IE 582 Statistical Learning for Data Mining
Homework 3, due January 1st, 2021

Instructions: Please solve the following exercises using R (http://www.r-project.org/) or Python (https://www.python.org/). You are expected to use GitHub Classroom and present your work as an html file (i.e. web page) on your progress journals. There are alternative ways to generate an html page for you work:

- A Jupyter Notebook including your codes and comments. This works for R and Python, to enable using R scripts in notebooks, please check:
    - https://docs.anaconda.com/anaconda/navigator/tutorials/r-lang/
    - https://medium.com/@kyleake/how-to-install-r-in-jupyter-with-irkernel-in-3-steps-917519326e41

  Things are little easier if you install Anaconda (https://www.anaconda.com/). Please export your work to an html file. Please provide your *. ipynb file in your repository and a link to this file in your html report will help us a lot.

- A Markdown html document. This can be created using RMarkdown for R and Python-Markdown for Python

Note that html pages are just to describe how you approach to the exercises in the homework. They should include your codes. You are also required to provide your R/Python codes separately in the repository so that anybody can run it with minimal change in the code. This can be presented as the script file itself or your notebook file (the one with *.ipynb file extension).

The last and the most important thing to mention is that academic integrity is expected! Do not share your code (except the one in your progress journals). You are always free to discuss about tasks but your work must be implemented by yourself. As a fundamental principle for any educational institution, academic integrity is highly valued and seriously regarded at Boğaziçi University.

Penalized Regression Approaches

Background

We have covered two penalized regression approaches in class, namely ridge and lasso regression/classification with linear models. There are also alternative penalization approaches for data with a specific structure such as time-series. Suppose we would like to perform regression given a time series data set. In other words, we have time series observations and there is a continuous response associated with this time series observations. A penalized regression approach with fused penalties (Tibshirani et al., 2005) minimizes the following loss function:

$$L(\lambda_1, \lambda_2, \beta) = |\mathbf{y} - \mathbf{X}\beta|^2 + \lambda_2|\beta|^2 + \lambda_1 \sum_{j=2}^{p} |\beta_j - \beta_{j-1}|$$

The first part is the sum of squared errors, second part is the ridge penalty over the coefficients and the last part is the fused lasso penalties. As you can see from the equation, the coefficients of the variables that are temporally close (i.e. coefficient for an observation at time *t* and time *t+1*) are motivated to be close to each other. This loss function will be revisited later.

Assume that we are interested in a predicting the tomorrow's hourly electricity consumption of Turkey. The consumption series are made publicly available by EPİAŞ @ https://seffaflik.epias.com.tr/transparency/.

Please download the consumption series using the "realized consumption" menu item under the "consumption". That should bring you the following link:

https://seffaflik.epias.com.tr/transparency/tuketim/gerceklesen-tuketim/gercek-zamanli-tuketim.xhtml
You can use the data from 1st of January, 2016 till the 1st of December, 2020. You can export the date-filtered data using the icon corresponding to "csv" file as a csv file.

Tasks

Our main task will be devising a penalized regression approach to forecast the next-day's hourly consumption. In other words, we are expected to provide 24 predictions corresponding to the hours of the next day. To perform this task, we need to extract features from the consumption series. In order to keep things easy, we will work on a simple feature extraction strategy to generate the necessary predictors (i.e. regressors). Electricity consumption is known to have certain seasonal patterns. It is known that daily and hourly seasonalities are very important in this type of prediction task. Therefore, I recommend you to use the last week's same day and same hour consumption as our primary predictor. In other words, while I am predicting the consumption level for 6th hour of November 1st, 2019 (Friday), I can use the 6th hour consumption level of last Friday (i.e. October 25th, 2019). Let's denote this predictor as our "naïve forecast" as it can be used as a prediction itself and the use of it is very common in many companies in the energy sector. From the time series perspective, this variable is also referred to as "*lag 168"* (i.e. 7 x 24 =168) consumption.

There are alternative strategies to model the consumption prediction. Our aim is to compare the alternative approaches in terms of their prediction performance. Use mean absolute percentage error (MAPE) as your primary metric for comparison. For each hour over the test period, you will obtain a prediction and a respective MAPE value.

a) Assume that you are willing to use 168 and 48 hours ago consumption values as your naïve approaches to predict next day's consumption. Suppose the test period includes the dates after 1st of November, 2020 (included). For both approaches, report the summary statistics of MAPE values for the test period.

b) Instead of using the lag consumptions in part (a) as a forecast, we would like to treat them as our features and build a linear regression model. The illustration of the data format for this prediction problem is given in Figure 1. This representation is so called "long" format. Train your model using the data till 1st of November, 2020 and test on the rest. Your linear regression model is expected to include aforementioned two features (i.e. Lag_48 and Lag_168) and an intercept. Report the summary statistics of MAPE values for the test period.

Date	Hour	Lag_48	Lag_168	Consumption
a	0			
a	1			
a	2			
a	3			
a	4			
a	5			
a	6			
.				
.				
.				
.				
e	17			
e	18			
e	19			
e	20			
e	21			
e	22			
e	23			

Figure 1. Data representation in "long" format

c) As mentioned earlier, hourly seasonality is important. Although we used the same hour's consumption value of the past days to handle this problem for part (b), we implicitly impose an assumption that prediction model for each hour has the same coefficients which may not be correct since the consumption behavior at nights can be different than the other hours. Therefore, modeling each hour separately is another way to approach to the same problem. Train linear regression models for each hour using the same training period (24 models) and report your test performance as in part (a).

d) One of your friends comes up with an alternative approach assuming that all hourly consumption values of last week (same day) can be important in the prediction of the next day's consumption. In other words, you can use the 24 consumption values of the last week to predict next day's consumption. This requires the transformation of your data into a so called "wide" format from the "long" format (i.e. the form of the data you used in part (a) is referred to as "long" format). Figure 2 illustrates the wide format for predicting the consumption of 18th hour.

Date	Lag_day7_hour_0	Lag_day7_hour_1	Lag_day7_hour_2	.	.	.	Lag_day7_hour_24	Consumption_18
a								
.								
.								
.								
.								
.								
.								
.								
e								

Figure 2. Data representation in "wide" format. This representation considers the prediction of 18th hour.

Assume that you have 48 features (hourly consumption from two days ago and last week's hourly consumption) in total. You are also willing to follow the same logic in part (c) and build a prediction model for each hour separately. Since there is a strong correlation (actually an autocorrelation) between these predictors, you are willing to use penalized regression approaches for modeling. Use L_1 penalty in your regression models for each hour. Note that the feature matrix will be the same for all your models, only the target variable will change for this task. In order to determine the regularization parameter (i.e. lambda), perform a 10-fold cross-validation.

Train penalized regression models with L_1 penalty (i.e. lasso regression) for each hour using the same training period (24 models) and report your test performance as in part (a). Also comment on the resulting models (i.e. coefficients and etc.).

Hint: You can use *glmnet* package in R to perform this task, *cv.glmnet* function performs the cross-validation for you. In Python, scikit learn have a similar functionality enabled by *Lasso* function under *linear_models*. *LassoCV* is a helper function to perform the cross-validation. Do not forget to set seed for both languages so that your results are reproducible.

e) **(BONUS)** An alternative approach to lasso regression in part (d) is to perform the same tasks using fused penalties since the predictors are basically time series. A sample loss function for fused lasso regression is provided in "Background" section. While building a model to predict an hour's consumption, we explicitly used two time series (Lag_Day_7 and Lag_Day_2 hourly consumption values). Therefore, penalizing the coefficients for consecutive predictors of each time series (Lag 7 and Lag 2) may provide in performance improvements. Following a similar logic as in part (d), train regression models with fused penalties for predicting each hour. You can use a single penalty term for the fused penalties (i.e. lambda_1) and L_1 penalty for coefficients (i.e. lambda_2). In order to tune your model, you can perform 10-fold cross-validation.

Hint: This question involves creation of a custom loss function which is not implemented in many packages. I suggest you to use CVXR package in R and cvxpy module in Python. I have R examples on Moodle (documentation is available @ https://cvxr.rbind.io/). Documentation and examples for Python is available @ https://www.cvxpy.org/index.html

f) Compare the results drawing a boxplot of MAPE values for each approach on same plot. Comment on your findings.