Research Project

Statistical Foundations of Machine Learning

Ward Gauderis & Fabian Denoodt

27/06/2022

Vrije Universiteit Brussel

Summary

1. Stochastic noise and weight decay regularisation
2. Support vector machine kernel comparison
3. Decision tree and k-nearest neighbours regressor forecasting

Stochastic noise and weight decay regularisation

What is the influence of stochastic noise in the dataset on the in- and out-of-sample error of a neural network and how does weight decay regularisation counter this?

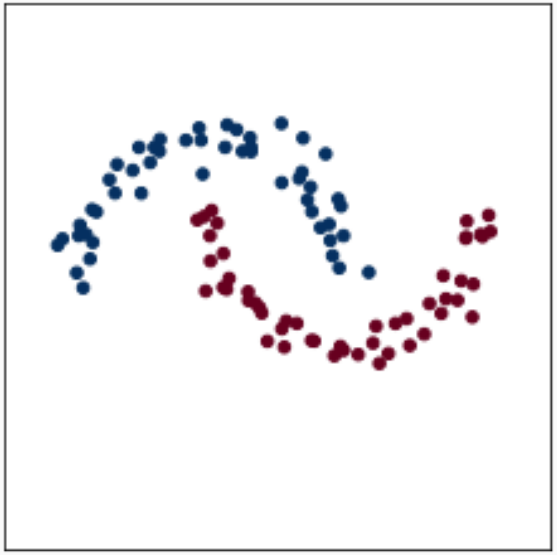
The datasets

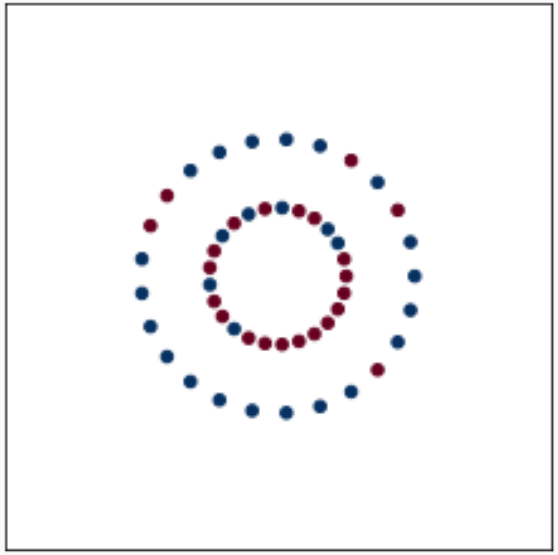
Synthetic normalised (x, y) with $x \in \mathbb{R}^2, y \in \{0, 1\}$

Controllable stochastic noise:

- **Data noise**: $x' = x + \epsilon$ with $\epsilon \sim \mathbb{N}(0, \sigma^2)$
- **Label noise**: swap fraction α of labels y

No confounding deterministic noise





The models

Neural network

- Predict $h(x) = P(y = 1|x)$
- Linear transformations with non-linear differentiable ReLU activation functions
- Maximise likelihood by minimising the cross-entropy error
- $(20, 20)$ hidden layers & 2000 epochs

Augmented error regularisation:

$$E_{aug}(h, \lambda, \Omega) = E_{in}(h) + \frac{\lambda}{N}\Omega(h)$$

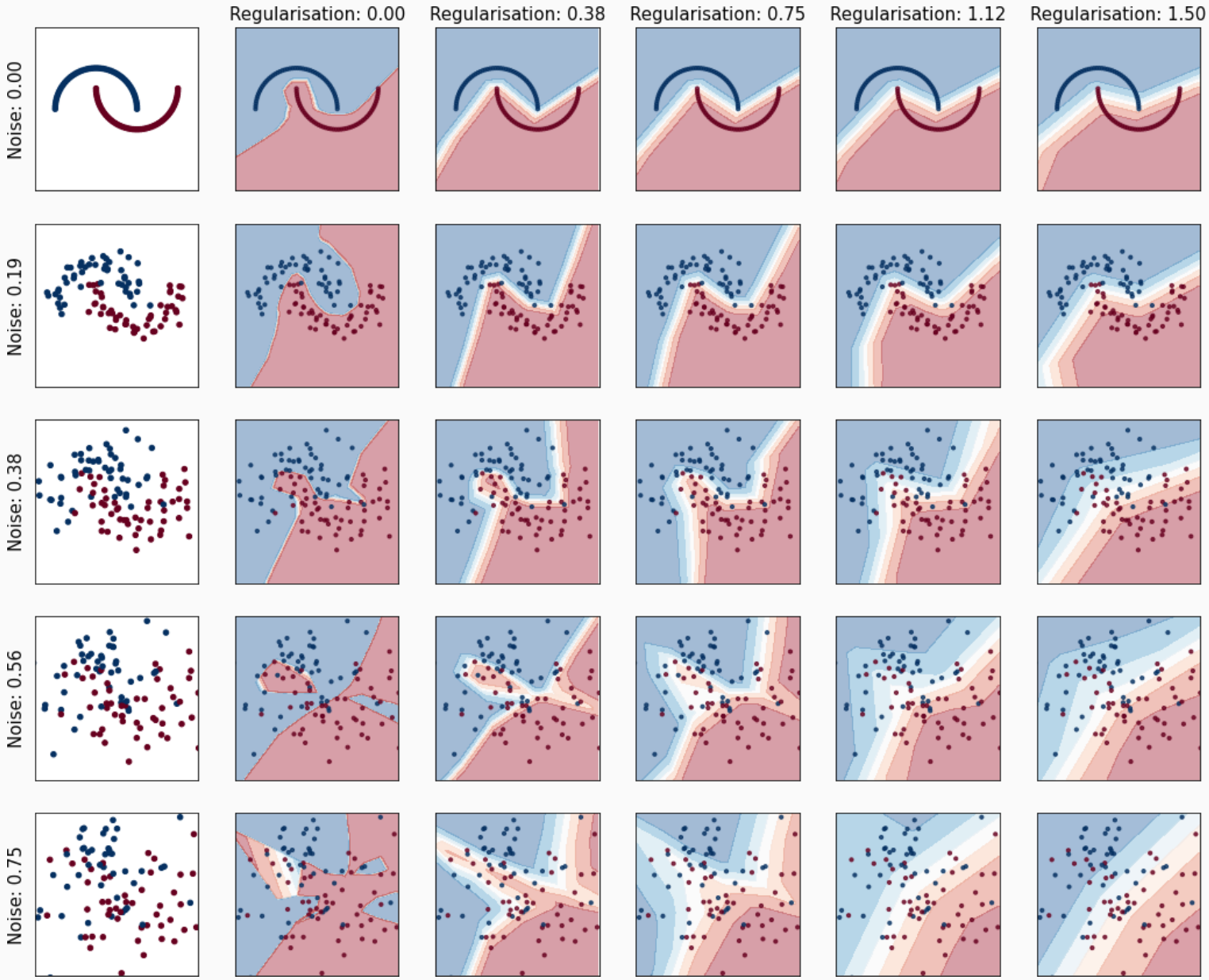
Weight-decay with L_2 norm:

$$E_{aug}(w) = E_{in}(w) + \frac{\lambda}{N}||w||^2$$

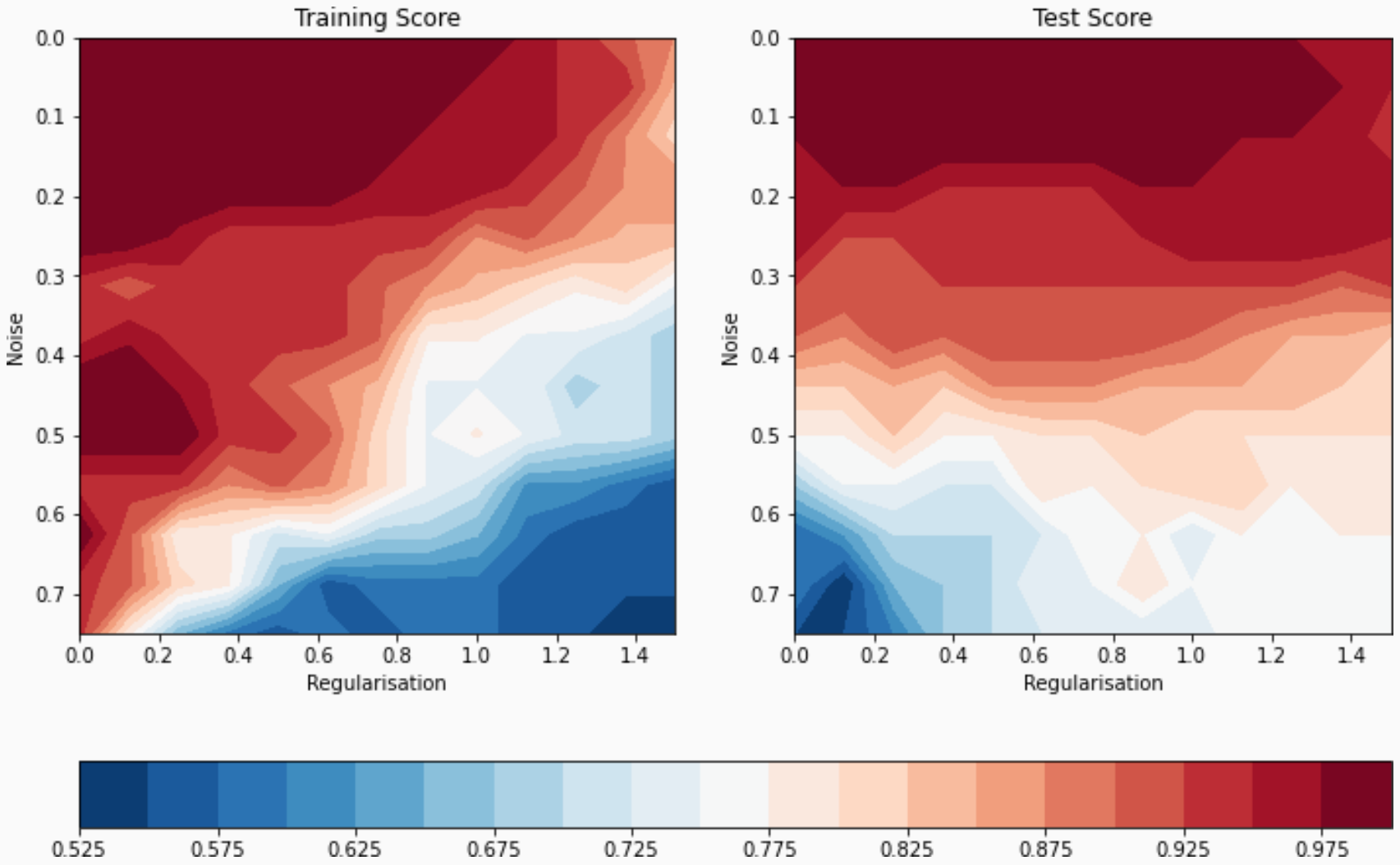
Experimental setup

Repeat for every combination of dataset type, label and data noise:

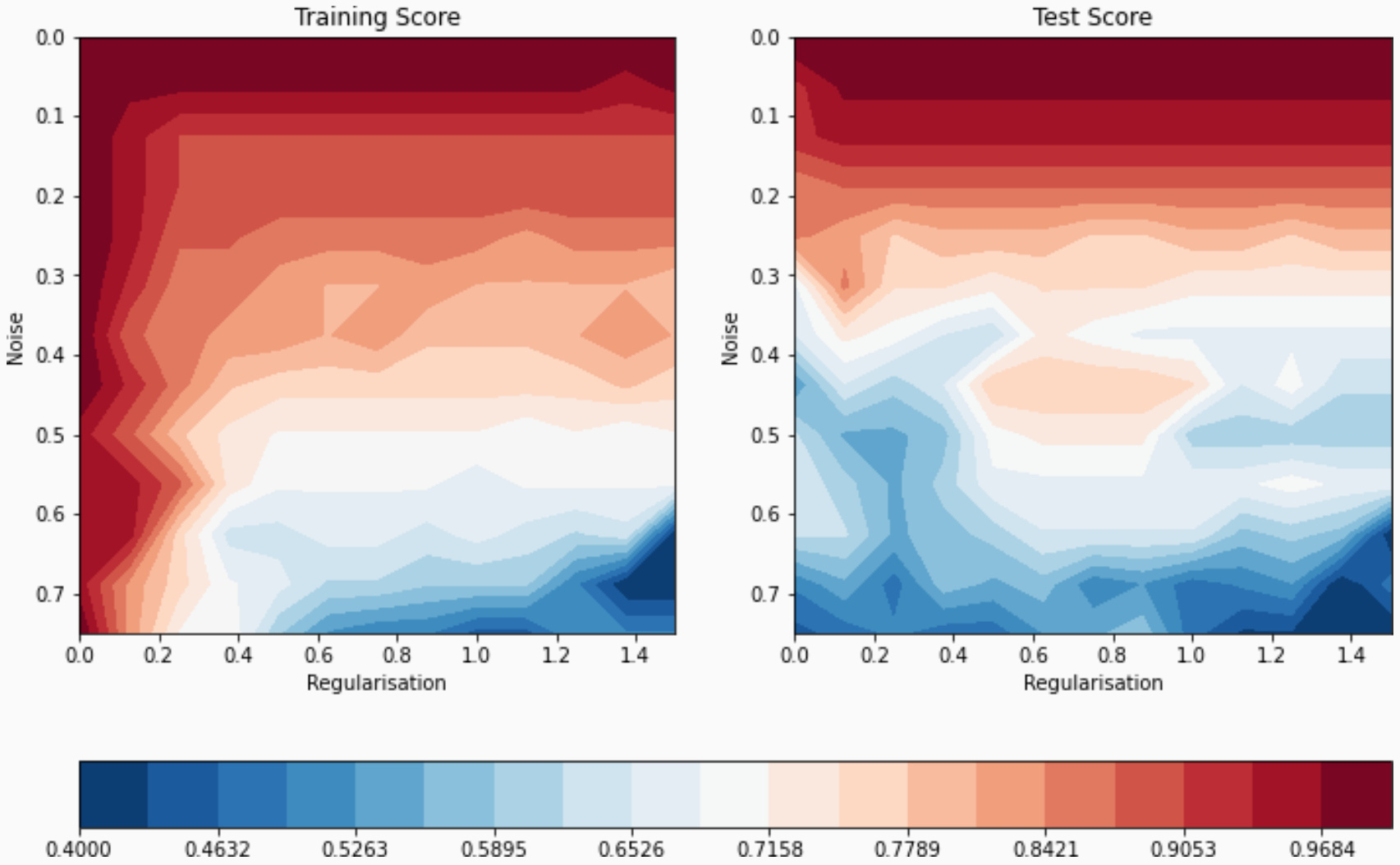
1. Generate 13 datasets of size 100 with noise $\in [0, 0.75]$
2. Create 13 models of with regularisation $\in [0, 1.5]$
3. Train every model on every dataset
4. Compare decision boundaries and training and testing accuracies



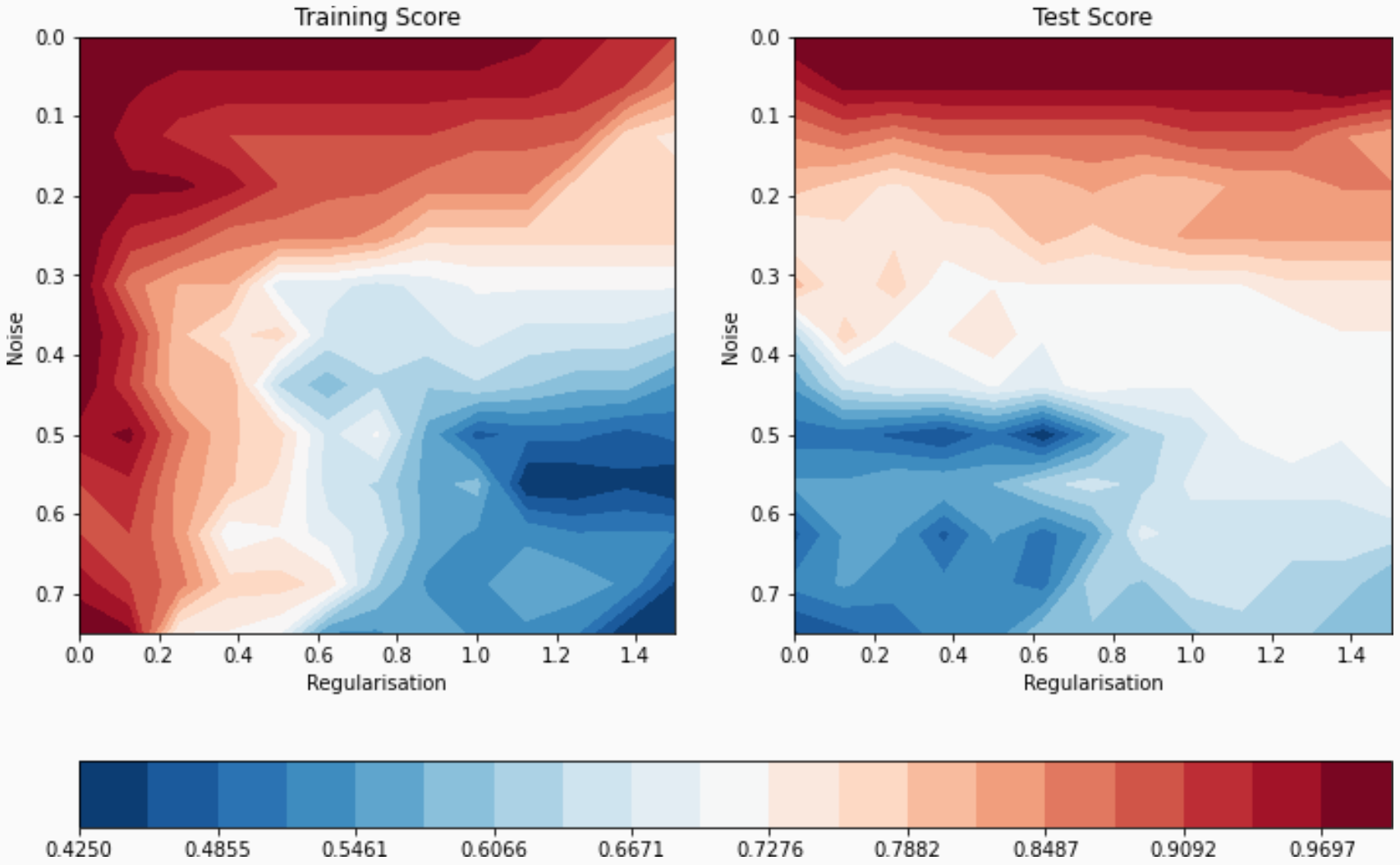
Moon dataset with data noise



Moon dataset with data noise



Circles dataset with label noise



Moon dataset with combined noise

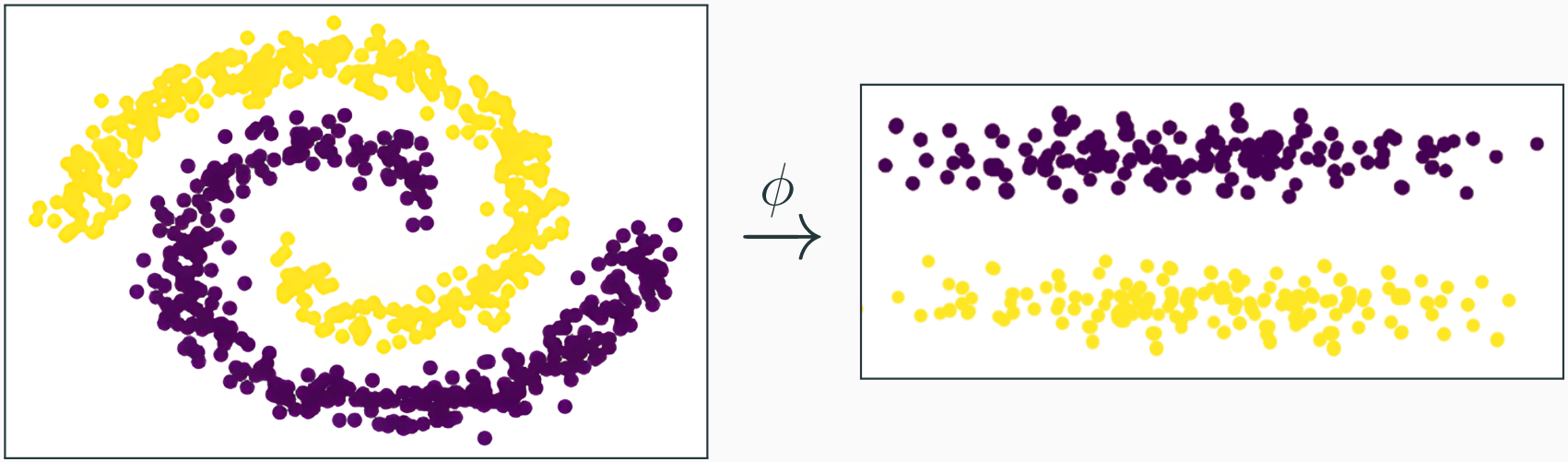
Conclusions

- Neural networks with less effective parameters generalise better on noisier datasets
- Label noise versus data noise
  - more detrimental to generalisability
  - can be combated with less regularisation
- Underfitting is less punishing than overfitting
- E_{in} becomes less informative about E_{out} with increasing noise and regularisation
- Optimal λ is hard to know up front without data snooping and should be chosen through model selection

Support vector machine kernel comparison

How do the linear kernel, polynomial kernel and radial basis function compare to each other, when applied to a synthetic two-dimensional dataset?

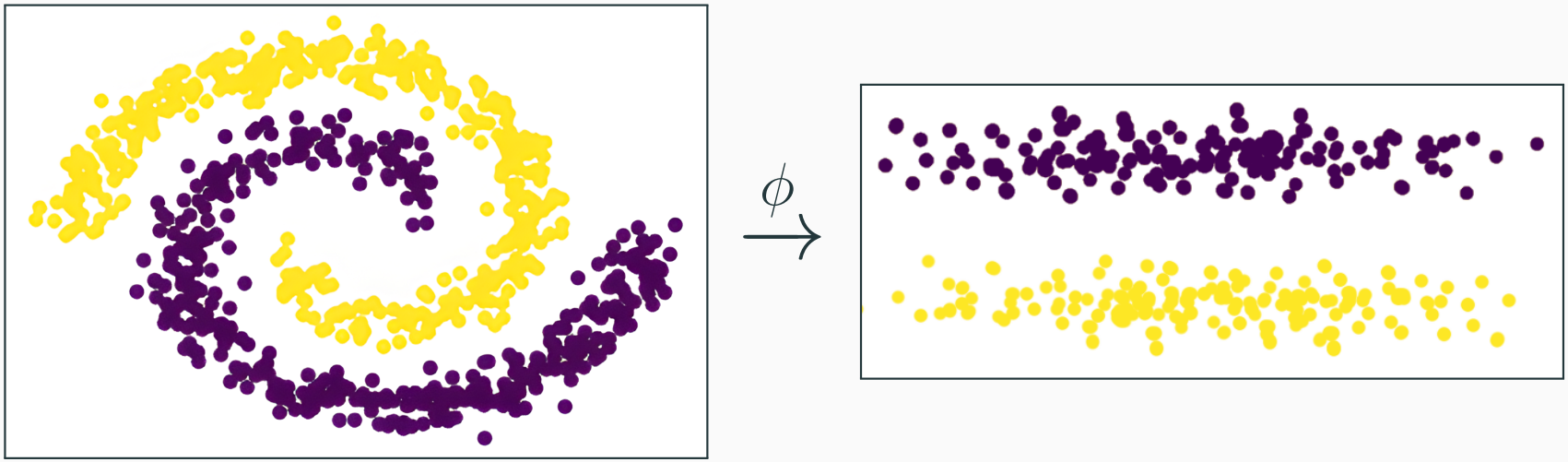
Why do we even need kernels?



$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x_n}^T \mathbf{x}_m$$

$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \Phi(\mathbf{x_n})^T \Phi(\mathbf{x}_m)$$

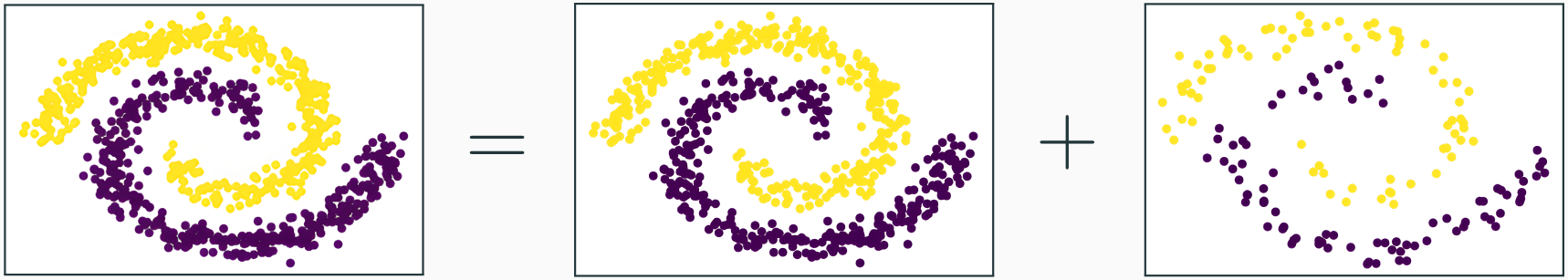
Why do we even need kernels?



$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x_n}^T \mathbf{x}_m$$

$$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m K(\mathbf{x_n}, \mathbf{x}_m)$$

Setup



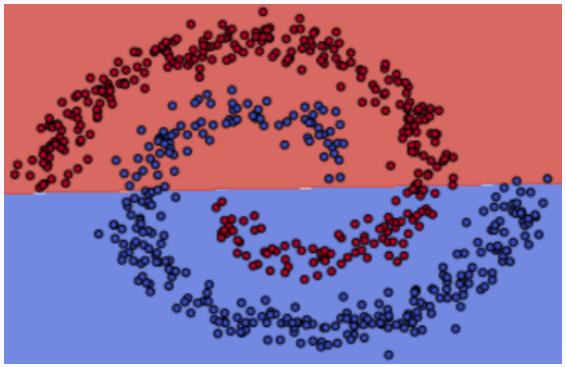
```
> linear = svm.SVC(kernel='linear', C=1000)
> linear = model.fit(X_train, y_train)

> poly = …
> rbf = …
```

Soft margin constraints

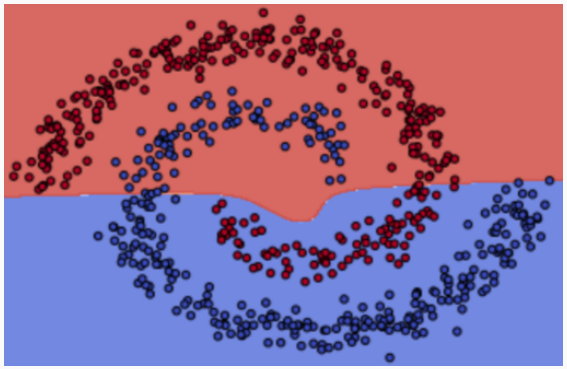
$$\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}\zeta_n$$

Results and Interpretation



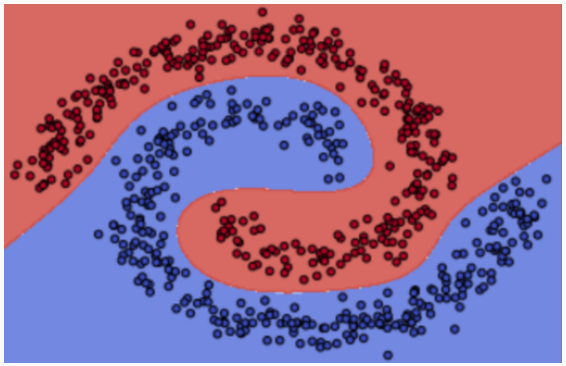
Linear

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$



Polynomial

$$K(\mathbf{x}, \mathbf{x}') = (1 + \gamma(\mathbf{x}^T \mathbf{x}'))^Q$$

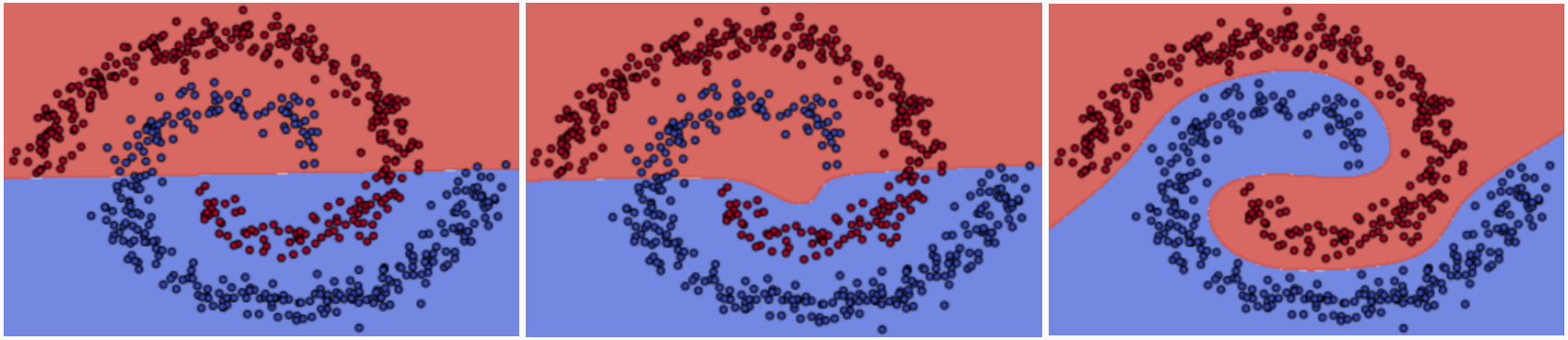


RBF

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma ||\mathbf{x} - \mathbf{x}'||^2}$$

```
Q = 3
gamma = 1 / ({N_train} * X_train.var())
```

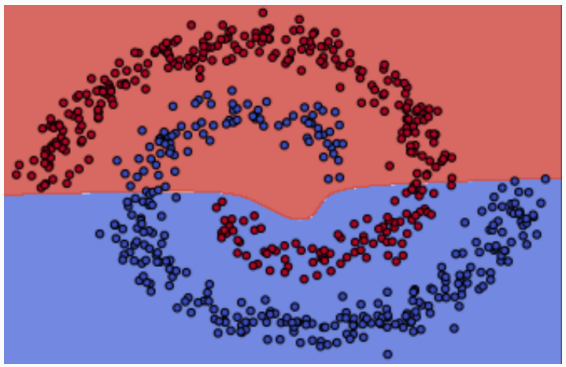
Results and Interpretation



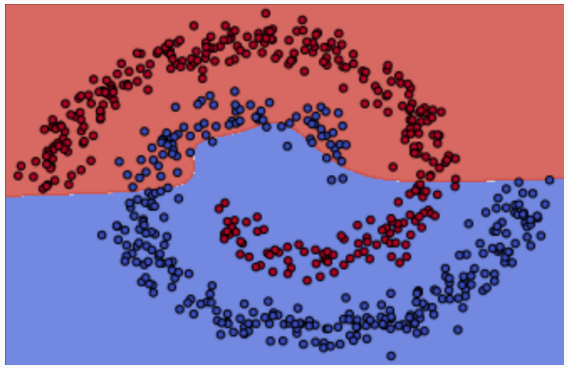
Linear Polynomial RBF

- Difference in the flexibility between the three kernels
- Only RBF can replicate the spiral shape

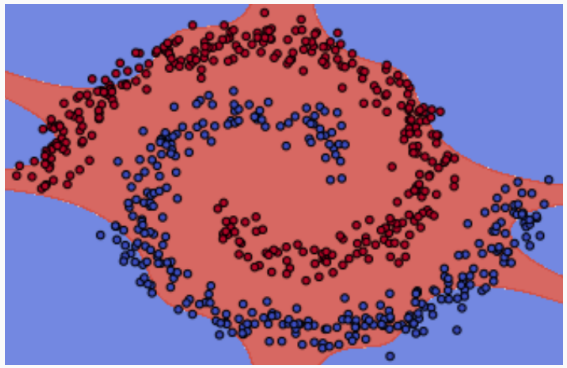
Polynomial Kernel: different degrees of freedom



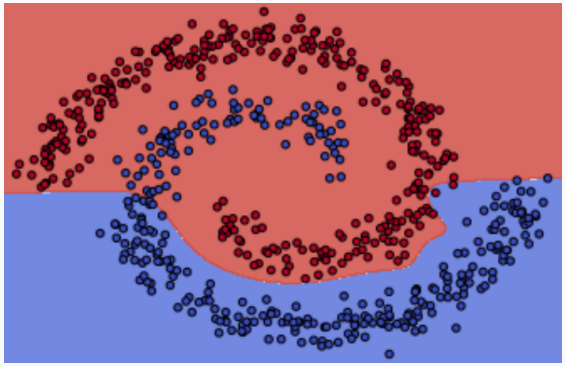
3 degr.



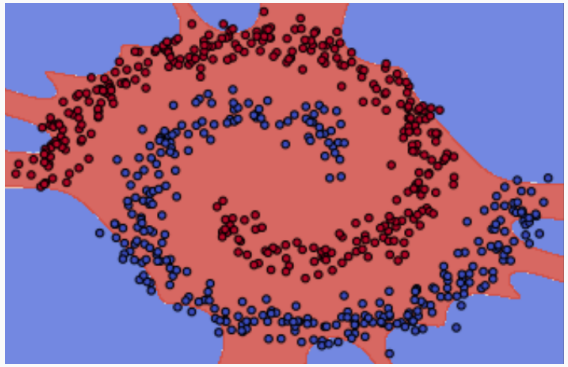
5 degr.



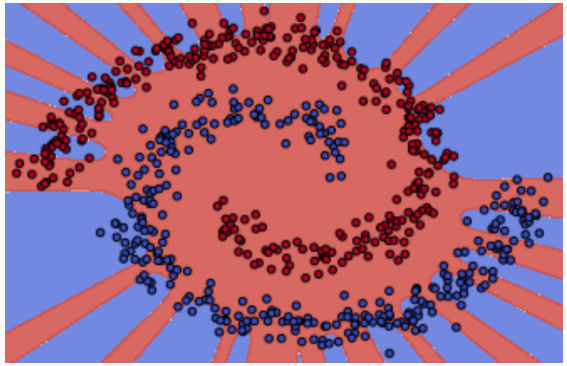
10 degr.



15 degr.

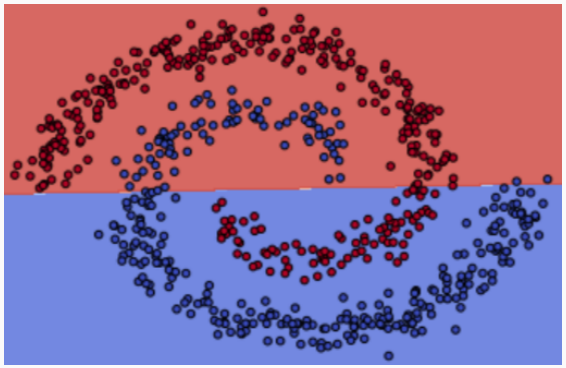


20 degr.



30 degr.

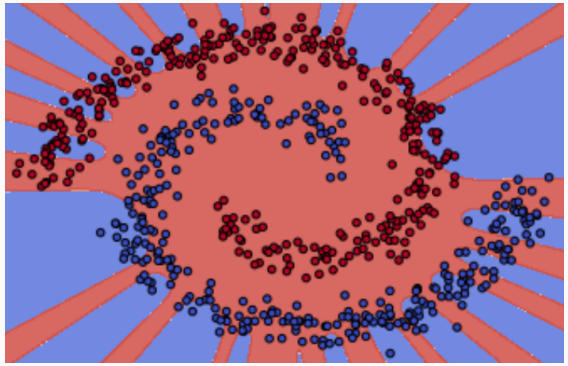
Conclusion



Linear

$E_{\text{test}}^{\text{linear}} = 0.22$

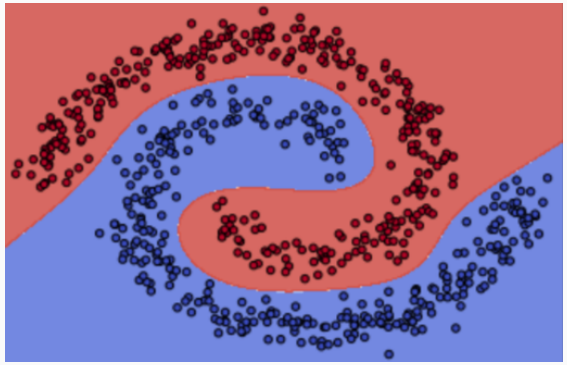
SV's: 371/800



Polynomial (30 degr.)

$E_{\text{test}}^{\text{poly-30degr}} = 0.54$

SV's: 492/800



RBF

$E_{\text{test}}^{\text{RBF}} = 0.013$

SV's: 15/800

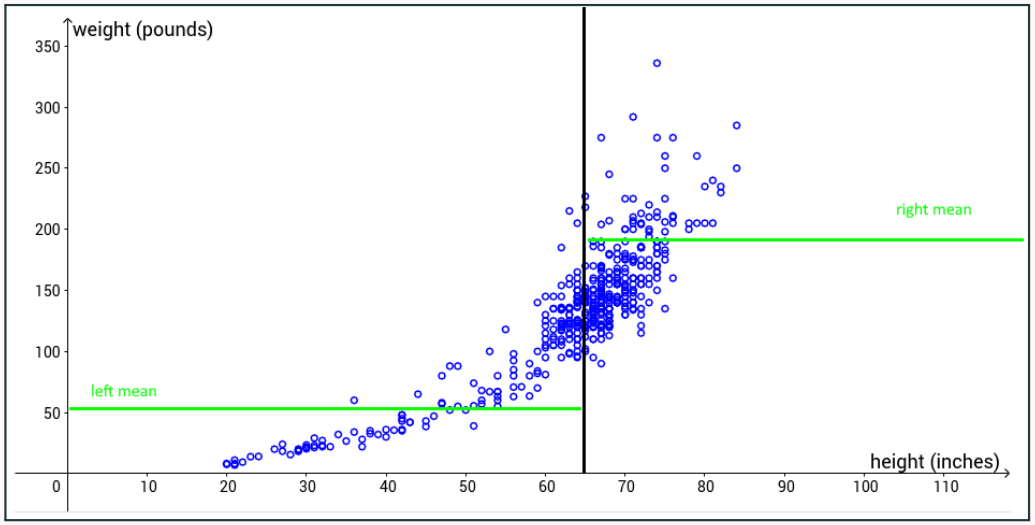
$$\mathbb{E}[E_{out}] \leq \frac{\mathbb{E}[\#\text{ of SVs}]}{\text{N} - 1} = 0.019$$

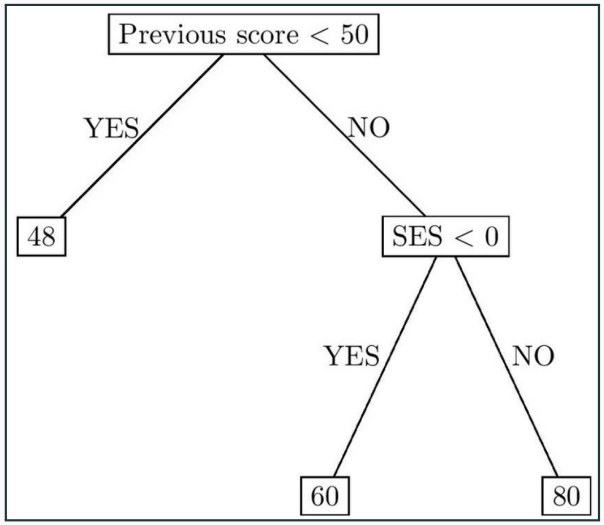
Decision tree and k-nearest neighbours regressor forecasting

How does the decision tree regressor model compare to the k-nearest neighbour regressor model in terms of in- and out-of-sample error for time series forecasting?

Decision Tree Regressor

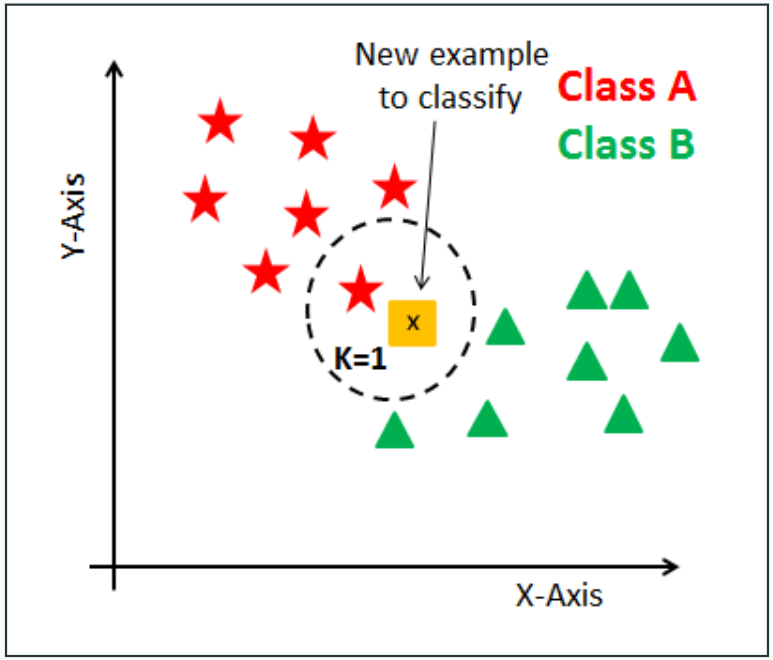
- Splits data into groups
- Training: searches best features/thresholds to split groups
- **Model = decision tree**





K-Nearest Neighbour Regressor (KNN)

- Model = data
- Prediction: mean of target values in neighborhood



Dataset

- **Given**: temperatures + humidity of specific rooms in the house
- **Objective**: predict Room Humidity in kitchen
- Measurement every 10 minutes, for 4 months

	date	Appliances	T2	T3	...	RH_2	RH_3	...	RH_1
0	2016-01-11 17:00:00	60	19.200000	19.790000	...	44.790000	44.730000	...	47.596667
1	2016-01-11 17:10:00	60	19.200000	19.790000		44.722500	44.790000		46.693333
2	2016-01-11 17:20:00	50	19.200000	19.790000	...	44.626667	44.933333	...	46.300000
3	2016-01-11 17:30:00	50	19.200000	19.790000		44.590000	45.000000		46.066667
4	2016-01-11 17:40:00	60	19.200000	19.790000	...	44.530000	45.000000	...	46.333333
...	...	...	...	...		...	...		...

Experimental setup

Root-mean-square error:

$$E = \sqrt{\frac{\sum_{n=0}^{N}(y_n - h(x_n))^2}{N}}$$

Regression score:

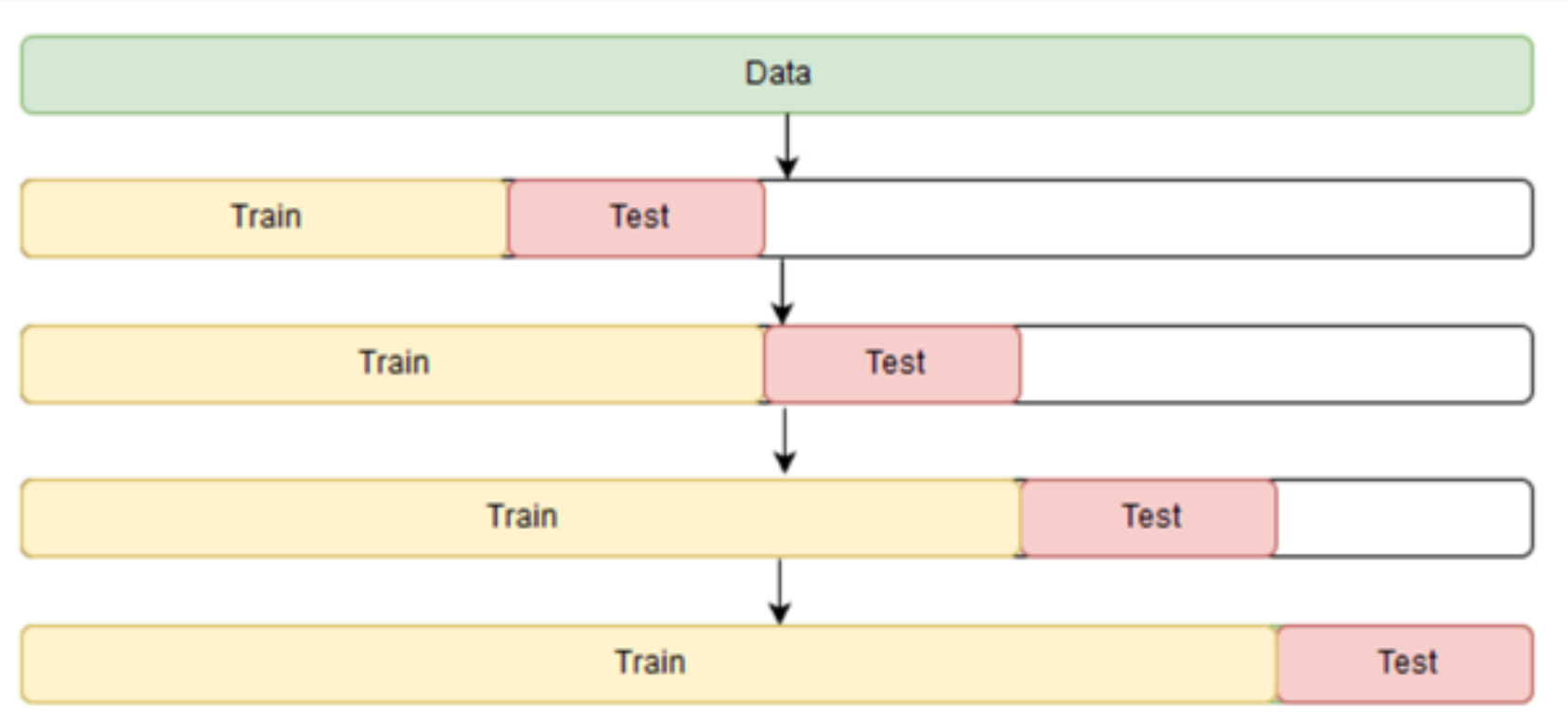
$$1 - \frac{\sum_{n=0}^{N}(y_n - h(x_n))^2}{\sum_{n=0}^{N}(y_n - \bar{y})^2}$$

Data:

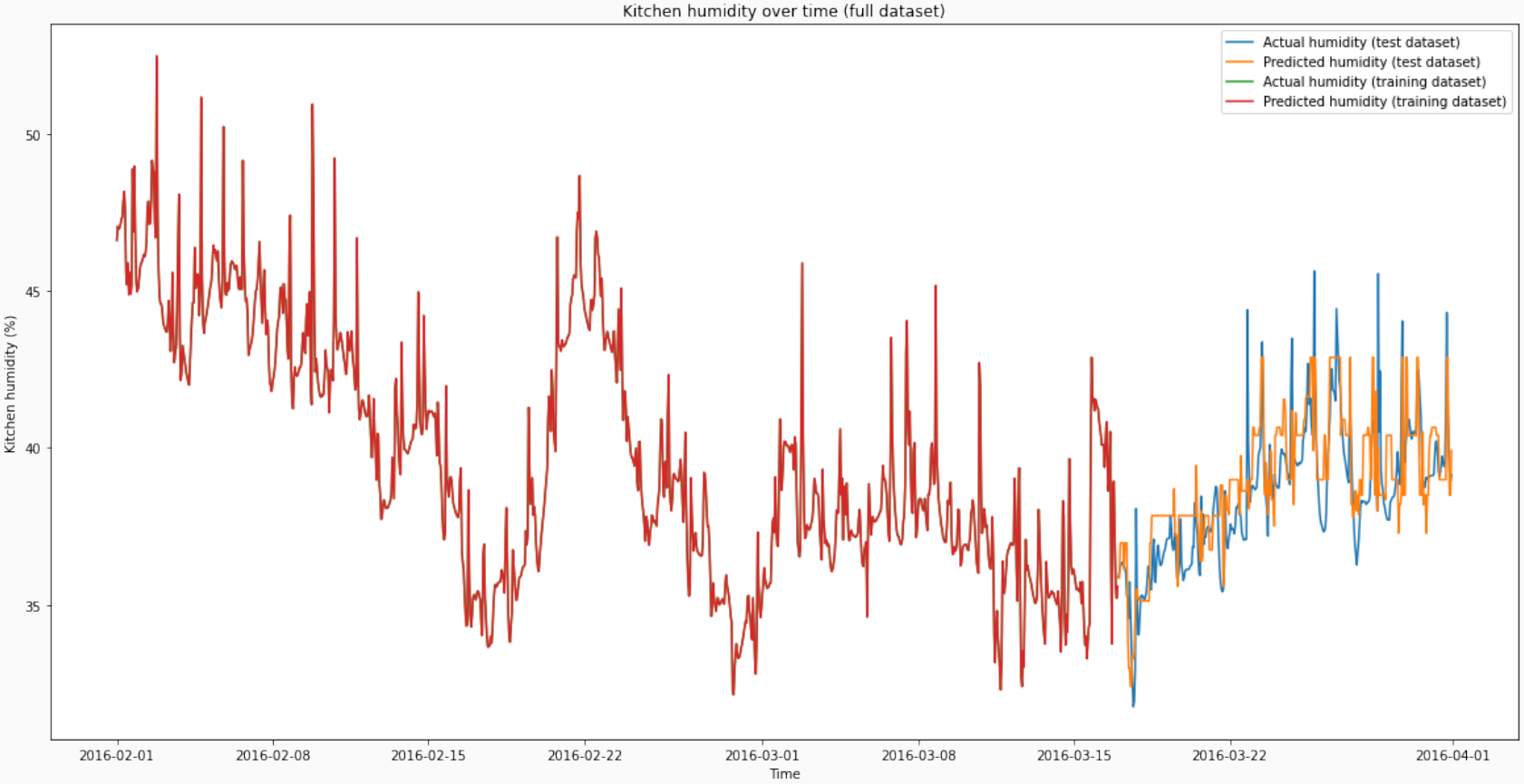
1. Split into subsequent training and testing set
2. Process normalised data

Model comparison:

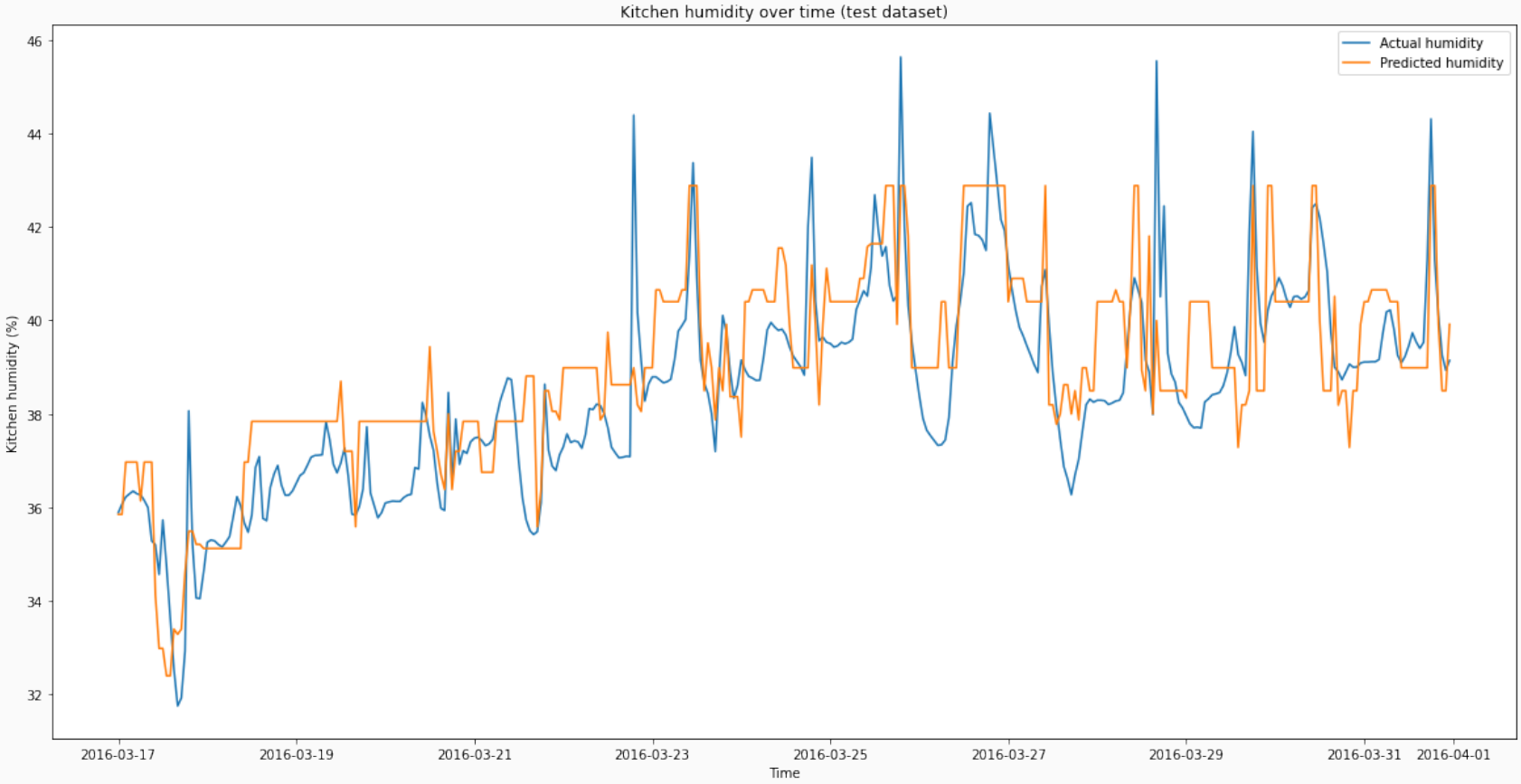
1. Baseline models
2. Equally tuned models through randomised grid search



Time series cross-validation for forecasting

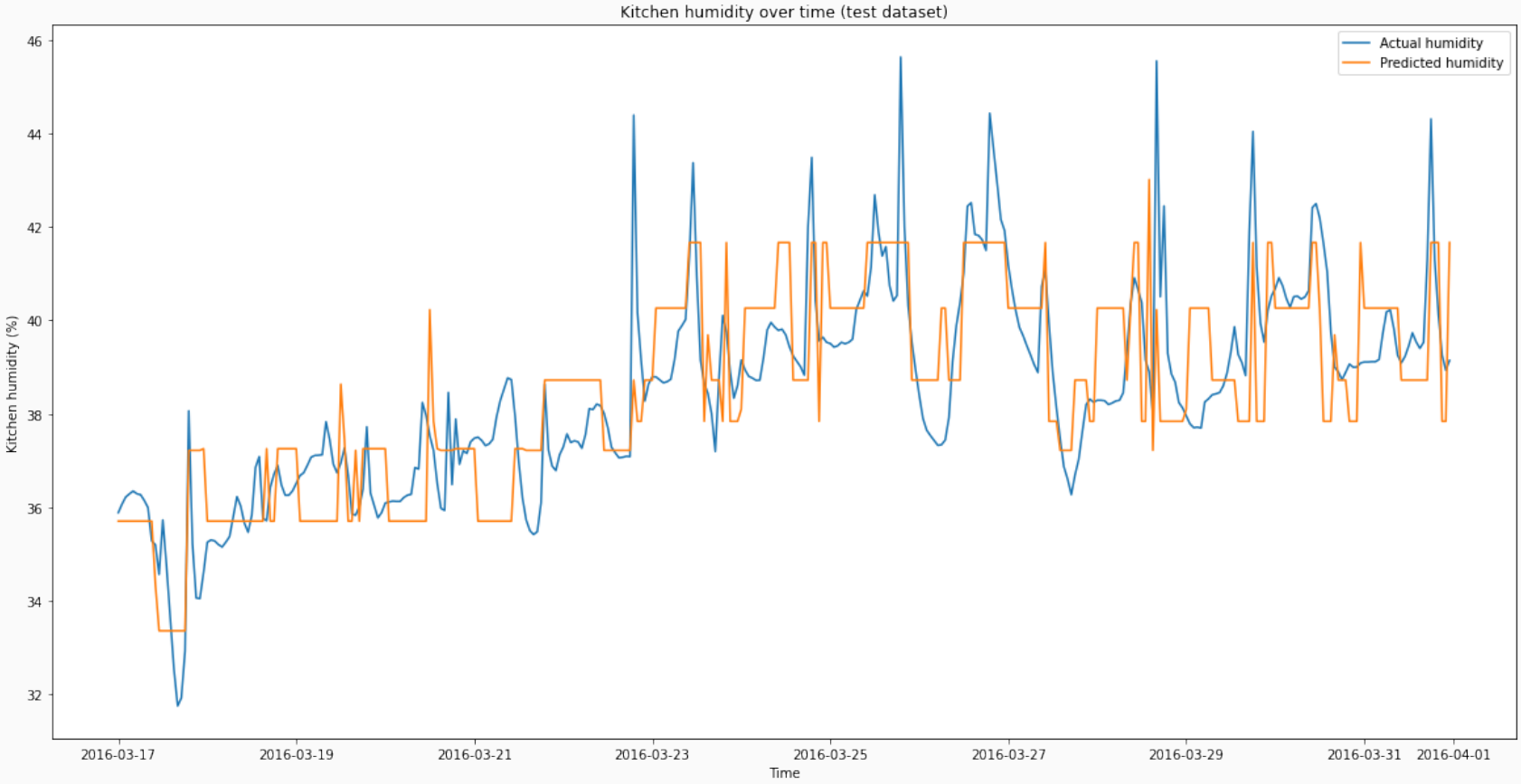


Baseline decision tree regressor



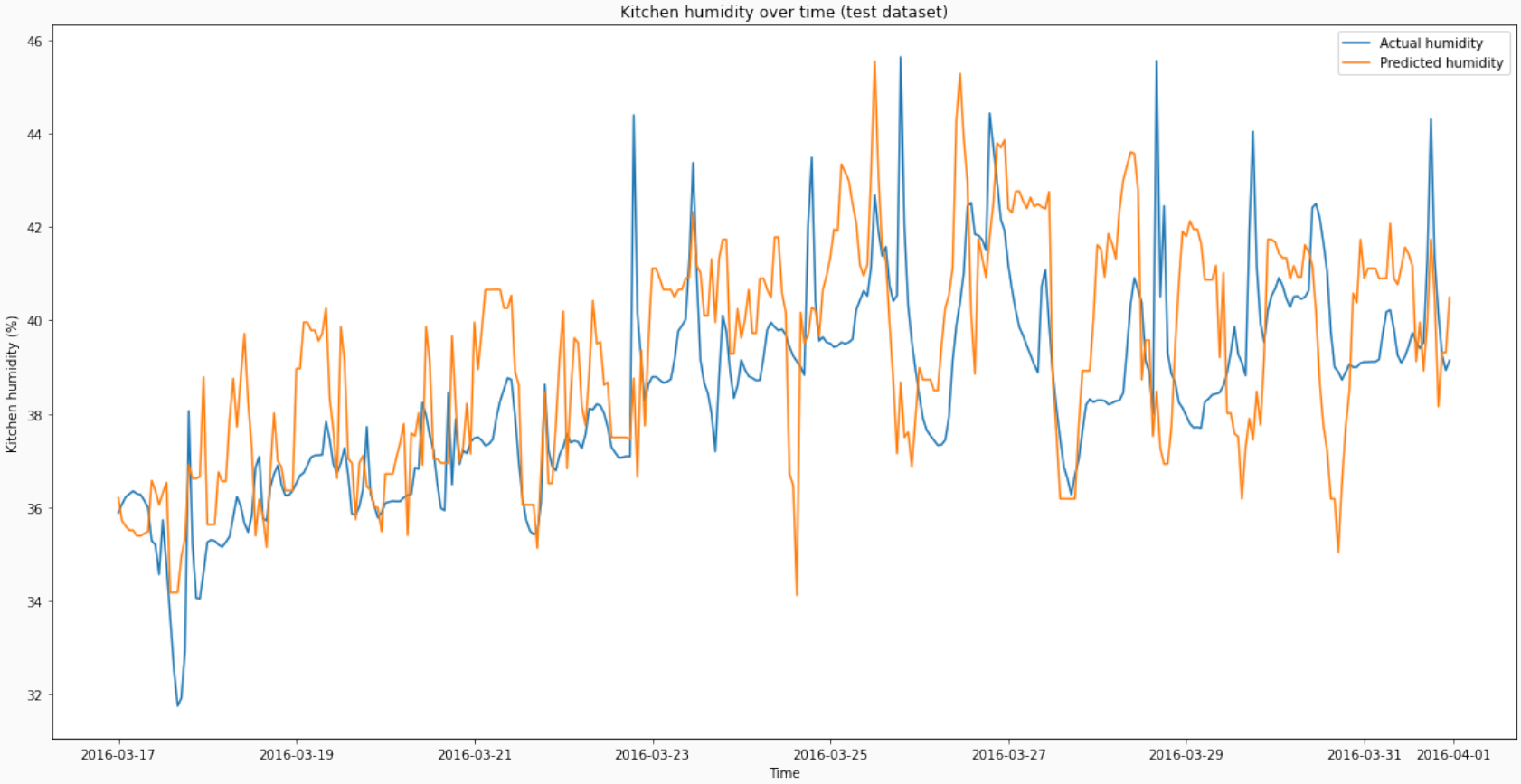
Baseline decision tree regressor

	Training	Testing
Score	1.000	0.479
RMSE	0.000	1.551



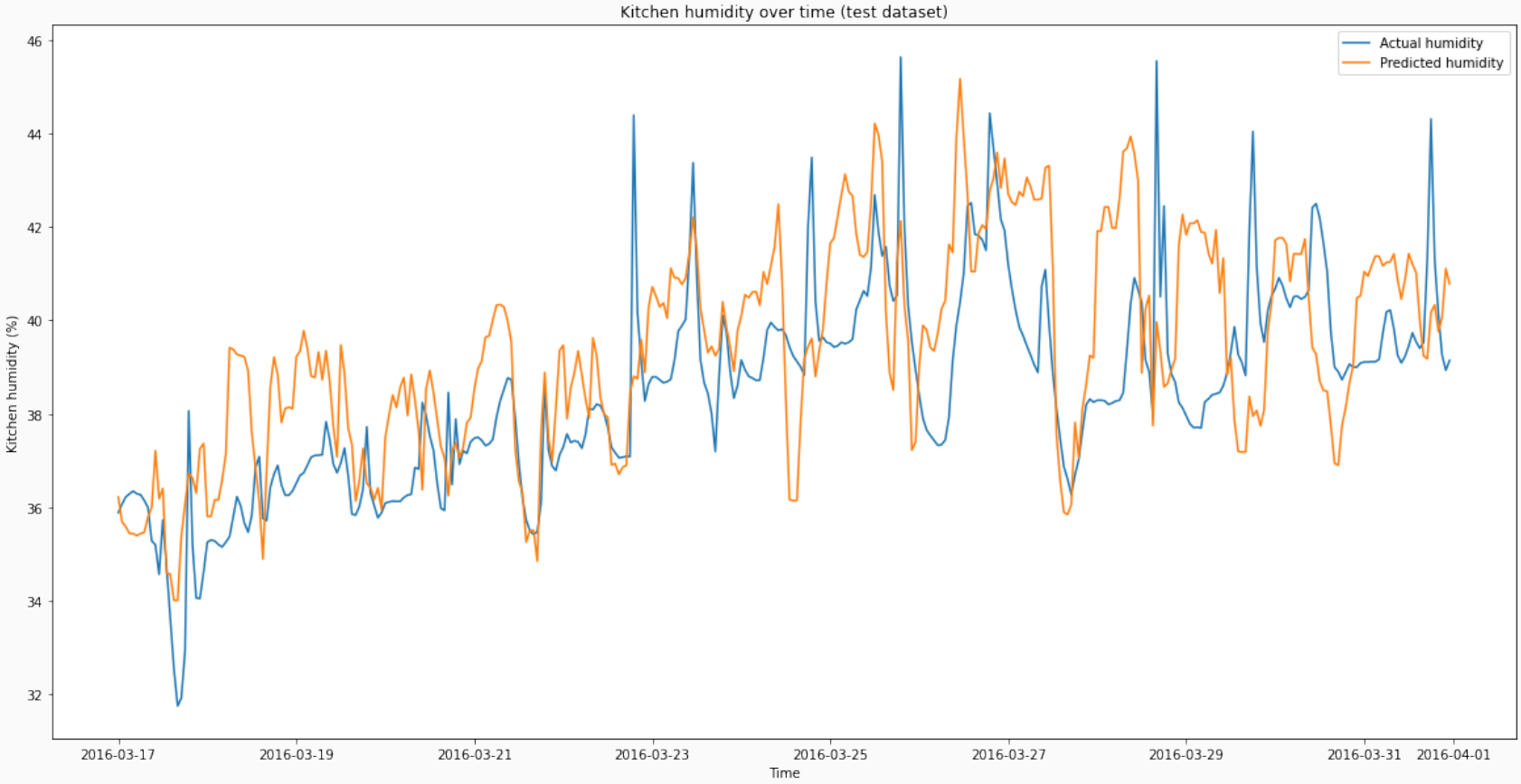
Tuned decision tree regressor

	Training	Testing
Score	0.949	0.554
RMSE	0.871	1.434



Baseline k-nearest neighbours regressor

	Training	Testing
Score	0.945	0.040
RMSE	0.908	2.105



Tuned k-nearest neighbours regressor

	Training	Testing
Score	1.000	0.089
RMSE	0.000	2.050

Conclusions

- Tuned decision tree regressor is most likely to generalise best
- Model nature is visible in predictions:
  - Decision tree predicts conservative smooth surfaces
  - K-nearest neighbours predicts noisy erratic changes resembling the training data
- Time-series forecasting is extrapolation
- Model selection can improve E_{out} by reducing or increasing model complexity
- Data preprocessing and feature selection is important and can be guided also by cross-validation