Homework3

March 6, 2022

[1]:
```
import matplotlib.pyplot as plt
# ^^^ pyforest auto-imports - don't write above this line
import numpy as np
import pandas as pd
```

[2]:
```
data=pd.read_csv('G:
↪\Documents\ReferenceBooks\MachineLearning\Rohban\Homework\HW3\TrainPreprocessed.
↪csv')
data_test = pd.read_csv('G:
↪\Documents\ReferenceBooks\MachineLearning\Rohban\Homework\HW3\TestPreprocessed.
↪csv')
```

[3]:
```
data
```

[3]:

	BsmtQual	1stFlrSF	MasVnrArea	SaleCondition_Partial	\
0	-0.623056	-0.793162	0.511243	-0.30589	
1	-0.623056	0.257052	-0.574214	-0.30589	
2	-0.623056	-0.627611	0.322950	-0.30589	
3	0.865071	-0.521555	-0.574214	-0.30589	
4	-0.623056	-0.045596	1.364102	-0.30589	
…	…	…	…	…	
1455	-0.623056	-0.542249	-0.574214	-0.30589	
1456	-0.623056	2.354894	0.084814	-0.30589	
1457	0.865071	0.065634	-0.574214	-0.30589	
1458	0.865071	-0.218907	-0.574214	-0.30589	
1459	0.865071	0.241532	-0.574214	-0.30589	

	GarageType_Detchd	OverallQual	GarageYrBlt	FullBath	TotalBsmtSF	\
0	-0.600353	0.651256	1.020807	0.789470	-0.459145	
1	-0.600353	-0.071812	-0.104447	0.789470	0.466305	
2	-0.600353	0.651256	0.937455	0.789470	-0.313261	
3	1.664545	0.651256	0.812427	-1.025689	-0.687089	
4	-0.600353	1.374324	0.895779	0.789470	0.199611	
…	…	…	…	…	…	
1455	-0.600353	-0.071812	0.854103	0.789470	-0.238040	
1456	-0.600353	-0.071812	-0.021095	0.789470	1.104547	
1457	-0.600353	0.651256	-1.563110	0.789470	0.215567	

```
1458          -0.600353    -0.794879    -1.188025 -1.025689     0.046889
1459          -0.600353    -0.794879    -0.562884 -1.025689     0.452629

      YearBuilt  …  BsmtFinSF1  GarageFinish  Neighborhood_NridgHt  \
0      1.050634  …    0.575228     -0.223588             -0.235877
1      0.156680  …    1.171591     -0.223588             -0.235877
2      0.984415  …    0.092875     -0.223588             -0.235877
3     -1.862993  …   -0.499103      1.041440             -0.235877
4      0.951306  …    0.463410     -0.223588             -0.235877
…           …  …         …            …                     …
1455   0.918196  …   -0.972685     -0.223588             -0.235877
1456   0.222899  …    0.759399      1.041440             -0.235877
1457  -1.002149  …   -0.369744     -0.223588             -0.235877
1458  -0.704164  …   -0.865252      1.041440             -0.235877
1459  -0.207523  …    0.847099     -1.488617             -0.235877

      MasVnrType_None  MSSubClass_60.0  GarageCars  GarageArea  \
0           -1.203608         1.969844    0.311618    0.350880
1            0.830266        -0.507307    0.311618   -0.060710
2           -1.203608         1.969844    0.311618    0.631510
3            0.830266        -0.507307    1.649742    0.790533
4           -1.203608         1.969844    1.649742    1.697903
…                 …                …           …           …
1455         0.830266         1.969844    0.311618   -0.060710
1456        -1.203608        -0.507307    0.311618    0.126376
1457         0.830266        -0.507307   -1.026506   -1.033560
1458         0.830266        -0.507307   -1.026506   -1.089686
1459         0.830266        -0.507307   -1.026506   -0.921308

      FireplaceQu_IsNull  GrLivArea  SalePrice
0               1.056020   0.370207   208500.0
1              -0.946303  -0.482347   181500.0
2              -0.946303   0.514836   223500.0
3              -0.946303   0.383528   140000.0
4              -0.946303   1.298881   250000.0
…                    …          …          …
1455           -0.946303   0.250316   175000.0
1456           -0.946303   1.061003   210000.0
1457           -0.946303   1.569110   266500.0
1458            1.056020  -0.832502   142125.0
1459            1.056020  -0.493765   147500.0

[1460 rows x 28 columns]
```

