The curse of dimensionality

DIMENSIONALITY REDUCTION IN PYTHON

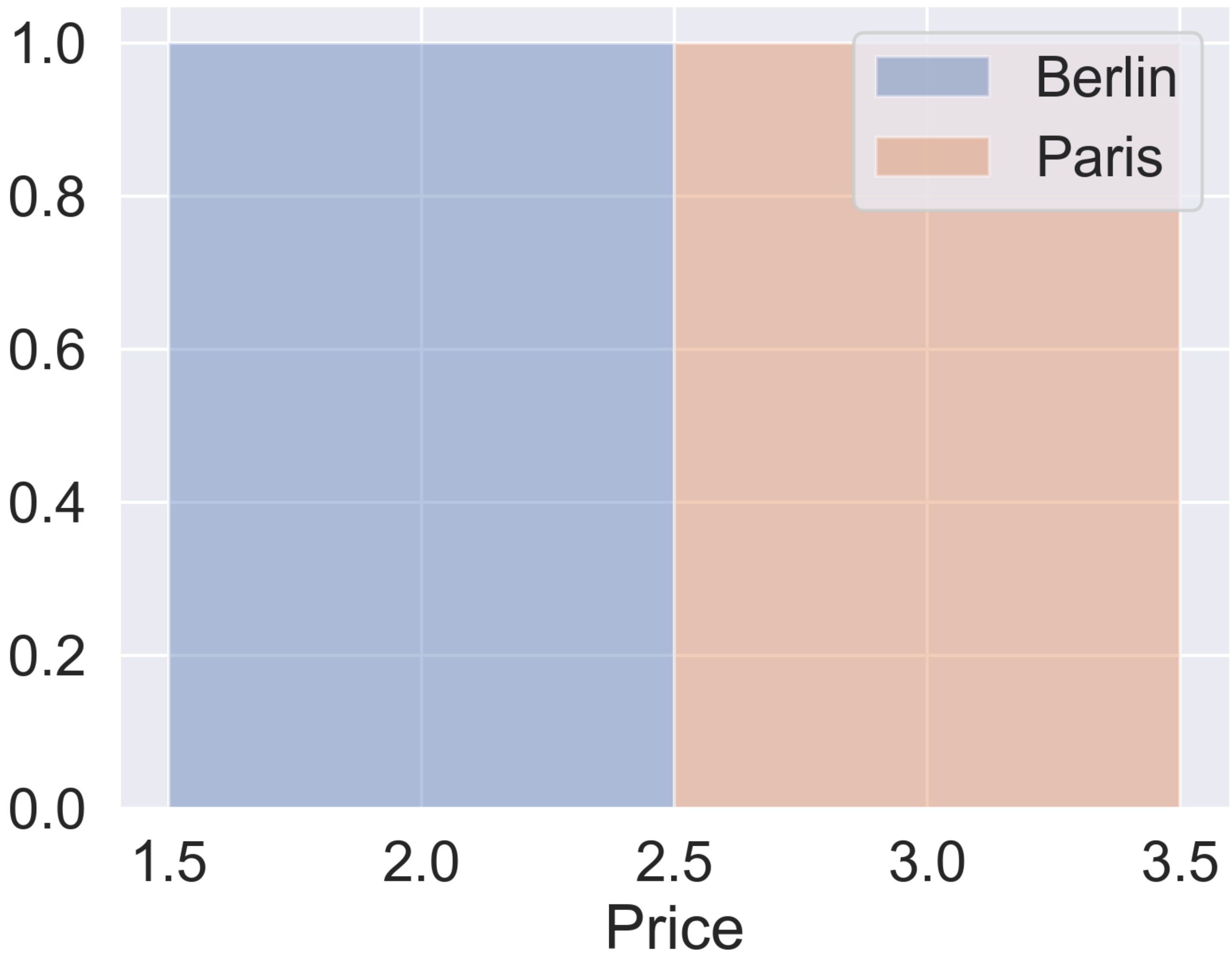
Jeroen Boeye
Head of Machine Learning, Faktion

