RandomForestRegressor_v1_seg_4

November 24, 2022

1 Random Forest regressor

[ ]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.neural_network import MLPRegressor
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import r2_score, mean_absolute_percentage_error,␣
↪mean_absolute_error, mean_squared_error

from sklearn.ensemble import RandomForestRegressor


from statsmodels.tools.eval_measures import stde
```

1.1 Read the etl info results

[ ]:
```
df_info = pd.read_csv('../dataset_clean/options_csv_v1_etl.csv')
df_info
```

[ ]:
```
   remove_time_features  generic_features  remove_atypical_values  \
0                 False              True                   False

   feature_combination remove_feature_selection  \
0                False                   Lasso

   remove_invalid_correlated_features
0                               False
```

1.2 Read the dataset

[ ]:
```
df = pd.read_csv('../dataset_clean/PlatteRiverWeir_features_v1_clean.csv')
df
```

[ ]:
```
                SensorTime          CaptureTime  Stage  Discharge    grayMean  \
0      2012-06-09 13:15:00  2012-06-09T13:09:07   2.99      916.0   97.405096
1      2012-06-09 13:15:00  2012-06-09T13:10:29   2.99      916.0  104.066757
2      2012-06-09 13:45:00  2012-06-09T13:44:01   2.96      873.0  105.636831
3      2012-06-09 14:45:00  2012-06-09T14:44:30   2.94      846.0  104.418949
4      2012-06-09 15:45:00  2012-06-09T15:44:59   2.94      846.0  106.763541
…                      …                    …    …         …          …
42054  2019-10-11 09:00:00  2019-10-11T08:59:53   2.54      434.0   82.872720
42055  2019-10-11 10:00:00  2019-10-11T09:59:52   2.54      434.0   89.028383
42056  2019-10-11 11:00:00  2019-10-11T10:59:52   2.54      434.0   94.722097
42057  2019-10-11 12:00:00  2019-10-11T11:59:53   2.54      434.0   96.693270
42058  2019-10-11 12:45:00  2019-10-11T12:59:52   2.54      434.0   98.738399

       graySigma       hMean     hSigma
0      39.623303  105.368375  41.572939
1      40.179745  112.399458  41.795584
2      40.533218  114.021526  42.145582
3      41.752678  112.612830  43.575351
4      44.442097  114.839424  46.302008
…            …          …         …
42054  57.702652   87.260572  61.485334
42055  55.840861   94.175906  59.006132
42056  54.355753  100.534577  56.921028
42057  52.787629  102.891159  55.083532
42058  52.025453  105.292067  53.994155

[42059 rows x 8 columns]
```

[ ]:
```
df['SensorTime'] = pd.to_datetime(df['SensorTime'])
df['Year'] = df['SensorTime'].dt.year
df['Month'] = df['SensorTime'].dt.month
```

[ ]:
```
df.dtypes
```

[ ]:
```
SensorTime     datetime64[ns]
CaptureTime            object
Stage                 float64
Discharge             float64
grayMean              float64
graySigma             float64
hMean                 float64
hSigma                float64
Year                    int64
```

```
Month                    int64
dtype: object
```

[ ]:
```
df = df[(df.Stage > 0) & (df.Discharge > 0)]
```

[ ]:
```
df.isna().sum()
```

[ ]:
```
SensorTime     0
CaptureTime    0
Stage          0
Discharge      0
grayMean       0
graySigma      0
hMean          0
hSigma         0
Year           0
Month          0
dtype: int64
```

1.3 Divide dataset to X and Y

[ ]:
```
np.random.seed(0)

df_train = df[(df.Year >= 2012) & (df.Year <= 2017)]
df_train = df_train.iloc[np.random.permutation(len(df_train))]

df_test = df[(df.Year >= 2018) & (df.Year <= 2019)]
```

[ ]:
```
df_train = df_train.drop(columns=["Year", "SensorTime", "CaptureTime"])
#df_val = df_val.drop(columns=["Year", "SensorTime", "CaptureTime"])
df_test = df_test.drop(columns=["Year", "SensorTime", "CaptureTime"])
```

[ ]:
```
y_train = df_train["Stage"]
X_train = df_train.drop(columns=["Stage", "Discharge"])

y_test = df_test["Stage"]
X_test = df_test.drop(columns=["Stage", "Discharge"])
```

[ ]:
```
print(X_train.shape)
print(y_train.shape)
```

