Exploring tutorial:
Looking for another regression "better" than OLS
Will be exploring Lasso regression in this assignment to determine if it works better in predicting missing data than using OLS

Recap on previous assignment discussion
From the figures above where predicted and actual values were plotted:
1) S11-DRP - worked well
2) S11-TP - worked okay (within somewhat expected range, but the model had weak R2 and p-value)
3) S11-TSS - worked okay (within somewhat expected range)
4) S12- DRP - worked well
5) S12 - TP - worked well
6) S12 - TSS - did not work at all (some of the predicted TSS concentrations were negative)
In summary, this OLS model worked well for DRP and TP, but not TSS. Another suitable model is needed to predict TSS concentration. A potential approach is to multiple-linear-regression to incorporate several parameters such as precipitation data (antecedent condition) and time of the year (seasonality - land cover density).

So...in this assignment we will be exploring Lasso regression to determine if it works better in predicting missing data (especially TSS) than using OLS

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import linear_model
from scipy import stats
```

In [2]:
```
# import data from excel
data = pd.read_excel('JY_linear_reg_data.xlsx', sheet_name = 'Data')

# drop 'VSS' column, rename the columns into shorter names
data = data.drop(columns=['VSS (mg/L)'])
data = data.rename(columns={'Sample date': 'Date', 'Flow (cms)':'Flow', 'DRP (mg P/L)':'DR
P', 'TP (mg P/L)':'TP', 'TSS (mg/L)':'TSS'})
data.head()
```

Out[2]:

	Date	Site	Flow	DRP	TP	TSS
0	2015-05-19	S11	0.028210	0.018	0.0015	22.333667
1	2015-06-16	S11	0.189993	0.027	0.1770	1006.666667
2	2015-06-30	S11	1.296460	0.203	1.2590	1338.666667
3	2015-08-18	S11	0.131722	0.050	1.3280	2026.000000
4	2015-08-25	S11	0.120204	0.010	3.1180	1722.000000

In [3]:

```
# separate data by site
data_11 = data[data['Site'] == 'S11']
data_12 = data[data['Site'] == 'S12']

# check the number of null values
print('S11')
print(data_11.isnull().sum())
print('S12')
print(data_12.isnull().sum())
```

```
S11
Date    0
Site    0
Flow    0
DRP     3
TP      3
TSS     3
dtype: int64
S12
Date    0
Site    0
Flow    0
DRP     4
TP      4
TSS     4
dtype: int64
```

In [4]:

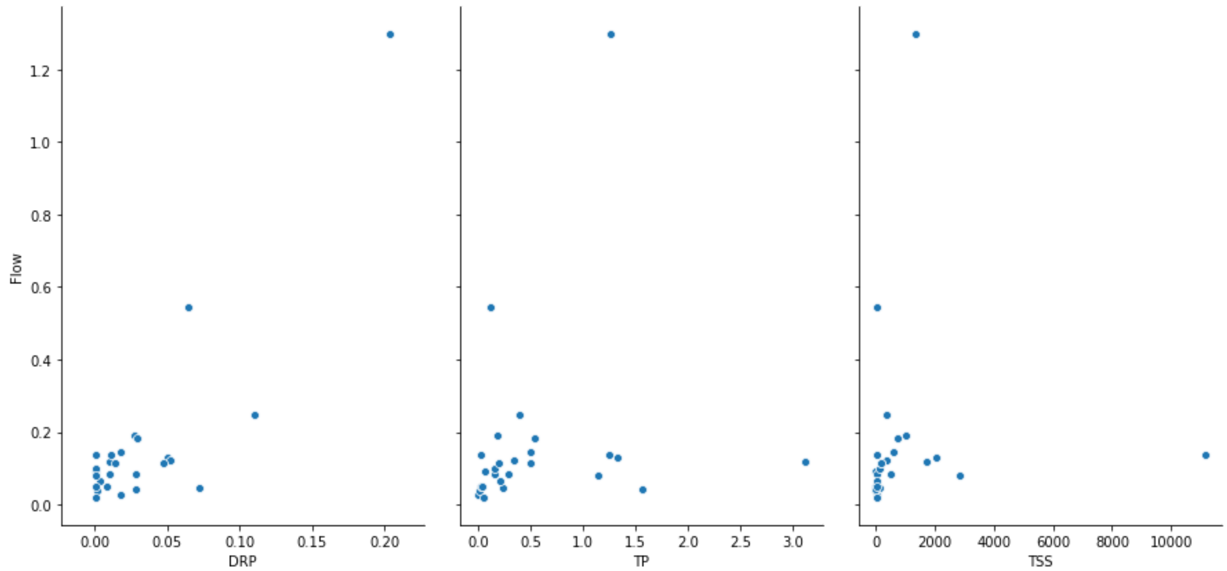
```
# create separate dataframe that only contain null values
data_11_null = data_11[data_11.isnull().any(axis=1)]
data_12_null = data_12[data_12.isnull().any(axis=1)]
print('S11')
print(data_11_null)
print('S12')
print(data_12_null)
```

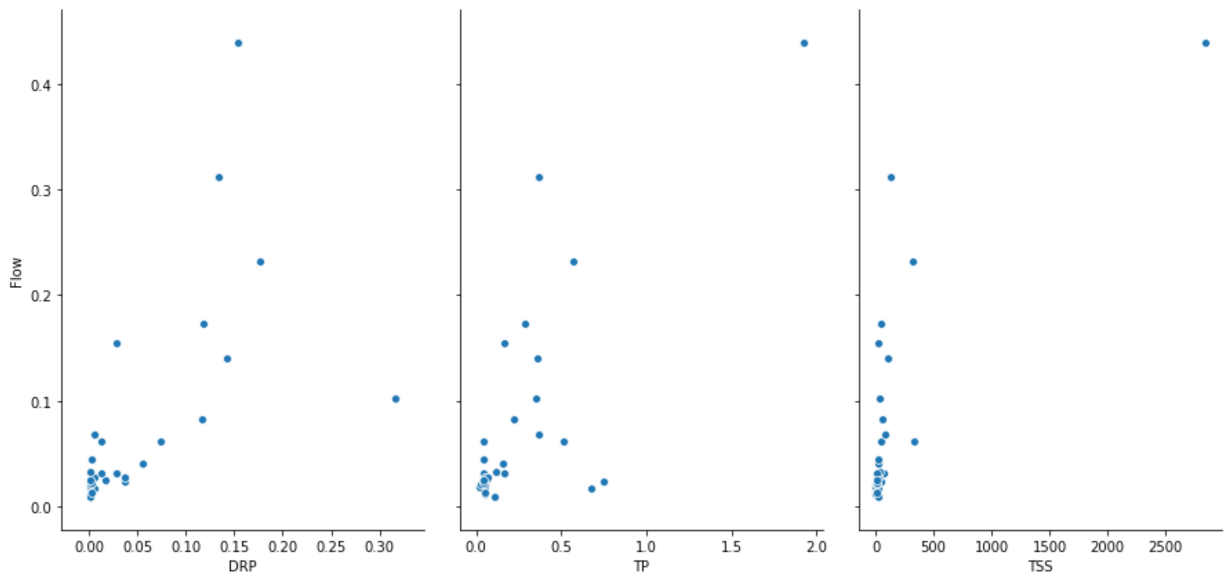
```
S11
          Date Site      Flow  DRP  TP  TSS
21 2018-06-22  S11  0.103414  NaN NaN  NaN
22 2018-06-27  S11  0.189099  NaN NaN  NaN
28 2018-10-11  S11  0.156634  NaN NaN  NaN
S12
          Date Site      Flow  DRP  TP  TSS
59 2018-08-21  S12  0.128687  NaN NaN  NaN
60 2018-08-31  S12  0.028923  NaN NaN  NaN
61 2018-09-06  S12  0.024730  NaN NaN  NaN
62 2018-09-26  S12  0.022594  NaN NaN  NaN
```

In [5]:
```
# drop the null values from the original dataset
data_11 = data_11.dropna()
data_12 = data_12.dropna()
print('S11')
print(data_11.isnull().sum())
print('S12')
print(data_12.isnull().sum())
```

```
S11
Date    0
Site    0
Flow    0
DRP     0
TP      0
TSS     0
dtype: int64
S12
Date    0
Site    0
Flow    0
DRP     0
TP      0
TSS     0
dtype: int64
```

In [6]:
```
# visualize the relationship between flow and responses
sns.pairplot(data_11, x_vars=['DRP', 'TP', 'TSS'], y_vars=['Flow'], height=6, aspect=0.7)
sns.pairplot(data_12, x_vars=['DRP', 'TP', 'TSS'], y_vars=['Flow'], height=6, aspect=0.7)
plt.show()
```





In [27]:
```
# sub 12 DRP
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import numpy as np

x = data_12['Flow']
X = x[:, np.newaxis]
Y = data_12['DRP']

X_norm = stats.shapiro(X) #check if flow data is normally distributed

if X_norm [1] <= 0.05: #if original flow dataset is not normally distributed, then log tra
nsform
    log_X = np.log(X)
    X_norm = stats.shapiro(log_X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, log_Y, test_siz
e = 0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size = 0.25
, random_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))


elif X_norm [1] >= 0.05: #if original flow dataset is normally distributed, move on to che
cking normality of analyte dataset
    X = X  #kind of redundant, just using this so that the indentation for this loop is th
e same as above
    X_norm = stats.shapiro(X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size = 
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

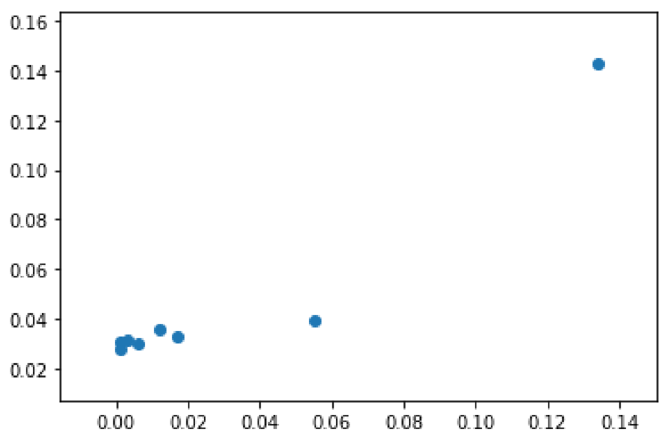
```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))
```




```
X, Y R2 = 0.7258980067912937
```

In [28]:
```
# sub 12 TP
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import numpy as np

x = data_12['Flow']
X = x[:, np.newaxis]
Y = data_12['TP']

X_norm = stats.shapiro(X) #check if flow data is normally distributed

if X_norm [1] <= 0.05: #if original flow dataset is not normally distributed, then log tra
nsform
    log_X = np.log(X)
    X_norm = stats.shapiro(log_X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, log_Y, test_siz
e = 0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size = 0.25
, random_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))


elif X_norm [1] >= 0.05: #if original flow dataset is normally distributed, move on to che
cking normality of analyte dataset
    X = X  #kind of redundant, just using this so that the indentation for this loop is th
e same as above
    X_norm = stats.shapiro(X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

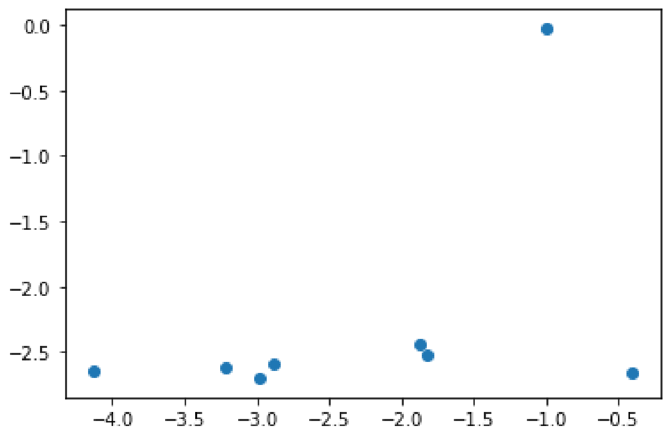
```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))
```



X, log_Y R2 = 0.1042146555248985

In [29]:

```
# sub 12 TSS
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import numpy as np

x = data_12['Flow']
X = x[:, np.newaxis]
Y = data_12['TSS']

X_norm = stats.shapiro(X) #check if flow data is normally distributed

if X_norm [1] <= 0.05: #if original flow dataset is not normally distributed, then log tra
nsform
    log_X = np.log(X)
    X_norm = stats.shapiro(log_X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, log_Y, test_siz
e = 0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size = 0.25
, random_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))


elif X_norm [1] >= 0.05: #if original flow dataset is normally distributed, move on to che
cking normality of analyte dataset
    X = X  #kind of redundant, just using this so that the indentation for this loop is th
e same as above
    X_norm = stats.shapiro(X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

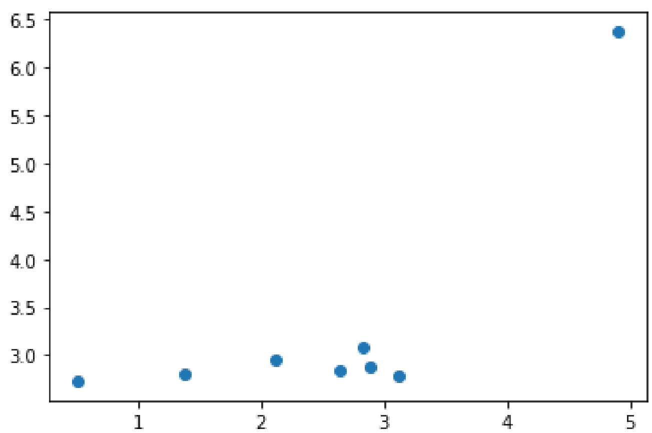
```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))
```



```
X, log_Y R2 = 0.15033624233952803
```

