Selecting features for model performance

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Machine Learning Engineer, Faktion

Ansur dataset sample

Gender	chestdepth	handlength	neckcircumference	shoulderlength	earlength
Female	243	176	326	136	62
Female	219	177	325	135	58
Male	259	193	400	145	71
Male	253	195	380	141	62

Pre-processing the data

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train_std = scaler.fit_transform(X_train)
```

Creating a logistic regression model

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

lr = LogisticRegression()
lr.fit(X_train_std, y_train)

X_test_std = scaler.transform(X_test)

y_pred = lr.predict(X_test_std)
print(accuracy_score(y_test, y_pred))
```

```
0.99
```

Inspecting the feature coefficients

```
print(lr.coef_)
```

```
array([[-3.  ,  0.14,  7.46,  1.22,  0.87]])
```

```
print(dict(zip(X.columns, abs(lr.coef_[0]))))
```

```
{'chestdepth': 3.0,
 'handlength': 0.14,
 'neckcircumference': 7.46,
 'shoulderlength': 1.22,
 'earlength': 0.87}
```

Features that contribute little to a model

```
X.drop('handlength', axis=1, inplace=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

lr.fit(scaler.fit_transform(X_train), y_train)

print(accuracy_score(y_test, lr.predict(scaler.transform(X_test))))
```

```
0.99
```

Recursive Feature Elimination

```python
from sklearn.feature_selection import RFE

rfe = RFE(estimator=LogisticRegression(), n_features_to_select=2, verbose=1)
rfe.fit(X_train_std, y_train)
```

```
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
```

Dropping a feature will affect other feature's coefficients

Inspecting the RFE results

```
X.columns[rfe.support_]
```

```
Index(['chestdepth', 'neckcircumference'], dtype='object')
```

```
print(dict(zip(X.columns, rfe.ranking_)))
```

```
{'chestdepth': 1,
 'handlength': 4,
 'neckcircumference': 1,
 'shoulderlength': 2,
 'earlength': 3}
```

```
print(accuracy_score(y_test, rfe.predict(X_test_std)))
```

```
0.99
```

Let's practice!