```
(27421, 5)
(27421,)
```

[ ]:
```
input_shape = X_train.shape
output_shape = y_train.shape
```

```
print(input_shape, output_shape)
```

```
(27421, 5) (27421,)
```

1.4 Train model

[ ]:
```
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('clf', RandomForestRegressor(random_state=0))
])

param_grid = {'clf__n_estimators': np.arange(50, 300, 1), 'clf__max_features':␣
↪["sqrt", 1.0, "log2"]}

clf = RandomizedSearchCV(pipeline, param_distributions=param_grid, n_iter=20,␣
↪n_jobs=8, verbose=3, scoring="neg_mean_squared_error")
```

[ ]:
```
clf.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
[CV 1/5] END clf__max_features=log2, clf__n_estimators=193;, score=-0.380 total
time=   7.1s
[CV 5/5] END clf__max_features=log2, clf__n_estimators=193;, score=-0.381 total
time=   7.7s
[CV 3/5] END clf__max_features=log2, clf__n_estimators=193;, score=-0.375 total
time=   7.7s
[CV 3/5] END clf__max_features=log2, clf__n_estimators=206;, score=-0.374 total
time=   7.8s
[CV 4/5] END clf__max_features=log2, clf__n_estimators=193;, score=-0.390 total
time=   8.2s
[CV 2/5] END clf__max_features=log2, clf__n_estimators=193;, score=-0.401 total
time=   8.3s
[CV 2/5] END clf__max_features=log2, clf__n_estimators=206;, score=-0.400 total
time=   8.7s
[CV 1/5] END clf__max_features=log2, clf__n_estimators=206;, score=-0.380 total
time=   8.9s
[CV 2/5] END clf__max_features=log2, clf__n_estimators=118;, score=-0.403 total
time=   4.8s
[CV 1/5] END clf__max_features=log2, clf__n_estimators=118;, score=-0.380 total
time=   5.1s
[CV 3/5] END clf__max_features=log2, clf__n_estimators=118;, score=-0.376 total
time=   4.6s
[CV 4/5] END clf__max_features=log2, clf__n_estimators=118;, score=-0.392 total
time=   4.9s
[CV 5/5] END clf__max_features=log2, clf__n_estimators=118;, score=-0.383 total
time=   4.7s
[CV 4/5] END clf__max_features=log2, clf__n_estimators=206;, score=-0.390 total
time=   8.3s
```

```
[CV 5/5] END clf__max_features=log2, clf__n_estimators=206;, score=-0.381 total
time=   8.8s
[CV 1/5] END clf__max_features=log2, clf__n_estimators=91;, score=-0.380 total
time=   3.5s
[CV 2/5] END clf__max_features=log2, clf__n_estimators=91;, score=-0.403 total
time=   3.9s
[CV 3/5] END clf__max_features=log2, clf__n_estimators=91;, score=-0.377 total
time=   3.4s
[CV 4/5] END clf__max_features=log2, clf__n_estimators=91;, score=-0.393 total
time=   3.3s
[CV 1/5] END clf__max_features=1.0, clf__n_estimators=162;, score=-0.383 total
time=  13.2s
[CV 5/5] END clf__max_features=log2, clf__n_estimators=91;, score=-0.384 total
time=   3.7s
[CV 2/5] END clf__max_features=1.0, clf__n_estimators=162;, score=-0.402 total
time=  12.9s
[CV 5/5] END clf__max_features=1.0, clf__n_estimators=162;, score=-0.380 total
time=  13.1s
[CV 3/5] END clf__max_features=1.0, clf__n_estimators=162;, score=-0.375 total
time=  14.2s
[CV 4/5] END clf__max_features=1.0, clf__n_estimators=162;, score=-0.399 total
time=  14.2s
[CV 2/5] END clf__max_features=sqrt, clf__n_estimators=187;, score=-0.401 total
time=   7.5s
[CV 1/5] END clf__max_features=sqrt, clf__n_estimators=187;, score=-0.380 total
time=   8.0s
[CV 3/5] END clf__max_features=sqrt, clf__n_estimators=187;, score=-0.375 total
time=   7.5s
[CV 4/5] END clf__max_features=sqrt, clf__n_estimators=187;, score=-0.390 total
time=   7.6s
[CV 5/5] END clf__max_features=sqrt, clf__n_estimators=187;, score=-0.381 total
time=   8.1s
[CV 1/5] END clf__max_features=1.0, clf__n_estimators=150;, score=-0.383 total
time=  11.9s
[CV 1/5] END clf__max_features=1.0, clf__n_estimators=118;, score=-0.384 total
time=   9.7s
[CV 3/5] END clf__max_features=1.0, clf__n_estimators=150;, score=-0.376 total
time=  12.5s
[CV 5/5] END clf__max_features=1.0, clf__n_estimators=150;, score=-0.380 total
time=  11.6s
[CV 2/5] END clf__max_features=1.0, clf__n_estimators=150;, score=-0.402 total
time=  12.8s
[CV 4/5] END clf__max_features=1.0, clf__n_estimators=150;, score=-0.400 total
time=  12.2s
[CV 2/5] END clf__max_features=1.0, clf__n_estimators=118;, score=-0.402 total
time=  10.1s
[CV 3/5] END clf__max_features=1.0, clf__n_estimators=118;, score=-0.376 total
time=  10.5s
```

