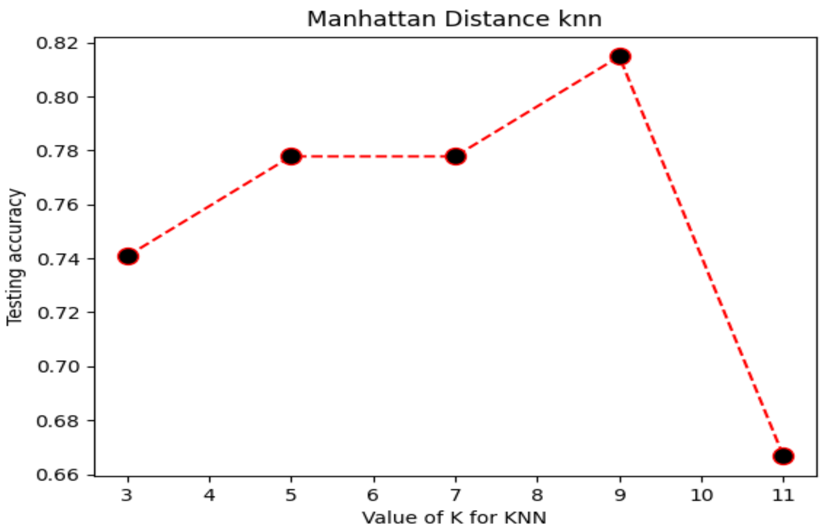
KNN variations(code is in knn_variations.py)

Question 1 Manhattan distance

1.



2.

Manhattan distance accuracy: 0.9230769230769231

3.

Confusion matrix

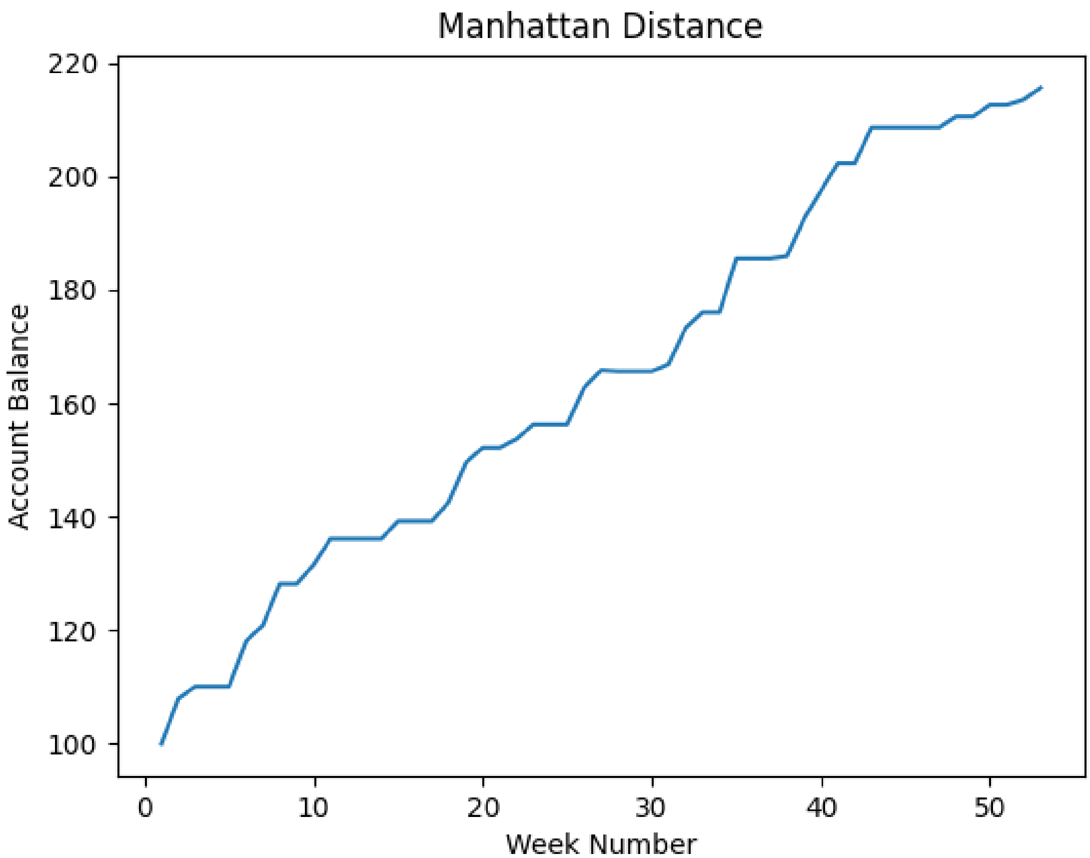
[[25 0]

 [4 23]]

4. The k in regular knn is 9 . I get the same k for manhattan distance as well.

5.True positive rate for year 2: 1.0, true negative rate for year2 : 0.8518518518518519

6.

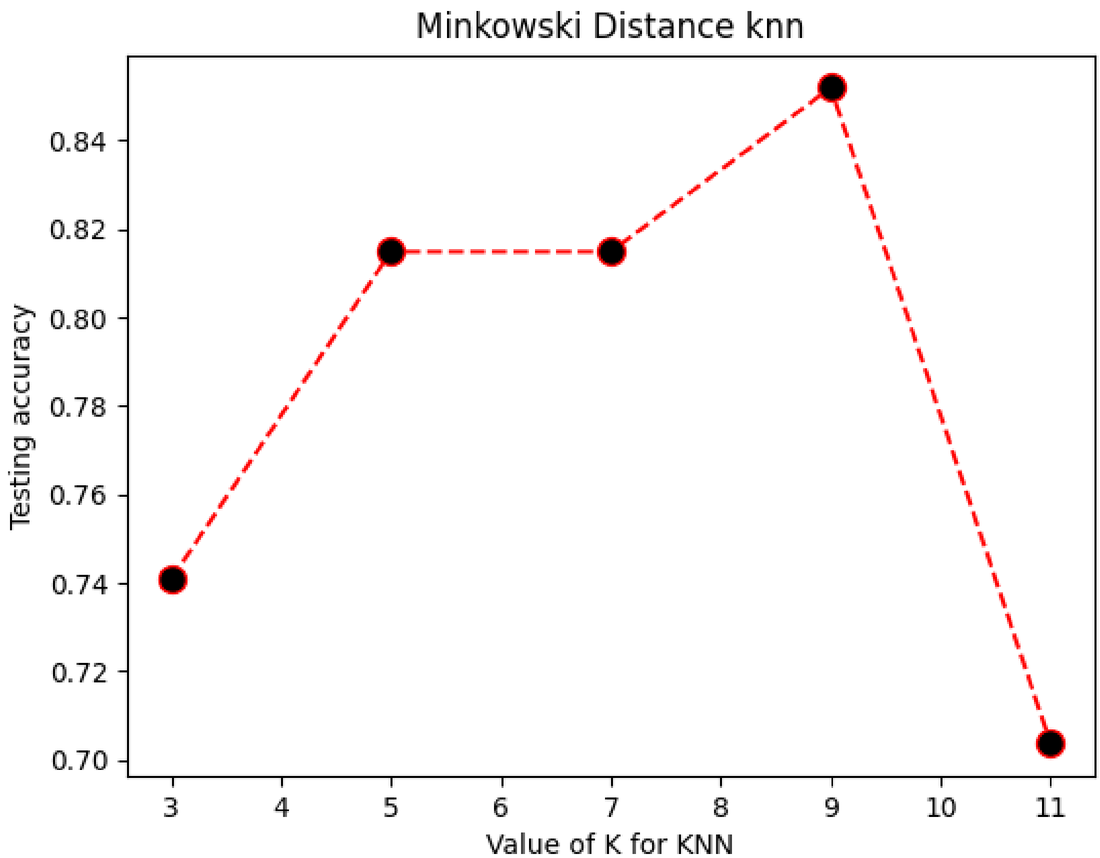


Yes this results in a larger amount at the end of the year compared to buy hold.

7. Yes this gives higher accuracy than euclidean distance(accuracy 0.904) for predicting labels

Question 2 (Minkowski)

1.

