main

August 7, 2023

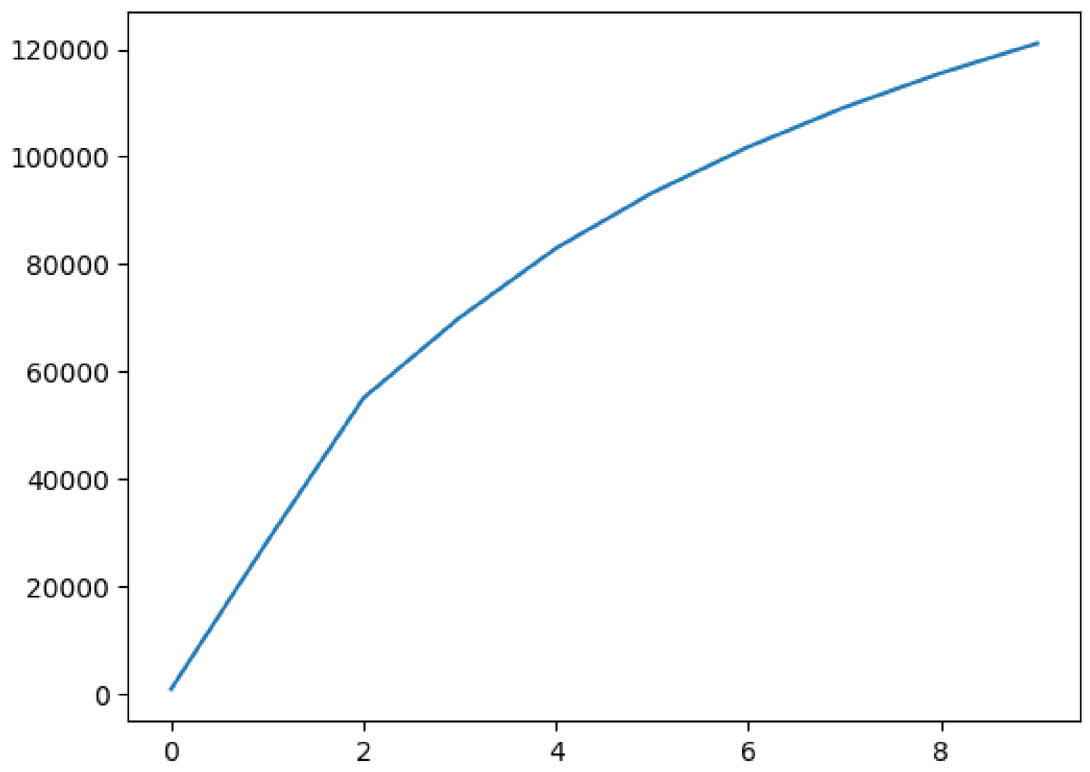
[1]:
```
import pathlib

import h5py
import numpy as np
import matplotlib.pyplot as plt
import sklearn.feature_selection
import sklearn.ensemble
import sklearn.impute
import sklearn.linear_model
import sklearn.svm
import rich
import rich.table

import lib_utils
```

[2]:
```
data = lib_utils.load_split_data("data/original.shuffled_and_split.h5")
```

[3]:
```
nan_counts_features = np.isnan(data["train"]["features"]).sum(axis=0)
```

[4]:
```
plt.plot([(nan_counts_features <= i).sum() for i in range(10)])
```

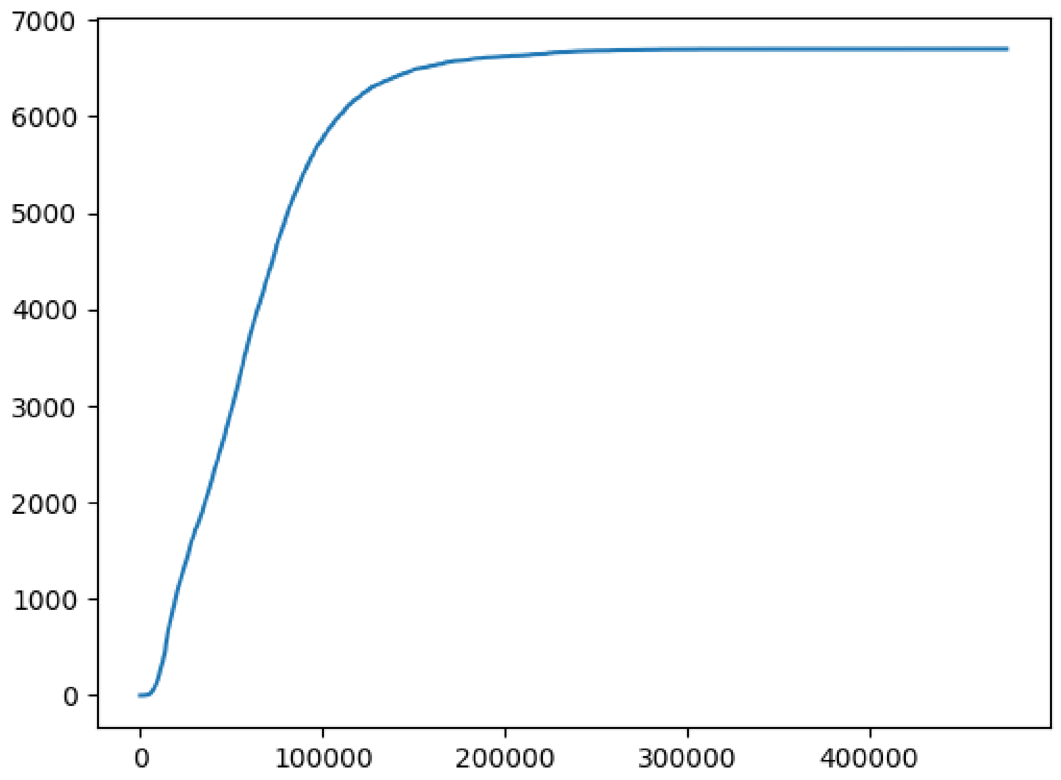
[4]:
```
[<matplotlib.lines.Line2D at 0x7fe3c1de25b0>]
```



```
[5]: nan_counts_ppl = np.isnan(data["train"]["features"]).sum(axis=1)
```

```
[6]: plt.plot([(nan_counts_ppl <= i).sum() for i in range(nan_counts_ppl.max())])
```

```
[6]: [<matplotlib.lines.Line2D at 0x7fe3be9d3610>]
```



```
[7]: fake_data = np.array([[1., 2., .0], [2, 4, .1], [3, 5, .2], [0., 1., .3],]).T

     print(np.nanmean(fake_data, axis=0).shape)
     fake_data -= np.nanmean(fake_data, axis=0)
     fake_data /= np.std(fake_data, axis=0)

     print(fake_data)
     print(fake_data.T.shape)
     np.correlate(np.arange(10) - np.arange(10).mean(), np.arange(10) - np.
      ↪arange(10).mean())
```

```
(4,)
[[ 0.         -0.02093352  0.13545709 -1.03422447]
 [ 1.22474487  1.23507745  1.15138528  1.35244738]
 [-1.22474487 -1.21414393 -1.28684238 -0.31822291]]
(4, 3)
```

```
[7]: array([82.5])
```

```
[8]: data = lib_utils.load_split_data("data/original.shuffled_and_split.h5")
```

[19]:
```
# Nan removal
subset_mask = (np.isnan(data["train"]["features"]).sum(axis=0) < 10)
subset_indices = np.arange(data["train"]["features"].shape[1])[subset_mask]
print(f"{subset_mask.shape = }")
print(f"{subset_indices.shape = }")

subset_data = {
    split: {
        information_type: information_data[:, subset_mask]
        if information_type == "features" else information_data
        for information_type, information_data in split_values.items()
    } for split, split_values in data.items()
}
del data
print(f"{subset_data['train']['features'].shape = }")
```

```
subset_mask.shape = (485512,)
subset_indices.shape = (121099,)
subset_data['train']['features'].shape = (6699, 121099)
```

[20]:
```
# Imputation
imputer = sklearn.impute.SimpleImputer(strategy="mean")
print("Training")
imputer.fit(subset_data["train"]["features"])

print("Transforming")

imputed_subset_data = {
    split: {
        information_type: imputer.transform(information_data)
        if information_type == "features" else information_data
        for information_type, information_data in split_values.items()
    } for split, split_values in subset_data.items()
}
del subset_data
print(f"{imputed_subset_data['train']['features'].shape = }")
```

```
Training
Transforming
imputed_subset_data['train']['features'].shape = (6699, 121099)
```

[21]:
```
# Normalization
mean = np.nanmean(imputed_subset_data["train"]["features"], axis=0,␣
 ↪keepdims=True)
print(mean.shape)
std = np.nanstd(imputed_subset_data["train"]["features"], axis=0, keepdims=True)
print(std.shape)
```

```

normalized_imputed_subset_data = {
    split: {
        information_type: (information_data - mean) / std
        if information_type == "features" else information_data
        for information_type, information_data in split_values.items()
    } for split, split_values in imputed_subset_data.items()
}
del imputed_subset_data
print(normalized_imputed_subset_data["train"]["features"].shape)
```

```
(1, 121099)
(1, 121099)
(6699, 121099)
```

[36]:
```
def prep_data(n_features, normalized_imputed_subset_data):

    feature_selector = sklearn.feature_selection.SelectKBest(
        sklearn.feature_selection.f_regression,
        k=n_features
    ).fit(
        normalized_imputed_subset_data["train"]["features"],
        normalized_imputed_subset_data["train"]["age"],
    )

    feature_selected_data = {
        split: {
            information_type: feature_selector.transform(information_data)
            if information_type == "features" else information_data
            for information_type, information_data in split_values.items()
        } for split, split_values in normalized_imputed_subset_data.items()
    }
    return feature_selected_data


def evaluate(*, regressor, feature_selected_data, n_features, **kwargs):
    preds = regressor.predict(feature_selected_data["validation"]["features"])
    mae = sklearn.metrics.mean_absolute_error(preds,␣
 ↪feature_selected_data['validation']['age'])
    mse = sklearn.metrics.mean_squared_error(preds,␣
 ↪feature_selected_data['validation']['age'])
    medae = sklearn.metrics.median_absolute_error(preds,␣
 ↪feature_selected_data['validation']['age'])

    r2 = regressor.score(
        feature_selected_data['validation']['features'],
        feature_selected_data['validation']['age'],
    )

    print(
        f"### Metrics ### " +
        f"{r2 = :.2f} # " +
        f"{medae = :0.2f} # " +
        f"{mae = :.2f} # " +
        f"{mse = :.2f}" +
        f"\n### HyperParams ### {n_features = } # " +
        " # ".join(f"{k} = {v}" for k, v in kwargs.items())
    )

def train_predict_evaluate(feature_selected_data, n_features, alpha, l1_ratio):
    if alpha == 0:
        regressor = sklearn.linear_model.LinearRegression(
        )
    elif l1_ratio == 0:
        regressor = sklearn.linear_model.Ridge(
            alpha=alpha,
            max_iter=10000,
        )
    elif l1_ratio == 1:
        regressor = sklearn.linear_model.Lasso(
            alpha=alpha,
            max_iter=10000,
        )
    else:
        regressor = sklearn.linear_model.ElasticNet(
            alpha=alpha,
            l1_ratio=l1_ratio,
            max_iter=10000,
        )

    regressor.fit(
        feature_selected_data["train"]["features"],
        feature_selected_data["train"]["age"],
    )

    evaluate(
        regressor=regressor,
        feature_selected_data=feature_selected_data,
        n_features=n_features,
        alpha=alpha,
        l1_ratio=l1_ratio,
    )

    return regressor
```

```python
def train_histogram_gradient_boosting(feature_selected_data, n_features,␣
 ↪learning_rate, max_iter, max_leaf_nodes):
    regressor = sklearn.ensemble.HistGradientBoostingRegressor(
        learning_rate=learning_rate,
        max_iter=max_iter,
        max_leaf_nodes=max_leaf_nodes,
    )

    regressor.fit(
        feature_selected_data["train"]["features"],
        feature_selected_data["train"]["age"],
    )

    evaluate(
        regressor=regressor,
        feature_selected_data=feature_selected_data,
        n_features=n_features,
        learning_rate=learning_rate,
        max_iter=max_iter,
        max_leaf_nodes=max_leaf_nodes,
    )

    return regressor


def train_svm(feature_selected_data, n_features, kernel, degree, C):

    regressor = sklearn.svm.SVR(kernel=kernel, degree=degree, C=C)

    regressor.fit(
        feature_selected_data["train"]["features"],
        feature_selected_data["train"]["age"],
    )

    evaluate(
        regressor=regressor,
        feature_selected_data=feature_selected_data,
        n_features=n_features,
        kernel=kernel,
        degree=degree,
        C=C,
    )

    return regressor
```

```
[37]: table = rich.table.Table(
          "[bold]N_features]",
          "[bold]Alpha",
          "[bold]L1 Ratio",
          "[bold]R2",
          "[bold]MEDIAN AE",
          "[bold]MAE",
          "[bold]MSE",
          show_lines=True,
      )

      for n_features_exp in np.linspace(1, 5, 20):
          n_features = int(10 ** n_features_exp)
          feature_selected_data = prep_data(n_features,␣
        ↪normalized_imputed_subset_data)
          for alpha in np.linspace(0, 0.2, 4):
              for l1_ratio in np.linspace(0, 1, (5 if alpha != 0 else 1)):
                  estimator = train_predict_evaluate(
                      feature_selected_data=feature_selected_data,
                      alpha=alpha,
                      l1_ratio=l1_ratio,
                      n_features=n_features
                  )
                  if hasattr(estimator, "coef_"):
                      print(f"Sparsity %: {np.isclose(estimator.coef_, 0).mean():0.
        ↪0%} sparse")
                      print(f"Sparsity #: {len(estimator.coef_) - np.
        ↪isclose(estimator.coef_, 0).sum()} / {len(estimator.coef_)}")
                      print()
```

```
### Metrics ### r2 = 0.74 # medae = 6.44 # mae = 8.90 # mse = 152.89
### HyperParams ### n_features = 10 # alpha = 0.0 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 10 / 10

### Metrics ### r2 = 0.74 # medae = 6.44 # mae = 8.90 # mse = 152.89
### HyperParams ### n_features = 10 # alpha = 0.06666666666666667 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 10 / 10

### Metrics ### r2 = 0.74 # medae = 6.57 # mae = 8.93 # mse = 153.17
### HyperParams ### n_features = 10 # alpha = 0.06666666666666667 # l1_ratio =
0.25
Sparsity %: 0% sparse
Sparsity #: 10 / 10
```

```
### Metrics ### r2 = 0.74 # medae = 6.51 # mae = 8.92 # mse = 153.08
### HyperParams ### n_features = 10 # alpha = 0.06666666666666667 # l1_ratio =
0.5
Sparsity %: 0% sparse
Sparsity #: 10 / 10

### Metrics ### r2 = 0.74 # medae = 6.45 # mae = 8.92 # mse = 153.09
### HyperParams ### n_features = 10 # alpha = 0.06666666666666667 # l1_ratio =
0.75
Sparsity %: 10% sparse
Sparsity #: 9 / 10

### Metrics ### r2 = 0.74 # medae = 6.43 # mae = 8.92 # mse = 153.24
### HyperParams ### n_features = 10 # alpha = 0.06666666666666667 # l1_ratio =
1.0
Sparsity %: 10% sparse
Sparsity #: 9 / 10

### Metrics ### r2 = 0.74 # medae = 6.44 # mae = 8.90 # mse = 152.89
### HyperParams ### n_features = 10 # alpha = 0.13333333333333333 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 10 / 10

### Metrics ### r2 = 0.74 # medae = 6.64 # mae = 8.99 # mse = 154.14
### HyperParams ### n_features = 10 # alpha = 0.13333333333333333 # l1_ratio =
0.25
Sparsity %: 0% sparse
Sparsity #: 10 / 10

### Metrics ### r2 = 0.74 # medae = 6.65 # mae = 8.96 # mse = 153.79
### HyperParams ### n_features = 10 # alpha = 0.13333333333333333 # l1_ratio =
0.5
Sparsity %: 0% sparse
Sparsity #: 10 / 10

### Metrics ### r2 = 0.74 # medae = 6.51 # mae = 8.94 # mse = 153.54
### HyperParams ### n_features = 10 # alpha = 0.13333333333333333 # l1_ratio =
0.75
Sparsity %: 0% sparse
Sparsity #: 10 / 10

### Metrics ### r2 = 0.74 # medae = 6.43 # mae = 8.94 # mse = 153.65
### HyperParams ### n_features = 10 # alpha = 0.13333333333333333 # l1_ratio =
1.0
Sparsity %: 10% sparse
Sparsity #: 9 / 10
```

```
### Metrics ### r2 = 0.74 # medae = 6.44 # mae = 8.90 # mse = 152.89
### HyperParams ### n_features = 10 # alpha = 0.2 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 10 / 10

### Metrics ### r2 = 0.74 # medae = 6.57 # mae = 9.05 # mse = 155.26
### HyperParams ### n_features = 10 # alpha = 0.2 # l1_ratio = 0.25
Sparsity %: 0% sparse
Sparsity #: 10 / 10

### Metrics ### r2 = 0.74 # medae = 6.60 # mae = 9.01 # mse = 154.69
### HyperParams ### n_features = 10 # alpha = 0.2 # l1_ratio = 0.5
Sparsity %: 0% sparse
Sparsity #: 10 / 10

### Metrics ### r2 = 0.74 # medae = 6.55 # mae = 8.98 # mse = 154.20
### HyperParams ### n_features = 10 # alpha = 0.2 # l1_ratio = 0.75
Sparsity %: 0% sparse
Sparsity #: 10 / 10

### Metrics ### r2 = 0.74 # medae = 6.40 # mae = 8.96 # mse = 154.16
### HyperParams ### n_features = 10 # alpha = 0.2 # l1_ratio = 1.0
Sparsity %: 10% sparse
Sparsity #: 9 / 10

### Metrics ### r2 = 0.77 # medae = 5.89 # mae = 8.30 # mse = 133.68
### HyperParams ### n_features = 16 # alpha = 0.0 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 16 / 16

### Metrics ### r2 = 0.77 # medae = 5.89 # mae = 8.30 # mse = 133.68
### HyperParams ### n_features = 16 # alpha = 0.06666666666666667 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 16 / 16

### Metrics ### r2 = 0.77 # medae = 6.03 # mae = 8.39 # mse = 135.28
### HyperParams ### n_features = 16 # alpha = 0.06666666666666667 # l1_ratio =
0.25
Sparsity %: 0% sparse
Sparsity #: 16 / 16

### Metrics ### r2 = 0.77 # medae = 5.96 # mae = 8.36 # mse = 134.76
### HyperParams ### n_features = 16 # alpha = 0.06666666666666667 # l1_ratio =
0.5
Sparsity %: 0% sparse
Sparsity #: 16 / 16
```

```
### Metrics ### r2 = 0.77 # medae = 6.00 # mae = 8.34 # mse = 134.31
### HyperParams ### n_features = 16 # alpha = 0.06666666666666667 # l1_ratio =
0.75
Sparsity %: 0% sparse
Sparsity #: 16 / 16

### Metrics ### r2 = 0.77 # medae = 5.88 # mae = 8.31 # mse = 134.09
### HyperParams ### n_features = 16 # alpha = 0.06666666666666667 # l1_ratio =
1.0
Sparsity %: 0% sparse
Sparsity #: 16 / 16

### Metrics ### r2 = 0.77 # medae = 5.89 # mae = 8.30 # mse = 133.68
### HyperParams ### n_features = 16 # alpha = 0.13333333333333333 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 16 / 16

### Metrics ### r2 = 0.77 # medae = 6.03 # mae = 8.50 # mse = 137.72
### HyperParams ### n_features = 16 # alpha = 0.13333333333333333 # l1_ratio =
0.25
Sparsity %: 6% sparse
Sparsity #: 15 / 16

### Metrics ### r2 = 0.77 # medae = 6.02 # mae = 8.45 # mse = 136.76
### HyperParams ### n_features = 16 # alpha = 0.13333333333333333 # l1_ratio =
0.5
Sparsity %: 12% sparse
Sparsity #: 14 / 16

### Metrics ### r2 = 0.77 # medae = 5.92 # mae = 8.39 # mse = 135.67
### HyperParams ### n_features = 16 # alpha = 0.13333333333333333 # l1_ratio =
0.75
Sparsity %: 12% sparse
Sparsity #: 14 / 16

### Metrics ### r2 = 0.77 # medae = 5.91 # mae = 8.34 # mse = 134.72
### HyperParams ### n_features = 16 # alpha = 0.13333333333333333 # l1_ratio =
1.0
Sparsity %: 12% sparse
Sparsity #: 14 / 16

### Metrics ### r2 = 0.77 # medae = 5.89 # mae = 8.30 # mse = 133.68
### HyperParams ### n_features = 16 # alpha = 0.2 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 16 / 16

### Metrics ### r2 = 0.76 # medae = 6.22 # mae = 8.59 # mse = 139.96
### HyperParams ### n_features = 16 # alpha = 0.2 # l1_ratio = 0.25
Sparsity %: 0% sparse
Sparsity #: 16 / 16

### Metrics ### r2 = 0.77 # medae = 6.12 # mae = 8.53 # mse = 138.72
### HyperParams ### n_features = 16 # alpha = 0.2 # l1_ratio = 0.5
Sparsity %: 6% sparse
Sparsity #: 15 / 16

### Metrics ### r2 = 0.77 # medae = 6.00 # mae = 8.46 # mse = 137.28
### HyperParams ### n_features = 16 # alpha = 0.2 # l1_ratio = 0.75
Sparsity %: 12% sparse
Sparsity #: 14 / 16

### Metrics ### r2 = 0.77 # medae = 5.91 # mae = 8.37 # mse = 135.63
### HyperParams ### n_features = 16 # alpha = 0.2 # l1_ratio = 1.0
Sparsity %: 12% sparse
Sparsity #: 14 / 16

### Metrics ### r2 = 0.81 # medae = 5.43 # mae = 7.66 # mse = 113.62
### HyperParams ### n_features = 26 # alpha = 0.0 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 26 / 26

### Metrics ### r2 = 0.81 # medae = 5.43 # mae = 7.66 # mse = 113.62
### HyperParams ### n_features = 26 # alpha = 0.06666666666666667 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 26 / 26

### Metrics ### r2 = 0.81 # medae = 5.43 # mae = 7.70 # mse = 114.70
### HyperParams ### n_features = 26 # alpha = 0.06666666666666667 # l1_ratio =
0.25
Sparsity %: 0% sparse
Sparsity #: 26 / 26

### Metrics ### r2 = 0.81 # medae = 5.42 # mae = 7.69 # mse = 114.45
### HyperParams ### n_features = 26 # alpha = 0.06666666666666667 # l1_ratio =
0.5
Sparsity %: 0% sparse
Sparsity #: 26 / 26

### Metrics ### r2 = 0.81 # medae = 5.53 # mae = 7.69 # mse = 114.25
### HyperParams ### n_features = 26 # alpha = 0.06666666666666667 # l1_ratio =
0.75
Sparsity %: 0% sparse
Sparsity #: 26 / 26
```

```
### Metrics ### r2 = 0.81 # medae = 5.44 # mae = 7.68 # mse = 114.14
### HyperParams ### n_features = 26 # alpha = 0.06666666666666667 # l1_ratio =
1.0
Sparsity %: 0% sparse
Sparsity #: 26 / 26

### Metrics ### r2 = 0.81 # medae = 5.43 # mae = 7.66 # mse = 113.62
### HyperParams ### n_features = 26 # alpha = 0.13333333333333333 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 26 / 26

### Metrics ### r2 = 0.80 # medae = 5.48 # mae = 7.76 # mse = 116.46
### HyperParams ### n_features = 26 # alpha = 0.13333333333333333 # l1_ratio =
0.25
Sparsity %: 4% sparse
Sparsity #: 25 / 26

### Metrics ### r2 = 0.80 # medae = 5.41 # mae = 7.75 # mse = 115.91
### HyperParams ### n_features = 26 # alpha = 0.13333333333333333 # l1_ratio =
0.5
Sparsity %: 4% sparse
Sparsity #: 25 / 26

### Metrics ### r2 = 0.81 # medae = 5.38 # mae = 7.74 # mse = 115.42
### HyperParams ### n_features = 26 # alpha = 0.13333333333333333 # l1_ratio =
0.75
Sparsity %: 4% sparse
Sparsity #: 25 / 26

### Metrics ### r2 = 0.81 # medae = 5.51 # mae = 7.74 # mse = 115.17
### HyperParams ### n_features = 26 # alpha = 0.13333333333333333 # l1_ratio =
1.0
Sparsity %: 8% sparse
Sparsity #: 24 / 26

### Metrics ### r2 = 0.81 # medae = 5.43 # mae = 7.66 # mse = 113.62
### HyperParams ### n_features = 26 # alpha = 0.2 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 26 / 26

### Metrics ### r2 = 0.80 # medae = 5.71 # mae = 7.83 # mse = 118.20
### HyperParams ### n_features = 26 # alpha = 0.2 # l1_ratio = 0.25
Sparsity %: 4% sparse
Sparsity #: 25 / 26

### Metrics ### r2 = 0.80 # medae = 5.53 # mae = 7.81 # mse = 117.40
### HyperParams ### n_features = 26 # alpha = 0.2 # l1_ratio = 0.5
```

```
Sparsity %: 8% sparse
Sparsity #: 24 / 26

### Metrics ### r2 = 0.80 # medae = 5.48 # mae = 7.80 # mse = 116.71
### HyperParams ### n_features = 26 # alpha = 0.2 # l1_ratio = 0.75
Sparsity %: 8% sparse
Sparsity #: 24 / 26

### Metrics ### r2 = 0.80 # medae = 5.47 # mae = 7.81 # mse = 116.36
### HyperParams ### n_features = 26 # alpha = 0.2 # l1_ratio = 1.0
Sparsity %: 12% sparse
Sparsity #: 23 / 26

### Metrics ### r2 = 0.83 # medae = 5.30 # mae = 7.30 # mse = 99.78
### HyperParams ### n_features = 42 # alpha = 0.0 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 42 / 42

### Metrics ### r2 = 0.83 # medae = 5.30 # mae = 7.30 # mse = 99.78
### HyperParams ### n_features = 42 # alpha = 0.06666666666666667 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 42 / 42

### Metrics ### r2 = 0.83 # medae = 5.19 # mae = 7.30 # mse = 99.48
### HyperParams ### n_features = 42 # alpha = 0.06666666666666667 # l1_ratio =
0.25
Sparsity %: 0% sparse
Sparsity #: 42 / 42

### Metrics ### r2 = 0.83 # medae = 5.22 # mae = 7.30 # mse = 99.44
### HyperParams ### n_features = 42 # alpha = 0.06666666666666667 # l1_ratio =
0.5
Sparsity %: 2% sparse
Sparsity #: 41 / 42

### Metrics ### r2 = 0.83 # medae = 5.24 # mae = 7.29 # mse = 99.41
### HyperParams ### n_features = 42 # alpha = 0.06666666666666667 # l1_ratio =
0.75
Sparsity %: 7% sparse
Sparsity #: 39 / 42

### Metrics ### r2 = 0.83 # medae = 5.21 # mae = 7.29 # mse = 99.37
### HyperParams ### n_features = 42 # alpha = 0.06666666666666667 # l1_ratio =
1.0
Sparsity %: 10% sparse
Sparsity #: 38 / 42
```

```
### Metrics ### r2 = 0.83 # medae = 5.30 # mae = 7.30 # mse = 99.78
### HyperParams ### n_features = 42 # alpha = 0.13333333333333333 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 42 / 42

### Metrics ### r2 = 0.83 # medae = 5.30 # mae = 7.36 # mse = 100.70
### HyperParams ### n_features = 42 # alpha = 0.13333333333333333 # l1_ratio =
0.25
Sparsity %: 2% sparse
Sparsity #: 41 / 42

### Metrics ### r2 = 0.83 # medae = 5.29 # mae = 7.35 # mse = 100.36
### HyperParams ### n_features = 42 # alpha = 0.13333333333333333 # l1_ratio =
0.5
Sparsity %: 5% sparse
Sparsity #: 40 / 42

### Metrics ### r2 = 0.83 # medae = 5.35 # mae = 7.35 # mse = 100.20
### HyperParams ### n_features = 42 # alpha = 0.13333333333333333 # l1_ratio =
0.75
Sparsity %: 10% sparse
Sparsity #: 38 / 42

### Metrics ### r2 = 0.83 # medae = 5.31 # mae = 7.34 # mse = 100.04
### HyperParams ### n_features = 42 # alpha = 0.13333333333333333 # l1_ratio =
1.0
Sparsity %: 17% sparse
Sparsity #: 35 / 42

### Metrics ### r2 = 0.83 # medae = 5.30 # mae = 7.30 # mse = 99.78
### HyperParams ### n_features = 42 # alpha = 0.2 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 42 / 42

### Metrics ### r2 = 0.83 # medae = 5.32 # mae = 7.43 # mse = 102.41
### HyperParams ### n_features = 42 # alpha = 0.2 # l1_ratio = 0.25
Sparsity %: 2% sparse
Sparsity #: 41 / 42

### Metrics ### r2 = 0.83 # medae = 5.35 # mae = 7.41 # mse = 101.86
### HyperParams ### n_features = 42 # alpha = 0.2 # l1_ratio = 0.5
Sparsity %: 7% sparse
Sparsity #: 39 / 42

### Metrics ### r2 = 0.83 # medae = 5.36 # mae = 7.41 # mse = 101.56
### HyperParams ### n_features = 42 # alpha = 0.2 # l1_ratio = 0.75
Sparsity %: 12% sparse
Sparsity #: 37 / 42

### Metrics ### r2 = 0.83 # medae = 5.34 # mae = 7.41 # mse = 101.37
### HyperParams ### n_features = 42 # alpha = 0.2 # l1_ratio = 1.0
Sparsity %: 24% sparse
Sparsity #: 32 / 42

### Metrics ### r2 = 0.87 # medae = 4.68 # mae = 6.39 # mse = 77.45
### HyperParams ### n_features = 69 # alpha = 0.0 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 69 / 69

### Metrics ### r2 = 0.87 # medae = 4.68 # mae = 6.39 # mse = 77.45
### HyperParams ### n_features = 69 # alpha = 0.06666666666666667 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 69 / 69

### Metrics ### r2 = 0.87 # medae = 4.71 # mae = 6.43 # mse = 78.68
### HyperParams ### n_features = 69 # alpha = 0.06666666666666667 # l1_ratio =
0.25
Sparsity %: 3% sparse
Sparsity #: 67 / 69

### Metrics ### r2 = 0.87 # medae = 4.68 # mae = 6.43 # mse = 78.58
### HyperParams ### n_features = 69 # alpha = 0.06666666666666667 # l1_ratio =
0.5
Sparsity %: 9% sparse
Sparsity #: 63 / 69

### Metrics ### r2 = 0.87 # medae = 4.63 # mae = 6.44 # mse = 78.46
### HyperParams ### n_features = 69 # alpha = 0.06666666666666667 # l1_ratio =
0.75
Sparsity %: 10% sparse
Sparsity #: 62 / 69

### Metrics ### r2 = 0.87 # medae = 4.63 # mae = 6.45 # mse = 78.28
### HyperParams ### n_features = 69 # alpha = 0.06666666666666667 # l1_ratio =
1.0
Sparsity %: 17% sparse
Sparsity #: 57 / 69

### Metrics ### r2 = 0.87 # medae = 4.68 # mae = 6.39 # mse = 77.45
### HyperParams ### n_features = 69 # alpha = 0.13333333333333333 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 69 / 69
```

```
### Metrics ### r2 = 0.86 # medae = 4.70 # mae = 6.51 # mse = 80.67
### HyperParams ### n_features = 69 # alpha = 0.13333333333333333 # l1_ratio =
0.25
Sparsity %: 7% sparse
Sparsity #: 64 / 69

### Metrics ### r2 = 0.86 # medae = 4.72 # mae = 6.51 # mse = 80.42
### HyperParams ### n_features = 69 # alpha = 0.13333333333333333 # l1_ratio =
0.5
Sparsity %: 17% sparse
Sparsity #: 57 / 69

### Metrics ### r2 = 0.86 # medae = 4.67 # mae = 6.52 # mse = 80.05
### HyperParams ### n_features = 69 # alpha = 0.13333333333333333 # l1_ratio =
0.75
Sparsity %: 25% sparse
Sparsity #: 52 / 69

### Metrics ### r2 = 0.87 # medae = 4.66 # mae = 6.53 # mse = 79.81
### HyperParams ### n_features = 69 # alpha = 0.13333333333333333 # l1_ratio =
1.0
Sparsity %: 28% sparse
Sparsity #: 50 / 69

### Metrics ### r2 = 0.87 # medae = 4.68 # mae = 6.39 # mse = 77.45
### HyperParams ### n_features = 69 # alpha = 0.2 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 69 / 69

### Metrics ### r2 = 0.86 # medae = 4.73 # mae = 6.59 # mse = 82.56
### HyperParams ### n_features = 69 # alpha = 0.2 # l1_ratio = 0.25
Sparsity %: 10% sparse
Sparsity #: 62 / 69

### Metrics ### r2 = 0.86 # medae = 4.65 # mae = 6.59 # mse = 82.22
### HyperParams ### n_features = 69 # alpha = 0.2 # l1_ratio = 0.5
Sparsity %: 22% sparse
Sparsity #: 54 / 69

### Metrics ### r2 = 0.86 # medae = 4.69 # mae = 6.60 # mse = 81.93
### HyperParams ### n_features = 69 # alpha = 0.2 # l1_ratio = 0.75
Sparsity %: 28% sparse
Sparsity #: 50 / 69

### Metrics ### r2 = 0.86 # medae = 4.69 # mae = 6.63 # mse = 81.93
### HyperParams ### n_features = 69 # alpha = 0.2 # l1_ratio = 1.0
Sparsity %: 38% sparse
Sparsity #: 43 / 69
```

```
### Metrics ### r2 = 0.89 # medae = 4.35 # mae = 5.92 # mse = 67.85
### HyperParams ### n_features = 112 # alpha = 0.0 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 112 / 112

### Metrics ### r2 = 0.89 # medae = 4.35 # mae = 5.92 # mse = 67.85
### HyperParams ### n_features = 112 # alpha = 0.06666666666666667 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 112 / 112

### Metrics ### r2 = 0.88 # medae = 4.37 # mae = 5.98 # mse = 68.80
### HyperParams ### n_features = 112 # alpha = 0.06666666666666667 # l1_ratio =
0.25
Sparsity %: 2% sparse
Sparsity #: 110 / 112

### Metrics ### r2 = 0.88 # medae = 4.36 # mae = 5.98 # mse = 68.80
### HyperParams ### n_features = 112 # alpha = 0.06666666666666667 # l1_ratio =
0.5
Sparsity %: 8% sparse
Sparsity #: 103 / 112

### Metrics ### r2 = 0.88 # medae = 4.30 # mae = 5.98 # mse = 68.70
### HyperParams ### n_features = 112 # alpha = 0.06666666666666667 # l1_ratio =
0.75
Sparsity %: 18% sparse
Sparsity #: 92 / 112

### Metrics ### r2 = 0.88 # medae = 4.35 # mae = 5.98 # mse = 68.40
### HyperParams ### n_features = 112 # alpha = 0.06666666666666667 # l1_ratio =
1.0
Sparsity %: 27% sparse
Sparsity #: 82 / 112

### Metrics ### r2 = 0.89 # medae = 4.35 # mae = 5.92 # mse = 67.85
### HyperParams ### n_features = 112 # alpha = 0.13333333333333333 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 112 / 112

### Metrics ### r2 = 0.88 # medae = 4.41 # mae = 6.05 # mse = 70.74
### HyperParams ### n_features = 112 # alpha = 0.13333333333333333 # l1_ratio =
0.25
Sparsity %: 6% sparse
Sparsity #: 105 / 112
```

```
### Metrics ### r2 = 0.88 # medae = 4.45 # mae = 6.06 # mse = 70.70
### HyperParams ### n_features = 112 # alpha = 0.13333333333333333 # l1_ratio =
0.5
Sparsity %: 15% sparse
Sparsity #: 95 / 112

### Metrics ### r2 = 0.88 # medae = 4.40 # mae = 6.07 # mse = 70.73
### HyperParams ### n_features = 112 # alpha = 0.13333333333333333 # l1_ratio =
0.75
Sparsity %: 26% sparse
Sparsity #: 83 / 112

### Metrics ### r2 = 0.88 # medae = 4.44 # mae = 6.07 # mse = 70.22
### HyperParams ### n_features = 112 # alpha = 0.13333333333333333 # l1_ratio =
1.0
Sparsity %: 36% sparse
Sparsity #: 72 / 112

### Metrics ### r2 = 0.89 # medae = 4.35 # mae = 5.92 # mse = 67.85
### HyperParams ### n_features = 112 # alpha = 0.2 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 112 / 112

### Metrics ### r2 = 0.88 # medae = 4.33 # mae = 6.13 # mse = 72.86
### HyperParams ### n_features = 112 # alpha = 0.2 # l1_ratio = 0.25
Sparsity %: 6% sparse
Sparsity #: 105 / 112

### Metrics ### r2 = 0.88 # medae = 4.38 # mae = 6.14 # mse = 72.98
### HyperParams ### n_features = 112 # alpha = 0.2 # l1_ratio = 0.5
Sparsity %: 21% sparse
Sparsity #: 88 / 112

### Metrics ### r2 = 0.88 # medae = 4.51 # mae = 6.15 # mse = 72.93
### HyperParams ### n_features = 112 # alpha = 0.2 # l1_ratio = 0.75
Sparsity %: 35% sparse
Sparsity #: 73 / 112

### Metrics ### r2 = 0.88 # medae = 4.57 # mae = 6.15 # mse = 72.56
### HyperParams ### n_features = 112 # alpha = 0.2 # l1_ratio = 1.0
Sparsity %: 48% sparse
Sparsity #: 58 / 112

### Metrics ### r2 = 0.90 # medae = 3.89 # mae = 5.51 # mse = 60.72
### HyperParams ### n_features = 183 # alpha = 0.0 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 183 / 183
```

```
### Metrics ### r2 = 0.90 # medae = 3.89 # mae = 5.51 # mse = 60.72
### HyperParams ### n_features = 183 # alpha = 0.06666666666666667 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 183 / 183

### Metrics ### r2 = 0.90 # medae = 3.89 # mae = 5.46 # mse = 59.59
### HyperParams ### n_features = 183 # alpha = 0.06666666666666667 # l1_ratio =
0.25
Sparsity %: 8% sparse
Sparsity #: 169 / 183

### Metrics ### r2 = 0.90 # medae = 3.97 # mae = 5.47 # mse = 59.66
### HyperParams ### n_features = 183 # alpha = 0.06666666666666667 # l1_ratio =
0.5
Sparsity %: 17% sparse
Sparsity #: 152 / 183

### Metrics ### r2 = 0.90 # medae = 3.92 # mae = 5.48 # mse = 59.67
### HyperParams ### n_features = 183 # alpha = 0.06666666666666667 # l1_ratio =
0.75
Sparsity %: 22% sparse
Sparsity #: 142 / 183

### Metrics ### r2 = 0.90 # medae = 3.98 # mae = 5.51 # mse = 59.85
### HyperParams ### n_features = 183 # alpha = 0.06666666666666667 # l1_ratio =
1.0
Sparsity %: 28% sparse
Sparsity #: 132 / 183

### Metrics ### r2 = 0.90 # medae = 3.89 # mae = 5.51 # mse = 60.72
### HyperParams ### n_features = 183 # alpha = 0.13333333333333333 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 183 / 183

### Metrics ### r2 = 0.90 # medae = 3.92 # mae = 5.51 # mse = 60.51
### HyperParams ### n_features = 183 # alpha = 0.13333333333333333 # l1_ratio =
0.25
Sparsity %: 10% sparse
Sparsity #: 165 / 183

### Metrics ### r2 = 0.90 # medae = 3.91 # mae = 5.54 # mse = 60.90
### HyperParams ### n_features = 183 # alpha = 0.13333333333333333 # l1_ratio =
0.5
Sparsity %: 21% sparse
Sparsity #: 144 / 183
```

```
### Metrics ### r2 = 0.90 # medae = 4.07 # mae = 5.57 # mse = 61.30
### HyperParams ### n_features = 183 # alpha = 0.13333333333333333 # l1_ratio =
0.75
Sparsity %: 34% sparse
Sparsity #: 121 / 183

### Metrics ### r2 = 0.90 # medae = 4.15 # mae = 5.63 # mse = 62.09
### HyperParams ### n_features = 183 # alpha = 0.13333333333333333 # l1_ratio =
1.0
Sparsity %: 46% sparse
Sparsity #: 99 / 183

### Metrics ### r2 = 0.90 # medae = 3.89 # mae = 5.51 # mse = 60.71
### HyperParams ### n_features = 183 # alpha = 0.2 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 183 / 183

### Metrics ### r2 = 0.90 # medae = 3.94 # mae = 5.59 # mse = 62.11
### HyperParams ### n_features = 183 # alpha = 0.2 # l1_ratio = 0.25
Sparsity %: 15% sparse
Sparsity #: 155 / 183

### Metrics ### r2 = 0.89 # medae = 3.99 # mae = 5.62 # mse = 62.69
### HyperParams ### n_features = 183 # alpha = 0.2 # l1_ratio = 0.5
Sparsity %: 31% sparse
Sparsity #: 126 / 183

### Metrics ### r2 = 0.89 # medae = 4.15 # mae = 5.68 # mse = 63.46
### HyperParams ### n_features = 183 # alpha = 0.2 # l1_ratio = 0.75
Sparsity %: 45% sparse
Sparsity #: 101 / 183

### Metrics ### r2 = 0.89 # medae = 4.10 # mae = 5.74 # mse = 64.39
### HyperParams ### n_features = 183 # alpha = 0.2 # l1_ratio = 1.0
Sparsity %: 56% sparse
Sparsity #: 81 / 183

### Metrics ### r2 = 0.91 # medae = 3.63 # mae = 5.15 # mse = 55.94
### HyperParams ### n_features = 297 # alpha = 0.0 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 297 / 297

### Metrics ### r2 = 0.91 # medae = 3.63 # mae = 5.15 # mse = 55.94
### HyperParams ### n_features = 297 # alpha = 0.06666666666666667 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 297 / 297
```

```
### Metrics ### r2 = 0.91 # medae = 3.72 # mae = 5.09 # mse = 54.19
### HyperParams ### n_features = 297 # alpha = 0.06666666666666667 # l1_ratio =
0.25
Sparsity %: 12% sparse
Sparsity #: 260 / 297

### Metrics ### r2 = 0.91 # medae = 3.69 # mae = 5.09 # mse = 54.49
### HyperParams ### n_features = 297 # alpha = 0.06666666666666667 # l1_ratio =
0.5
Sparsity %: 24% sparse
Sparsity #: 226 / 297

### Metrics ### r2 = 0.91 # medae = 3.63 # mae = 5.11 # mse = 55.01
### HyperParams ### n_features = 297 # alpha = 0.06666666666666667 # l1_ratio =
0.75
Sparsity %: 33% sparse
Sparsity #: 198 / 297

### Metrics ### r2 = 0.91 # medae = 3.69 # mae = 5.14 # mse = 55.32
### HyperParams ### n_features = 297 # alpha = 0.06666666666666667 # l1_ratio =
1.0
Sparsity %: 46% sparse
Sparsity #: 161 / 297

### Metrics ### r2 = 0.91 # medae = 3.63 # mae = 5.15 # mse = 55.94
### HyperParams ### n_features = 297 # alpha = 0.13333333333333333 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 297 / 297

### Metrics ### r2 = 0.91 # medae = 3.69 # mae = 5.15 # mse = 55.00
### HyperParams ### n_features = 297 # alpha = 0.13333333333333333 # l1_ratio =
0.25
Sparsity %: 21% sparse
Sparsity #: 235 / 297

### Metrics ### r2 = 0.91 # medae = 3.74 # mae = 5.18 # mse = 55.65
### HyperParams ### n_features = 297 # alpha = 0.13333333333333333 # l1_ratio =
0.5
Sparsity %: 37% sparse
Sparsity #: 187 / 297

### Metrics ### r2 = 0.91 # medae = 3.72 # mae = 5.21 # mse = 56.22
### HyperParams ### n_features = 297 # alpha = 0.13333333333333333 # l1_ratio =
0.75
Sparsity %: 49% sparse
Sparsity #: 152 / 297
```

```
### Metrics ### r2 = 0.90 # medae = 3.65 # mae = 5.25 # mse = 56.38
### HyperParams ### n_features = 297 # alpha = 0.13333333333333333 # l1_ratio =
1.0
Sparsity %: 60% sparse
Sparsity #: 119 / 297

### Metrics ### r2 = 0.91 # medae = 3.63 # mae = 5.15 # mse = 55.94
### HyperParams ### n_features = 297 # alpha = 0.2 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 297 / 297

### Metrics ### r2 = 0.91 # medae = 3.79 # mae = 5.23 # mse = 56.19
### HyperParams ### n_features = 297 # alpha = 0.2 # l1_ratio = 0.25
Sparsity %: 24% sparse
Sparsity #: 225 / 297

### Metrics ### r2 = 0.90 # medae = 3.82 # mae = 5.28 # mse = 57.02
### HyperParams ### n_features = 297 # alpha = 0.2 # l1_ratio = 0.5
Sparsity %: 42% sparse
Sparsity #: 172 / 297

### Metrics ### r2 = 0.90 # medae = 3.76 # mae = 5.33 # mse = 57.62
### HyperParams ### n_features = 297 # alpha = 0.2 # l1_ratio = 0.75
Sparsity %: 56% sparse
Sparsity #: 130 / 297

### Metrics ### r2 = 0.90 # medae = 3.85 # mae = 5.37 # mse = 58.08
### HyperParams ### n_features = 297 # alpha = 0.2 # l1_ratio = 1.0
Sparsity %: 69% sparse
Sparsity #: 91 / 297

### Metrics ### r2 = 0.92 # medae = 3.45 # mae = 4.74 # mse = 47.60
### HyperParams ### n_features = 483 # alpha = 0.0 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 483 / 483

### Metrics ### r2 = 0.92 # medae = 3.45 # mae = 4.74 # mse = 47.60
### HyperParams ### n_features = 483 # alpha = 0.06666666666666667 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 483 / 483

### Metrics ### r2 = 0.92 # medae = 3.41 # mae = 4.68 # mse = 47.82
### HyperParams ### n_features = 483 # alpha = 0.06666666666666667 # l1_ratio =
0.25
Sparsity %: 16% sparse
Sparsity #: 406 / 483
```

```
### Metrics ### r2 = 0.92 # medae = 3.40 # mae = 4.69 # mse = 48.01
### HyperParams ### n_features = 483 # alpha = 0.06666666666666667 # l1_ratio =
0.5
Sparsity %: 31% sparse
Sparsity #: 333 / 483

### Metrics ### r2 = 0.92 # medae = 3.37 # mae = 4.72 # mse = 48.18
### HyperParams ### n_features = 483 # alpha = 0.06666666666666667 # l1_ratio =
0.75
Sparsity %: 41% sparse
Sparsity #: 286 / 483

### Metrics ### r2 = 0.92 # medae = 3.41 # mae = 4.76 # mse = 48.54
### HyperParams ### n_features = 483 # alpha = 0.06666666666666667 # l1_ratio =
1.0
Sparsity %: 52% sparse
Sparsity #: 230 / 483

### Metrics ### r2 = 0.92 # medae = 3.45 # mae = 4.74 # mse = 47.60
### HyperParams ### n_features = 483 # alpha = 0.13333333333333333 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 483 / 483

### Metrics ### r2 = 0.92 # medae = 3.46 # mae = 4.75 # mse = 49.11
### HyperParams ### n_features = 483 # alpha = 0.13333333333333333 # l1_ratio =
0.25
Sparsity %: 28% sparse
Sparsity #: 347 / 483

### Metrics ### r2 = 0.92 # medae = 3.49 # mae = 4.80 # mse = 49.88
### HyperParams ### n_features = 483 # alpha = 0.13333333333333333 # l1_ratio =
0.5
Sparsity %: 45% sparse
Sparsity #: 265 / 483

### Metrics ### r2 = 0.91 # medae = 3.45 # mae = 4.86 # mse = 50.47
### HyperParams ### n_features = 483 # alpha = 0.13333333333333333 # l1_ratio =
0.75
Sparsity %: 59% sparse
Sparsity #: 198 / 483

### Metrics ### r2 = 0.91 # medae = 3.50 # mae = 4.91 # mse = 50.66
### HyperParams ### n_features = 483 # alpha = 0.13333333333333333 # l1_ratio =
1.0
Sparsity %: 71% sparse
Sparsity #: 141 / 483
```

```
### Metrics ### r2 = 0.92 # medae = 3.45 # mae = 4.74 # mse = 47.60
### HyperParams ### n_features = 483 # alpha = 0.2 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 483 / 483

### Metrics ### r2 = 0.91 # medae = 3.45 # mae = 4.83 # mse = 50.51
### HyperParams ### n_features = 483 # alpha = 0.2 # l1_ratio = 0.25
Sparsity %: 34% sparse
Sparsity #: 321 / 483

### Metrics ### r2 = 0.91 # medae = 3.54 # mae = 4.91 # mse = 51.52
### HyperParams ### n_features = 483 # alpha = 0.2 # l1_ratio = 0.5
Sparsity %: 54% sparse
Sparsity #: 223 / 483

### Metrics ### r2 = 0.91 # medae = 3.65 # mae = 4.98 # mse = 52.01
### HyperParams ### n_features = 483 # alpha = 0.2 # l1_ratio = 0.75
Sparsity %: 67% sparse
Sparsity #: 157 / 483

### Metrics ### r2 = 0.91 # medae = 3.71 # mae = 5.07 # mse = 52.42
### HyperParams ### n_features = 483 # alpha = 0.2 # l1_ratio = 1.0
Sparsity %: 79% sparse
Sparsity #: 101 / 483

### Metrics ### r2 = 0.91 # medae = 3.13 # mae = 4.67 # mse = 51.90
### HyperParams ### n_features = 784 # alpha = 0.0 # l1_ratio = 0.0
Sparsity %: 0% sparse
Sparsity #: 784 / 784

### Metrics ### r2 = 0.91 # medae = 3.12 # mae = 4.67 # mse = 51.89
### HyperParams ### n_features = 784 # alpha = 0.06666666666666667 # l1_ratio =
0.0
Sparsity %: 0% sparse
Sparsity #: 784 / 784


/home/mila/g/gagnonju/.main/lib/python3.9/site-
packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 1.806e+03, tolerance: 3.931e+02
  model = cd_fast.enet_coordinate_descent(

### Metrics ### r2 = 0.92 # medae = 3.14 # mae = 4.49 # mse = 47.82
### HyperParams ### n_features = 784 # alpha = 0.06666666666666667 # l1_ratio =
0.25
Sparsity %: 22% sparse
Sparsity #: 613 / 784
```

```
### Metrics ### r2 = 0.92 # medae = 3.26 # mae = 4.52 # mse = 47.52
### HyperParams ### n_features = 784 # alpha = 0.06666666666666667 # l1_ratio =
0.5
Sparsity %: 39% sparse
Sparsity #: 478 / 784
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[37], line 17
     15 for alpha in np.linspace(0, 0.2, 4):
     16     for l1_ratio in np.linspace(0, 1, (5 if alpha != 0 else 1)):
---> 17         estimator = train_predict_evaluate(
     18             feature_selected_data=feature_selected_data,
     19             alpha=alpha,
     20             l1_ratio=l1_ratio,
     21             n_features=n_features
     22         )
     23         if hasattr(estimator, "coef_"):
     24             print(f"Sparsity %: {np.isclose(estimator.coef_, 0).mean():.
 ↪0%} sparse")

Cell In[36], line 63, in train_predict_evaluate(feature_selected_data,␣
 ↪n_features, alpha, l1_ratio)
     56 else:
     57     regressor = sklearn.linear_model.ElasticNet(
     58         alpha=alpha,
     59         l1_ratio=l1_ratio,
     60         max_iter=10000,
     61     )
---> 63 regressor.fit(
     64     feature_selected_data["train"]["features"],
     65     feature_selected_data["train"]["age"],
     66 )
     68 evaluate(
     69     regressor=regressor,
     70     feature_selected_data=feature_selected_data,
   (...)
     73     l1_ratio=l1_ratio,
     74 )
     76 return regressor

File ~/.main/lib/python3.9/site-packages/sklearn/linear_model/
 ↪_coordinate_descent.py:1004, in ElasticNet.fit(self, X, y, sample_weight,␣
 ↪check_input)
   1002 else:
   1003     this_Xy = None
```

```
-> 1004 _, this_coef, this_dual_gap, this_iter = self.path(
   1005     X,
   1006     y[:, k],
   1007     l1_ratio=self.l1_ratio,
   1008     eps=None,
   1009     n_alphas=None,
   1010     alphas=[alpha],
   1011     precompute=precompute,
   1012     Xy=this_Xy,
   1013     copy_X=True,
   1014     coef_init=coef_[k],
   1015     verbose=False,
   1016     return_n_iter=True,
   1017     positive=self.positive,
   1018     check_input=False,
   1019     # from here on **params
   1020     tol=self.tol,
   1021     X_offset=X_offset,
   1022     X_scale=X_scale,
   1023     max_iter=self.max_iter,
   1024     random_state=self.random_state,
   1025     selection=self.selection,
   1026     sample_weight=sample_weight,
   1027 )
   1028 coef_[k] = this_coef[:, 0]
   1029 dual_gaps_[k] = this_dual_gap[0]

File ~/.main/lib/python3.9/site-packages/sklearn/linear_model/
 ↪_coordinate_descent.py:631, in enet_path(X, y, l1_ratio, eps, n_alphas,␣
 ↪alphas, precompute, Xy, copy_X, coef_init, verbose, return_n_iter, positive,␣
 ↪check_input, **params)
    617     model = cd_fast.enet_coordinate_descent_gram(
    618         coef_,
    619         l1_reg,
   (…)
    628         positive,
    629     )
    630 elif precompute is False:
--> 631     model = cd_fast.enet_coordinate_descent(
    632         coef_, l1_reg, l2_reg, X, y, max_iter, tol, rng, random, positi e
    633     )
    634 else:
    635     raise ValueError(
    636         "Precompute should be one of True, False, 'auto' or array-like.␣
 ↪Got %r"
    637         % precompute
    638     )
```

```
KeyboardInterrupt:
```