[38]:
```python
class GaussianKernels:
    def __init__(self,sigma,X_train,Y_train):
        self.X_train = X_train
```

```
        self.Y_train = Y_train
        self.sigma = sigma

    def calculate_kernel(self,x_i,x_j):
        return 1/(np.sqrt(2*np.pi))*np.exp(-0.5*np.square(np.linalg.norm(x=x_i␣
↪- x_j,axis=-1))/self.sigma)

    def predict(self,x_test):
        kernels=self.calculate_kernel(self.X_train,x_test)
        if np.sum(kernels)==0:
            return self.Y_train.mean()
        weights = kernels / np.sum(kernels)
        return np.dot(weights,self.Y_train)
```

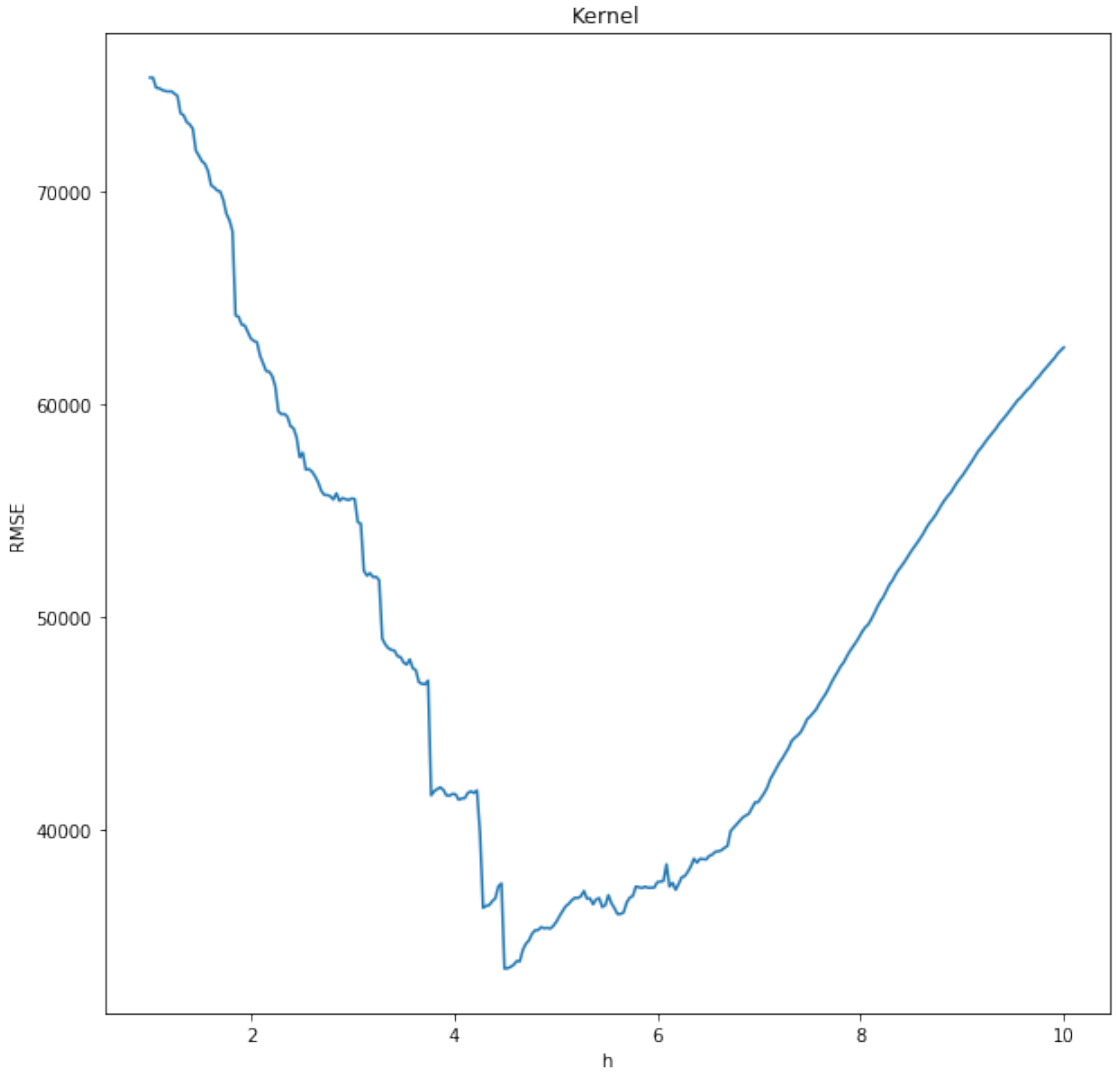
[39]:
```
class IndicatorKernels:
    def __init__(self,h,X_train,Y_train):
        self.X_train = X_train
        self.Y_train = Y_train
        self.h = h

    def calculate_kernel(self,x_i,x_j):
         return np.where(np.abs(np.linalg.norm(x=x_i - x_j,axis=-1)) <= self.
↪h,1,0)

    def predict(self,x_test):
        kernels=self.calculate_kernel(self.X_train,x_test)
        if np.sum(kernels)==0:
            return self.Y_train.mean()
        weights = kernels / np.sum(kernels)
        return np.dot(weights,self.Y_train)
```

