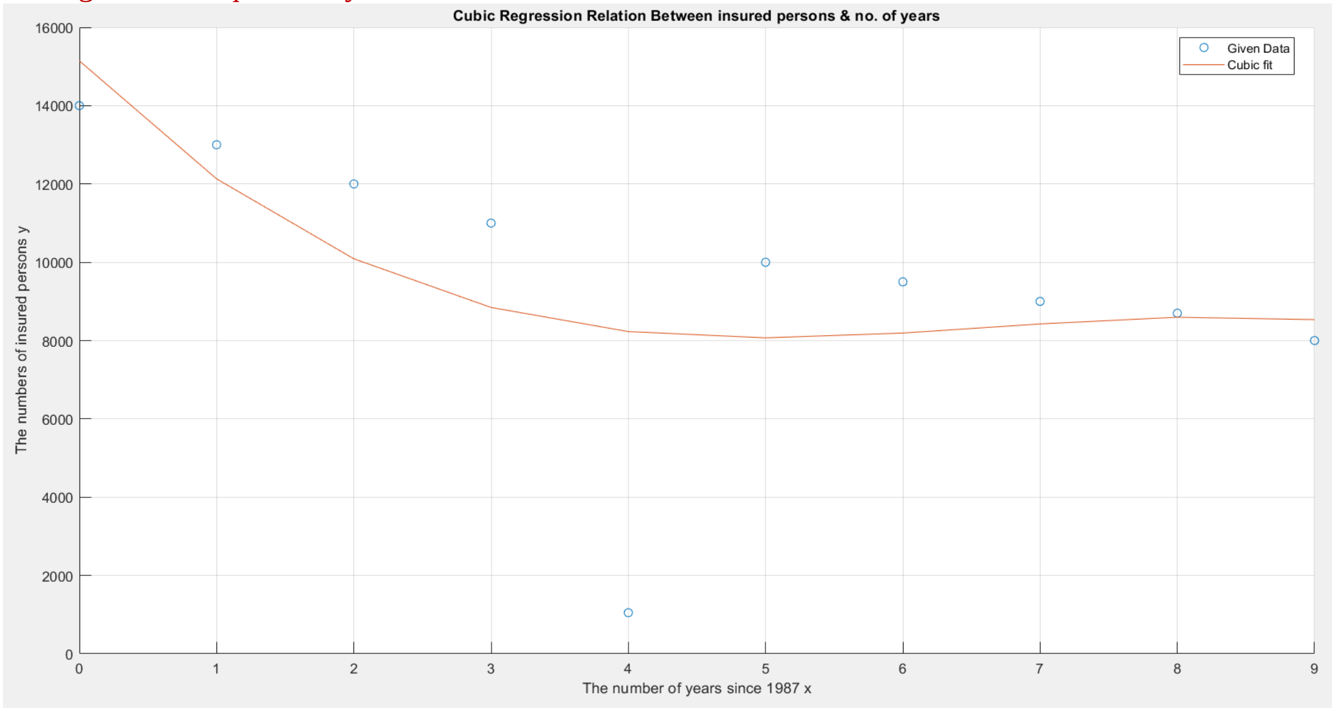
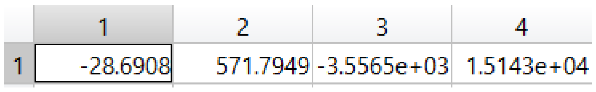


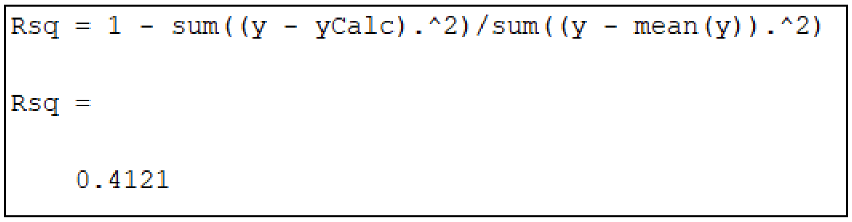
Fig.15 Cubic Regression Relation Between insured persons & number of years (with the outlier)



	1	2	3	4
1	-28.6908	571.7949	-3.5565e+03	1.5143e+04

Fig.16 Cubic Regression Parameters (with the outlier)

