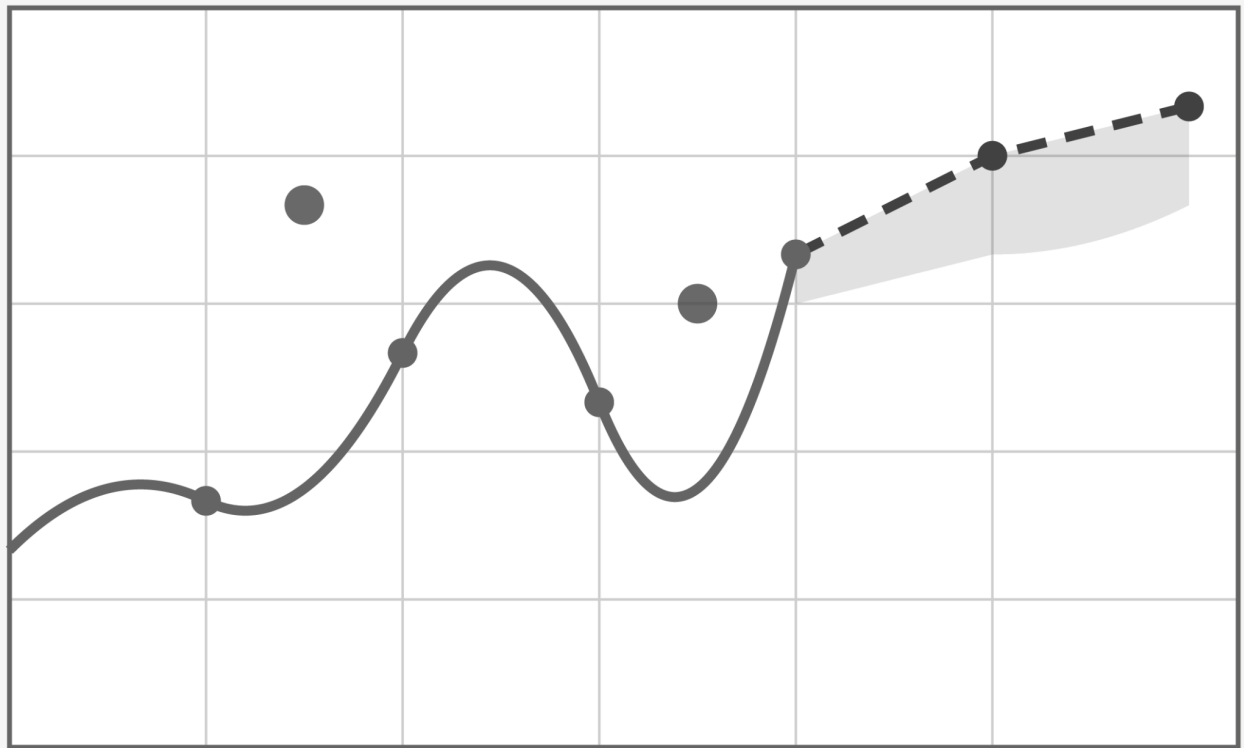


Exercises 6 (Further Aspects)

DATA.ML.450 Time Series Analysis using Machine Learning (Autumn 2025)



Anas Uddin

10.10.2025
MSc in Data Science

Exercise 1

I optimized the Random Forest classifier using `HalvingRandomSearchCV` with an 80/20 train/test split of data. I expanded the hyperparameter ranges, and by doing so, the model achieved a training accuracy of 95.8% and a testing accuracy of 93.3%. It shows good generalization without overfitting. The expanded parameter ranges and proper train/test split of data improved the performance compared to the original solution in “`HalvingRandomSearchCV_random_forest.py`”.

Original model output:

```
{'max_depth': 3, 'min_samples_split': 3, 'n_estimators': 9}
```

Training score: 0.9733333333333334 (on full data)

Improved model output:

```
{'bootstrap': True, 'max_depth': 19, 'max_features': None, 'min_samples_leaf': 8, 'min_samples_split': 3, 'n_estimators': 243}
```

Training score: 0.9583333333333334

Testing score: 0.9333333333333333

Exercise 2

I changed the SVC classifier to an `AdaBoostClassifier` and optimized its three main parameters (`n_estimators`, `learning_rate`, and `algorithm`) using Bayesian optimization.

The optimized `AdaBoost` classifier achieved training accuracy of approximately 0.9666666666666667 and testing accuracy of approximately 0.9333333333333333, with the following parameters: `{'algorithm': 'SAMME', 'learning_rate': 0.0136, 'n_estimators': 204}`.

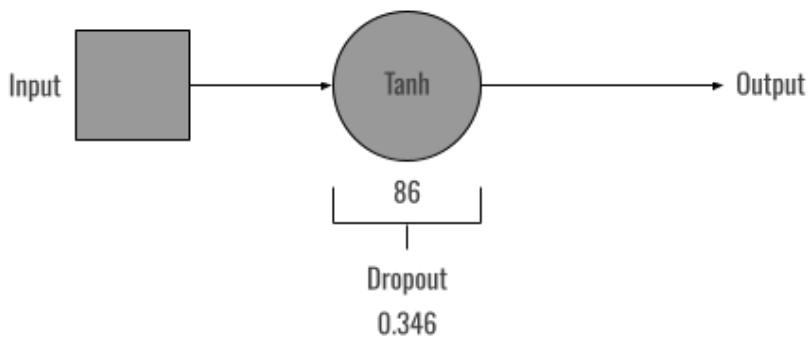
The optimized SVC had a testing accuracy of approximately 0.9736842105263158. Compared to that, `AdaBoost` performed slightly worse on unseen data. In this case, the SVC was better suited for the Iris dataset, although `AdaBoost` can be effective for many problems. We can also see from the results that `AdaBoost` generalizes well, as training and testing scores are close, but its maximum performance is lower than SVC's for this dataset.

Exercise 3

I used Optuna to optimize a small MLP on a subset of FashionMNIST. The best model achieved a validation accuracy of approximately 0.84765625. The optimized architecture consists of:

- Activation function: Tanh
- Number of layers: 1
- Hidden units in the layer: 86
- Dropout rate: 0.3464462838992882
- Optimizer: Adam with learning rate 0.004522268903646998

This shows that a single-layer network with a moderate number of neurons and Tanh activation generalizes best for this small subset. A dropout rate of around 0.35 helps to prevent overfitting. Also, Adam effectively adapts the learning rate during training.



Schematic diagram