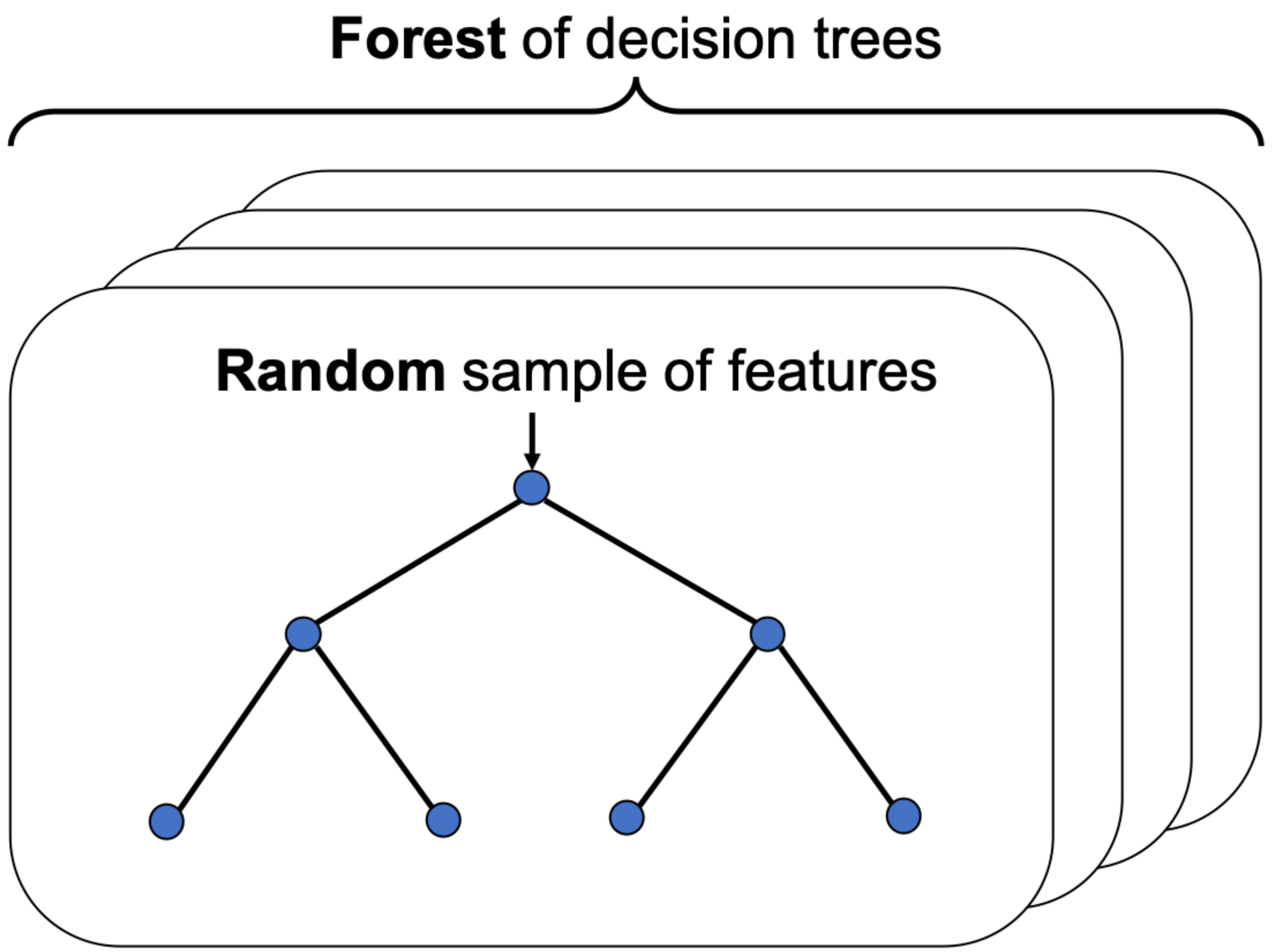
DIMENSIONALITY REDUCTION IN PYTHON

Tree-based feature selection

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Machine Learning Engineer, Faktion

Random forest classifier

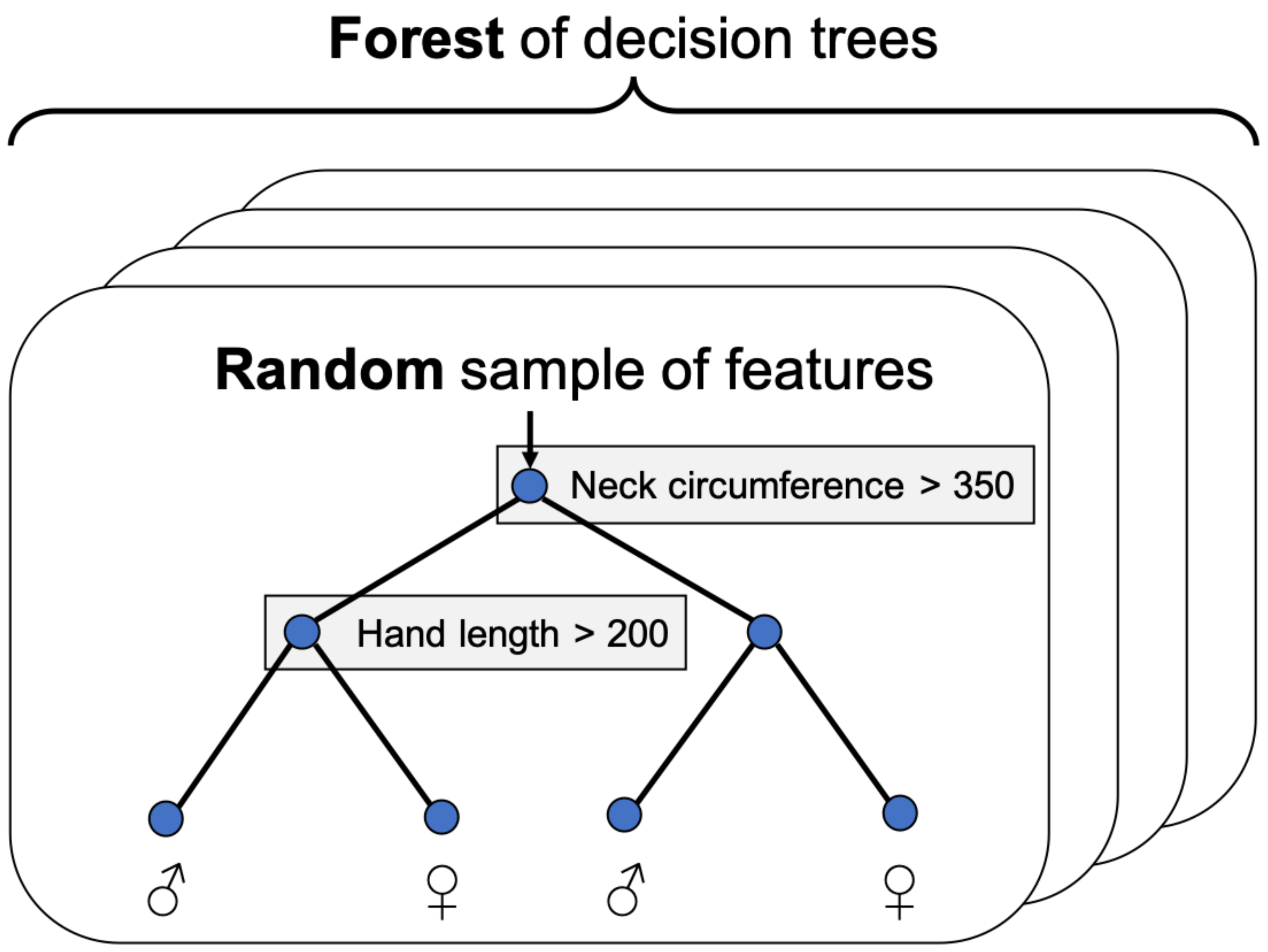


Random forest classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

rf = RandomForestClassifier()