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



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

Rsq =

    0.4121
```

Fig.17 Cubic Regression R^2 (with the outlier)

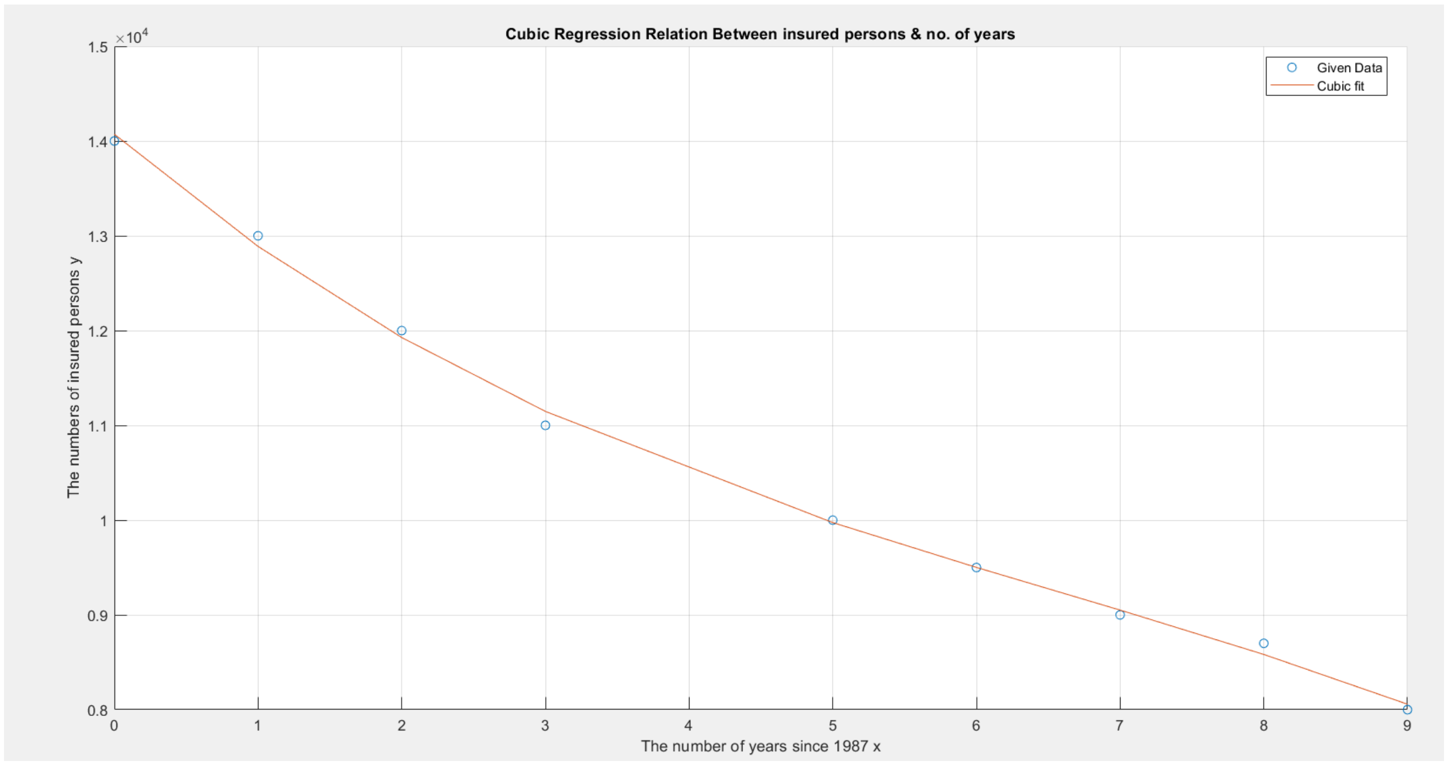
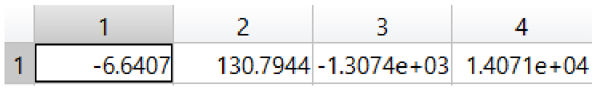
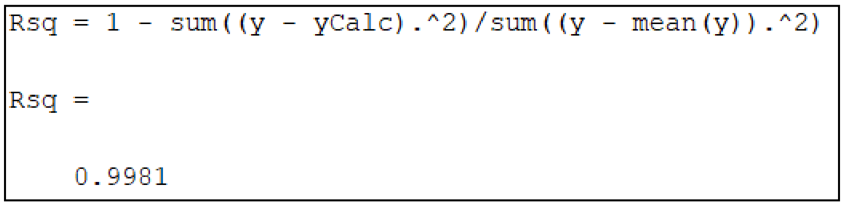


Fig.18 Cubic Regression Relation Between insured persons & number of years (with no outlier)



	1	2	3	4
1	-6.6407	130.7944	-1.3074e+03	1.4071e+04

Fig.19 Cubic Regression Parameters (with no outlier)



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

Rsq =

    0.9981
```

Fig.20 Cubic Regression R^2 (with no outlier)

a) The function that best fits the data is the Linear Regression model with no outlier as it has very high R^2 with lower number of parameters, although Quadratic and Cubic Regressions have higher R^2 than Linear Regression but the priority is to the Regression model with high enough R^2 and lower parameters and as the difference between the values of R^2 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 : $y = 13464 – 633.5135 * x$

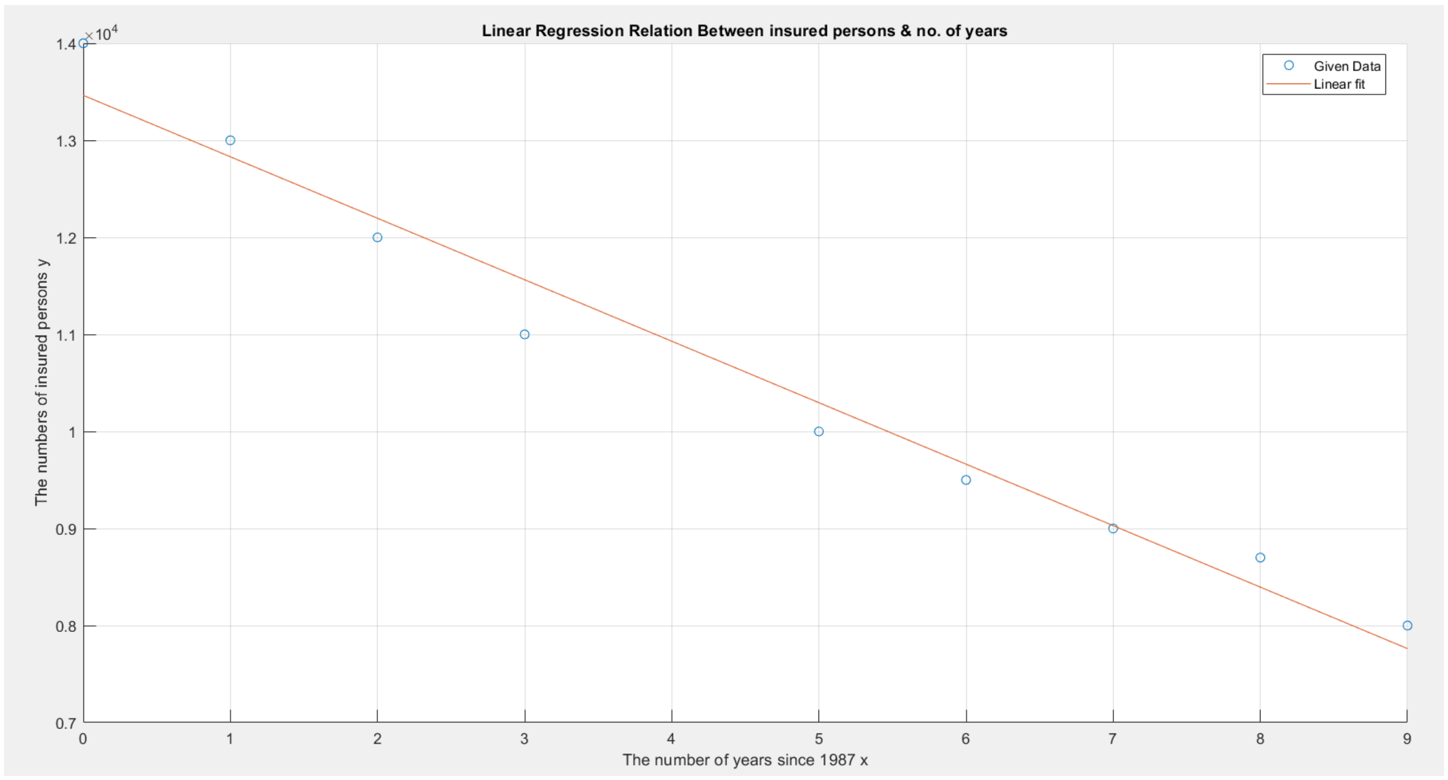
- Quadratic Regression equation :

- With outlier : $y = 14420 – 2233.9 * x + 184.4697 * x^2$
- With no outlier : $y = 13901 – 993.2363 * x + 40.2107 * x^2$

- Cubic Regression equation :

- With outlier : $y = 15143 – 3556.5 * x + 571.7949 * x^2 – 28.6908 * x^3$
- With no outlier : $y = 14071 – 1307.4 * x + 130.7944 * x^2 – 6.6407 * x^3$

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

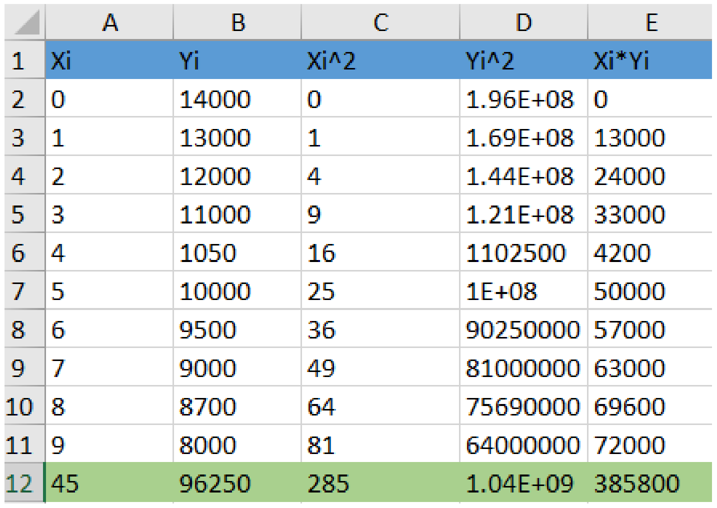
Note:

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 R^2 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.

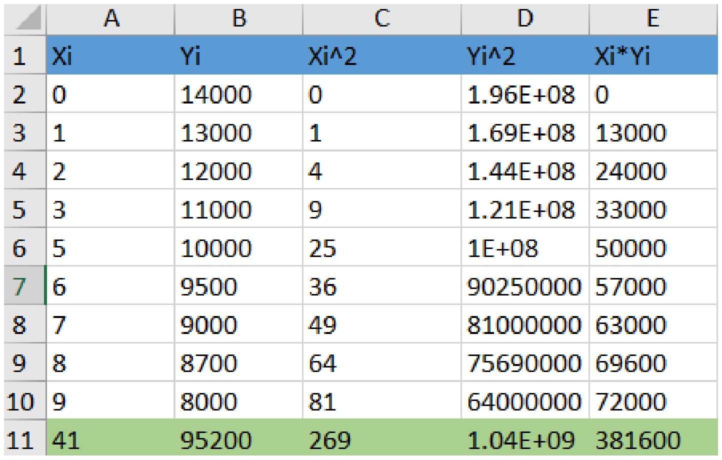
Ans: -With the outlier : equation : y = a + bx



	A	B	C	D	E
1	Xi	Yi	Xi^2	Yi^2	Xi*Yi
2	0	14000	0	1.96E+08	0
3	1	13000	1	1.69E+08	13000
4	2	12000	4	1.44E+08	24000
5	3	11000	9	1.21E+08	33000
6	4	1050	16	1102500	4200
7	5	10000	25	1E+08	50000
8	6	9500	36	90250000	57000
9	7	9000	49	81000000	63000
10	8	8700	64	75690000	69600
11	9	8000	81	64000000	72000
12	45	96250	285	1.04E+09	385800

$b = \frac{\sum_{i=1}^{10} X_i * Y_i - \frac{(\sum_{i=1}^{10} X_i)(\sum_{i=1}^{10} Y_i)}{10}}{\sum_{i=1}^{10} X_i^2 - \frac{(\sum_{i=1}^{10} X_i)^2}{10}}$ = -573.636 , $a = \bar{Y} - b * \bar{X}$ =9625- (-573.636)*(4.5) = 12206.36

-With no outlier :



