# Saudi Stocks

Load the data

In [1]:
```
#imports
import os
import boto3
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from pandas import read_csv
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
import math
```

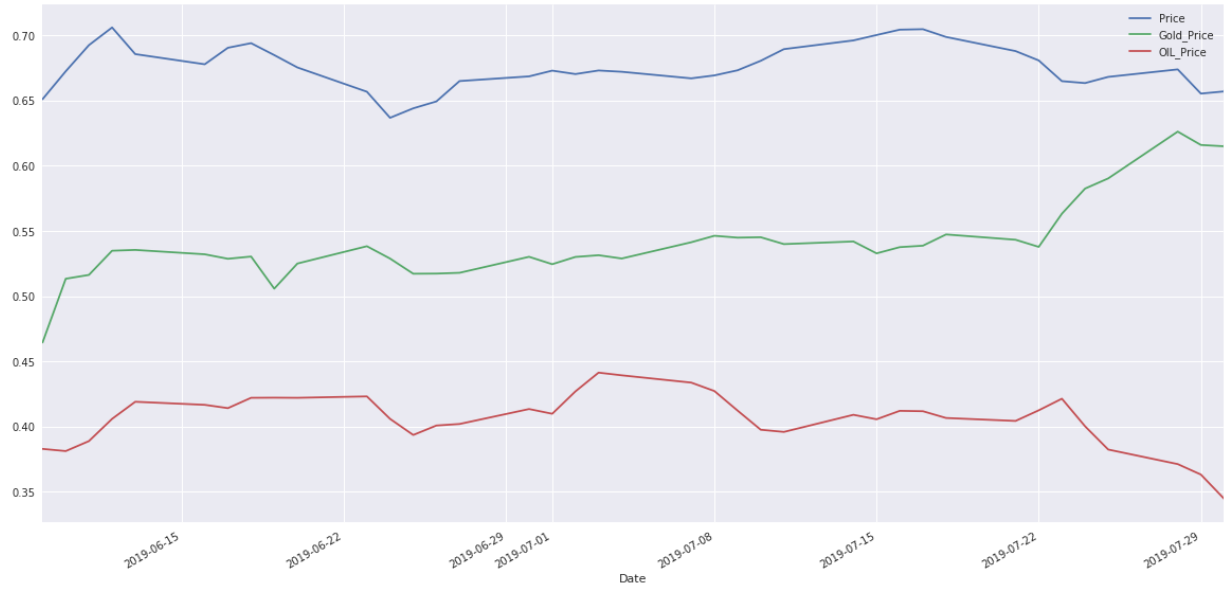
In [2]:
```
data_dir = '../data'
file_name = "combined.csv"

series = pd.read_csv(file_name, header=0, index_col=0, parse_dates=True, squeeze=
series.sort_values(by=['Date'], inplace=True, ascending=True)

sc = MinMaxScaler(feature_range=(0,1))
series.iloc[:,:] = sc.fit_transform(series.iloc[:,:])

print(series.shape)
print(series.head(5))
```

```
(2743, 42)
              Price  ALRAJHI_Price  ALRAJHI_Vol.  BIST_Price  BIST_Vol.  \
Date
2009-01-03  0.127250       0.204545      0.067419    0.058005   0.282602
2009-01-04  0.130824       0.204545      0.100591    0.049655   0.196784
2009-01-05  0.149635       0.204545      0.085960    0.032455   0.217641
2009-01-06  0.165218       0.178322      0.061113    0.035110   0.222320
2009-01-07  0.169846       0.178322      0.044338    0.027325   0.133602

            DAX_Price  DAX_Vol.  DFM_Price  DFM_Vol.  DJI_Price  ...  \
Date                                                              ...
2009-01-03   0.070779  0.197595   0.182197  0.250013   0.102868  ...
2009-01-04   0.055613  0.000000   0.146036  0.110078   0.101427  ...
2009-01-05   0.054436  0.000000   0.132128  0.080997   0.094970  ...
2009-01-06   0.055473  0.000000   0.143255  0.073285   0.089202  ...
2009-01-07   0.035201  0.000000   0.126565  0.061768   0.087878  ...

            SABIC_Price  SABIC_Vol.  SCECO_Price  SCECO_Vol.  SSEC_Price  \
Date
2009-01-03     0.107231    0.528982     0.049853    0.303602    0.209565
2009-01-04     0.151500    0.144367     0.017595    0.101030    0.201908
2009-01-05     0.156419    0.574652     0.008798    0.064792    0.204261
2009-01-06     0.176094    0.525526     0.000000    0.055789    0.185967
2009-01-07     0.158879    0.738077     0.008798    0.041609    0.183675

            SSEC_Vol.  STC_Price  STC_Vol.  TA_Price  TA_Vol.
Date
2009-01-03   0.037494   0.139286  0.026379  0.332791      0.0
2009-01-04   0.019352   0.191667  0.076003  0.311306      0.0
2009-01-05   0.004233   0.177381  0.047263  0.302999      0.0
2009-01-06   0.012941   0.179762  0.041189  0.295000      0.0
2009-01-07   0.012941   0.155952  0.028762  0.252667      0.0

[5 rows x 42 columns]
```

Visualize data Here will try different plot to aid us in understanding our data Ref [https://machinelearningmastery.com/time-series-data-visualization-with-python/](https://machinelearningmastery.com/time-series-data-visualization-with-python/) (https://machinelearningmastery.com/time-series-data-visualization-with-python/)

Ad hoc grapy , we can selectivly view featuers in a specific timeframe

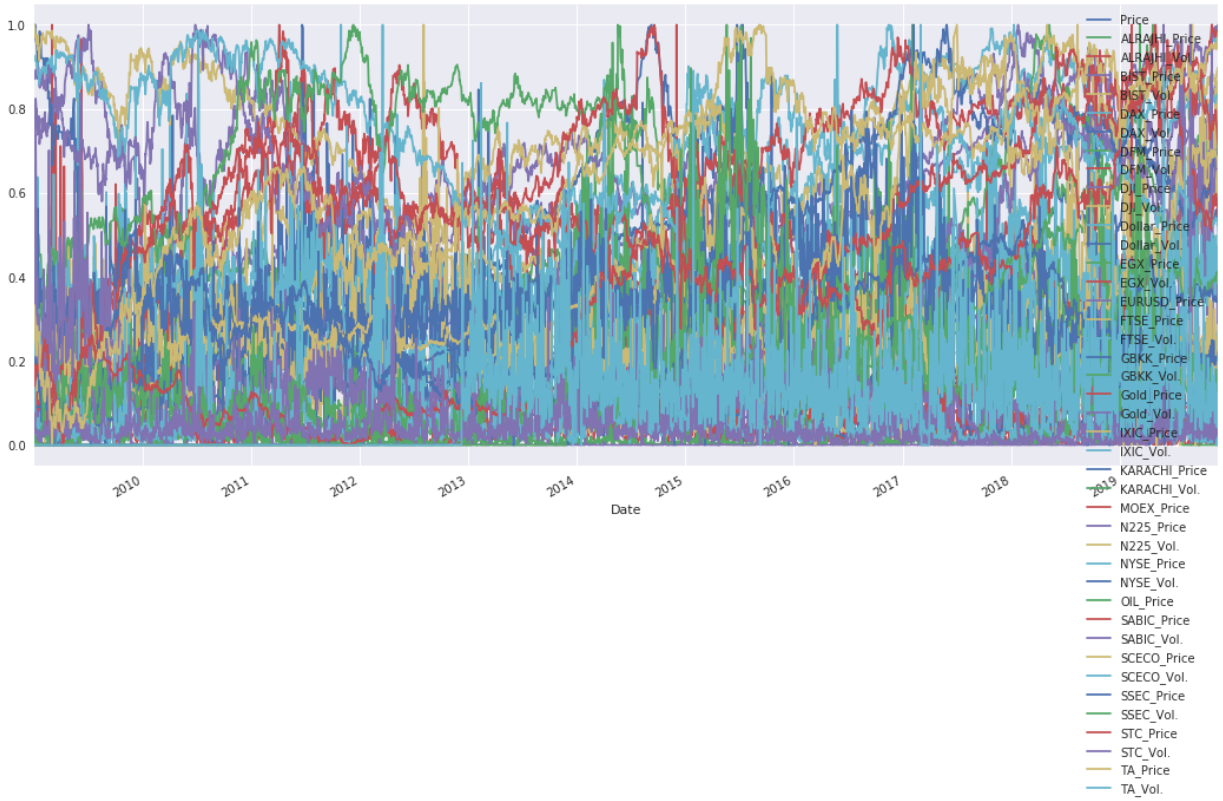
In [3]:
```
# select featuers
tasi_oil = series.loc[:,['Price','Gold_Price','OIL_Price']]
# select time frame
tasi_oil.loc['2019-06-01':'2019-07-30'].plot(figsize=(20,10),legend=True)

plt.show()
```