```
[CV 2/5] END clf__max_features=log2, clf__n_estimators=166;, score=-0.402 total
time=   6.0s
[CV 3/5] END clf__max_features=log2, clf__n_estimators=166;, score=-0.375 total
time=   6.7s
[CV 4/5] END clf__max_features=log2, clf__n_estimators=166;, score=-0.391 total
time=   6.5s
[CV 1/5] END clf__max_features=log2, clf__n_estimators=166;, score=-0.381 total
time=   7.2s
[CV 5/5] END clf__max_features=log2, clf__n_estimators=166;, score=-0.380 total
time=   6.3s
[CV 4/5] END clf__max_features=1.0, clf__n_estimators=118;, score=-0.401 total
time=  10.1s
[CV 5/5] END clf__max_features=1.0, clf__n_estimators=118;, score=-0.380 total
time=   9.0s
[CV 1/5] END clf__max_features=sqrt, clf__n_estimators=209;, score=-0.380 total
time=   8.2s
[CV 1/5] END clf__max_features=log2, clf__n_estimators=145;, score=-0.380 total
time=   6.3s
[CV 2/5] END clf__max_features=log2, clf__n_estimators=145;, score=-0.402 total
time=   6.0s
[CV 4/5] END clf__max_features=sqrt, clf__n_estimators=209;, score=-0.390 total
time=   8.1s
[CV 2/5] END clf__max_features=sqrt, clf__n_estimators=209;, score=-0.401 total
time=   9.0s
[CV 3/5] END clf__max_features=log2, clf__n_estimators=145;, score=-0.375 total
time=   6.3s
[CV 3/5] END clf__max_features=sqrt, clf__n_estimators=209;, score=-0.375 total
time=   8.3s
[CV 5/5] END clf__max_features=sqrt, clf__n_estimators=209;, score=-0.381 total
time=   9.0s
[CV 4/5] END clf__max_features=log2, clf__n_estimators=145;, score=-0.392 total
time=   6.0s
[CV 1/5] END clf__max_features=sqrt, clf__n_estimators=68;, score=-0.383 total
time=   2.7s
[CV 5/5] END clf__max_features=log2, clf__n_estimators=145;, score=-0.381 total
time=   5.9s
[CV 2/5] END clf__max_features=sqrt, clf__n_estimators=68;, score=-0.403 total
time=   2.7s
[CV 3/5] END clf__max_features=sqrt, clf__n_estimators=68;, score=-0.376 total
time=   3.0s
[CV 4/5] END clf__max_features=sqrt, clf__n_estimators=68;, score=-0.394 total
time=   2.6s
[CV 1/5] END clf__max_features=sqrt, clf__n_estimators=52;, score=-0.383 total
time=   1.9s
[CV 5/5] END clf__max_features=sqrt, clf__n_estimators=68;, score=-0.385 total
time=   2.4s
[CV 2/5] END clf__max_features=sqrt, clf__n_estimators=52;, score=-0.403 total
time=   2.3s
```

```
[CV 1/5] END clf__max_features=log2, clf__n_estimators=290;, score=-0.379 total
time=  10.8s
[CV 3/5] END clf__max_features=sqrt, clf__n_estimators=52;, score=-0.378 total
time=   2.0s
[CV 4/5] END clf__max_features=sqrt, clf__n_estimators=52;, score=-0.394 total
time=   2.0s
[CV 3/5] END clf__max_features=log2, clf__n_estimators=290;, score=-0.374 total
time=  11.7s
[CV 5/5] END clf__max_features=sqrt, clf__n_estimators=52;, score=-0.386 total
time=   2.3s
[CV 2/5] END clf__max_features=log2, clf__n_estimators=290;, score=-0.401 total
time=  12.0s
[CV 5/5] END clf__max_features=log2, clf__n_estimators=290;, score=-0.380 total
time=  12.4s
[CV 4/5] END clf__max_features=log2, clf__n_estimators=290;, score=-0.390 total
time=  12.6s
[CV 1/5] END clf__max_features=sqrt, clf__n_estimators=97;, score=-0.381 total
time=   3.6s
[CV 3/5] END clf__max_features=sqrt, clf__n_estimators=97;, score=-0.377 total
time=   3.6s
[CV 2/5] END clf__max_features=sqrt, clf__n_estimators=97;, score=-0.403 total
time=   3.6s
[CV 4/5] END clf__max_features=sqrt, clf__n_estimators=97;, score=-0.392 total
time=   4.1s
[CV 5/5] END clf__max_features=sqrt, clf__n_estimators=97;, score=-0.383 total
time=   4.2s
[CV 1/5] END clf__max_features=log2, clf__n_estimators=88;, score=-0.380 total
time=   3.4s
[CV 3/5] END clf__max_features=log2, clf__n_estimators=88;, score=-0.378 total
time=   3.4s
[CV 2/5] END clf__max_features=log2, clf__n_estimators=88;, score=-0.403 total
time=   3.7s
[CV 4/5] END clf__max_features=log2, clf__n_estimators=88;, score=-0.394 total
time=   3.4s
[CV 5/5] END clf__max_features=log2, clf__n_estimators=88;, score=-0.384 total
time=   3.3s
[CV 1/5] END clf__max_features=sqrt, clf__n_estimators=146;, score=-0.380 total
time=   6.2s
[CV 1/5] END clf__max_features=1.0, clf__n_estimators=177;, score=-0.384 total
time=  14.4s
[CV 2/5] END clf__max_features=sqrt, clf__n_estimators=146;, score=-0.402 total
time=   5.3s
[CV 4/5] END clf__max_features=1.0, clf__n_estimators=177;, score=-0.399 total
time=  13.9s
[CV 3/5] END clf__max_features=1.0, clf__n_estimators=177;, score=-0.375 total
time=  15.5s
[CV 2/5] END clf__max_features=1.0, clf__n_estimators=177;, score=-0.402 total
time=  15.8s
```