	A	B	C	D	E
1	Xi	Yi	Xi^2	Yi^2	Xi*Yi
2	0	14000	0	1.96E+08	0
3	1	13000	1	1.69E+08	13000
4	2	12000	4	1.44E+08	24000
5	3	11000	9	1.21E+08	33000
6	5	10000	25	1E+08	50000
7	6	9500	36	90250000	57000
8	7	9000	49	81000000	63000
9	8	8700	64	75690000	69600
10	9	8000	81	64000000	72000
11	41	95200	269	1.04E+09	381600

$b = \frac{\sum_{i=1\ \forall\ i\neq 5}^{10} X_i * Y_i - \frac{(\sum_{i=1\ \forall\ i\neq 5}^{10} X_i)(\sum_{i=1\ \forall\ i\neq 5}^{10} Y_i)}{9}}{\sum_{i=1\ \forall\ i\neq 5}^{10} X_i^2 - \frac{(\sum_{i=1\ \forall\ i\neq 5}^{10} X_i)^2}{9}}$ = -633.514 , $a = \bar{Y} - b * \bar{X}$ =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 R^2 is not high enough.

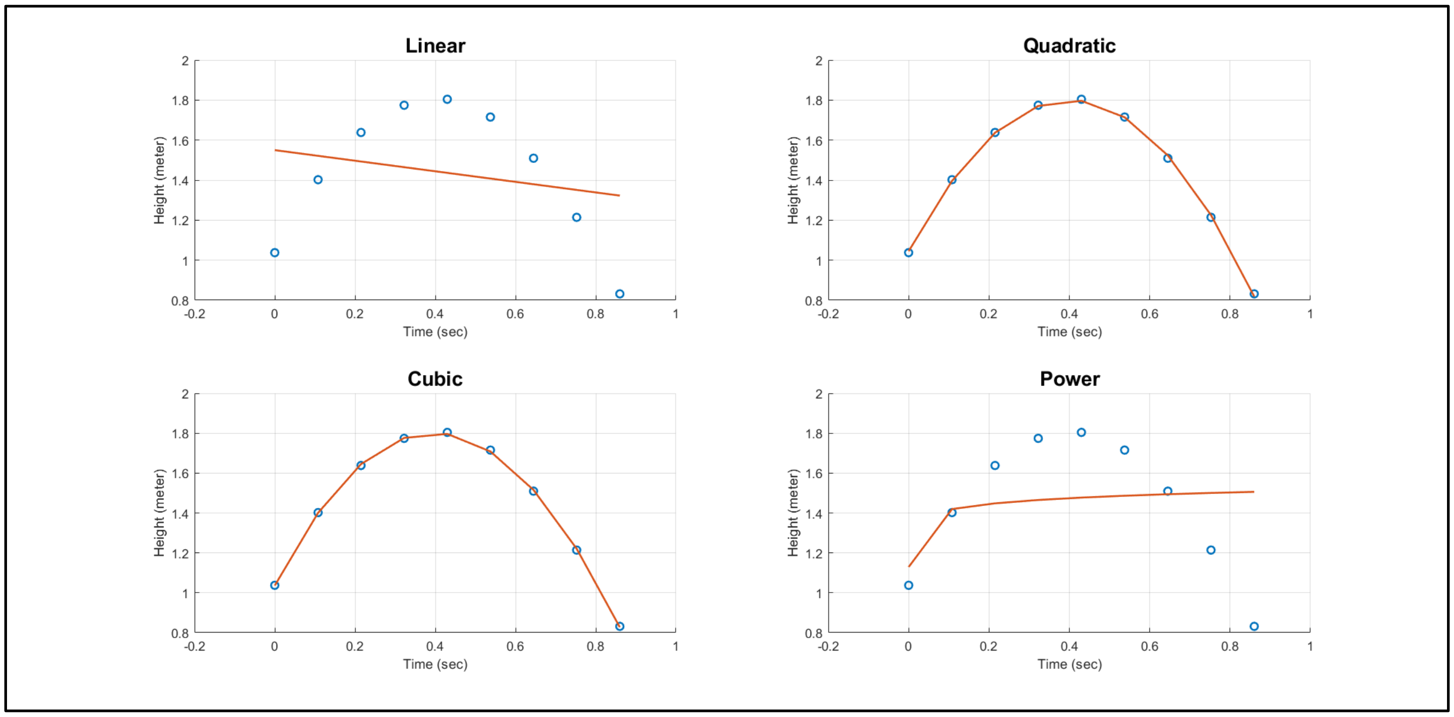
1.2. Throwing a Rubber Ball

2. The following data was obtained by throwing a rubber ball.

Time (sec)	Height (m)
0.0000	1.03754
0.1080	1.40205
0.2150	1.63806
0.3225	1.77412
0.4300	1.80392
0.5375	1.71522
0.6450	1.50942
0.7525	1.21410
0.8600	0.83173

a) Fit linear, quadratic, cubic, and power functions to the data. By comparing the values of R^2, determine the function that best fits the data.
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) Determine the maximum height of the ball (in meters).
e) With the model you selected in part (b), predict when the height of the ball is *at least* 1.5 meters.

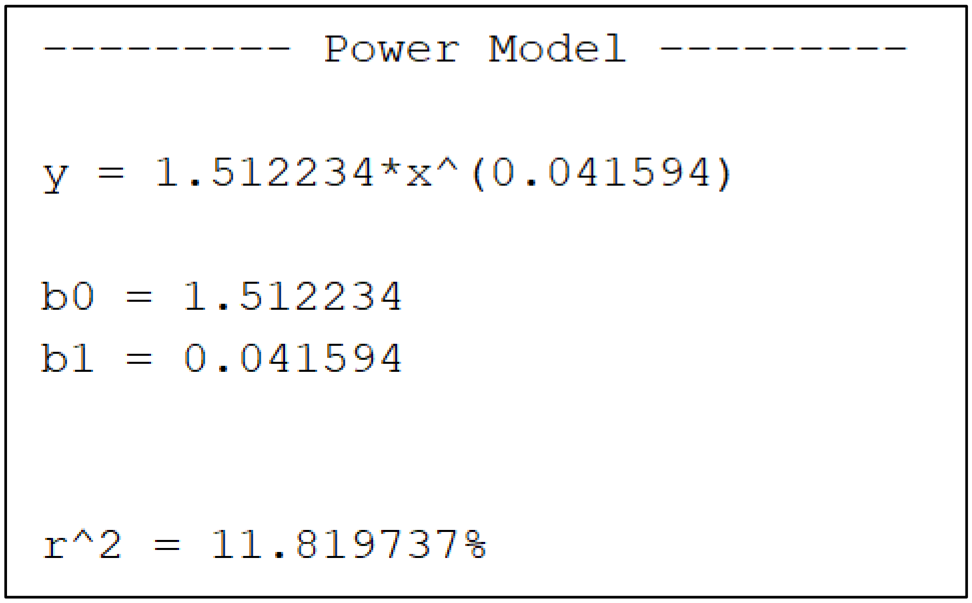
1.2.1. Results



```
--------- Linear Model ----------

y = 1.549869 + -0.264220*x

b0 = 1.549869
b1 = -0.264220


r^2 = 5.102809%
```

```
--------- Quadratic Model ---------

y = 1.044973 + 3.758278*x + -4.676403*x^2

b0 = 1.044973
b1 = 3.758278
b2 = -4.676403


r^2 = 99.930731%
```

```
--------- Cubic Model ---------

y = 1.036098 + 3.935985*x + -5.224364*x^2 + 0.424784*x^3

b0 = 1.036098
b1 = 3.935985
b2 = -5.224364
b3 = 0.424784


r^2 = 99.972635%
```



```
---------- Power Model ----------

y = 1.512234*x^(0.041594)

b0 = 1.512234
b1 = 0.041594


r^2 = 11.819737%
```

1.2.2. Comments

Linear: $y = 1.55 - 0.26x$ $R^2 = 5.1\%$	Quadratic: $y = 1.045 + 3.758x - 4.7x^2$ $R^2 = 99.93\%$
Cubic: $y = 1.04 + 3.94x - 5.2x^2 + 0.4x^3$ $R^2 = 99.97\%$	Power: $y = 1.5x^{0.04}$ $R^2 = 11.82\%$

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

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \dots \end{bmatrix} \qquad \text{output matrix}$$

$$X = \left[\begin{array}{cc} ones & x \\ 1 & 0 \\ 1 & 0.1080 \\ 1 & 0.2150 \\ \dots & \dots \end{array}\right] \qquad \text{independent variables matrix}$$

$$b = (X^T X)^{-1} X^T Y \qquad \text{parameters matrix}$$

And for power model:

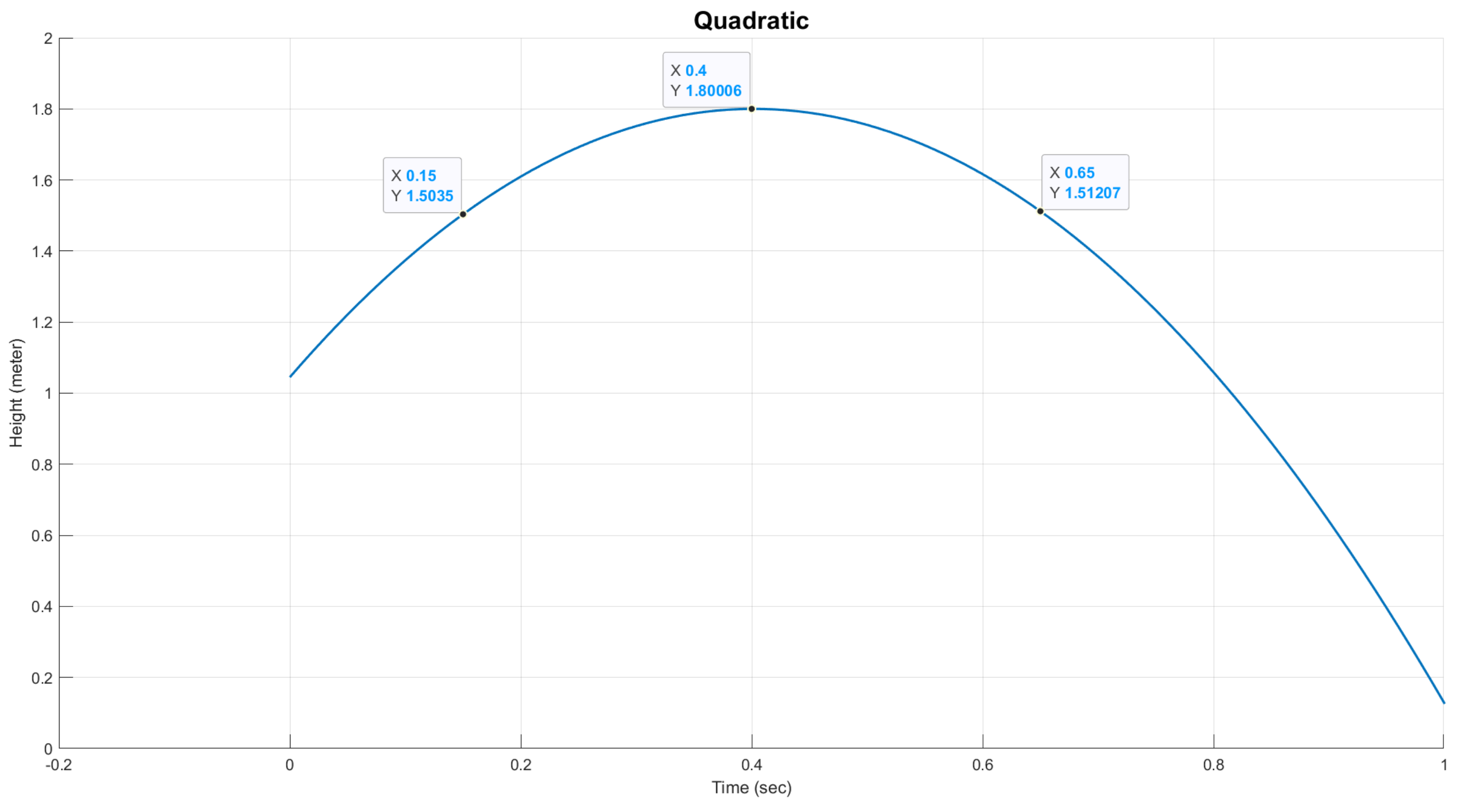
$$\ln(y) = \mathrm{b}_0 + \mathrm{b}_1 \ln(x) \qquad X = [ones \quad \ln(x)] \qquad b = (X^T X)^{-1} X^T Y$$

By comparing the R^2 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:



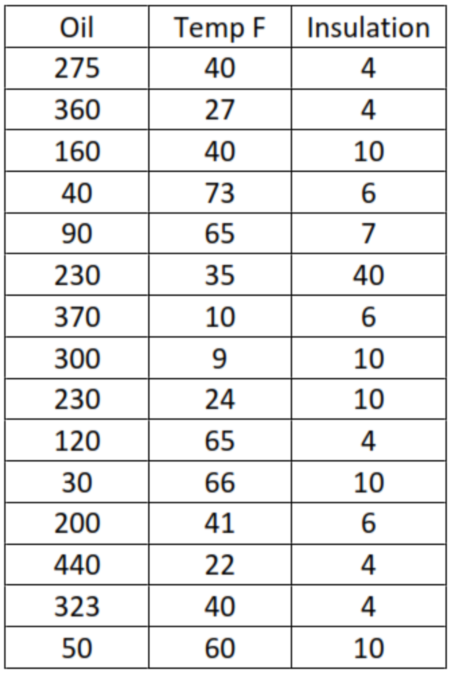
By sweeping over a range of time and calculate the height from the model

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

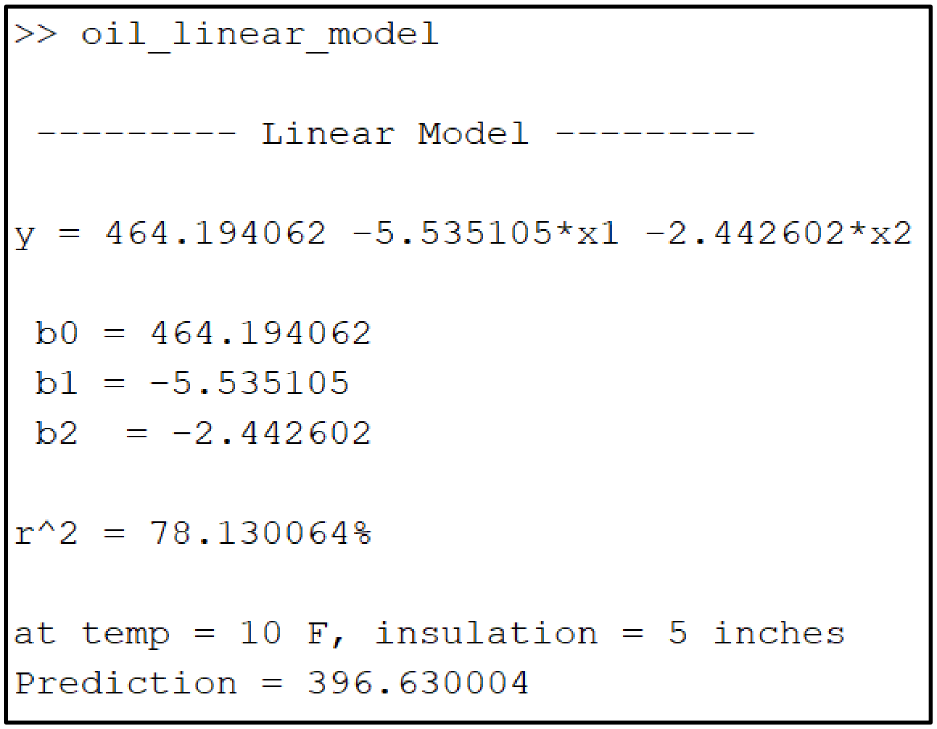
1.3. Heating Oil Consumption Model

3. Develop a model for estimating heating oil used for a single-family home in the month of January based on average temperature and amount of insulation in inches.



Oil	Temp F	Insulation
275	40	4
360	27	4
160	40	10
40	73	6
90	65	7
230	35	40
370	10	6
300	9	10
230	24	10
120	65	4
30	66	10
200	41	6
440	22	4
323	40	4
50	60	10