From observation to pattern

City	Price
Berlin	2
Paris	3

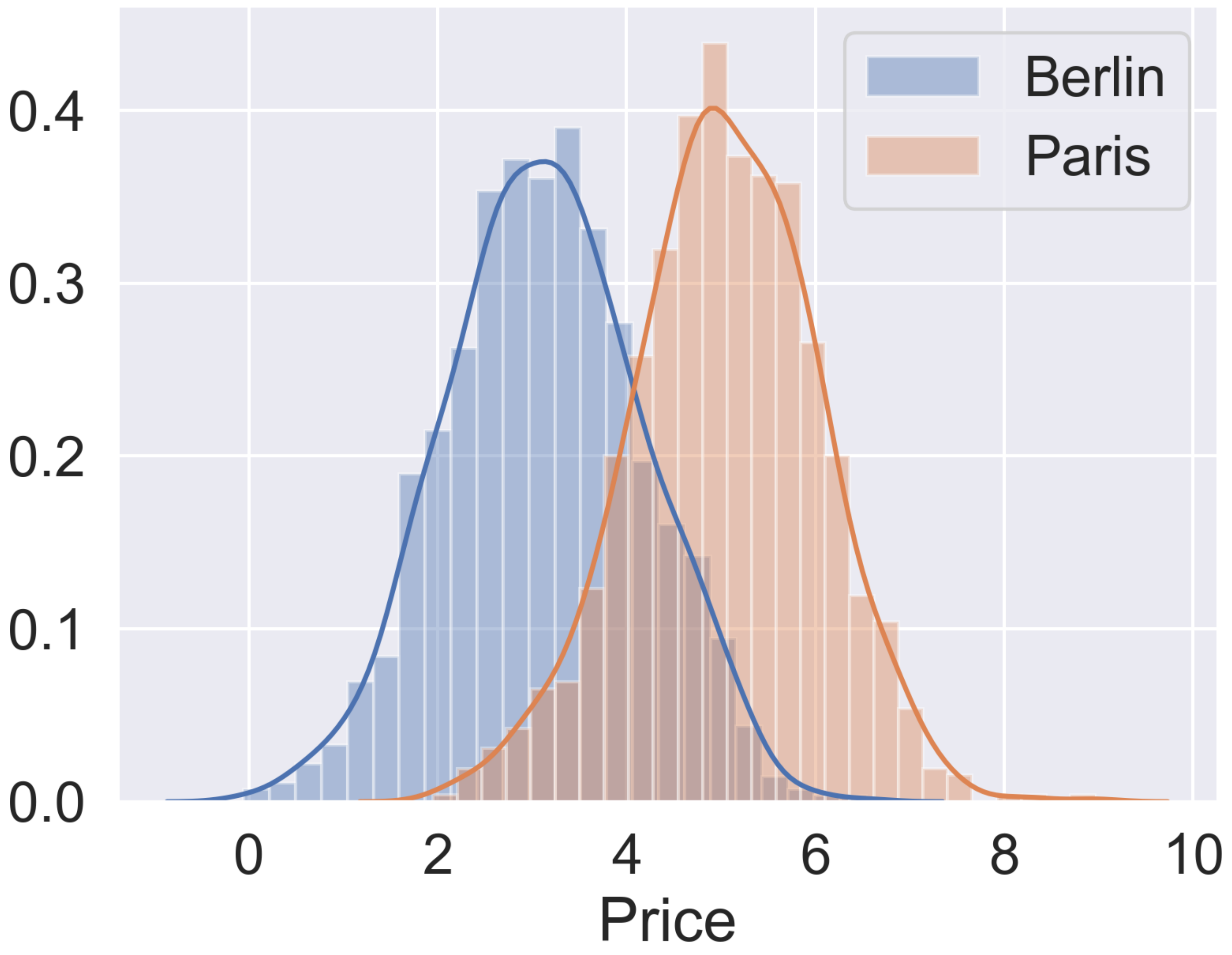
From observation to pattern

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)))
```

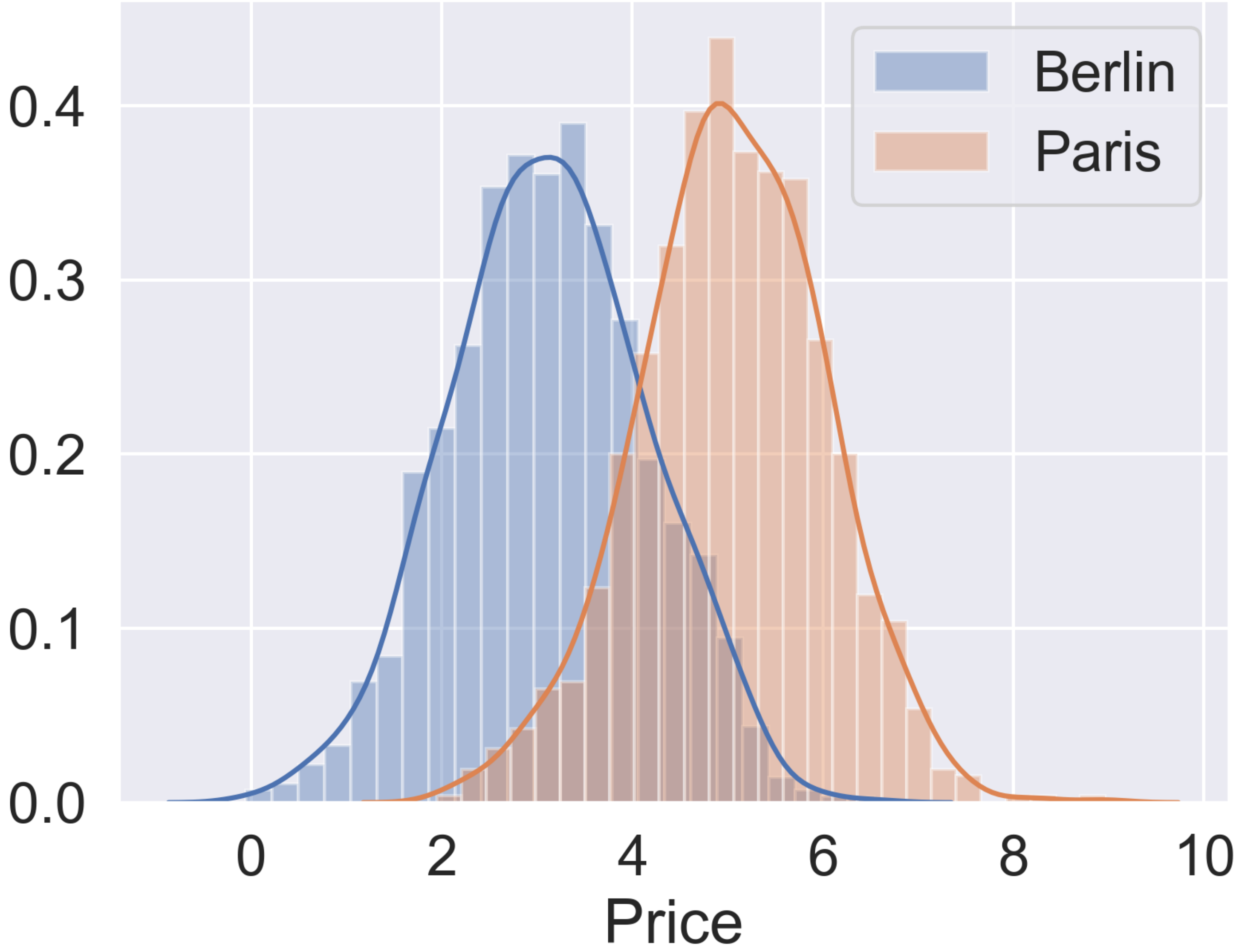
```
0.826
```

```
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
Head of Machine Learning, 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)
```

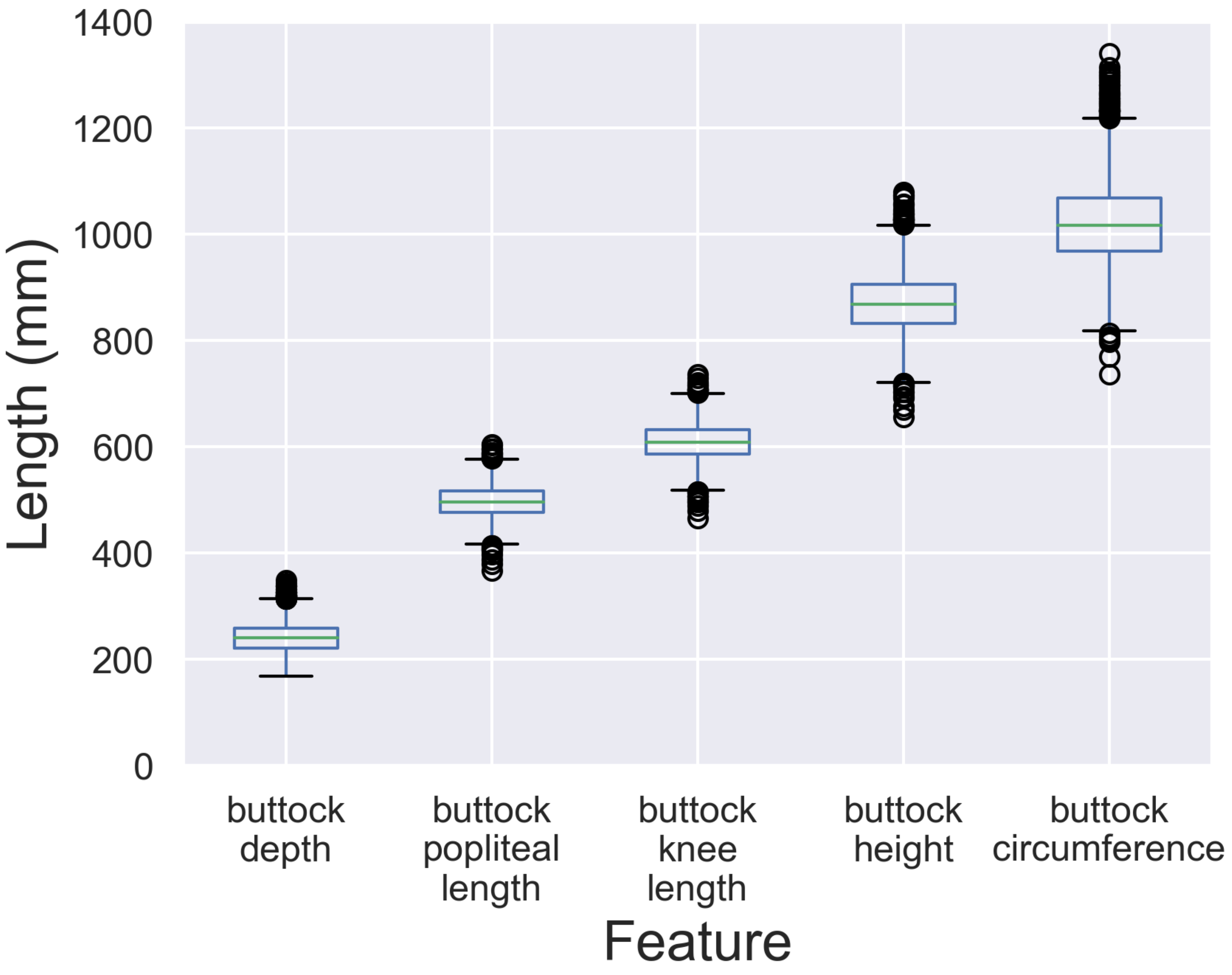
```
(6068, 94)
```

```
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

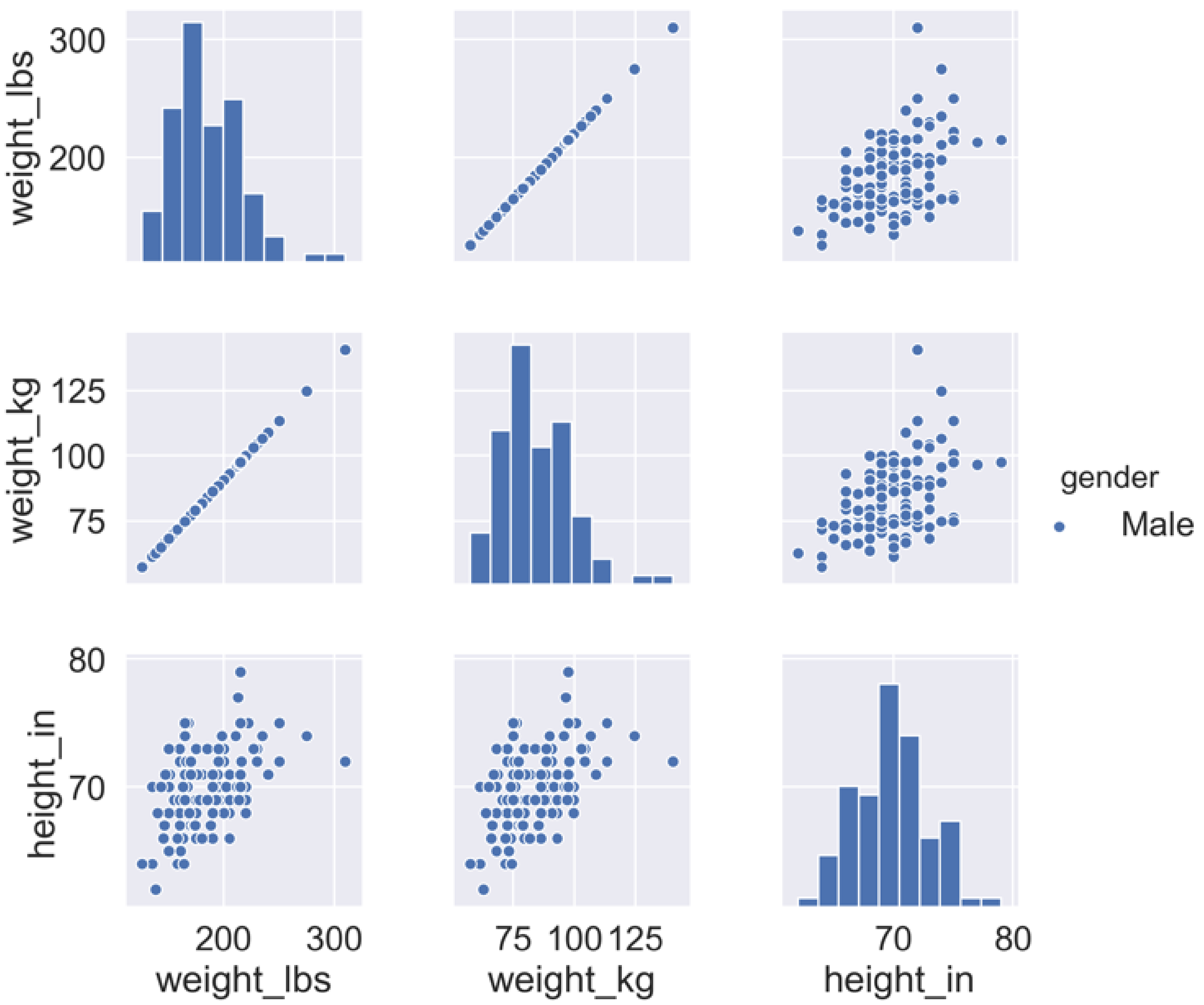
Pairwise correlation

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Head of Machine Learning, Faktion

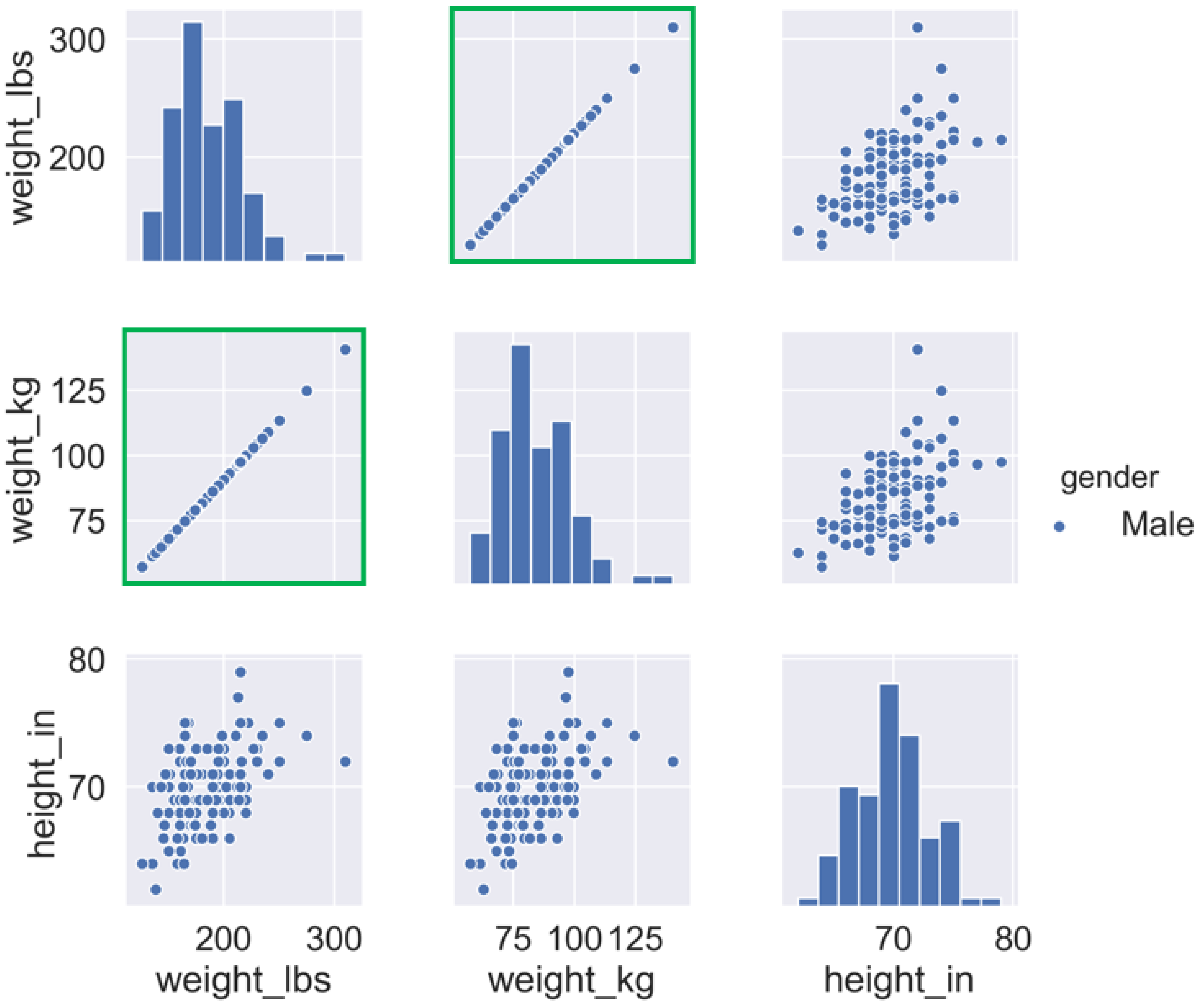
Pairwise correlation

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

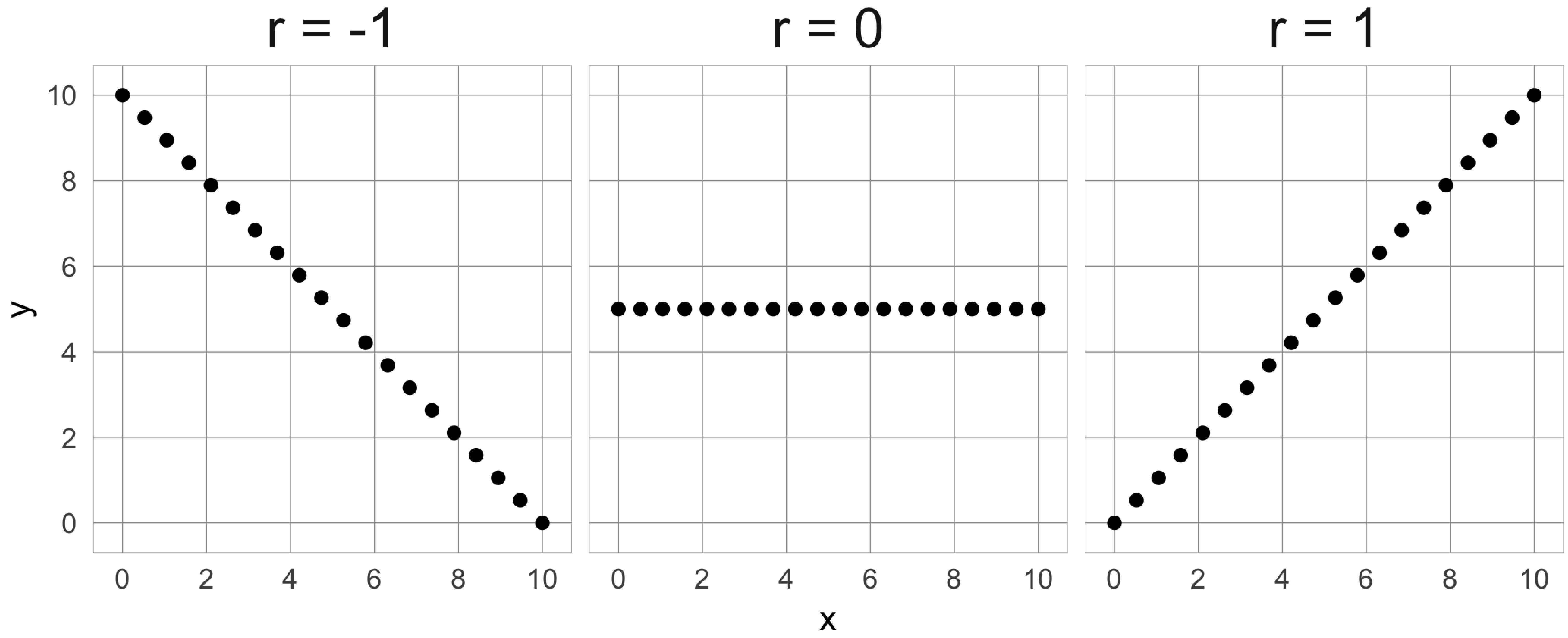


Pairwise correlation

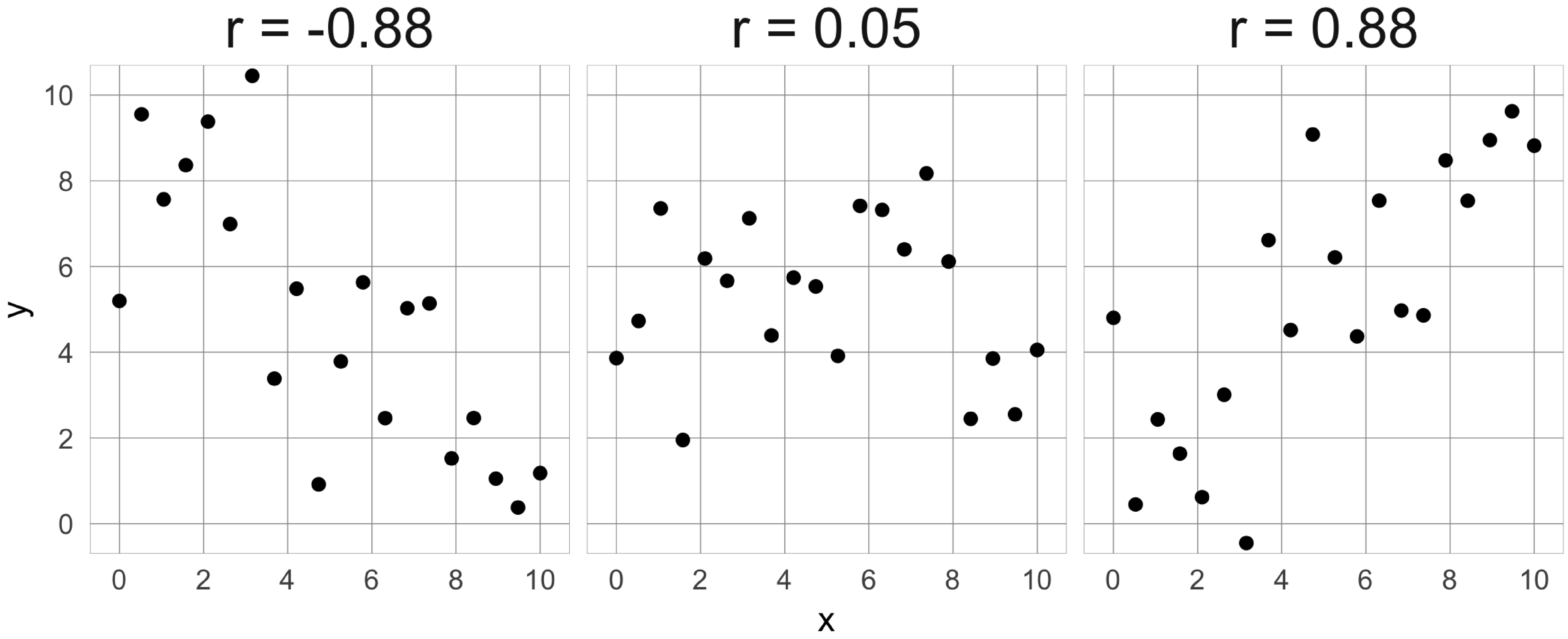
```
sns.pairplot(ansur, hue="gender")
```



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

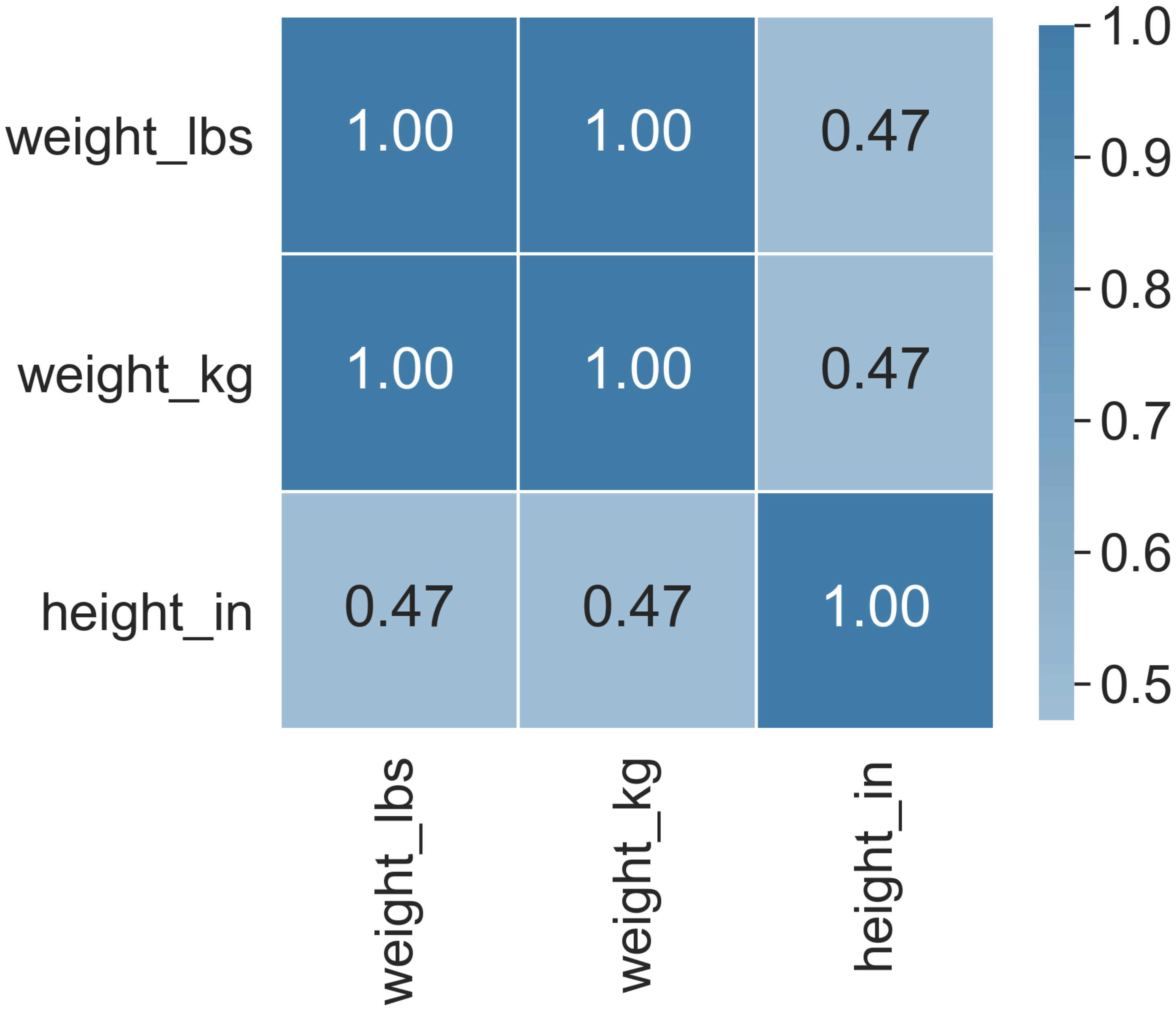
```
weights_df.corr()
```

	weight_lbs	weight_kg	height_in
weight_lbs	$\boxed{1.00}$	1.00	0.47
weight_kg	1.00	$\boxed{1.00}$	0.47
height_in	0.47	0.47	$\boxed{1.00}$

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

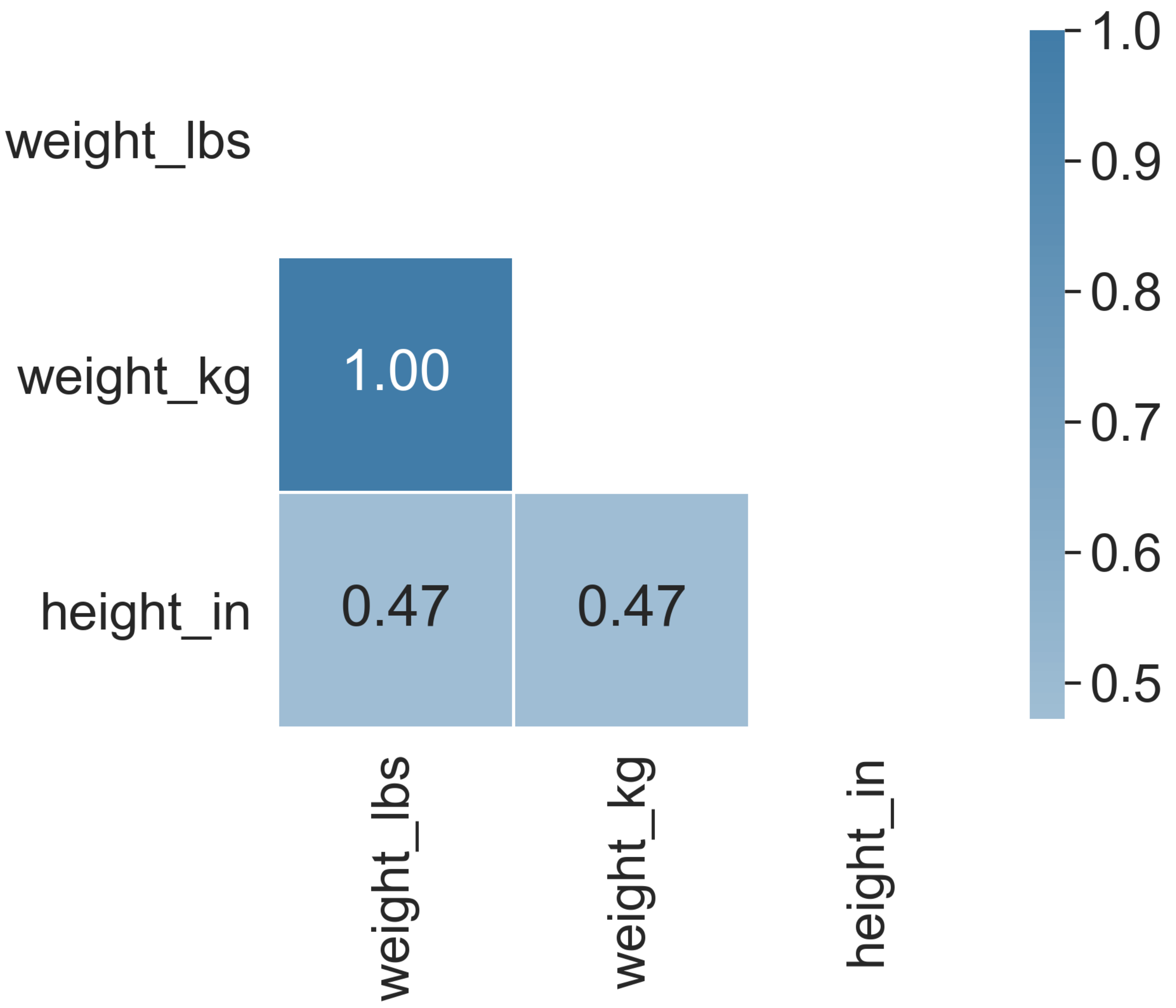
```
corr = weights_df.corr()


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

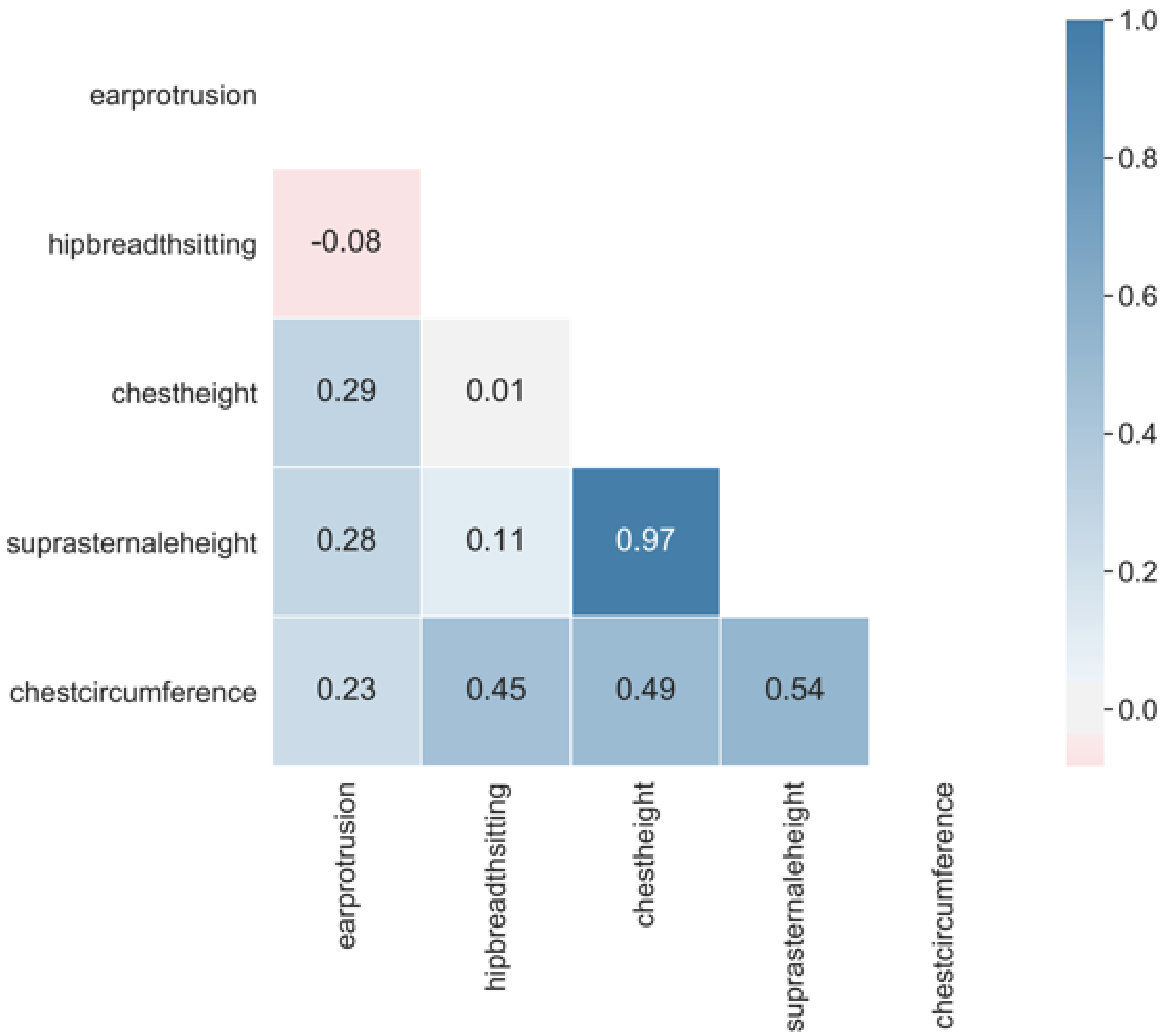
```
array([[ True,  True,  True],
       [False,  True,  True],
       [False, False,  True]])
```

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!

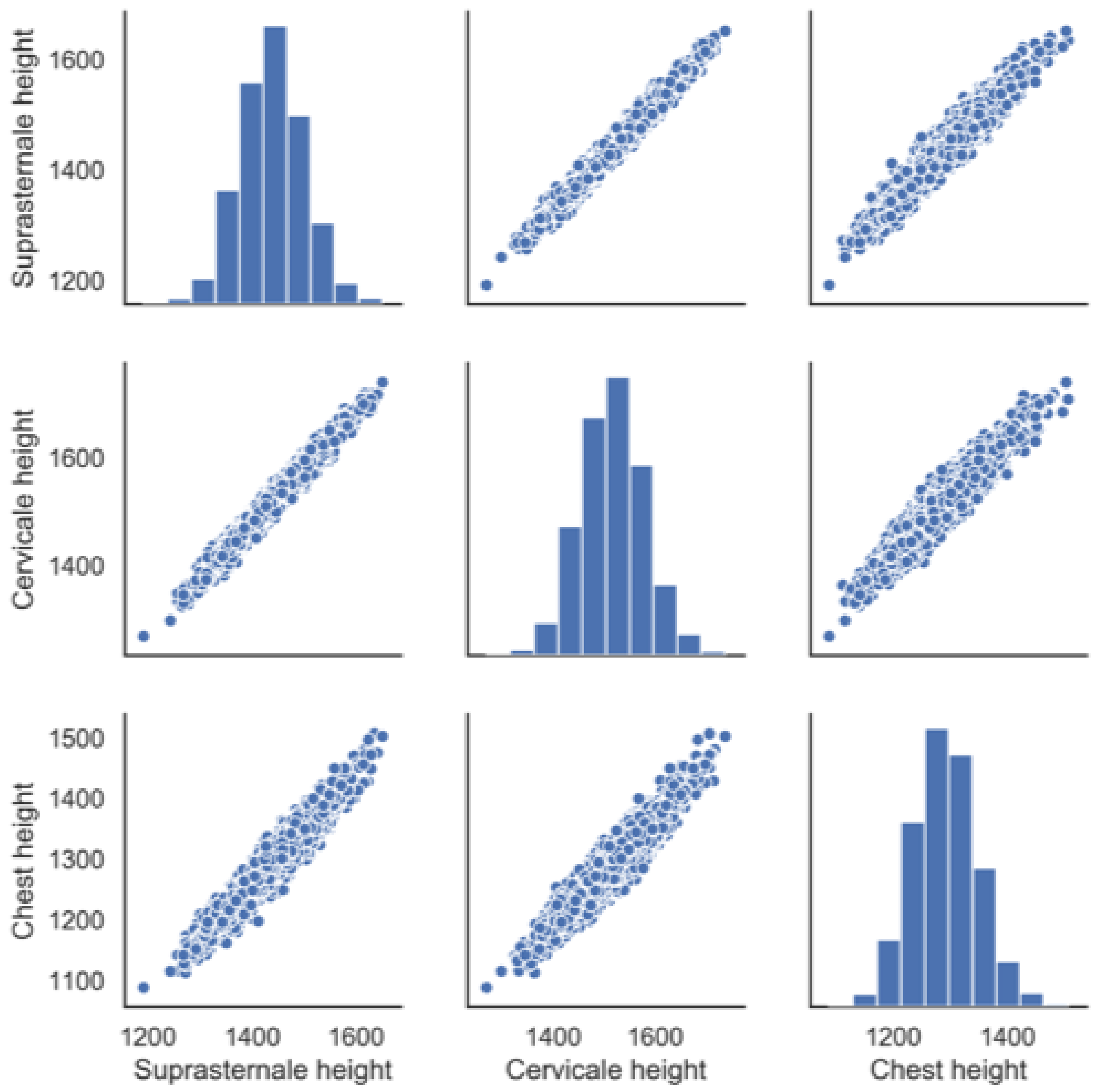
DIMENSIONALITY REDUCTION IN PYTHON

Removing highly correlated features

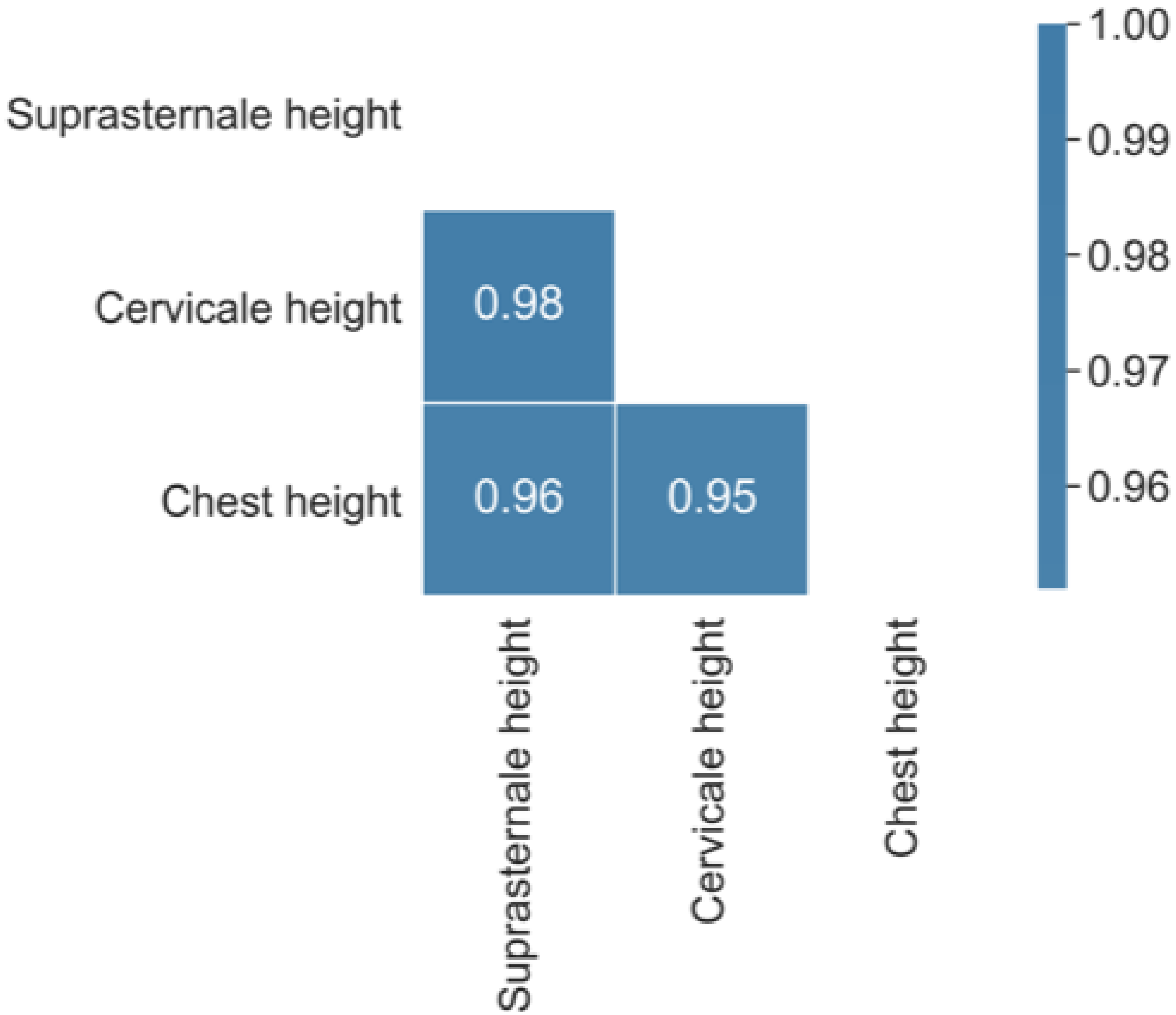
DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Head of Machine Learning, 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_df.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

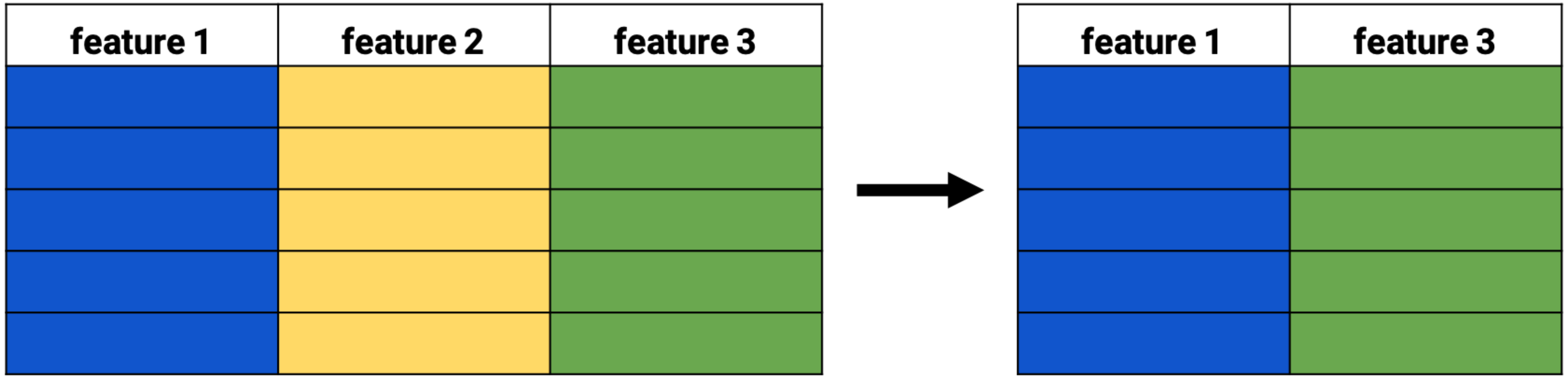
```
# Find columns that meet threshold
to_drop = [c for c in tri_df.columns if any(tri_df[c] > 0.95)]

print(to_drop)
```

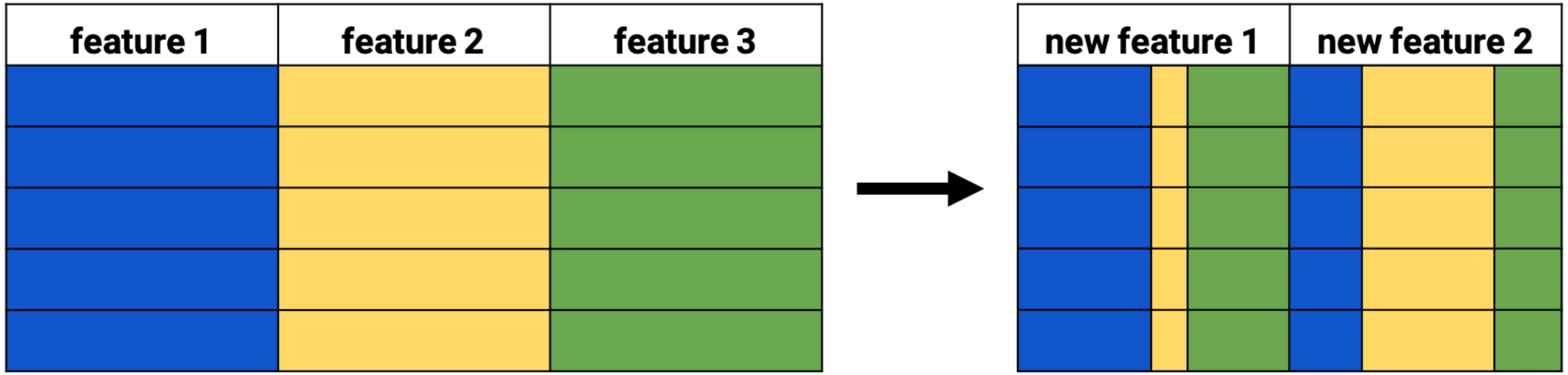
```
['Suprasternale height', 'Cervicale height']
```

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

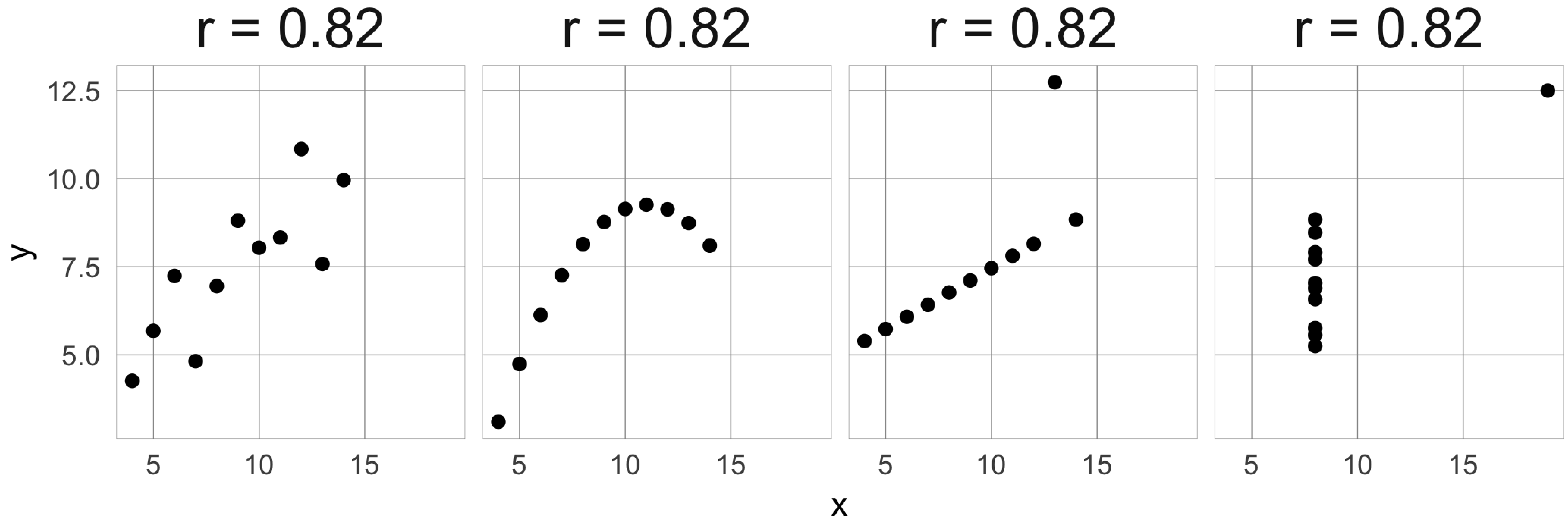
Feature selection



Feature extraction

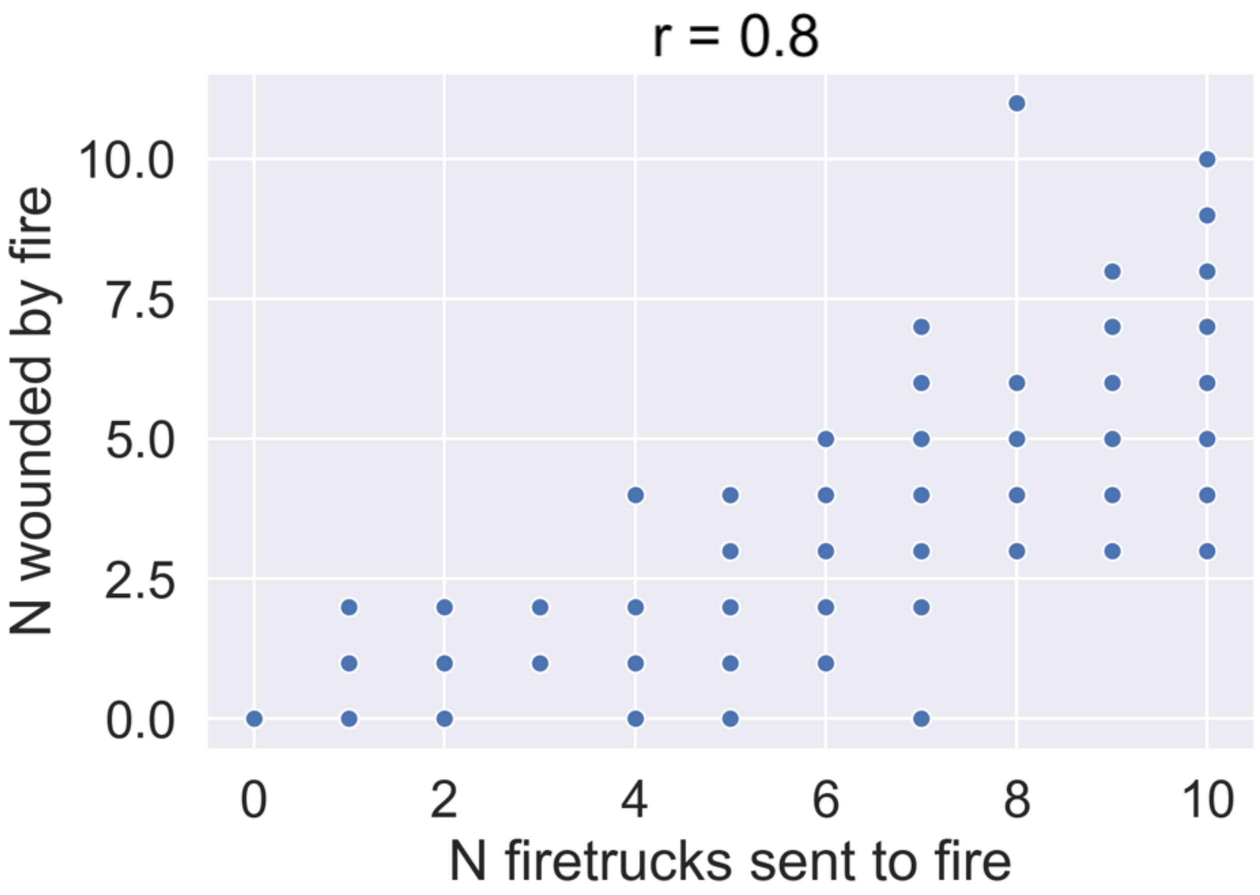


Correlation caveats - Anscombe's quartet



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