rf.fit(X_train, y_train)

print(accuracy_score(y_test, rf.predict(X_test)))
```

```
0.99
```

Random forest classifier



Feature importance values

```
rf = RandomForestClassifier()

rf.fit(X_train, y_train)

print(rf.feature_importances_)
```

```
array([0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.04, 0.  , 0.01, 0.01,
       0.  , 0.  , 0.  , 0.  , 0.01, 0.01, 0.  , 0.  , 0.  , 0.  , 0.05,
       ...
       0.  , 0.14, 0.  , 0.  , 0.  , 0.06, 0.  , 0.  , 0.  , 0.  , 0.  ,
       0.  , 0.07, 0.  , 0.  , 0.01, 0.  ])
```

```
print(sum(rf.feature_importances_))
```

```
1.0
```

Feature importance as a feature selector

```
mask = rf.feature_importances_ > 0.1

print(mask)
```

```
array([False, False, ..., True, False])
```

```
X_reduced = X.loc[:, mask]

print(X_reduced.columns)
```

```
Index(['chestheight', 'neckcircumference', 'neckcircumferencebase',
       'shouldercircumference'], dtype='object')
```

RFE with random forests

```python
from sklearn.feature_selection import RFE

rfe = RFE(estimator=RandomForestClassifier(),
          n_features_to_select=6, verbose=1)

rfe.fit(X_train,y_train)
```

```
Fitting estimator with 94 features.
Fitting estimator with 93 features
...
Fitting estimator with 8 features.
Fitting estimator with 7 features.
```

```python
print(accuracy_score(y_test, rfe.predict(X_test))
```

```
0.99
```

RFE with random forests

```
from sklearn.feature_selection import RFE

rfe = RFE(estimator=RandomForestClassifier(),
          n_features_to_select=6, step=10, verbose=1)

rfe.fit(X_train,y_train)
```

```
Fitting estimator with 94 features.
Fitting estimator with 84 features.
...
Fitting estimator with 24 features.
Fitting estimator with 14 features.
```

