Machine learning and statistical learning

Error measure, Overfitting

S. Hué

1 Context and objectives

In this exercise, you will be asked to predict a continuous variable using a machine learning algorithm by varying the values of a hyperparameter of this algorithm. The objective is to observe the evolution of the measurement error on the training sample as well as on the test sample, according to the flexibility of the model.
The data used in this exercise give the selling price of cars (in rupees) and provide details on the caracteristics of the car. These were downloaded from Kaggle[1] and cleaned (you can download the raw dataset on Kaggle and clean it yourself if you prefer).
The columns of the dataset are the following:

- `name`: name of the car
- `year`: year in which the car was bought
- `selling_price`: price the owner wants to sell the car at (in thousand rupees)
- `km_driven`: distance completed by the car in km
- `fuel`: fuel type of the car
- `seller_type`: tells if car is sold by individual or dealer
- `transmission`: Gear transmission of the car (Automatic/Manual)
- `owner`: number of previous owners
- `mileage`: mileage of the car
- `engine`: engine capacity of the car
- `max_power`: max power of engine
- `torque`: torque of the car
- `seats`: number of seats in the car
- `sample`: whether the observation belongs to the training or testing set

[1] `https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho`

2 First steps: preparing the data

1. From AMeTICE, download the cars dataset (cars.csv).
2. Load the CSV file into `R` or Python.
3. Randomly split your dataset into two parts:
   a) a train set that will contain 80% of the observations
   b) a test set that will contain the remaining 20%.
4. From the train set, create two datasets:
   - a training set that will contain 80% of the observations from the train set
   - a validation set that waill contain the remining 20%.

3 Getting to know the data: descriptive statistics

1. Compute some summary statistics for the whole dataset (and comment the outputs):
   - for numerical variables: mean, minimum, maximum, quartiles
   - for categorical variables: for each category: proportion of observations; mean, minimum, maximum, quartiles of the target variable (`selling_price`)
2. Create graphs to show the relationship between the target variable and the explanatory variables. Comment on these.

4 Standardization

The optimization of many machine learning algorithms works better when the data are scaled to a standard range. In this exercise, you will scale your data using the min-max scaler. The values x (both for the target variable and the features) will be scaled using the following formula:

$$\frac{x - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})},$$

where $\min(\mathbf{x})$ and $\min(\mathbf{x})$ are computed on the training set.
The standardization needs to be made on both the training and the validation sets.

- Scale the target variable on the training and the validation sets, using the min-max scaler (using the min and max values computed on the training set).
- Do the same for each feature.

If you use `R`, you can use the `preProcess()` function from caret on the train set to prepare the scaler, and then on the `predict()` function to apply the scaling.
If you use `python`, you can use the `MinMaxScaler()` function from the `preprocessing` module of the `sklearn` library to create a scaler, then you can use the `fit()` function on the training set to fit the scaler. The transform function can then be used to apply the scaling.
Note: this particular scaler will be influenced by potential outliers.

5 Estimations

You will train two different models: a random forest and a SVM. You will understand in more details how both algorithm work in the subsequent lessons. For now, all you need to understand is that you can vary some hyperparameters for both models. Among the different values that you will try for these hyperparameters, you need to select which provides the best fit, on the validation set.

5.1 A first model: Random forest

You will first try to predict the selling price of the cars using a random forest. You will make 3 hyperparameters vary.

1. Using a random forest algorithm, predict the selling price of the car in the training set. Use the following variables: `year`, `fuel`, `km_driven`.

   If you use `R`, you can use the `randomForest` function from {`randomForest`}. If you use Python, you can use the `RandomForestRegressor` function from `sklearn.ensemble`. You will use the following hyperparameters for the algorithm:

Argument	R	Python	Value
Number of tree	`ntree`	`n_estimators`	20
Number of variables randomly sampled as candidates at each split	`mtry`	`max_features`	3
Minimum size of terminal nodes	`nodesize`	`min_samples_leaf`	10

2. Compute the mean squared error both for the training and the validation datasets. Compare them with each other.

3. Now, using a loop, make the minimum size of terminal nodes vary as follows: 10, 20, 30, ..., 100. At each iteration, compute the mean squared error (and store it) for both samples.

4. On a graph, plot the mean squared error as a function of cost, for both samples (one curve for each sample). Comment.

5. What is the value for your hyperparameter (the cost here, as you only consider a linear kernel in this exercise) that provide the best fit with regard to the MSE?

5.2 A second model: SVM

Now you will train another machine learning model: a support vector machine. You will only consider a linear kernel, and make only one hyperparameter vary (called the cost parameter; low values for this hyperparameter lead to a smoother decision surface)

1. Using a Support Vector Machine (SVM) with a linear kernel, predict the selling price of the car in the training set. Use the following variables: `year`, `fuel`, `km_driven`.

   If you use `R`, you can use the `svm` function from {`e1071`}. If you use Python, you can use the `svm.SVR` function from `sklearn`. You will use the following hyperparameters for the algorithm:

Argument	R	Python	Value
Kernel	`kernel`	`n_estimators`	"linear"
Cost	`cost`	`C`	10

2. Compute the mean squared error both for the training and the validation datasets. Compare them with each other.

3. Now, using a loop, make the cost vary:

   a) with R: `10^seq(3, -2, length = 50)`

   b) with python: `10**np.linspace(start=3, stop=-2, num=50)`

   At each iteration, compute the mean squared error (and store it) for both samples.

4. On a graph, plot the mean squared error as a function of the node size, for both samples (one curve for each sample). Comment.

5. What are the values for your hyperparameters (the number of trees, the number of variables randomly sampled as candidates at each split, and the minimum size of terminal nodes) that provide the best fit with regard to the MSE?

5.3 Select your model

Now that you have selected, both for the random forest and for the SVM the values of the hyperparameters that produce the best fits on the validation set, you can select the model that gives the best results on the test set.

1. Compute the MSE on the test set for the random forest with the selected values for the hyperparameters.

2. Do the same for your best SVM.

3. Which model gives the best results on the test set? Comment.