[35]:
```
mask = np.random.rand(len(data)) <= 0.85
train =data[mask]
test = data[~mask]

X_train = train.drop(['SalePrice'],axis=1).to_numpy().astype(np.
↪dtype('float64'))
X_test = test.drop(['SalePrice'],axis=1).to_numpy().astype(np.dtype('float64'))
Y_train = train['SalePrice'].to_numpy().astype(np.dtype('float64'))
Y_test = test['SalePrice'].to_numpy().astype(np.dtype('float64'))
```

[45]:
```
answers = {}
for h in np.linspace(1,10,300):
    kernel_regressor=IndicatorKernels(h,X_train,Y_train)
    sum = 0
    predicted=[]
    for test in X_test:
```

```
        y_prediction=kernel_regressor.predict(test)
        predicted.append(y_prediction)

    predicted = np.array(predicted)
    result = np.sqrt(np.mean(np.square(predicted-Y_test)))
    answers[h] = result

plt.figure(figsize=(10,10))
answers_np=np.array(list(answers.items()))
plt.title('Kernel')
plt.xlabel('h')
plt.ylabel('RMSE')
print(np.nanmin(answers_np[:,1]))
plt.plot(answers_np[:,0],answers_np[:,1])

min_indicator = np.argmin(answers_np[:,1])
print(answers_np[min_indicator,:])
```

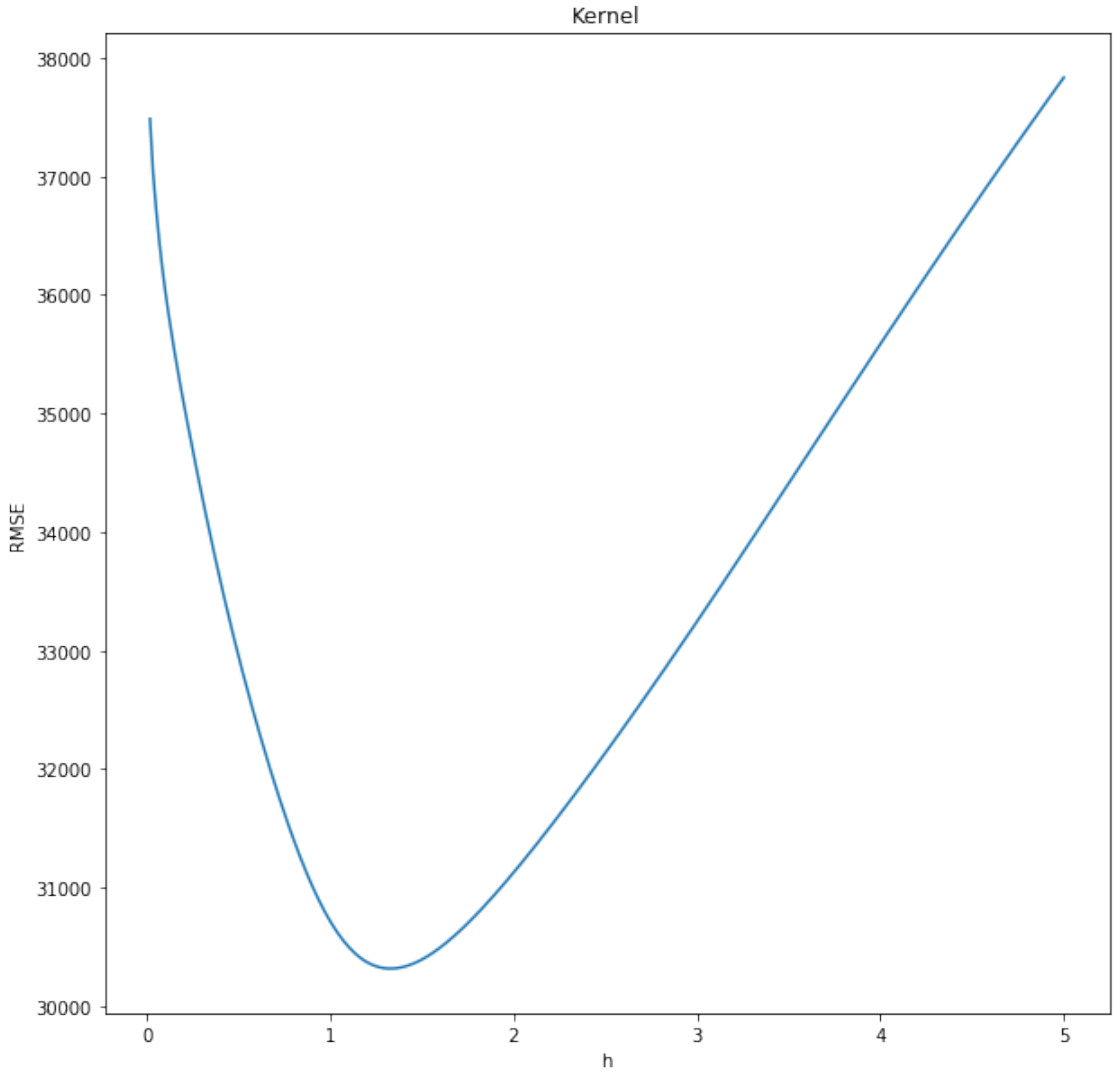
```
<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

33498.15061130628

<IPython.core.display.Javascript object>

[4.49163880e+00 3.34981506e+04]
```



[46]:
```
answers = {}
for h in np.linspace(0,5,300):
    kernel_regressor=GaussianKernels(h,X_train,Y_train)
    sum = 0
    predicted=[]
    for test in X_test:
        y_prediction=kernel_regressor.predict(test)
        predicted.append(y_prediction)

    predicted = np.array(predicted)
    result = np.sqrt(np.mean(np.square(predicted-Y_test)))
    answers[h] = result

plt.figure(figsize=(10,10))
```

```
answers_np2=np.array(list(answers.items()))
plt.title('Kernel')
plt.xlabel('h')
plt.ylabel('RMSE')
print(np.nanmin(answers_np2[:,1]))
plt.plot(answers_np2[:,0],answers_np2[:,1])

min_gaussian = np.nanargmin(answers_np2[:,1])
print(answers_np2[min_gaussian,:])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:8:
RuntimeWarning: divide by zero encountered in true_divide

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:8:
RuntimeWarning: invalid value encountered in true_divide


<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

30318.27613520855

<IPython.core.display.Javascript object>

[1.32107023e+00 3.03182761e+04]
```



[50]:
```
kernel_regressor=IndicatorKernels(answers_np[min_indicator,0],X_train,Y_train)
predictionOfIndicator = []
for row in data_test.to_numpy().astype(np.dtype('float64')):
    predictionOfIndicator.append(kernel_regressor.predict(row))
predictionOfIndicatorNumpy = np.array(predictionOfIndicator)
```

[54]:
```
kernel_gaussian=GaussianKernels(answers_np2[min_gaussian,0],X_train,Y_train)
predictionOfGuassian = []
for row in data_test.to_numpy().astype(np.dtype('float64')):
    predictionOfGuassian.append(kernel_gaussian.predict(row))
predictionOfGaussianNumpy = np.array(predictionOfGuassian)
```

0.1 Plot of test results

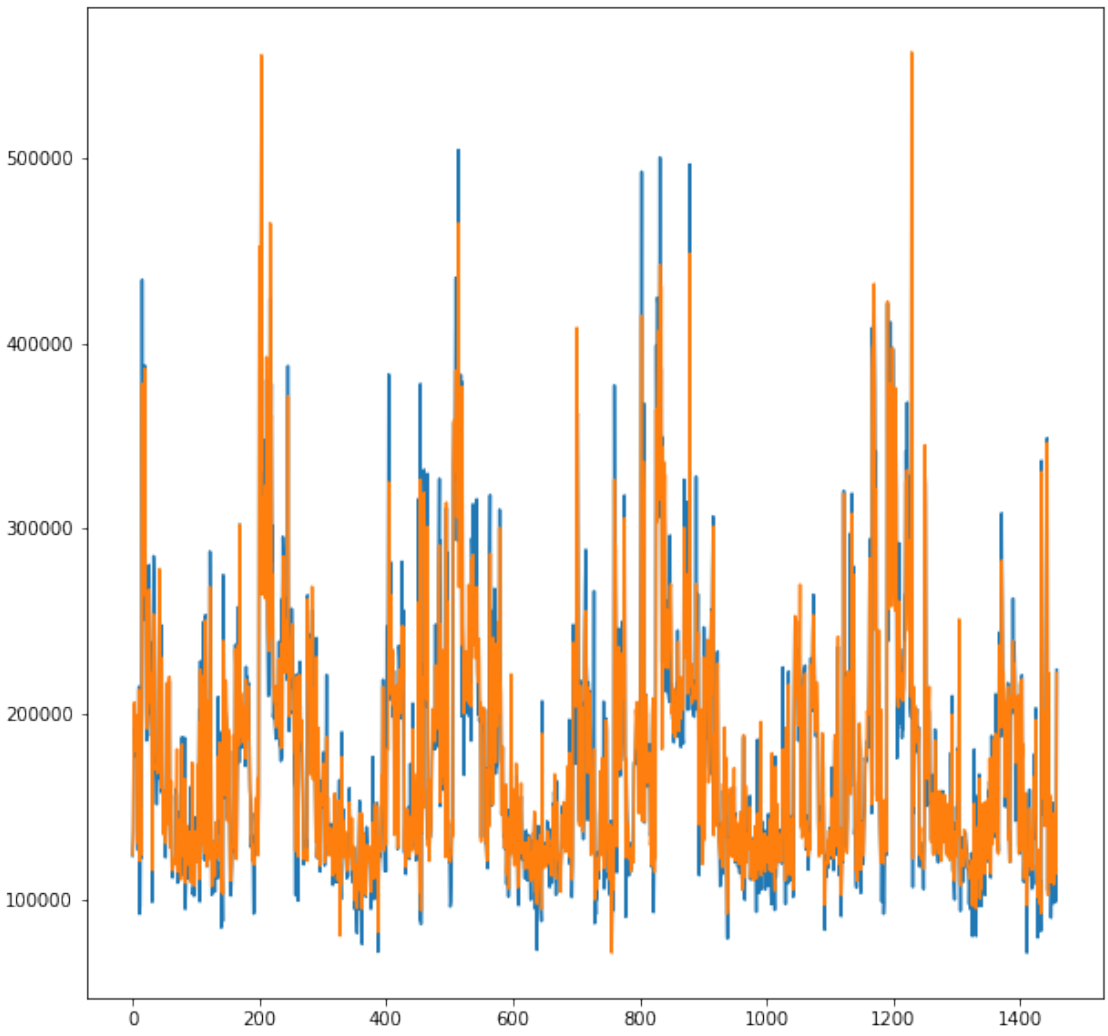
[68]:
```
plt.figure(figsize=(10,10))
plt.plot(predictionOfGaussianNumpy)
plt.plot(predictionOfIndicatorNumpy)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

[68]:
```
[<matplotlib.lines.Line2D at 0x1b6379101c8>]
```



0.2 plot of best results of gaussian kernel vs actual result

