Standardization

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

James Chapman

Curriculum Manager, DataCamp

What is standardization?

Standardization: transform *continuous* data to appear *normally distributed*

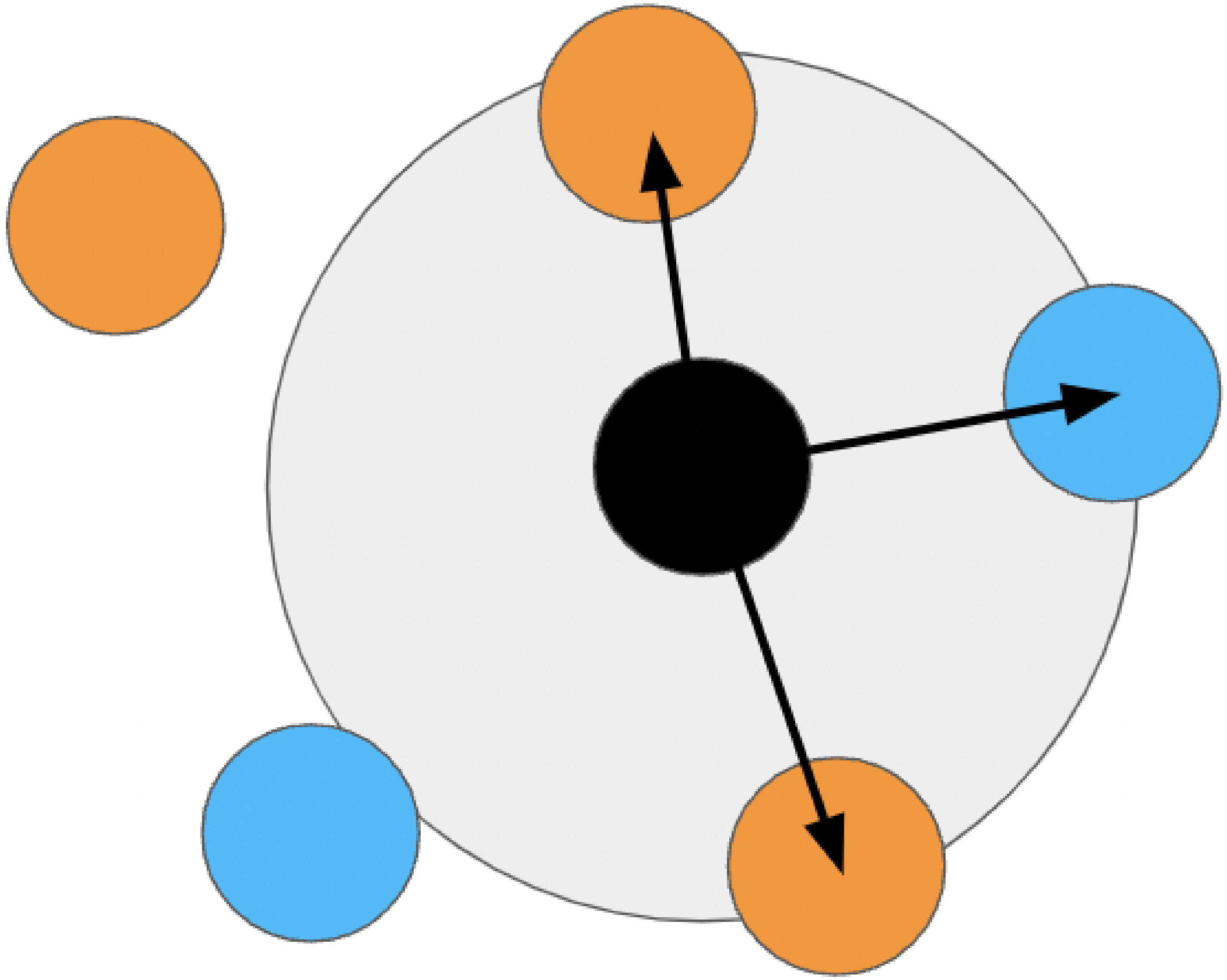
- `scikit-learn` models assume normally distributed data
- Using non-normal training data can introduce *bias*
- **Log normalization** and feature **scaling** in this course
- Applied to continuous numerical data

When to standardize: linear distances

- Model in *linear* space

Examples:

- k-Nearest Neighbors (kNN)
- Linear regression
- K-Means Clustering

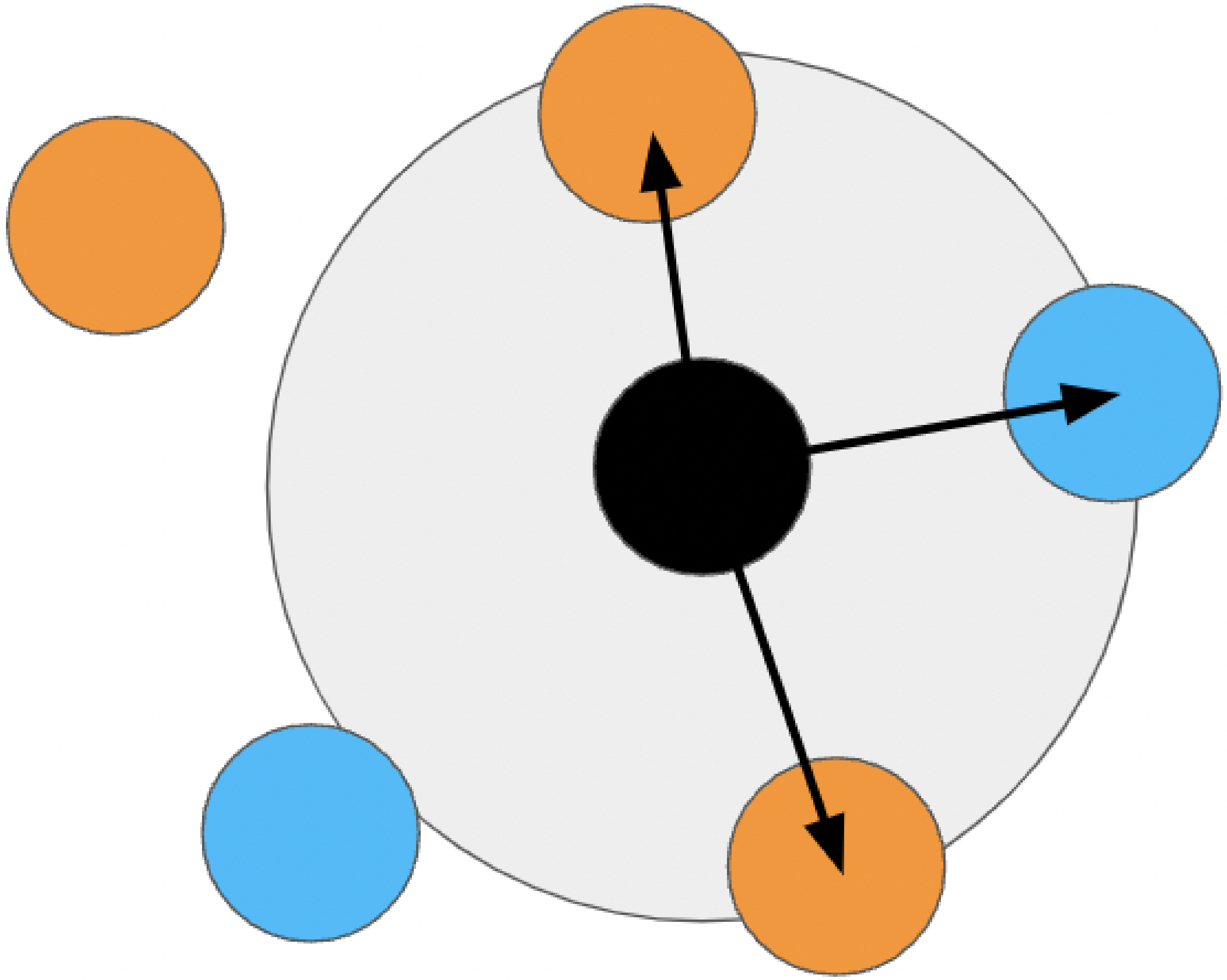


When to standardize: high variance

- Model in *linear* space

Examples:

- k-Nearest Neighbors (kNN)
- Linear regression
- K-Means Clustering



- Dataset features have *high variance*

When to standardize: different scales

- Features are on *different scales*

Example:

- Predicting house prices using *no. bedrooms* and *last sale price*

- Linearity assumptions

Let's practice!

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Log normalization

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

James Chapman
Curriculum Manager, DataCamp

What is log normalization?

- Useful for features with *high variance*
- Applies logarithm transformation
- Natural log using the constant e (≈ 2.718)

What is log normalization?

- Useful for features with *high variance*
- Applies logarithm transformation
- Natural log using the constant e (≈ 2.718)
- $e^{3.4} = 30$

Number	Log
30	3.4
300	5.7
3000	8

- Captures relative changes, the magnitude of change, and keeps everything positive

Log normalization in Python

```
print(df)
```

```
   col1   col2
0  1.00    3.0
1  1.20   45.5
2  0.75   28.0
3  1.60  100.0
```

```
print(df.var())
```

```
col1        0.128958
col2     1691.729167
dtype: float64
```

```
import numpy as np
df["log_2"] = np.log(df["col2"])
print(df)
```

```
   col1   col2     log_2
0  1.00    3.0  1.098612
1  1.20   45.5  3.817712
2  0.75   28.0  3.332205
3  1.60  100.0  4.605170
```

```
print(df[["col1", "log_2"]].var())
```

```
col1     0.128958
log_2    2.262886
dtype: float64
```

Let's practice!

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Scaling data

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

James Chapman
Curriculum Manager, DataCamp

What is feature scaling?

- Features on different scales
- Model with linear characteristics
- Center features around 0 and transform to variance of 1
- Transforms to approximately normal distribution

How to scale data

```python
print(df)
```

```
   col1  col2   col3
0  1.00  48.0  100.0
1  1.20  45.5  101.3
2  0.75  46.2  103.5
3  1.60  50.0  104.0
```

```python
print(df.var())
```

```
col1    0.128958
col2    4.055833
col3    3.526667
dtype: float64
```

How to scale data

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df),
                         columns=df.columns)
```

```
print(df_scaled)
```

```
       col1      col2      col3
0 -0.442127  0.329683 -1.352726
1  0.200967 -1.103723 -0.553388
2 -1.245995 -0.702369  0.799338
3  1.487156  1.476409  1.106776
```

```
print(df_scaled.var())
```

```
col1    1.333333
col2    1.333333
col3    1.333333
dtype: float64
```

Let's practice!

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Standardized data and modeling

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

James Chapman
Curriculum Manager, DataCamp

K-nearest neighbors

- **Data leakage:** non-training data is used to train the model

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)
knn = KNeighborsClassifier()
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

knn.fit(X_train_scaled, y_train)
knn.score(X_test_scaled, y_test)
```

Let's practice!

PREPROCESSING FOR MACHINE LEARNING IN PYTHON