In [33]:
```
# sub 11 DRP
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import numpy as np

x = data_11['Flow']
X = x[:, np.newaxis]
Y = data_11['DRP']

X_norm = stats.shapiro(X) #check if flow data is normally distributed

if X_norm [1] <= 0.05: #if original flow dataset is not normally distributed, then log tra
nsform
    log_X = np.log(X)
    X_norm = stats.shapiro(log_X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, log_Y, test_siz
e = 0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size = 0.25
, random_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))


elif X_norm [1] >= 0.05: #if original flow dataset is normally distributed, move on to che
cking normality of analyte dataset
    X = X  #kind of redundant, just using this so that the indentation for this loop is th
e same as above
    X_norm = stats.shapiro(X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

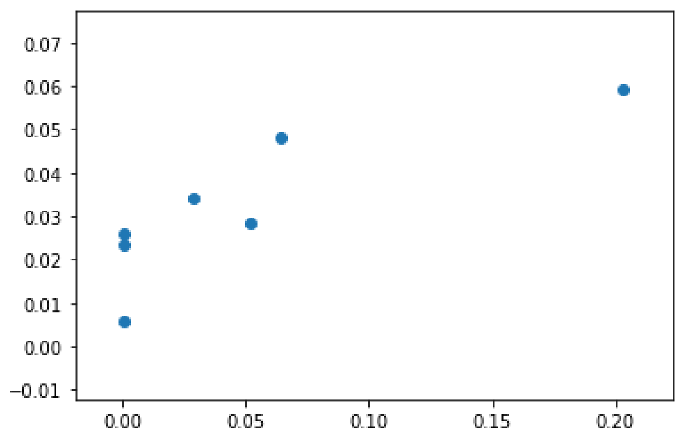
```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))
```




```
log_X, Y R2 = 0.2755098334150292
```

In [34]:
```
# sub 11 TP
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import numpy as np

x = data_11['Flow']
X = x[:, np.newaxis]
Y = data_11['TP']

X_norm = stats.shapiro(X) #check if flow data is normally distributed

if X_norm [1] <= 0.05: #if original flow dataset is not normally distributed, then log tra
nsform
    log_X = np.log(X)
    X_norm = stats.shapiro(log_X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, log_Y, test_siz
e = 0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size = 0.25
, random_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))


elif X_norm [1] >= 0.05: #if original flow dataset is normally distributed, move on to che
cking normality of analyte dataset
    X = X  #kind of redundant, just using this so that the indentation for this loop is th
e same as above
    X_norm = stats.shapiro(X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

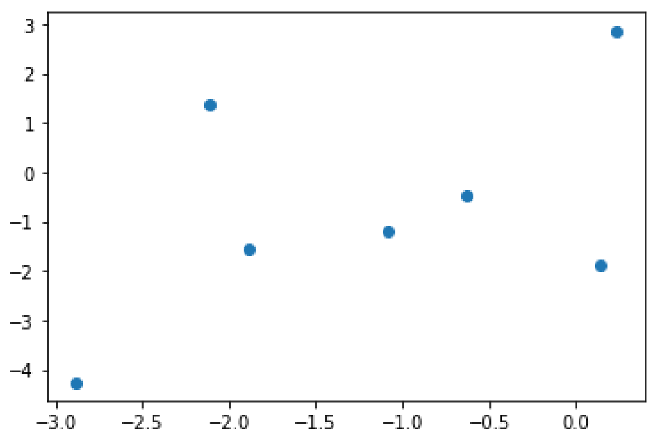
```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))
```




```
log_X, log_Y R2 = -2.0379285248515138
```

In [35]:
```
# sub 11 TSS
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import numpy as np

x = data_11['Flow']
X = x[:, np.newaxis]
Y = data_11['TSS']

X_norm = stats.shapiro(X) #check if flow data is normally distributed

if X_norm [1] <= 0.05: #if original flow dataset is not normally distributed, then log tra
nsform
    log_X = np.log(X)
    X_norm = stats.shapiro(log_X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(log_X, log_Y, test_siz
e = 0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('log_X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
```

```
            X_train, X_test, Y_train, Y_test = train_test_split(log_X, Y, test_size = 0.25
, random_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('log_X, Y','R2 =',r2_score(Y_test, Y_pred))


elif X_norm [1] >= 0.05: #if original flow dataset is normally distributed, move on to che
cking normality of analyte dataset
    X = X  #kind of redundant, just using this so that the indentation for this loop is th
e same as above
    X_norm = stats.shapiro(X)

    if X_norm [1] <= 0.05: #if log-transform flow dataset is not still normally distribute
d, then run the original dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))

    elif X_norm [1] >=0.05: #if log-transform flow data is normally distributed, then only
use the log-transformed dataset
        Y_norm = stats.shapiro(Y)

        if Y_norm [1] <= 0.05: #if original analyte dataset is not normally distributed, t
hen log transform
            log_Y = np.log(Y)
            Y_norm = stats.shapiro(log_Y)

            if Y_norm [1] <= 0.05:   #if log-transform analyte dataset is not normally dis
tributed,then run the original dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25
, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)

                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, Y','R2 =',r2_score(Y_test, Y_pred))
            elif Y_norm [1] >= 0.05: #if log-transform analyte data is normally distribute
d, then only use the log-transformed dataset
                X_train, X_test, Y_train, Y_test = train_test_split(X, log_Y, test_size =
0.25, random_state = 0)

                reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obta
in higher r2

                # train the model
                reg.fit(X_train, Y_train)

                # predict outputs using test dataset
                Y_pred = reg.predict(X_test)
```

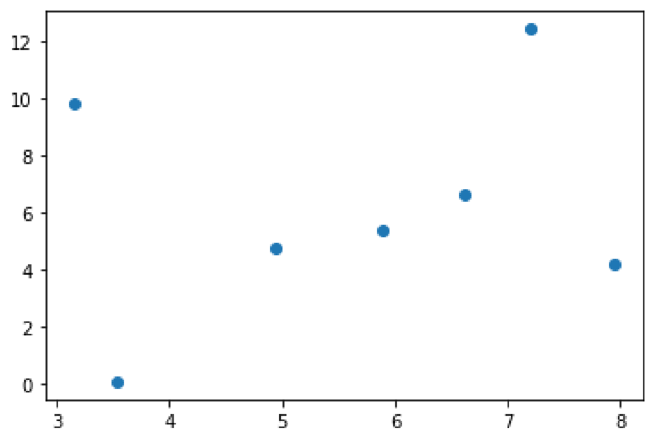
```
                # compare predict vs actual values
                plt.scatter(Y_test, Y_pred)
                plt.show()
                print('X, log_Y','R2 =',r2_score(Y_test, Y_pred))
        elif Y_norm [1] >= 0.05: #if original analyte dataset is  normally distributed, th
en use orginal analyte dataset
            X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, ra
ndom_state = 0)

            reg = linear_model.Lasso(alpha=0.001) # try different alpha values to obtain h
igher r2

            # train the model
            reg.fit(X_train, Y_train)

            # predict outputs using test dataset
            Y_pred = reg.predict(X_test)

            # compare predict vs actual values
            plt.scatter(Y_test, Y_pred)
            plt.show()
            print('X, Y','R2 =',r2_score(Y_test, Y_pred))
```




```
log_X, log_Y R2 = -3.9215690989908722
```

Discussion:

Methods:

The flow and analytes (DRP, TP, TSS) dataset was tested for normality using Shaprio Wilk test:
1) If the dataset is normal, then the original dataset was used in the Lasso model.
2) If the dataset is not normal, then the dataset was log-transformed. It is then tested again for normality. If the log-transformed dataset is normal, then log-transformed dataset was used in the Lasso mode. If the log-transformed dataset is still not normal, then original dataset was used in the Lasso model.
The type of dataset used (original or log-transformed) is indicated below the figure (e.g. log_X, X, log_Y, Y)

Results:

Some of the Lasso model can perdict better than OLS:
S12 DRP: improved r2 from 0.413 (OLS model) to 0.726 (Lasso model)
S12 TP: decreased r2 from 0.558 to 0.104
S12 TSS: decreased r2 from 0.556 to 0.150
S11 DRP: decreased r2 from 0.719 to 0.276
S11 TP: decreased r2 from 0.036 to -2.038 (it's not likely possible to improve model performance using only flow as predictor)
S11 TSS: decreased r2 from 0.003 to -3.922 (it's not likely possible to improve model performance using only flow as predictor)
In summary, multiple models can be tested while tweaking the model parameters (using loop iteration) to obtain the "best" prediction model. In future development, I can run the dataset through different potentially suitable model, and select the best model for each dataset. However, there is no guarantee that any models would work on a certain dataset (unpredictable using flow as the only parameter) such as S11-TP and S11-TSS concentration. Other parameters such as antecedent soil moisture, land cover, and crop type should be considered in the model development.

```
In [ ]:
```