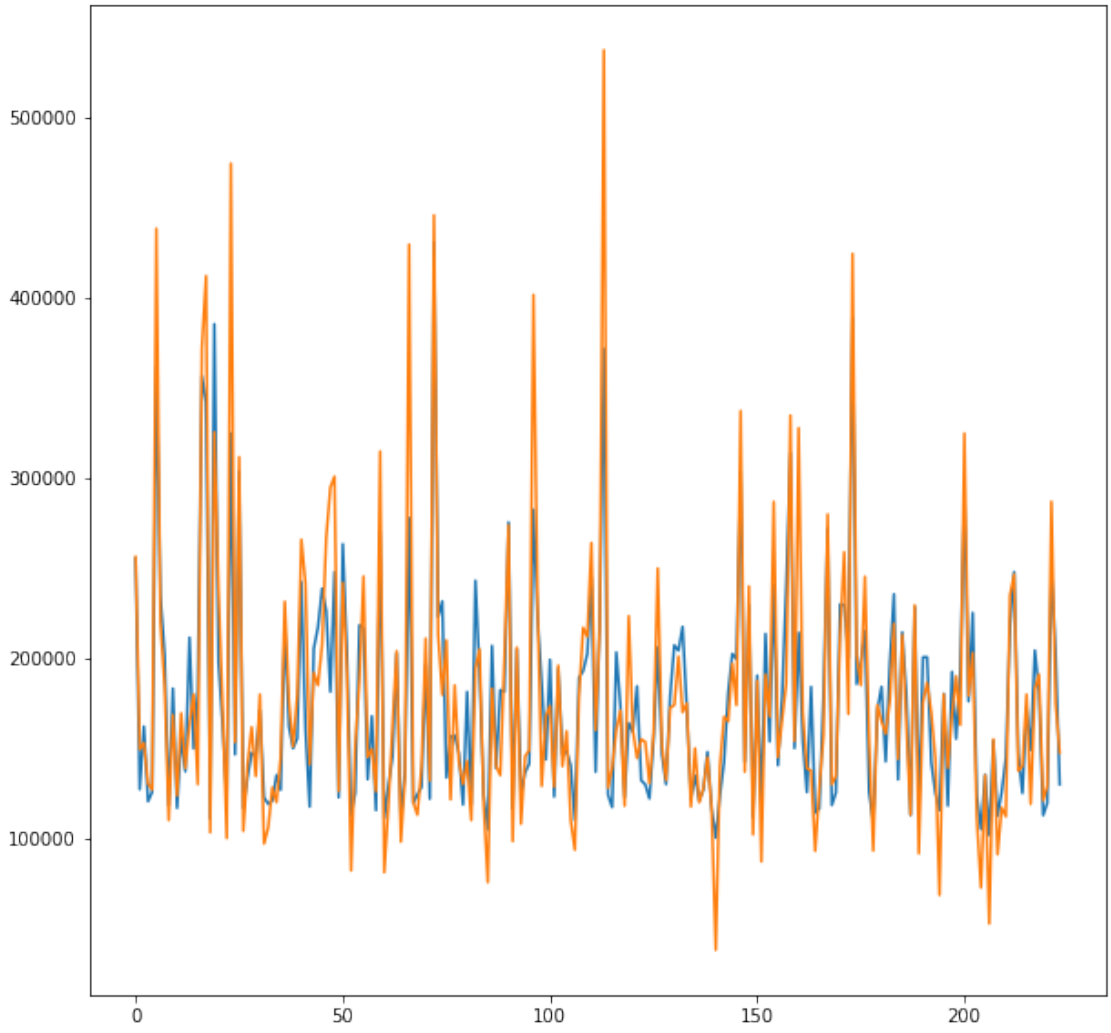
[71]:
```
kernel_regressor=IndicatorKernels(answers_np[min_indicator,0],X_train,Y_train)
predictionOfIndicator = []
for row in X_test:
    predictionOfIndicator.append(kernel_regressor.predict(row))
predictionOfIndicatorNumpy = np.array(predictionOfIndicator)

plt.figure(figsize=(10,10))
plt.plot(predictionOfIndicatorNumpy)
plt.plot(Y_test)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

[71]:
```
[<matplotlib.lines.Line2D at 0x1b63793c1c8>]
```



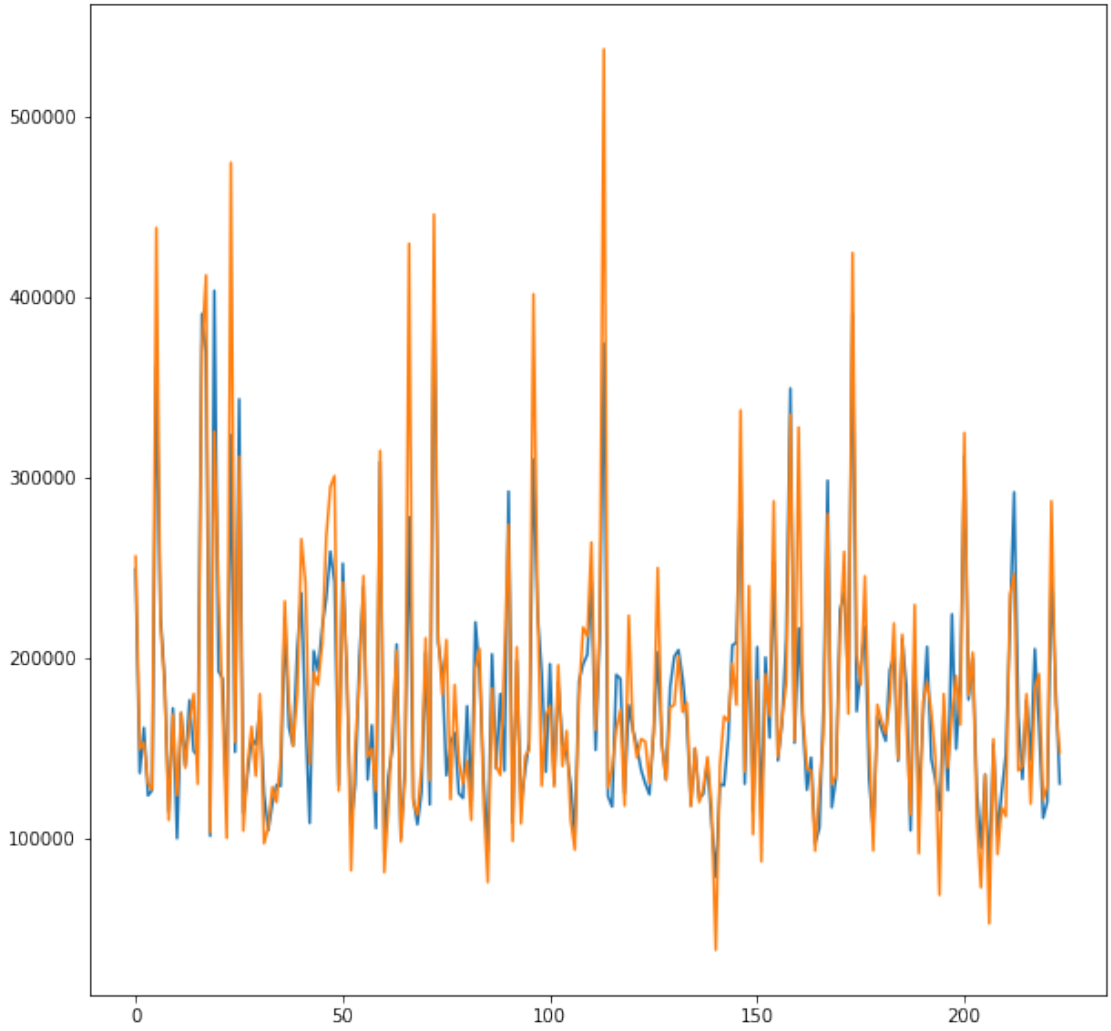
0.3 plot of best results of indicator kernel vs actual result

[72]:
```
kernel_gaussian=GaussianKernels(answers_np2[min_gaussian,0],X_train,Y_train)
predictionOfGuassian = []
for row in X_test:
    predictionOfGuassian.append(kernel_gaussian.predict(row))
predictionOfGaussianNumpy = np.array(predictionOfGuassian)

plt.figure(figsize=(10,10))
plt.plot(predictionOfGaussianNumpy)
plt.plot(Y_test)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
```

[72]:
```
[<matplotlib.lines.Line2D at 0x1b6376e9d48>]
```



[73]:
```
np.savetxt("G:
↪\Documents\ReferenceBooks\MachineLearning\Rohban\Homework\HW3\TestResultIndicator.
↪csv",predictionOfIndicatorNumpy , delimiter=",")
```

[74]:
```
np.savetxt("G:
↪\Documents\ReferenceBooks\MachineLearning\Rohban\Homework\HW3\TestResultGaussian.
↪csv",predictionOfGaussianNumpy , delimiter=",")
```

[ ]:
```

```