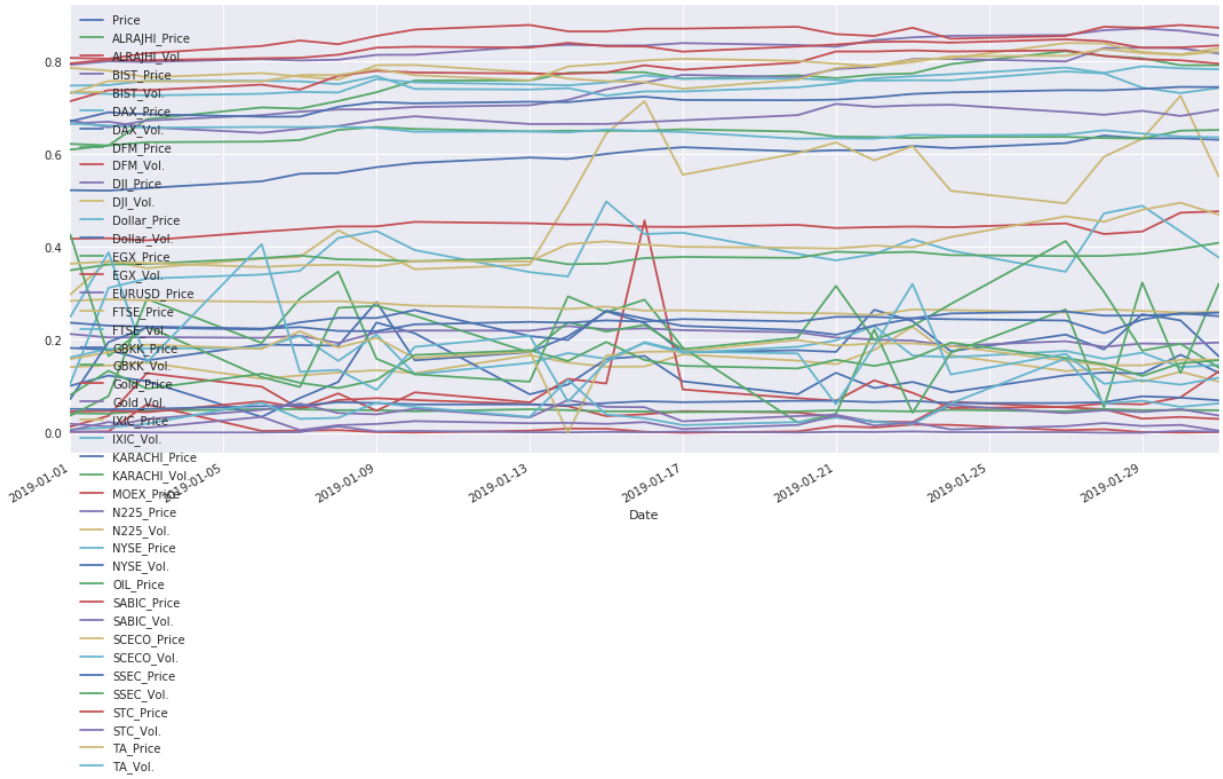

Ten Years look at TASI

In [4]:
```
plt.figure(figsize = (18,9))
plt.plot(range(series.shape[0]),(series['Price']))
#plt.xticks(range(0,series.shape[0],500),series.loc[::500],rotation=45)
plt.xlabel('2009-2019 TASI',fontsize=18)
plt.ylabel('Price',fontsize=18)
plt.show()
```

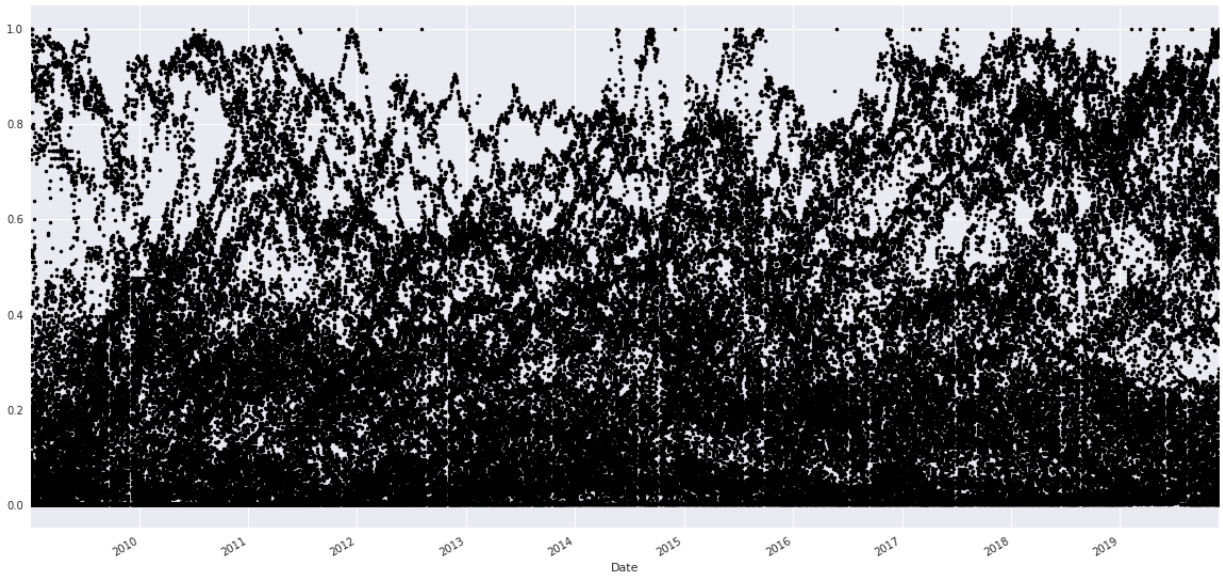
In [5]:
```
#Ten year View
series.plot(figsize=(18,8),legend=True)

plt.show()
```



In [6]:
```
#One Month View
series.loc['2019-01-01':'2019-01-31'].plot(figsize=(18,8),legend=True)

plt.show()
```



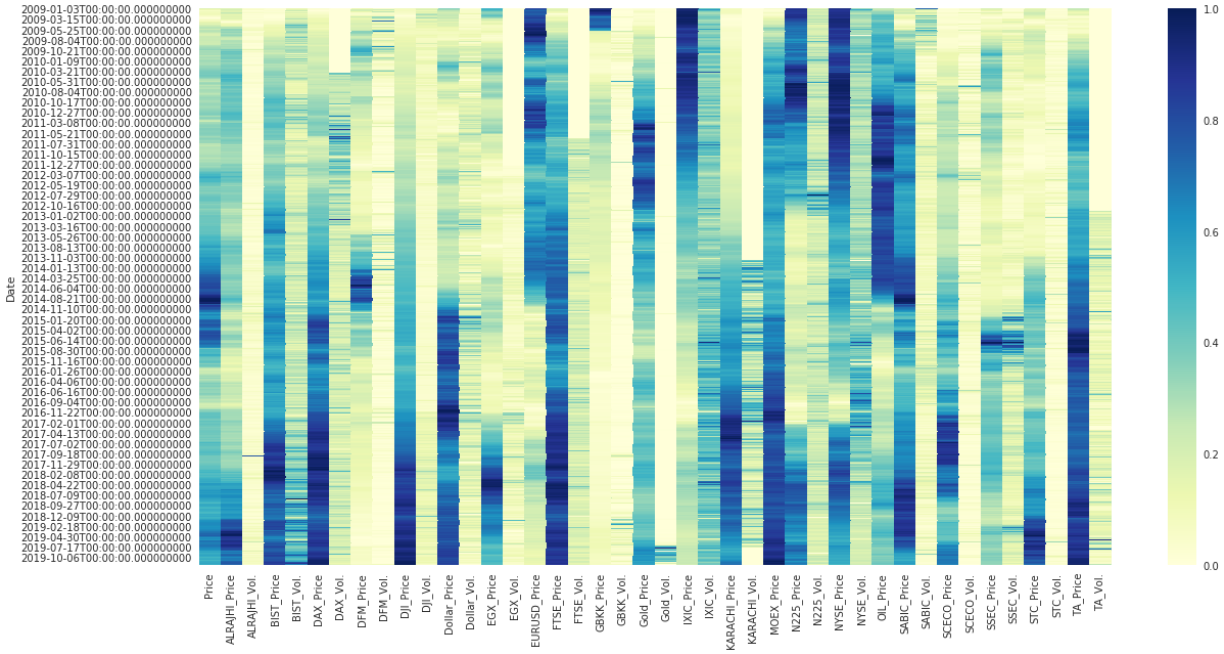

In [7]:
```
series.plot(style='k.',figsize=(20,10),legend=False)
plt.show()
```




A heatmap of our ten year data

In [8]:
```
plt.subplots(figsize=(20,10))
sns.heatmap(series,cmap="YlGnBu") #center=series.loc["January", 2010]
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb2d6c21e10>




Split the data

In [9]:
```
#9 Years of training data
train_df = series.loc['2009':'2018']

#1 year for test data
test_df = series.loc['2019']
print(test_df.shape)

x_train = train_df.iloc[:,1:]
x_test  = test_df.iloc[:,1:]
y_train = train_df.iloc[:,0:1]
y_test  = test_df.iloc[:,0:1]

print('X train ')
print(x_train.head(5))
print('Y train ')
print(y_train.head(5))
print('X test ')
print(x_test.head(5))
print('Y test ')
print(y_test.head(5))
```

```
(227, 42)
X train
            ALRAJHI_Price  ALRAJHI_Vol.  BIST_Price  BIST_Vol.  DAX_Price  \
Date
2009-01-03       0.204545      0.067419    0.058005   0.282602   0.070779
2009-01-04       0.204545      0.100591    0.049655   0.196784   0.055613
2009-01-05       0.204545      0.085960    0.032455   0.217641   0.054436
2009-01-06       0.178322      0.061113    0.035110   0.222320   0.055473
2009-01-07       0.178322      0.044338    0.027325   0.133602   0.035201

            DAX_Vol.  DFM_Price  DFM_Vol.  DJI_Price  DJI_Vol.  ...  \
Date                                                             ...
2009-01-03  0.197595   0.182197  0.250013   0.102868  0.099986  ...
2009-01-04  0.000000   0.146036  0.110078   0.101427  0.089759  ...
2009-01-05  0.000000   0.132128  0.080997   0.094970  0.121490  ...
2009-01-06  0.000000   0.143255  0.073285   0.089202  0.135466  ...
2009-01-07  0.000000   0.126565  0.061768   0.087878  0.158834  ...

            SABIC_Price  SABIC_Vol.  SCECO_Price  SCECO_Vol.  SSEC_Price  \
Date
2009-01-03     0.107231    0.528982     0.049853    0.303602    0.209565
2009-01-04     0.151500    0.144367     0.017595    0.101030    0.201908
2009-01-05     0.156419    0.574652     0.008798    0.064792    0.204261
2009-01-06     0.176094    0.525526     0.000000    0.055789    0.185967
2009-01-07     0.158879    0.738077     0.008798    0.041609    0.183675

            SSEC_Vol.  STC_Price  STC_Vol.  TA_Price  TA_Vol.
Date
2009-01-03   0.037494   0.139286  0.026379  0.332791      0.0
2009-01-04   0.019352   0.191667  0.076003  0.311306      0.0
2009-01-05   0.004233   0.177381  0.047263  0.302999      0.0
2009-01-06   0.012941   0.179762  0.041189  0.295000      0.0
2009-01-07   0.012941   0.155952  0.028762  0.252667      0.0

[5 rows x 41 columns]
Y train
```

```
                Price
Date
2009-01-03  0.127250
2009-01-04  0.130824
2009-01-05  0.149635
2009-01-06  0.165218
2009-01-07  0.169846
X test
            ALRAJHI_Price  ALRAJHI_Vol.  BIST_Price  BIST_Vol.  DAX_Price  \
Date
2019-01-01       0.609946      0.012807    0.673034   0.297699   0.733351
2019-01-02       0.618881      0.036518    0.658046   0.370571   0.729901
2019-01-03       0.675602      0.128339    0.672685   0.354552   0.726712
2019-01-06       0.700855      0.099329    0.684139   0.375116   0.730339
2019-01-07       0.698329      0.051275    0.691764   0.379015   0.734326

            DAX_Vol.  DFM_Price  DFM_Vol.  DJI_Price  DJI_Vol.  ...  \
Date                                                             ...
2019-01-01  0.182772   0.042003  0.002025   0.792862  0.141592  ...
2019-01-02  0.178687   0.041168  0.003029   0.800448  0.144868  ...
2019-01-03  0.156016   0.047288  0.061021   0.798914  0.151091  ...
2019-01-06  0.189724   0.050904  0.003878   0.804879  0.116496  ...
2019-01-07  0.188807   0.050626  0.004422   0.802126  0.123328  ...

