Robust Scaling

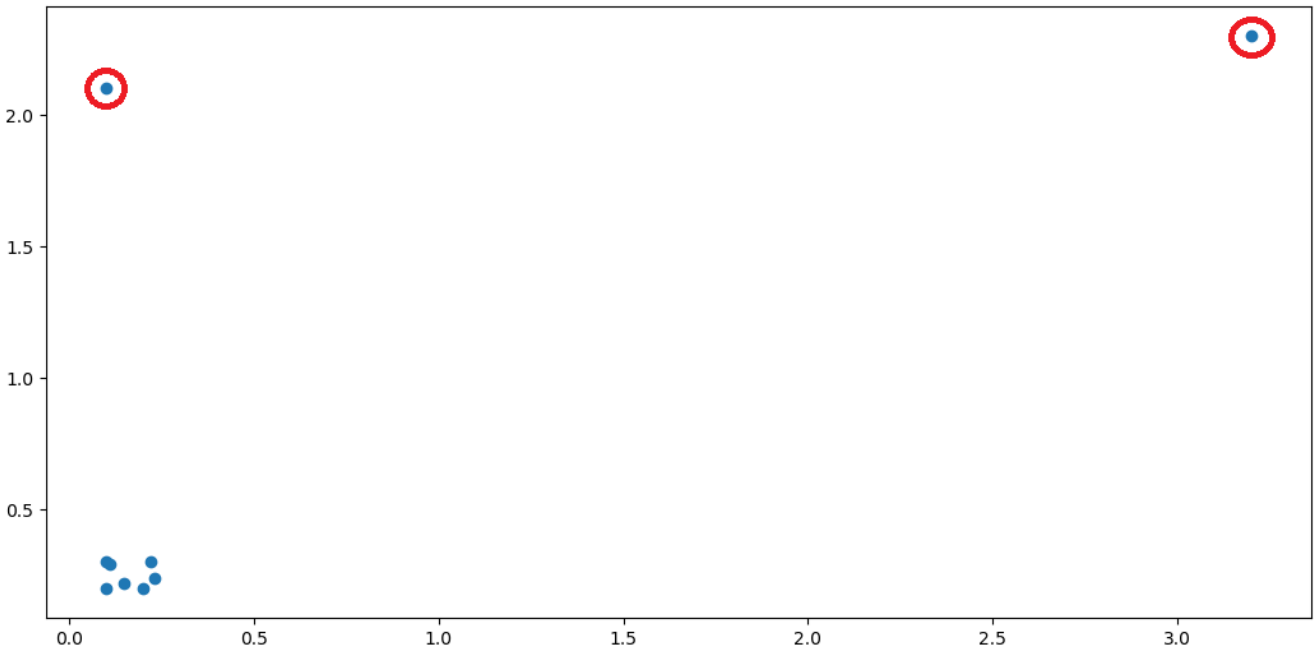
Understand how outliers can affect data and implement robust scaling.

Chapter Goals:

- Learn how to scale data without being affected by outliers

A. Data outliers

An important aspect of data that we have to deal with is *outliers*. In general terms, an outlier is a data point that is significantly further away from the other data points. For example, if we had watermelons of weights 5, 4, 6, 7, and 20 pounds, the 20 pound watermelon is an outlier.



A 2-D data plot with the outlier data points circled. Note that the outliers in this plot are exaggerated, and in real life outliers are not usually this far from the non-outlier data.

The data scaling methods from the previous two chapters are both affected by outliers. Data standardization uses each feature's mean and standard deviation, while ranged scaling uses the maximum and minimum feature values, meaning that they're both susceptible to being skewed by outlier values.

We can robustly scale the data, i.e. avoid being affected by outliers, by using use the data's median and Interquartile Range (IQR). Since the median and IQR are percentile measurements of the data (50% for median, 25% to 75% for the IQR), they are not affected by outliers. For the scaling method, we just subtract the median from each data value then scale to the IQR.

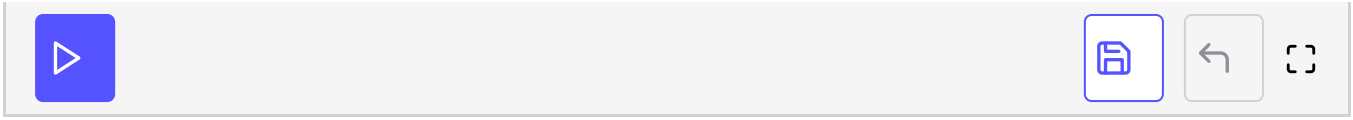
B. Robust scaling with scikit-learn

In scikit-learn, we perform robust scaling with the `RobustScaler` module. It is another transformer object, with the same `fit`, `transform`, and `fit_transform` functions described in the previous chapter.

The code below shows how to use the `RobustScaler`.

```
# predefined data
print('{}\n'.format(repr(data)))

from sklearn.preprocessing import RobustScaler
robust_scaler = RobustScaler()
transformed = robust_scaler.fit_transform(data)
print('{}\n'.format(repr(transformed)))
```



Time to Code!

The coding exercise in this chapter uses `RobustScaler` (imported in backend) to complete the `robust_scaling` function.

The function will apply outlier-independent scaling to the input NumPy array, `data`.

Set `robust_scaler` equal to `RobustScaler` initialized without any parameters.

Set `scaled_data` equal to `robust_scaler.fit_transform` applied with `data` as the only argument. Then return `scaled_data`.

```
def robust_scaling(data):
  # CODE HERE
  pass
```