University of Applied Sciences and Arts Northwestern Switzerland
School of Business

DS-05 Lab: Subset Selection & Regularization

Dr. Gwendolin Wilke

Task 1: Apply Best Subset Selection to the `Credit` Data Set

- Explore the `Credit` data set from the `ISLR` library.
- Apply Best Subset Selection for Linear Regression on the `Credit` data set.
- The response variable is `balance`.

1. **Load the ISLR library.**

2. **Explore the `Credit` data set, using the „usual suspects“ for data exploration:**
   - Use `help(Credit)` to see the data dictionary.
   - Make a copy of the original data set : `Cr1 <- Credit`.
     Remark: As a general rule, always work with a copy, so that the original stays unchanged.
   - Use `View(), str(), summary(), plot() pairs(), corrplot(), hist(), sum(is.na())` etc. to explore the dataset.
   - What may be the most influencial predictors for the response variable `balance`? Are there collinearities among the predictors? Missing data? What else do you notice?

3. **Perform a manual feature pre-selection:**
   - Remove the `ID` column.
   - Remove collinear predictors, if applicable.
   - Use `lm()` and `vif()` to check for hidden collinearities, and remove them if applicable.

4. Apply the first 2 steps of the Best Subset Selection Algorithm using `regsubsets()`:

- Apply `regsubsets()` from the library `leaps` as follows:

```
sets <- regsubsets(Balance ~ ., Cr1, nvmax = <nvmax>)
```

Here, you need to set the parameter `nvmax`. It is the maximum number of predictors you want to examine in Best Subset Selection.
We want to try out *all* predictors we have, so we could use `dim(Cr1)` or `str(Cr1)` to check how many variables we have in the dataset.
Yet, `regsubsets()` works with the model matrix `.` Thus, instead of checking the dimension of the *dataset*, we need to check the dimension of the *model matrix*:

```
dim(model.matrix(Balance ~ ., Cr1))
```

- After applying `regsubsets()`, you can inspect the result by applying `summary()` to the output:

```
summary(sets)
```

Remember from the lecture: The first 2 steps of the Best Subset Selection Algorithm give you the best perfroming model *per model size.*
I.e., for each model size (number of predictors) `k = 1,2,…,<nvmax>`, you get the best model of size `k`.

Therefore `summary(sets)` outputs a list of `<nvmax>` „best" models, ordered by size: You find the best model with 1 predictor in the top row, followed by the best model with 2 predictors in the second row, etc. The stars indicate which predictors are included in the corresponding model.

Algorithm 6.1 *Best subset selection*

1. Let $\mathcal{M}_0$ denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \ldots p$:
   (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here *best* is defined as having the smallest RSS, or equivalently largest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

5. Apply step 3 of the Best Subset Selection Algorithm using the error metrics AdjR2, BIC and CP

- `regsubsets()` uses RSS to find the best model per model size `k`. Yet, we cannot use RSS to compare models of *different* sizes, because RSS always decreases when we add predictors.
- Therefore we need to use a different metric for the comparison, e.g., AdjR2, BIC or CP.
- Conveniently, these metrics are stored in the output of `regsubsets()`. You can list them per model, e.g., like this:

```
sets_summary <- summary(sets)
data.frame("adjr2"=sets_summary$adjr2, "cp"=sets_summary$cp, "bic"=sets_summary$bic)
```

- For each of the 3 metrics, which model size wins?
  - Use `which.max()` and `which.min()` to find the winners quickly. E.g., `which.max(sets_summary$adjr2)`.
  - Remember that higher AdjR2 are better, while lower CP and BIC are better.
- To inspect the models visually, use

```
plot(sets, scale="adjr2")
plot(sets, scale="Cp")
plot(sets, scale="bic")
```

Each plotted table shows one row per model. The models are sorted in descending order of performance from top to bottom. I.e., the best model is in the top row, the worst model is in the bottom row. From the plots, we can easily see which predictors are included in each model. The grey scale values represent the values of the performance metrics.

Algorithm 6.1 *Best subset selection*

1. Let $\mathcal{M}_0$ denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \ldots p$:
   (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here *best* is defined as having the smallest RSS, or equivalently largest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

6. Apply step 3 of the Best Subset Selection Algorithm using cross validation error

- In our solution in the last step, we used the error metrics AdjR2, BIC or CP to determine the overall best model.
- Notice that these metrics are stored in the `summary()` of `regsubsets()`, and that we applied `regsubsets()` to the whole dataset.
  Thus, these metrics are training error metrics!
- Using training error metrics is actually not such a good idea, since models can overfit on the training set. If we would want to have a more trustworthy comparison, we should *not* use the values stored the summary of `regsubsets()`, but instead apply the validation set approach and calculate AdjR2, Cp and BIC on the test set for each of the models manually.
- An even better alternative is to use the cross-validation error instead of AdjR2, Cp and BIC. As you know, the CV error is a robust version of the test error, and thus even more reliable.
- Your task is now to compare the all the „best models" per model size from `regsubsets()` using cross validation:
  - List again all the best models using `summary(sets)`.
  - Fit each of them using `glm()`.
  - Calculate the cross validation error for each of them using `cv.glm()`.
  - Find the one with the smallest cv error.

Which model is the winner now?

Task 2: Apply Stepwise Selection

- Again apply automated feature selection for Linear Regression on the `Credit` data set, where `balance` is the response variable.
- This time use Forward & Backward Stepwise Selection instead of Best Subset Selection.

- With *Best Subset Selection* we can be sure to find the *best* feature subset for our task. Yet, it comes with the disadvantage of high computational effort.
- If it runs too long, or if our machine runs out of memory, we can choose to use a heuristic search strategy, e.g., *Stepwise Forward Selection* or *Stepwise Backward Selection*.
- To apply them, we use the same approach as in Task 1, but we set the parameter `method` in `regsubsets()` to `"forward"` or „`backward"`:

```
regsubsets(Balance ~ ., Cr1, nvmax = <nvmax>, method = "forward")
regsubsets(Balance ~ ., Cr1, nvmax = <nvmax>, method = "backward")
```

Try it out at home!

Task 3: Apply Ridge and Lasso Regression

- Again apply automated feature selection for Linear Regression on the `Credit` data set, where `balance` is the response variable.
- This time apply Ridge Regression and Lasso Regression.
- **We do this together in class.** (For explanations see comments in the R-code.)