```
print(X.columns[rfe.support_])
```

```
Index(['biacromialbreadth', 'handbreadth', 'handcircumference',
       'neckcircumference', 'neckcircumferencebase', 'shouldercircumference'], dtype='object')
```

Let's practice!

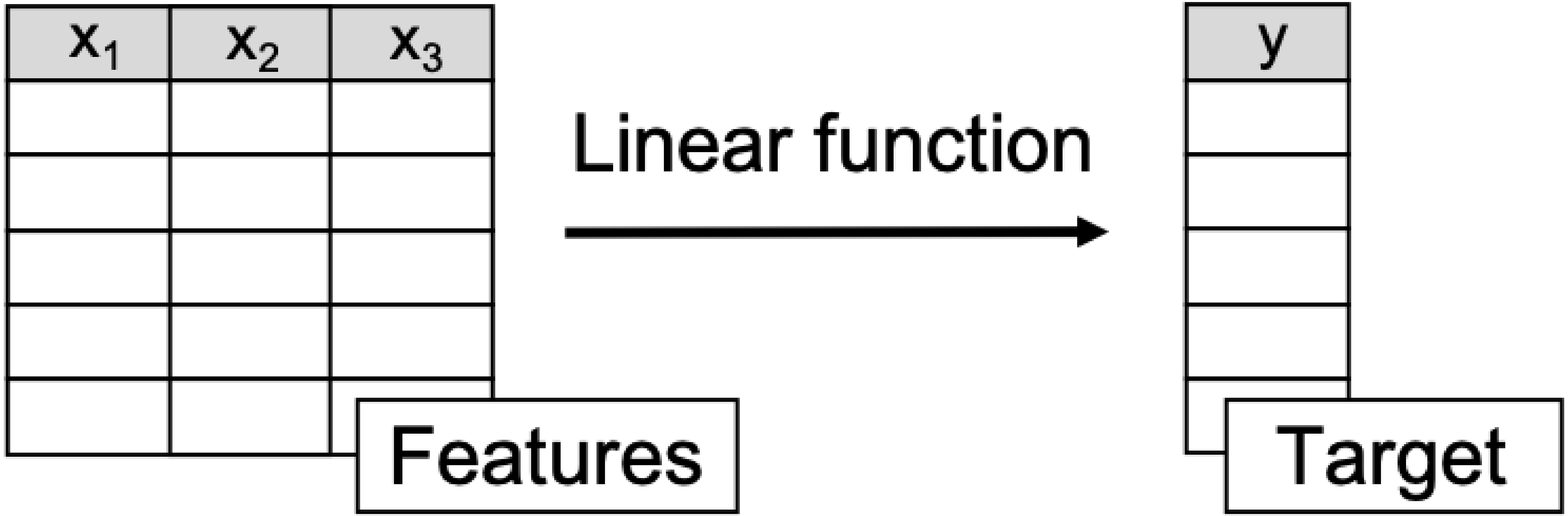
DIMENSIONALITY REDUCTION IN PYTHON

Regularized linear regression

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Machine Learning Engineer, Faktion

Linear model concept

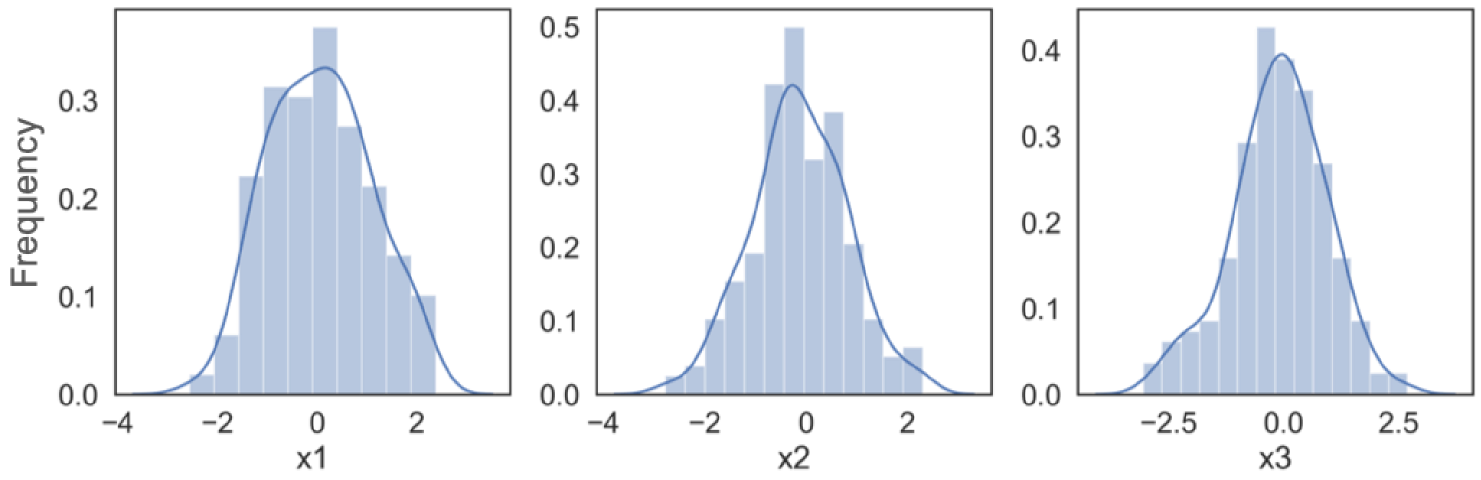


Creating our own dataset

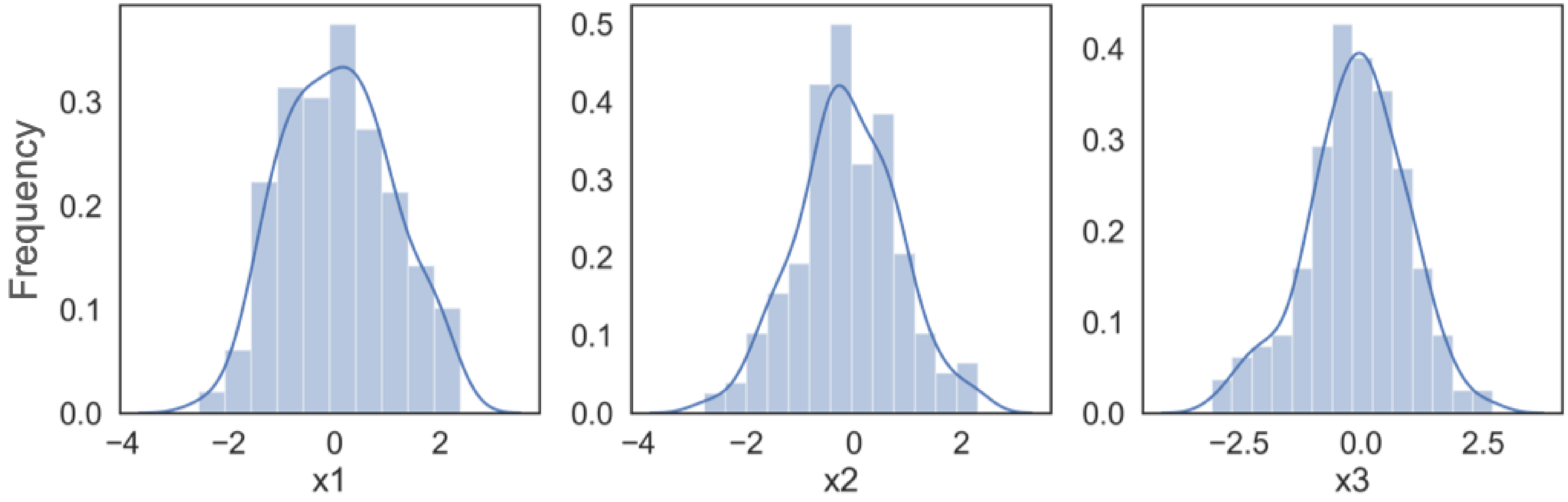
x1	x2	x3
1.76	-0.37	-0.60
0.40	-0.24	-1.12
0.98	1.10	0.77
...	...	...

Creating our own dataset

x1	x2	x3
1.76	-0.37	-0.60
0.40	-0.24	-1.12
0.98	1.10	0.77
...	...	...



Creating our own dataset



Creating our own target feature:

> $$y = 20 + 5x_1 + 2x_2 + 0x_3 + error$$

Linear regression in Python

```python
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_train, y_train)

# Actual coefficients = [5 2 0]
print(lr.coef_)
```

```
[ 4.95  1.83 -0.05]
```

```python
# Actual intercept = 20
print(lr.intercept_)
```

```
19.8
```

Linear regression in Python

```python
# Calculates R-squared
print(lr.score(X_test, y_test))
```

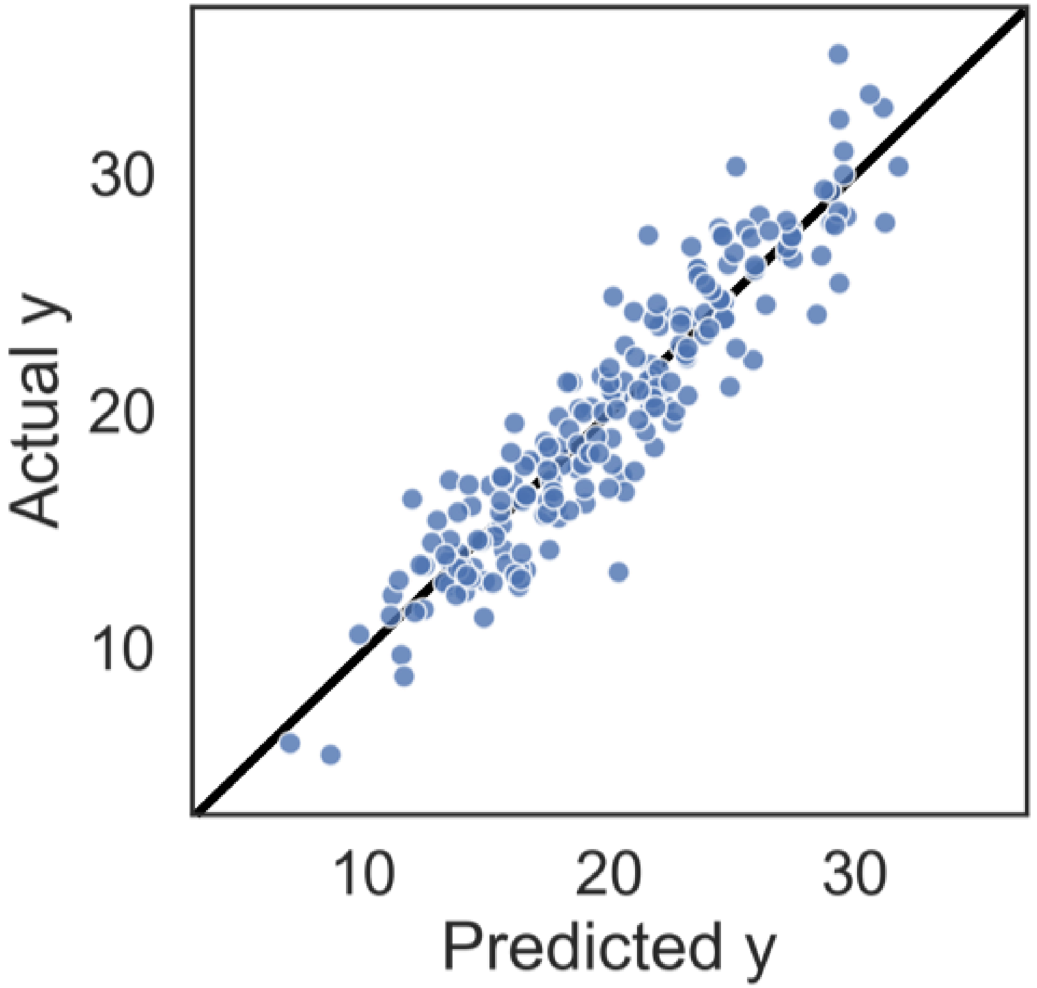
```
0.976
```

Linear regression in Python

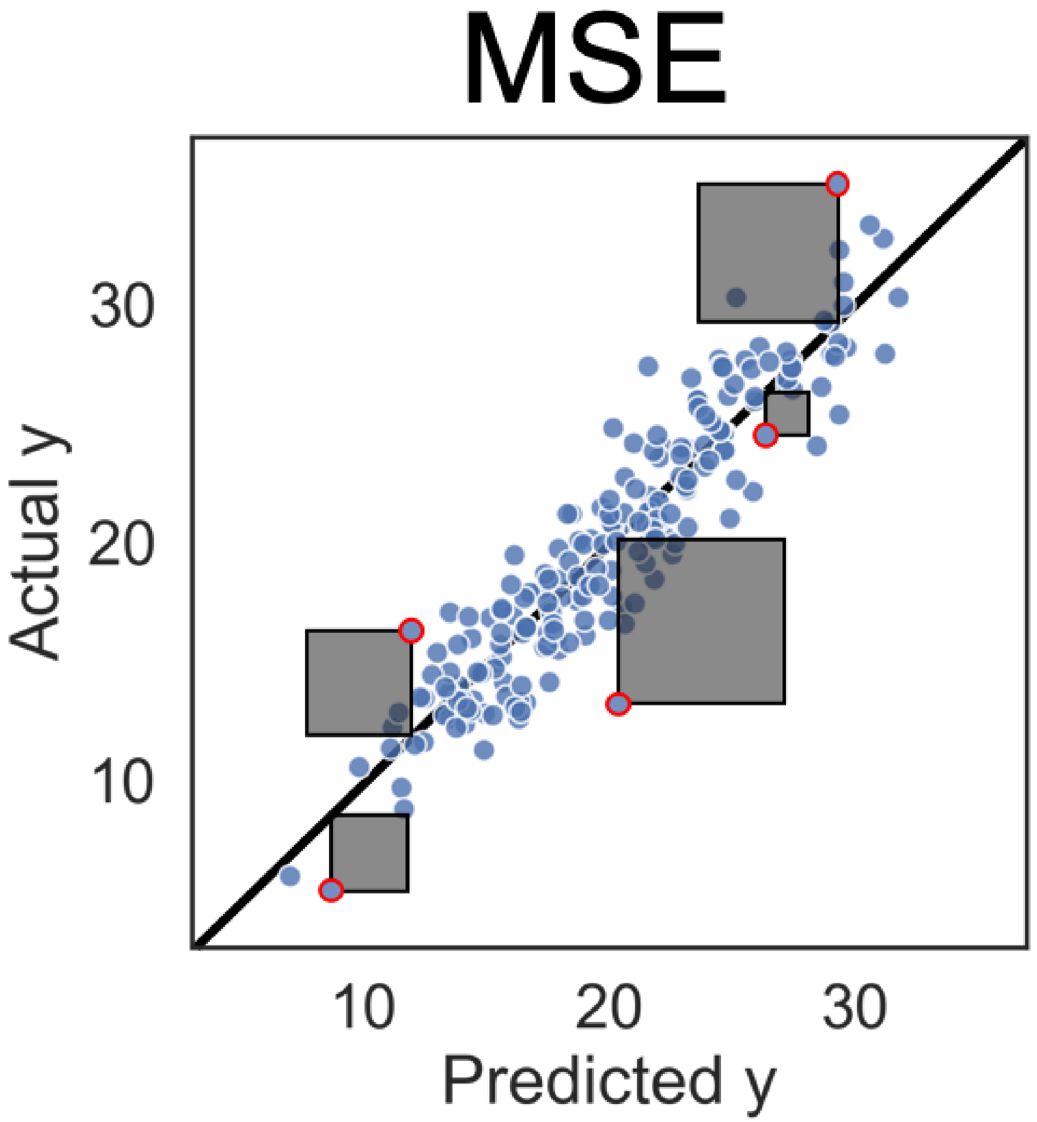
```python
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_train, y_train)

# Actual coefficients = [5 2 0]
print(lr.coef_)
```

```
[ 4.95  1.83 -0.05]
```

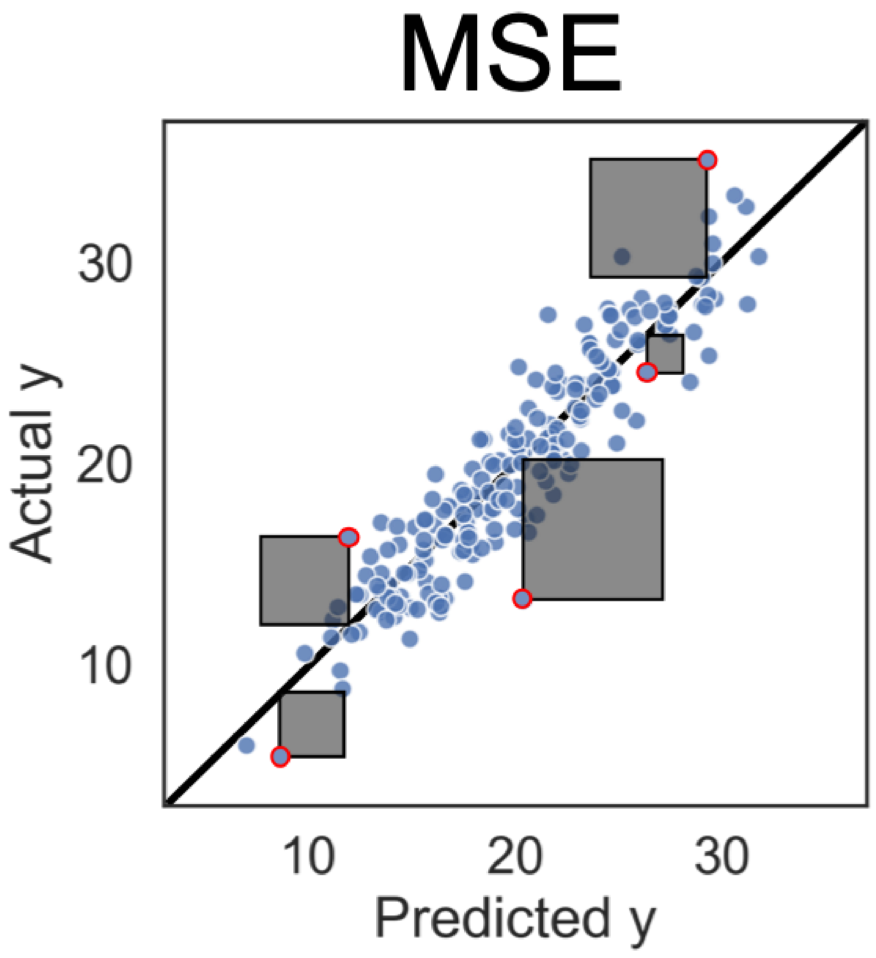
Loss function: Mean Squared Error



Loss function: Mean Squared Error

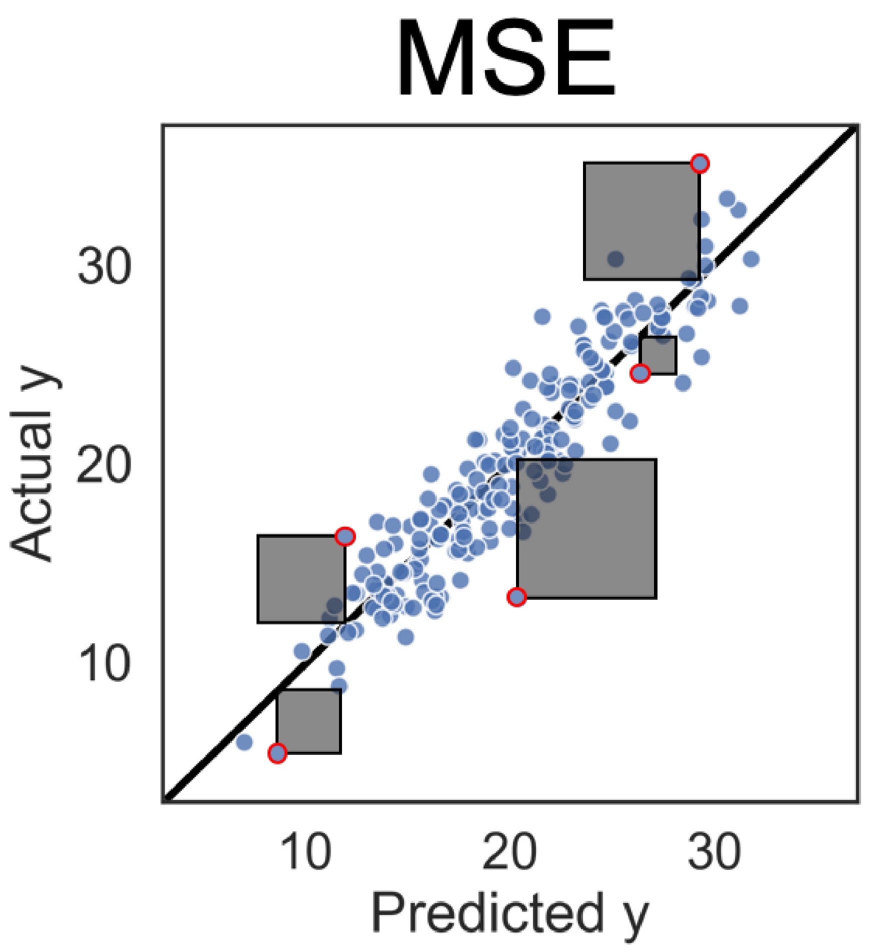


Adding regularization



$$+ \overbrace{\alpha(|\beta_1|+|\beta_2|+|\beta_3|)}^{\text{Regularization term}}$$

Adding regularization



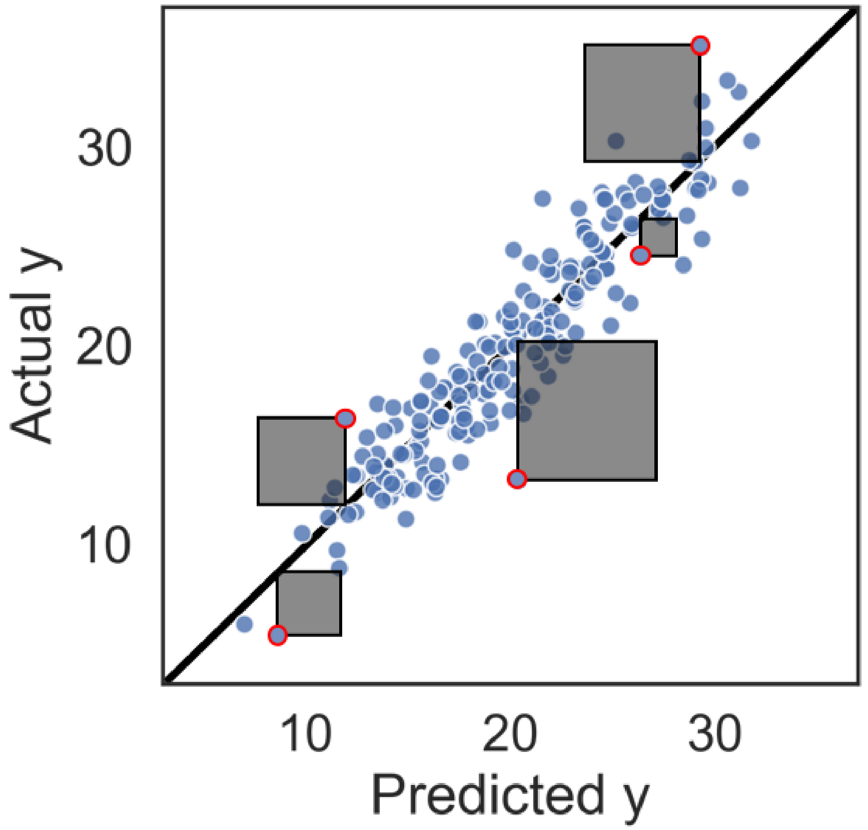
Tries to make model accurate

$$+ \underbrace{\alpha(|\beta_1|+|\beta_2|+|\beta_3|)}_{\text{Regularization term}}$$

Tries to make model simple

Adding regularization

MSE



Tries to make model accurate

$$+ \overbrace{\alpha(|\beta_1|+|\beta_2|+|\beta_3|)}^{\text{Regularization term}}$$

Tries to make model simple

[1] alpha, when it's too low the model might overfit, when it's too high the model might become too simple and inaccurate. One linear model that includes this type of regularization is called Lasso, for least absolute shrinkage

Lasso regressor

