

Appendix

April 19, 2018

```
In [1]: import time
import numpy as np
import pandas as pd
pd.options.display.float_format = '{:,.3f}'.format

# %matplotlib notebook
import seaborn as sns
sns.set(style="ticks", color_codes=True)
import matplotlib.pyplot as plt

from sklearn.decomposition import PCA, NMF
from sklearn.manifold import TSNE
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.svm import SVR
from sklearn.ensemble import BaggingRegressor, ExtraTreesRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.neural_network import MLPRegressor

class Data:
    #storage class for data
    def __init__(self, filename, subset_count=None):
        x = pd.read_csv(filename, index_col='id')
        if subset_count is not None: # Nice to use for testing
            permut = np.random.permutation(x.shape[0])[:subset_count]
            x = x.iloc[permut, :]

        self.df_y = x['loss']
        y = x['loss'].values
        del x['loss']

        convertedX = pd.get_dummies(x, drop_first=True)
        X = convertedX.values
```

```

self.X = X
self.y = y

self.df_X = convertedX
self.df_X_test = None

def get_split(self, test_size=0.20, pca_components=None, nmf_components=None):
    X_train, X_test, y_train, y_test = train_test_split(self.X, self.y,
                                                         test_size=test_size)

    if pca_components is not None:
        pca = PCA(n_components=pca_components)
        pca.fit(X_train)
        X_train = pca.transform(X_train)
        X_test = pca.transform(X_test)
    elif nmf_components:
        nmf = NMF(n_components=nmf_components)
        X_train = nmf.fit_transform(X_train)
        X_test = nmf.transform(X_test)
    return X_train, X_test, y_train, y_test

def remove_columns(self, columns):
    columns = tuple(columns)
    for col in self.df_X.columns:
        if col.startswith(columns):
            del self.df_X[col]
    self.X = self.df_X.values

def read_test_data(self, filename):
    X_test = pd.read_csv(filename, index_col='id')
    X_test = pd.get_dummies(X_test, drop_first=True)

    additional_columns = set(X_test.columns) - set(self.df_X.columns)
    X_test = X_test.drop(columns=additional_columns)

    missing_columns = set(self.df_X.columns) - set(X_test.columns)
    for col in missing_columns:
        X_test[col] = 0

    self.df_X_test = X_test
    return X_test.values

def evaluate(name, estimator, X_train, X_test, y_train, y_test):
    t_0 = time.time()
    print(f'{name}:')
    estimator.fit(X_train, y_train)

    t_1 = time.time()
    print(f'\tTime elapsed for model construction {t_1 - t_0:.3f} sec')

```

```

        y_test_predict = estimator.predict(X_test)
        error_test = mean_absolute_error(y_test, y_test_predict)
        error_train = mean_absolute_error(y_train, estimator.predict(X_train))
        print(f'\tTime elapsed for prediction {time.time() - t_1:.3f} sec')
        print(f'\tTest error: {error_test:.3f}')
        print(f'\tTrain error: {error_train:.3f}')
        return error_test

In [ ]: # First overall test
data = Data("train.csv")
data.remove_columns(['cont9', 'cont12', 'cat2', 'cat3', 'cat4', 'cat5', 'cat6', 'cat7'])
X_train, X_test, y_train, y_test = data.get_split()

ESTIMATORS = {
    # Linear
    "LinearRegression": LinearRegression(n_jobs=-1),
    "Ridge": Ridge(),
    "Lasso": Lasso(),
    "ElasticNet": ElasticNet(),
    # Non-linear
    "BaggingRegressor": BaggingRegressor(n_jobs=-1),
    "ExtraTreesRegressor": ExtraTreesRegressor(n_jobs=-1),
    "RandomForestRegressor": RandomForestRegressor(n_jobs=-1),
    "GradientBoostingRegressor": GradientBoostingRegressor(loss='huber'),
    "MLP": MLPRegressor(hidden_layer_sizes=(150,)),
    "KNeighborsRegressor": KNeighborsRegressor(n_jobs=-1),
    "SVR": SVR(),
}

for name, estimator in ESTIMATORS.items():
    evaluate(name, estimator, X_train, X_test, y_train, y_test)

LinearRegression:
    Time elapsed for model construction 23.044 sec
    Time elapsed for prediction 0.478 sec
    Test error: 8798305800.629
    Train error: 1480.444

Ridge:
    Time elapsed for model construction 5.009 sec
    Time elapsed for prediction 0.391 sec
    Test error: 1477.830
    Train error: 1481.719

Lasso:
    Time elapsed for model construction 55.786 sec
    Time elapsed for prediction 0.389 sec
    Test error: 1481.435
    Train error: 1493.583

ElasticNet:

```

```

        Time elapsed for model construction 6.423 sec
        Time elapsed for prediction 0.389 sec
        Test error: 1745.976
        Train error: 1764.948
BaggingRegressor:
    Time elapsed for model construction 136.550 sec
    Time elapsed for prediction 34.328 sec
    Test error: 1519.984
    Train error: 605.857
ExtraTreesRegressor:
    Time elapsed for model construction 279.494 sec
    Time elapsed for prediction 1.537 sec
    Test error: 1535.685
    Train error: 0.072
RandomForestRegressor:
    Time elapsed for model construction 139.742 sec
    Time elapsed for prediction 1.370 sec
    Test error: 1519.550
    Train error: 602.167
GradientBoostingRegressor:
    Time elapsed for model construction 822.191 sec
    Time elapsed for prediction 1.641 sec
    Test error: 1444.904
    Train error: 1453.168
MLP:

```

```

C:\Users\Notandi\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:5
% self.max_iter, ConvergenceWarning)

```

```

        Time elapsed for model construction 2397.275 sec
        Time elapsed for prediction 2.283 sec
        Test error: 1410.916
        Train error: 1392.052
KNeighborsRegressor:
    Time elapsed for model construction 119.529 sec
    Time elapsed for prediction 17134.402 sec
    Test error: 1672.010
    Train error: 1364.940
SVR:

```

```

In [ ]: ESTIMATORS = {
        # Linear
        "LinearRegression": LinearRegression(n_jobs=-1),
        "Ridge": Ridge(),
        "Lasso": Lasso(),

```

```

    "ElasticNet": ElasticNet(),
    # Non-linear
    "BaggingRegressor": BaggingRegressor(n_jobs=-1),
    "ExtraTreesRegressor": ExtraTreesRegressor(n_jobs=-1),
    "RandomForestRegressor": RandomForestRegressor(n_jobs=-1),
    "GradientBoostingRegressor": GradientBoostingRegressor(loss='huber'),
    "MLP": MLPRegressor(hidden_layer_sizes=(150,)),
}

# Test PCA and NMF
data = Data("train.csv")

X_train, X_test, y_train, y_test = data.get_split(pca_components=120)
print(f'PCA with 120 components')
for name, estimator in ESTIMATORS.items():
    evaluate(name, estimator, X_train, X_test, y_train, y_test)

X_train, X_test, y_train, y_test = data.get_split(nmf_components=90)
print(f'NMF with 90 components')
for name, estimator in ESTIMATORS.items():
    evaluate(name, estimator, X_train, X_test, y_train, y_test)

```

PCA with 120 components

LinearRegression:

```

Time elapsed for model construction 1.222 sec
Time elapsed for prediction 0.052 sec
Test error: 1336.491
Train error: 1326.450

```

Ridge:

```

Time elapsed for model construction 0.375 sec
Time elapsed for prediction 0.047 sec
Test error: 1336.488
Train error: 1326.447

```

Lasso:

```

Time elapsed for model construction 0.735 sec
Time elapsed for prediction 0.050 sec
Test error: 1335.520
Train error: 1325.040

```

ElasticNet:

```

Time elapsed for model construction 0.688 sec
Time elapsed for prediction 0.059 sec
Test error: 1503.928
Train error: 1487.781

```

BaggingRegressor:

```

Time elapsed for model construction 159.103 sec
Time elapsed for prediction 7.503 sec
Test error: 1385.234
Train error: 551.043

```

ExtraTreesRegressor:
Time elapsed for model construction 37.692 sec
Time elapsed for prediction 0.782 sec
Test error: 1379.454
Train error: 0.001

RandomForestRegressor:
Time elapsed for model construction 151.161 sec
Time elapsed for prediction 0.754 sec
Test error: 1390.771
Train error: 552.164

GradientBoostingRegressor:
Time elapsed for model construction 333.502 sec
Time elapsed for prediction 0.793 sec
Test error: 1317.153
Train error: 1290.940

MLP:

C:\Users\Notandi\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:5
% self.max_iter, ConvergenceWarning)

Time elapsed for model construction 327.754 sec
Time elapsed for prediction 1.168 sec
Test error: 1224.814
Train error: 1195.139

NMF with 90 components

LinearRegression:
Time elapsed for model construction 0.858 sec
Time elapsed for prediction 0.047 sec
Test error: 1372.249
Train error: 1369.867

Ridge:
Time elapsed for model construction 0.288 sec
Time elapsed for prediction 0.031 sec
Test error: 1367.990
Train error: 1366.349

Lasso:
Time elapsed for model construction 1.047 sec
Time elapsed for prediction 0.047 sec
Test error: 1380.740
Train error: 1380.795

ElasticNet:
Time elapsed for model construction 0.509 sec
Time elapsed for prediction 0.047 sec
Test error: 1969.046
Train error: 1961.323

BaggingRegressor:

```

        Time elapsed for model construction 63.872 sec
        Time elapsed for prediction 6.347 sec
        Test error: 1350.245
        Train error: 536.754
ExtraTreesRegressor:
        Time elapsed for model construction 28.637 sec
        Time elapsed for prediction 0.648 sec
        Test error: 1359.542
        Train error: 0.009
RandomForestRegressor:
        Time elapsed for model construction 59.382 sec
        Time elapsed for prediction 0.608 sec
        Test error: 1344.332
        Train error: 537.979
GradientBoostingRegressor:
        Time elapsed for model construction 150.457 sec
        Time elapsed for prediction 0.901 sec
        Test error: 1263.213
        Train error: 1253.464
MLP:

```

```

C:\Users\Notandi\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:5
% self.max_iter, ConvergenceWarning)

```

```

        Time elapsed for model construction 1372.075 sec
        Time elapsed for prediction 6.929 sec
        Test error: 1363.668
        Train error: 1361.900

```

```

In [ ]: data = Data("train.csv")
        X_train, X_test, y_train, y_test = data.get_split()
        for n_estimators in np.r_[np.arange(200, 501, 50), np.arange(500, 1001, 100)]:
            estimator = GradientBoostingRegressor(loss='huber', alpha=0.5,
                                                    n_estimators=n_estimators, max_depth=6,
                                                    learning_rate=0.1, min_samples_leaf=10,
                                                    min_samples_split=10)
            evaluate(f'GBR n_estimators={n_estimators}', estimator, X_train, X_test, y_train, y_test)

GBR n_estimators=200:
        Time elapsed for model construction 9316.946 sec
        Time elapsed for prediction 4.889 sec
        Test error: 1163.757
        Train error: 1086.791
GBR n_estimators=250:
        Time elapsed for model construction 5754.714 sec
        Time elapsed for prediction 4.712 sec

```

```

        Test error: 1162.030
        Train error: 1076.098
GBR n_estimators=300:
    Time elapsed for model construction 6771.478 sec
    Time elapsed for prediction 5.666 sec
    Test error: 1160.265
    Train error: 1066.413
GBR n_estimators=350:
    Time elapsed for model construction 23563.061 sec
    Time elapsed for prediction 7.427 sec
    Test error: 1159.374
    Train error: 1057.450
GBR n_estimators=400:
    Time elapsed for model construction 8711.057 sec
    Time elapsed for prediction 7.002 sec
    Test error: 1158.555
    Train error: 1052.084
GBR n_estimators=450:
    Time elapsed for model construction 9692.810 sec
    Time elapsed for prediction 7.645 sec
    Test error: 1158.072
    Train error: 1046.006
GBR n_estimators=500:
    Time elapsed for model construction 10792.895 sec
    Time elapsed for prediction 8.363 sec
    Test error: 1158.072
    Train error: 1039.049
GBR n_estimators=500:

```

```

In [ ]: data = Data("train.csv")
        X_train, X_test, y_train, y_test = data.get_split()
        for n_estimators in np.arange(600, 1001, 100):
            estimator = GradientBoostingRegressor(loss='huber', alpha=0.5,
                                                    n_estimators=n_estimators, max_depth=6,
                                                    learning_rate=0.1, min_samples_leaf=10,
                                                    min_samples_split=10)
            evaluate(f'GBR n_estimators={n_estimators}', estimator, X_train, X_test, y_train, y_test)

GBR n_estimators=600:

```