            SABIC_Price  SABIC_Vol.  SCECO_Price  SCECO_Vol.  SSEC_Price  \
Date
2019-01-01     0.807673    0.019207     0.364223    0.006720    0.237108
2019-01-02     0.805706    0.013282     0.368915    0.011115    0.230694
2019-01-03     0.817511    0.029575     0.365396    0.019317    0.228806
2019-01-06     0.833251    0.049324     0.357185    0.064792    0.224413
2019-01-07     0.845057    0.062902     0.360704    0.029904    0.227537

            SSEC_Vol.  STC_Price  STC_Vol.  TA_Price   TA_Vol.
Date
2019-01-01   0.081035   0.714286  0.005018  0.731275  0.249702
2019-01-02   0.133527   0.738095  0.023183  0.757735  0.388585
2019-01-03   0.094944   0.738095  0.010792  0.756931  0.151570
2019-01-06   0.127117   0.750000  0.034095  0.756958  0.406068
2019-01-07   0.107523   0.739286  0.005519  0.770833  0.131093

[5 rows x 41 columns]
Y test
                Price
Date
2019-01-01  0.522610
2019-01-02  0.521537
2019-01-03  0.527180
2019-01-06  0.541678
2019-01-07  0.558110
```

Calc Correlation

In [10]:
```
# Create correlation matrix for just Features to determine different models to te
corr_matrix = series.corr().abs().round(2)
# display shows all of a dataframe
display(corr_matrix)
```

| | Price | ALRAJHI_Price | ALRAJHI_Vol. | BIST_Price | BIST_Vol. | DAX_Price | DAX_V |
|---|---|---|---|---|---|---|---|
| **Price** | 1.00 | 0.41 | 0.28 | 0.52 | 0.20 | 0.55 | 0. |
| **ALRAJHI_Price** | 0.41 | 1.00 | 0.16 | 0.41 | 0.56 | 0.33 | 0. |
| **ALRAJHI_Vol.** | 0.28 | 0.16 | 1.00 | 0.27 | 0.22 | 0.31 | 0. |
| **BIST_Price** | 0.52 | 0.41 | 0.27 | 1.00 | 0.55 | 0.91 | 0. |
| **BIST_Vol.** | 0.20 | 0.56 | 0.22 | 0.55 | 1.00 | 0.55 | 0. |
| **DAX_Price** | 0.55 | 0.33 | 0.31 | 0.91 | 0.55 | 1.00 | 0. |
| **DAX_Vol.** | 0.09 | 0.07 | 0.08 | 0.18 | 0.12 | 0.04 | 1. |
| **DFM_Price** | 0.49 | 0.23 | 0.06 | 0.18 | 0.30 | 0.11 | 0. |
| **DFM_Vol.** | 0.16 | 0.20 | 0.04 | 0.41 | 0.17 | 0.34 | 0. |
| **DJI_Price** | 0.52 | 0.55 | 0.31 | 0.91 | 0.67 | 0.94 | 0. |
| **DJI_Vol.** | 0.19 | 0.29 | 0.07 | 0.20 | 0.41 | 0.19 | 0. |
| **Dollar_Price** | 0.34 | 0.17 | 0.27 | 0.67 | 0.45 | 0.82 | 0. |
| **Dollar_Vol.** | 0.23 | 0.17 | 0.03 | 0.16 | 0.13 | 0.14 | 0. |
| **EGX_Price** | 0.26 | 0.57 | 0.21 | 0.61 | 0.62 | 0.57 | 0. |
| **EGX_Vol.** | 0.27 | 0.03 | 0.19 | 0.59 | 0.27 | 0.64 | 0. |
| **EURUSD_Price** | 0.35 | 0.16 | 0.23 | 0.65 | 0.37 | 0.77 | 0. |
| **FTSE_Price** | 0.61 | 0.41 | 0.26 | 0.92 | 0.49 | 0.94 | 0. |
| **FTSE_Vol.** | 0.38 | 0.04 | 0.12 | 0.53 | 0.15 | 0.56 | 0. |
| **GBKK_Price** | 0.40 | 0.21 | 0.13 | 0.65 | 0.26 | 0.63 | 0. |
| **GBKK_Vol.** | 0.06 | 0.46 | 0.06 | 0.11 | 0.29 | 0.07 | 0. |
| **Gold_Price** | 0.02 | 0.21 | 0.13 | 0.11 | 0.14 | 0.10 | 0. |
| **Gold_Vol.** | 0.11 | 0.37 | 0.18 | 0.21 | 0.34 | 0.20 | 0. |
| **IXIC_Price** | 0.53 | 0.12 | 0.26 | 0.75 | 0.31 | 0.85 | 0. |
| **IXIC_Vol.** | 0.05 | 0.07 | 0.04 | 0.18 | 0.11 | 0.26 | 0. |
| **KARACHI_Price** | 0.46 | 0.20 | 0.27 | 0.84 | 0.47 | 0.95 | 0. |
| **KARACHI_Vol.** | 0.48 | 0.08 | 0.22 | 0.43 | 0.14 | 0.63 | 0. |
| **MOEX_Price** | 0.25 | 0.44 | 0.16 | 0.79 | 0.50 | 0.78 | 0. |
| **N225_Price** | 0.40 | 0.41 | 0.09 | 0.21 | 0.25 | 0.30 | 0. |
| **N225_Vol.** | 0.16 | 0.20 | 0.12 | 0.15 | 0.20 | 0.19 | 0. |
| **NYSE_Price** | 0.36 | 0.18 | 0.16 | 0.36 | 0.02 | 0.53 | 0. |
| **NYSE_Vol.** | 0.10 | 0.49 | 0.06 | 0.04 | 0.22 | 0.10 | 0. |

| | Price | ALRAJHI_Price | ALRAJHI_Vol. | BIST_Price | BIST_Vol. | DAX_Price | DAX_V |
|---|---|---|---|---|---|---|---|
| **OIL_Price** | 0.03 | 0.12 | 0.22 | 0.33 | 0.30 | 0.52 | 0. |
| **SABIC_Price** | 0.71 | 0.65 | 0.12 | 0.58 | 0.30 | 0.49 | 0. |
| **SABIC_Vol.** | 0.35 | 0.35 | 0.01 | 0.42 | 0.16 | 0.34 | 0. |
| **SCECO_Price** | 0.40 | 0.21 | 0.23 | 0.83 | 0.42 | 0.88 | 0. |
| **SCECO_Vol.** | 0.06 | 0.02 | 0.13 | 0.16 | 0.18 | 0.17 | 0. |
| **SSEC_Price** | 0.05 | 0.04 | 0.09 | 0.38 | 0.19 | 0.46 | 0. |
| **SSEC_Vol.** | 0.37 | 0.02 | 0.18 | 0.30 | 0.12 | 0.44 | 0. |
| **STC_Price** | 0.56 | 0.61 | 0.36 | 0.70 | 0.67 | 0.81 | 0. |
| **STC_Vol.** | 0.09 | 0.04 | 0.35 | 0.16 | 0.11 | 0.13 | 0. |
| **TA_Price** | 0.59 | 0.34 | 0.26 | 0.81 | 0.41 | 0.86 | 0. |
| **TA_Vol.** | 0.40 | 0.11 | 0.20 | 0.54 | 0.25 | 0.59 | 0. |

42 rows × 42 columns

Benchamrk sklearn LinearRegression

In [11]:
```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(x_train, y_train)