```
[CV 3/5] END clf__max_features=sqrt, clf__n_estimators=146;, score=-0.375 total
time=   5.6s
[CV 5/5] END clf__max_features=1.0, clf__n_estimators=177;, score=-0.379 total
time=  15.4s
[CV 4/5] END clf__max_features=sqrt, clf__n_estimators=146;, score=-0.392 total
time=   6.0s
[CV 5/5] END clf__max_features=sqrt, clf__n_estimators=146;, score=-0.381 total
time=   5.9s
[CV 1/5] END clf__max_features=sqrt, clf__n_estimators=62;, score=-0.383 total
time=   2.5s
[CV 2/5] END clf__max_features=sqrt, clf__n_estimators=62;, score=-0.403 total
time=   2.2s
[CV 4/5] END clf__max_features=sqrt, clf__n_estimators=62;, score=-0.394 total
time=   2.2s
[CV 3/5] END clf__max_features=sqrt, clf__n_estimators=62;, score=-0.377 total
time=   2.6s
[CV 5/5] END clf__max_features=sqrt, clf__n_estimators=62;, score=-0.385 total
time=   2.2s
[CV 1/5] END clf__max_features=1.0, clf__n_estimators=163;, score=-0.383 total
time=  13.0s
[CV 4/5] END clf__max_features=1.0, clf__n_estimators=163;, score=-0.399 total
time=  12.8s
[CV 3/5] END clf__max_features=1.0, clf__n_estimators=163;, score=-0.375 total
time=  13.3s
[CV 2/5] END clf__max_features=1.0, clf__n_estimators=163;, score=-0.402 total
time=  13.9s
[CV 5/5] END clf__max_features=1.0, clf__n_estimators=163;, score=-0.380 total
time=  13.9s
```

```
[ ]: RandomizedSearchCV(estimator=Pipeline(steps=[('scaler', StandardScaler()),
                                                  ('clf',
     RandomForestRegressor(random_state=0))]),
                        n_iter=20, n_jobs=8,
                        param_distributions={'clf__max_features': ['sqrt', 1.0,
                                                                   'log2'],
                                             'clf__n_estimators': array([ 50,  51,
     52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,
             63,  64,  65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,
             76,  77,  78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,
             89,  90…
            219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231,
            232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244,
            245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257,
            258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270,
            271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283,
            284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296,
            297, 298, 299])},
```

```
                    scoring='neg_mean_squared_error', verbose=3)
```

1.5 Test model

[ ]:
```
clf.best_score_
```

[ ]:
```
-0.38479923491844636
```

[ ]:
```
clf.best_params_
```

[ ]:
```
{'clf__n_estimators': 290, 'clf__max_features': 'log2'}
```

[ ]:
```
clf.score(X_test, y_test)
```

[ ]:
```
-0.4719475693452559
```

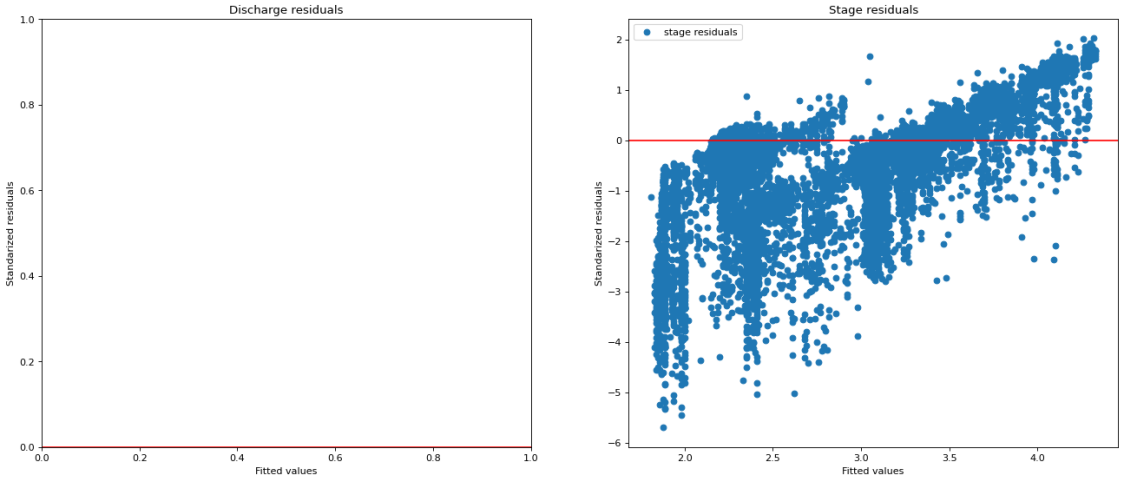
[ ]:
```
y_pred = clf.predict(X_test)
```

[ ]:
```
print("R^2: ", r2_score(y_test, y_pred))
print("mse: ", mean_squared_error(y_test, y_pred))
print("rmse: ", mean_squared_error(y_test, y_pred, squared=False))
print("mae: ", mean_absolute_error(y_test, y_pred))
print("mape: ", mean_absolute_percentage_error(y_test, y_pred))
print("Error estandar: ", stde(y_test.squeeze(),
      y_pred.squeeze(), ddof=2))
```

