linear_regression_v1_5

October 20, 2022

1 Linear regression

```python
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
from statsmodels.tools.eval_measures import stde
```

1.1 Read the etl info results

```python
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                     False

   remove_invalid_correlated_features
0                               False
```

1.2 Read the dataset

```python
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  entropyMean  entropySigma       hMean      hSigma  ... \
0      39.623303     0.203417      0.979825  105.368375   41.572939  ...
1      40.179745     0.206835      1.002624  112.399458   41.795584  ...
2      40.533218     0.204756      0.994246  114.021526   42.145582  ...
3      41.752678     0.202428      0.983170  112.612830   43.575351  ...
4      44.442097     0.202661      0.989625  114.839424   46.302008  ...
...          ...          ...           ...         ...         ...  ...
42054  57.702652     0.221708      1.076393   87.260572   61.485334  ...
42055  55.840861     0.233168      1.124774   94.175906   59.006132  ...
42056  54.355753     0.240722      1.151833  100.534577   56.921028  ...
42057  52.787629     0.244789      1.171987  102.891159   55.083532  ...
42058  52.025453     0.252812      1.213278  105.292067   53.994155  ...

       WeirPt2X  WeirPt2Y  WwRawLineMin  WwRawLineMax  WwRawLineMean  \
0            -1        -1           0.0           0.0       0.000000
1            -1        -1           0.0           0.0       0.000000
2            -1        -1           0.0           0.0       0.000000
3            -1        -1           0.0           0.0       0.000000
4            -1        -1           0.0           0.0       0.000000
...         ...       ...           ...           ...            ...
42054      2446      1900        9284.0       77521.0   38385.370066
42055      2440      1900       10092.0       74614.0   40162.989292
42056      2447      1900        7067.0       83260.0   42095.946590
42057      2443      1900        6283.0       83045.0   45345.490954
42058      2436      1900        7375.0       89813.0   47877.870782

       WwRawLineSigma  WwCurveLineMin  WwCurveLineMax  WwCurveLineMean  \
0            0.000000             0.0             0.0         0.000000
1            0.000000             0.0             0.0         0.000000
2            0.000000             0.0             0.0         0.000000
3            0.000000             0.0             0.0         0.000000
4            0.000000             0.0             0.0         0.000000
...               ...             ...             ...              ...
42054    15952.029728             0.0         70085.0     37550.894823
42055    15467.708856             0.0         70061.0     39397.339095
42056    16770.357949             0.0         76335.0     41350.006568
```

```
42057        17498.432849                0.0        78882.0      44553.920296
42058        19963.166359                0.0        82630.0      47280.270559

        WwCurveLineSigma
0                0.000000
1                0.000000
2                0.000000
3                0.000000
4                0.000000
...                   ...
42054         16444.401209
42055         16009.008049
42056         17489.374617
42057         18268.294896
42058         20559.358767

[42059 rows x 48 columns]
```

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

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

```python
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

```python
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"])
```

```python
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
 →random_state=0)
```

1.4 Train model

```python
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([0.64457013, 0.62752881, 0.61573169, 0.64012765, 0.6317496 ])
```

```
[ ]: 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_absolute_error(y_test, y_pred))
     print("rmse: ", np.sqrt(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:  0.3088987524584208
mse:  258.93715036714417
rmse:  16.091524177875264
mape:  6.608196898267452e+16
Error estandar:  [3.81475331e-01 5.44333053e+02]
```

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

```
[ ]:           Stage     Discharge
     28811  0.576519    727.453216
     28812  1.366664   1850.210392
     28813  1.022390   1465.949565
     28814  0.296845    263.784384
     28815  0.366077    379.183572
     ...         ...           ...
     42054  0.025829     40.681667
     42055  0.173448    232.445362
     42056  0.200580    211.101055
     42057  0.068820     12.297510
     42058  0.117753    119.375216

     [13248 rows x 2 columns]
```
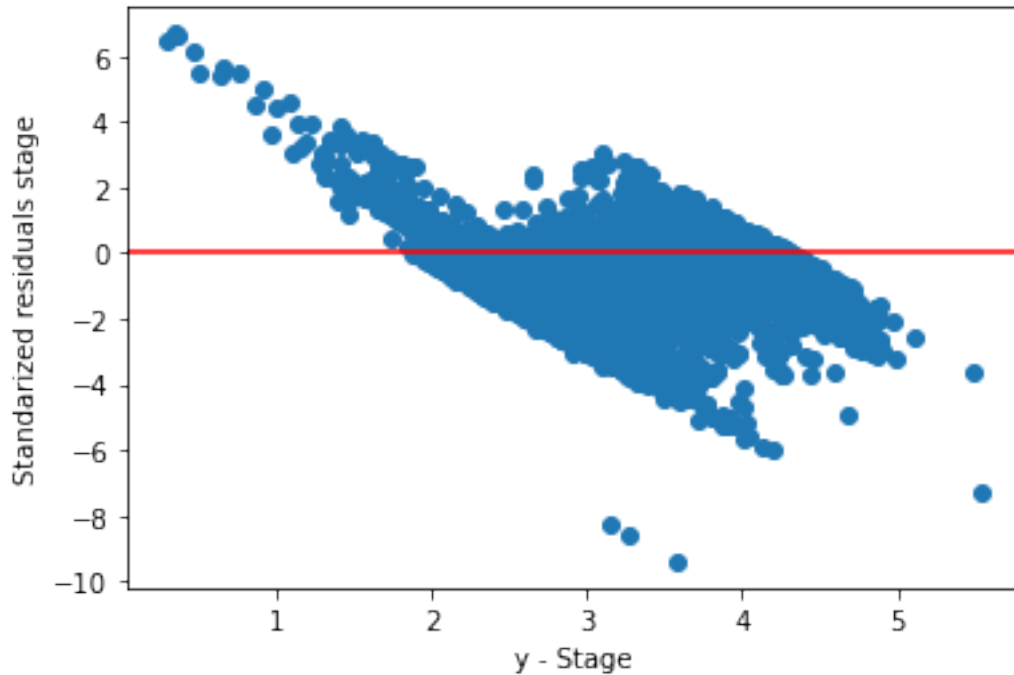
```
[ ]: resid = np.array(residuals["Stage"])
     norm_resid = resid / resid.std()

     plt.scatter([i[0] for i in y_pred], norm_resid)
```

```python
plt.axhline(y = 0.0, color = 'r', linestyle = '-')
plt.xlabel("y - Stage")
plt.ylabel("Standarized residuals stage")
```
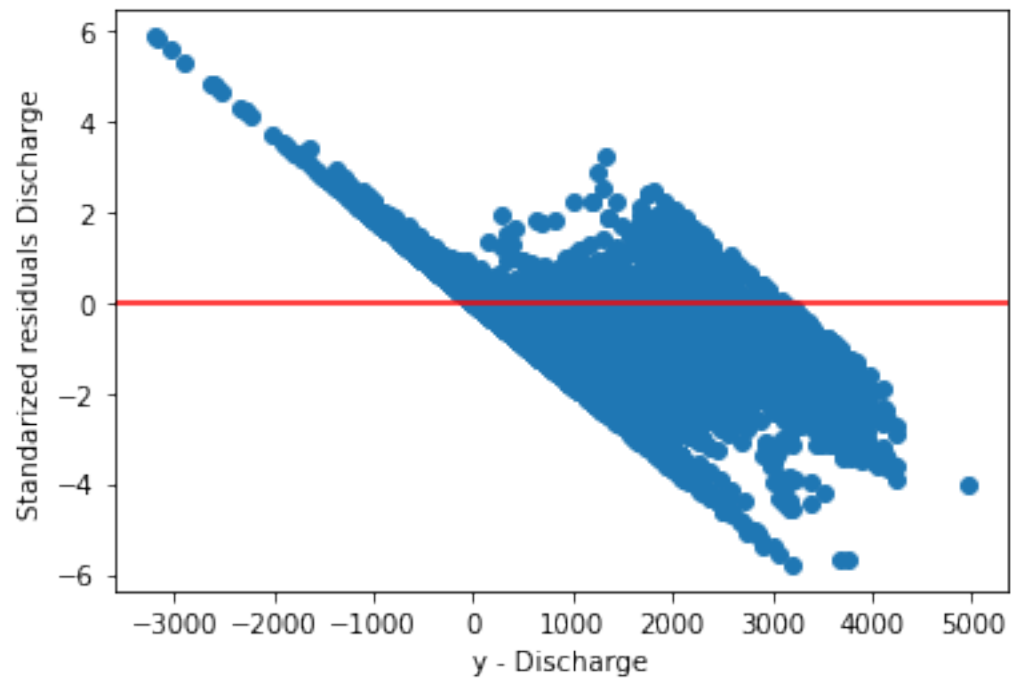
[ ]: Text(0, 0.5, 'Standarized residuals stage')


```python
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("Standarized residuals Discharge")
```

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

[ ]: