MLPRegressor_v1_6

October 21, 2022

1 MLPRegressor

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

     import matplotlib.pyplot as plt

     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 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.dtypes
```

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

1.3 Divide dataset to X and Y

[ ]:
```
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"])
```

[ ]:
```
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', MLPRegressor(shuffle=False, max_iter=2000))
])

#param_grid = {'clf__hidden_layer_sizes': [(10), (10, 20), (10, 5, 15), (20,␣
 ↪30, 10, 15)], 'clf__alpha': np.arange(1e-3, 1, 0.001),␣
 ↪'clf__learning_rate_init': np.arange(1e-3, 0.1, 0.001), 'clf__activation':␣
 ↪['tanh', 'relu']}

param_grid = {'clf__hidden_layer_sizes': [(10), (10, 20), (10, 5, 15), (20, 30,␣
 ↪10, 15)], 'clf__alpha': np.arange(1e-3, 0.1, 0.001), 'clf__activation':␣
 ↪['tanh', 'relu']}

clf = RandomizedSearchCV(pipeline, param_distributions=param_grid, n_iter=10,␣
 ↪n_jobs=10, verbose=3, scoring="neg_mean_squared_error")
```

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

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

[ ]:
```
-801310.3656676637
```

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

[ ]:
```
{'clf__hidden_layer_sizes': (20, 30, 10, 15),
 'clf__alpha': 0.005,
 'clf__activation': 'relu'}
```

1.5 Test model

[ ]:

```
clf.score(X_test, y_test)
```

[ ]:

```
-425408.1781875586
```

[ ]:

```
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=len(X_train.columns) + 1))
```

```
R^2:  -3.0602087176170656
mse:  425408.1781875586
rmse:  461.9982421415349
mae:  395.82005800870127
mape:  9.600147102952742e+16
Error estandar:  [  0.85764752 778.77825615]
```

[ ]:

```

```