MS&E 349: Homework 4

Markus Pelger

Due 24pm, March 17th 2021. Submit on Canvas

Please submit one homework assignment per group. The homework solution should be submitted online to Canvas. Please indicate clearly the names of all group members. I prefer that solutions are typed in Latex, but it is also fine to submit scanned copies of handwritten solutions. Include the commented code in an appendix section. Please also submit the executable and commented code.

Question 1 Machine Learning in Finance

In this question, we are going to study different machine learning methods to predict monthly S&P 500 index returns with financial variables. (Welch and Goyal(2007)) examines the predictability of these characteristics on the equity premium. The data is available on Canvas. All methods are implemented in the *scikit* package in Python, i.e. you just need to run the commands. Our goal is to estimate the conditional mean of the index return. We assume that the conditional mean is a function of parameters z_t:

$$r_{t+1} = \mathbb{E}_t[r_{t+1}] + \varepsilon_{t+1}$$

where

$$\mathbb{E}_t[r_{t+1}] = g^*(z_t)$$

and $z_t \in \mathbb{R}^p$ is a p-dimensional predictor vector.

Sample splitting and tuning: The following machine learning methods rely on a choice of hyperparameters to control model complexity and avoid overfitting. Hyperparameters include, for example, the penalization parameters in LASSO and elastic net, the number of iterated trees in boosting, the number of random trees in a forest, and the depth of the trees. In most cases, there is little theoretical guidance for how to "tune" hyperparameters for optimized out-of-sample performance.

We follow the most common approach in the literature and select tuning parameters adaptively from the data in a validation sample. In particular, we divide our sample into three disjoint time periods that maintain the temporal ordering of the data.

1. The first, or training subsample is used to estimate the model subject to a specific set of tuning parameter values.

2. The second, or validation subsample is used for tuning the hyperparameters.

3. The third or testing subsample is used for neither estimation nor tuning, is truly out-of-sample and thus is used to evaluate a method's predictive performance.

   In this problem, we are going to use the data from January 1927 to January 1985 as the training set, the data from February 1985 to January 1997 as the validation set and and the data from January 1997 to Nov 2017 as the test set. We are going to use two measures to evaluate the performance of the method

   (a) Mean of squared errors (MSE):

   $$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i),$$

   where $\hat{y}_i$ is the predicted y_i.

(b) R-squared R^2:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2},$$

where the unconditional mean $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is only estimated on the training data and used on the training, validation and test data set.

Report the two measures for the training, validation and test data for each of the following methods and briefly interpret your results.

1. **Linear Model**: The simple linear model imposes that conditional expectations $g^*(z_t)$ can be approximated by a linear function of the raw predictor variables and the parameter vector, θ,

$$g(z_t; \theta) = z_t^\top \theta$$

We estimate θ by minimizing the standard least squares, or ℓ_2, objective function

$$\mathcal{L}(\theta) = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} (r_{t+1} - g(z_t; \theta))^2 \tag{1}$$

There is no hyperparameter in the objective function (1), so you can estimate θ in the training set and report the MSE and R^2 for the training, validation and test data.

Hint: You can use

```
sklearn.linear_model.LinearRegression()
```

More details can be found `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html`

2. **Penalized Linear Model**: The penalized linear model also imposes that conditional expectations $g^*(z_t)$ can be approximated by a linear function of the raw predictor variables and the vector θ,

$$g(z_t; \theta) = z_t^\top \theta$$

However, the penalized methods differ by incorporating a new term in the loss function:

$$\mathcal{L}(\theta; \cdot) = \underbrace{\mathcal{L}(\theta)}_{\text{Loss Function}} + \underbrace{\phi(\theta; \cdot)}_{\text{Penalty}},$$

where $\mathcal{L}(\theta)$ is defined in (1). There are several choices for the penalty function $\phi(\cdot)$. We focus on the popular "elastic net" penalty, which takes the form

$$\phi(\theta; \lambda, \rho) = \lambda(1-\rho) \sum_{j=1}^{p} |\theta_j| + \frac{1}{2}\lambda\rho \sum_{j=1}^{p} \theta_j^2.$$

The elastic net involves two non-negative hyperparameters, λ and ρ, and includes two well known regularizers as special cases. The $\rho = 0$ case corresponds to the LASSO and uses an absolute value, or ℓ_1, parameter penalization. LASSO sets coefficients on a subset of covariates to exactly zero. In this sense, the LASSO imposes sparsity on the specification and can thus be thought of as a variable selection method. The $\rho = 1$ case corresponds to the to ridge regression, which uses an ℓ_2 parameter penalization, that draws all coefficient estimates closer to zero but does not impose exact zeros anywhere. In this sense, ridge is a shrinkage method that helps prevent coefficients from becoming unduly large in magnitude. For intermediate values of ρ, the elastic net encourages simple models through both shrinkage and selection.

