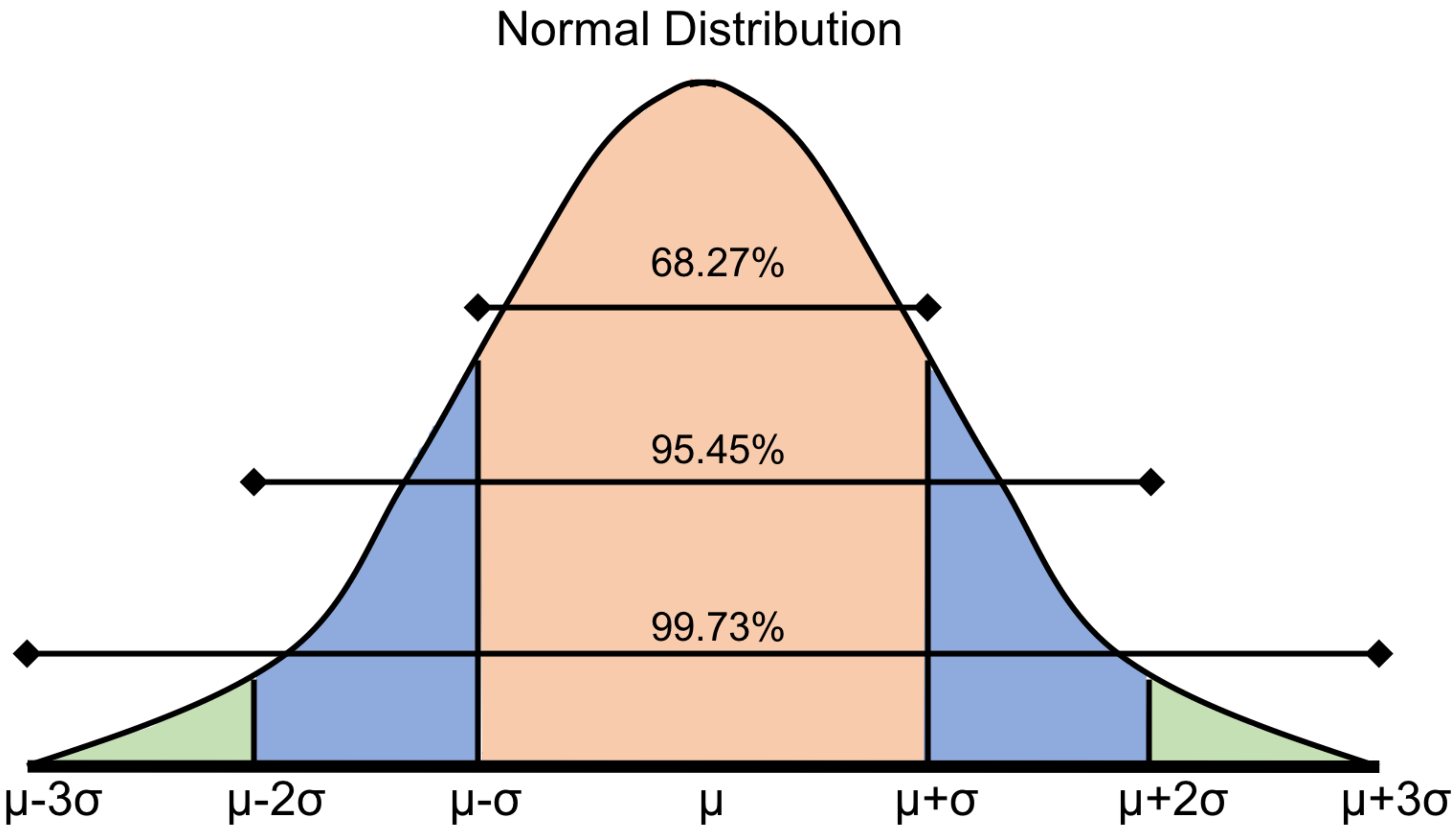
Data distributions

FEATURE ENGINEERING FOR MACHINE LEARNING IN PYTHON

Robert O'Callaghan
Director of Data Science,
Ordergroove

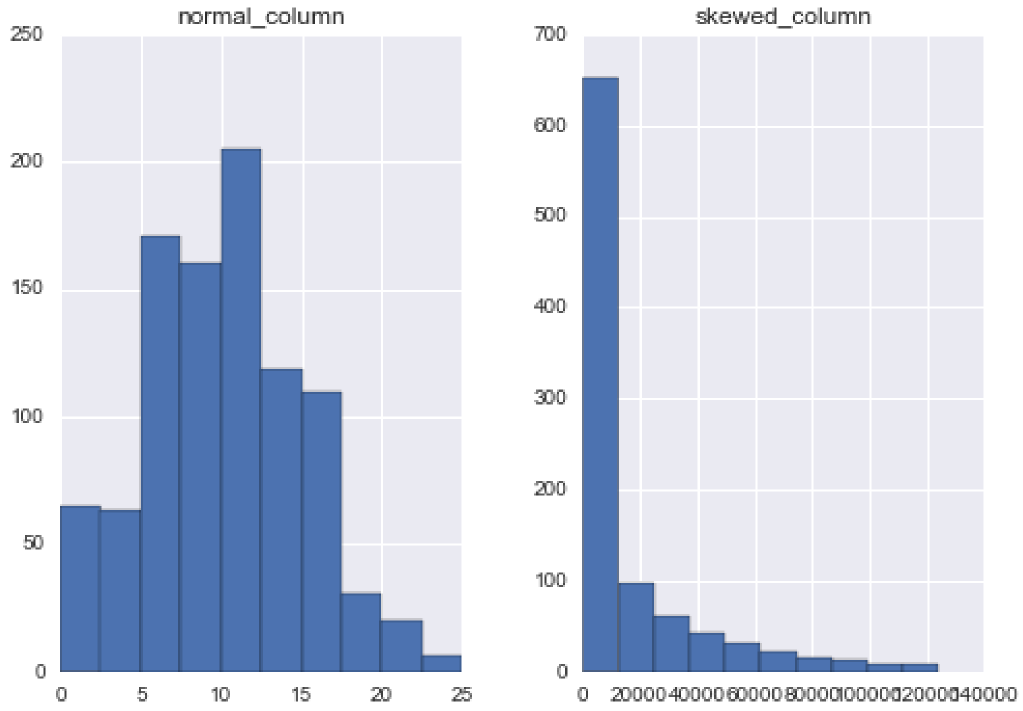
Distribution assumptions



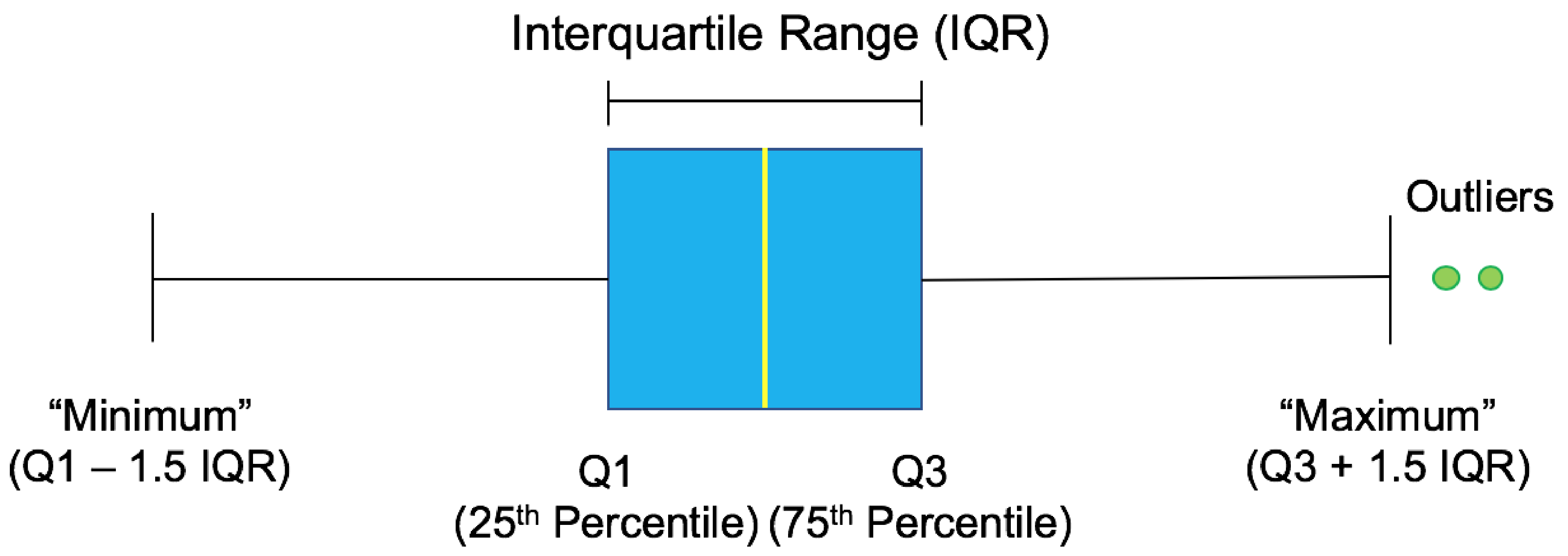
Observing your data

```python
import matplotlib as plt

df.hist()
plt.show()
```

