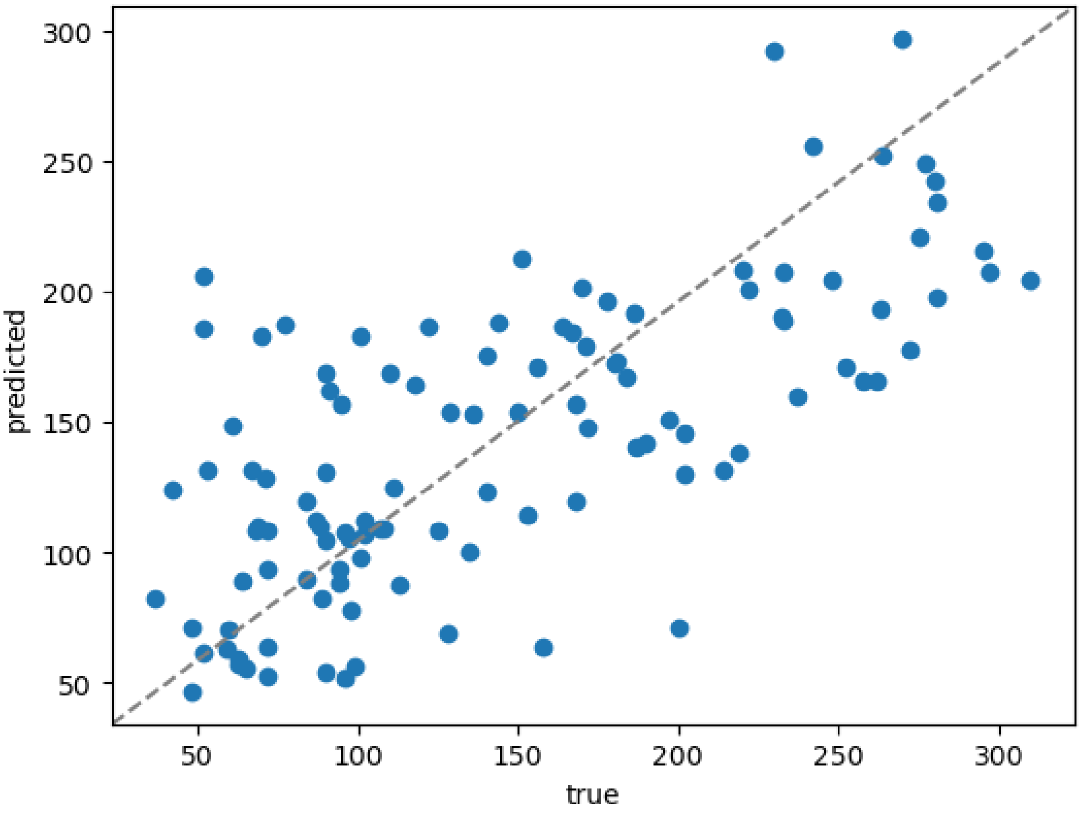
```
import numpy as np
import pandas as pd
from sklearn import *
import matplotlib.pyplot as plt

data = datasets.load_diabetes()

x = pd.DataFrame(data['data'], columns=data['feature_names'])
x['patient_id'] = np.arange(x.shape[0])
x['species'] = 1

y = data['target']
x['y'] = y   # Oops...

x
```

```
          age       sex       bmi        bp        s1        s2
s3  \
0    0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -
0.043401
1   -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163
0.074412
2    0.085299  0.050680  0.044451 -0.005670 -0.045599 -0.034194 -
0.032356
3   -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -
0.036038
4    0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596
0.008142
..        ...       ...       ...       ...       ...       ...
...
437  0.041708  0.050680  0.019662  0.059744 -0.005697 -0.002566 -
0.028674
438 -0.005515  0.050680 -0.015906 -0.067642  0.049341  0.079165 -
0.028674
439  0.041708  0.050680 -0.015906  0.017293 -0.037344 -0.013840 -
0.024993
440 -0.045472 -0.044642  0.039062  0.001215  0.016318  0.015283 -
0.028674
441 -0.045472 -0.044642 -0.073030 -0.081413  0.083740  0.027809
0.173816

           s4        s5        s6  patient_id  species      y
0   -0.002592  0.019907 -0.017646           0        1  151.0
1   -0.039493 -0.068332 -0.092204           1        1   75.0
2   -0.002592  0.002861 -0.025930           2        1  141.0
3    0.034309  0.022688 -0.009362           3        1  206.0
4   -0.002592 -0.031988 -0.046641           4        1  135.0
..        ...       ...       ...         ...      ...    ...