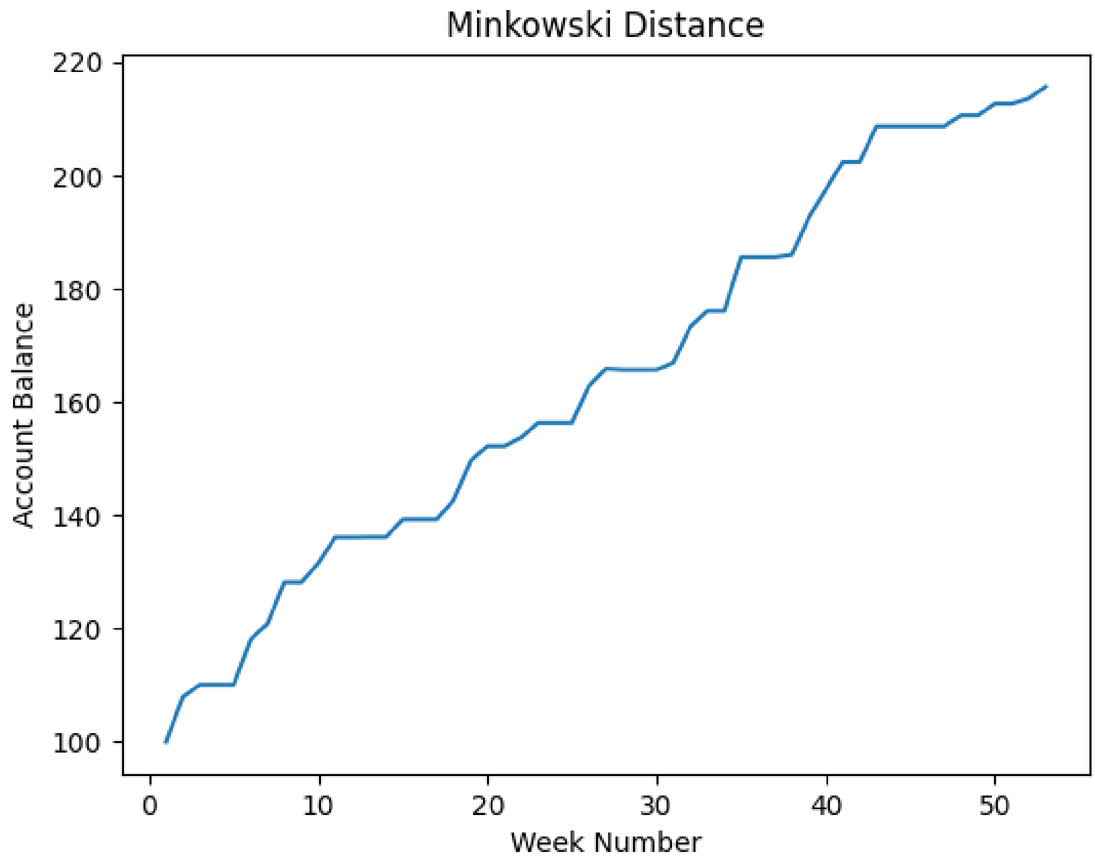


2.
Minkowski distance accuracy: 0.9038461538461539
3.
[[25 0]
 [5 22]]

4. It has the same k as regular knn.

5.
True positive rate for year 2: 1.0, true negative rate for year2 : 0.8148148148148148

6.



Yes this results in a larger amount at the end of the year compared to buy hold.

7. This has the accuracy as original knn

Question 3 (Centroid)

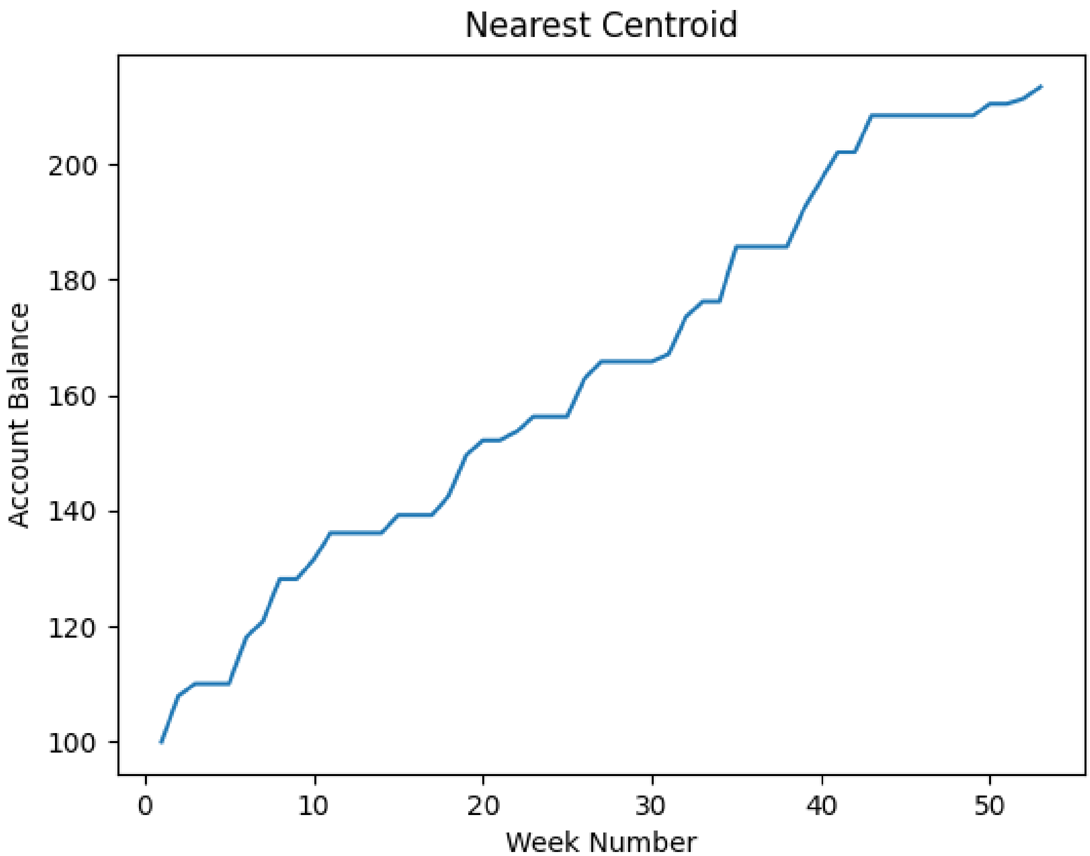
1.

	Mean,	Median
Green centroid:	(0.8205600000000001,	2.4777951797895463)
Red centroid:	(-0.9415347826086957,	2.7742643305431742)

Red has a bigger sphere.

2.
True positive rate for year 2: 0.92, true negative rate for year2 : 0.8888888888888888

3.