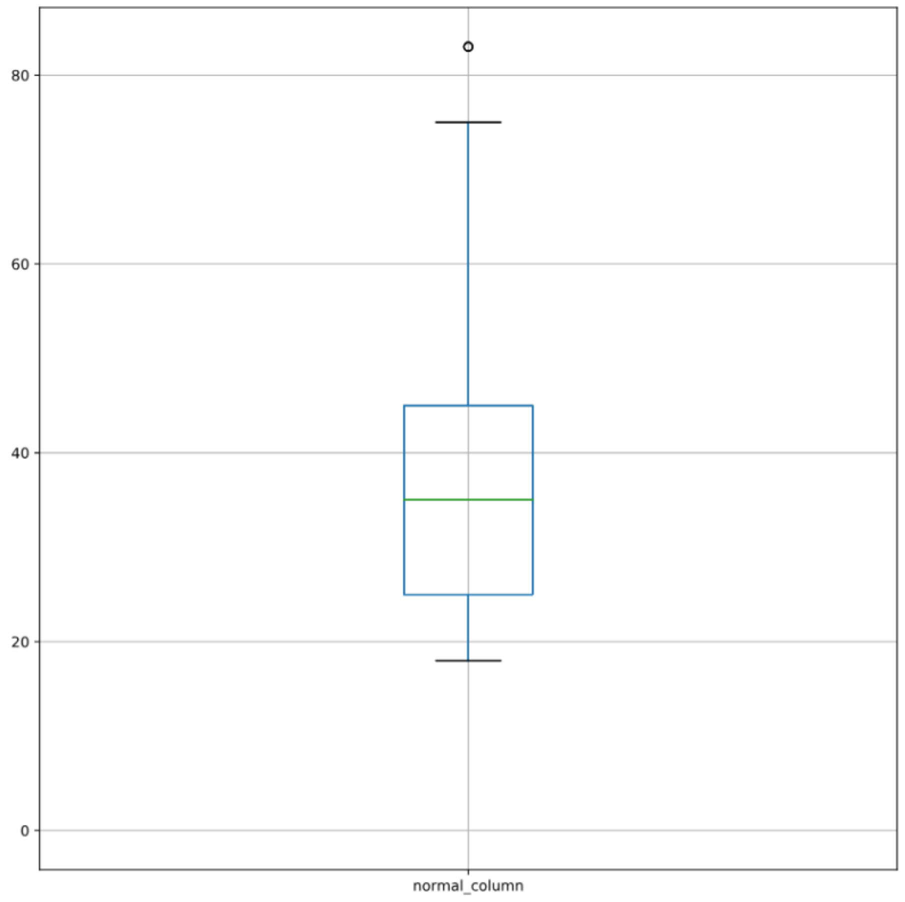


Delving deeper with box plots



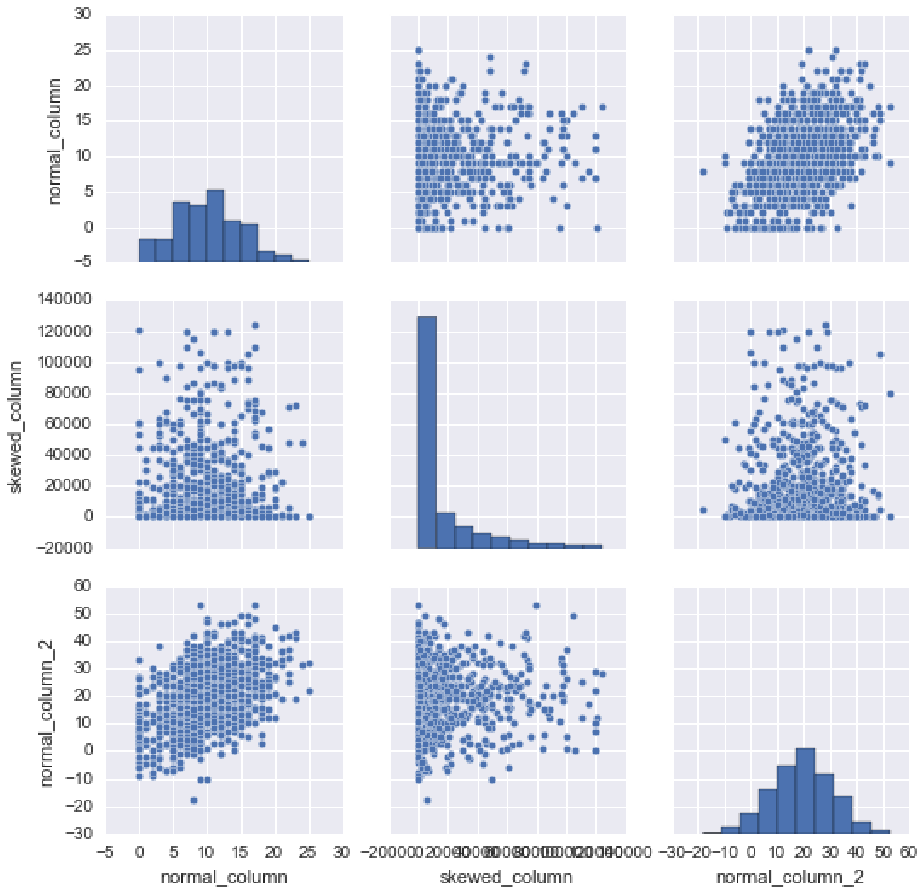
Box plots in pandas

```
df[['column_1']].boxplot()
plt.show()
```



Paring distributions

```python
import seaborn as sns
sns.pairplot(df)
```



Further details on your distributions

```
df.describe()
```

	Col1	Col2	Col3	Col4
count	100.000000	100.000000	100.000000	100.000000
