PRNN Assignment 2

Manogna Sreenivas

SR No. 18659

Linear least squares for 2-class classification:

- Classifier $\hat{y}(X) = \text{sign}\left(W^T X\right)$ is obtained by minimising the mean squared error loss $J(W)$ which has a closed form solution W^*.

$$J(W) = \frac{1}{2}\sum_{i=1}^{n}\left(W^T X_i - y_i\right)^2$$

$$W^* = \left(A^T A\right)^{-1} A^T Y \tag{1}$$

where A is the data matrix and Y is a vector whose components are the target labels $y_i \in \{-1, 1\}$.
- This method models the class posterior probabilities as a linear function.

Logistic regression for 2-class classification:

- For class labels $y_i \in \{-1, 1\}$, the logistic regression classifier is given by $\hat{y}(X) = \text{sign}\left(W^T X\right)$ is obtained by minimising the mean squared error loss $J(W)$.

$$J(W) = \frac{1}{2}\sum_{i=1}^{n}\left(\sigma(yW^T X_i) - 1\right)^2 \tag{2}$$

- The optimal solution is obtained by iteratively minimising the above loss criterion.

Decision boundaries for 2D data:

For both the classifiers with 2D data, we can visualise the decision boundary $W^{*T}X = 0$ for the classifier as:

$$W^{*T}X = w_0^* + w_1^* x_1 + w_2^* x_2 = 0$$

$$x_2 = -\frac{1}{w_2^*}(w_0^* + w_1^* x_1)$$

1 Linear Least squares vs Logistic Regression

For each of the given datasets P1a, P1b, P1c, two linear classifiers are learnt with varying sample sizes. As we are comparing the performance of the two classifiers, in order to avoid any bias because of the samples drawn, the whole experiment is repeated for 10 runs and the average test accuracy over 10 runs, obtained on the entire test data is reported as follows:

(a) 2D Gamma distributed data

samples	Linear	Logistic
10	86.86	91.01
50	88.76	93.06
100	88.83	93.10
500	89.33	93.55
1000	89.1	93.6

(b) 2D Uniformly distributed data

samples	Linear	Logistic
10	92.12	95.99
50	92.93	98.39
100	92.9	98.71
500	92.78	99.01
1000	93.10	99.00

(c) 10D Normal distributed data

samples	Linear	Logistic
10	79.77	81.61
50	93.21	91.04
100	93.84	93.28
500	94.15	94.18
1000	94.20	94.20