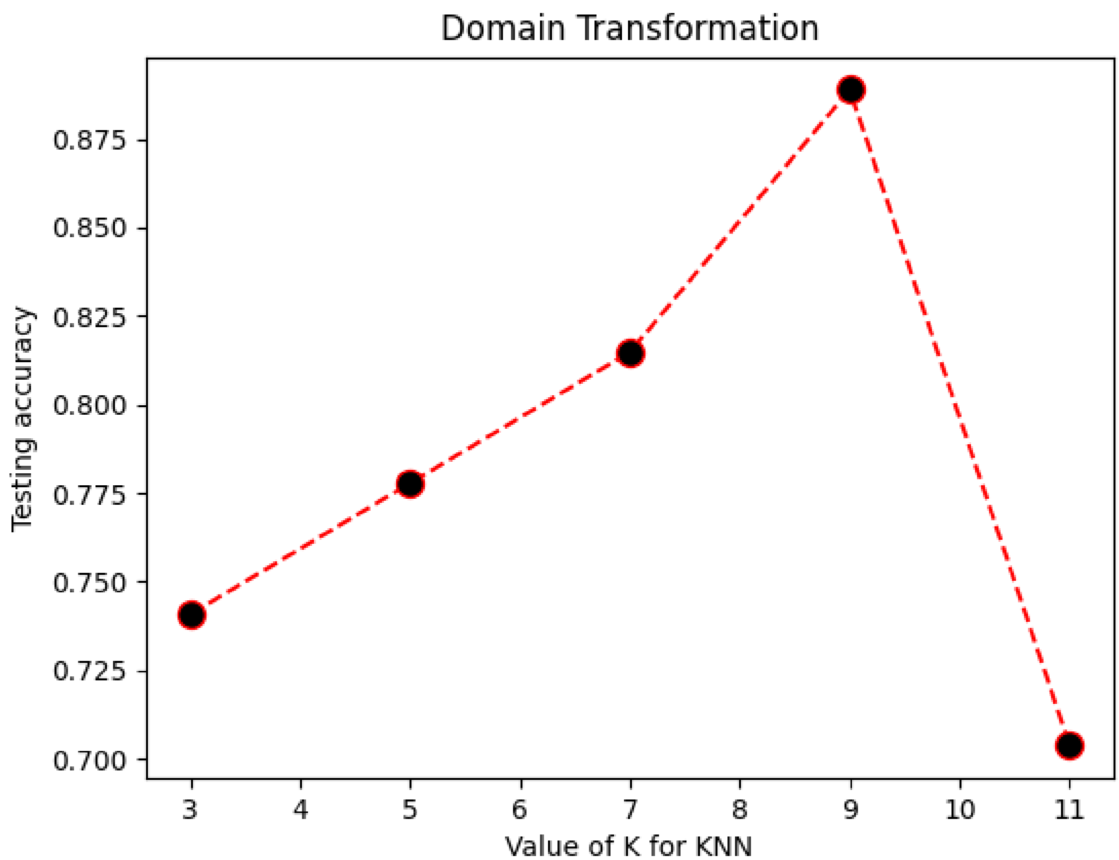


Yes this performs better than buy and hold strategy.

4. This gives the same accuracy as the regular k-nn.

Question 4 (Domain Transformation)

1.



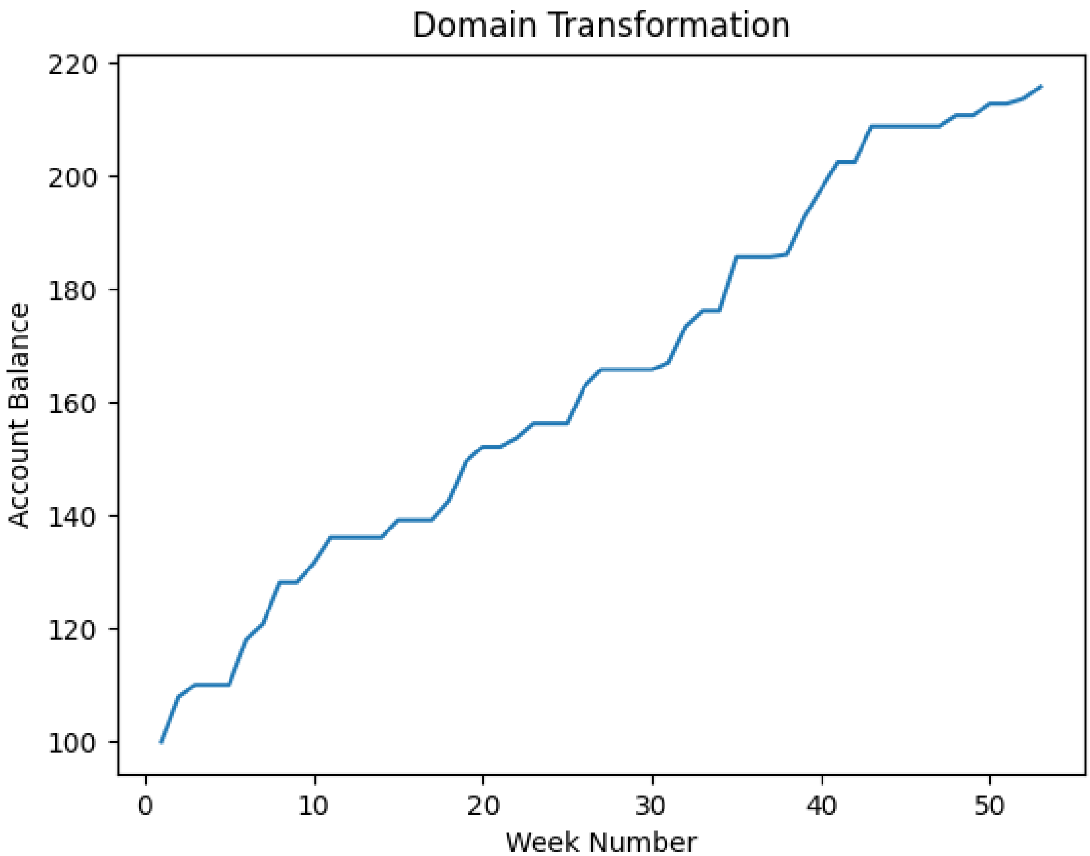
2.
Domain transformation accuracy: 0.9038461538461539

3.
[[24 1]
 [4 23]]

4.
No I have the same value as the original knn

5.
True positive rate for year 2: 0.96, true negative rate for year2 : 0.8518518518518519

6.

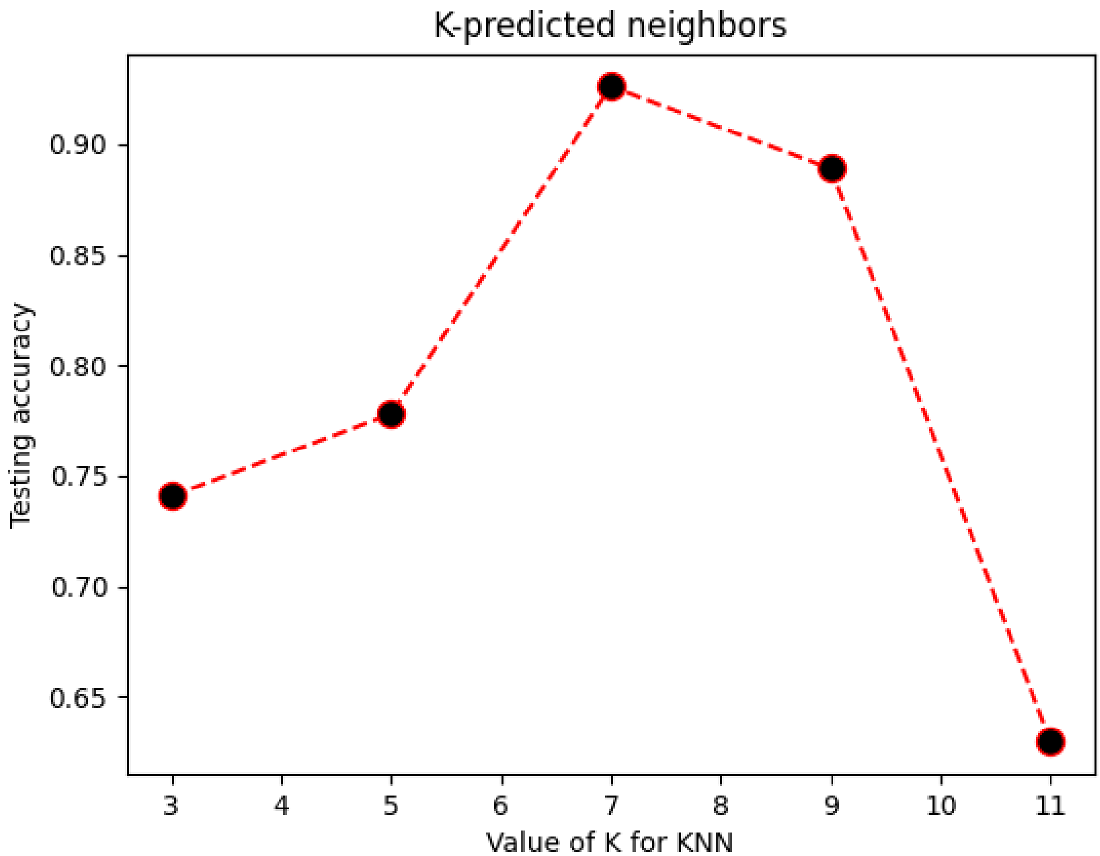


Yes this performs higher than the buy hold strategy.

7.
No it gives about the same performance as the original method.

Question 5

1.



2.

K-predicted neighbors accuracy: 0.7884615384615384

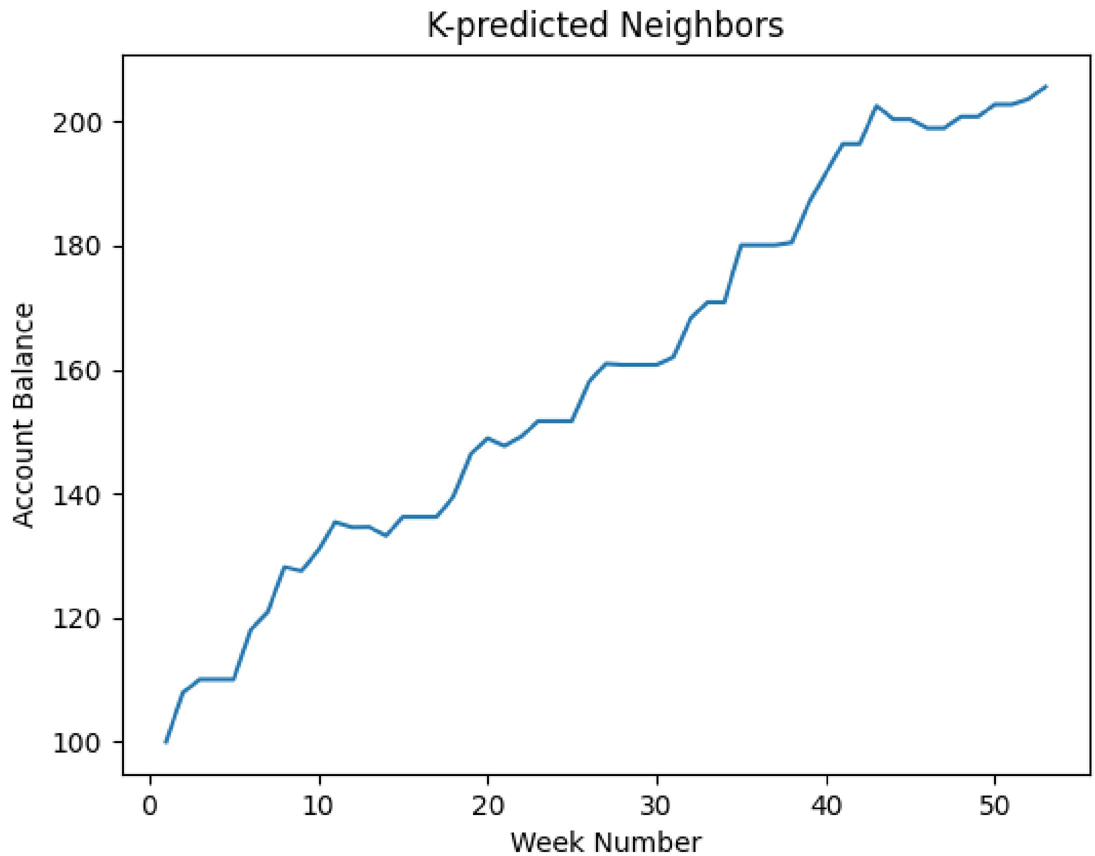
3.

[[25 0]
 [11 16]]

4.
 Yes the best value was 7.

5. True positive rate for year 2: 1.0, true negative rate for year2 : 0.5925925925925926

6.



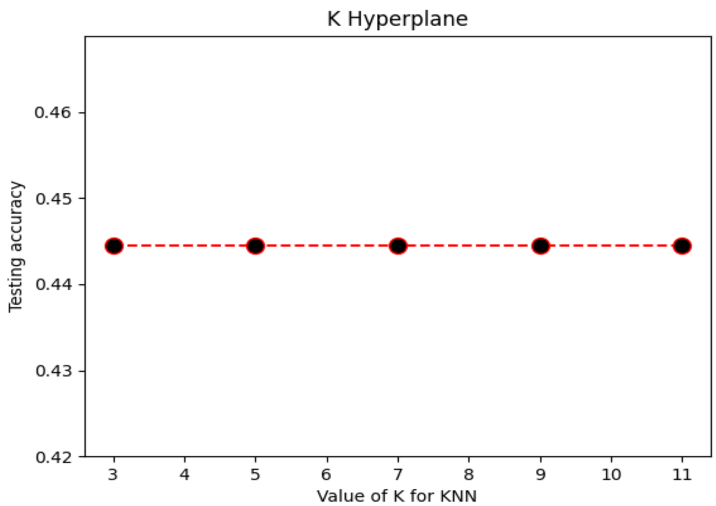
This is more than the buy and hold strategy

7.
No it performs worse than the original knn.

(Continued on next page)

Question 6 k-hyperplanes method.

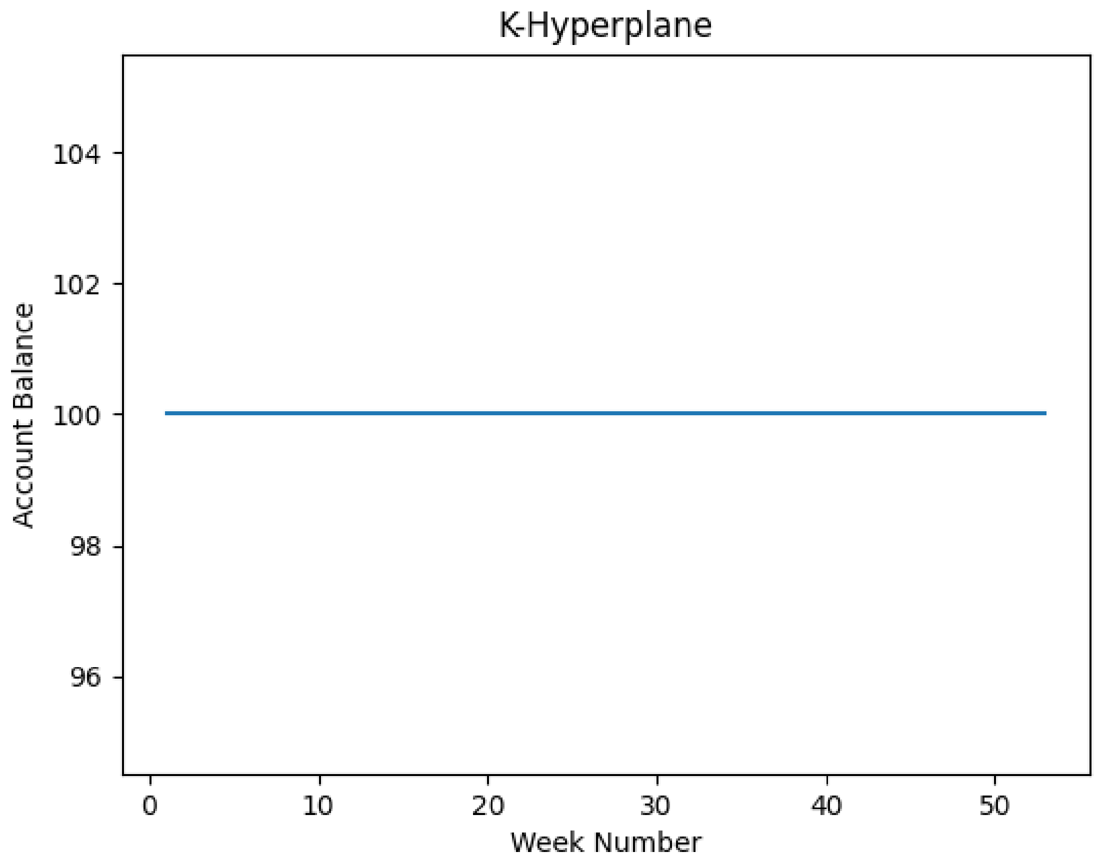
1.



2.

True positive rate for year 2: 1.0, true negative rate for year2 : 0.5925925925925926

3.



The performance is worse than buy and hold strategy.

4. This performs far worse than the regular knn.

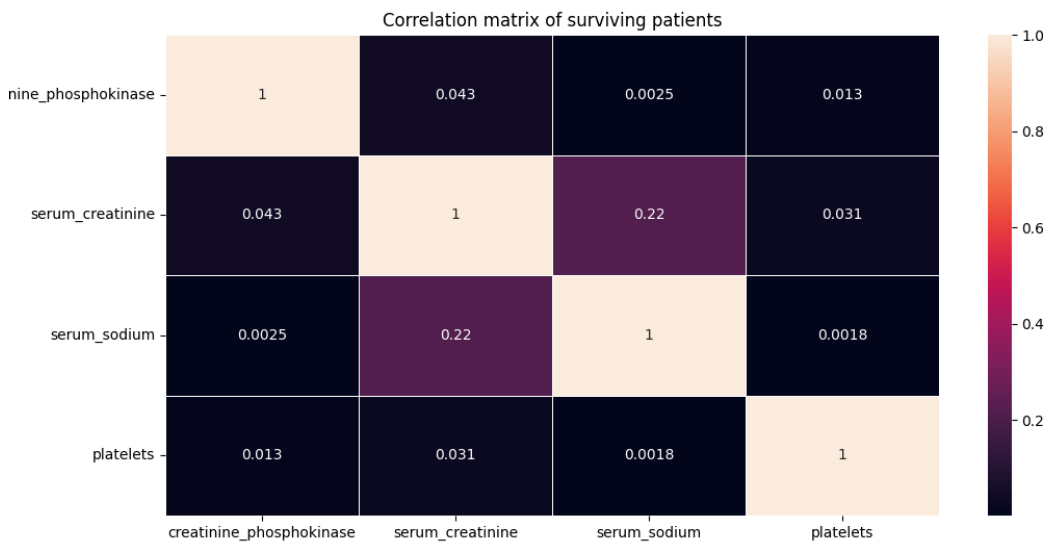
7.

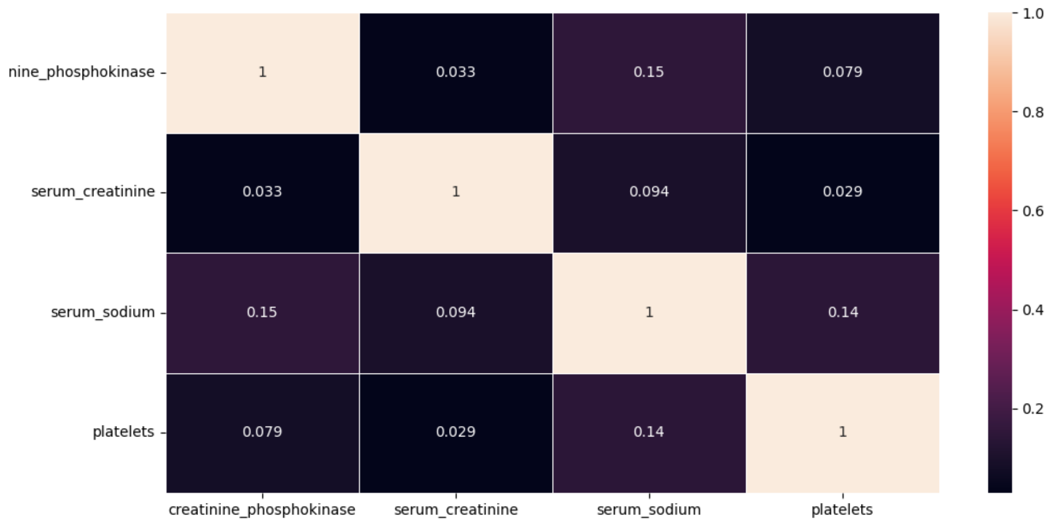
Method	Best K	Accuracy	Amount
B	NA		180
KNN (Euclidean)	9	0.903846153846153	215.87
Manhattan	9	0.923076923076923	215.87
Minkowski	9	0.903846153846153	215.87
Centroid	NA	0.903846153846153	213.4
Doma Transformation	9	0.903846153846153	215.87

K-predicted	7	0.788461538461538 4	205.63
K-hyperplane	3	0.519230769230769 3	100

Linear Models (all the code is in linear_models.py)

1.
    1. Done in Code
    2.



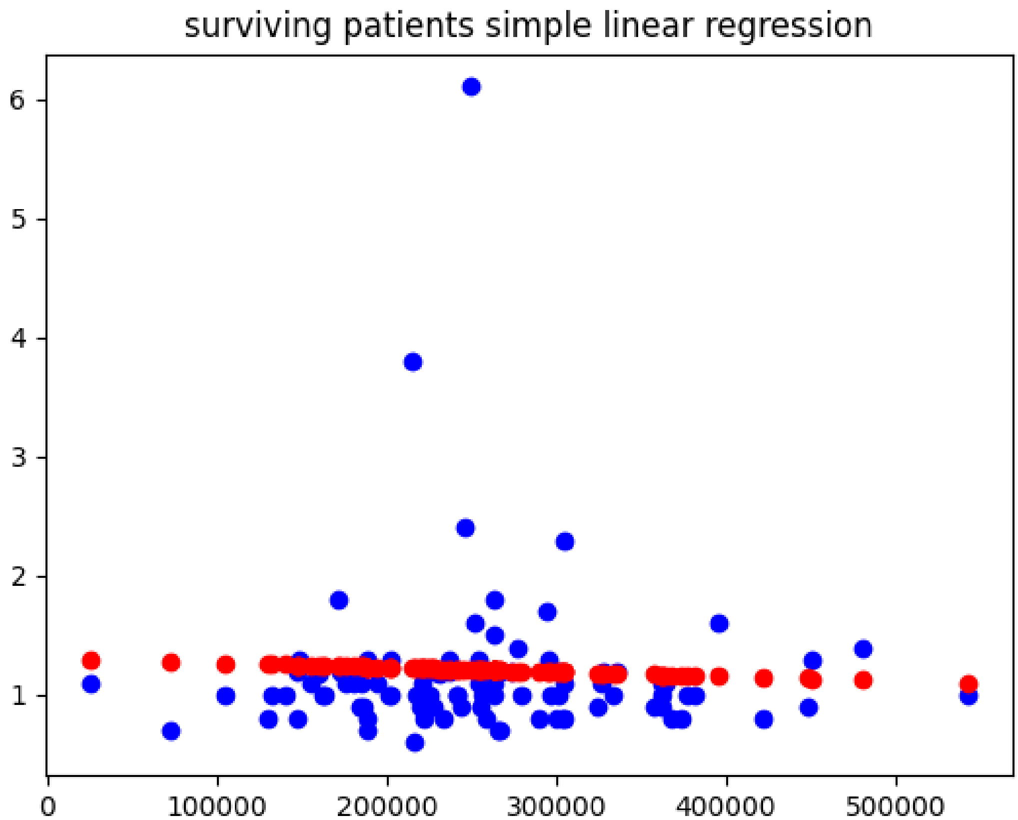


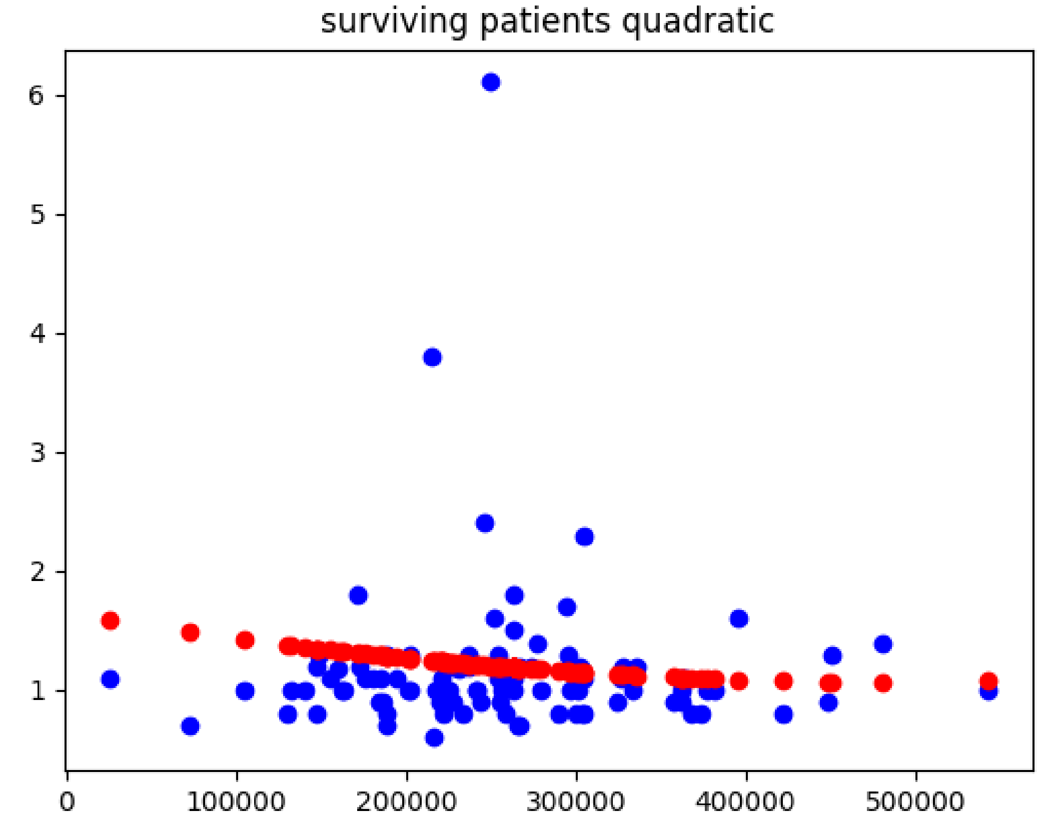
3.

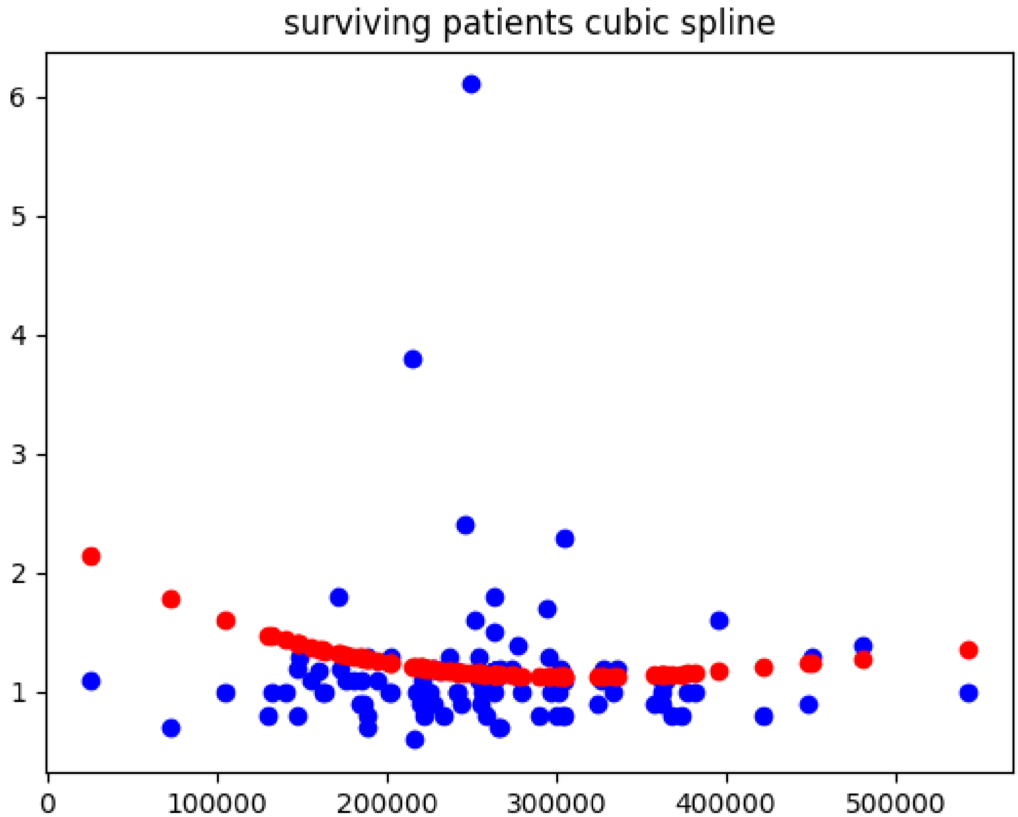
a) Serum sodium and serum creatinine have the highest correlation for surviving patients
b) Platelets and Serum Sodium have the lowest correlation for surviving patients
c) Serum Sodium and Creatinine Phosphokinase have the highest correlation for deceased patients
d) Platelets and Serum Creatinine have the lowest correlation for deceased patients
e) No it is not the same for both cases

2.

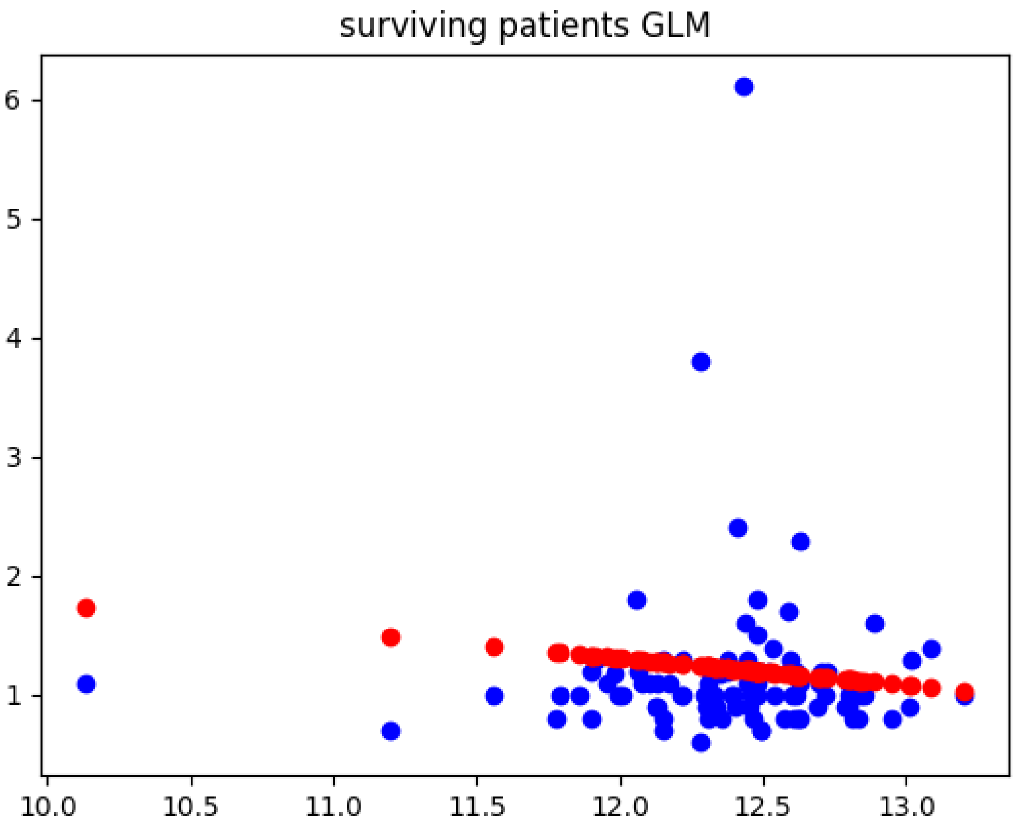
For surviving patients
weights for simple linear regression: [-3.75060955e-07 1.30664356e+00]
Loss function (SSE for model) 40.59
weights for quadratic: [2.50443057e-12 -2.41228012e-06 1.64949822e+00]
Loss function (SSE for model) 41.74
weights for cubic spline: [-1.53189090e-17 2.23002154e-11 -9.43465465e-06 2.36554701e+00]
Loss function (SSE for model) 44.34
weights for GLM: [-0.22619491 4.02287064]
Loss function (SSE for model) 41.75
weights for GLM2: [-0.15188661 1.97936928]
Loss function (SSE for model) 41.53

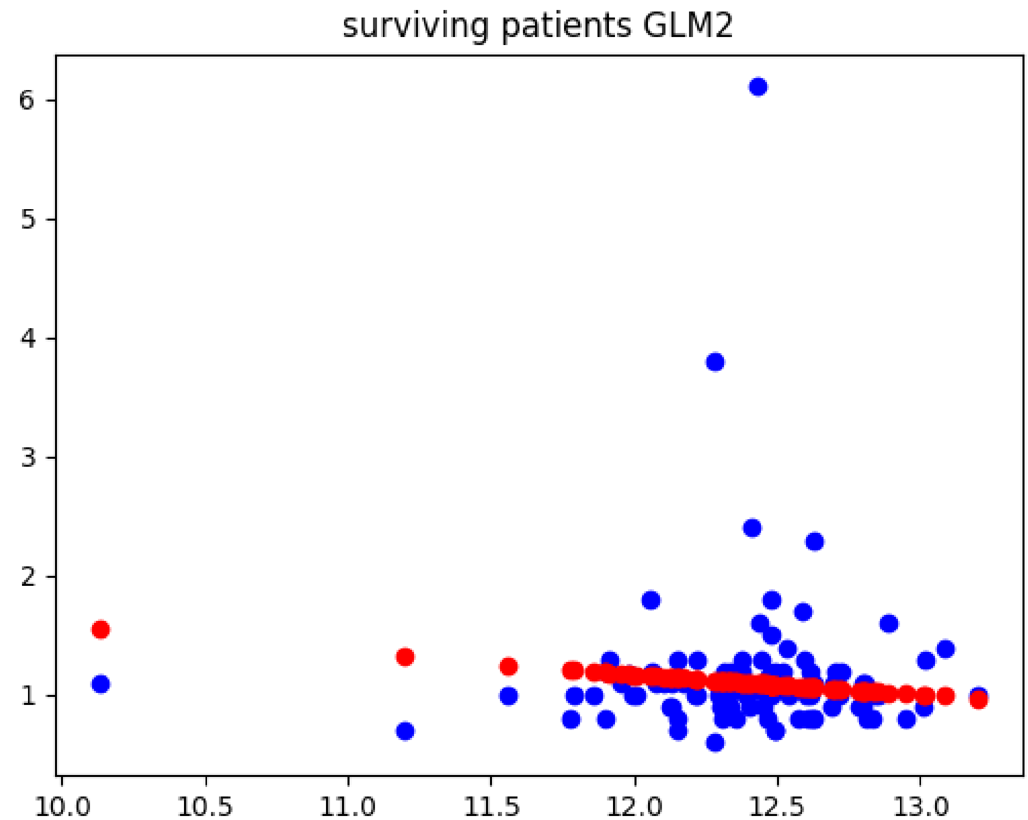

surviving patients simple linear regression
6
5
4
3
2
1
0
100000
200000
300000
400000
500000


surviving patients quadratic
6
5
4
3
2
1
0
100000
200000
300000
400000
500000

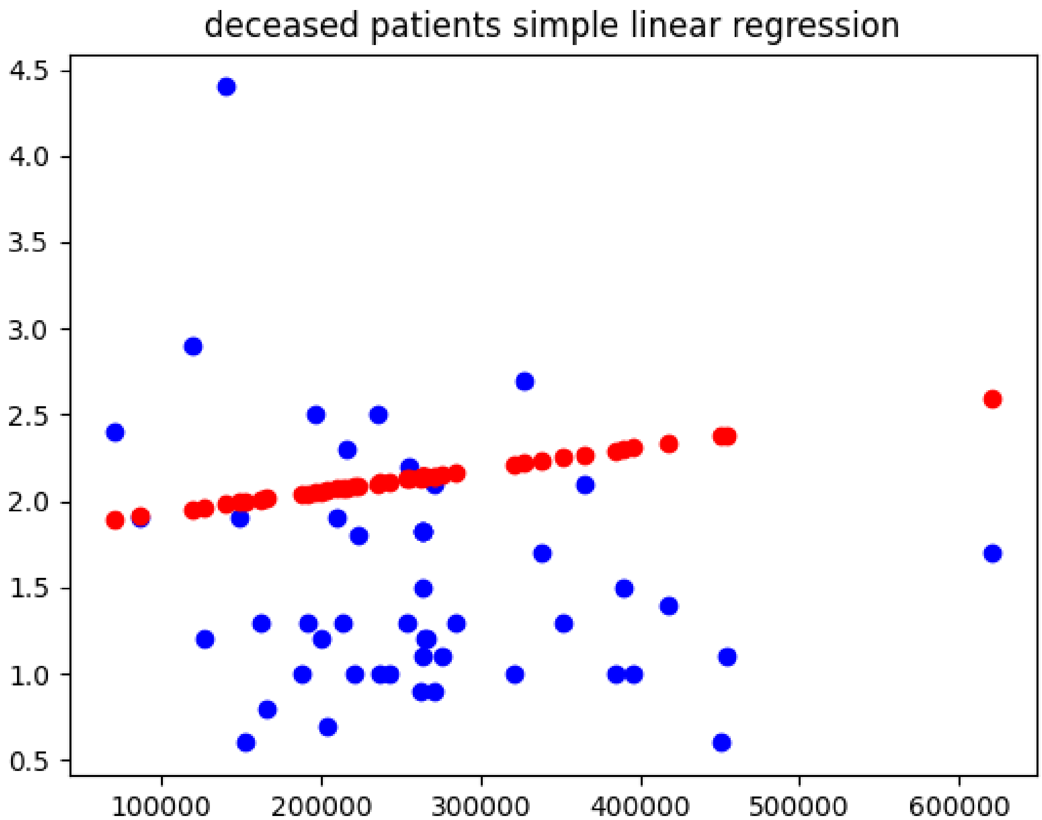


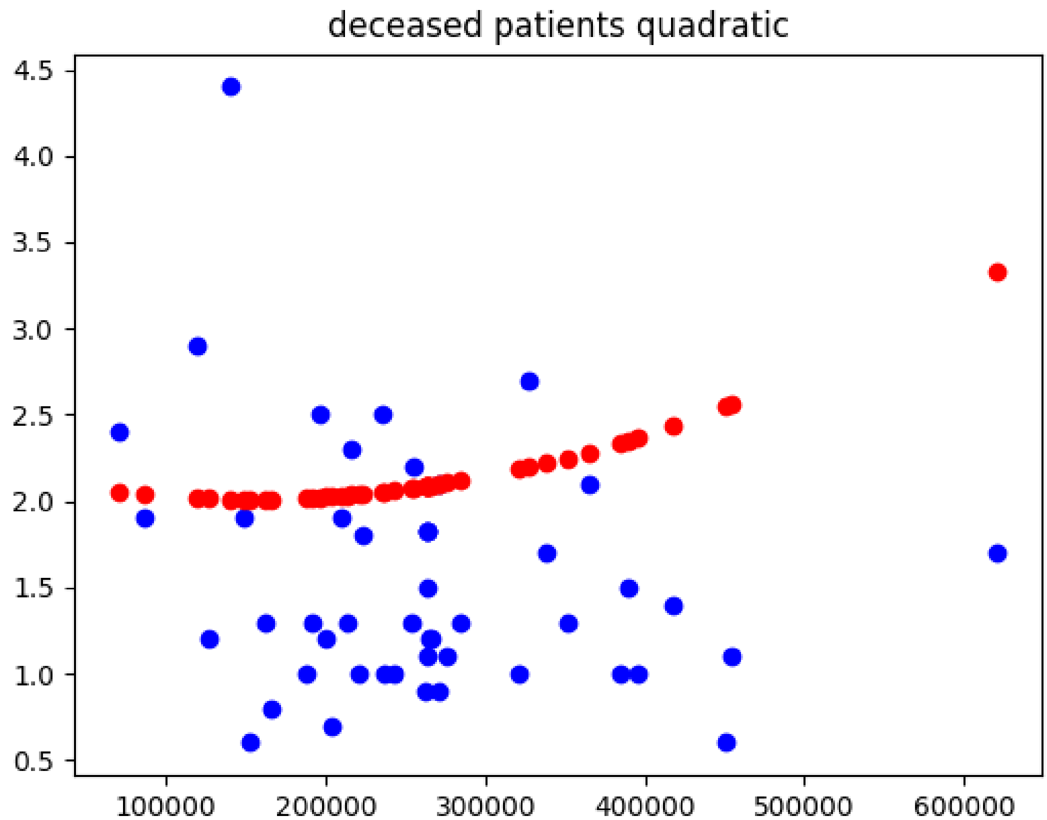
surviving patients cubic spline
6
5
4
3
2
1
0
100000
200000
300000
400000
500000

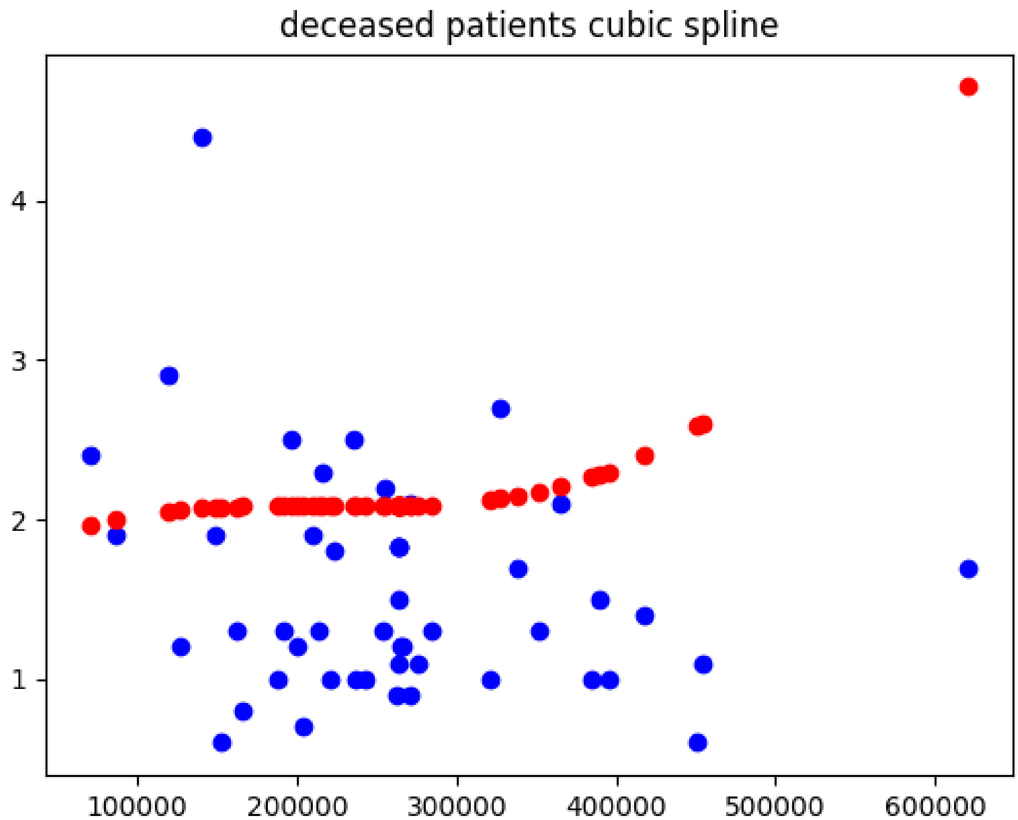

surviving patients GLM
6
5
4
3
2
1
10.0
10.5
11.0
11.5
12.0
12.5
13.0

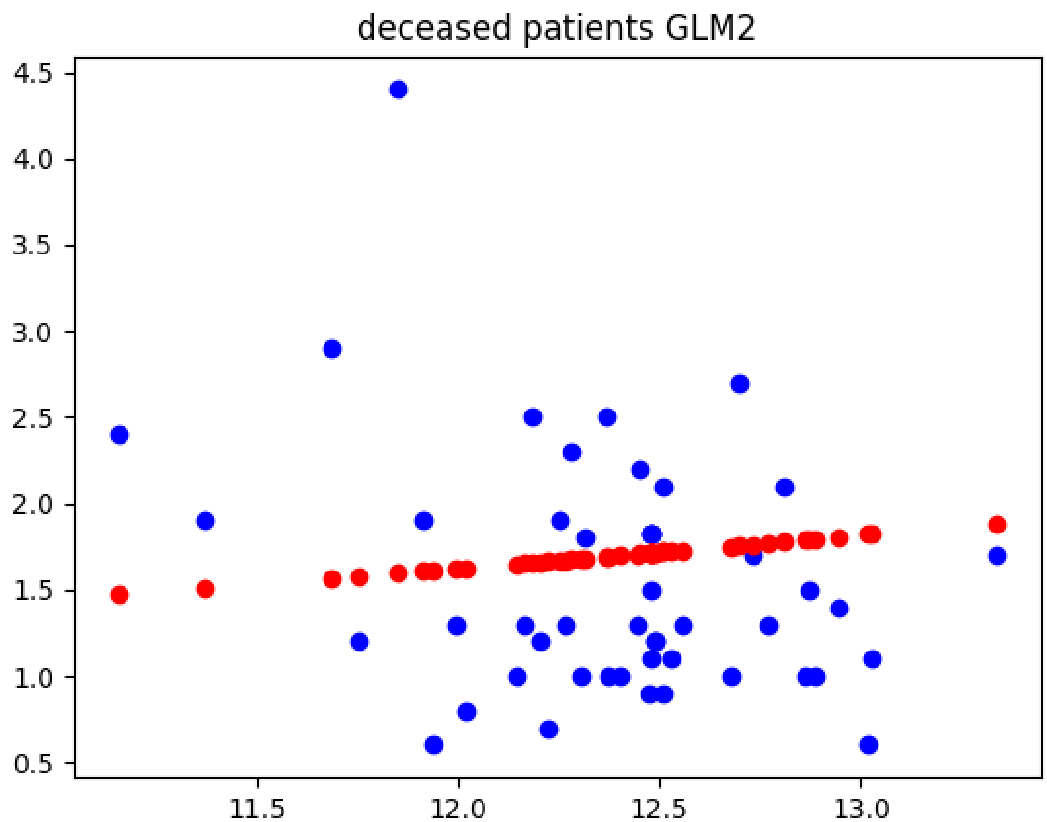


For deceased patients
weights for simple linear regression: [1.28337038e-06 1.79975250e+00]
Loss function (SSE for model) 43.09
weights for quadratic: [5.98958549e-12 -1.81701269e-06 2.14904195e+00]
Loss function (SSE for model) 44.8
weights for cubic spline: [4.02046103e-17 -2.61588949e-11 5.61565487e-06
1.68505377e+00]
Loss function (SSE for model) 51.49
weights for GLM: [0.25466632 -1.02171458]
Loss function (SSE for model) 42.73
weights for GLM2: [0.11258057 -0.86662119]
Loss function (SSE for model) 26.54


deceased patients simple linear regression
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
100000
200000
300000
400000
500000
600000


deceased patients quadratic
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
100000
200000
300000
400000
500000
600000


deceased patients cubic spline
1
2
3
4
100000
200000
300000
400000
500000
600000



3.

Model	SSE (death event=0)	(death event=1)
y = ax + b	40.59	43.09
y = ax2 + bx + c	41.74	44.8
y = ax3 + bx2 + cx + d	44.34	51.49
y = a log x + b	41.75	42.73
log y = a log x + b	41.53	26.54

a)Model 1(y= ax+b) has the smallest SSE for surviving patients and Model 5(log y = a log x + b) has the smallest SSE for surviving patients
b)Model 3 (y= ax3 + bx2 + cx + d) has the largest SSE for both surviving and deceased patients