```
R^2:  -0.20845196214129502
mse:  0.4719475693452559
rmse:  0.6869844025487448
mae:  0.44318905774581835
mape:  0.1755325182997975
Error estandar:  0.6118019412804875
```

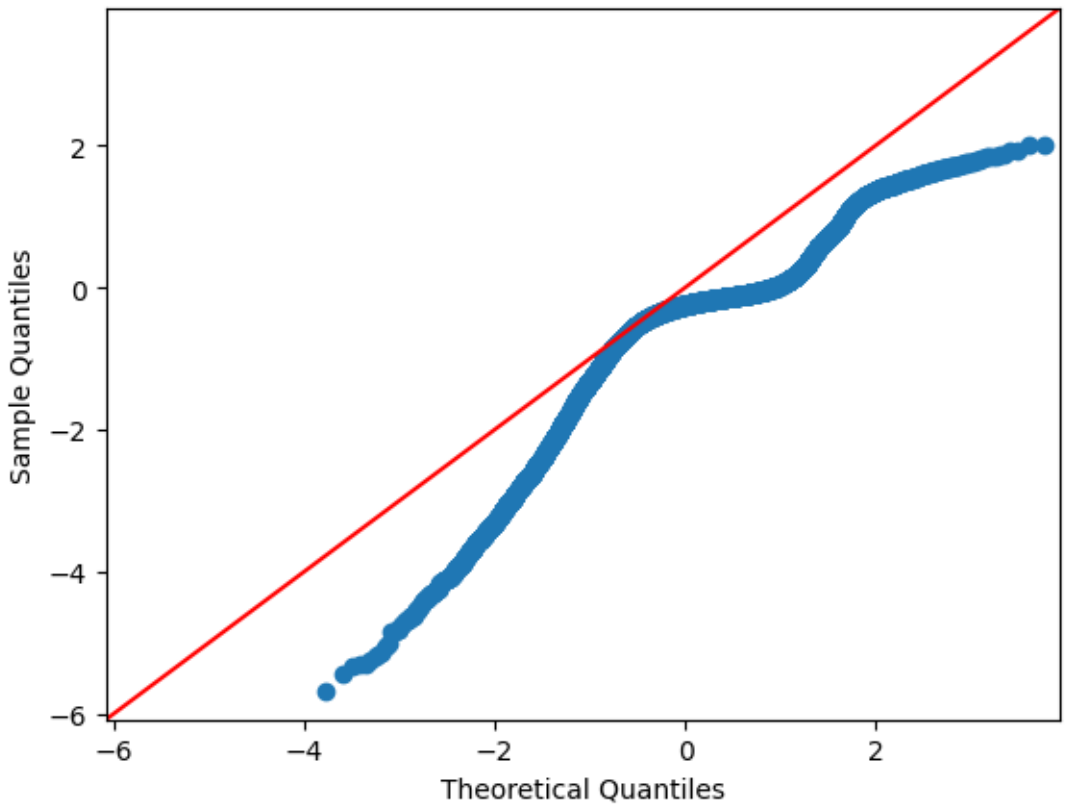
[ ]:
```
residuals = y_test - y_pred
residuals_std = residuals / residuals.std()

y_real_stage = y_test
residual_stage = residuals

#y_real_discharge = np.array([i[-1] for i in y_test])
#residual_discharge = np.array([i[-1] for i in residuals])


figure, ax = plt.subplots(ncols=2, figsize=(20, 8), dpi=80)

ax[1].scatter(y_real_stage, residual_stage / residual_stage.std(), label="stage␣
 ↪residuals")
```