y_linear_test = y_test.loc[:,'Price'].values

# Make predictions using the testing set
y_linear_pred = regr.predict(x_test)
```

Calculate root mean squared error

In [12]:
```
testScore = math.sqrt(mean_squared_error(y_linear_test, y_linear_pred))
print('Test Score: %.2f RMSE' % (testScore))
```

```
Test Score: 0.12 RMSE
```

Plot Predictions vs Actual Test data (TASI for 2019)

In [13]:
```
plt.subplots(figsize=(20,10))
plt.plot(y_linear_test)
plt.plot(y_linear_pred)
plt.show()
```


LSTM Model

LSTM

#https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/ (https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/)

In [14]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.metrics import mean_squared_error
```

```
Using MXNet backend
```

In [15]:
```
# create and fit the LSTM network
total_featuers = 41

print(x_train.shape)
print(x_train)

X_train_init = np.asarray(x_train)
X_train = np.hstack(X_train_init).reshape(len(x_train),1,total_featuers)


Y_train_init = np.asarray(y_train)
Y_train = np.hstack(Y_train_init).reshape(len(y_train),1,1)


X_test_init = np.asarray(x_test)
X_test = np.hstack(X_test_init).reshape(len(x_test),1,total_featuers)


Y_test_init = np.asarray(y_test)
Y_test = np.hstack(Y_test_init).reshape(len(y_test),1,1)



X_val_init = np.asarray(x_val)
X_val = np.hstack(X_val_init).reshape(len(x_val),1,total_featuers)


Y_val_init = np.asarray(y_val)
Y_val = np.hstack(Y_val_init).reshape(len(y_val),1,1)
```

| | | | | | |
|---|---|---|---|---|---|
| 2018-11-20 | 0.588182 | 0.042572 | 0.701880 | 0.625483 | 0.755275 |
| 2018-11-21 | 0.592269 | 0.029077 | 0.712325 | 0.463320 | 0.760754 |
| 2018-11-22 | 0.575758 | 0.016086 | 0.711515 | 0.348595 | 0.777132 |
| 2018-11-25 | 0.579643 | 0.034374 | 0.715344 | 0.320069 | 0.772521 |
| 2018-11-26 | 0.577117 | 0.043708 | 0.725150 | 0.548263 | 0.771487 |
| 2018-11-27 | 0.580808 | 0.025924 | 0.720596 | 0.486486 | 0.771422 |
| 2018-11-28 | 0.594794 | 0.034122 | 0.726260 | 0.486486 | 0.767278 |
| 2018-11-29 | 0.621406 | 0.047491 | 0.737042 | 0.463320 | 0.788325 |
| 2018-12-02 | 0.626457 | 0.031095 | 0.740012 | 0.637066 | 0.775171 |
| 2018-12-03 | 0.623737 | 0.025293 | 0.735497 | 0.559846 | 0.761517 |
| 2018-12-04 | 0.617521 | 0.021762 | 0.724313 | 0.536680 | 0.722171 |
| 2018-12-05 | 0.608586 | 0.030212 | 0.725229 | 0.455598 | 0.719857 |
| 2018-12-06 | 0.606255 | 0.011798 | 0.713545 | 0.447876 | 0.703076 |
| 2018-12-09 | 0.604895 | 0.015329 | 0.722462 | 0.385514 | 0.719091 |
| 2018-12-10 | 0.609946 | 0.032861 | 0.707411 | 0.327000 | 0.734143 |
| 2018-12-11 | 0.612471 | 0.023023 | 0.703051 | 0.397683 | 0.733665 |
| 2018-12-12 | 0.608586 | 0.021635 | 0.685161 | 0.501931 | 0.727709 |
| 2018-12-13 | 0.602370 | 0.012933 | 0.698814 | 0.358656 | 0.718251 |
| 2018-12-16 | 0.613831 | 0.028698 | 0.690043 | 0.401544 | 0.715086 |
| 2018-12-17 | 0.614996 | 0.043077 | 0.684249 | 0.300888 | 0.717645 |

In [16]:
```
model = Sequential()
model.add(LSTM(4, input_shape=(1, total_featuers)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(X_train, y_train, epochs=100, batch_size=1, verbose=2)
```

```
 - 25s - loss: 4.1108e-04
Epoch 87/100
 - 9s - loss: 4.0956e-04
Epoch 88/100
 - 9s - loss: 4.1331e-04
Epoch 89/100
 - 9s - loss: 4.0082e-04
Epoch 90/100
 - 9s - loss: 4.0199e-04
Epoch 91/100
 - 9s - loss: 4.0032e-04
Epoch 92/100
 - 11s - loss: 3.9955e-04
Epoch 93/100
 - 38s - loss: 3.9490e-04
Epoch 94/100
 - 34s - loss: 3.9140e-04
Epoch 95/100
 - 35s - loss: 3.8605e-04
Epoch 96/100
```

In [17]:
```
# make predictions
testPredict  = model.predict(X_test)
```

Calculate root mean squared error

In [18]:
```
testScore = math.sqrt(mean_squared_error(y_test, testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
```

```
Test Score: 0.04 RMSE
```

In [19]:
```
#Plot the predictions

testValuesPlot = y_test.loc[:,'Price'].values
testLSTMPredictPlot = testPredict
testLinearPredictPlot = y_linear_pred
```

In [20]:
```
# LSTM 2019 Prediction vs Actual 2019
plt.subplots(figsize=(20,10))
plt.plot(testValuesPlot)
plt.plot(testLSTMPredictPlot)
plt.show()
```

In [21]:
```
# LSTM 2019 Prediction vs sklearn Linear Regression vs Actual 2019
plt.subplots(figsize=(20,10))
plt.plot(testValuesPlot)
plt.plot(testLinearPredictPlot)
plt.plot(testLSTMPredictPlot)
plt.show()
```


Fin.