

linear_regression_v1_6

October 21, 2022

1 Linear regression

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

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import r2_score, mean_absolute_percentage_error, \
    mean_absolute_error, mean_squared_error
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 False 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	grayMean0	hMean0	entropyMean1 \
0	39.623303	105.368375	41.572939	97.084576	106.047217	0.092532
1	40.179745	112.399458	41.795584	105.668610	114.886049	0.090279
2	40.533218	114.021526	42.145582	106.786307	116.053131	0.090561
3	41.752678	112.612830	43.575351	107.674299	117.005027	0.095616
4	44.442097	114.839424	46.302008	114.858589	124.519271	0.101601
...
42054	57.702652	87.260572	61.485334	43.737485	46.616662	0.120668
42055	55.840861	94.175906	59.006132	46.268458	49.716207	0.113951
42056	54.355753	100.534577	56.921028	49.841325	53.984763	0.110346
42057	52.787629	102.891159	55.083532	53.912185	58.857575	0.112571
42058	52.025453	105.292067	53.994155	59.611803	65.697745	0.110247

	entropySigma1	hMean1	WwRawLineMean	WwRawLineSigma \
0	0.632319	169.963345	0.000000	0.000000
1	0.620077	175.220945	0.000000	0.000000
2	0.620853	179.554842	0.000000	0.000000
3	0.651642	180.921521	0.000000	0.000000
4	0.688024	183.131779	0.000000	0.000000
...
42054	0.824195	126.181417	38385.370066	15952.029728
42055	0.783437	131.754200	40162.989292	15467.708856
42056	0.766074	138.014068	42095.946590	16770.357949
42057	0.777376	146.470365	45345.490954	17498.432849
42058	0.760248	156.957374	47877.870782	19963.166359

	WwCurveLineMean	WwCurveLineSigma
0	0.000000	0.000000
1	0.000000	0.000000
2	0.000000	0.000000
3	0.000000	0.000000
4	0.000000	0.000000
...
42054	37550.894823	16444.401209
42055	39397.339095	16009.008049
42056	41350.006568	17489.374617

```
42057      44553.920296      18268.294896
42058      47280.270559      20559.358767
```

```
[42059 rows x 17 columns]
```

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

```
[ ]: df_train = df[(df.Year >= 2012) & (df.Year <= 2017)]
df_test = df[(df.Year >= 2018) & (df.Year <= 2019)]
```

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

1.3 Divide dataset to X and Y

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

```
[ ]: #X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
↳ random_state=0)
```

1.4 Train model

```
[ ]: pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('clf', LinearRegression())
])

folds = KFold(n_splits = 5, shuffle = True, random_state = 100)
clf = cross_val_score(pipeline, X_train, y_train, scoring='r2', cv=folds)
```

```
[ ]: clf
```

```
[ ]: array([0.53525285, 0.50164879, 0.51134036, 0.52088088, 0.51037645])
```

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

```
[ ]: Pipeline(steps=[('scaler', StandardScaler()), ('clf', LinearRegression())])
```

1.5 Test Model

```
[ ]: y_pred = pipeline.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=len(X_train.columns) + 1))
```

```
R^2: 0.373124526987939
mse: 213729.92019354252
rmse: 327.12408905596396
mae: 244.34624629172143
mape: 5.430417554322287e+16
Error estandar: [4.03637584e-01 5.77961421e+02]
```

```
[ ]: residuals = y_test - y_pred
      residuals
```

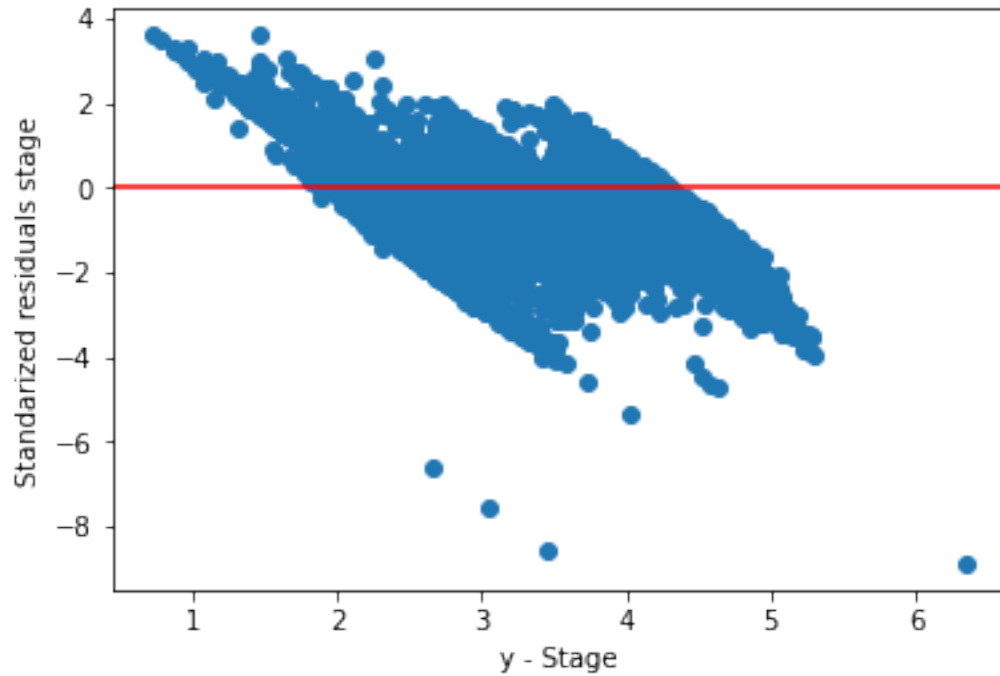
```
[ ]:
      Stage      Discharge
28811  0.708571    587.242215
28812  0.455103    674.541071
28813  0.266308    510.926784
28814 -0.150912   -386.781792
28815 -0.000826   -153.112795
...
42054 -0.734245   -1186.849453
42055 -0.684354   -1138.988056
42056 -0.654575   -1143.137572
42057 -0.712917   -1213.642199
42058 -0.718341   -1186.481649

[13248 rows x 2 columns]
```

```
[ ]: resid = np.array(residuals["Stage"])
      norm_resid = resid / resid.std()

      plt.scatter([i[0] for i in y_pred], norm_resid)
      plt.axhline(y = 0.0, color = 'r', linestyle = '-')
      plt.xlabel("y - Stage")
      plt.ylabel("Standarized residuals stage")
```

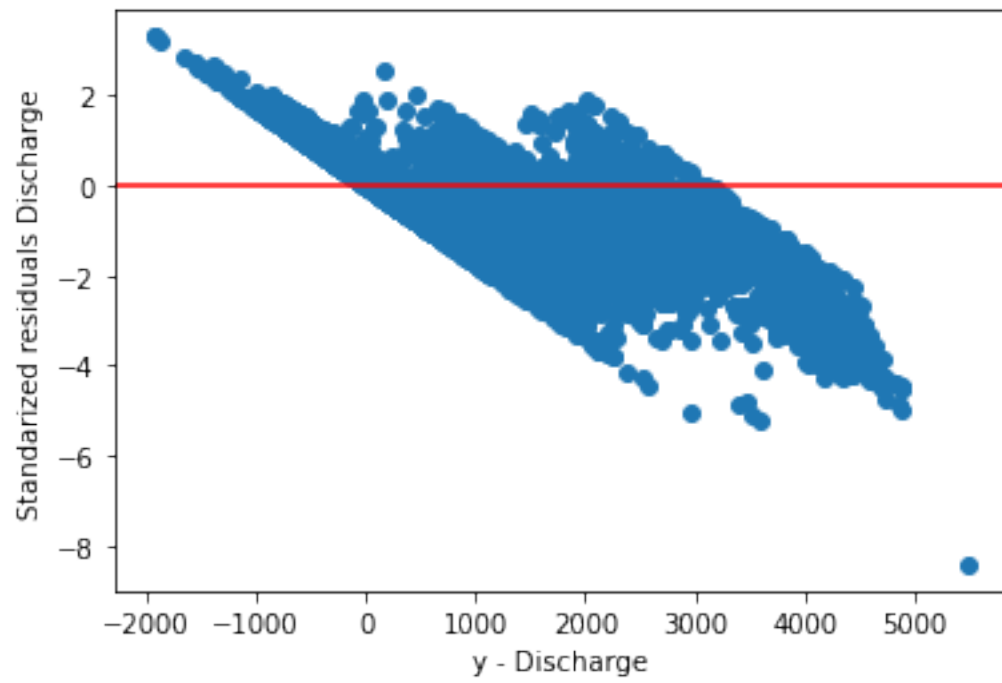
```
[ ]: Text(0, 0.5, 'Standarized residuals stage')
```



```
[ ]: resid = np.array(residuals["Discharge"])
norm_resid = resid / resid.std()

plt.scatter([i[1] for i in y_pred], norm_resid)
plt.axhline(y = 0.0, color = 'r', linestyle = '-')
plt.xlabel("y - Discharge")
plt.ylabel("Standardized residuals Discharge")
```

```
[ ]: Text(0, 0.5, 'Standardized residuals Discharge')
```



[]: