Introduction

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye

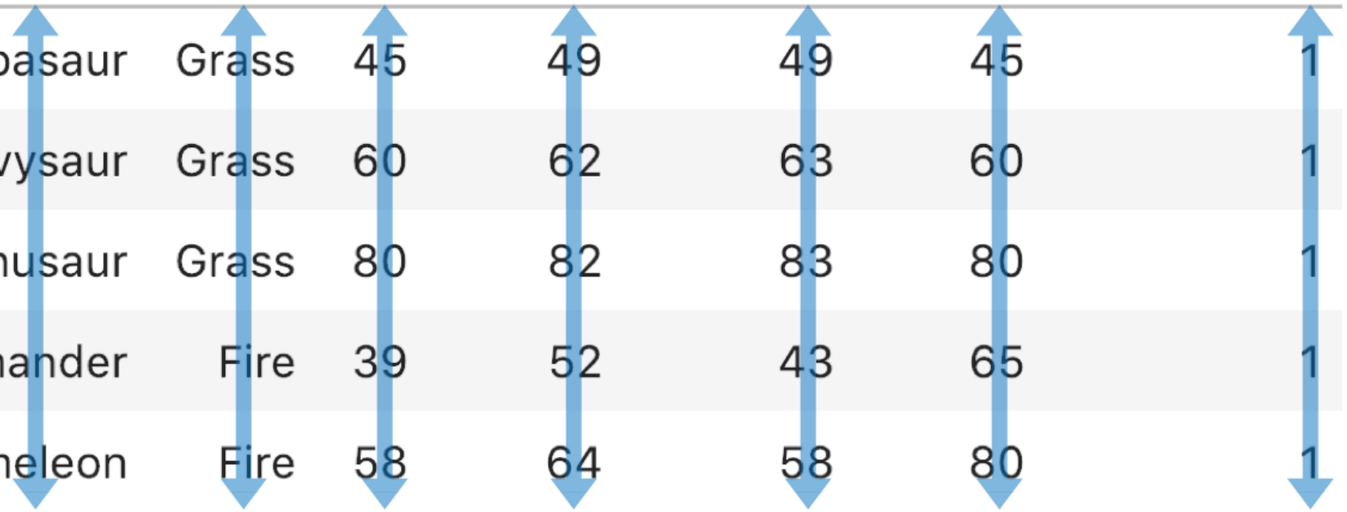
Machine Learning Engineer, Faktion

Tidy data

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1

Tidy data

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1



Tidy data

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1

Tidy data

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1

The shape attribute

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1

```
pokemon_df.shape
```

```
(5, 7)
```

When to use dimensionality reduction?

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1

When to use dimensionality reduction?

Name	Type	HP	Attack	Defense	Speed	Generation
Bulbasaur	Grass	45	49	49	45	1
Ivysaur	Grass	60	62	63	60	1
Venusaur	Grass	80	82	83	80	1
Charmander	Fire	39	52	43	65	1
Charmeleon	Fire	58	64	58	80	1

The describe method

```
pokemon_df.describe()
```

	HP	Attack	Defense	Speed	Generation
count	5.0	5.0	5.0	5.0	5.0
mean	56.4	61.8	59.2	66.0	1.0
std	15.9	13.0	15.4	14.7	0.0
min	39.0	49.0	43.0	45.0	1.0
25%	45.0	52.0	49.0	60.0	1.0
50%	58.0	62.0	58.0	65.0	1.0
75%	60.0	64.0	63.0	80.0	1.0
max	80.0	82.0	83.0	80.0	1.0

The describe method

```
pokemon_df.describe()
```

	HP	Attack	Defense	Speed	Generation
count	5.0	5.0	5.0	5.0	5.0
mean	56.4	61.8	59.2	66.0	1.0
std	15.9	13.0	15.4	14.7	0.0
min	39.0	49.0	43.0	45.0	1.0
25%	45.0	52.0	49.0	60.0	1.0
50%	58.0	62.0	58.0	65.0	1.0
75%	60.0	64.0	63.0	80.0	1.0
max	80.0	82.0	83.0	80.0	1.0

The describe method

```
pokemon_df.describe(exclude='number')
```

	Name	Type
count	5	5
unique	5	2
top	Charmander	Grass
freq	1	3

The describe method

```
pokemon_df.describe(exclude='number')
```

	Name	Type
count	5	5
unique	5	2
top	Charmander	Grass
freq	1	3

Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

Feature selection vs feature extraction

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Machine Learning Engineer, Faktion

Why reduce dimensionality?

Your dataset will:

- be less complex
- require less disk space
- require less computation time
- have lower chance of model overfitting

Feature selection

income	age	favorite color
10000	18	Black
50000	47	Blue
20000	40	Blue
30000	29	Green
20000	22	Purple

Feature selection

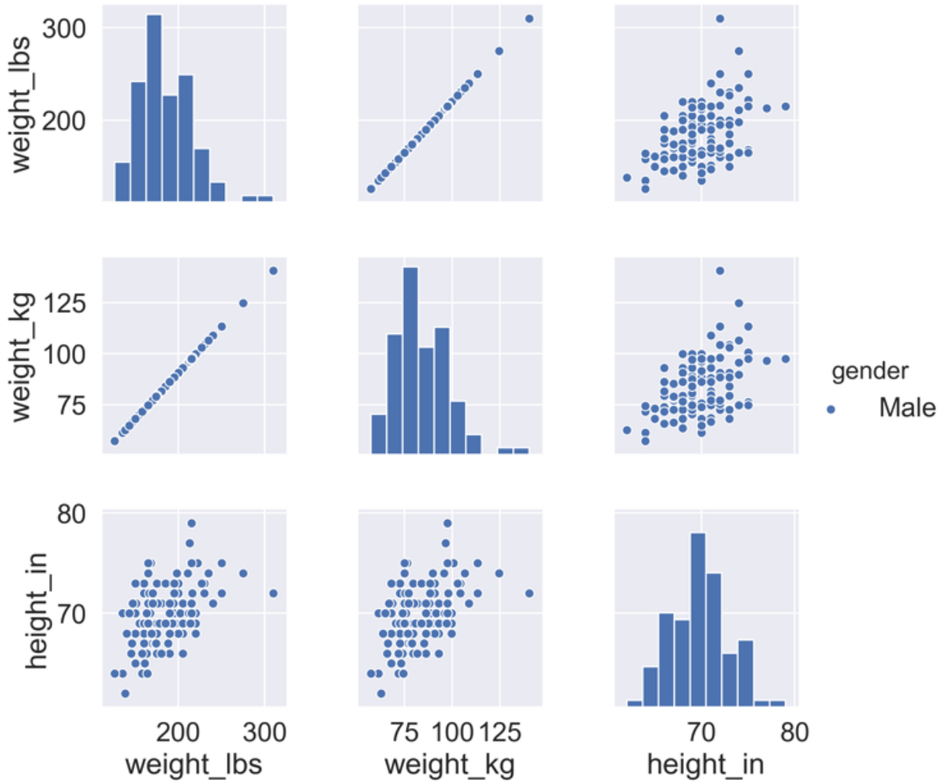
income	age	favorite color
10000	18	Black
50000	47	Blue
20000	40	Blue
30000	29	Green
20000	22	Purple

income	age
10000	18
50000	47
20000	40
30000	29
20000	22

```
insurance_df.drop('favorite color', axis=1)
```

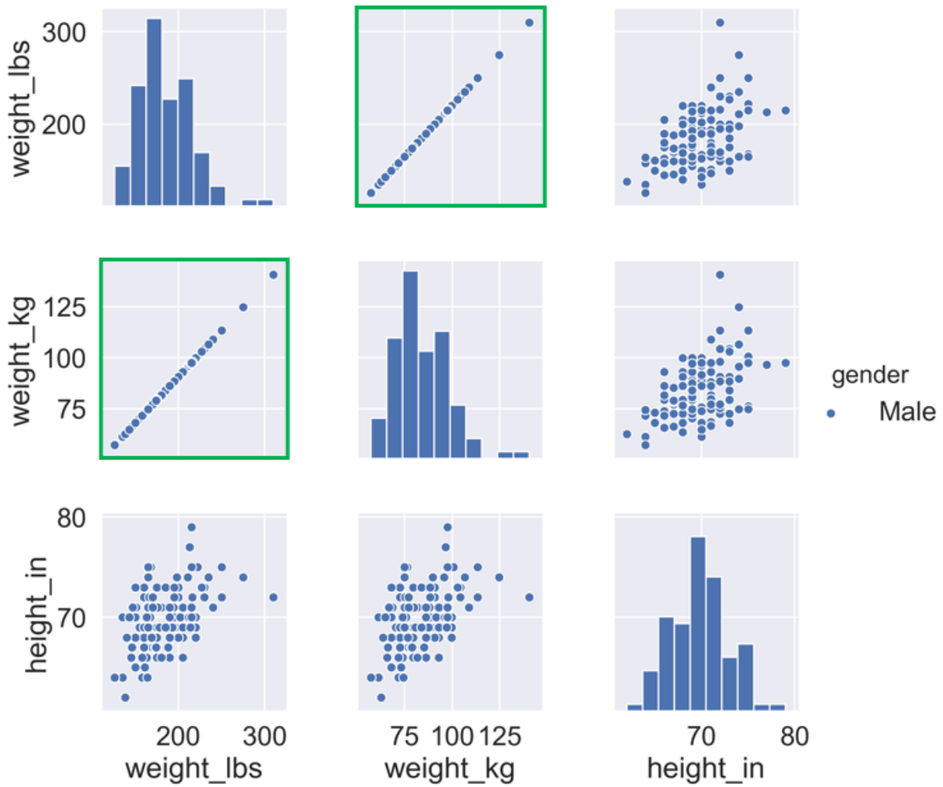
Building a pairplot on ANSUR data

```
sns.pairplot(ansur_df, hue="gender", diag_kind='hist')
```



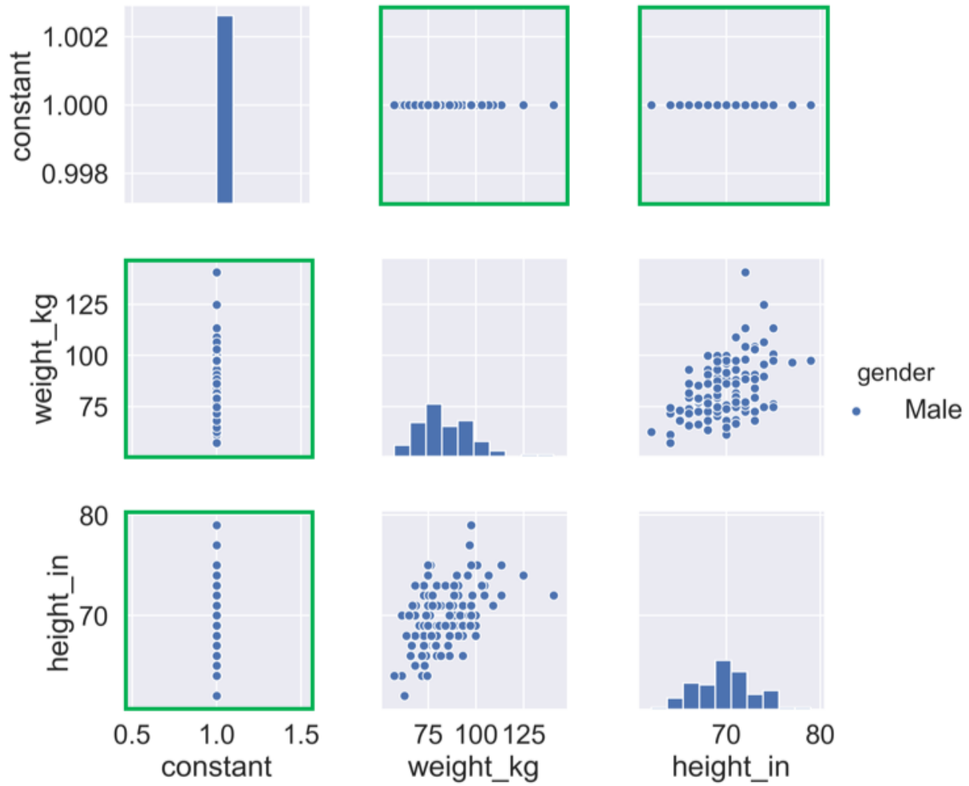
Building a pairplot on ANSUR data

```
sns.pairplot(ansur_df, hue="gender", diag_kind='hist')
```

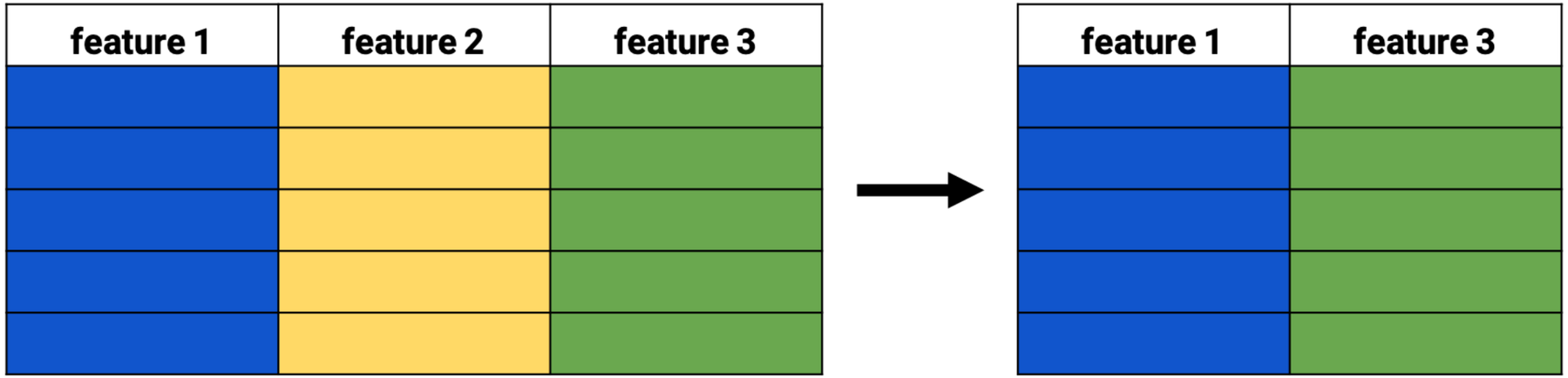


Building a pairplot on ANSUR data

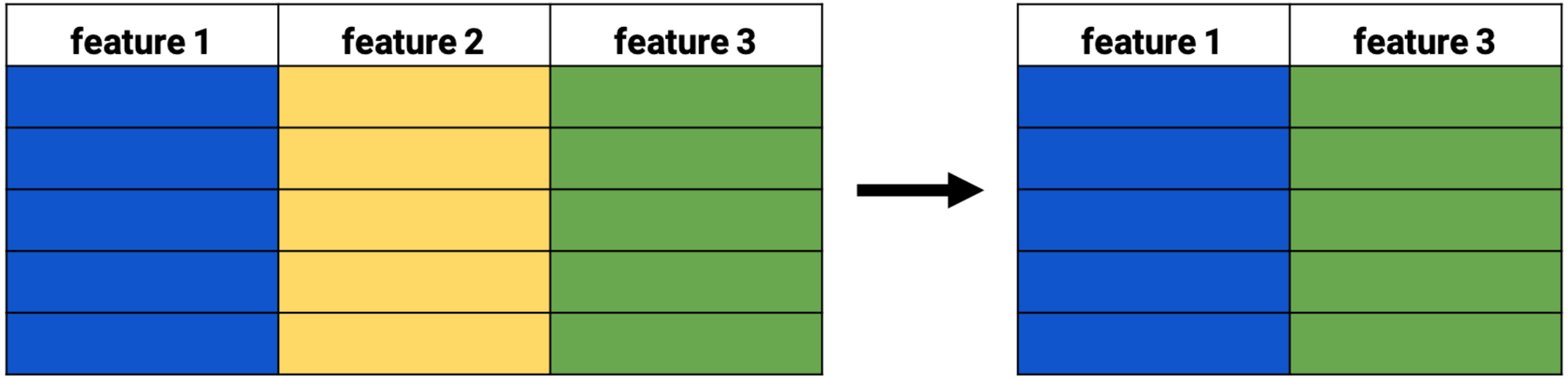
```
sns.pairplot(ansur_df, hue="gender", diag_kind='hist')
```



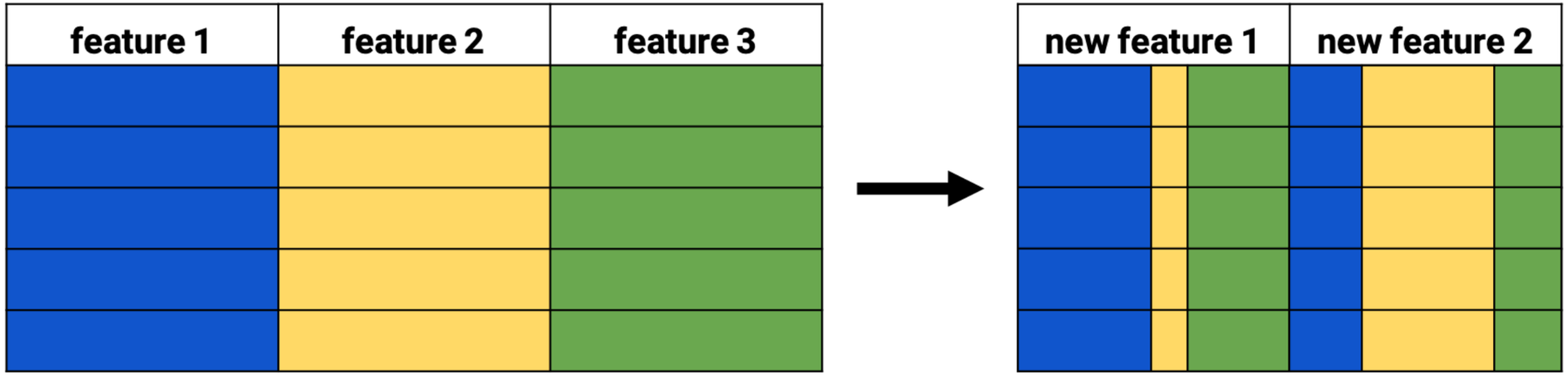
Feature selection



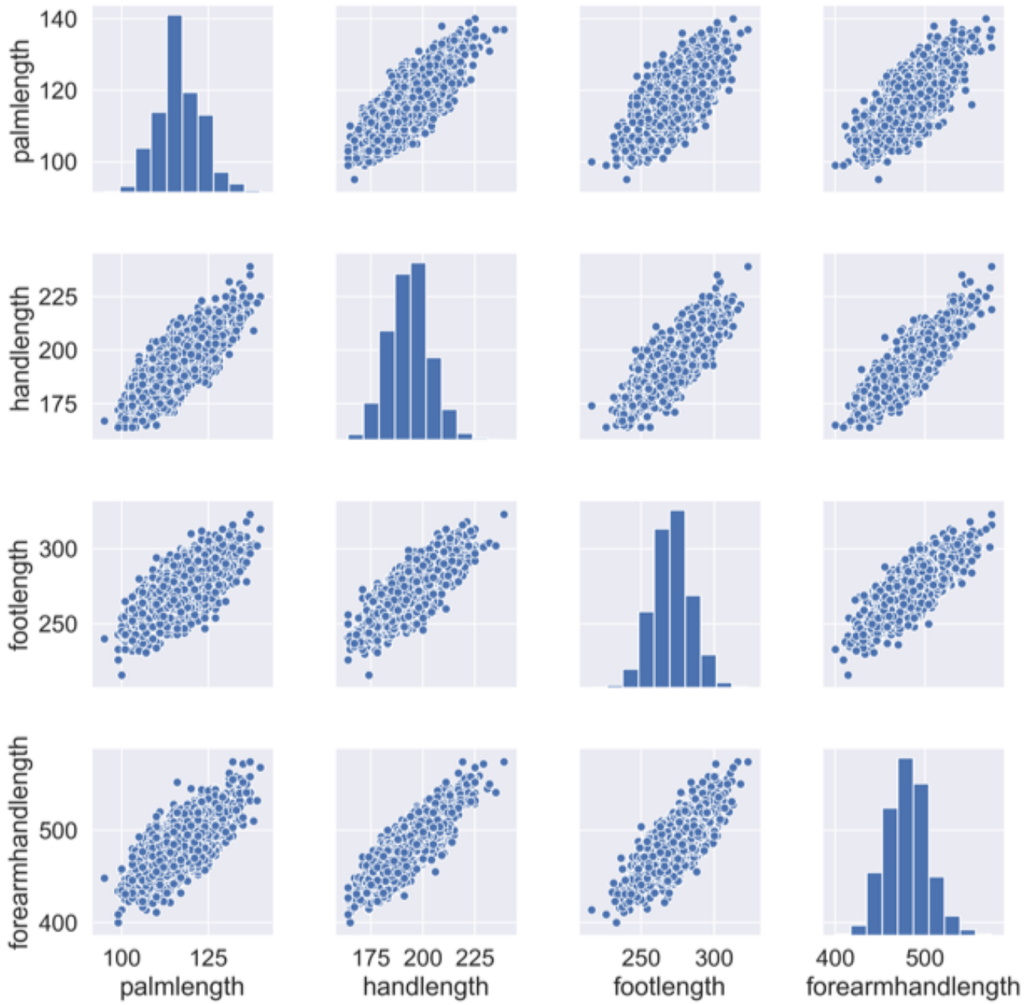
Feature selection



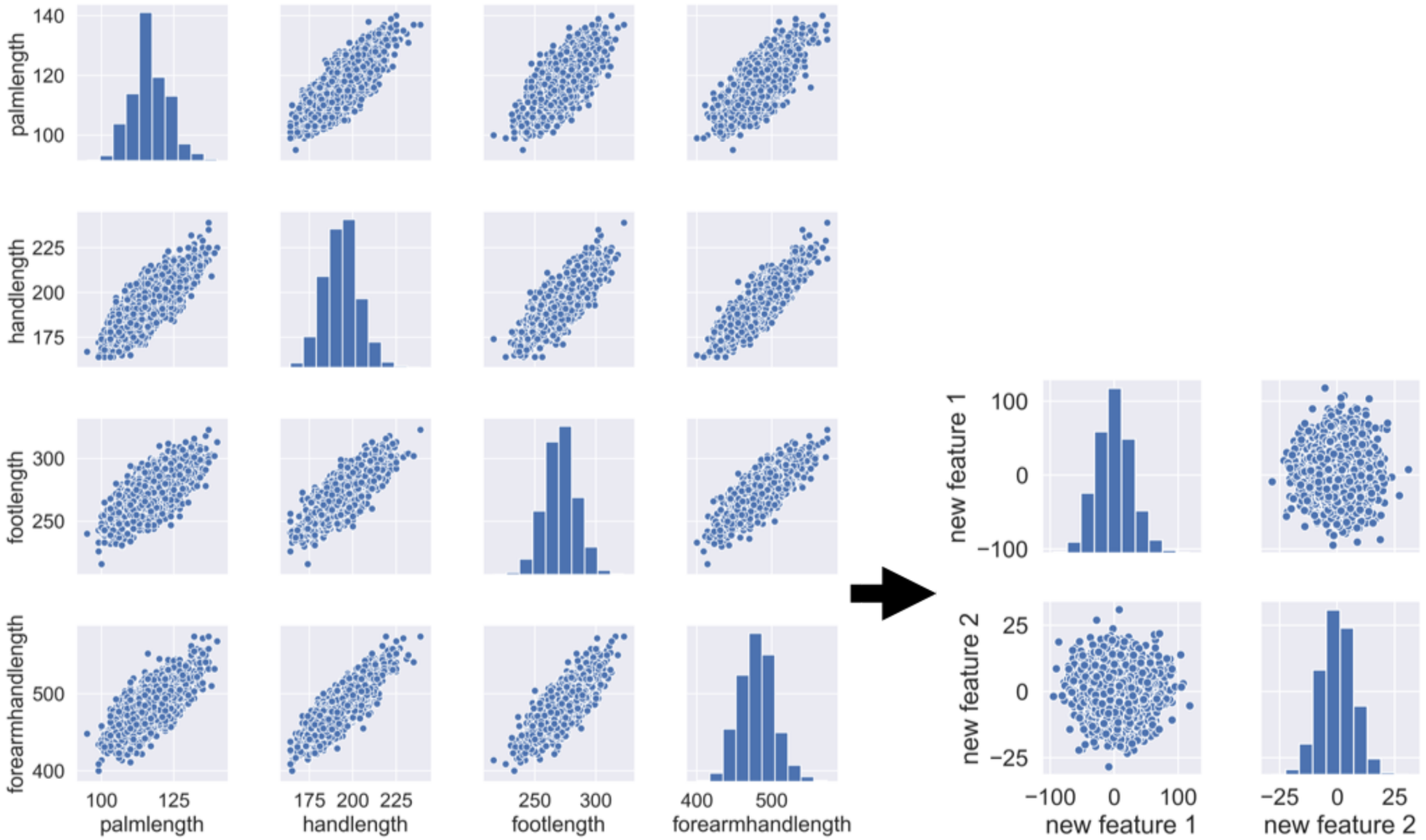
Feature extraction



Feature extraction - Example



Feature extraction - Example



Let's practice!

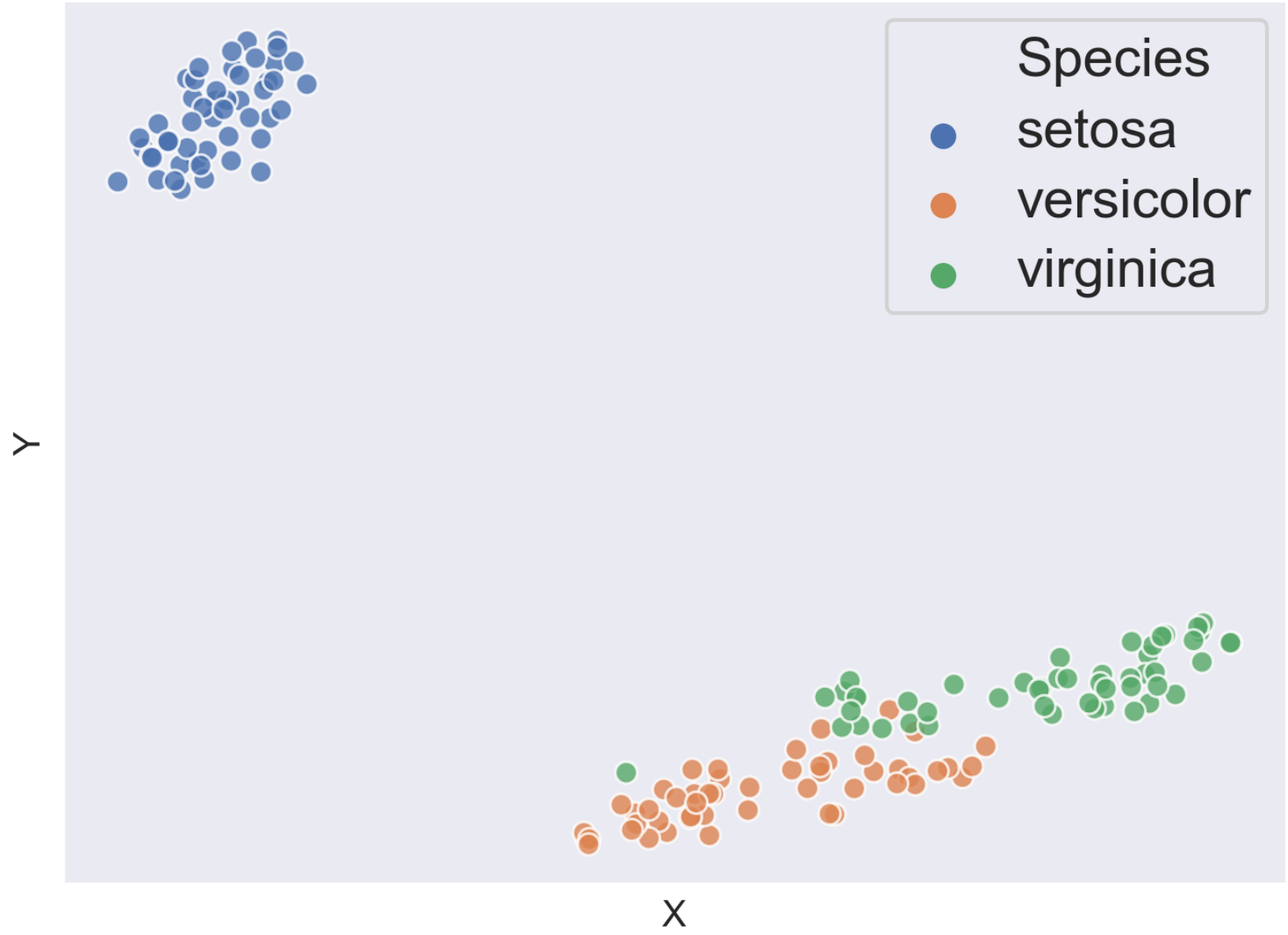
DIMENSIONALITY REDUCTION IN PYTHON

t-SNE visualization of high-dimensional data

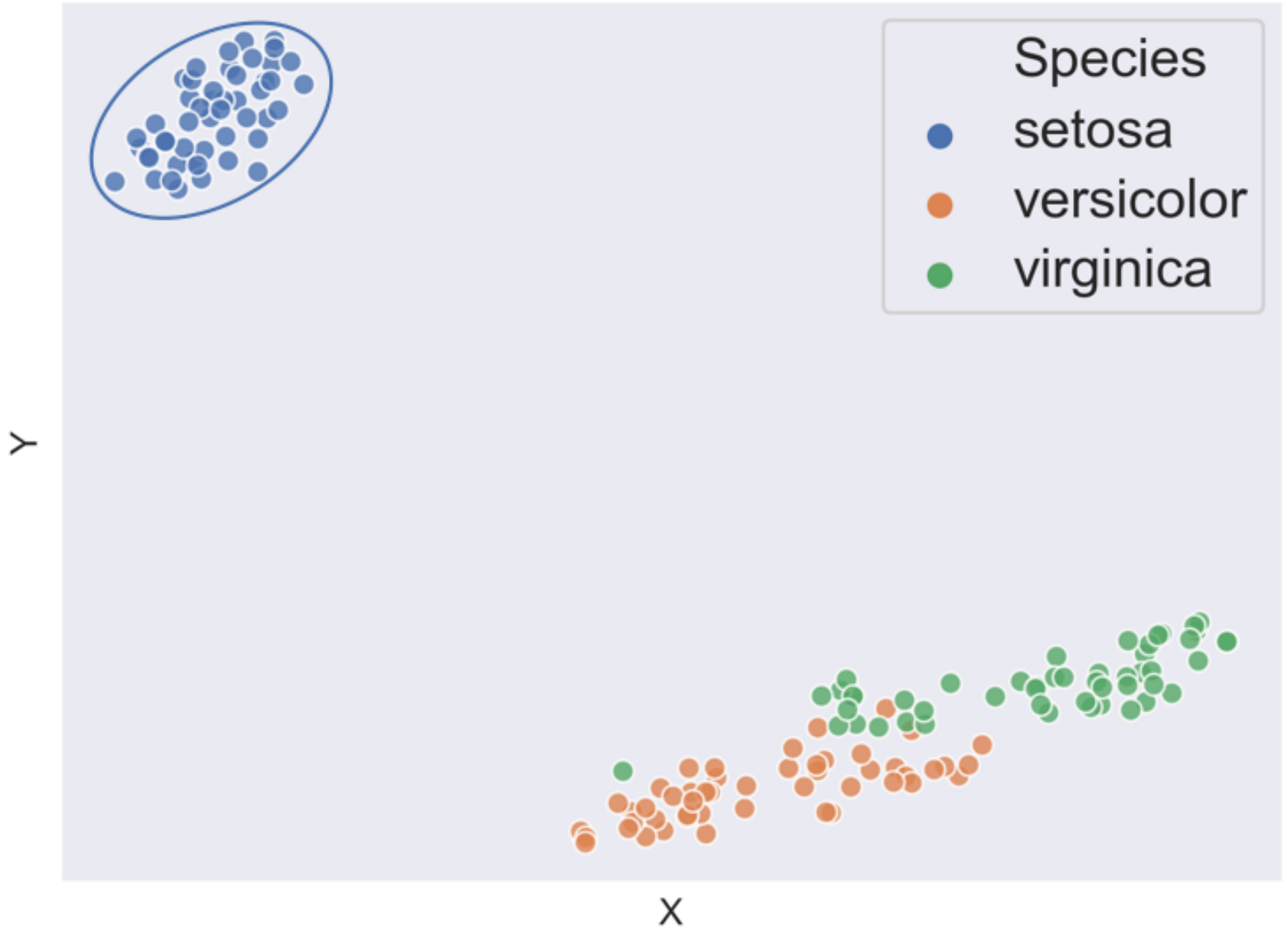
DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Machine Learning Engineer, Faktion

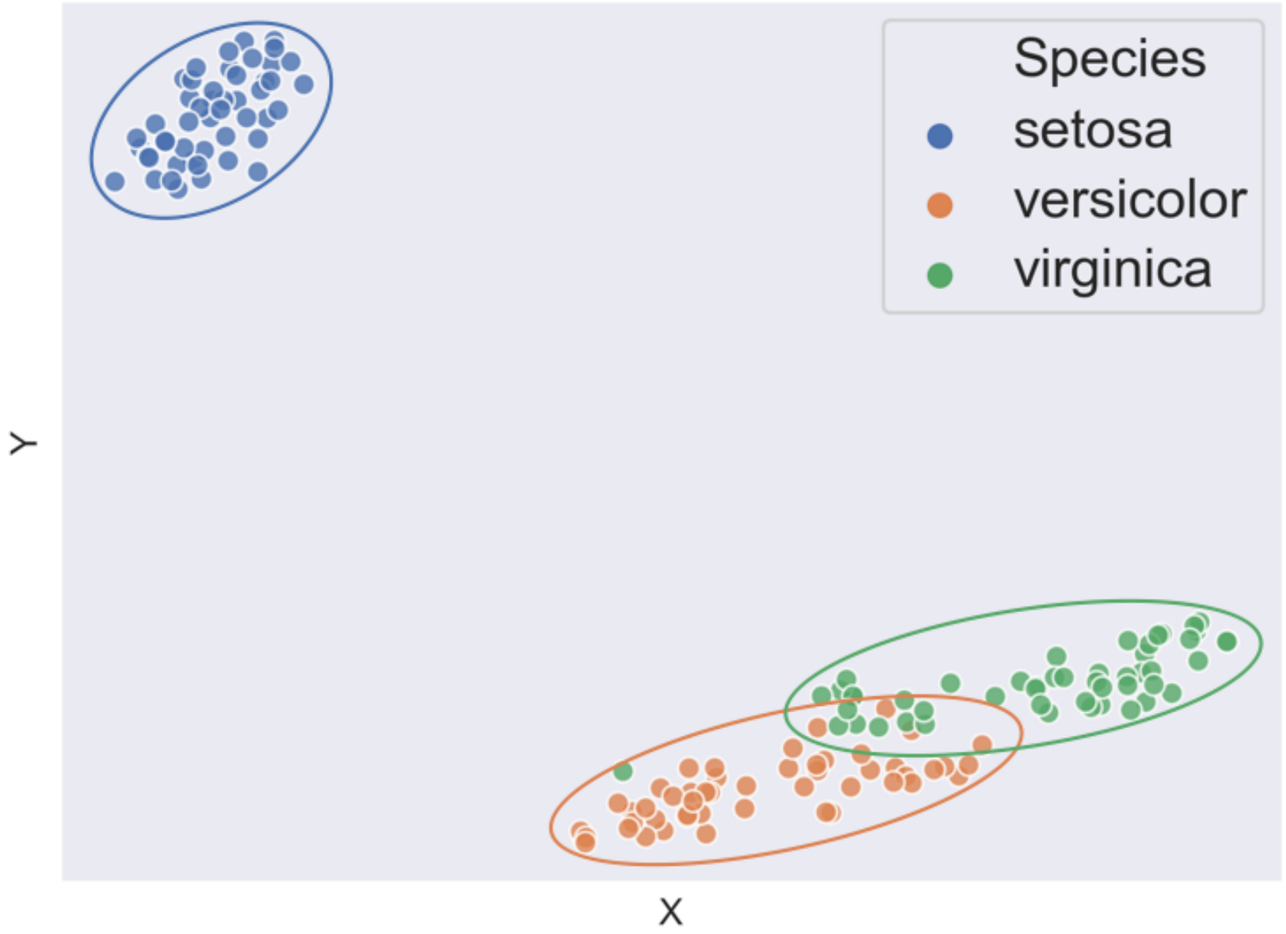
t-SNE on IRIS dataset



t-SNE on IRIS dataset



t-SNE on IRIS dataset



t-SNE on female ANSUR dataset

```
df.shape
```

```
(1986, 99)
```

```
non_numeric = ['BMI_class', 'Height_class',
               'Gender',  'Component', 'Branch']

df_numeric = df.drop(non_numeric, axis=1)

df_numeric.shape
```

```
(1986, 94)
```

Fitting t-SNE

```
from sklearn.manifold import TSNE

m = TSNE(learning_rate=50)
```

```
tsne_features = m.fit_transform(df_numeric)

tsne_features[1:4,:]
```

```
array([[-37.962185,  15.066088],
       [-21.873512,  26.334448],
       [ 13.97476 ,  22.590828]], dtype=float32)
```

Assigning t-SNE features to our dataset

```
tsne_features[1:4,:]
```

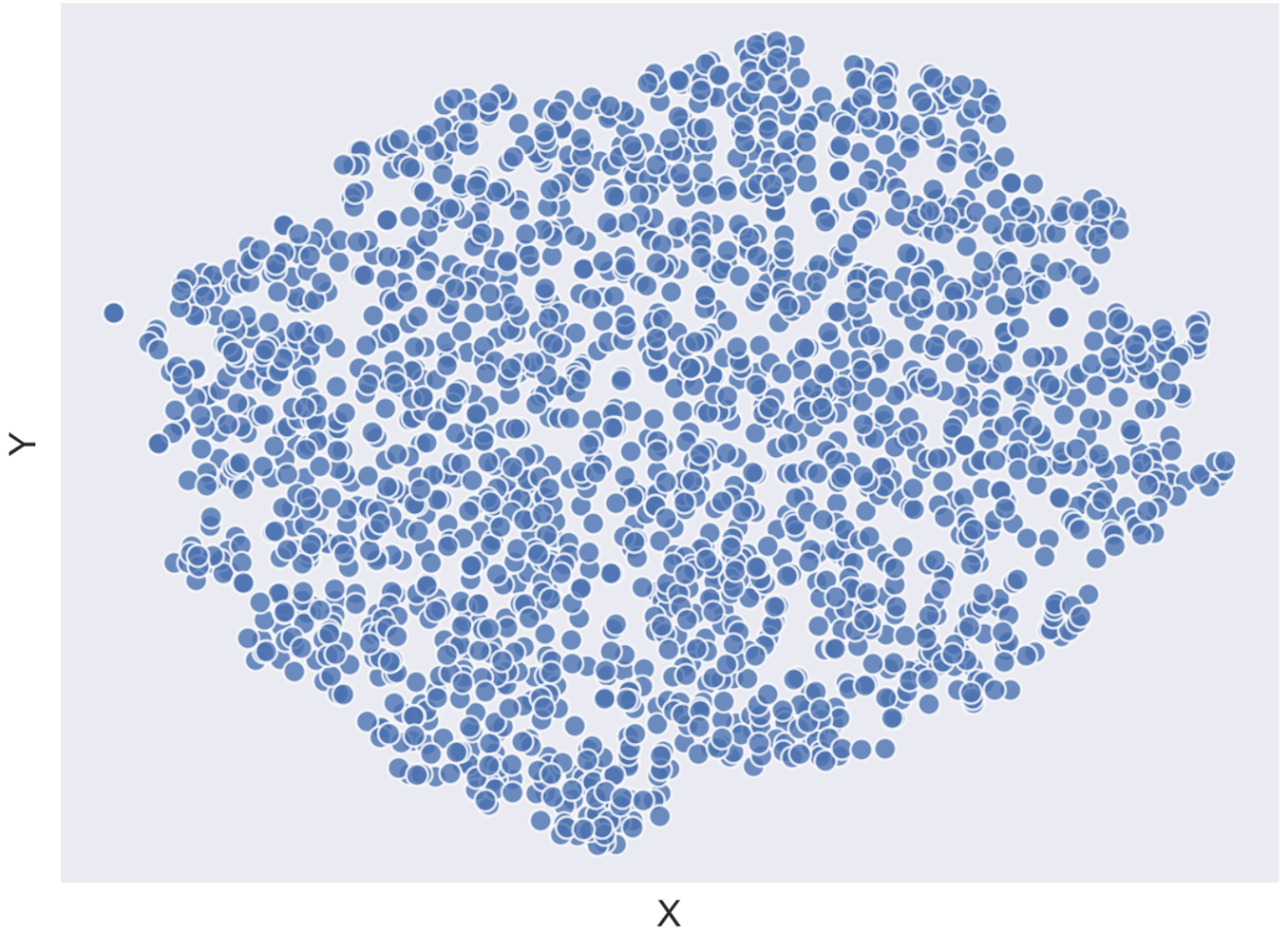
```
array([[-37.962185,  15.066088],
       [-21.873512,  26.334448],
       [ 13.97476 ,  22.590828]], dtype=float32)
```

```
df['x'] = tsne_features[:,0]

df['y'] = tsne_features[:,1]
```

Plotting t-SNE

```
import seaborn as sns

sns.scatterplot(x="x", y="y", data=df)

plt.show()
```

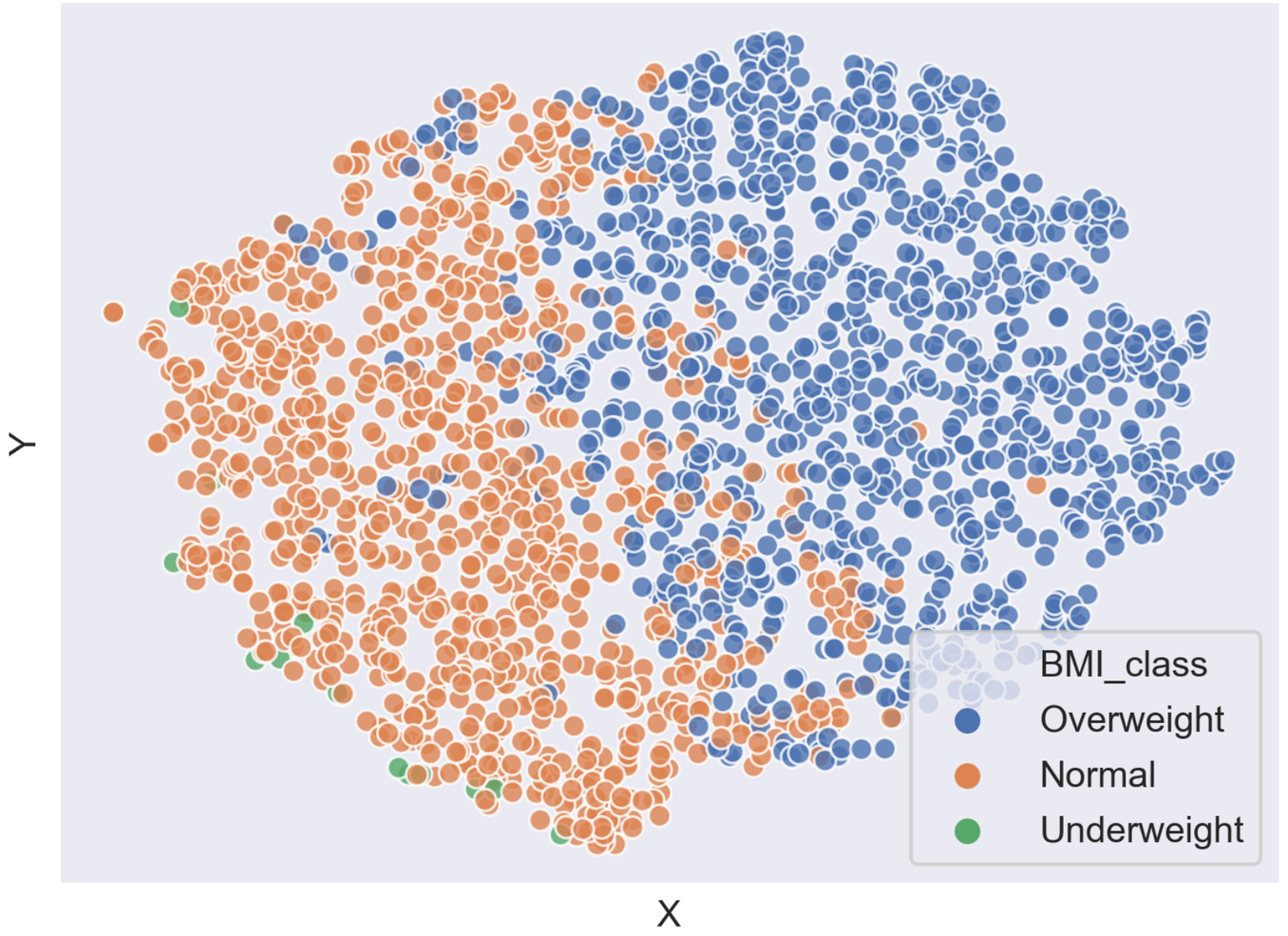
Plotting t-SNE



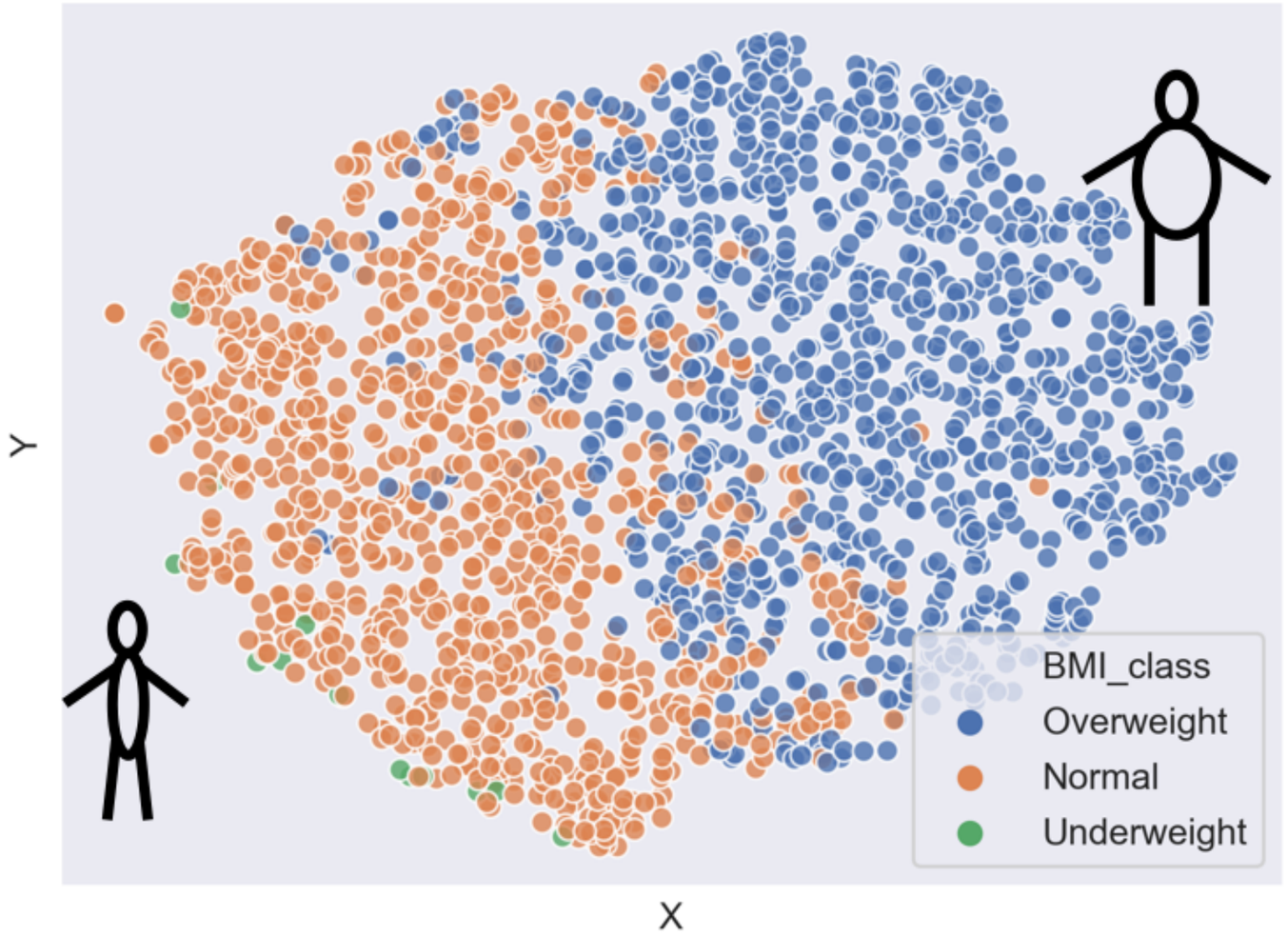
Coloring points according to BMI category

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.scatterplot(x="x", y="y", hue='BMI_class', data=df)

plt.show()
```

Coloring points according to BMI category



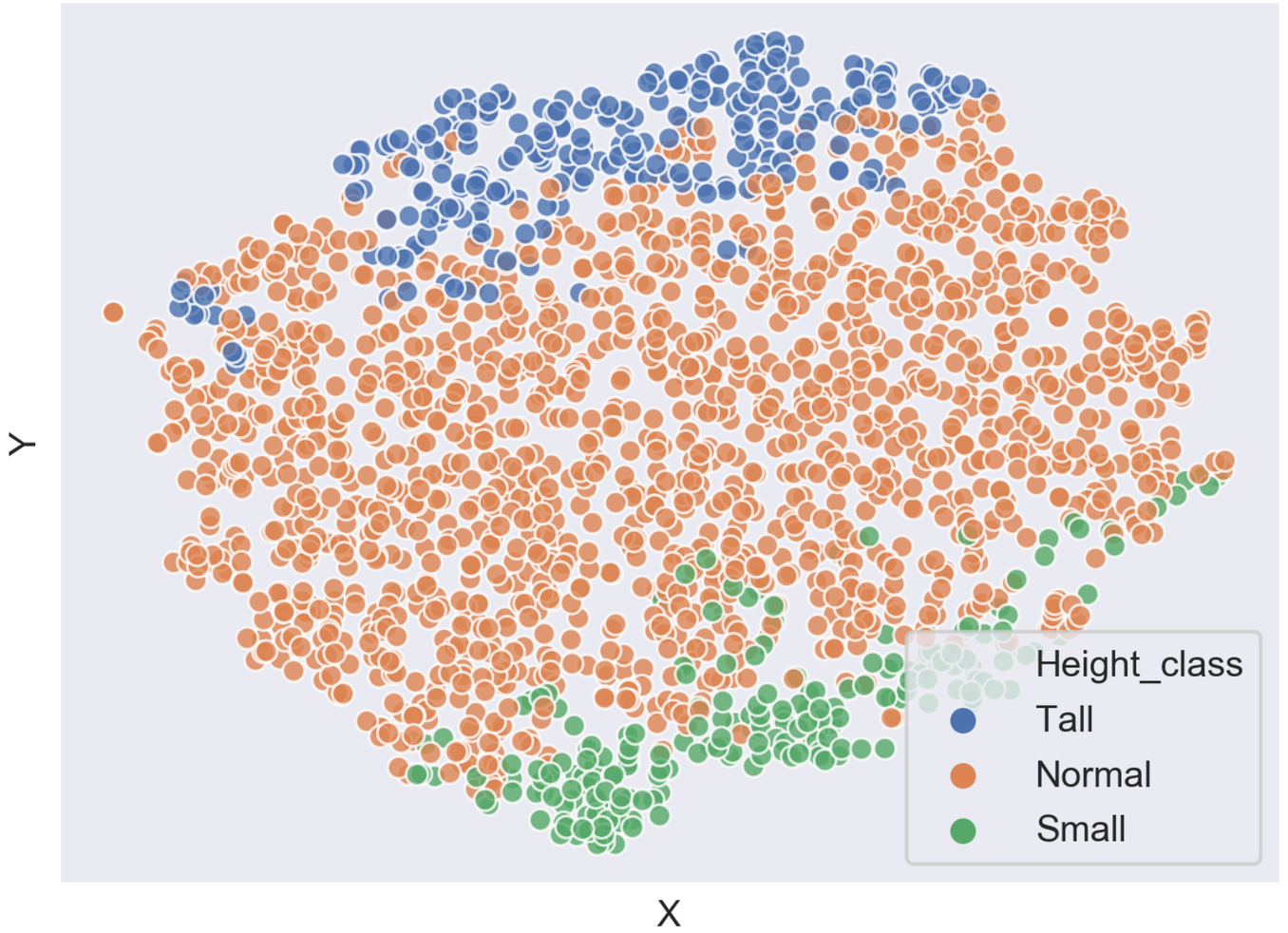
Coloring points according to BMI category



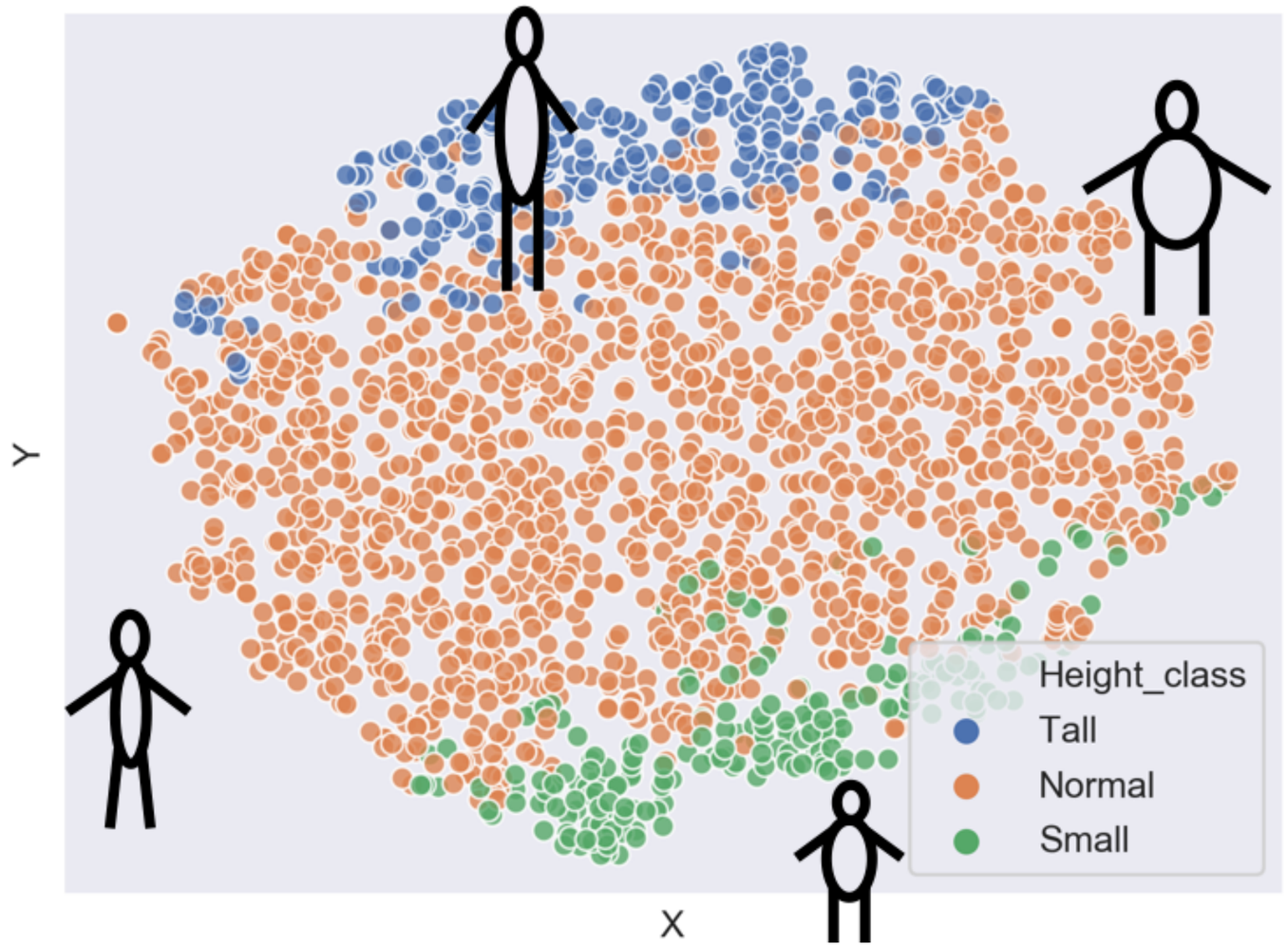
Coloring points according to height category

```
import seaborn as sns

import matplotlib.pyplot as plt

sns.scatterplot(x="x", y="y", hue='Height_class', data=df)

plt.show()
```

Coloring points according to height category



Coloring points according to height category



Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

The curse of dimensionality

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye

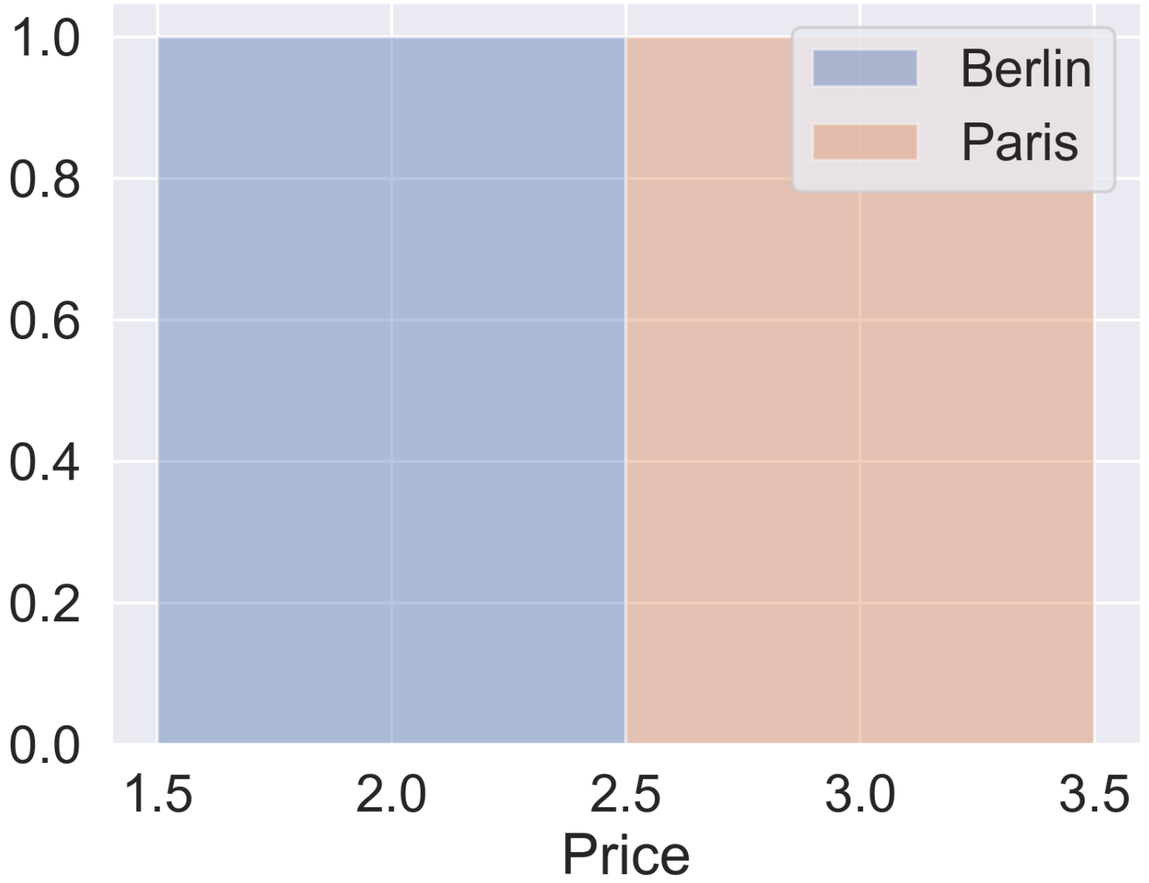
Machine Learning Engineer, Faktion

From observation to pattern

City	Price
Berlin	2
Paris	3

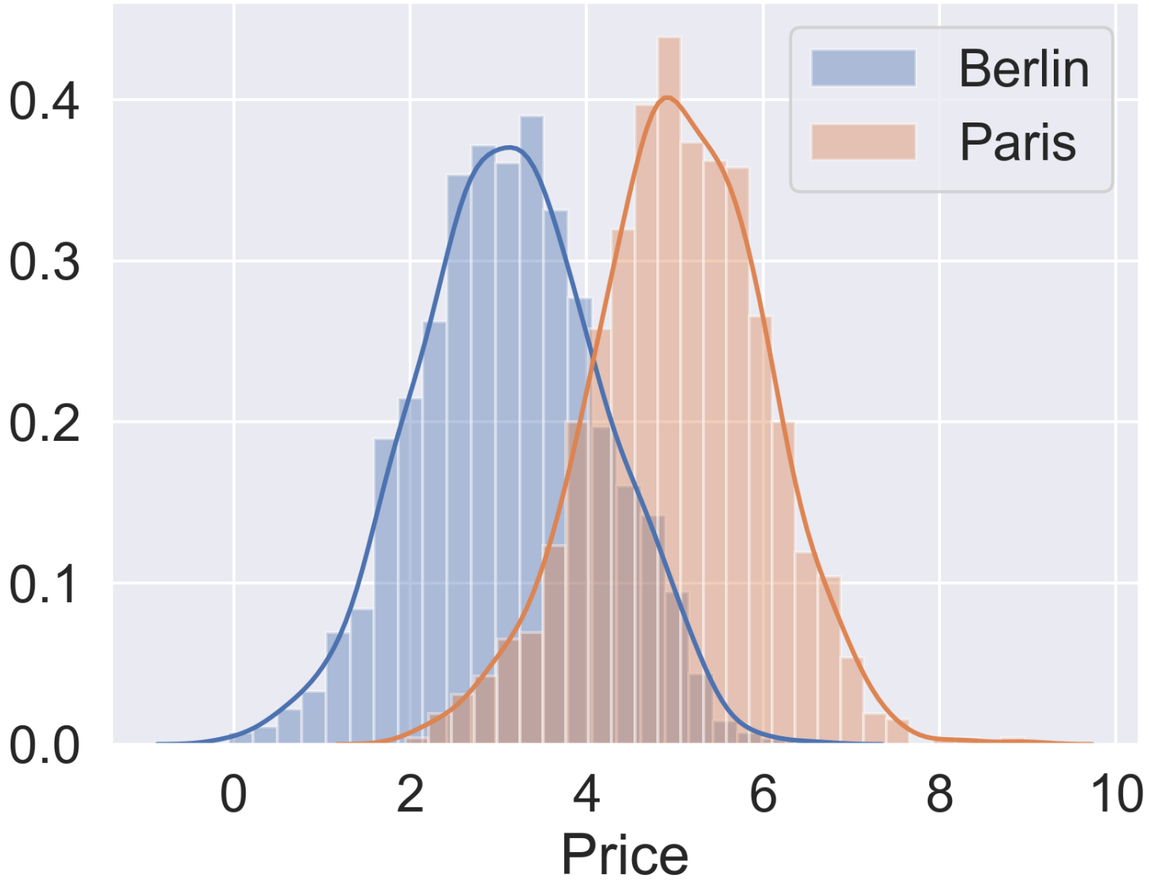
From observation to pattern

City	Price
Berlin	2
Paris	3



From observation to pattern

City	Price
Berlin	2.0
Berlin	3.1
Berlin	4.3
Paris	3.0
Paris	5.2
...	...



Building a city classifier - data split

Separate the feature we want to predict from the ones to train the model on.

```
y = house_df['City']

X = house_df.drop('City', axis=1)
```

Perform a 70% train and 30% test data split

```
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)
```

Building a city classifier - model fit

Create a Support Vector Machine Classifier and fit to training data

```
from sklearn.svm import SVC

svc = SVC()

svc.fit(X_train, y_train)
```

Building a city classifier - predict

```
from sklearn.metrics import accuracy_score

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

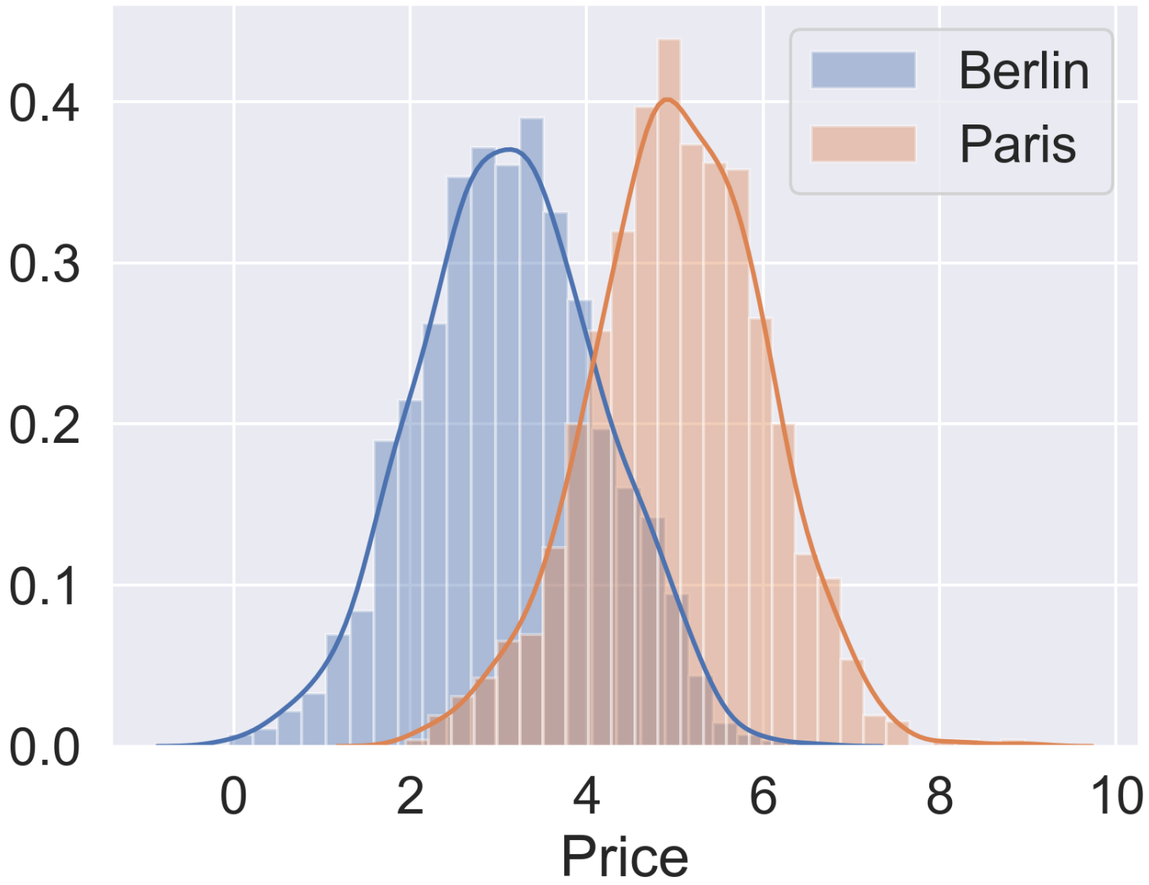
```
0.826
```

```
print(accuracy_score(y_train, svc.predict(X_train)))
```

```
0.832
```

Adding features

City	Price
Berlin	2.0
Berlin	3.1
Berlin	4.3
Paris	3.0
Paris	5.2
...	...



Adding features

City	Price	n_floors	n_bathroom	surface_m2
Berlin	2.0	1	1	190
Berlin	3.1	2	1	187
Berlin	4.3	2	2	240
Paris	3.0	2	1	170
Paris	5.2	2	2	290
...	...	...	...	...

Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

Features with missing values or little variance

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Machine Learning Engineer, Faktion

Creating a feature selector

```
print(ansur_df.shape)
```

```
(6068, 94)
```

```
from sklearn.feature_selection import VarianceThreshold

sel = VarianceThreshold(threshold=1)
sel.fit(ansur_df)

mask = sel.get_support()
print(mask)
```

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

Applying a feature selector

```
print(ansur_df.shape)
```

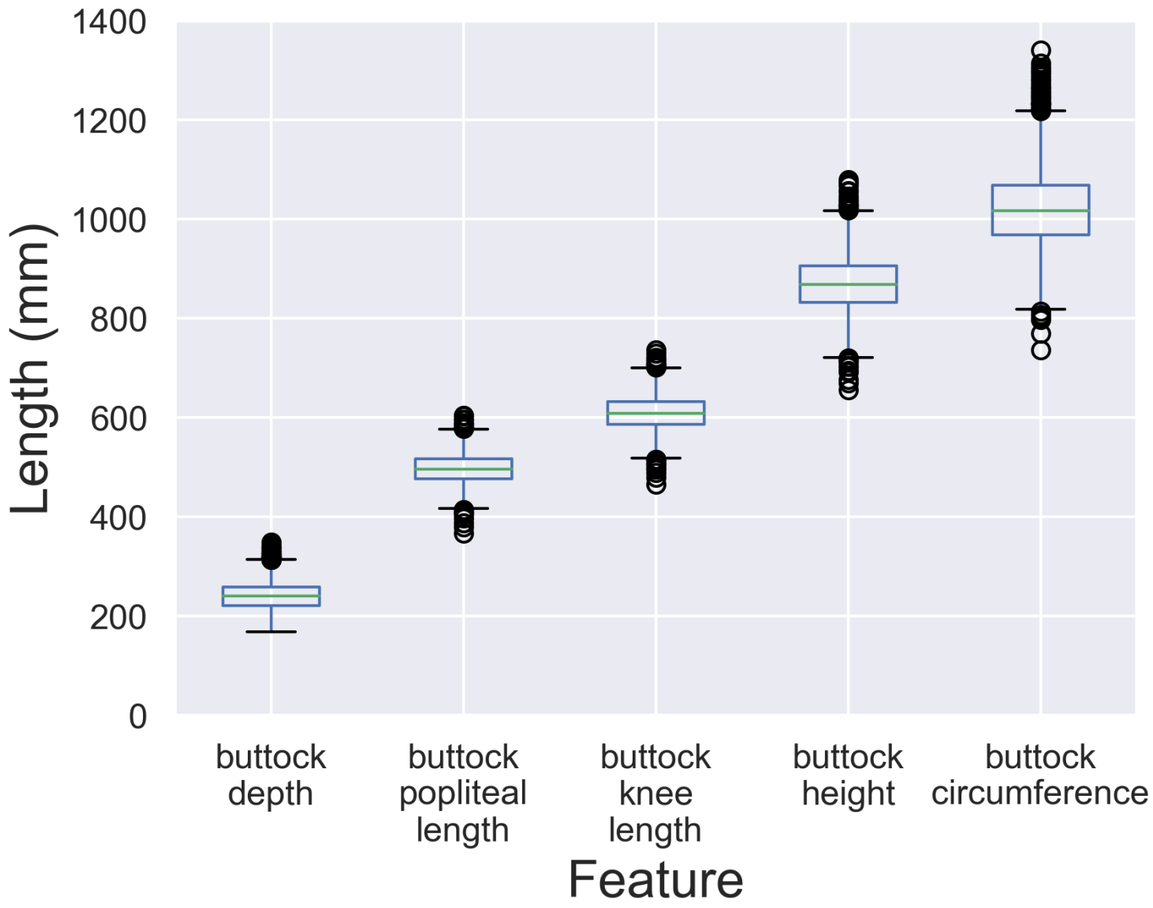
```
(6068, 94)
```

```
reduced_df = ansur_df.loc[:, mask]
print(reduced_df.shape)
```

```
(6068, 93)
```

Variance selector caveats

```
buttock_df.boxplot()
```



Normalizing the variance

```
from sklearn.feature_selection import VarianceThreshold

sel = VarianceThreshold(threshold=0.005)

sel.fit(ansur_df / ansur_df.mean())
mask = sel.get_support()
reduced_df = ansur_df.loc[:, mask]
print(reduced_df.shape)
```

```
(6068, 45)
```

Missing value selector

Name	Type 1	Type 2	Total	HP	Attack	Defense
Bulbasaur	Grass	Poison	318	45	49	49
Ivysaur	Grass	Poison	405	60	62	63
Venusaur	Grass	Poison	525	80	82	83
Charmander	Fire	NaN	309	39	52	43
Charmeleon	Fire	NaN	405	58	64	58

Missing value selector

Name	Type 1	Type 2	Total	HP	Attack	Defense
Bulbasaur	Grass	Poison	318	45	49	49
Ivysaur	Grass	Poison	405	60	62	63
Venusaur	Grass	Poison	525	80	82	83
Charmander	Fire	NaN	309	39	52	43
Charmeleon	Fire	NaN	405	58	64	58

Identifying missing values

```
pokemon_df.isna()
```

Name	Type 1	Type 2	Total	HP	Attack	Defense
False	False	False	False	False	False	False
False	False	False	False	False	False	False
False	False	False	False	False	False	False
False	False	True	False	False	False	False
False	False	True	False	False	False	False

Counting missing values

```
pokemon_df.isna().sum()
```

```
Name          0
Type 1        0
Type 2      386
Total         0
HP            0
Attack        0
Defense       0
dtype: int64
```

Counting missing values

```
pokemon_df.isna().sum() / len(pokemon_df)
```

```
Name        0.00
Type 1      0.00
Type 2      0.48
Total       0.00
HP          0.00
Attack      0.00
Defense     0.00
dtype: float64
```

Applying a missing value threshold

```
# Fewer than 30% missing values = True value
mask = pokemon_df.isna().sum() / len(pokemon_df) < 0.3
print(mask)
```

```
Name         True
Type 1       True
Type 2      False
Total        True
HP           True
Attack       True
Defense      True
dtype: bool
```

Applying a missing value threshold

```
reduced_df = pokemon_df.loc[:, mask]


reduced_df.head()
```

Name	Type 1	Total	HP	Attack	Defense
Bulbasaur	Grass	318	45	49	49
Ivysaur	Grass	405	60	62	63
Venusaur	Grass	525	80	82	83
Charmander	Fire	309	39	52	43
Charmeleon	Fire	405	58	64	58

Let's practice

DIMENSIONALITY REDUCTION IN PYTHON

Pairwise correlation

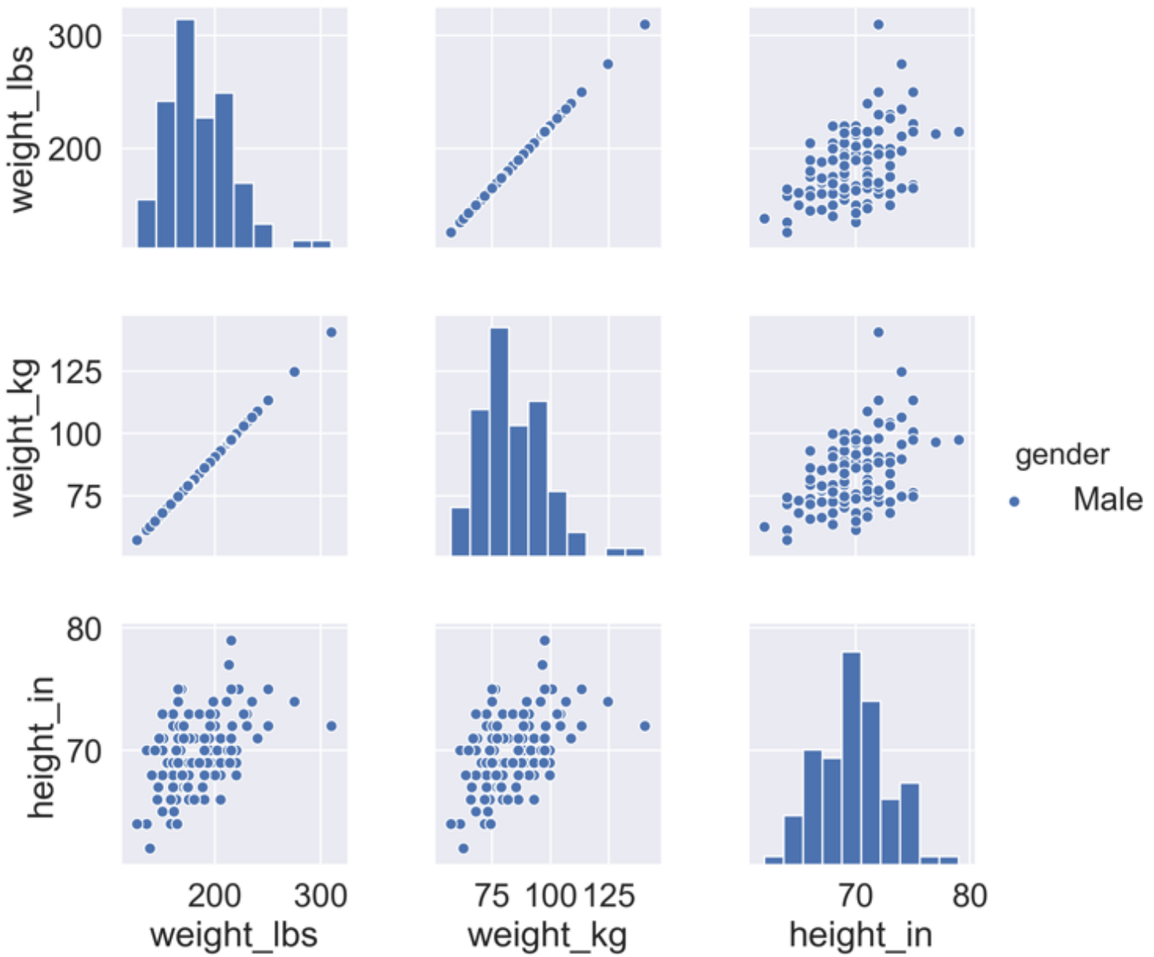
DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye

Machine Learning Engineer, Faktion

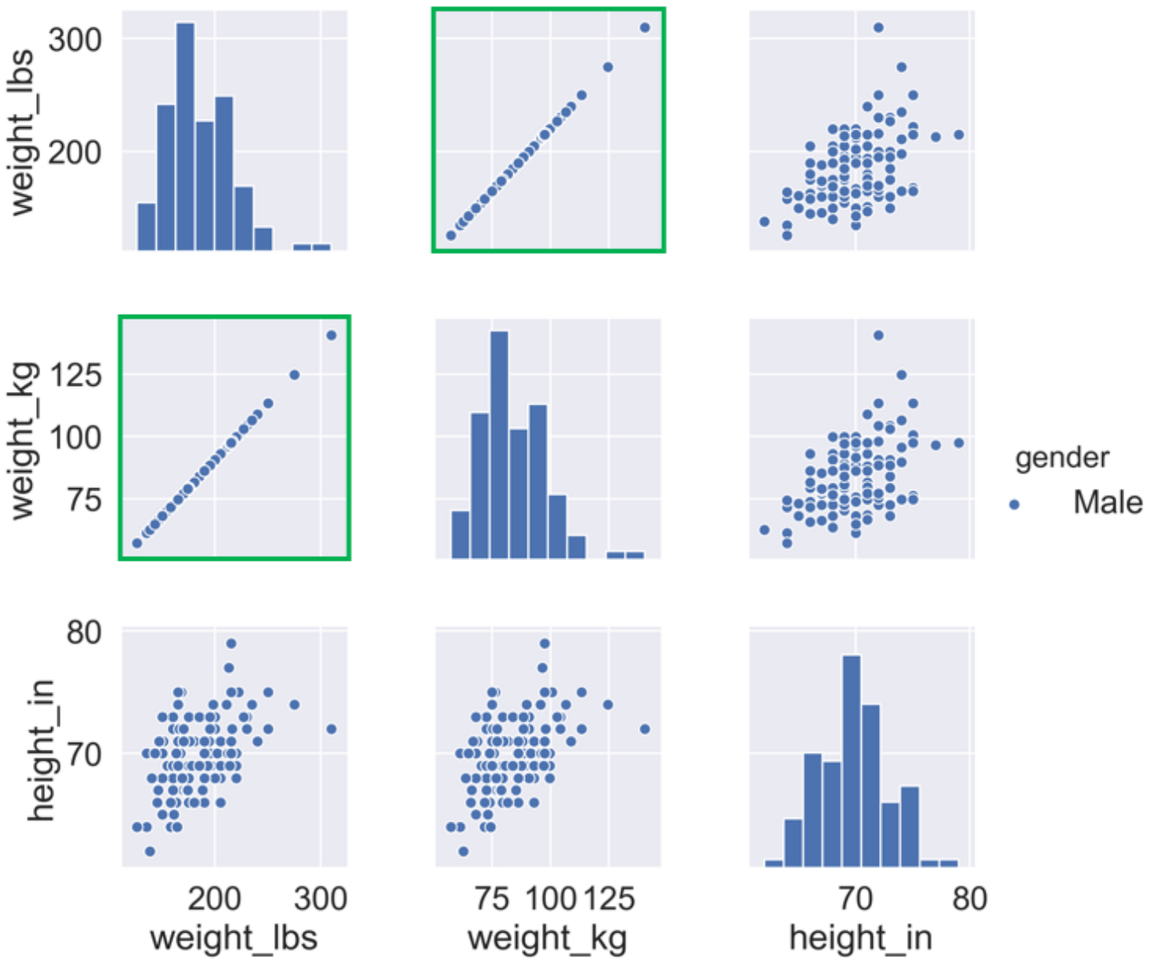
Pairwise correlation

```
sns.pairplot(ansur, hue="gender")
```

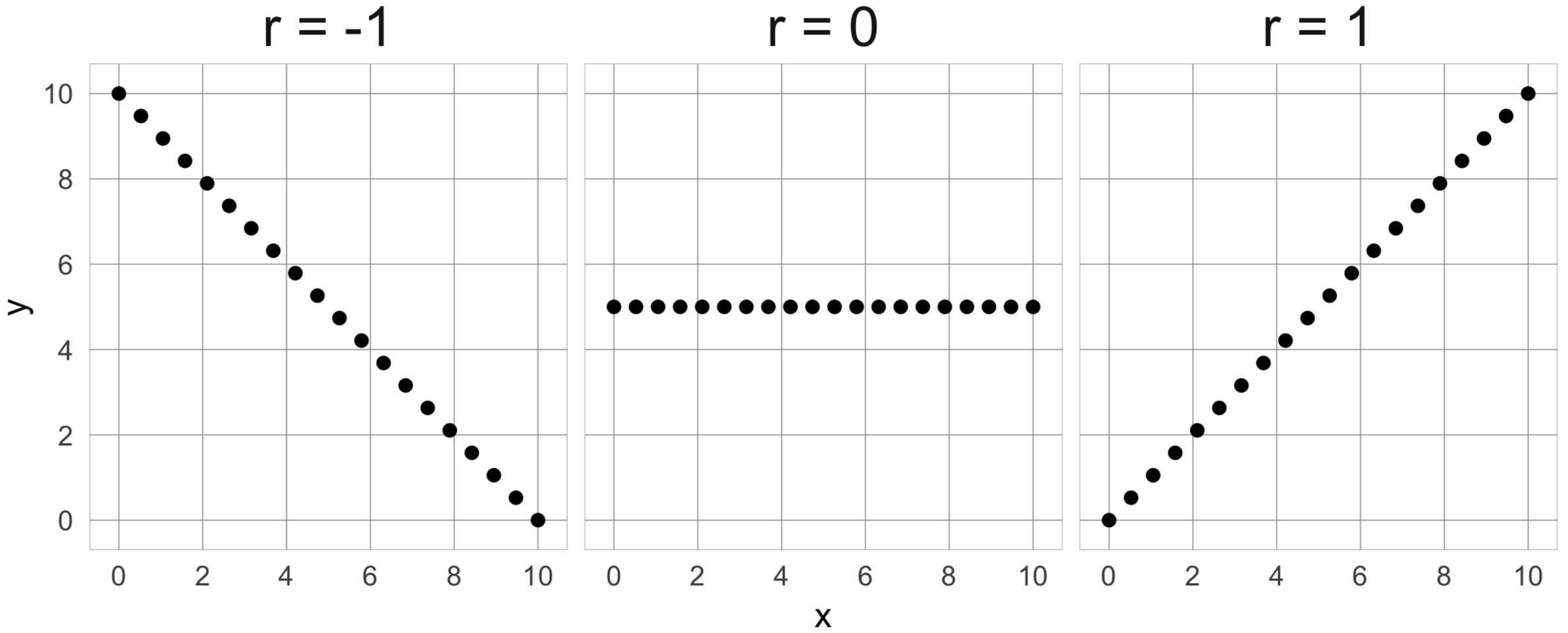


Pairwise correlation

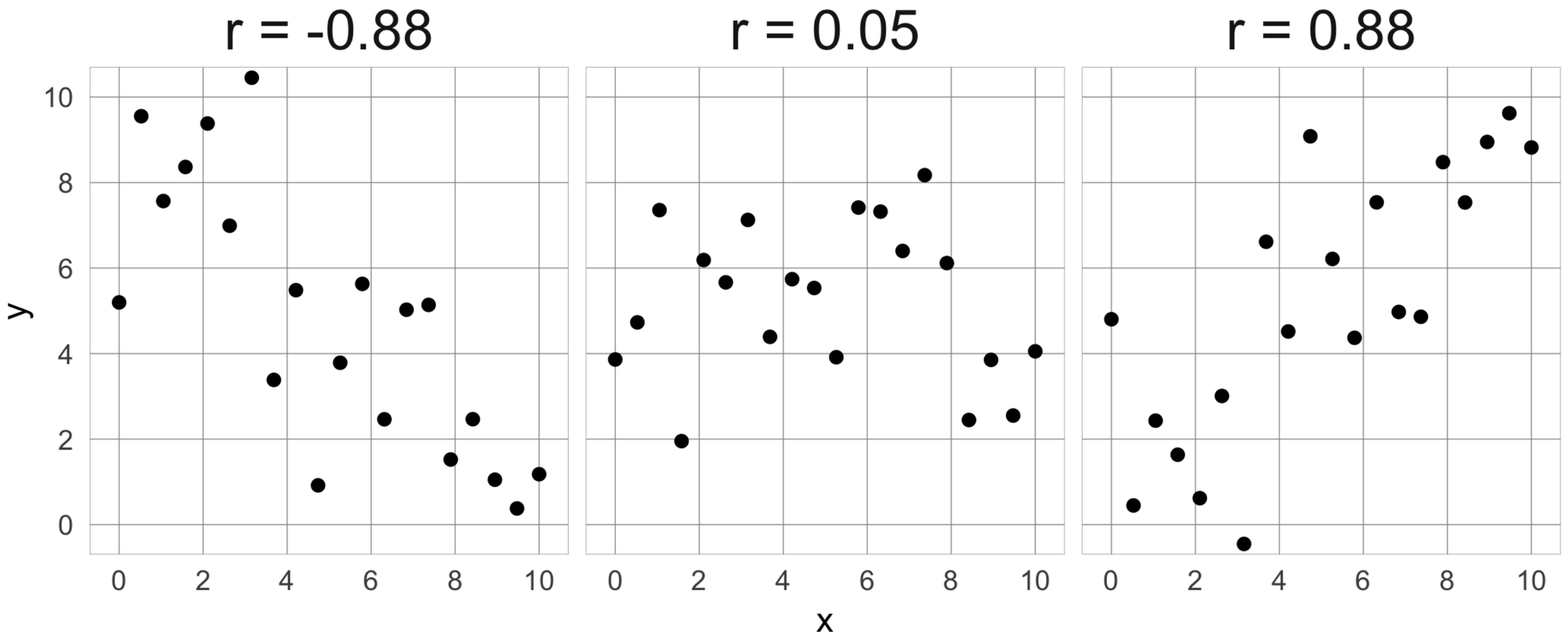
```
sns.pairplot(ansur, hue="gender")
```



Correlation coefficient



Correlation coefficient



Correlation matrix

```
weights_df.corr()
```

	weight_lbs	weight_kg	height_in
weight_lbs	1.00	1.00	0.47
weight_kg	1.00	1.00	0.47
height_in	0.47	0.47	1.00

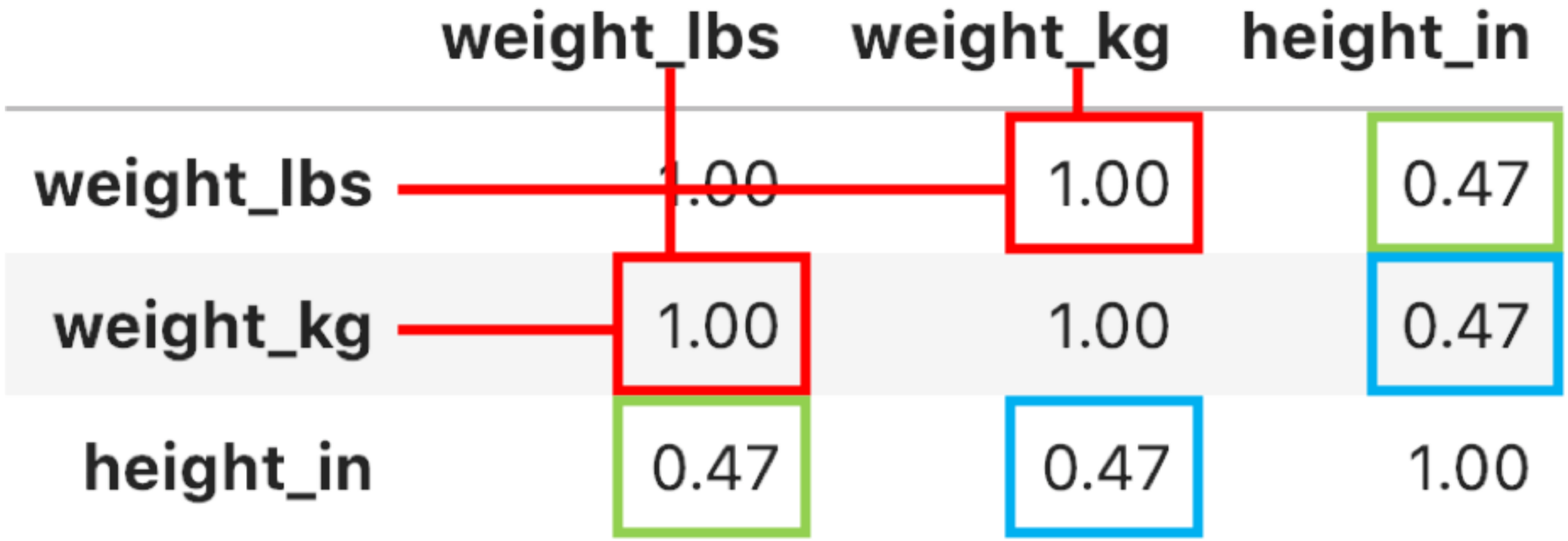
Correlation matrix

```
weights_df.corr()
```

	weight_lbs	weight_kg	height_in
weight_lbs	1.00	1.00	0.47
weight_kg	1.00	1.00	0.47
height_in	0.47	0.47	1.00

Correlation matrix

```
weights_df.corr()
```



	weight_lbs	weight_kg	height_in
weight_lbs	1.00	1.00	0.47
weight_kg	1.00	1.00	0.47
height_in	0.47	0.47	1.00

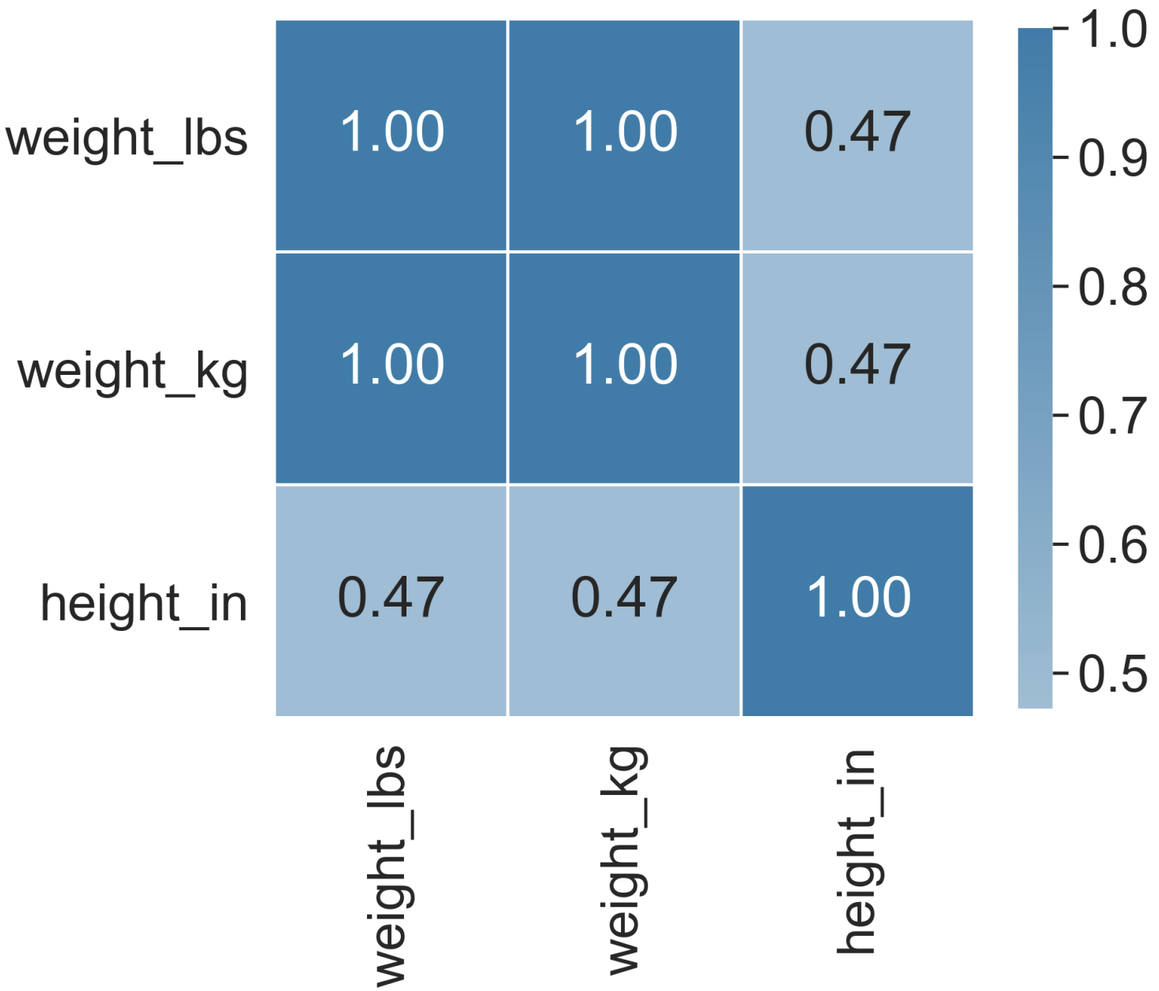
Correlation matrix

```
weights_df.corr()
```

	weight_lbs	weight_kg	height_in
weight_lbs	1.00	1.00	0.47
weight_kg	1.00	1.00	0.47
height_in	0.47	0.47	1.00

Visualizing the correlation matrix

```
cmap = sns.diverging_palette(h_neg=10,
                             h_pos=240,
                             as_cmap=True)

sns.heatmap(weights_df.corr(), center=0,
            cmap=cmap, linewidths=1,
            annot=True, fmt=".2f")
```



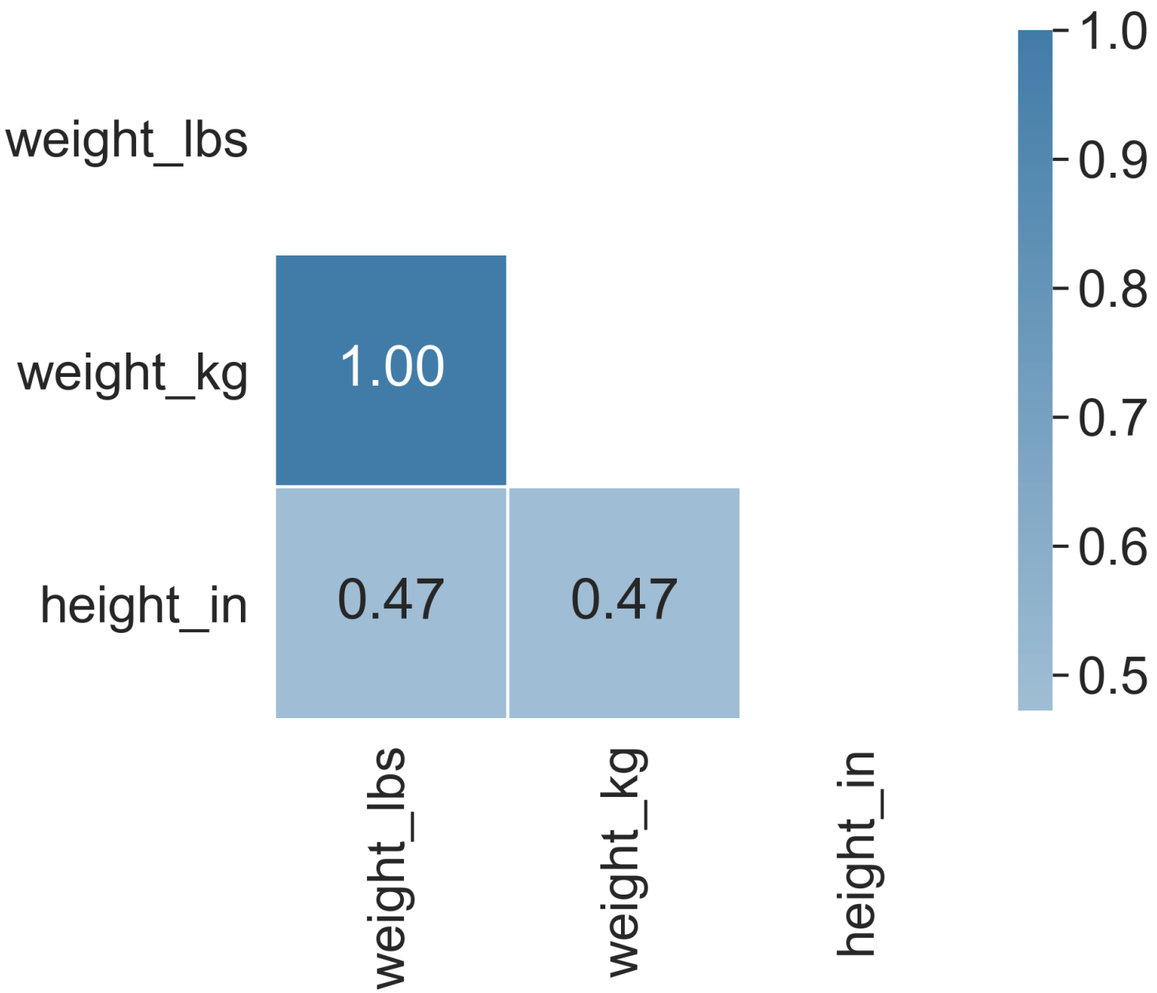
Visualizing the correlation matrix

```
corr = weights_df.corr()


mask = np.triu(np.ones_like(corr, dtype=bool))
```

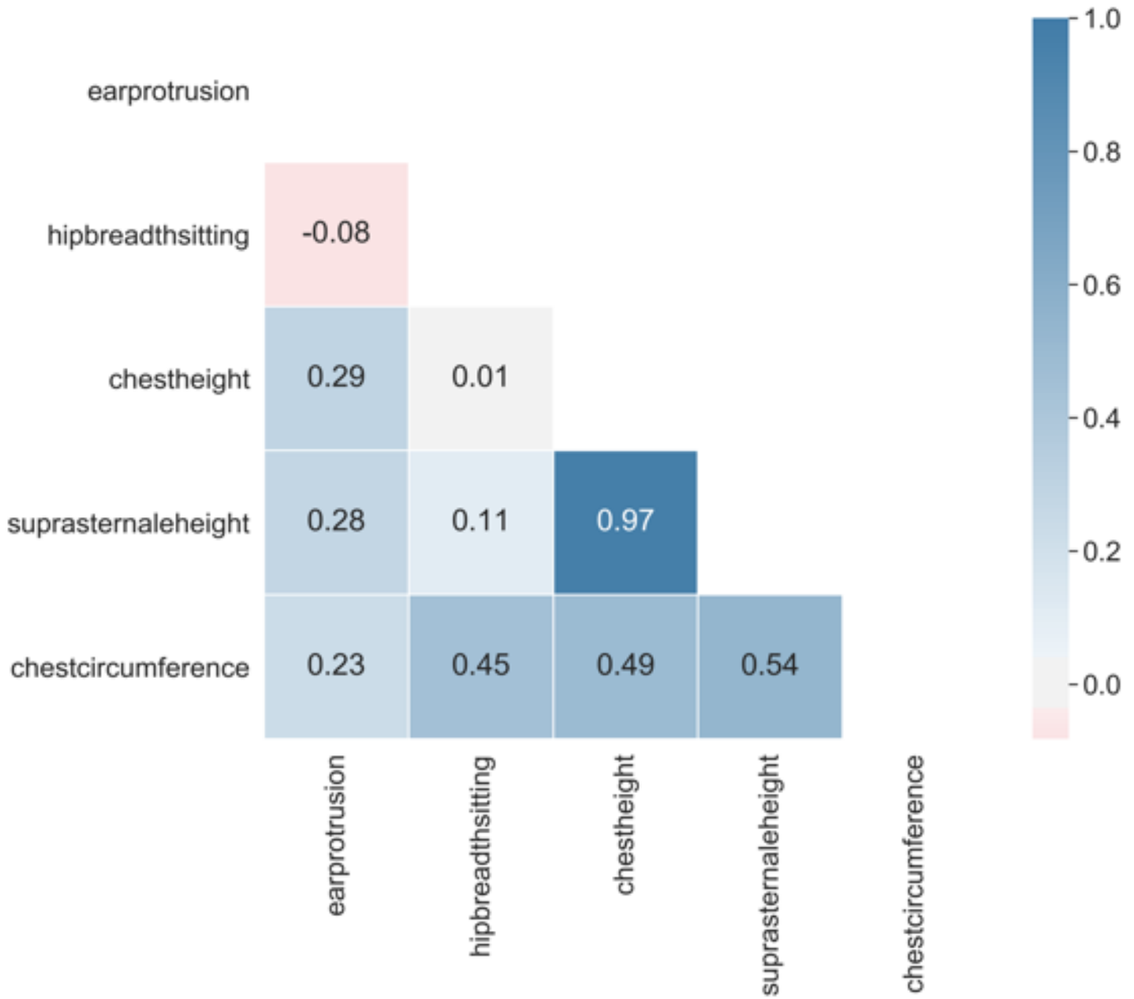
```
array([[ True,  True,  True],
       [False,  True,  True],
       [False, False,  True]])
```

Visualizing the correlation matrix

```
sns.heatmap(weights_df.corr(), mask=mask,
            center=0, cmap=cmap, linewidths=1,
            annot=True, fmt=".2f")
```



Visualising the correlation matrix



Let's practice!

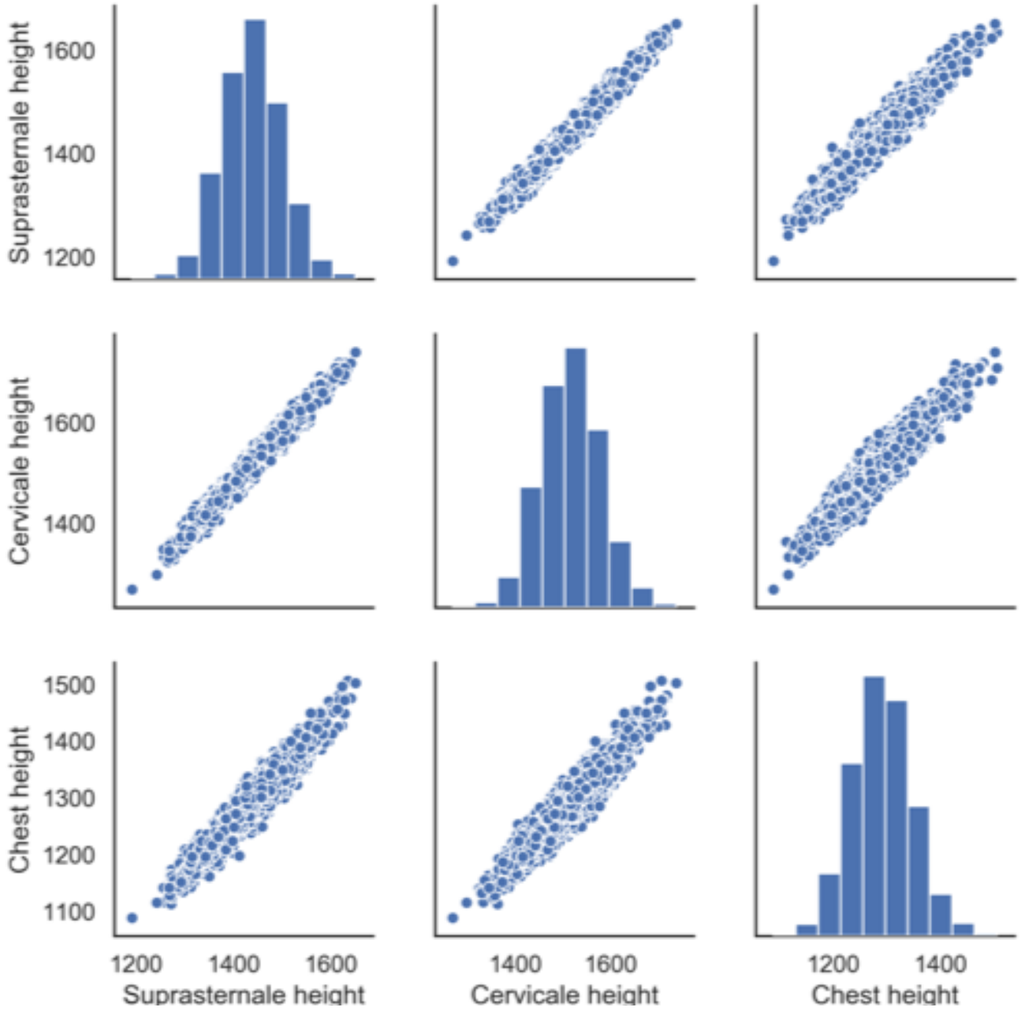
DIMENSIONALITY REDUCTION IN PYTHON

Removing highly correlated features

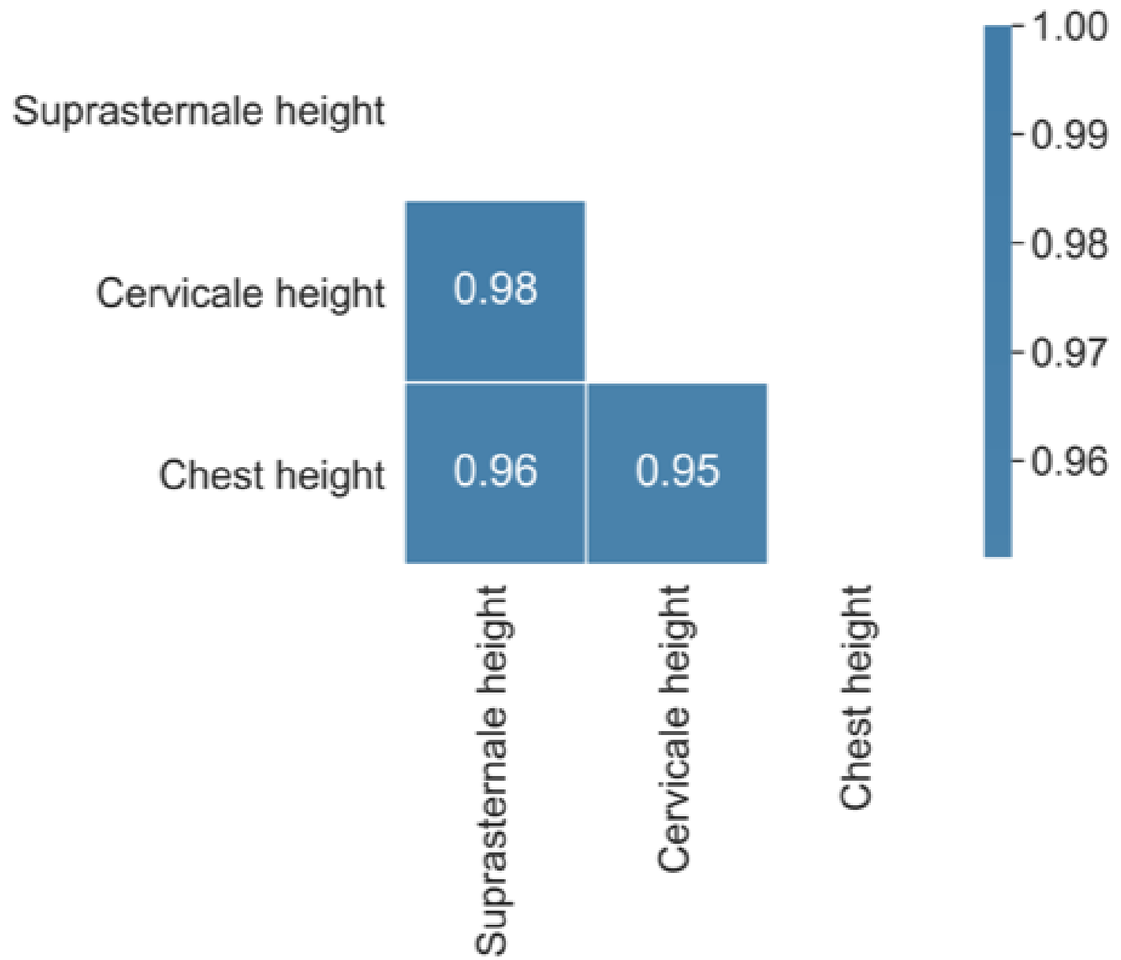
DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Machine Learning Engineer, Faktion

Highly correlated data



Highly correlated features



Removing highly correlated features

```
# Create positive correlation matrix
corr_df = chest_df.corr().abs()

# Create and apply mask
mask = np.triu(np.ones_like(corr_df, dtype=bool))

tri_df = corr_matrix.mask(mask)


tri_df
```

	Suprasternale height	Cervicale height	Chest height
Suprasternale height	NaN	NaN	NaN
Cervicale height	0.983033	NaN	NaN
Chest height	0.956111	0.951101	NaN

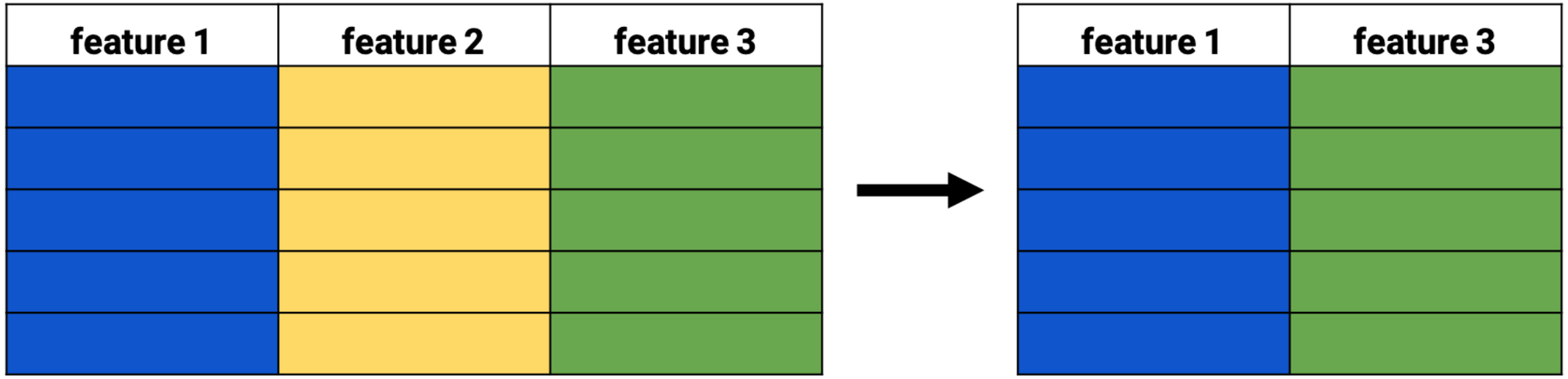
Removing highly correlated features

```
# Find columns that meet treshold
to_drop = [c for c in tri_df.columns if any(tri_df[c] > 0.95)]

print(to_drop)
```

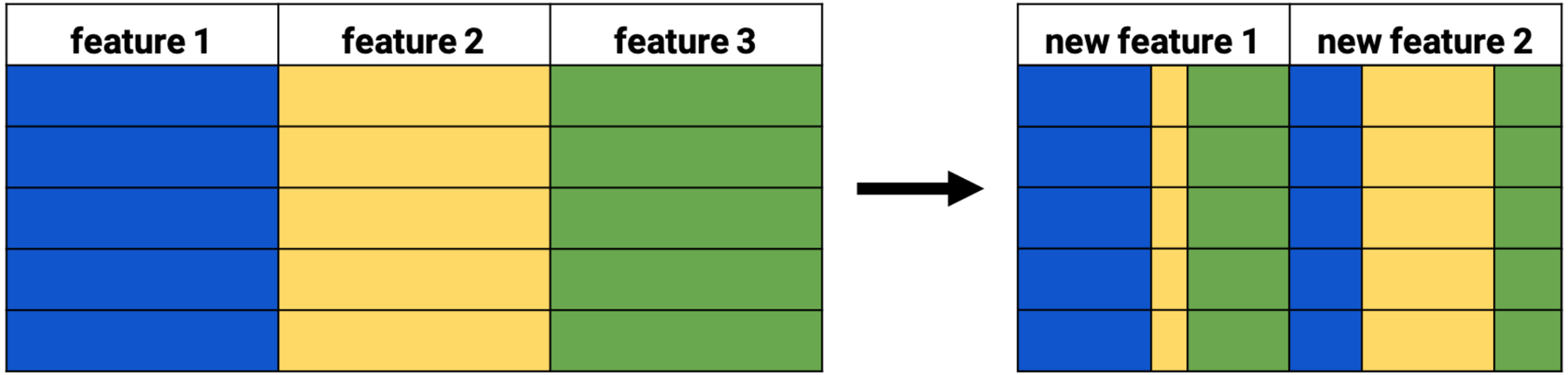
```
['Suprasternale height', 'Cervicale height']
```

```
# Drop those columns
reduced_df = chest_df.drop(to_drop, axis=1)
```

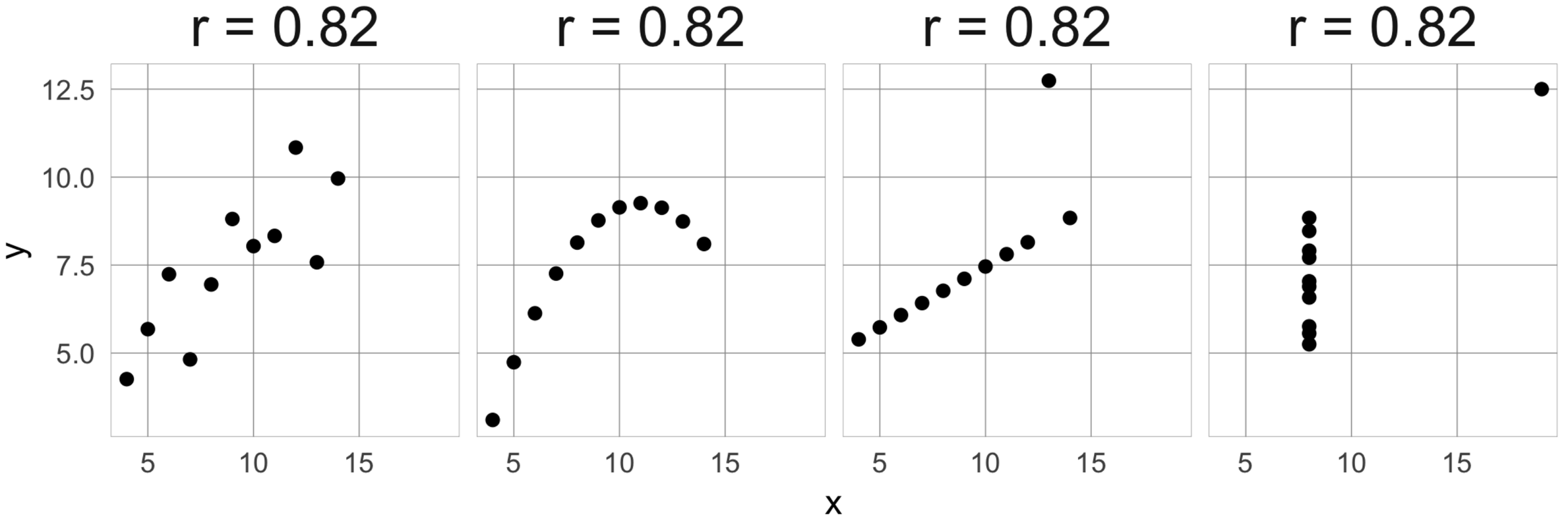
Feature selection



Feature extraction

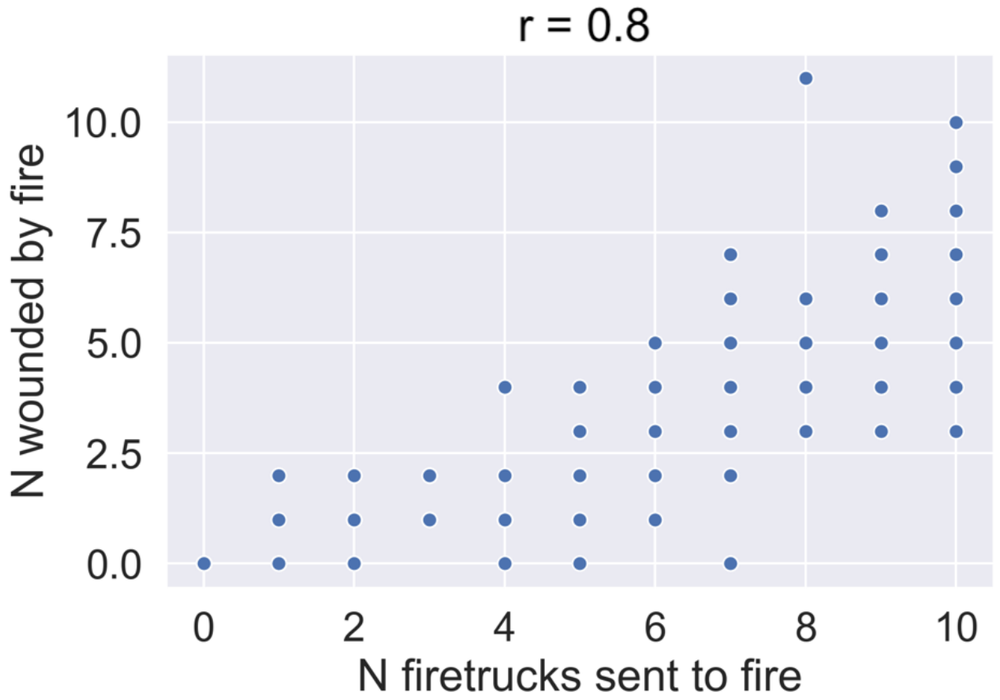


Correlation caveats - Anscombe's quartet



Correlation caveats - causation

```
sns.scatterplot(x="N firetrucks sent to fire",
                y="N wounded by fire",data=fire_df)
```



Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

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

Creating a logistic regression model

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


scaler = StandardScaler()
X_std = scaler.fit_transform(X)

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

Creating a logistic regression model

```
lr = LogisticRegression()
lr.fit(X_train, y_train)


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

```
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

```
scaler = StandardScaler()
X_std = scaler.fit_transform(X.drop('handlength', axis=1))

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


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


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

```
0.99
```

Recursive Feature Elimination

```
from sklearn.feature_selection import RFE

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

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

Dropping one feature at a time is safest since it 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)))
```

```
0.99
```

Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

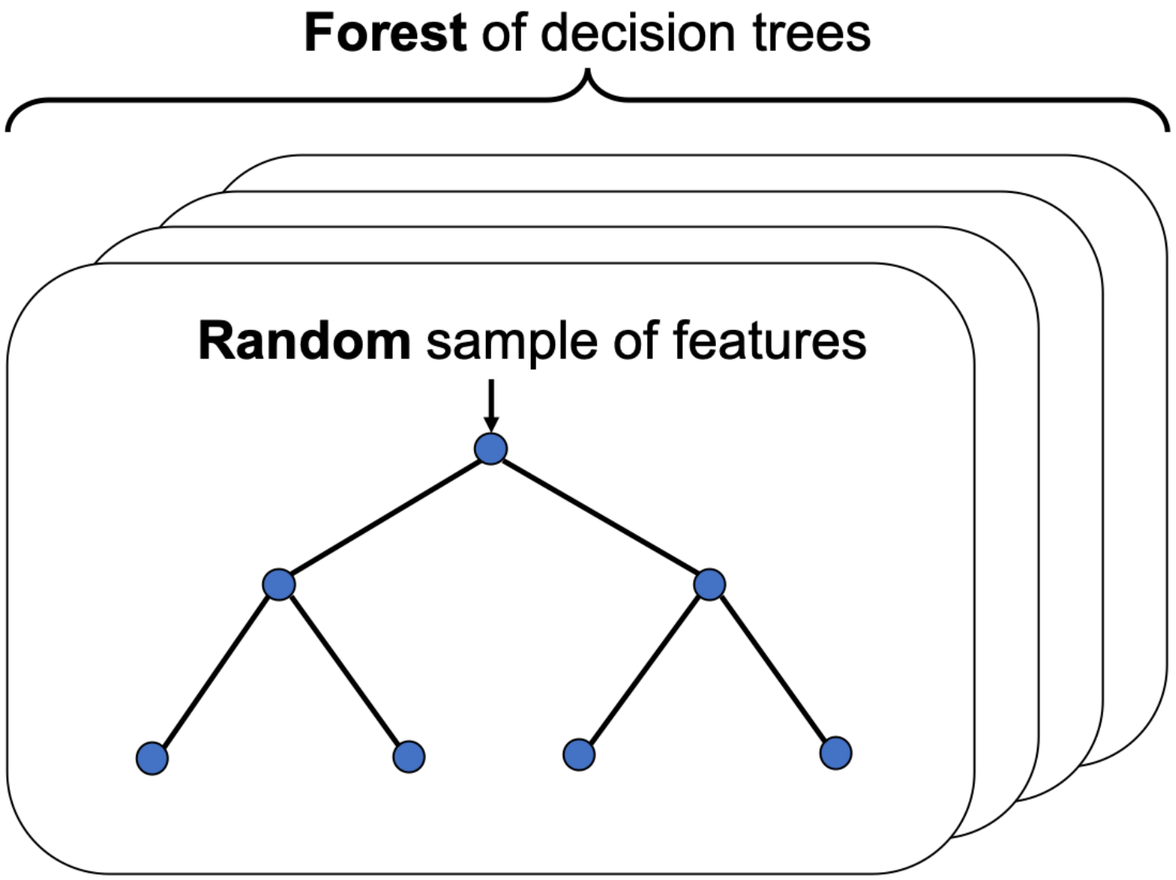
Tree-based feature selection

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye

Machine Learning Engineer, Faktion

Random forest classifier

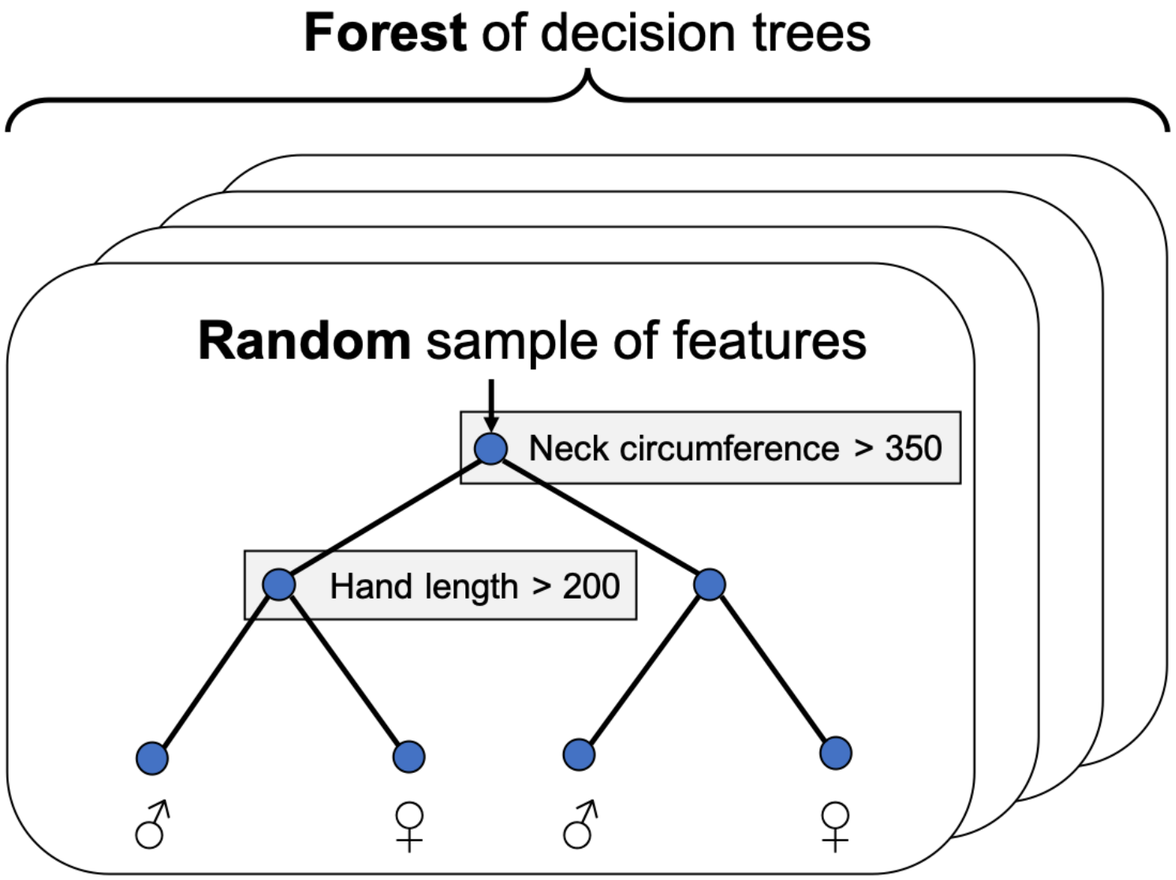


Random forest classifier

```
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

```
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.
```

```
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

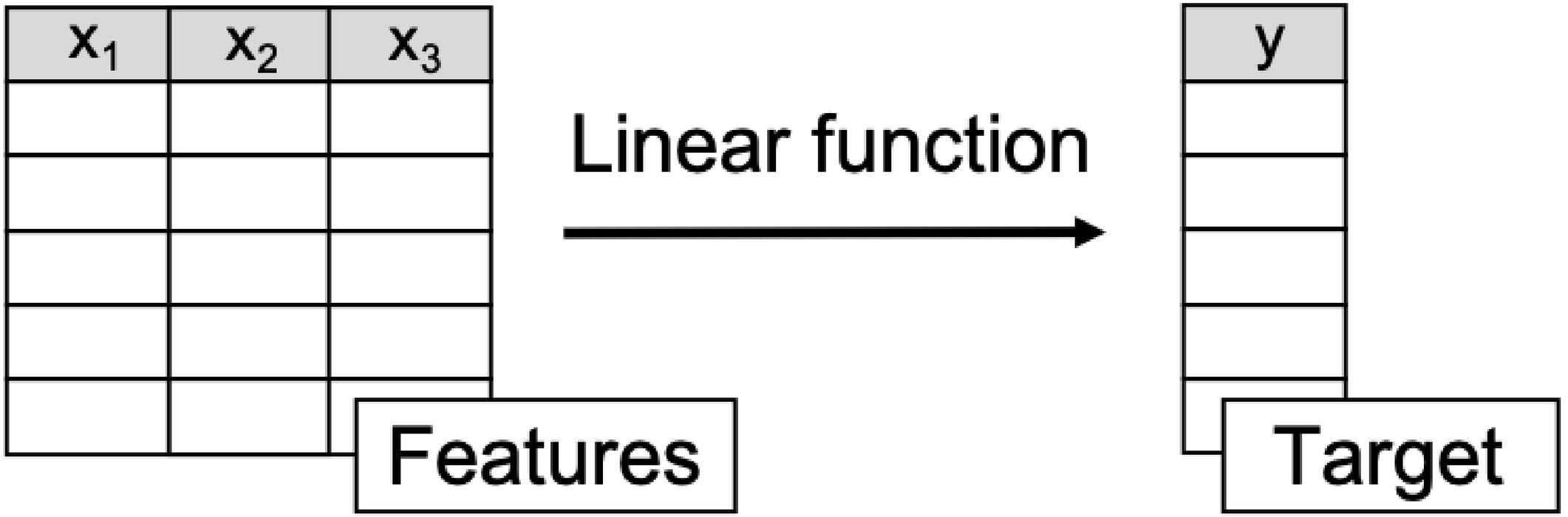
Regularized linear regression

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye

Machine Learning Engineer, Faktion

Linear model concept

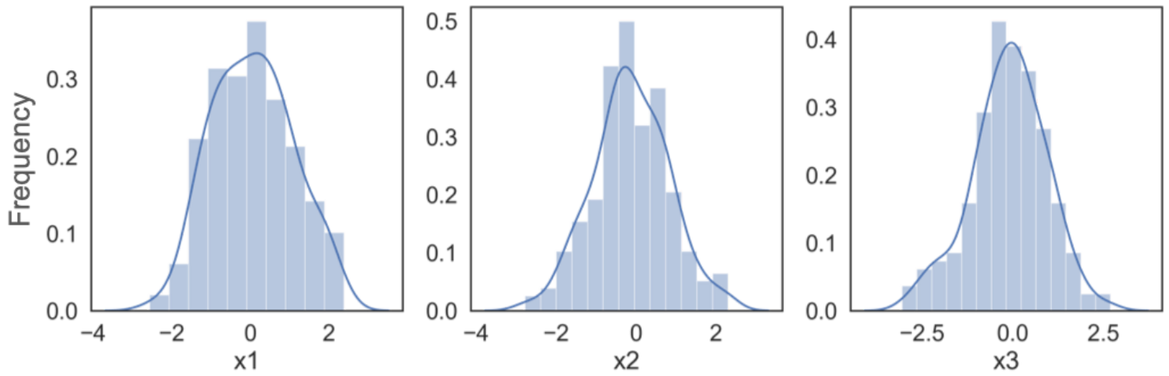


Creating our own dataset

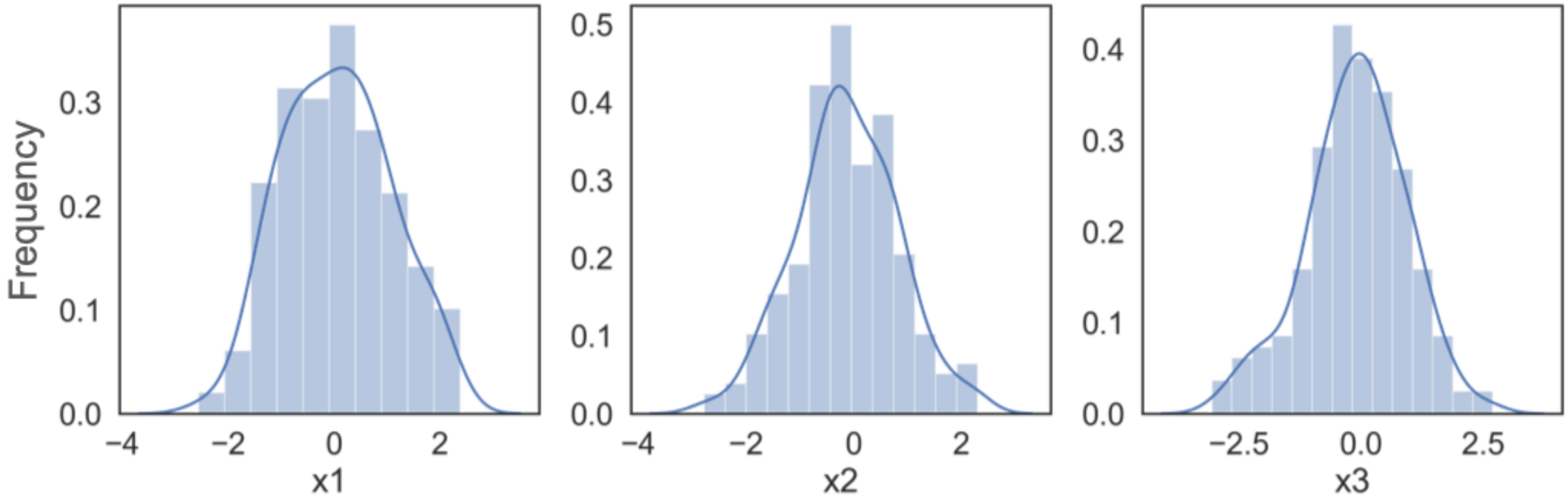
x1	x2	x3
1.76	-0.37	-0.60
0.40	-0.24	-1.12
0.98	1.10	0.77
...	...	...

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

```
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]
```

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

```
19.8
```

Linear regression in Python

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

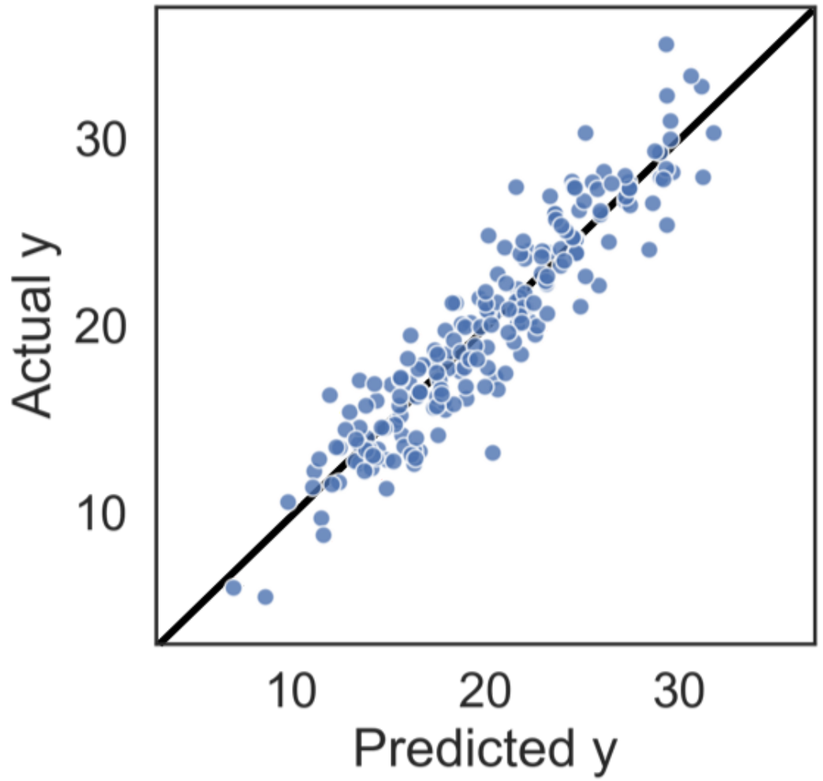
```
0.976
```

Linear regression in Python

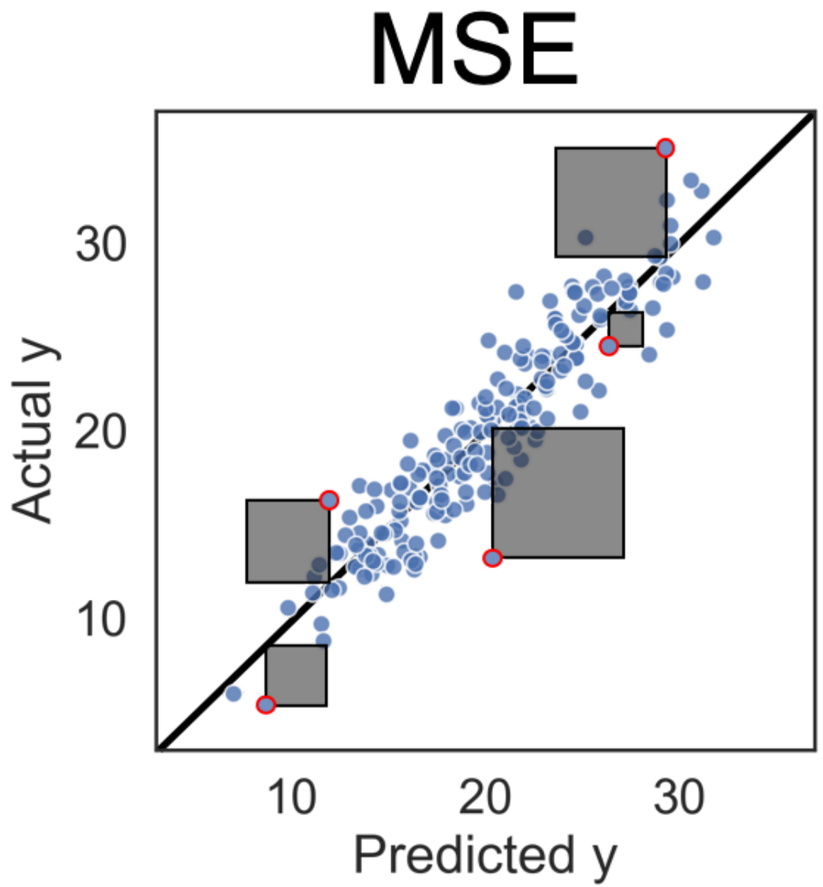
```
from sklearn.linear_model import LinearRegression

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

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

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

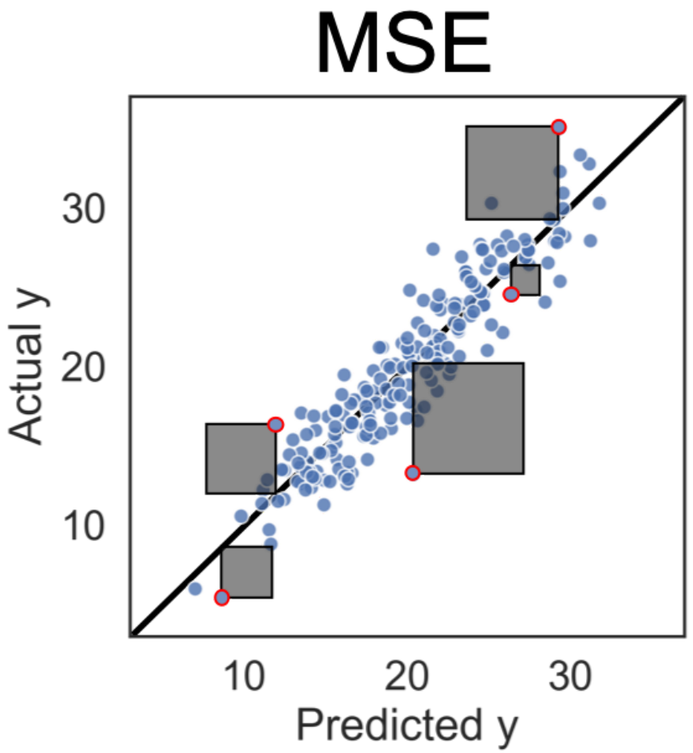
Loss function: Mean Squared Error



Loss function: Mean Squared Error

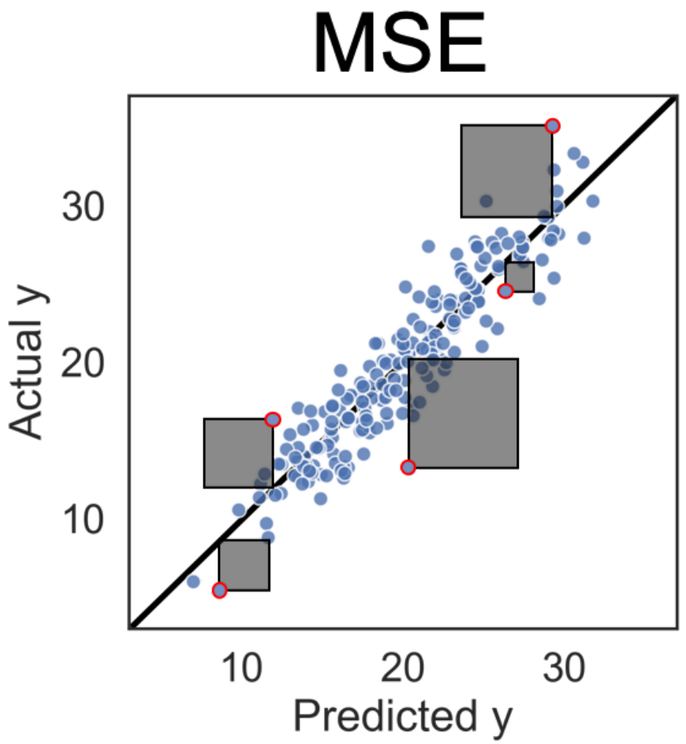


Adding regularization



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

Adding regularization



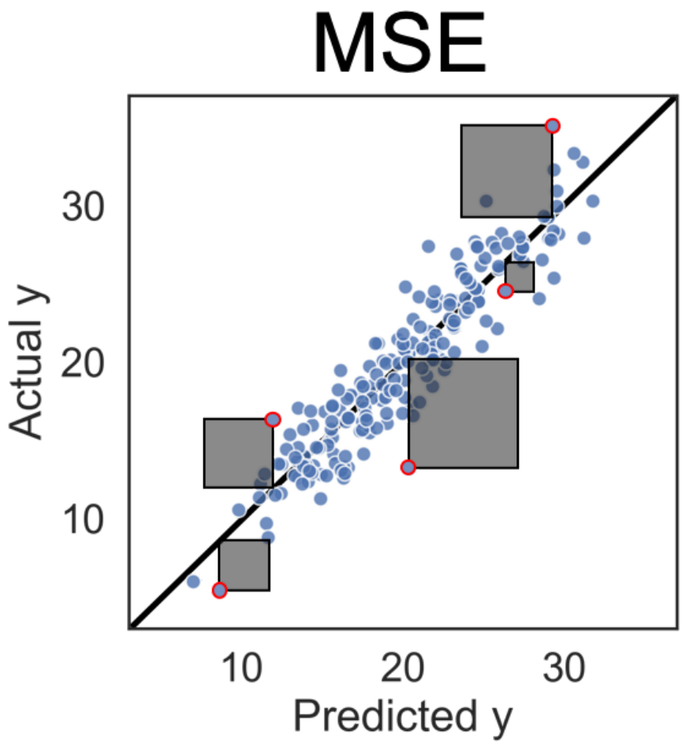
Tries to make model accurate

+ $\alpha(|\beta_1|+|\beta_2|+|\beta_3|)$

Regularization term

Tries to make model simple

Adding regularization



Tries to make model accurate

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

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 and

Lasso regressor

```
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.  ]
```

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

Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

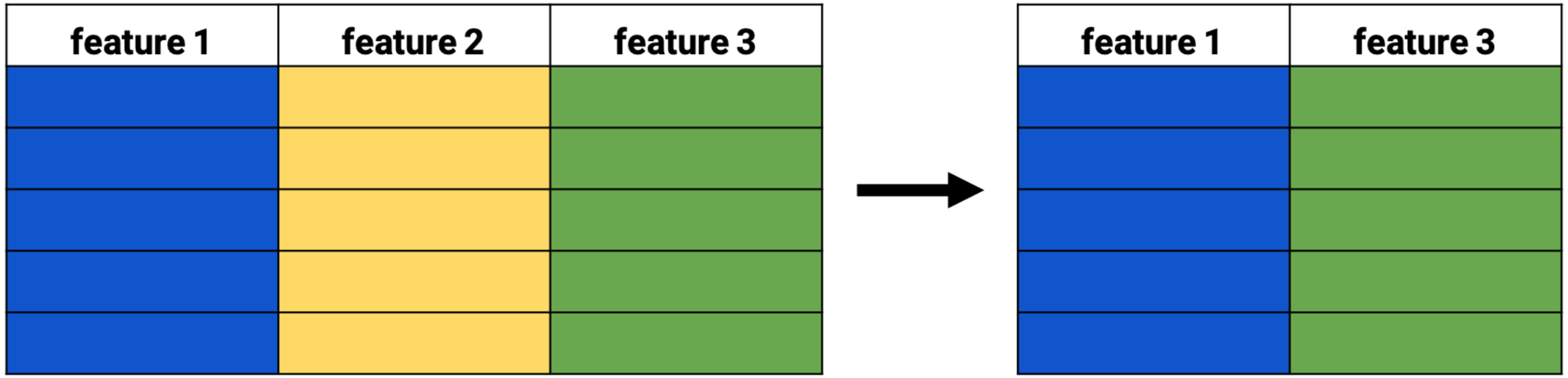
Feature extraction

DIMENSIONALITY REDUCTION IN PYTHON

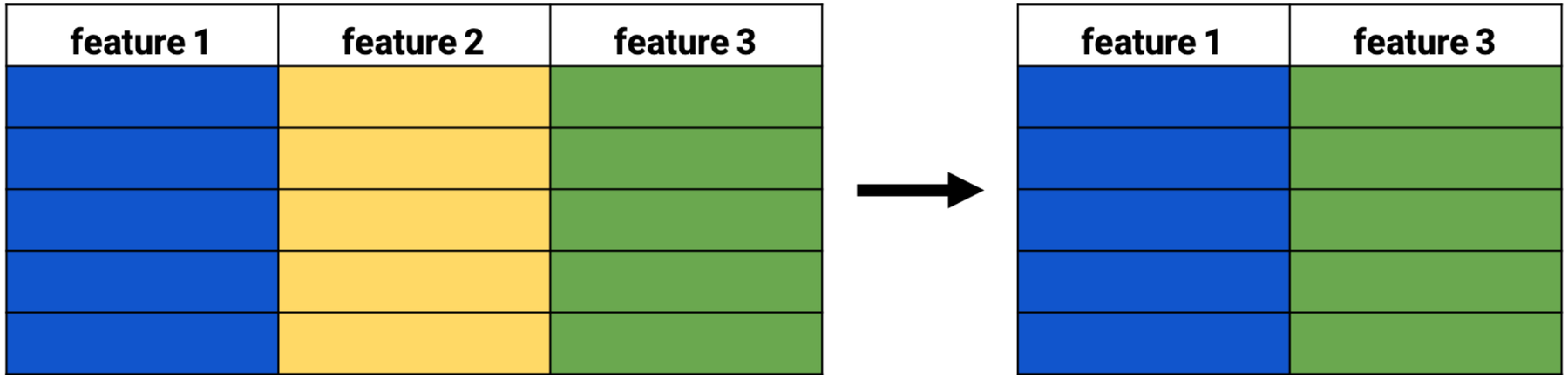
Jeroen Boeye

Machine Learning Engineer, Faktion

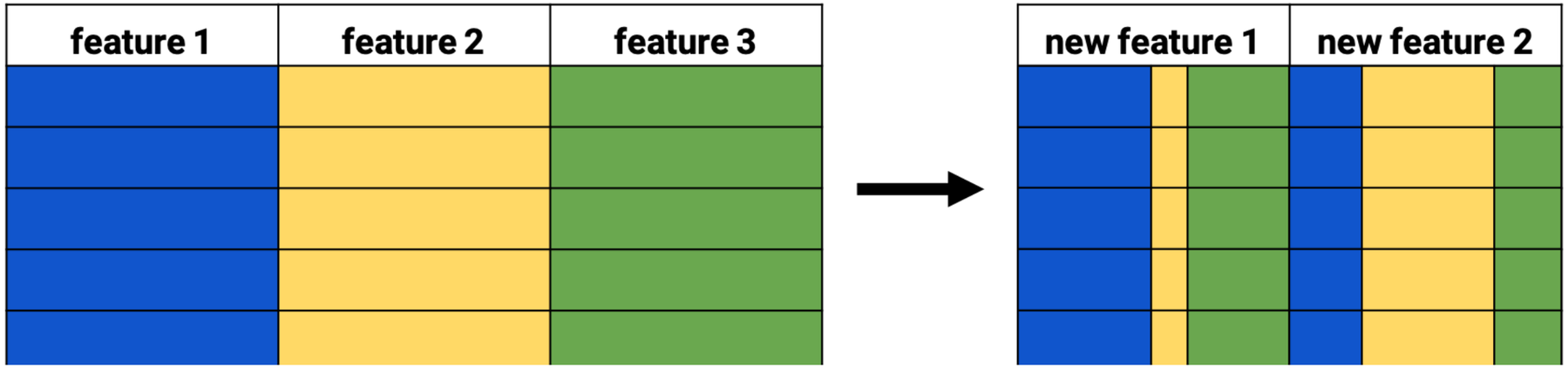
Feature selection



Feature selection



Feature extraction



Feature generation - BMI

```
df_body['BMI'] = df_body['Weight kg'] /  df_body['Height m'] ** 2
```

Feature generation - BMI

```
df_body['BMI'] = df_body['Weight kg'] /  df_body['Height m'] ** 2
```

Weight kg	Height m	BMI
81.5	1.776	25.84
72.6	1.702	25.06
92.9	1.735	30.86

Feature generation - BMI

```
df_body.drop(['Weight kg', 'Height m'], axis=1)
```

BMI
25.84
25.06
30.86

Feature generation - averages

left leg mm	right leg mm
882	885
870	869
901	900

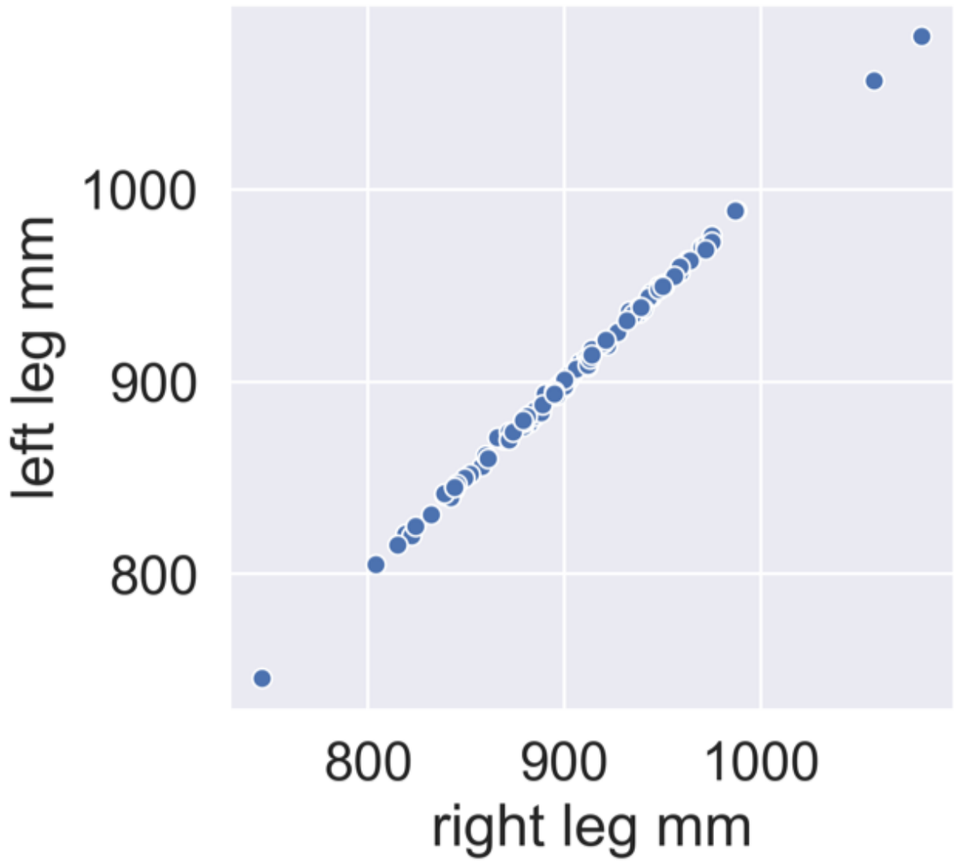
```
leg_df['leg mm'] = leg_df[['right leg mm', 'left leg mm']].mean(axis=1)
```

Feature generation - averages

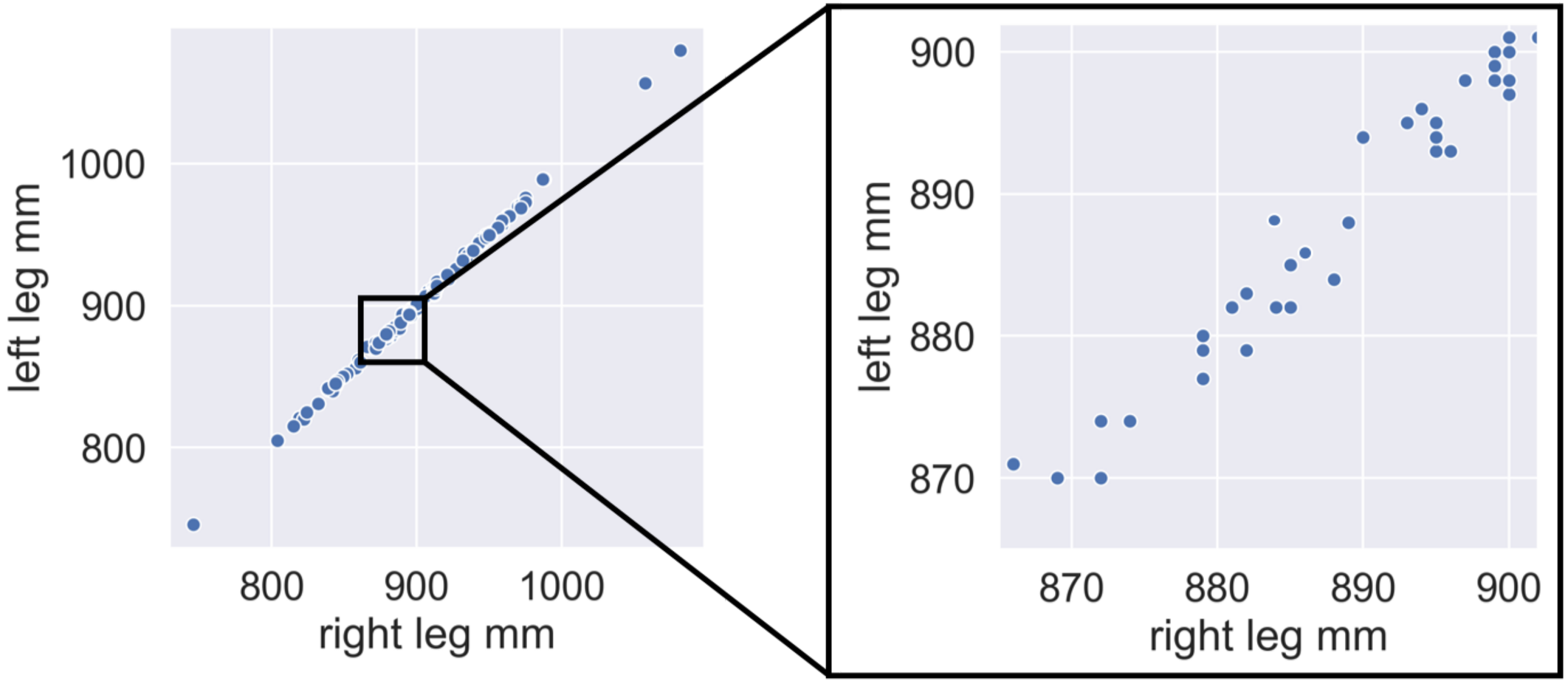
```
leg_df.drop(['right leg mm', 'left leg mm'], axis=1)
```

leg mm
883.5
869.5
900.5

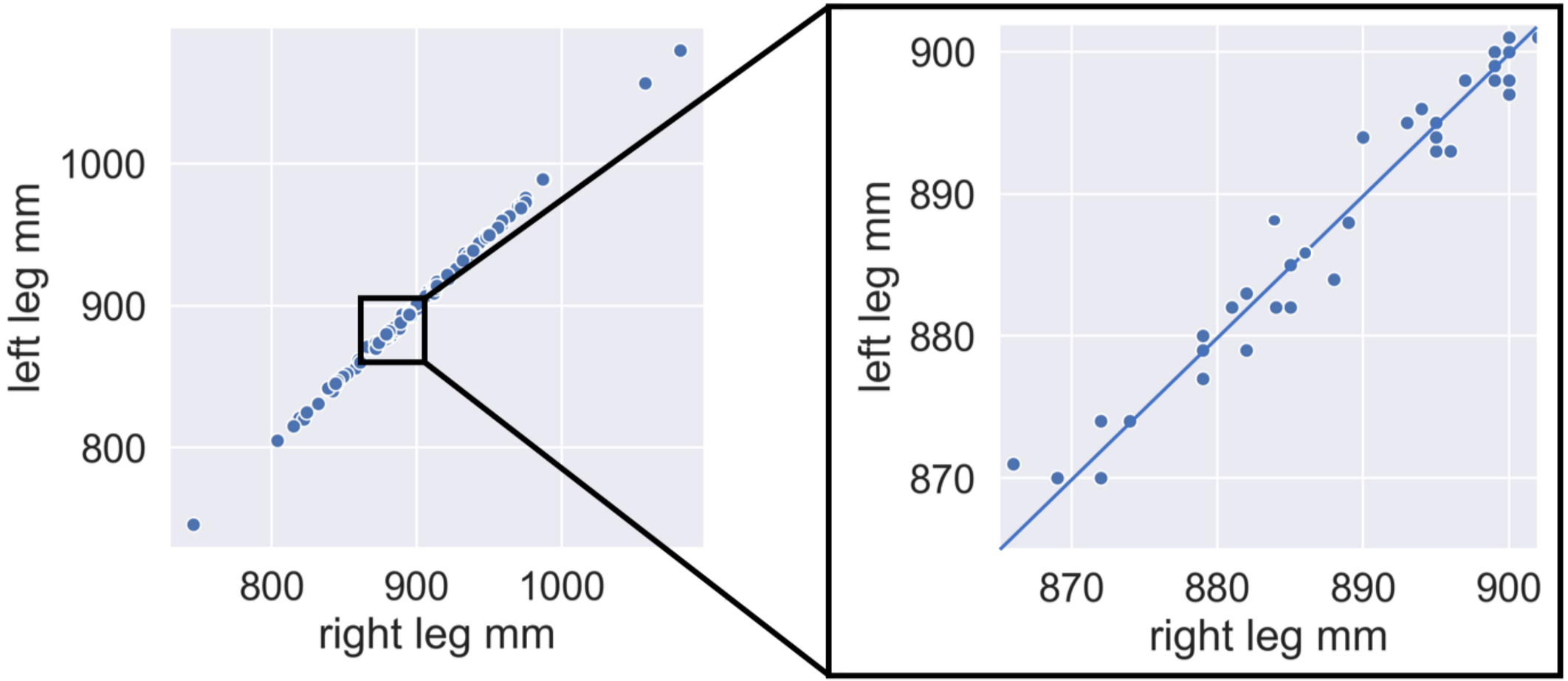
Cost of taking the average



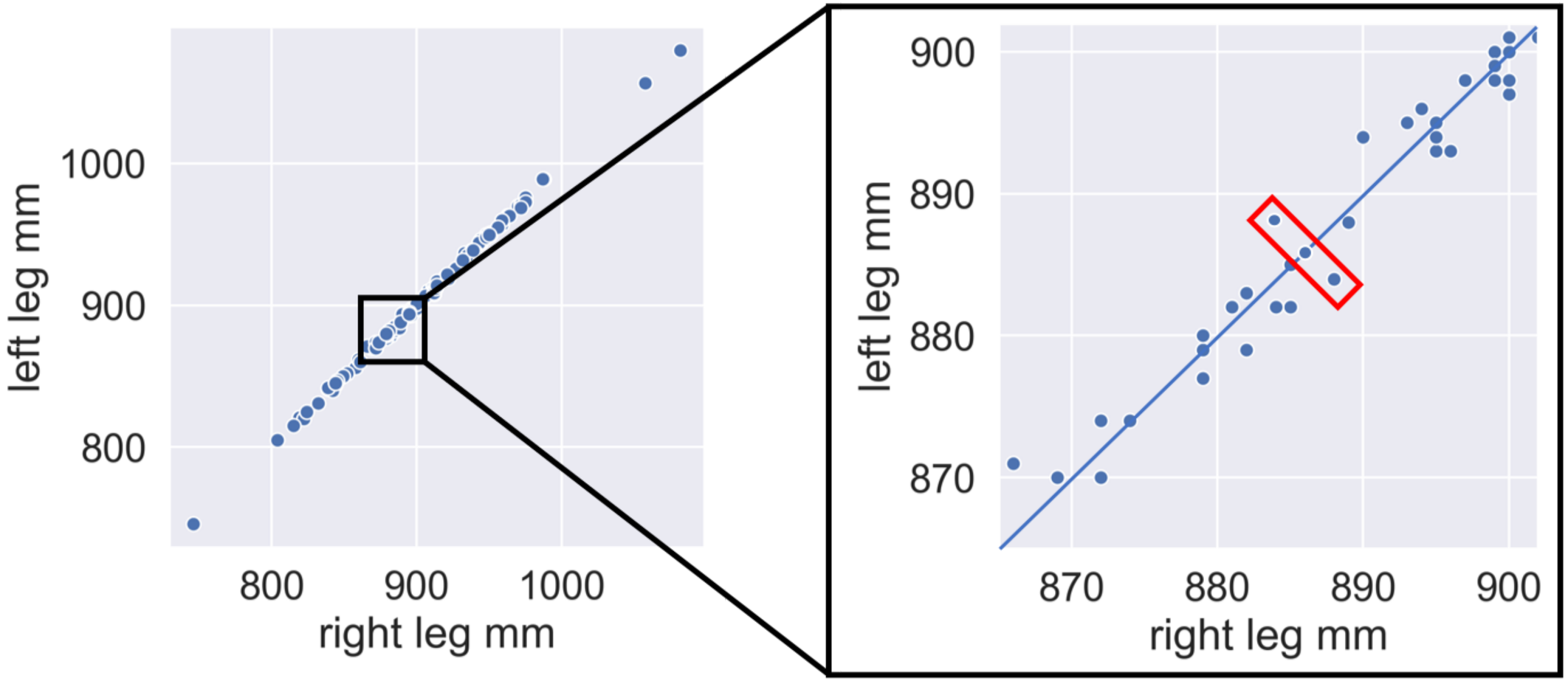
Cost of taking the average



Cost of taking the average

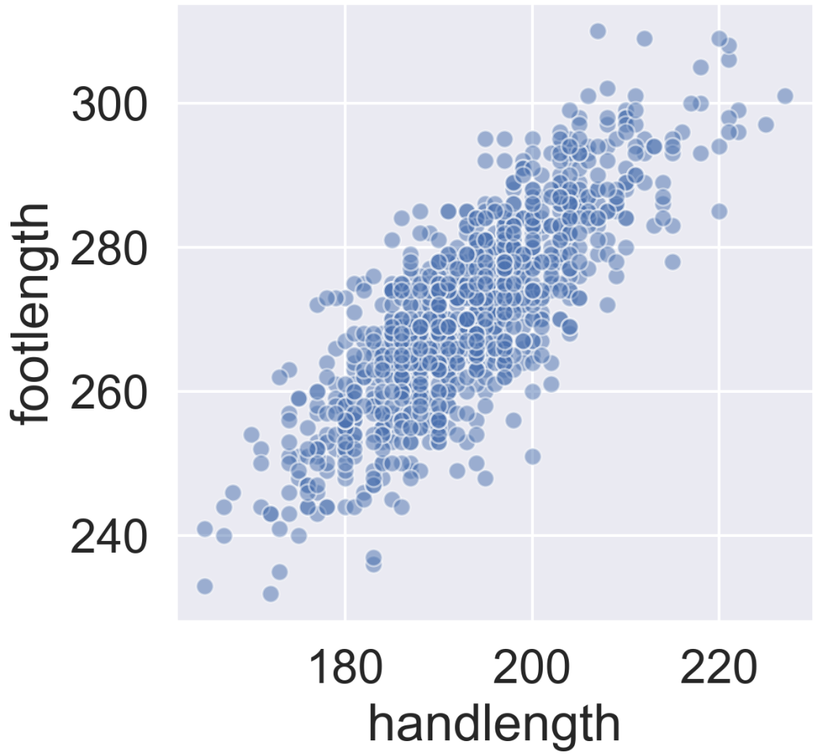


Cost of taking the average



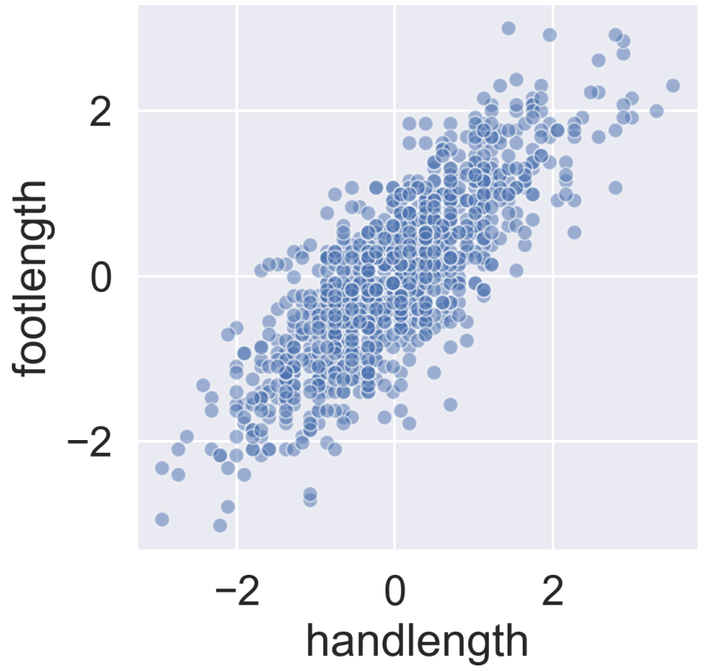
Intro to PCA

```
sns.scatterplot(data=df, x='handlength', y='footlength')
```



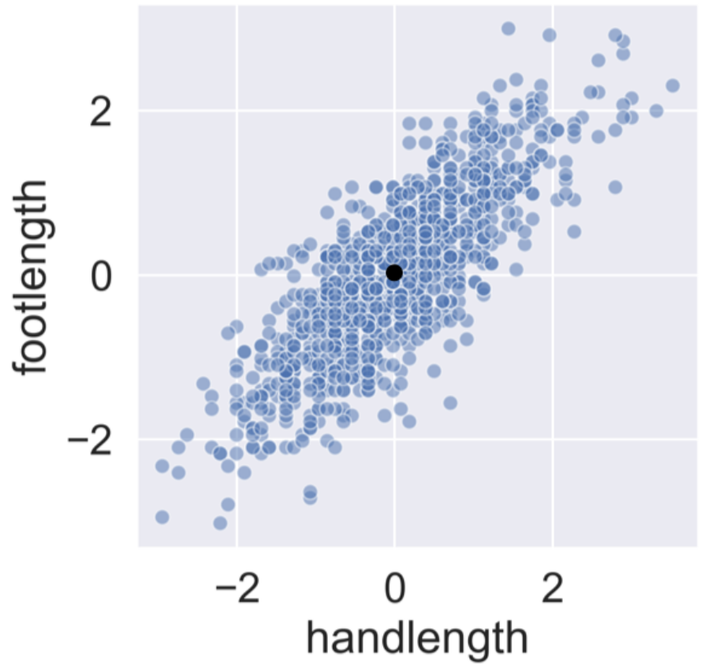
Intro to PCA

```
scaler = StandardScaler()
df_std = pd.DataFrame(scaler.fit_transform(df), columns = df.columns)
```



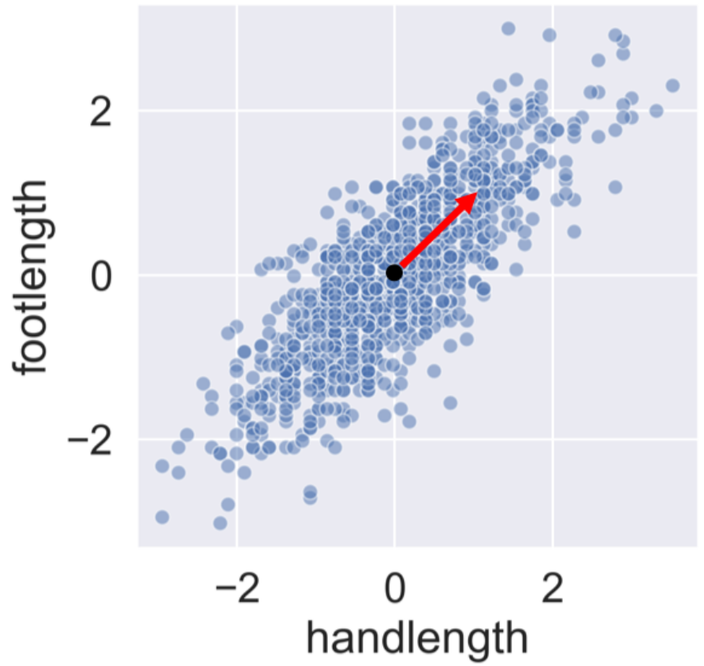
Intro to PCA

```
scaler = StandardScaler()
df_std = pd.DataFrame(scaler.fit_transform(df), columns = df.columns)
```



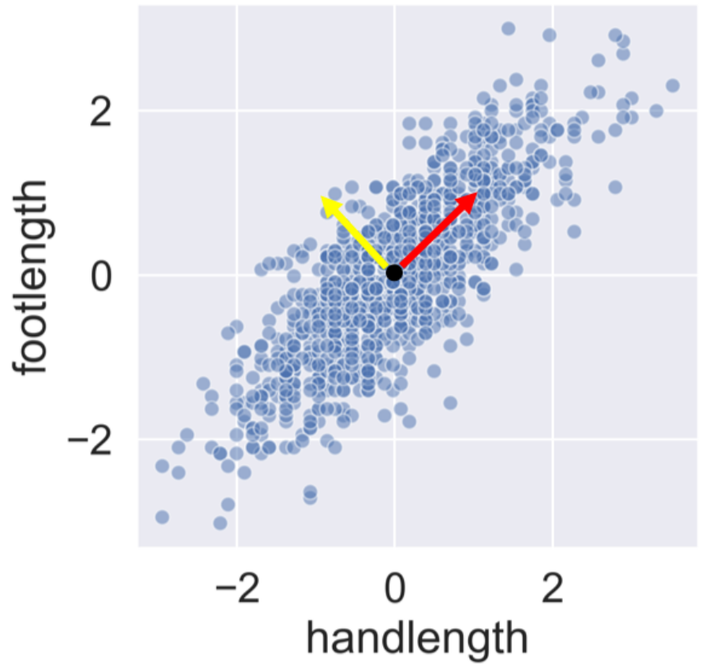
Intro to PCA

```
scaler = StandardScaler()
df_std = pd.DataFrame(scaler.fit_transform(df), columns = df.columns)
```



Intro to PCA

```
scaler = StandardScaler()
df_std = pd.DataFrame(scaler.fit_transform(df), columns = df.columns)
```



Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

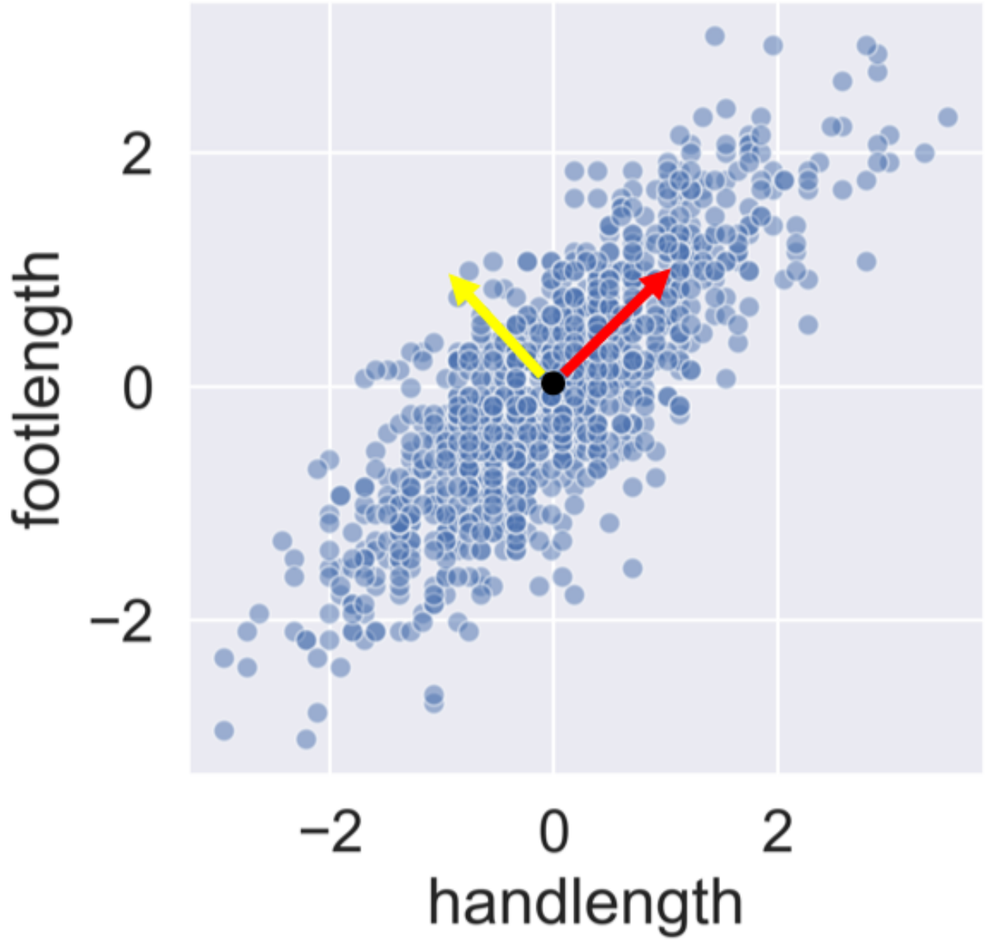
Principal component analysis

DIMENSIONALITY REDUCTION IN PYTHON

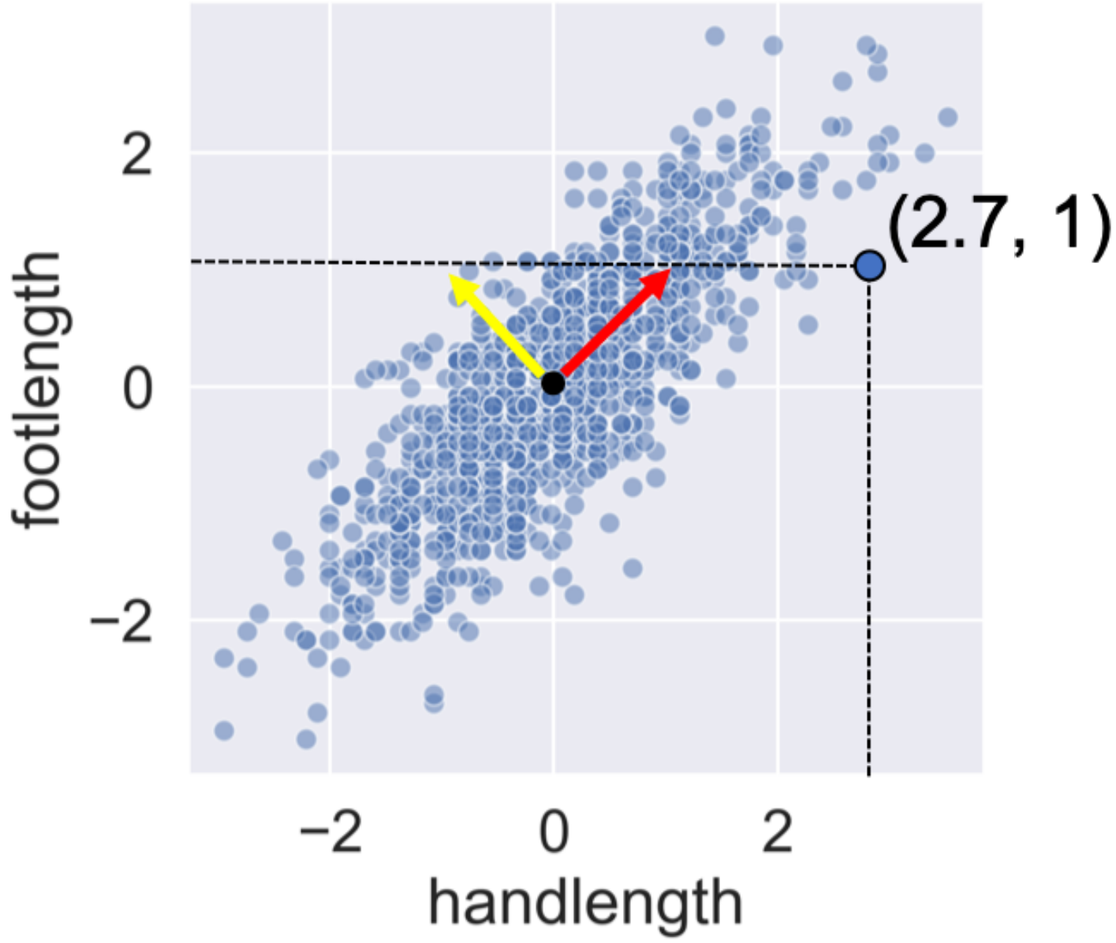
Jeroen Boeye

Machine Learning Engineer, Faktion

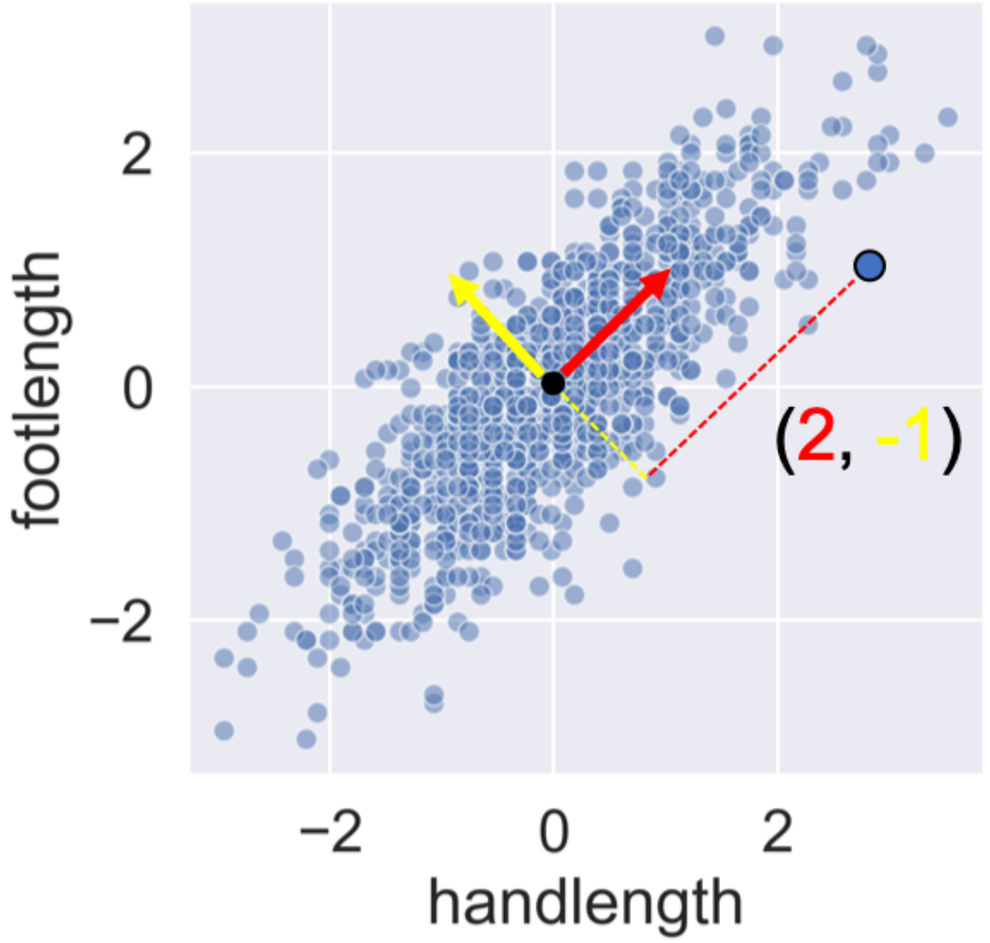
PCA concept



PCA concept



PCA concept

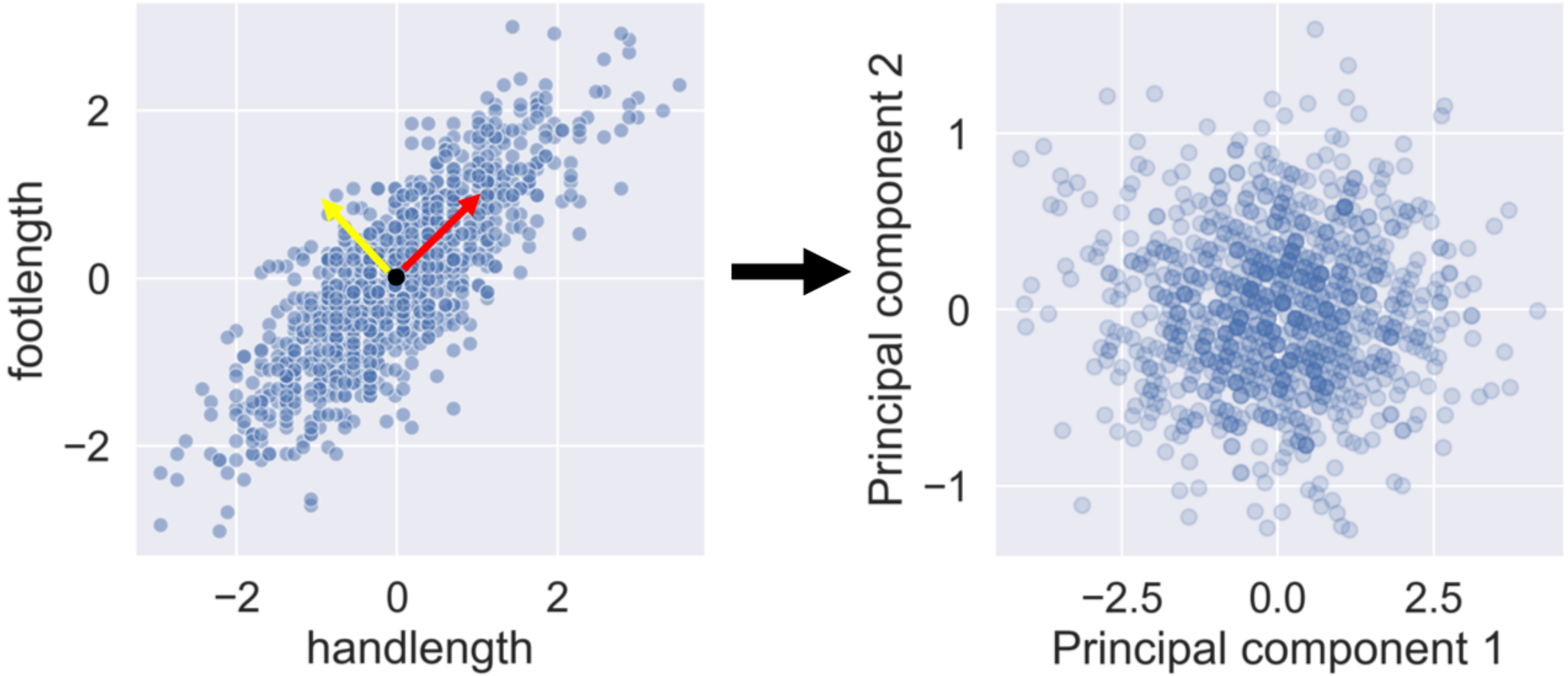


Calculating the principal components

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
std_df = scaler.fit_transform(df)
from sklearn.decomposition import PCA

pca = PCA()
print(pca.fit_transform(std_df))
```

```
[[-0.08320426 -0.12242952]
 [ 0.31478004  0.57048158]
 ...
 [-0.5609523   0.13713944]
 [-0.0448304  -0.37898246]]
```

PCA removes correlation

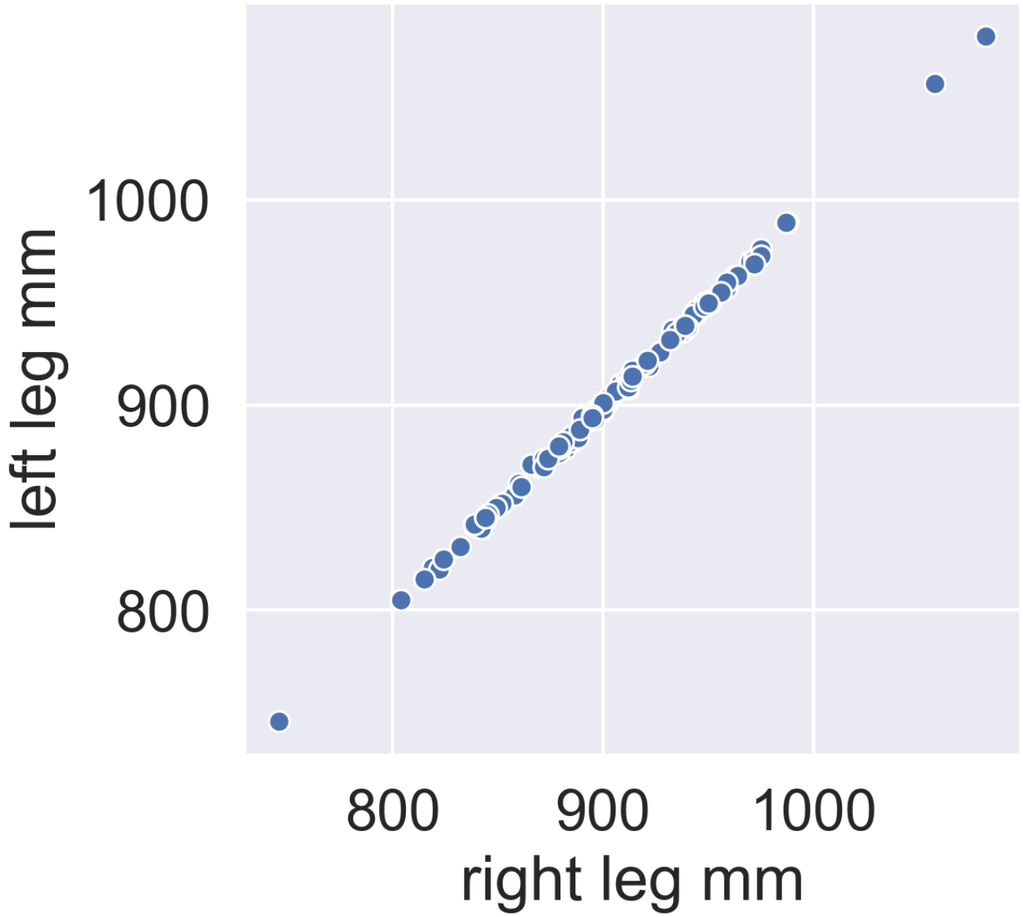


Principal component explained variance ratio

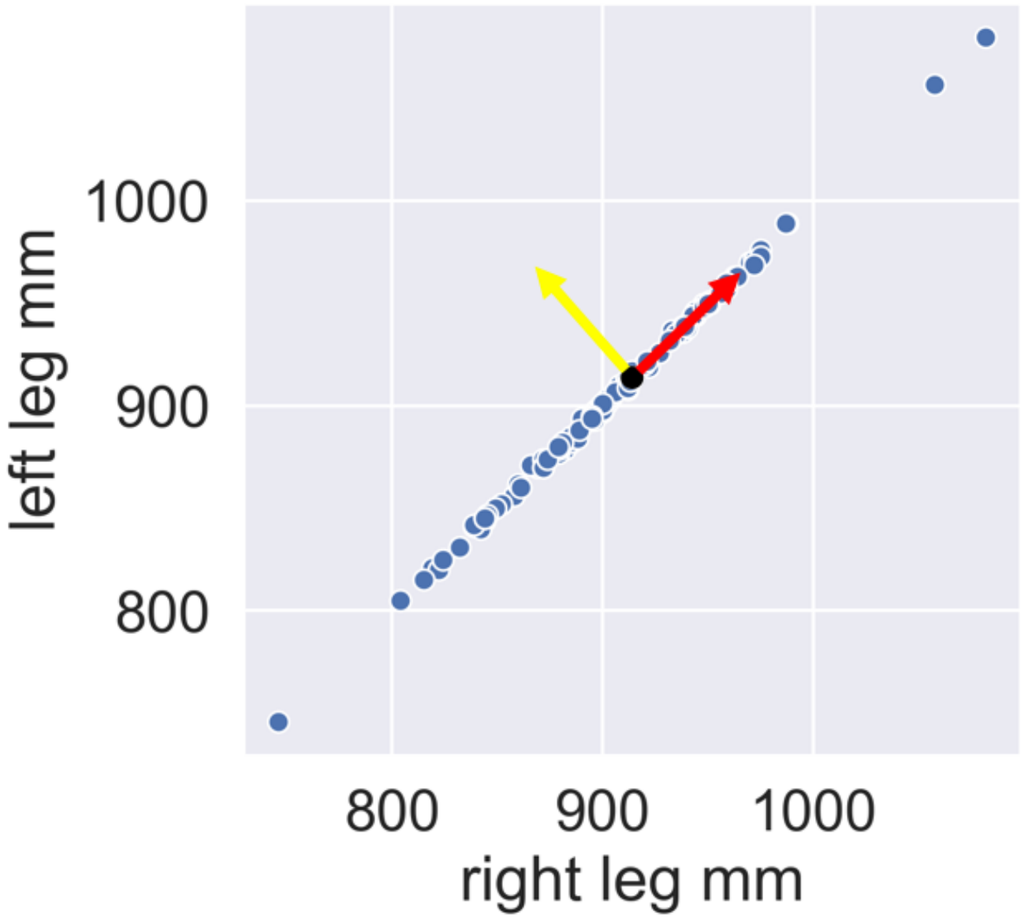
```
from sklearn.decomposition import PCA

pca = PCA()

pca.fit(std_df)

print(pca.explained_variance_ratio_)
```

```
array([0.90, 0.10])
```

PCA for dimensionality reduction



PCA for dimensionality reduction




```
print(pca.explained_variance_ratio_)
```

```
array([0.9997, 0.0003])
```

PCA for dimensionality reduction

```
pca = PCA()

pca.fit(ansur_std_df)

print(pca.explained_variance_ratio_)
```

```
array([0.44, 0.18, 0.04, 0.03, 0.02, 0.02, 0.02, 0.01, 0.01, 0.01, 0.01,
       0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,
       0.01, 0.01, 0.01, 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ,
       ...
       0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ,
       0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ,
       0.  , 0.  , 0.  , 0.  , 0.  , 0.  ])
```

PCA for dimensionality reduction

```
pca = PCA()

pca.fit(ansur_std_df)

print(pca.explained_variance_ratio_.cumsum())
```

```
array([0.44, 0.62, 0.66, 0.69, 0.72, 0.74, 0.76, 0.77, 0.79, 0.8 , 0.81,
       0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.87, 0.88, 0.89, 0.89, 0.9 ,
       0.9 , 0.91, 0.92, 0.92, 0.92, 0.93, 0.93, 0.94, 0.94, 0.94, 0.95,
       ...
       0.99, 0.99, 0.99, 0.99, 0.99, 1.  , 1.  , 1.  , 1.  , 1.  , 1.  ,
       1.  , 1.  , 1.  , 1.  , 1.  , 1.  , 1.  , 1.  , 1.  , 1.  , 1.  ,
       1.  , 1.  , 1.  , 1.  , 1.  , 1.  ])
```

Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

PCA applications

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye

Machine Learning Engineer, Faktion

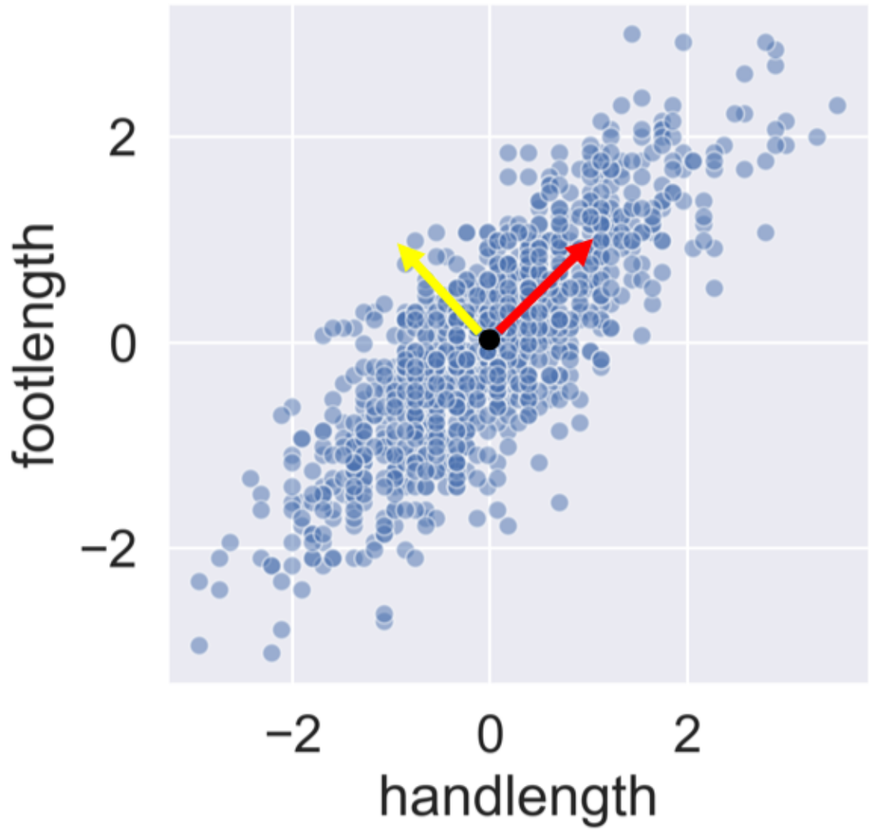
Understanding the components

```
print(pca.components_)
```

```
array([[  0.71, 0.71],
       [ -0.71, 0.71]])
```

PC 1 = 0.71 x Hand length + 0.71 x Foot length

PC 2 = -0.71 x Hand length + 0.71 x Foot length



PCA for data exploration

PCA in a pipeline

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline

pipe = Pipeline([
        ('scaler', StandardScaler()),
        ('reducer', PCA())])
pc = pipe.fit_transform(ansur_df)

print(pc[:,:2])
```

```
array([[-3.46114925,  1.5785215 ],
       [ 0.90860615,  2.02379935],
       ...,
       [10.7569818 , -1.40222755],
       [ 7.64802025,  1.07406209]])
```

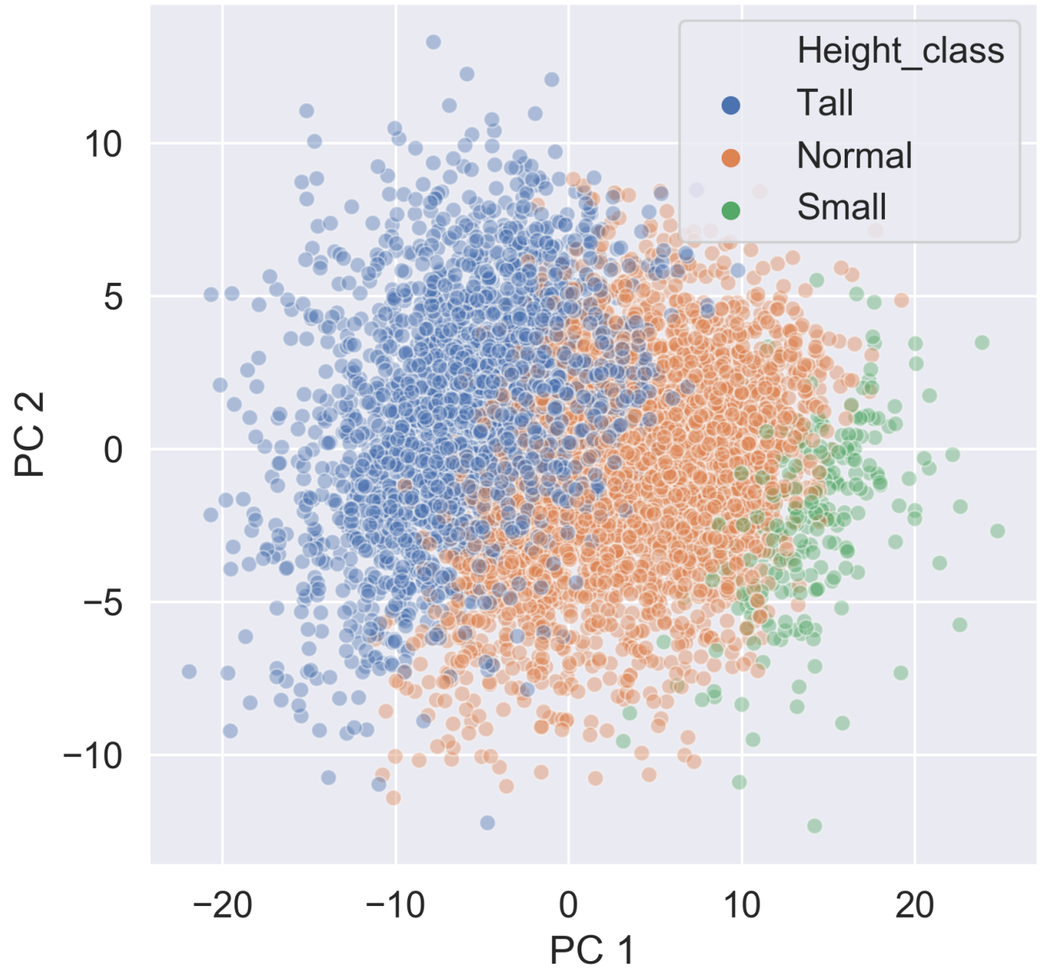
Checking the effect of categorical features

```
print(ansur_categories.head())
```

Branch	Component	Gender	BMI_class	Height_class
Combat Arms	Regular Army	Male	Overweight	Tall
Combat Support	Regular Army	Male	Overweight	Normal
Combat Support	Regular Army	Male	Overweight	Normal
Combat Service Support	Regular Army	Male	Overweight	Normal
Combat Service Support	Regular Army	Male	Overweight	Tall

Checking the effect of categorical features

```
ansur_categories['PC 1'] = pc[:,0]
ansur_categories['PC 2'] = pc[:,1]

sns.scatterplot(data=ansur_categories,
                x='PC 1', y='PC 2',
                hue='Height_class', alpha=0.4)
```

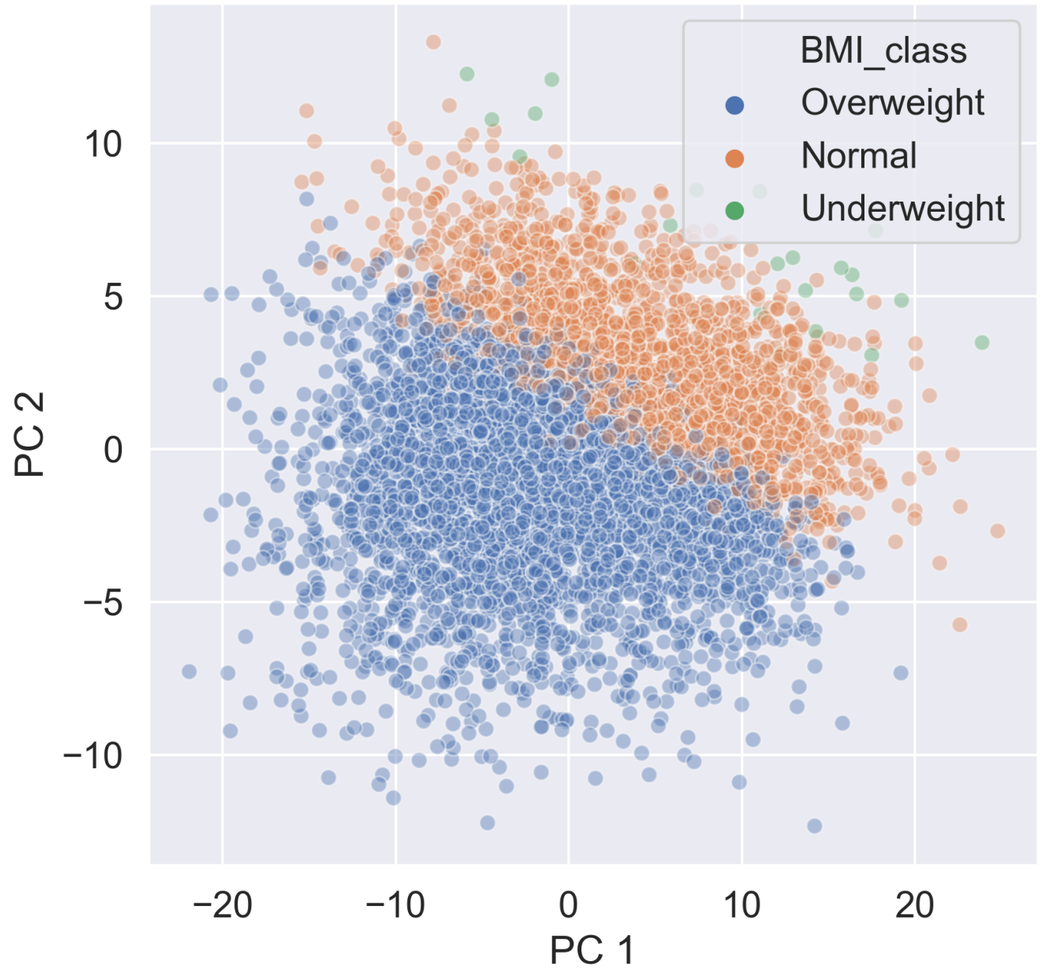


Checking the effect of categorical features

```
sns.scatterplot(data=ansur_categories,
                x='PC 1', y='PC 2',
                hue='Gender', alpha=0.4)
```

Checking the effect of categorical features

```
sns.scatterplot(data=ansur_categories,
                x='PC 1', y='PC 2',
                hue='BMI_class', alpha=0.4
```



PCA in a model pipeline

```
pipe = Pipeline([
        ('scaler', StandardScaler()),
        ('reducer', PCA(n_components=3)),
        ('classifier', RandomForestClassifier())])

pipe.fit(X_train, y_train)

print(pipe.steps[1])
```

```
('reducer',
 PCA(copy=True, iterated_power='auto', n_components=3, random_state=None,
   svd_solver='auto', tol=0.0, whiten=False))
```

PCA in a model pipeline

```
pipe.steps[1][1].explained_variance_ratio_.cumsum()
```

```
array([0.56, 0.69, 0.74])
```

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

```
0.986
```

Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

Principal Component selection

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye

Machine Learning Engineer, Faktion

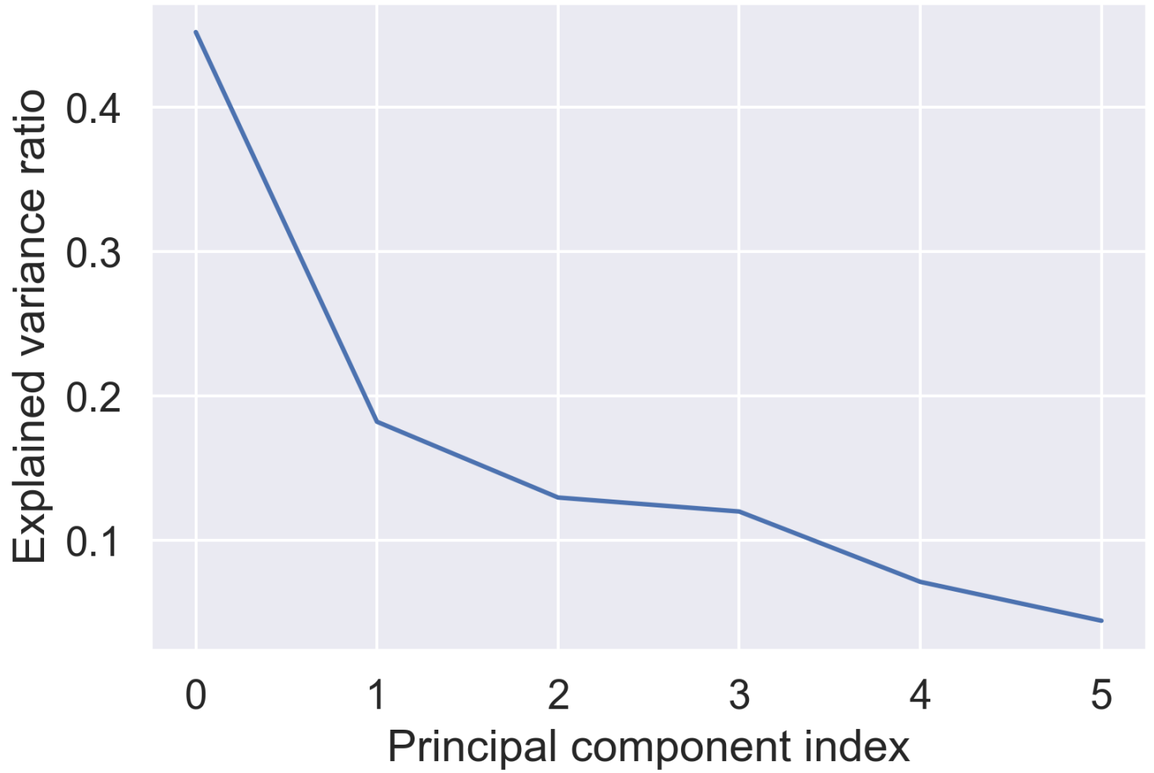
Setting an explained variance treshold

```
pipe = Pipeline([
        ('scaler', StandardScaler()),
        ('reducer', PCA(n_components=0.9))])

# Fit the pipe to the data
pipe.fit(poke_df)


print(len(pipe.steps[1][1].components_))
```

```
5
```

An optimal number of components

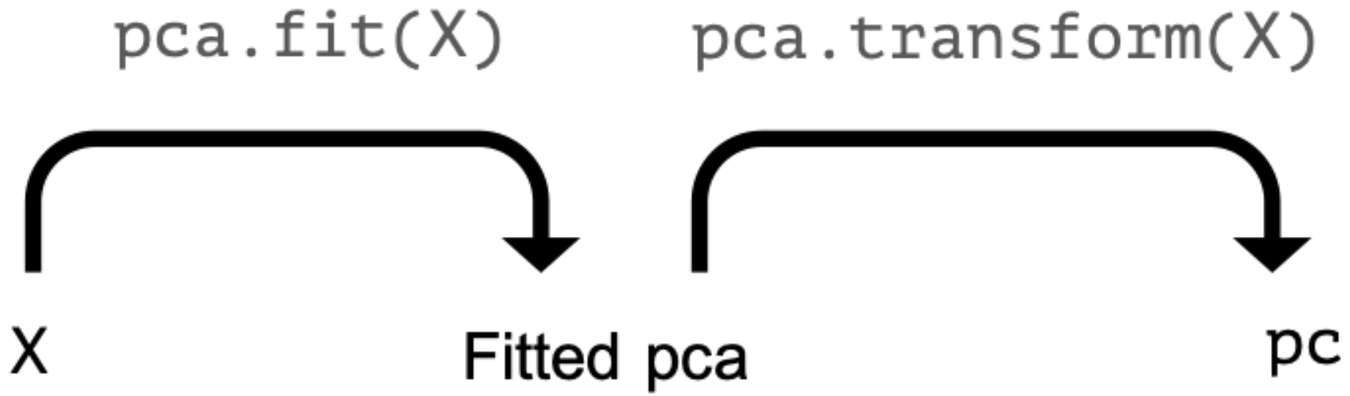
```
pipe.fit(poke_df)

var = pipe.steps[1][1].explained_variance_ratio_

plt.plot(var)

plt.xlabel('Principal component index')
plt.ylabel('Explained variance ratio')
plt.show()
```



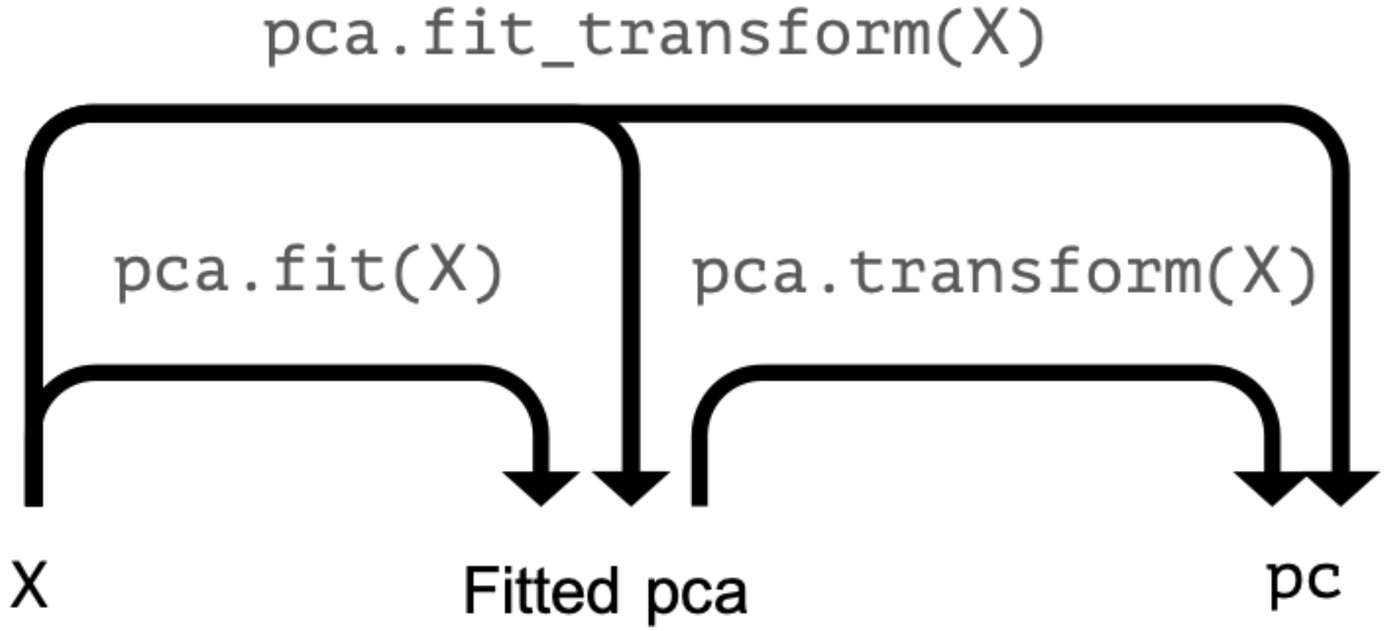
An optimal number of components

```
pipe.fit(poke_df)

var = pipe.steps[1][1].explained_variance_ratio_

plt.plot(var)

plt.xlabel('Principal component index')
plt.ylabel('Explained variance ratio')
plt.show()
```

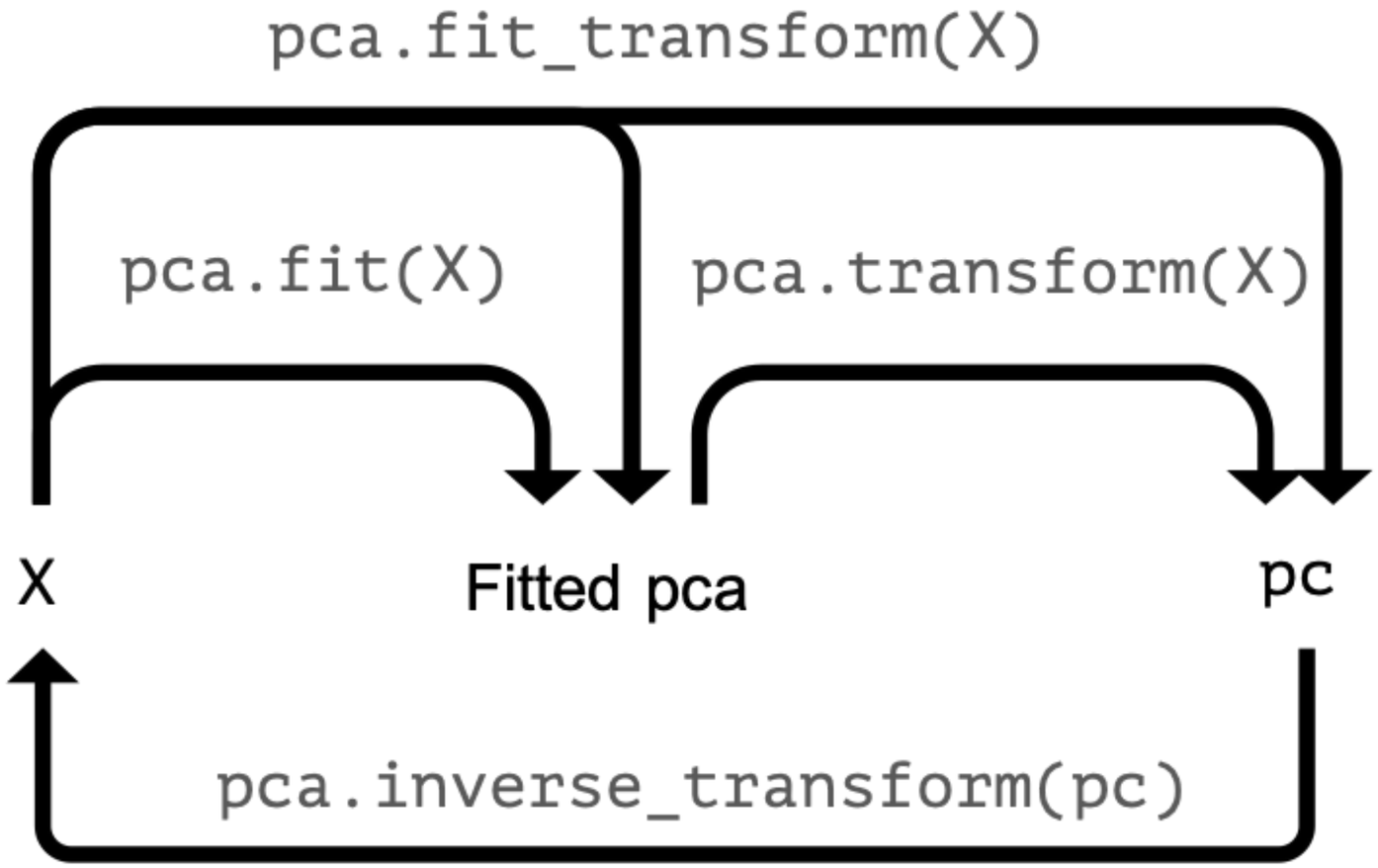
PCA operations



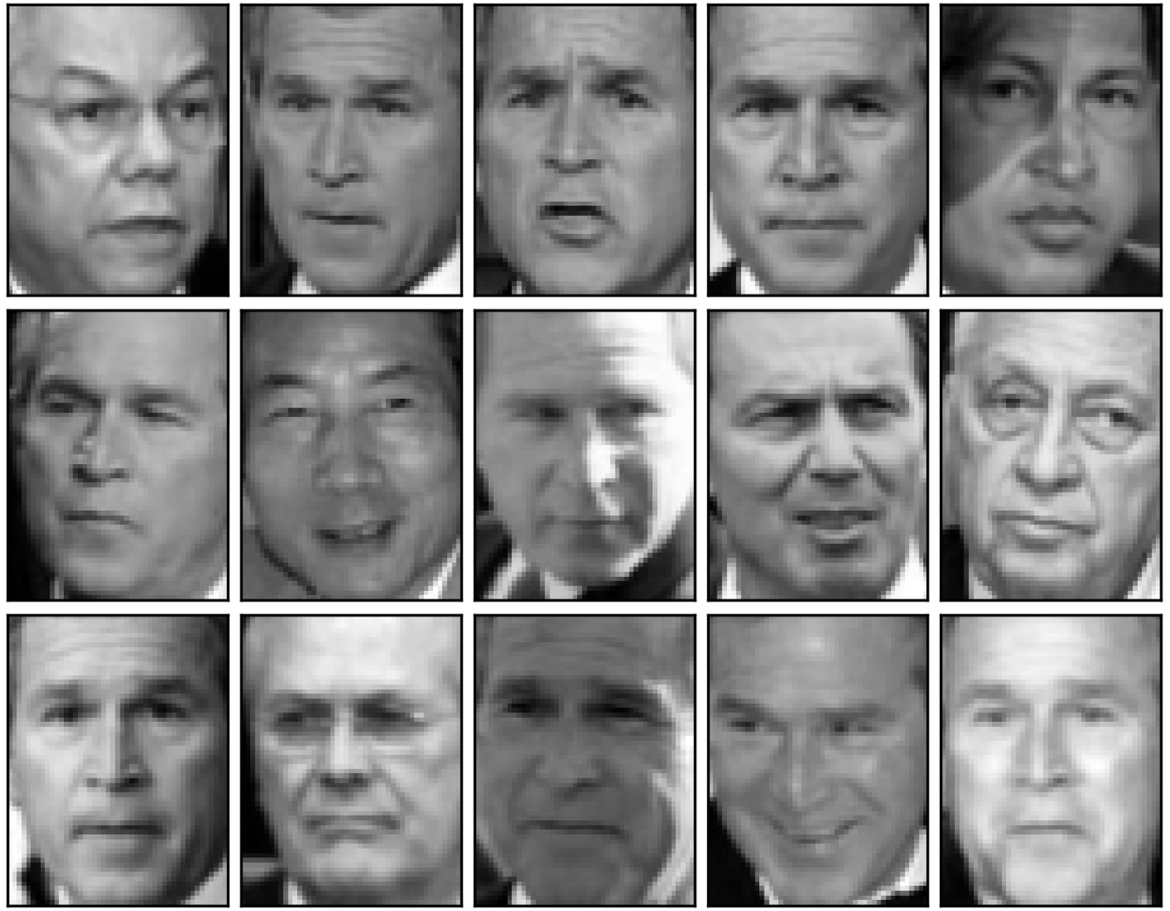
PCA operations



PCA operations



Compressing images



Compressing images

```
print(X_test.shape)
```

```
(15, 2914)
```

62 x 47 pixels = 2914 grayscale values

```
print(X_train.shape)
```

```
(1333, 2914)
```

Compressing images

```
pipe = Pipeline([
        ('scaler', StandardScaler()),
        ('reducer', PCA(n_components=290))])

pipe.fit(X_train)

pc = pipe.fit_transform(X_test)


print(pc.shape)
```

```
(15, 290)
```

Rebuilding images

```
pc = pipe.transform(X_test)

print(pc.shape)
```

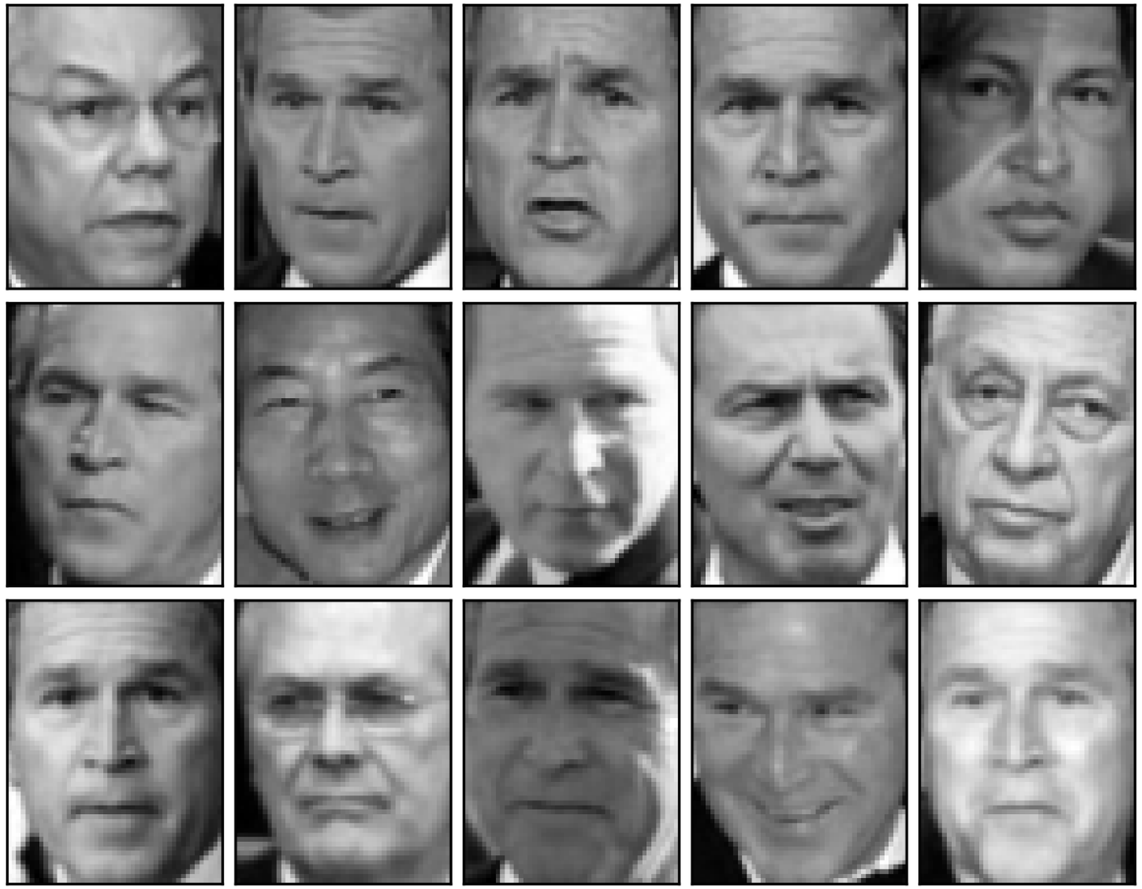
```
(15, 290)
```

```
X_rebuilt = pipe.inverse_transform(pc)

print(X_rebuilt.shape)
```

```
(15, 2914)
```

```
img_plotter(X_rebuilt)
```

Rebuilding images



Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

Congratulations!

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen

Machine Learning Engineer, Faktion

What you've learned

- Why dimensionality reduction is important & when to use it
- Feature selection vs extraction
- High dimensional data exploration with t-SNE & PCA
- Use models to find important features
- Remove unimportant ones

Thank you!

DIMENSIONALITY REDUCTION IN PYTHON