1.3.1. Linear Model Results



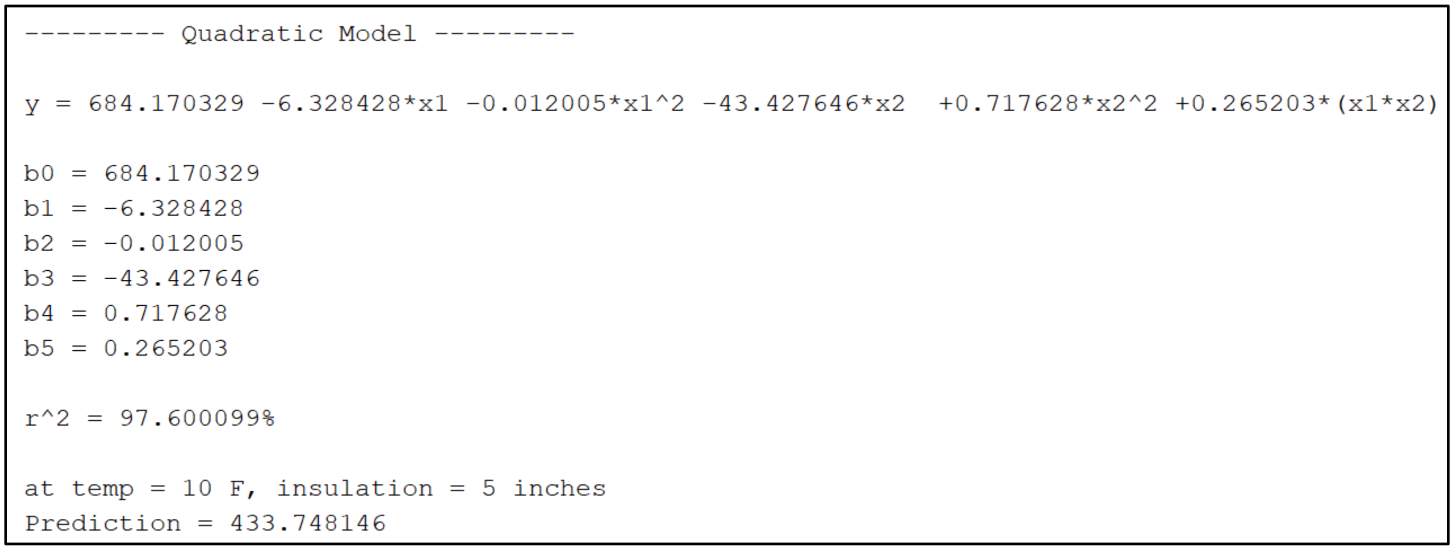
```
>> oil_linear_model

 --------- Linear Model ---------

y = 464.194062 -5.535105*x1 -2.442602*x2

 b0 = 464.194062
 b1 = -5.535105
 b2  = -2.442602

r^2 = 78.130064%

at temp = 10 F, insulation = 5 inches
Prediction = 396.630004
```

The final model equation is

$$y = 464.2 - 5.54x_1 - 2.44x_2$$

1.3.2. Quadratic Model Results

only difference is the added parameters



```
--------- Quadratic Model ---------

y = 684.170329 -6.328428*x1 -0.012005*x1^2 -43.427646*x2  +0.717628*x2^2 +0.265203*(x1*x2)

b0 = 684.170329
b1 = -6.328428
b2 = -0.012005
b3 = -43.427646
b4 = 0.717628
b5 = 0.265203

r^2 = 97.600099%

at temp = 10 F, insulation = 5 inches
Prediction = 433.748146
```

The final model equation is

$$y = 684.2 - 6.33x_1 - 0.012x_1^2 - 43.43x_2 + 0.7.2x_2^2 + 0.27x_1x_2$$

1.3.2. Comments

Used the matrix method to calculate the parameters. For linear model the regression equation is in the form $y = b_0 + b_1x_1 + b_2x_2$

$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \dots \end{bmatrix}$ output matrix

$X = \begin{bmatrix} ones & x_1 & x_2 \\ 1 & 40 & 4 \\ 1 & 27 & 4 \\ 1 & 40 & 10 \\ \dots & \dots & \dots \end{bmatrix}$ independent variables matrix

$b = (X^TX)^{-1}X^TY$ parameters matrix

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

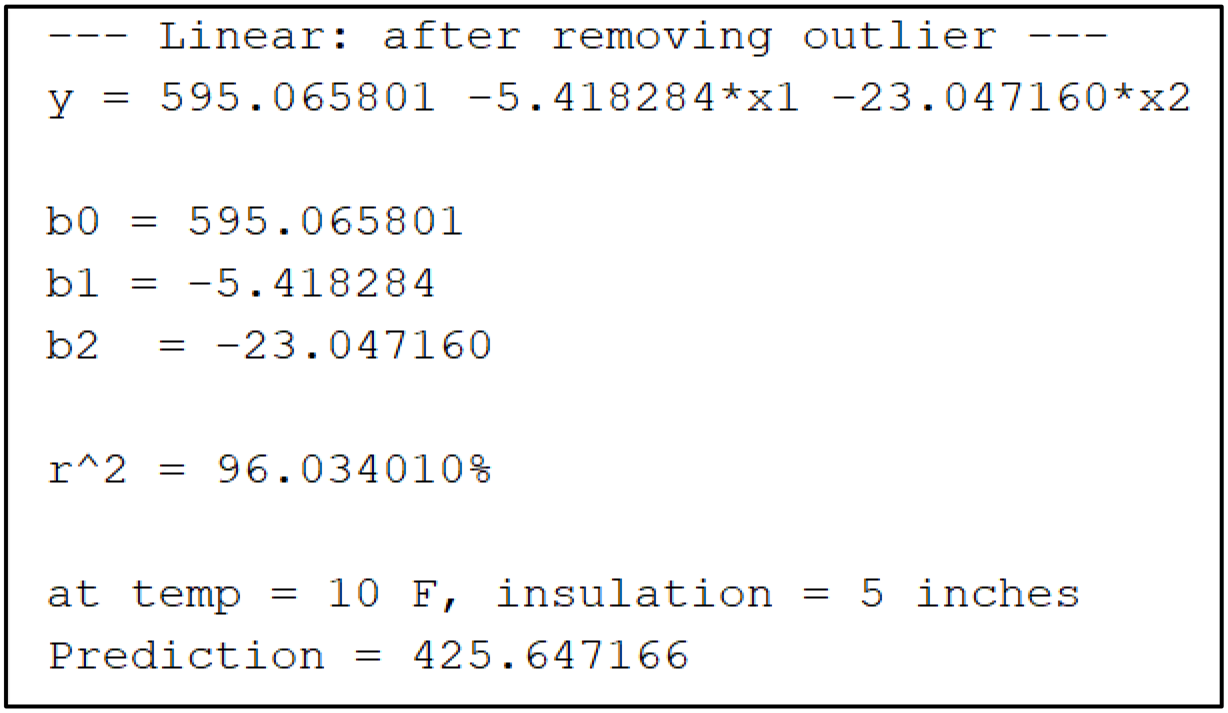
Linear: $R^2 = 78.13\%$

Quadratic: $R^2 = 97.6\%$

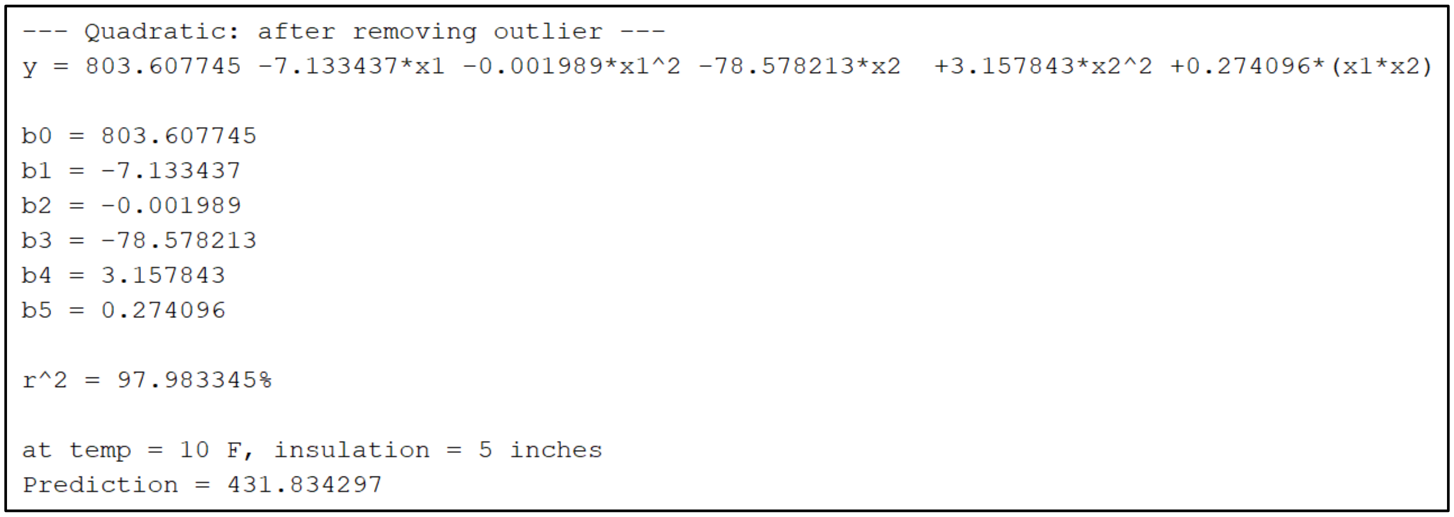
There is a large gap of improvement in the coefficient of determinism of quadratic over linear. Hence, the quadratic model is more recommended in this case.

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.



```
--- Linear: after removing outlier ---
y = 595.065801 -5.418284*x1 -23.047160*x2

b0 = 595.065801
b1 = -5.418284
b2  = -23.047160

r^2 = 96.034010%

at temp = 10 F, insulation = 5 inches
Prediction = 425.647166
```



```
--- Quadratic: after removing outlier ---
y = 803.607745 -7.133437*x1 -0.001989*x1^2 -78.578213*x2  +3.157843*x2^2 +0.274096*(x1*x2)

b0 = 803.607745
b1 = -7.133437
b2 = -0.001989
b3 = -78.578213
b4 = 3.157843
b5 = 0.274096

r^2 = 97.983345%

at temp = 10 F, insulation = 5 inches
Prediction = 431.834297
```

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

2. Part 2

2.1 Results

Table 1: Final Results

Classifier		Features					
		DCT		PCA		ICA	
		Accuracy	Training Time	Accuracy	Training Time	Accuracy	Training Time
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

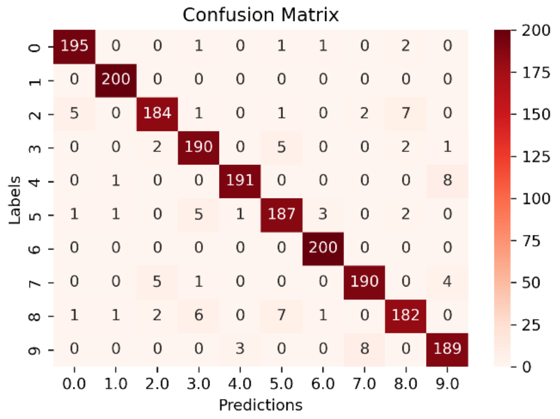


Figure 1: K-means, DCT, 32 Clusters

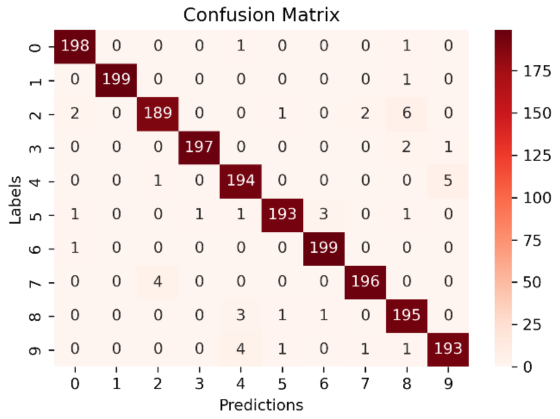


Figure 2: SVM, PCA, rbf (nonlinear)

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[ ]:
```