```
#ax[0].scatter(y_real_discharge, residual_discharge / residual_discharge.std(),␣
 ↪label="discharge residuals")
ax[1].axhline(y=0.0, color='r', linestyle='-')
ax[0].axhline(y=0.0, color='r', linestyle='-')

ax[1].set_title("Stage residuals")
ax[0].set_title("Discharge residuals")

ax[1].set_xlabel("Fitted values")
ax[0].set_xlabel("Fitted values")
ax[1].set_ylabel("Standarized residuals")
ax[0].set_ylabel("Standarized residuals")

plt.legend()
plt.show()
```

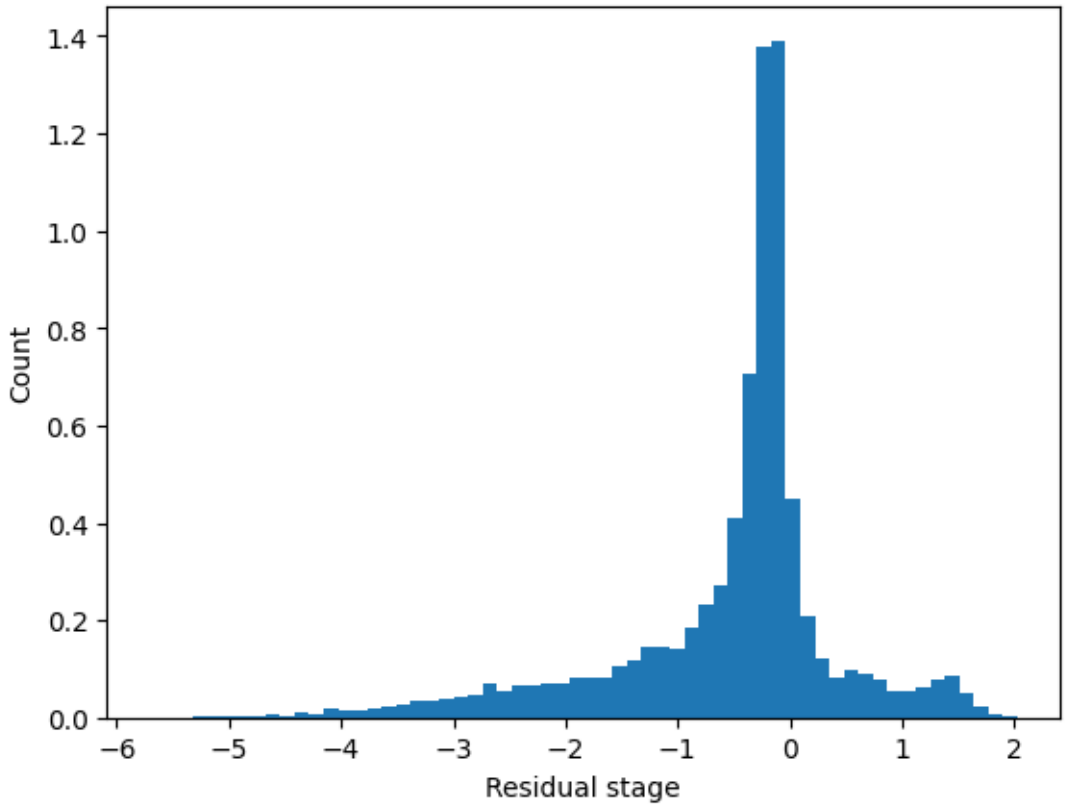


[ ]:
```
import statsmodels.api as sm
from statsmodels.stats.diagnostic import normal_ad

#figure = sm.qqplot(residual_stage / residual_stage.std(), line ='45',␣
 ↪label='stage')
plt.show()
```

[ ]:
```
figure = sm.qqplot(residual_stage / residual_stage.std(), line='45',␣
 ↪label='discharge')
plt.show()
```



[ ]:
```
plt.hist(residual_stage / residual_stage.std(), density=True, bins = 60)
plt.ylabel('Count')
plt.xlabel('Residual stage');
plt.show()
```



[ ]:
```
"""plt.hist(residual_discharge / residual_discharge.std(), density=True, bins =␣
↪60)
plt.ylabel('Count')
plt.xlabel('Residual discharge');
plt.show()"""
```

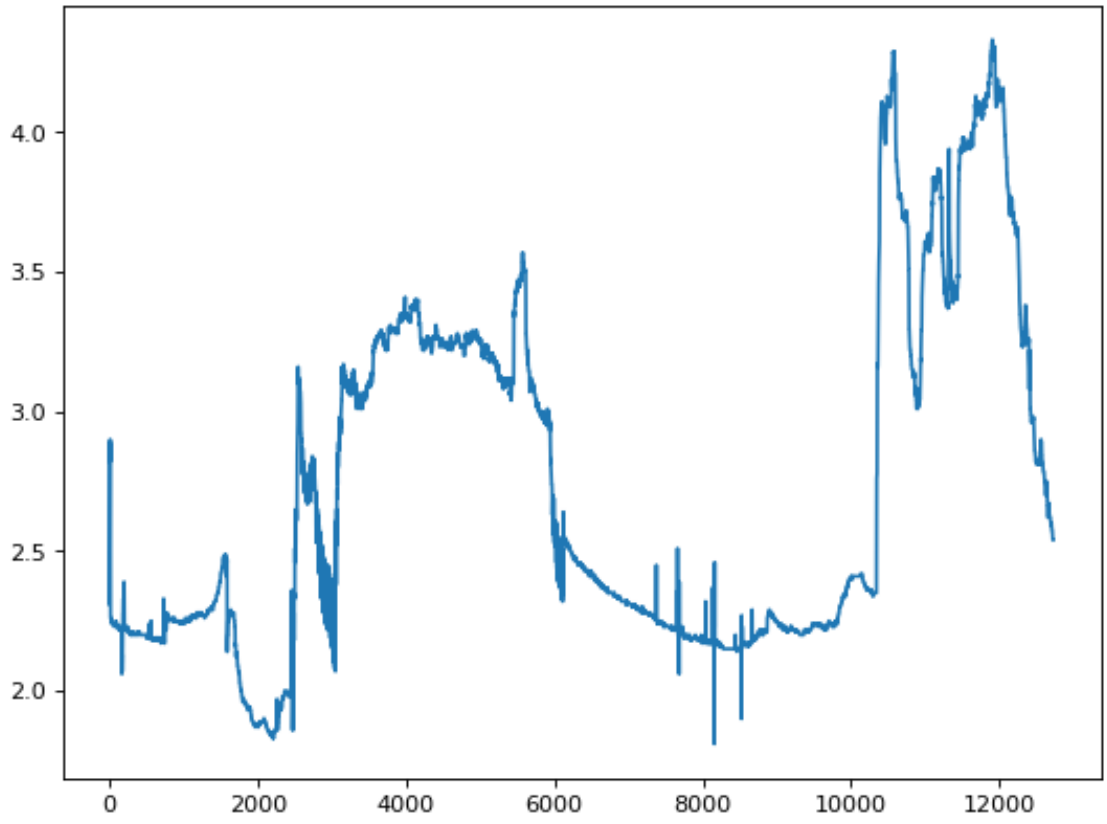
[ ]:
```
"plt.hist(residual_discharge / residual_discharge.std(), density=True, bins =
60)\nplt.ylabel('Count')\nplt.xlabel('Residual discharge');\nplt.show()"
```

[ ]:
```
stat, pval = normal_ad(residual_stage / residual_stage.std())
print("p-value:", pval)

if pval < 0.05:
    print("Hay evidencia de que los residuos no provienen de una distribución␣
↪normal.")
else:
    print("No hay evidencia para rechazar la hipótesis de que los residuos␣
↪vienen de una distribución normal.")
```

```
p-value: 0.0
Hay evidencia de que los residuos no provienen de una distribución normal.
```

[ ]:
```
plt.figure(figsize=(8, 6), dpi=80)
plt.plot(np.arange(len(y_test)), y_test, label="Stage real")
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7f7bc35c40d0>]
```



[ ]:
```
figure, ax = plt.subplots(ncols=2, figsize=(20, 8), dpi=80)

ax[0].plot(np.arange(len(y_test)), y_test, label="Stage real")
ax[0].plot(np.arange(len(y_test)), y_pred, label="Stage pred")

ax[0].set_title("Stage predictions")
ax[1].set_title("Discharge predictions")

ax[1].set_ylabel("Values")
ax[0].set_ylabel("Values")
ax[1].set_xlabel("Time")
ax[0].set_xlabel("Time")

plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```