```python
from sklearn.linear_model import Lasso

la = Lasso()
la.fit(X_train, y_train)

# Actual coefficients = [5 2 0]
print(la.coef_)
```

```
[4.07 0.59 0.  ]
```

```python
print(la.score(X_test, y_test))
```

```
0.861
```

Lasso regressor

```
from sklearn.linear_model import Lasso

la = Lasso(alpha=0.05)
la.fit(X_train, y_train)

# Actual coefficients = [5 2 0]
print(la.coef_)
```

```
[ 4.91  1.76 0.   ]
```

```
print(la.score(X_test, y_test))
```

```
0.974
```

Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

Combining feature selectors

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Machine Learning Engineer, Faktion

Lasso regressor

```
from sklearn.linear_model import Lasso

la = Lasso(alpha=0.05)
la.fit(X_train, y_train)

# Actual coefficients = [5 2 0]
print(la.coef_)
```

```
[ 4.91  1.76 0. ]
```

```
print(la.score(X_test, y_test))
```

```
0.974
```

LassoCV regressor

```
from sklearn.linear_model import LassoCV

lcv = LassoCV()

lcv.fit(X_train, y_train)

print(lcv.alpha_)
```

```
0.09
```

LassoCV regressor

```
mask = lcv.coef_ != 0

print(mask)
```

```
[ True True False ]
```

```
reduced_X = X.loc[:, mask]
```

Taking a step back

- Random forest is combination of decision trees.
- We can use combination of models for feature selection too.

Feature selection with LassoCV

```
from sklearn.linear_model import LassoCV

lcv = LassoCV()
lcv.fit(X_train, y_train)

lcv.score(X_test, y_test)
```

```
0.99
```

```
lcv_mask = lcv.coef_ != 0
sum(lcv_mask)
```

```
66
```

Feature selection with random forest

```
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestRegressor

rfe_rf = RFE(estimator=RandomForestRegressor(),
             n_features_to_select=66, step=5, verbose=1)

rfe_rf.fit(X_train, y_train)
rf_mask = rfe_rf.support_
```

Feature selection with gradient boosting

```
from sklearn.feature_selection import RFE
from sklearn.ensemble import GradientBoostingRegressor

rfe_gb = RFE(estimator=GradientBoostingRegressor(),
             n_features_to_select=66, step=5, verbose=1)

rfe_gb.fit(X_train, y_train)
gb_mask = rfe_gb.support_
```

Combining the feature selectors

```
import numpy as np

votes = np.sum([lcv_mask, rf_mask, gb_mask], axis=0)

print(votes)
```

```
array([3, 2, 2, ..., 3, 0, 1])
```

```
mask = votes >= 2
reduced_X = X.loc[:, mask]
```

Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON