Untitled

November 21, 2022

[2]:
```python
# Libraries
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.linear_model import Lasso, LassoCV
```

[3]:
```python
# Import data
data_set_r_working = pd.read_csv("../M2. module_2_data.csv")
data_set_r_working.head()
```

[3]:
```
       Date       DXY    METALS       OIL    US_STK  INTL_STK   X13W_TB  \
0  1/4/2016  0.002433  0.024283 -0.007559 -0.013980 -0.019802  0.047297
1  1/5/2016  0.005361 -0.004741 -0.021491  0.001691 -0.001263  0.322581
2  1/6/2016 -0.002213  0.013642 -0.055602 -0.012614 -0.015171  0.000000
3  1/7/2016 -0.009679  0.035249 -0.020606 -0.023992 -0.019255 -0.073171
4  1/8/2016  0.003258 -0.028064 -0.003306 -0.010977 -0.010471  0.000000

   X10Y_TBY    EURUSD  YEAR
0 -0.010577 -0.007316  2016
1  0.001336 -0.002436  2016
2 -0.031584 -0.006978  2016
3 -0.011024  0.002512  2016
4 -0.010683  0.013636  2016
```

List of our variables, Dependent Variable:

DXY: U.S. Dollar Index daily return

Independent Variables:

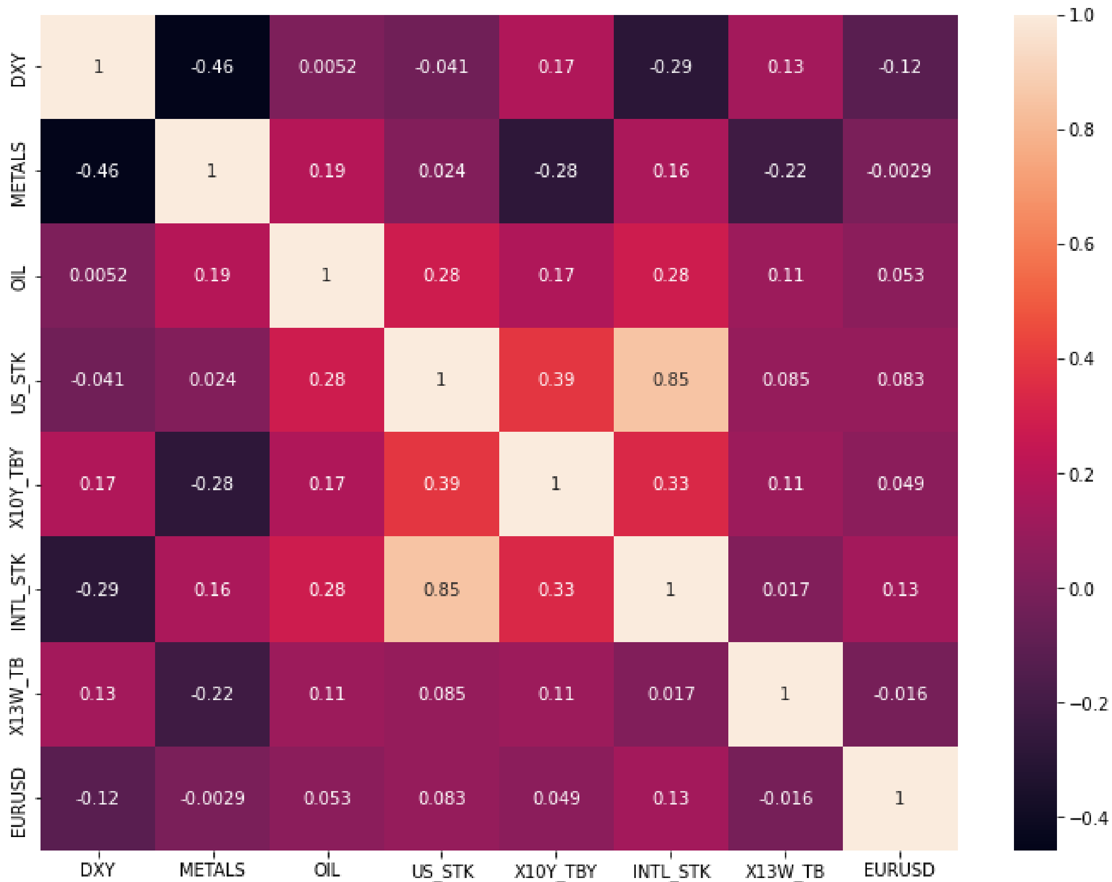
- **METALS**: Gold and Silver Index daily return
- **US_STK**: S&P 500 Index daily return
- **X13W_TB**: 13-week Treasury Bills daily return
- **X10Y_TBY**: 10 Year Treasury Bond Yield daily return
- **EURUSD**: EURUSD daily return

[6]:
```python
data_set_r_working.columns
```

```
[6]: Index(['Date', 'DXY', 'METALS', 'OIL', 'US_STK', 'INTL_STK', 'X13W_TB',
       'X10Y_TBY', 'EURUSD', 'YEAR'],
      dtype='object')
```

```
[ ]: # Correlation plot of all the variables
import seaborn as sns
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (12, 9)

data = data_set_r_working[
["DXY",
 "METALS",
 "OIL",
 "US_STK",
 "X10Y_TBY",
 "INTL_STK",
 "X13W_TB",
 "EURUSD"
]]
c = data.corr()
sns.heatmap(c, annot=True)
plt.show()
```



From above we can see that Oil have very low correlation with the dependent variable.

[36]:
```
# Create Training Dataset and Testing Dataset

np.random.seed(11111)  # Random seed
nrow = data_set_r_working.shape[0]

# choosing 50% data for training model and 50% data for testing the model

train_sequence = sorted(np.random.choice(nrow, int(nrow * 0.5), replace=False))
test_sequence = sorted(set(list(range(0, nrow))) - set(train_sequence))

train = data_set_r_working.filter(items=train_sequence, axis=0)
test = data_set_r_working.filter(items=test_sequence, axis=0)

# Make sure X matrix is in matrix form and Y is in vector form
ind_var = ['METALS', 'US_STK', 'INTL_STK', 'X13W_TB',
       'X10Y_TBY', 'EURUSD']

train_x = train.loc[:, ind_var]
train_y = train.DXY

test_x = test.loc[:, ind_var]
test_y = test.DXY

test_tot = test.loc[:, ['DXY', 'METALS', 'OIL', 'US_STK', 'INTL_STK', 'X13W_TB',
       'X10Y_TBY', 'EURUSD']]
```

[37]:
```
train.describe()
```

[37]:

	DXY	METALS	OIL	US_STK	INTL_STK	X13W_TB \
count	125.000000	125.000000	125.000000	125.000000	125.000000	125.000000
mean	0.000798	0.000695	-0.001704	0.000255	-0.000282	0.005339
std	0.005063	0.029413	0.028570	0.008601	0.011678	0.073086
min	-0.015981	-0.093950	-0.059488	-0.035909	-0.070854	-0.282511
25%	-0.002109	-0.014942	-0.020041	-0.003238	-0.004816	-0.029821
50%	0.000102	0.002102	0.000000	0.000501	-0.000427	0.000000
75%	0.003778	0.018354	0.011371	0.004641	0.005896	0.045627
max	0.020528	0.081514	0.090078	0.024377	0.031033	0.197044

	X10Y_TBY	EURUSD	YEAR
count	125.000000	125.000000	125.0
mean	0.002086	-0.000396	2016.0
std	0.027133	0.005246	0.0
min	-0.092007	-0.013065	2016.0

```
25%     -0.015240  -0.003512  2016.0
50%      0.000000  -0.000938  2016.0
75%      0.017162   0.002272  2016.0
max      0.112782   0.017844  2016.0
```

[38]:
```
test.describe()
```

[38]:
```
              DXY      METALS         OIL      US_STK    INTL_STK     X13W_TB  \
count  125.000000  125.000000  125.000000  125.000000  125.000000  125.000000
mean    -0.000476    0.004697    0.005549    0.000719    0.000758    0.010594
std      0.004352    0.032281    0.030456    0.007865    0.009341    0.089800
min     -0.016011   -0.082299   -0.067112   -0.023992   -0.023583   -0.225694
25%     -0.002824   -0.017177   -0.008769   -0.002475   -0.003527   -0.023256
50%     -0.000205    0.005514    0.000522    0.000291    0.000409    0.000000
75%      0.002405    0.025668    0.019947    0.004425    0.005788    0.048485
max      0.011420    0.104278    0.123235    0.023507    0.024597    0.473988

          X10Y_TBY      EURUSD    YEAR
count  125.000000  125.000000   125.0
mean    -0.000882    0.000158  2016.0
std      0.021867    0.005518     0.0
min     -0.075364   -0.025438  2016.0
25%     -0.012515   -0.003278  2016.0
50%      0.000632    0.000482  2016.0
75%      0.011546    0.002989  2016.0
max      0.063260    0.017237  2016.0
```

[39]:
```
# OLS Regression
ols_final = sm.OLS(train_y, sm.add_constant(train_x)).fit()
print(ols_final.summary2().tables[1])  # print only coefficients

# Compute test R^2 and test mean squared error
ols_pred = ols_final.predict(sm.add_constant(test_x))
ols_pred = pd.DataFrame(ols_pred, columns=["ols_p"])
ols_actual = test.DXY

ols_rss = np.sum(np.power(ols_pred.ols_p - ols_actual, 2))
ols_tss = np.sum(np.power(ols_actual - np.mean(ols_actual), 2))
ols_rsq = 1 - (ols_rss / ols_tss)
print("\n OLS_R^2", ols_rsq)

ols_MSE = np.sqrt(ols_rss / test.shape[0])
print(" OLS_SME", ols_MSE)
```

```
            Coef.  Std.Err.         t          P>|t|    [0.025    0.975]
const    0.000620  0.000383  1.617653  1.084075e-01 -0.000139  0.001378
METALS  -0.053341  0.014154 -3.768626  2.577904e-04 -0.081370 -0.025312
```

```
US_STK     0.381644  0.093239  4.093181  7.821219e-05  0.197005  0.566283
INTL_STK -0.367056  0.068249 -5.378157  3.861547e-07 -0.502209 -0.231904
X13W_TB  -0.001994  0.005349 -0.372725  7.100220e-01 -0.012587  0.008599
X10Y_TBY  0.006433  0.016621  0.387055  6.994125e-01 -0.026481  0.039348
EURUSD   -0.028720  0.074309 -0.386497  6.998250e-01 -0.175871  0.118431

 OLS_R^2 0.33835925492018204
 OLS_SME 0.003525914773312663
```

[40]:

```python
OLS_df = pd.DataFrame(ols_final.summary2().tables[1]['Coef.'])
```

[41]:

```python
# LASSO Regression

# generate a sequence of lambdas to try
lambdas = [np.power(10, i) for i in np.arange(8, -8, -0.1)]

# Compile model
lasso_cv = LassoCV(cv=10, alphas=lambdas)
lasso_cv.fit(train_x, train_y)  # Fit Model

# Scale
# train_x_scale = scale(train_x) #In case you want to scale the variables.

# Build final LASSO regression model
lasso_final = Lasso(alpha=lasso_cv.alpha_, fit_intercept=True)
lasso_final.fit(train_x, train_y)

# Print results
# print('Intercept:', lasso_final.intercept_)
print(
    "\n",
    pd.DataFrame(
        (lasso_final.coef_),
        index=['METALS', 'US_STK', 'INTL_STK', 'X13W_TB',
       'X10Y_TBY', 'EURUSD'],
        columns=["Coef."],
    ),
)

# R squared formula and mean squared error
lasso_pred = lasso_final.predict(test_x)
lasso_actual = test.DXY
lasso_rss = np.sum(np.power(lasso_pred - lasso_actual, 2))
lasso_tss = np.sum(np.power(lasso_actual - np.mean(lasso_actual), 2))
lasso_rsq = 1 - lasso_rss / lasso_tss
print("\n LASSO_R^2: ", lasso_rsq)
```

```
lasso_MSE = np.sqrt(lasso_rss / test.shape[0])
print("LASSO_SME: ", lasso_MSE)
```

```
              Coef.
METALS    -0.053353
US_STK     0.380525
INTL_STK -0.366259
X13W_TB   -0.001991
X10Y_TBY   0.006442
EURUSD    -0.028433

 LASSO_R^2:  0.3385069409911685
LASSO_SME:  0.003525521238327993
```

[42]:
```
OLS_df = pd.DataFrame(ols_final.summary2().tables[1]["Coef."]).rename(
    columns={"Coef.": "LS"}
)

OLS_df.rename(index = {"const" : "Intercept"}, inplace = True)

Lasso_df = pd.DataFrame(
    np.insert(lasso_final.coef_, 0, lasso_final.intercept_),
    index=["Intercept", "METALS","US_STK", "INTL_STK", "X13W_TB","X10Y_TBY",␣
 ↪"EURUSD"],
    columns=["Lasso"],
)

df = OLS_df.merge(Lasso_df, left_index=True, right_index=True)

df.append(
    pd.DataFrame(
        {
            "LS": [ols_rsq, ols_MSE],
            "Lasso": [lasso_rsq, lasso_MSE],
        },
        index=["R sq", "Mean Sq. Err"],
    ),
    ignore_index=False,
)
```

[42]:

	LS	Lasso
Intercept	0.000620	0.000620
METALS	-0.053341	-0.053353
US_STK	0.381644	0.380525
INTL_STK	-0.367056	-0.366259
X13W_TB	-0.001994	-0.001991

```
X10Y_TBY      0.006433  0.006442
EURUSD       -0.028720 -0.028433
R sq          0.338359  0.338507
Mean Sq. Err  0.003526  0.003526
```

As you can see that Lasso does slightly better than LS but compared to lesson notes 3, R square value has been improved drastically. R-square values from previous model, OLS LASSO R sq 0.106332 0.141510

Comparing current R-sq value for LS and Lasso, both value are very close to each other but Lasso coefficient of determination is slightly better here.

```
[ ]:
```