Table 1: Test Accuracy on the respective datasets

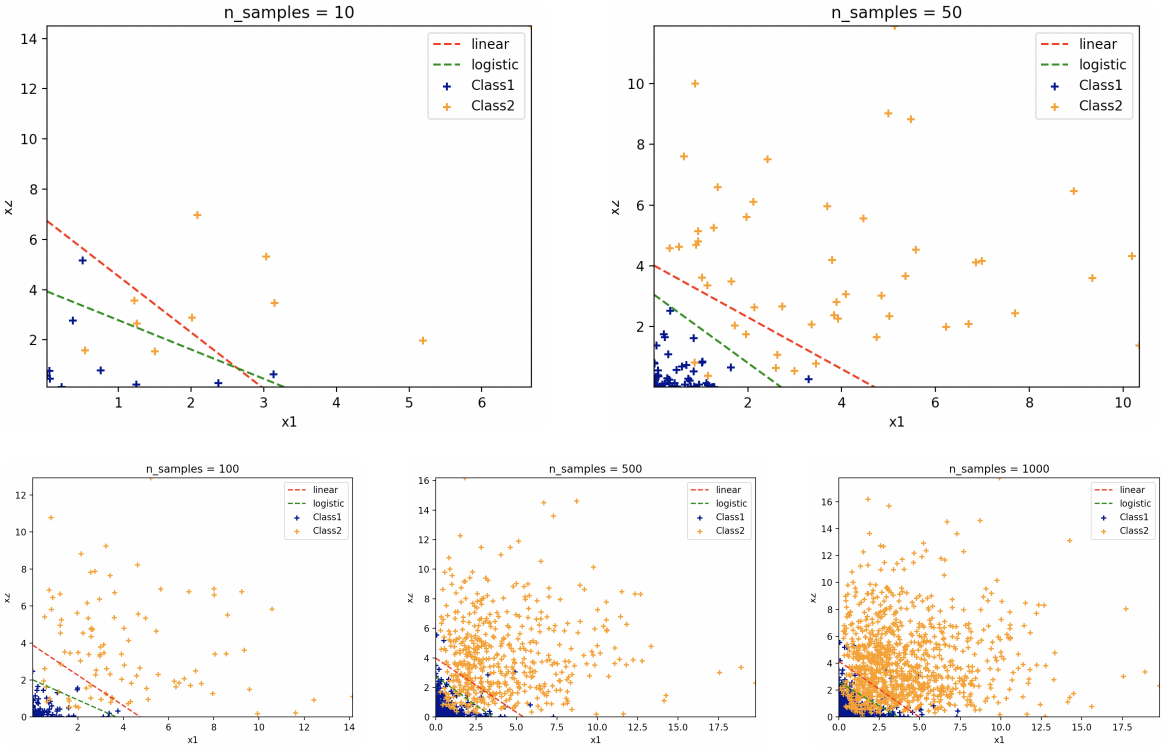


Fig 1. Decision boundaries obtained from linear least squares for 2D gamma distributed data

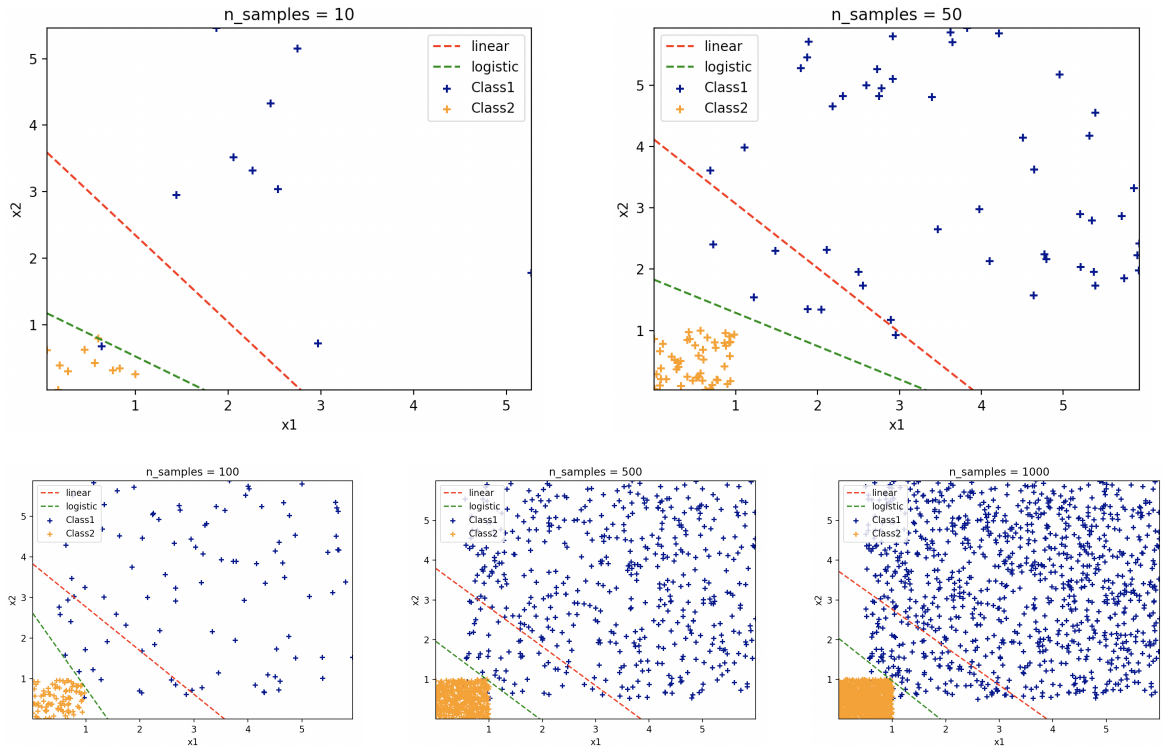


Fig 2. Decision boundaries obtained from linear least squares for 2D uniformly distributed data

The above plots show decision boundaries obtained for a set of random samples drawn for the 2D datasets given.