437 -0.002592  0.031193  0.007207         437        1  178.0
438  0.034309 -0.018114  0.044485         438        1  104.0
```

```
439 -0.011080 -0.046883  0.015491         439        1  132.0
440  0.026560  0.044529 -0.025930         440        1  220.0
441 -0.039493 -0.004222  0.003064         441        1   57.0

[442 rows x 13 columns]
```

```
x_train, x_test, y_train, y_test = model_selection.train_test_split(x,
y, random_state=42)

# columns = x.columns
columns = x.columns.difference({'y', 'patient_id', 'species'})
x_train = x_train[columns]
x_test = x_test[columns]

# regressor = svm.SVR(**{'C': 700, 'kernel': 'linear'})
regressor = linear_model.LinearRegression()
regressor.fit(x_train, y_train)
y_hat = regressor.predict(x_test)
print(regressor.score(x_test, y_test))

plt.scatter(y_test, y_hat)
plt.xlabel('true')
plt.ylabel('predicted')
xlim = plt.xlim()
ylim = plt.ylim()
plt.plot(xlim, ylim, '--', c='grey')
plt.xlim(xlim)
plt.ylim(ylim)
```

```
0.4849058889476757

(34.09984366041451, 309.2288595404606)
```



```
x_filtered =
feature_selection.VarianceThreshold(threshold=0.2).fit_transform(x_tra
in)
regressor.fit(x_filtered, y_train)
y_hat = regressor.predict(x_test)
print(regressor.score(x_test, y_test))

plt.scatter(y_test, y_hat)
plt.xlabel('true')
plt.ylabel('predicted')
xlim = plt.xlim()
ylim = plt.ylim()
plt.plot(xlim, ylim, '--', c='grey')
plt.xlim(xlim)
plt.ylim(ylim)

----------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In [28], line 1
----> 1 x_filtered =
feature_selection.VarianceThreshold(threshold=0.2).fit_transform(x_tra
in)
```

```
      2 regressor.fit(x_filtered, y_train)
      3 y_hat = regressor.predict(x_test)

File /usr/lib/python3/dist-packages/sklearn/utils/_set_output.py:142,
in _wrap_method_output.<locals>.wrapped(self, X, *args, **kwargs)
    140 @wraps(f)
    141 def wrapped(self, X, *args, **kwargs):
--> 142     data_to_wrap = f(self, X, *args, **kwargs)
    143     if isinstance(data_to_wrap, tuple):
    144         # only wrap the first output for cross decomposition
    145         return (
    146             _wrap_data_with_container(method, data_to_wrap[0],
X, self),
    147             *data_to_wrap[1:],
    148         )

File /usr/lib/python3/dist-packages/sklearn/base.py:859, in
TransformerMixin.fit_transform(self, X, y, **fit_params)
    855 # non-optimized default implementation; override when a better
    856 # method is possible for a given clustering algorithm
    857 if y is None:
    858     # fit method of arity 1 (unsupervised transformation)
--> 859     return self.fit(X, **fit_params).transform(X)
    860 else:
    861     # fit method of arity 2 (supervised transformation)
    862     return self.fit(X, y, **fit_params).transform(X)

File
/usr/lib/python3/dist-packages/sklearn/feature_selection/_variance_thr
eshold.py:125, in VarianceThreshold.fit(self, X, y)
    123     if X.shape[0] == 1:
    124         msg += " (X contains only one sample)"
--> 125     raise ValueError(msg.format(self.threshold))
    127 return self

ValueError: No feature in X meets the variance threshold 0.20000
```

```
gcv = model_selection.GridSearchCV(
    regressor, {
        # 'n_estimators': np.linspace(10, 100, 9, endpoint=True,
dtype=int),
        'criterion': ['squared_error', 'absolute_error',
'friedman_mse', 'poisson'],
        'max_depth': np.arange(1, 5),
        'max_features': ['sqrt', 'log2', None],
    },
cv=4)
gcv.fit(x_train, y_train)
print(gcv.best_params_)
print(gcv.best_score_)
```

```
{'criterion': 'poisson', 'max_depth': 4, 'max_features': 'sqrt'}
0.40225974404591913
```

```
regressor = svm.SVR()
gcv = model_selection.GridSearchCV(
    regressor, [
        {
            'kernel': ['linear'],
            # 'C': [ 1e2, 1e3, 1e5, 1e7],
            'C': np.arange(100, 2100, 100),
        },
        # {
        #     'kernel': ['poly'],
        #     'degree': np.arange(1, 5),
        #     'gamma': ['auto', 'scale'],
        #     'coef0': [0, 1e-2, 1e-1, 1],
        #     'C': [1e-2, 1e-1, 1, 1e1, 1e2, 1e3],
        # },
        # {
        #     'kernel': ['sigmoid'],
        #     'gamma': ['auto', 'scale'],
        #     'coef0': [0, 1e-2, 1e-1, 1],
        #     'C': [1e-2, 1e-1, 1, 1e1, 1e2, 1e3],
        # },
        # {
        #     'kernel': ['rbf'],
        #     'gamma': ['auto', 'scale'],
        #     'C': [1e-2, 1e-1, 1, 1e1, 1e2, 1e3],
        # },
    ],
cv=4, n_jobs=8)
gcv.fit(x_train, y_train)
print(gcv.best_params_)
print(gcv.best_score_)
```

```
{'C': 700, 'kernel': 'linear'}
0.4419789143972433
```