mean	-0.163779	-0.014801	-0.087965	-0.045790
std	1.046370	0.920881	0.936678	0.916474
min	-2.781872	-2.156124	-2.647595	-1.957858
25%	-0.849232	-0.655239	-0.602699	-0.736089
50%	-0.179495	0.032115	-0.051863	0.066803
75%	0.663515	0.615688	0.417917	0.689591
max	2.466219	2.353921	2.059511	1.838561

Let's practice!

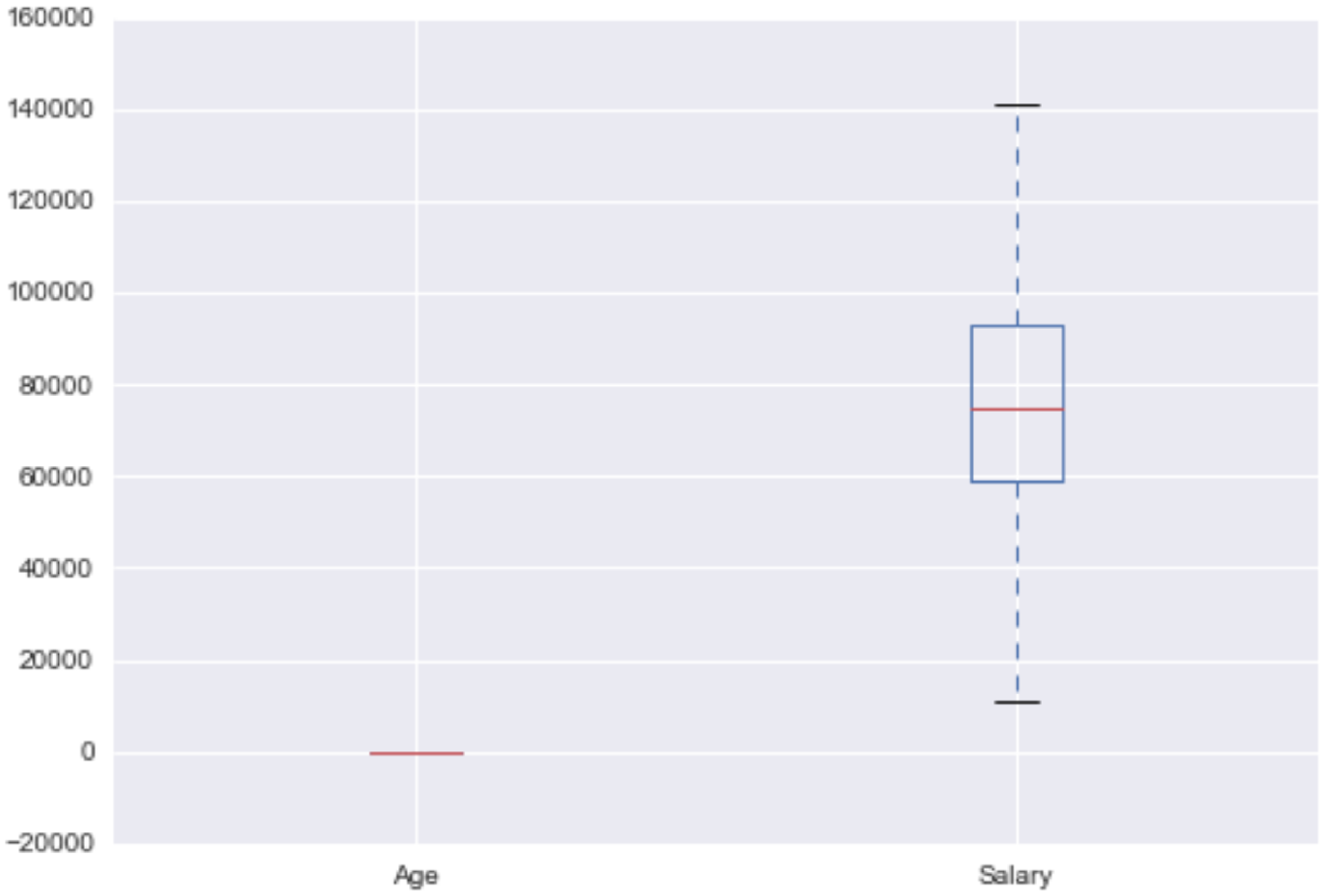
FEATURE ENGINEERING FOR MACHINE LEARNING IN PYTHON

Scaling and transformations

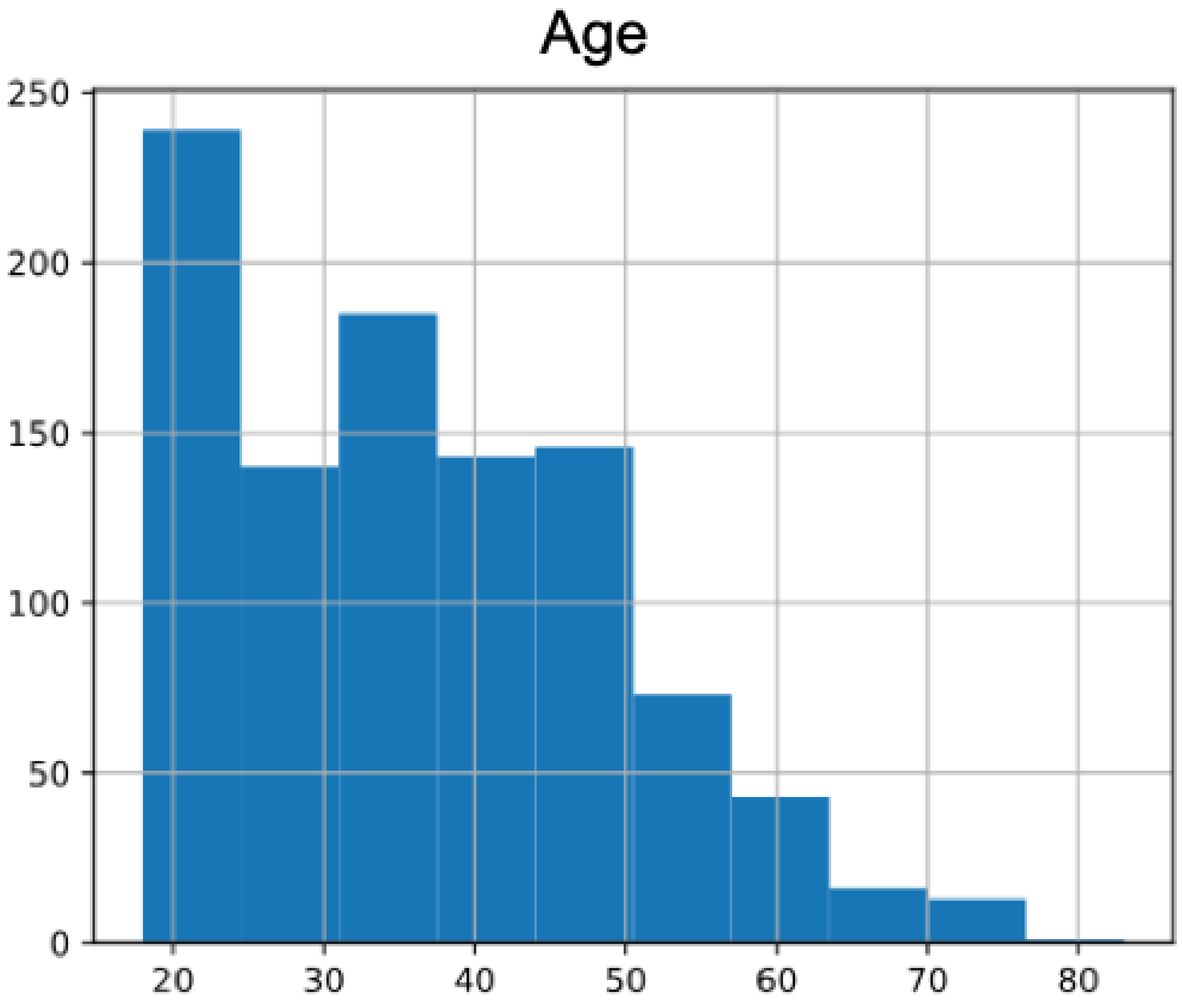
FEATURE ENGINEERING FOR MACHINE LEARNING IN PYTHON

Robert O'Callaghan
Data Scientist

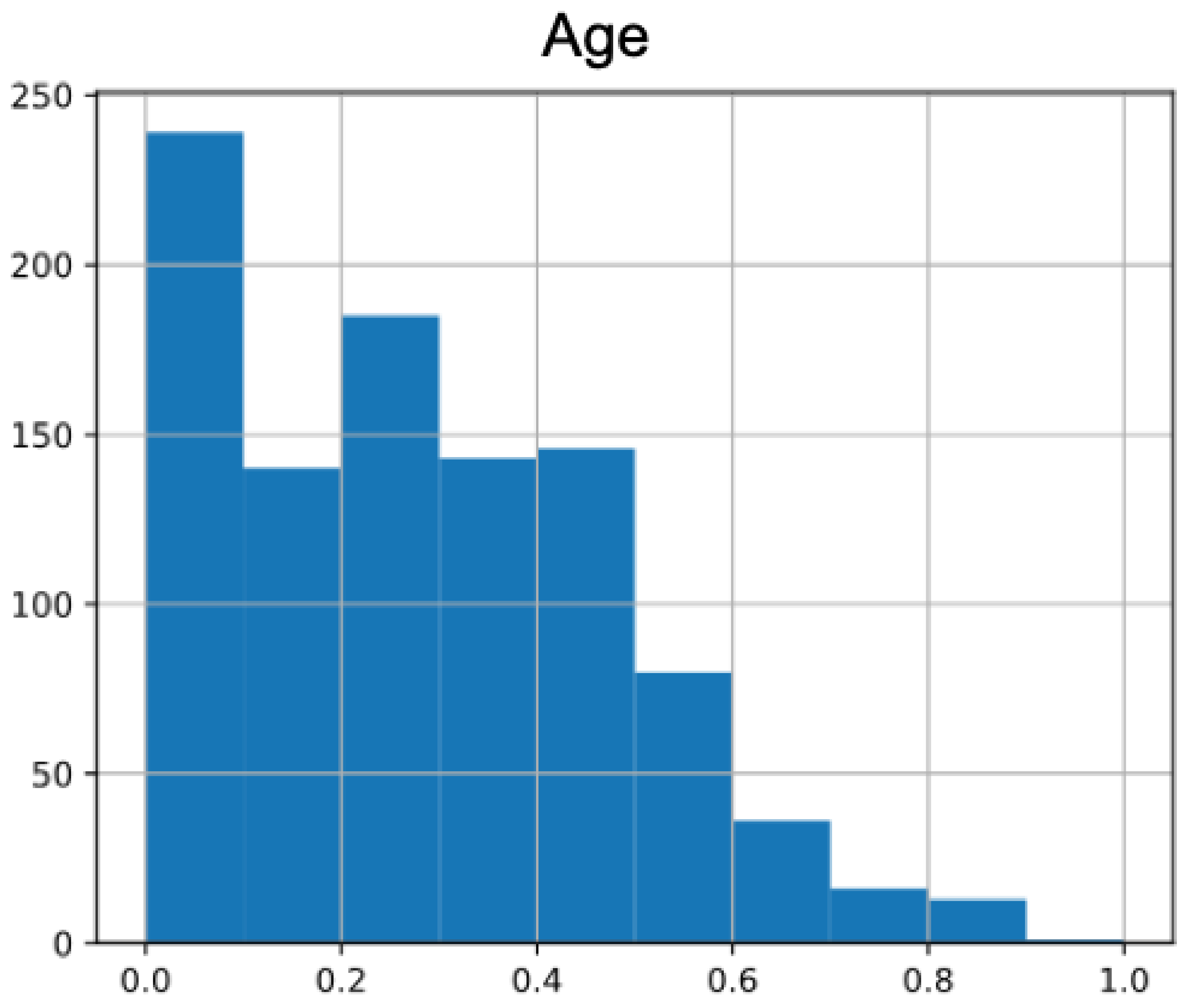
Scaling data



Min-Max scaling



Min-Max scaling



Min-Max scaling in Python

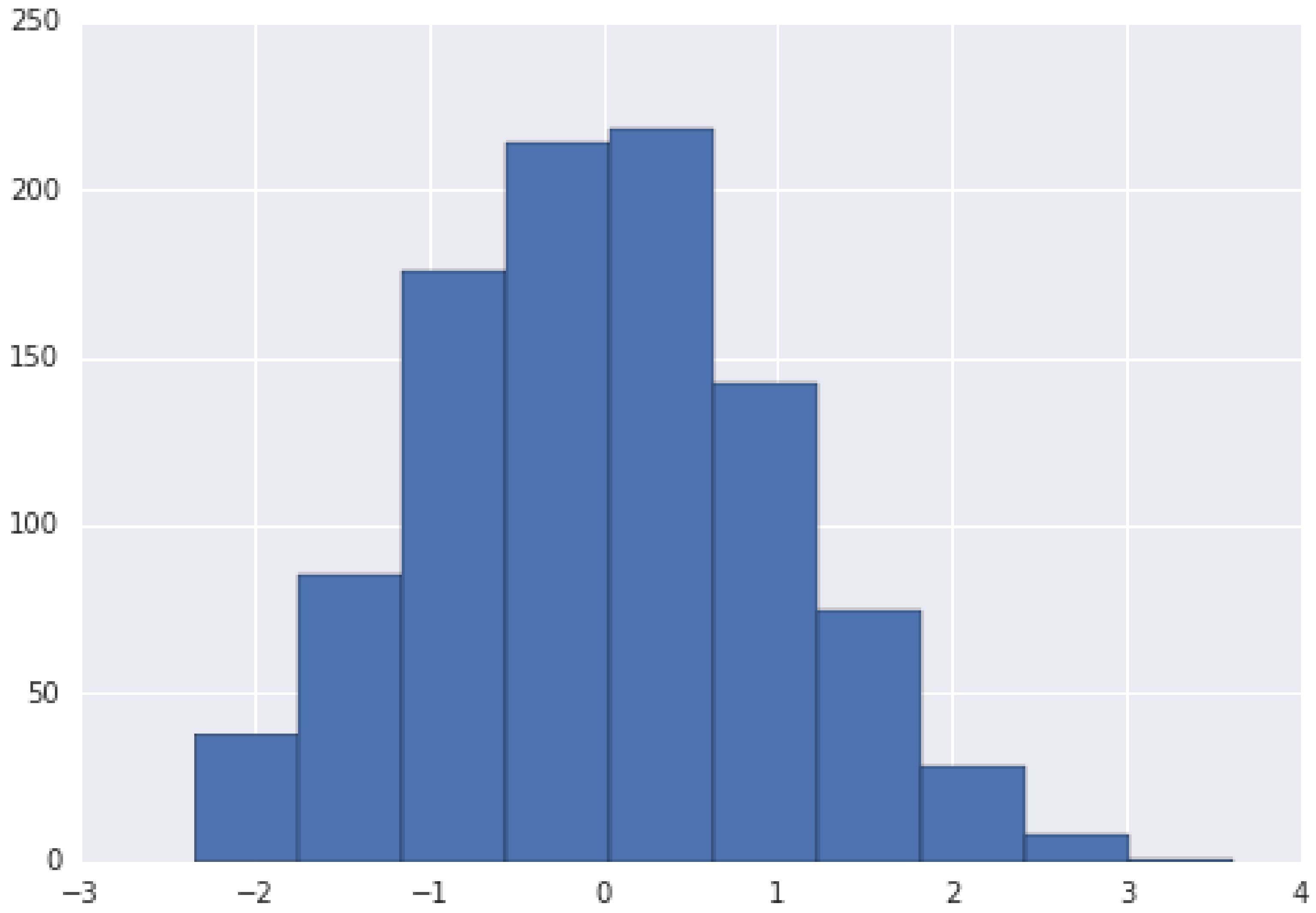
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit(df[['Age']])

df['normalized_age'] = scaler.transform(df[['Age']])
```

Standardization



Standardization in Python

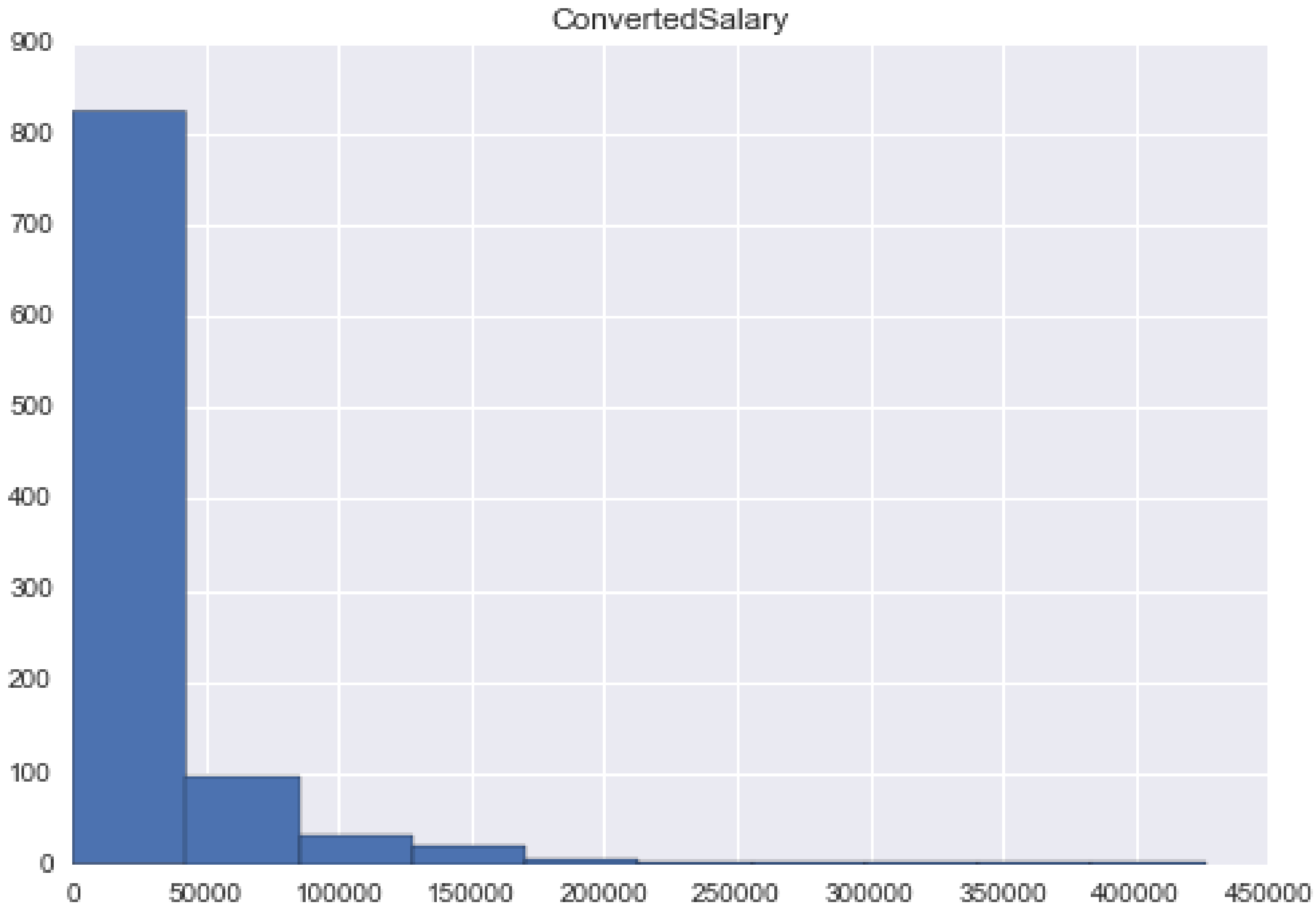
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(df[['Age']])

df['standardized_col'] = scaler\
                        .transform(df[['Age']])
```

Log Transformation



Log transformation in Python

```
from sklearn.preprocessing import PowerTransformer

log = PowerTransformer()

log.fit(df[['ConvertedSalary']])

df['log_ConvertedSalary'] =
     log.transform(df[['ConvertedSalary']])
```

Final Slide

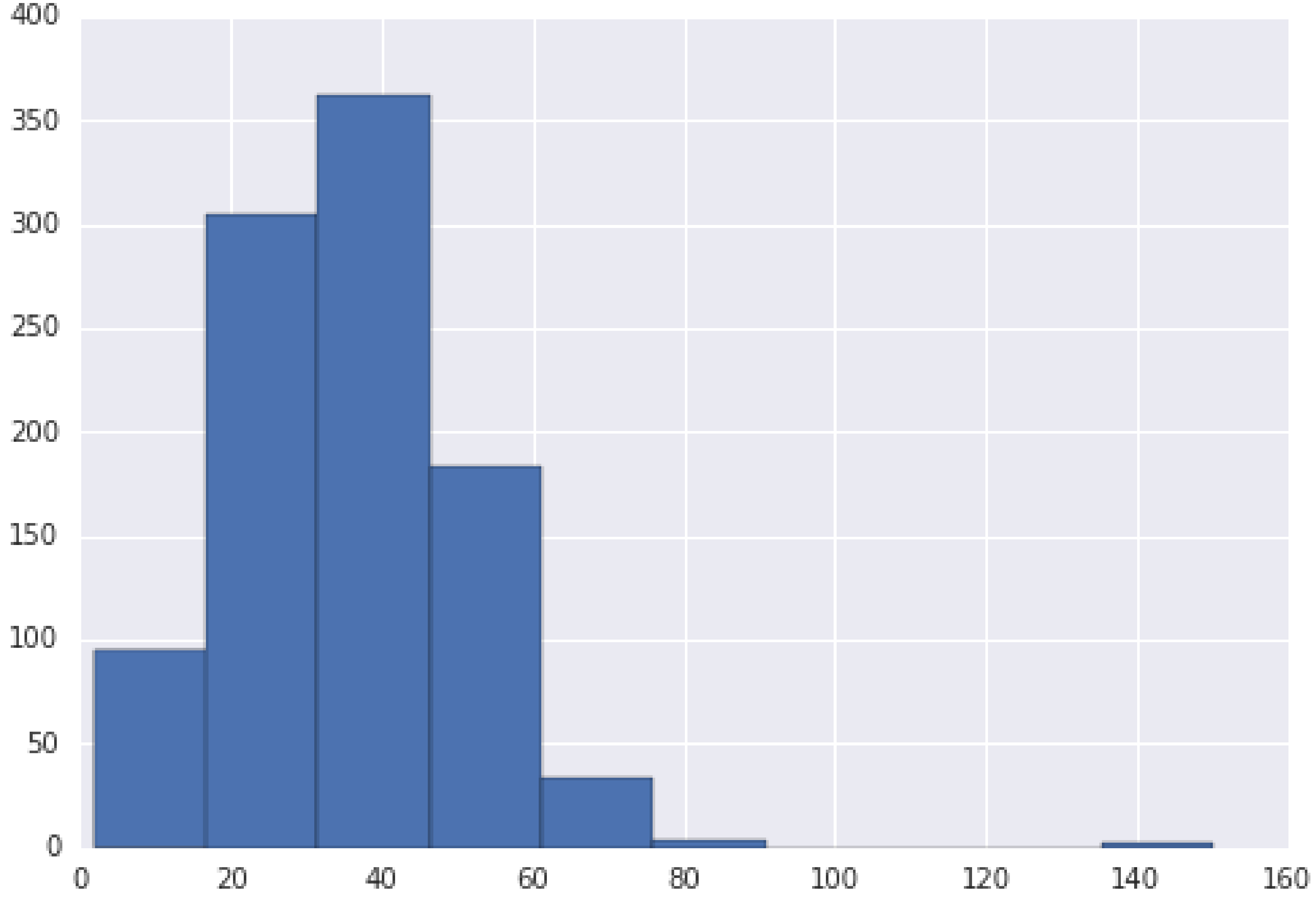
FEATURE ENGINEERING FOR MACHINE LEARNING IN PYTHON

Removing outliers

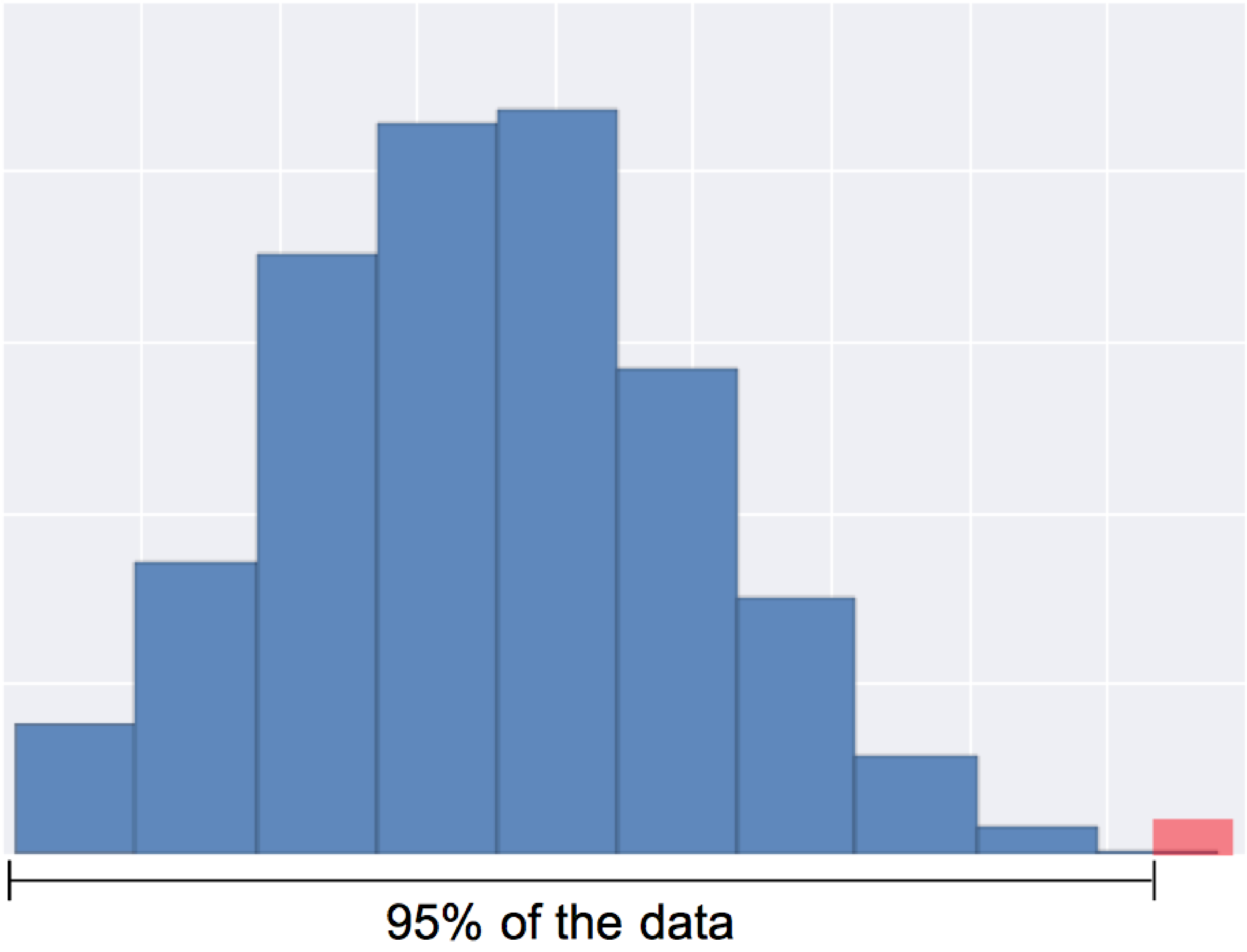
FEATURE ENGINEERING FOR MACHINE LEARNING IN PYTHON

Robert O'Callaghan
Director of Data Science, Ordergroove

What are outliers?



Quantile based detection



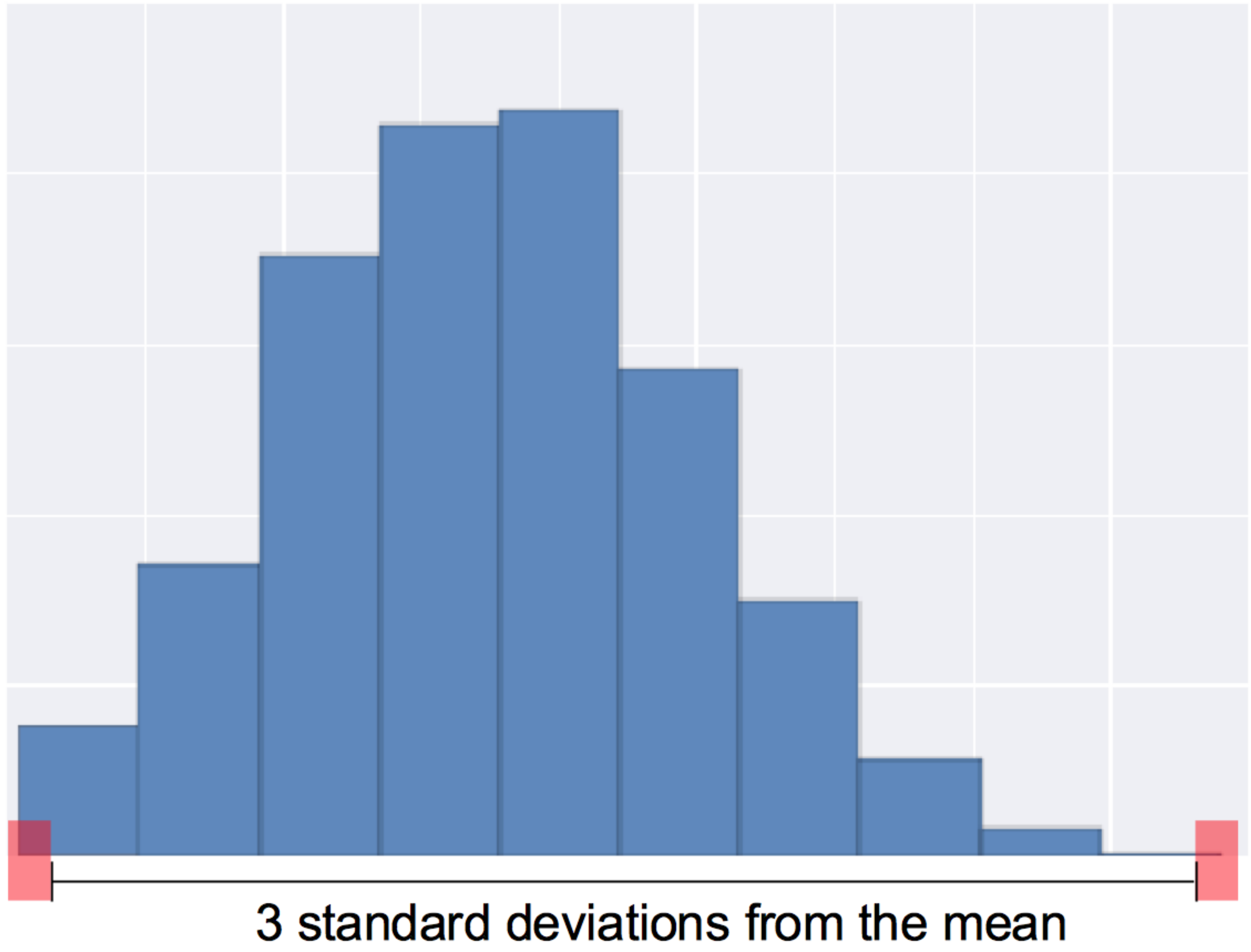
Quantiles in Python

```
q_cutoff = df['col_name'].quantile(0.95)

mask = df['col_name'] < q_cutoff

trimmed_df = df[mask]
```

Standard deviation based detection



Standard deviation detection in Python

```
mean = df['col_name'].mean()
std = df['col_name'].std()
cut_off = std * 3


lower, upper = mean - cut_off, mean + cut_off
new_df = df[(df['col_name'] < upper) &
                 (df['col_name'] > lower)]
```

Let's practice!

FEATURE ENGINEERING FOR MACHINE LEARNING IN PYTHON

Scaling and transforming new data

FEATURE ENGINEERING FOR MACHINE LEARNING IN PYTHON

Robet O'Callaghan
Director of Data Science, Ordergroove

Reuse training scalers

```
scaler = StandardScaler()

scaler.fit(train[['col']])

train['scaled_col'] = scaler.transform(train[['col']])

# FIT SOME MODEL
# ....

test = pd.read_csv('test_csv')

test['scaled_col'] = scaler.transform(test[['col']])
```

Training transformations for reuse

```
train_mean = train[['col']].mean()
train_std = train[['col']].std()

cut_off = train_std * 3
train_lower = train_mean - cut_off
train_upper = train_mean + cut_off

# Subset train data

test = pd.read_csv('test_csv')

# Subset test data
test = test[(test[['col']] < train_upper) &
              (test[['col']] > train_lower)]
```

Why only use training data?

Data leakage: Using data that you won't have access to when assessing the performance of your model

Avoid data leakage!

FEATURE ENGINEERING FOR MACHINE LEARNING IN PYTHON