In this problem,

(a) Let $\rho = 0$, find the optimal λ from the "cross validation" set and report the MSE and R^2 in the training, cross validation and test sets. under the optimal λ

More details can be found `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`

Hint: You can use

```
sklearn.linear_model.Lasso()
```

and hyperparamter λ corresponds to the α argument. More details can be found `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html`

(b) Let $\rho = 1$, find the optimal λ and report the MSE and R^2 as part 1.

Hint: You can use

```
sklearn.linear_model.Ridge()
```

and hyperparameter λ corresponds to the α argument. More details can be found `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html`

(c) Find the optimal λ and ρ ($0 \leq \rho \leq 1$) and report the estimation errors as part 1.

Hint: You can use

```
sklearn.linear_model.ElasticNet()
```

and hyperparameters λ and ρ corresponds to the α and $l1\_ratio$ arguments. More details can be found `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html`

3. **Principle Component Regression**: Principle Component Regression (PCR) consists of a two-step procedure. In the first step, principal components analysis (PCA) combines regressors into a small set of linear combinations that best preserve the covariance structure among the predictors. In the second step, a few leading components are used in standard predictive

regression. That is, PCR regularizes the prediction problem by zeroing out coefficients on higher order components.

In particular, we stack r_{t+1} as a $T \times 1$ vector R, z_t as a $T \times p$ matrix Z and e_{t+1} as a $T \times 1$ vector E. Then we can write linear regression $r_{t+1} = z_t^\top \theta + \varepsilon_{t+1}$ in a matrix form

$$R = Z\Theta + E$$

The forecasting method for PCR is written as

$$R = (ZW_K)\theta_K + E, \tag{2}$$

where $W_K = \begin{bmatrix} w_1 & w_2 & \cdots & w_K \end{bmatrix}$. W_K are the right singular vectors of matrix Z corresponding to the top K singular values.

In this problem, find the optimal K from the cross validation set and report the MSE and R^2 in the training, cross validation and test sets.

Hint: you can use

```
sklearn.decomposition.PCA()
```

More details can be found on `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`. You may find this useful: `http://www.science.smith.edu/~jcrouser/SDS293/labs/lab11-py.html`

4. **Partial Least Squares**: The forecasting method for Partial Least Squares (PLS) can also be written as

$$R = (ZW_K)\theta_K + E, \tag{3}$$

W_K is estimated from

$$w_j = \arg\max_w Cov^2(R, Zw), \quad w^\top w = 1, \quad Cov(Zw, Zw_l) = 0, \quad l = 1, 2, \cdots, j-1.$$

In this problem, find the optimal K from the cross validation set and report the estimation errors on the training and test sets.

Hint: You can use

```
sklearn.cross_decomposition.PLSRegression()
```

More details can be found on `https://scikit-learn.org/stable/modules/generated/sklearn.cross_decomposition.PLSRegression.html`. You may find this useful: `http://www.science.smith.edu/~jcrouser/SDS293/labs/lab11-py.html`

5. **Regression Tree**: Regression trees are a popular machine learning approach for incorporating multi-way predictor interactions. Unlike linear models, trees are fully nonparametric and possess a logic that departs markedly from traditional regressions. A tree "grow" in a sequence of steps. At each step, a new "branch" sorts the data leftover from the preceding step into bins based on one of the predictor variables. This sequential branching slices the space of predictors into rectangular partitions, and approximates the unknown function $g^*(\cdot)$ with the average value of the outcome variable within each partition.

   Formally, the prediction of a tree, $\mathcal{T}$, with K "leaves" (terminal nodes), and depth L, can be written as

   $$g(z_t; \theta, K, L) = \sum_{k=1}^{K} \theta \mathbb{1}(z_t \in C_k(L)),$$

   where $C_k(L)$ is one of the K partitions of the data. Each partition is a product of up to L indicator functions of the predictors. The constant associated with partition partition k (denoted θ_k) is defined to be the sample average of outcomes within the partition.

   The basic idea to estimate the tree is to myopically optimize forecast error at the start of each branch. At each new level, we choose a sorting variable from the set of predictors and the split value to maximize the discrepancy among average outcomes in each bin. The loss associated with the forecast error for a branch C is often called "impurity" We choose the most popular ℓ_2 impurity for each branch of the tree:

   $$H(\theta, C) = \frac{1}{|C|} \sum_{z_t \in C} (r_{t+1} - \theta)^2,$$

   where $|C|$ denotes the number of observations in set C. Given C, it is clear that the optimal choice of θ: $\theta = \frac{1}{|C|} \sum_{z_t \in C} r_{t+1}$. The procedure is equivalent to finding the branch C that locally minimizes the impurity. Branching halts when the number of leaves or the depth of the tree reach a pre-specified threshold that can be selected adaptively using a validation sample.

   In this problem, find the optimal number of leaves K and depth L from the cross validation set and report the MSE and R^2 in the training, cross validation and test sets.

   Hint: You can use

   ```
   sklearn.tree.DecisionTreeRegressor()
   ```

   and the hyperparameters L and K correspond to the *max_depth* and *max_leaf_nodes* arguments. More details can be found on `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html`

6. **Boosted Regression Tree**: Boosted regression tree recursively combines forecasts from many over-simplified trees. Shallow trees on their own are "weak learners" with minuscule

predictive power. The theory behind boosting suggests that many weak learners may, as an ensemble, comprise a single "strong learner" with greater stability than a single complex tree.

The Boosted regression trees starts by fitting a shallow tree (e.g., with depth $L = 1$). This over-simplified tree is sure to be a weak predictor with large bias in the training sample. Next, a second simple tree (with the same shallow depth L) is used to fit the prediction residuals from the first tree. Forecasts from these two trees are added together to form an ensemble prediction of the outcome, but the forecast component from the second tree is shrunken by a factor $\nu \in (0, 1)$ to help prevent the model from overfitting the residuals. At each new step b, a shallow tree is fitted to the residuals from the model with $b - 1$ trees, and its residual forecast is added to the total with a shrinkage weight of ν. This is iterated until there are a total of B trees in the ensemble. The final output is therefore an additive model of shallow trees with three tuning parameters (L, ν, B) which we adaptively choose in the validation step.

In this problem, find the optimal (L, ν, B) from the cross validation set and report the MSE and R^2 in the training, cross validation and test sets.

Hint: You can use

```
sklearn.ensemble.GradientBoostingRegressor()
```

and the hyperparameters L, ν and B correspond to the *max_depth*, *learning_rate* and *n_estimators* arguments. More details can be found on `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html`

7. **Random Forests**: A random forest is an ensemble method that combines forecasts from many different trees. It is a variation on a more general procedure known as bootstrap aggregation, or "bagging." The baseline tree bagging procedure draws B different bootstrap samples of the data, fits a separate regression tree to each, then averages their forecasts. Trees for individual bootstrap samples tend to be deep and overfit, making their individual predictions inefficiently variable. Averaging over multiple predictions reduces this variation, thus stabilizing the trees' predictive performance.

   In this problem, find the optimal depth of tree L and number of bootstrap samples B from the cross validation set and report the MSE and R^2 in the training, cross validation and test sets.

   Hint: You can use

   ```
   sklearn.ensemble.RandomForestRegressor()
   ```

   and the hyperparameters L and B correspond to the *max_depth* and *n_estimators* arguments. More details can be found on `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`

b/m	Book Value	tbl	Treasury Bills
AAA	AAA-rated Corporate Bond Yields	BAA	BAA-rated Corporate Bond Yields
lty	Long Term Yield	ntis	Net Equity Expansion
Rfree	Risk-free Rate	infl	Inflation
ltr	Long Term Rate of Returns	corpr	Corporate Bond Returns
svar	Stock Variance	d/p	Dividend Price Ratio
d/y	Dividend Lagged Price Ratio	e/p	Earning to Price Ratio
d/e	Dividend Payout Ratio	ret	Index Return

Table 1: List of financial variables

. . . . . .

References

Ivo Welch and Amit Goyal. A comprehensive look at the empirical performance of equity premium prediction. *The Review of Financial Studies*, 21(4):1455–1508, 2007.