[38]:
```
### Metrics ### r2 = 0.90 # medae = 3.85 # mae = 5.37 # mse = 58.08
### HyperParams ### {n_features = } # alpha = 0.2 # l1_ratio = 1.0

# n_features = 112 # alpha = 0.2 # l1_ratio = 1.0
alpha = 0.2
l1_ratio = 1.0
n_features = 112
feature_selected_data = prep_data(n_features, normalized_imputed_subset_data)
estimator = train_predict_evaluate(
    feature_selected_data=feature_selected_data,
    alpha=alpha,
    l1_ratio=l1_ratio,
    n_features=n_features
)
```

```
### Metrics ### r2 = 0.88 # medae = 4.57 # mae = 6.15 # mse = 72.56
### HyperParams ### n_features = 112 # alpha = 0.2 # l1_ratio = 1.0
```

[ ]:
```
alpha = 0.1
l1_ratio = 0.6

for i in np.linspace(1, 6, 20):
    n_features = int(i ** 10)
    feature_selected_data = prep_data(n_features,␣
 ↪normalized_imputed_subset_data)

    for j in np.linspace(1, 3, 10):
        max_leaf_nodes = 1 + int(j ** 10)
        estimator = train_histogram_gradient_boosting(
            feature_selected_data, n_features, learning_rate=0.1, max_iter=100,␣
 ↪max_leaf_nodes=max_leaf_nodes
        )


if hasattr(estimator, "coef_"):
    print(f"Sparsity %: {np.isclose(estimator.coef_, 0).mean():0.0%} sparse")
    print(f"Sparsity #: {len(estimator.coef_) - np.isclose(estimator.coef_, 0).
 ↪sum()} / {len(estimator.coef_)}")
```

```
n_features = 1 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 2 r2 = 0.63
medae = 8.85 mae = 11.39 mse = 219.67
n_features = 1 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 8 r2 = 0.63
medae = 8.95 mae = 11.31 mse = 219.00
n_features = 1 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 40 r2 = 0.62
medae = 9.07 mae = 11.40 mse = 222.69
```

```
n_features = 1 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 166 r2 = 0.62
medae = 9.11 mae = 11.42 mse = 223.75
n_features = 1 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 579 r2 = 0.62
medae = 9.11 mae = 11.42 mse = 223.75
n_features = 1 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 1759 r2 =
0.62 medae = 9.11 mae = 11.42 mse = 223.75
n_features = 1 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 4784 r2 =
0.62 medae = 9.11 mae = 11.42 mse = 223.75
n_features = 1 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 11882 r2 =
0.62 medae = 9.11 mae = 11.42 mse = 223.75
n_features = 1 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 27352 r2 =
0.62 medae = 9.11 mae = 11.42 mse = 223.75
n_features = 1 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 59050 r2 =
0.62 medae = 9.11 mae = 11.42 mse = 223.75
n_features = 10 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 2 r2 = 0.77
medae = 6.63 mae = 8.67 mse = 135.88
n_features = 10 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 8 r2 = 0.85
medae = 4.82 mae = 6.69 mse = 87.33
n_features = 10 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 40 r2 = 0.88
medae = 4.19 mae = 5.93 mse = 72.77
n_features = 10 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 166 r2 =
0.88 medae = 4.07 mae = 5.79 mse = 69.74
n_features = 10 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 579 r2 =
0.88 medae = 4.07 mae = 5.76 mse = 70.25
n_features = 10 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 1759 r2 =
0.88 medae = 4.07 mae = 5.76 mse = 70.25
n_features = 10 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 4784 r2 =
0.88 medae = 4.07 mae = 5.76 mse = 70.25
n_features = 10 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 11882 r2 =
0.88 medae = 4.07 mae = 5.76 mse = 70.25
n_features = 10 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 27352 r2 =
0.88 medae = 4.07 mae = 5.76 mse = 70.25
n_features = 10 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 59050 r2 =
0.88 medae = 4.07 mae = 5.76 mse = 70.25
n_features = 68 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 2 r2 = 0.82
medae = 6.06 mae = 7.68 mse = 103.83
n_features = 68 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 8 r2 = 0.90
medae = 4.23 mae = 5.45 mse = 57.31
n_features = 68 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 40 r2 = 0.92
medae = 3.36 mae = 4.68 mse = 45.21
n_features = 68 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 166 r2 =
0.93 medae = 3.03 mae = 4.50 mse = 44.34
n_features = 68 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 579 r2 =
0.92 medae = 3.17 mae = 4.60 mse = 45.28
n_features = 68 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 1759 r2 =
0.92 medae = 3.17 mae = 4.60 mse = 45.28
n_features = 68 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 4784 r2 =
0.92 medae = 3.17 mae = 4.60 mse = 45.28
```

```
n_features = 68 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 11882 r2 =
0.92 medae = 3.17 mae = 4.60 mse = 45.28
n_features = 68 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 27352 r2 =
0.92 medae = 3.17 mae = 4.60 mse = 45.28
n_features = 68 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 59050 r2 =
0.92 medae = 3.17 mae = 4.60 mse = 45.28
n_features = 336 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 2 r2 = 0.85
medae = 5.49 mae = 7.21 mse = 90.65
n_features = 336 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 8 r2 = 0.92
medae = 3.68 mae = 4.91 mse = 48.37
n_features = 336 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 40 r2 =
0.93 medae = 3.01 mae = 4.23 mse = 39.02
n_features = 336 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 166 r2 =
0.94 medae = 2.69 mae = 4.08 mse = 38.39
n_features = 336 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 579 r2 =
0.94 medae = 2.77 mae = 4.11 mse = 38.32
n_features = 336 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 1759 r2 =
0.94 medae = 2.77 mae = 4.11 mse = 38.32
n_features = 336 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 4784 r2 =
0.94 medae = 2.77 mae = 4.11 mse = 38.32
n_features = 336 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 11882 r2 =
0.94 medae = 2.77 mae = 4.11 mse = 38.32
n_features = 336 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 27352 r2 =
0.94 medae = 2.77 mae = 4.11 mse = 38.32
n_features = 336 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 59050 r2 =
0.94 medae = 2.77 mae = 4.11 mse = 38.32
n_features = 1327 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 2 r2 =
0.86 medae = 5.42 mae = 6.95 mse = 84.86
n_features = 1327 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 8 r2 =
0.92 medae = 3.44 mae = 4.78 mse = 46.76
n_features = 1327 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 40 r2 =
0.94 medae = 2.80 mae = 4.04 mse = 35.10
n_features = 1327 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 166 r2 =
0.94 medae = 2.74 mae = 4.01 mse = 36.83
n_features = 1327 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 579 r2 =
0.93 medae = 2.89 mae = 4.13 mse = 38.86
n_features = 1327 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 1759 r2 =
0.93 medae = 2.89 mae = 4.13 mse = 38.86
n_features = 1327 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 4784 r2 =
0.93 medae = 2.89 mae = 4.13 mse = 38.86
n_features = 1327 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 11882 r2 =
0.93 medae = 2.89 mae = 4.13 mse = 38.86
n_features = 1327 learning_rate = 0.1 max_iter = 100 max_leaf_nodes = 27352 r2 =
0.93 medae = 2.89 mae = 4.13 mse = 38.86
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
```

```
Cell In[54], line 10
      8     for j in np.linspace(1, 3, 10):
      9         max_leaf_nodes = 1 + int(j ** 10)
---> 10         estimator = train_histogram_gradient_boosting(
     11             feature_selected_data, n_features, learning_rate=0.1,␣
 ↪max_iter=100, max_leaf_nodes=max_leaf_nodes
     12         )
     15 if hasattr(estimator, "coef_"):
     16     print(f"Sparsity %: {np.isclose(estimator.coef_, 0).mean():0.0%}␣
 ↪sparse")

Cell In[42], line 84, in␣
 ↪train_histogram_gradient_boosting(feature_selected_data, n_features,␣
 ↪learning_rate, max_iter, max_leaf_nodes)
     77 def train_histogram_gradient_boosting(feature_selected_data, n_features ␣
 ↪learning_rate, max_iter, max_leaf_nodes):
     78     regressor = sklearn.ensemble.HistGradientBoostingRegressor(
     79         learning_rate=learning_rate,
     80         max_iter=max_iter,
     81         max_leaf_nodes=max_leaf_nodes,
     82     )
---> 84     regressor.fit(
     85         feature_selected_data["train"]["features"],
     86         feature_selected_data["train"]["age"],
     87     )
     89     evaluate(
     90         regressor=regressor,
     91         feature_selected_data=feature_selected_data,
   (…)
     95         max_leaf_nodes=max_leaf_nodes,
     96     )
     98     return regressor

File ~/.main/lib/python3.9/site-packages/sklearn/ensemble/
 ↪_hist_gradient_boosting/gradient_boosting.py:670, in BaseHistGradientBoosting
 ↪fit(self, X, y, sample_weight)
    668 # Build `n_trees_per_iteration` trees.
    669 for k in range(self.n_trees_per_iteration_):
--> 670     grower = TreeGrower(
    671         X_binned=X_binned_train,
    672         gradients=g_view[:, k],
    673         hessians=h_view[:, k],
    674         n_bins=n_bins,
    675         n_bins_non_missing=self._bin_mapper.n_bins_non_missing_,
    676         has_missing_values=has_missing_values,
    677         is_categorical=self.is_categorical_,
    678         monotonic_cst=monotonic_cst,
    679         interaction_cst=interaction_cst,
    680         max_leaf_nodes=self.max_leaf_nodes,
    681         max_depth=self.max_depth,
    682         min_samples_leaf=self.min_samples_leaf,
    683         l2_regularization=self.l2_regularization,
    684         shrinkage=self.learning_rate,
    685         n_threads=n_threads,
    686     )
    687     grower.grow()
    689     acc_apply_split_time += grower.total_apply_split_time

File ~/.main/lib/python3.9/site-packages/sklearn/ensemble/
  ↪_hist_gradient_boosting/grower.py:332, in TreeGrower.__init__(self, X_binned, 
  ↪gradients, hessians, max_leaf_nodes, max_depth, min_samples_leaf, 
  ↪min_gain_to_split, n_bins, n_bins_non_missing, has_missing_values, 
  ↪is_categorical, monotonic_cst, interaction_cst, l2_regularization, 
  ↪min_hessian_to_split, shrinkage, n_threads)
    330 self.total_apply_split_time = 0.0  # time spent splitting nodes
    331 self.n_categorical_splits = 0
--> 332 self._intilialize_root(gradients, hessians, hessians_are_constant)
    333 self.n_nodes = 1

File ~/.main/lib/python3.9/site-packages/sklearn/ensemble/
  ↪_hist_gradient_boosting/grower.py:416, in TreeGrower._intilialize_root(self, 
  ↪gradients, hessians, hessians_are_constant)
    411     self.root.allowed_features = np.fromiter(
    412         allowed_features, dtype=np.uint32, count=len(allowed_features)
    413     )
    415 tic = time()
--> 416 self.root.histograms = self.histogram_builder.compute_histograms_brute(
    417     self.root.sample_indices, self.root.allowed_features
    418 )
    419 self.total_compute_hist_time += time() - tic
    421 tic = time()

KeyboardInterrupt:
```

[ ]:
```
print(f"Sparsity %: {np.isclose(estimator.coef_, 0).mean():0.0%} sparse")
print(f"Sparsity #: {len(estimator.coef_) - np.isclose(estimator.coef_, 0).
 ↪sum()} / {len(estimator.coef_)}")
```

```
Sparsity %: 67% sparse
Sparsity #: 659 / 2000
```

[ ]:
```
np.arange(1 - np.isclose(estimator.coef_, 0))
```

[ ]:
```
1378
```

[ ]:
```
len(estimator.coef_) * (1 - .71)
```

```
[ ]: 1401.2800000000002
```

```
[ ]:
```