- Linear least squares models the posterior probabilities as a linear function. whereas logistic regression approximates it with a sigmoid function. Linear approximation obtained for the posterior probability, is the ML estimate for the discriminative model assuming labels with gaussian noise. This is not a

reasonable assumption for classification task with labels $y_i \in \{-1, 1\}$.
- Visually, we can see that linear least squares result in poor decision boundaries. In both the gamma and uniformly distributed data, we have one class widely spread with high variance and the other one dense with low variance. In the least square error loss, the class with high variance mainly drives the minimisation. The class samples coming from a dense distribution contribute less towards the loss. This happens because we are trying to associate all the samples from one class to a single label 1 or -1, ignoring the class densities the samples are drawn from. Minimising such a criterion results in moving away from the true decision boundary when the class conditional densities have different characteristics(like variance). This loss is very sensitive to the data points, adding or removing a few data points would result in very different decision boundaries. In both gamma and uniformly distributed data, the linear least squares model correctly classifies the data from the class with less variance(close to 100%) and the error is mostly due to the samples coming from the widely spread distribution, which is also evident visually.
- Logistic regression models the class posteriors as a sigmoid function and associates meaningful probabilities while classifying samples unlike linear regression. The results above show that the decision boundaries obtained for logistic regression are much better even for small sample sizes. Also they are more stable to any data outliers.
- For gamma distribution, higher samples sizes result in decent classifiers for the samples drawn. But in the case of uniformly distributed data, even for higher sample sizes the linear classifier obtained is poor. In all cases, for sufficiently large sample sizes(≥ 50), the test accuracy saturates at 93% and increasing sample size seems to have no effect.
- For 10D Normal distributed data, both linear and logistic regression perform similarly for all sample sizes. As the covariance matrix of both the class are equal and equal number of samples are drawn from the 2 classes, both classes contribute equally to the loss criterion unlike the other two datasets where the classes have different variance. So linear least squares performs well in this case.
- Bayes classifier is linear for normal densities with equal covariance matrices. Logistic regression formulation for this case is optimal w.r.t minimizing bayes error. Given a set of samples drawn, logistic regression gives the best linear classifier minimising the empirical bayes error.
- Gamma distribution belongs to the exponential family(with u(x)=x in the exponential form). For 2-class data with exponential densities, the bayes classifier is linear in u(x) and logistic regression formulation is Bayes optimal as u(x)=x, returning the linear classifier minimising the empirical bayes error.
- Bayes classifier for the given uniformly distributed data is linear($x_1 + x_2 = 2$). Logistic regression solution is bayes optimal. For large sample sizes, the decision boundary gives the optimal bayes classifier as we can see in the plots above.

2 Multiclass classification

(i) One vs Rest Classifiers:

- Here 3 classifiers are learnt using linear least squares. The given dataset has labels {1,2,3} corresponding to the three classes.
- Let C_1, C_2, C_3 be the three 2-class classifiers to be learnt. C_i denotes the classifier for Class i vs others.
- To learn classifier C_i, i $\in \{1, 2, 3\}$, the target labels y_{c_i} are obtained from the true labels as:

$$y_{c_i} = 1 \quad if \quad y_{true} = i$$

$$y_{c_i} = -1 \quad if \quad y_{true} \neq i$$

- Then the optimal linear least squares solution is obtained solving Eq 1.

(ii) 3-Class classifier using Linear Least squares:

- Here, 3 linear discriminant functions $h_i(X) = W_i^T X$ are learnt doing vector valued regression, where $W_i = \begin{bmatrix} w_0 & w_1 & w_2 & w_3 & w_4 \end{bmatrix}^T$ and $X = \begin{bmatrix} 1 & x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T$.
- The given class labels {1,2,3} are mapped to one-hot vector targets {[1,0,0], [0,1,0], [0,0,1]} respectively.
- The classifier is then given by

$$y(X) = i \quad if \quad h_i(X) \geq h_j(X) \quad \forall j$$

- The weight vectors can be obtained by regressing for the one-hot labels. This optimal weights can be obtained from the closed form solution which is of the form :

$$W^* = \left(A^T A\right)^{-1} A^T Y_{vec} \tag{3}$$

A is the data matrix with augmented features along the rows and is of dimension nx5.
Y_{vec} has the one-hot labels along the rows and is of dimension nx3.
$W^* = \begin{bmatrix} W_0^* & W_1^* & W_2^* \end{bmatrix}$ has the weights for the discriminant functions $h_i(X)$ and is of dimension 5x3.
- This linear discriminant functions here approximate the class posterior probabilities.

IRIS dataset: As the features are 4D, in order to visualise the data, 2D pairs(6 pairs possible for 4D data) of features are plotted as shown below.

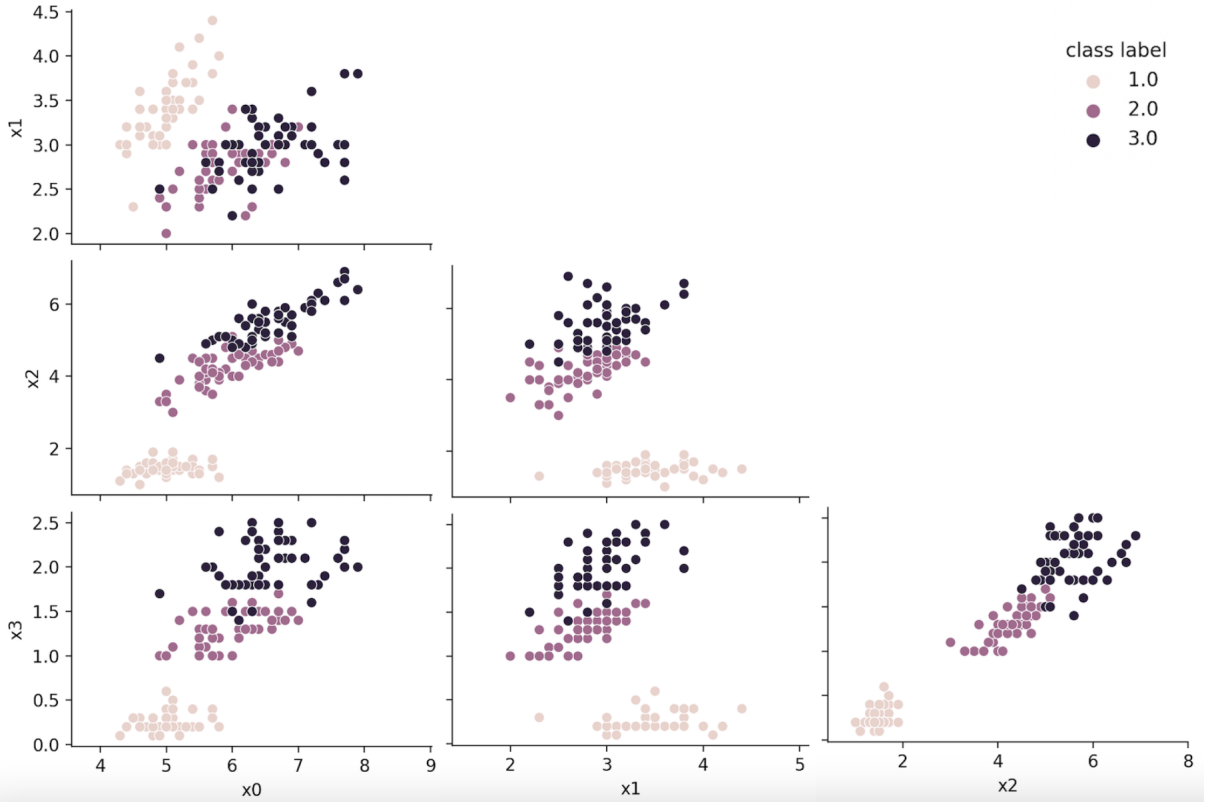


Fig 3. Plots of 2D pairs of feature components from 4D iris dataset

Experiments: The given data has 50 samples per class. Each class is split into 80:20 train-test split resulting in 40 samples per class for training and the remaining 10 samples per class for testing. The classifiers are learnt on this data split as described above. To evaluate the performance of these classifiers, this is repeated for 10 random splits and the average test accuracy for all the classifiers learnt is reported as follows:

(a) Class1 vs others

Data	Avg test accuracy
Class1	100
Others	100
Total	100

(b) Class2 vs others

Data	Avg test accuracy
Class2	55
Others	85.5
Total	75.2

(c) Class3 vs others

Data	Avg test accuracy
Class3	90
Others	95
Total	92.5

Table 2: Average test accuracy for One class classifiers over 10 runs

Data	Avg test accuracy
Class1	100
Class2	71
Class3	80
Total	83.3

Table 3: Average test accuracy for 3-class classifier over 10 runs

- The plots above show that only Class1 data is linearly separable from the other two. So, the linear classifier for the case (a)Class1 vs others, learns the decision boundary accurately resulting in a 100% test accuracy.
- For case (b)Class2 vs Others, the linear classifier classifies only 55% of the Class2 test samples correctly. The plots show that Class2 samples are kind of stuck between the other two classes. A linear classifier is expected to perform poorly here as the plots show the no linear decision boundary can put apart class2 samples from the other two classes.
- For case (c)Class3 vs Others, the linear classifier works quite well. Though class 3 is not linearly separable from class2, the plots show that a linear classifier would be reasonable choice to put class3 samples apart from the other two classes. This classifier correctly classifies 90% of the class3 data. The 10% samples would be misclassified as class2 and class3 samples are not linearly separable.
- Using such 3 one vs rest classifiers for 3-class classification would result in ambiguous regions where multiple classes could be assigned.
- In the second method, where we learn a 3-class classifier, the linear discriminants approximate class posteriors. A test sample is assigned to the class with highest posterior probability. This is a better choice for multi class classification as it does not result in any ambiguous regions.
- The 3-class linear classifier correctly classifies all the test samples from class1 as expected because its linearly separable from the other two classes. 71% of the class2 test samples are correctly classified. Class2 test accuracy is the least with 71% as the data is kind of present between the other 2 classes and some samples could be wrongly classified to either class1 or class3. Class3 samples could be misclassified only as class2 due to the overlap and results in accuracy of 80%.

3 Linear vs Logistic on german credit data

The given data is randomly split to 80:20 train-test. The two classes labeled as {1,2} are mapped to {1,-1} and are referred as Class1 and Class2 respectively. Linear and logistic regression classifiers are then learnt using Eq (1) and (2). This repeated over 10 random splits and the average test accuracy is reported as follows:

Data	Linear	Logistic
Class1	91.12	91.241
Class2	41.27	44.44
Total	75.45	76.50

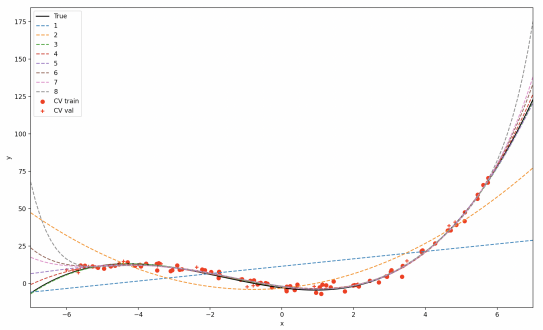
Table 4: Test accuracy averaged over 10 runs

- The given data has 24D features associated with some financial attributes. Some of these attributes take a fixed no. of discrete values while some take continuous values. The distribution of each feature component in this data is unknown and would have different characteristics. They would have different mean and variance.
- This is an unbalanced dataset with 70% of the samples coming from Class1 and the remaining 30% from Class2. Both the classifiers classify 91% of theclass1 test samples correctly as the models optimize to classify class1 samples better because they contribute more to the loss. - Performing preprocessing like min-max normalization results in the same classifier as this is a linear operation and the discriminative model learnt would effectively be the same and hence gives the same accuracy for the same samples drawn.
- Mean-variance normalization on all features would be bad as some features are categorical as the normal distribution is not representative of the true features.

4 Linear Regression for Polynomial curve fitting

The objective here is to fit an n^{th} order polynomial for the given data and decide on the best possible degree. For a data point x, polynomial features of degree n are computed as $X = \begin{bmatrix} 1 & x & x^2 & ... & x^n \end{bmatrix}^T$. The linear coefficient vector $W = \begin{bmatrix} w_0 & w_1 & w_2 & ... & w_n \end{bmatrix}^T$ to fit n_{th} degree polynomial:

$$f_n(x) = \sum_{i=0}^{n} w_i x^i = W^T X$$



(a) Polynomials fit for varying degrees

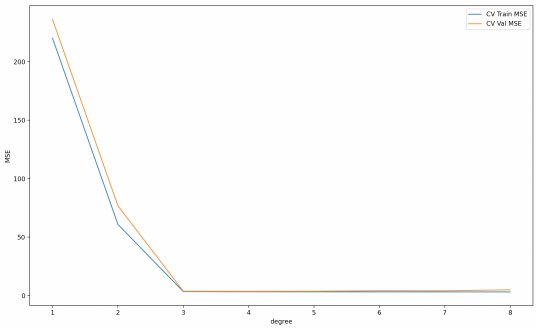


Fig 4. Average MSE for 5-fold cross validation

Figure 5: Linear Regression for Polynomial Curve fitting

degree	Avg train MSE	Avg Val MSE
1	213.042	309.360
2	60.405	85.981
3	3.406	3.699
4	3.296	3.675
5	3.251	3.589
6	3.206	3.635
7	3.199	3.680
8	3.174	3.690

Table 5: MSE averaged over 5-fold cross validation splits

The optimal least squares solution W^* is then obtained solving Eq 1.
- In order to decide on the degree of the polynomial, K-fold cross-validation is done and the average MSE obtained over the K validation splits(K=5 here) is used as the metric here. Evaluating over K random splits also results in similar results.
- Fig.5a shows the polynomial curves fit for varying degrees for one of the train-val during cross-validation. The linear and quadratic curves fit have large error w.r.t the true polynomial. For degrees $\geq$3, all the curves fit the data well and overlap with the true polynomial in the range [-6,6] from which the data is sampled. The Average MSE plot obtained by averaging the MSE over 5-fold cross-validations quantifies this observation. The average MSE is high for linear and quadratic fits but remains almost same for degrees¿=3 over both the training and validation splits. - This observation suggests that simply treating the best degree to be the one with least K-fold avg validation MSE might not be the best criterion. As we randomly split the given data into K mutually exclusive sets, this randomness can result in slightly varying MSE and the algorithm could give us any of 3,4,5,6,7,8 for different data splits.
- We can see that the average MSE over training splits consistently decrease for increasing degree(5). This would be because the polynomial functions tend to model the noise in the target variables y for degrees greater than the true degree. Though the training error decreases, this is undesirable as a more complex function is fit and would not be representative of the true data, resulting in increasing error on the validation or unknown test data.
- Fig. 5(a) shows the polynomials extrapolated to the range [-7,7] but the given data comes from the range [-6,6]. Choosing polynomials of degrees 3,4,...,8 would work perfectly well in the range [-6,6] as the results show, if the test data could come from a domain outside [-6,6], only the cubic polynomial is

close to the true polynomial and the error would significantly increase for increasing degrees beyond 3 as we can see the functions deviate a lot from the true polynomial in the range [-7,-6] and [6,7].
The following algorithm was used to decide on the degree of the polynomial.

Algorithm to obtain the best degree:

for k **in** $\{1, 2, \dots K\}$:
Get (D^k_{train}, D^k_{val}), the k^{th} training and validation split
for j **in** $\{1, 2, \dots 8\}$:
- Fit j^{th} degree polynomial on D^k_{train} using linear least squares
- Compute MSE on D^k_{train}, D^k_{val}

$$\text{TrainMSE}^k_j = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \left(y_i - f^j\left(x_i\right)^2 \right) \qquad (x_i, y_i) \in D^k_{\text{train}}$$

$$\text{ValMSE}^k_j = \frac{1}{N_{val}} \sum_{i=1}^{N_{val}} \left(y_i - f^j\left(x_i\right)^2 \right) \qquad (x_i, y_i) \in D^k_{\text{val}}$$

end
end

$$\frac{1}{K} \sum_{k=1}^{K} ValMSE^k_j$$

(Criterion 1) Return the degree with least average MSE over K-fold validation as:

$$d^* = \underset{j}{\text{argmin}} \frac{1}{K} \sum_{k=1}^{K} ValMSE^k_j$$

(OR)

(Criterion 2) Return the least degree such that the average MSE over K-fold validation is below a certain error threshold ϵ

If this data came from a real scenario and we knew the domain would be limited to [-6,6], then using Criterion 1 would be fine as all degrees above 3 are good fits. But if it were to come from a range beyond [-6,6], then the Criterion 1 returns the best degree to be **5** and that would perform poorly while predicting on the test data. As the errors are very close for degrees $\geq$3, this could return a different degree for another random split and hence isn't a very good criterion. An alternative better criterion would be to return the least degree such that the average MSE over K-fold validation is less than a threshold ϵ. Using Criterion 2 returns degree **3** consistently for different random splits. The threshold ϵ can be set based on the noise variance of the target data if known.