CodeLab – 2

Regression models for load forecast

Electricity is becoming the primary energy source of our modern society. From transportation to heating, electricity demand is increasing with higher variability. In Figure 1, one can find the energy consumption per capita in the Netherlands [1]. Overall we need a secure operation of the power systems to ensure the electricity supply.

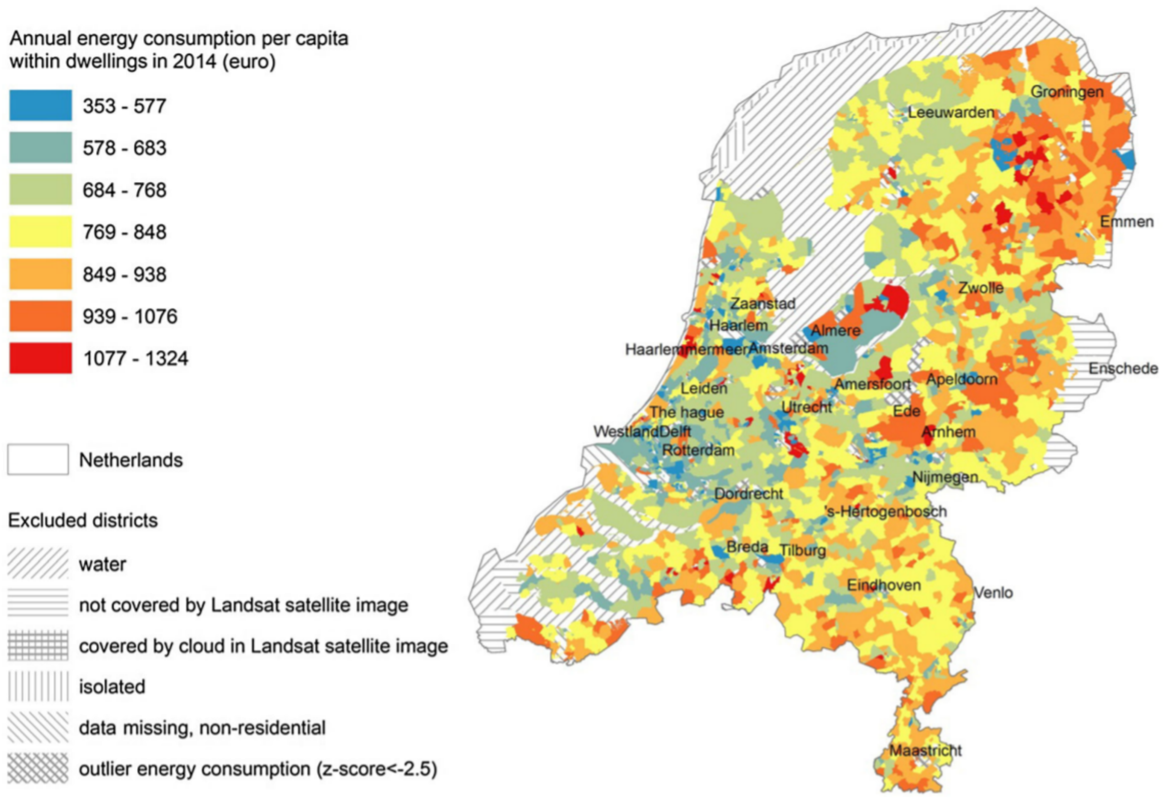

Figure 1:Netherlands annual energy consumption per capita.

Power system operation relies on the balance between generation and demand. Traditionally, the amount of generation is dispatched the day before. Current market mechanisms require day-ahead bidding to determine the hourly schedules.

In this CodeLab, you will be the energy supplier of the Netherlands. Although you are a monopoly in the market, you should still follow the rules and bid in the market. In order to achieve a profitable business, you must determine the next day's hourly demand profile. Since you have some historical and publicly available data related to electricity consumption behavior, you can develop a machine learning algorithm for prediction.

ENTSO-E is the European Network of Transmission System Operators and represents 42 electricity transmission system operators from 35 countries across Europe. ENTSO-E has a transparent platform where information regarding generation, load, outages, and many more statistics are published regularly

[2]. Actual and forecasted demand (MW) of the Netherlands are taken from the transparency platform. Data related to weather behavior in the Netherlands is retrieved from the "Modern-Era Retrospective Analysis for Research and Applications 2" (MERRA-2), the latest atmospheric reanalysis of the modern satellite era produced by NASA. This dataset contains various weather data like temperature or precipitation, which have a direct impact on our daily routines. For example, on a sunny, hot day, people will tend to go outside rather than vacuum the house, which causes a reduction in electricity consumption. Combining two different data types, you should generate a proper machine learning model to predict future electricity consumption. Figure 2 illustrates the model structure you will develop in this CodeLab.

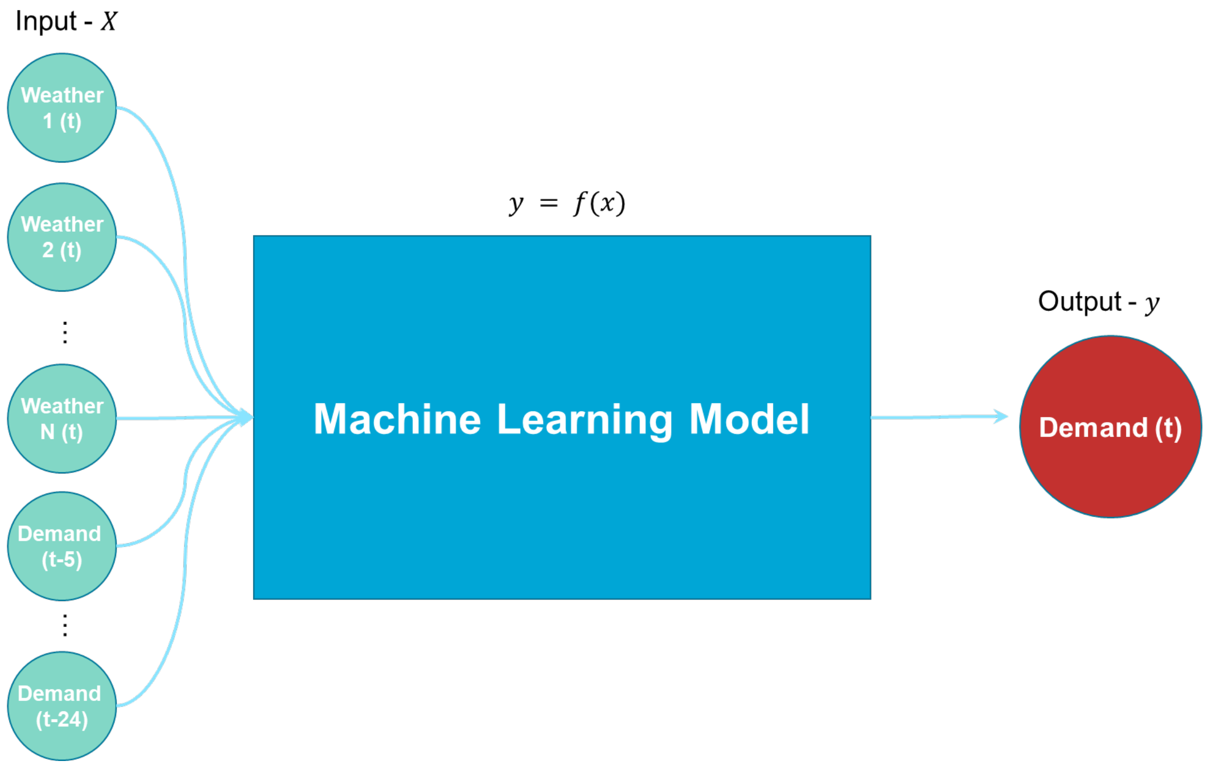

Figure 2: Machine learning model structure for load forecast problem.

Provide your answers in a report in PDF format that contains all the requested outputs and answers to the questions. You will obtain a Pass or Fail from each CodeLab, where a sufficient amount of work and answers are required for each part of the CodeLab. Missing parts and question answers will cause a Fail for the associated CodeLab. You do not have to submit the Jupyter Notebook, but TAs can request it after submission. Using LLM models is not prohibited, but the exact answers to written questions from different students will be regarded as plagiarism and automatic Fail. The report must be submitted to Brightspace before the deadline.

Task 1 – Load energy data

The "*Data.csv*" file contains all related information and can be loaded/read using the Pandas library. Pandas library is useful for investigating and processing datasets.

- Load the dataset using the "*pandas.read_csv()*" function.

- Find the date and hour corresponding to the maximum electricity consumption and write it in the report.
- Plot the following variables for the entire day (00:00-23:00) of the maximum consumption in subplots. **Subplot 1**: Actual load (MW) and Forecasted load (MW) together, **subplot 2**: temperature, **subplot 3**: irradiance_surface, and **subplot 4**: snowfall. The X-axis of all subplots is the hour of the day.

The process of training a machine learning model relies on data or features. Properly prepared features are essential for learning. Real-life applications require knowledge of data and models. This process is called feature engineering, which will be covered in CodeLab 4. In this CodeLab, three different feature matrices are given, and you will work on these 3 different feature matrices X_1, X_2, and X_3.

- Use the following pickle command *pickle.load* to load the dictionary "*D*", which contains features and the target variable. Print the keys of the dictionary with "*.keys()*" function and write them in the report.

"Feature1" (X_1) contains historic load demand and weather predictions. "Feature2" (X_2) has the entire content of "Feature1" and the load forecast values from Entso-E. "Categorical" (C) features represent the categorical time element of the data in a one-hot encoded way. For example, $C[i] = [100001100]$ represents data taken from a weekend in the fall season and a daytime (07:00-16:00) load. Details regarding one-hot encoding and the structure of C are included in the bonus task of the CodeLab. "Target" contains the target variable (y) of the ML model.

- Extract the contents of the dictionary in following variables X_1, X_2, C, and y.

In this analysis, the first week of the data is excluded from prediction. In other words, the target variable (y) considers the period between 08.01.2019 and 31.12.2019. Historical data can be retrieved from the first week. For example, $t = 0$ corresponds to 08.01.2019-00:00, and $t-1$ should be 07.01.2019-23:00.

X_1: Contains only weather forecasts and historical demand. The model can reach weather data at the time t. For historical demand at the time t ("$P_{load}(t)$"), five different feature vectors are generated. $t-5$, $t-6$, $t-7$, $t-8$, and $t-24$ hours are the selected time shifts for historical data to predict actual demand at time t ("$y(t)$").

$$y(t) = f(X_1(t)) = f(\,[Weather_1(t), Weather_2(t), \dots, P_{load}(t-5), P_{load}(t-6), \dots, P_{load}(t-24)]\,)$$

X_2: Extended version of the X_1. Additionally, it contains the forecasted values coming from the ENTSO-E platform.

- Calculate and write the sparsity of X_1, X_2, and C in the report.

X_3: Extended version of the X_2. It is generated by combining X_2 and matrix C, which contains categorical variables regarding time.

Pandas: Basics

Numpy basics: Numpy

Task 1 – Questions

1) What is the difference (compute) between the forecasted and actual loads at the hour of the maximum consumption?
2) What do you observe from the generated subplots?
3) Which features in the subplots might be good features for a machine learning model and why?
4) Calculate and write the sparsity of C.
5) Explain why to use encoding instead of numeric date values in a regression problem.
6) Explain why we need the time information for the load forecast problem.
7) Why is it important to calculate sparsity? What does having a high/low sparsity mean, and how would it potentially affect the regression?

Task 2 – Process data and prepare for machine learning algorithms

In this task, you will generate all feature sets and standardize or scale the data for the training process.

The standardization process is essential for many machine learning models. For instance, many algorithms (such as the RBF kernel of Support Vector Machines) assume all features are centered around zero and have variance in the same order. If a feature has a variance that is orders of magnitude larger than others, it might dominate and make the estimator unable to learn from other features correctly as expected.

Features are standardized by removing the mean and scaling to the unit variance. This process is called standard scaling or z normalization. $z = (x - \mu)/\sigma$ where μ is the mean and σ is the standard deviation. All noncategorical variables must be standardized.

- Use Sklearn *StandardScaler(.)* to scale X_1.
- Define a new scaler using *StandardScaler(.)* to scale X_2.
- Concatenate the categorical variable matrix C to the scaled X_2, which will generate the third feature set X_3.

The dataset must be split into training and test sets. 25% of the data will be used for test proposes. To generate reproducible results, please use the given *Shuffle_state* variable as a random state of the splitter function. Note: When you call the *Shuffle_state*, *y_train*, and *y_test* (target variables) from different splits will be the same.

- Split X_1, X_2, and X_3 using the Sklearn *train_test_split()* function into 3 training and test data sets.
- Write the mean of X_1, scaled X_1, training data of X_1 (scaled), and test data of X_1 (scaled) in the report.

Sklearn : StandardScaler, Preprocessing, Train/Test Split

Task 2 – Questions

1) Why do you use two different scalers? What happens if the first one is used to scale X_2 as well?
2) Why are categorical variables not scaled?

Task 3 – Training and evaluation of linear regressor

In this CodeLab, different regression models are implemented. Using linear regression models, you should identify the best feature set to investigate further. After the most appropriate feature set is identified, regularization on linear regression is investigated through Ridge and Lasso models.

- Construct and train three linear regression models for each feature set X_1, X_2 and X_3.
- Define a custom function *"test_mymodel(model,X_test,y_test)"* to evaluate the model performance. The function takes the model, test features, and test target variable. The output is R^2 score and MSE. Provide your function in the report.
- Evaluate models with *test_mymodel()* and write the output in the report.
- Plot 2 bar graphs for R^2 and MSE of all linear regression models.
- Plot test data predictions of the three models and test data together for the first 64 sample points *([0:63])*.
- Write the model weights of the best model and the absolute maximum difference between weights.

Ridge & Lasso regression models are modified forms of linear regression. The complexity parameter "*alpha**" controls the complexity of the model. Adjust the *alpha* parameter to obtain the best results. Although *alpha* in both models adjusts the regularization in the optimization problem, their mathematical interpretations are different, so the *alpha* values will have different effects in training.

Ridge & Lasso models construct an optimization problem. Therefore, for numerical stability, we have scaled the target variable with the following code block. Please use the new *y_train_s* and *y_test_s* to fit and test models. Otherwise, convergence problems might be observed.

- Train ridge and lasso regression models with *alpha=0.1*, using the best-performing feature set in the previous linear regression task.
- Test both models with *test_mymodel()* and record (and print) R^2 and MSE values in the report.
- Train and test both models again with varying alpha values from 0.005 to 1.0 with a step size equal to 0.005. Record R^2 and MSE at each step.
- Plot "*alpha* vs. MSE" and "*alpha* vs. R^2 " figures in the report and find the best-performing regularization parameter alpha and its effect on prediction.

Sklearn linear regression: Linear Models Guide , LinearRegression , Ridge , Lasso

Matplotlib : Bar Plot , Pyplot

Task 3 – Questions

1) Write down R^2 and MSE scores of linear regression models and compare/comment on the results.
2) What does MSE represent for the linear regression problem in this CodeLab?
3) Which feature set gives the best prediction capability?
4) What is the absolute difference between the maximum and minimum weights of the best-performing model?
5) What do weights correspond to in a linear regression?

6) What is the role of *alpha* in Ridge and Lasso models?
7) Why is the MSE of Ridge & Lasso a thousand times smaller than the linear regression?
8) Compare the results of regularized models with previous linear regressors.
9) What is the value of *alpha* that gives the smallest error in the test set?

Task 4 – Support Vector Machines

Support vector machines (SVMs) is a supervised machine learning algorithm for classification, regression, and outliers detection. This algorithm will be investigated in detail in week three. However, now you can treat it as a black box model. If you want to discover what it is and how it works, this link may help you.

https://medium.com/cube-dev/support-vector-machines-tutorial-c1618e635e93

In this CodeLab, SVM is investigated with different parameters and kernels. Kernels are decision functions applied to transform the feature space into higher dimensions. The nonlinear relationships between features can be found in the high dimensional space.

Compared to the linear models, the construction of SVM is much more expensive. The time library can be utilized to measure the training time of SVM. For this task, please use again the scaled target variables for the numerical stability.

Note: Add maximum iteration as *1e6* or *1e7* inside the model definition to avoid long training times.

- Construct a linear SVM regressor with the default parameters and record performance metrics, as well as the training time in the report.
- Conduct a grid search (apply all possible pairs) on the regularization parameter C and ϵ with the given list of parameters. $C = [0.01, 0.1, 1, 5]$, $\epsilon = [0.01, 0.1, 1, 5]$
- Write the best-performing parameters in the report. (It is possible to observe different sets of parameters for the lowest MSE and the highest R^2. Please select one performance metric to find the hyperparameters.)

Hint: The grid search process can be done by nested loops. Construct two *for* loops to iterate each hyperparameter and train with different parameter combinations at each iteration. Lists are useful for recording performance metrics.

- Construct a polynomial SVM regressor with the best regressor hyperparameters identified in the previous task and default settings for the other parameters (gamma, degree, etc.). Write R^2, MSE and training time in the report.
- Adjust the gamma parameter from 'auto' to 'scale' and observe the effect by measuring time and test scores.
- Construct an RBF SVM regressor with best-performing hyperparameters from the polynomial SVM, and record R^2, MSE.
- Change the kernel scale gamma using the given kernel scale set (γ) and write the time and performance of the optimization problem in the report. $\gamma = [0.01, 0.05, 0.1, 0.2, 0.3, 0.4]$
- Plot γ vs R^2 and γ vs MSE curves in the report. **Optional:** You can plot the prediction and label together.

*: Alpha corresponds to regularization parameter λ in lecture 2.

- **Optional:** Find and tune the best SVM model for the regression problem. (You might find a better model than the solution manual: R^2 =0.97624, MSE=0.02406)

SVM: Function, Guide

Task 4 – Questions

1) Write and compare the results of default and tuned linear SVMs.
2) Write and compare the results of auto and scaled gamma parameters for polynomial SVM.
3) Write and compare the results of different gamma values for RBF SVM.
4) Which model (with parameters) performs most in the load forecast problem?

Bonus Task

In this bonus task, you have to implement a regression model that predicts the total electricity consumption of a day using only time information in a categorical form.

Some data types cannot be represented as numeric values, like a day of the month. Entering a numeric value of the day can cause the last weeks of the month to have a larger impact on prediction, which may affect the prediction capability of many machine learning models. Each numeric value can be transformed into a binary representation.

For example, the colour of apples 'red' and 'green' can be codded with 2 binary vectors $a1$ and $a2$ where $a1$ represents whether the apple is red or not, $a2$ shows green or not with binary values 1 and 0. ($a1[55]$=1 , 56th sample point is a red apple, which requires $a2[55]$=0 because it cannot be both red and green at the same time.)

3 different categorical variables are used to model the time element. Matrix C has a shape of $8592 * 9$. Each column contains a specific categorical binary value. Order of columns in the matrix C must be identical in the same order given below.

a) Weekend and weekday
b) Seasons: Winter, Spring, Summer, Fall
c) Hours: Day (07:00-16:00), Peak (17:00-21:00), Night (22:00-06:00)

$$C = [Weekend, Weekday, Winter, .., Fall, Day, .., Night]$$

For this task, create a new feature matrix or vector X_{bonus} from the given Input CSV file that can only contain time-related features. After that, create the vector y_{day} which contains the total daily consumption level. Finally, create your regressor model, train, and evaluate the model performance. Provide your observations and findings.

References

[1]: Mashhoodi, B., Stead, D. & van Timmeren, A. Local and national determinants of household energy consumption in the Netherlands. *GeoJournal* **85,** 393–406 (2020). https://doi.org/10.1007/s10708-018-09967-9

[2]: Hirth, Lion & Mühlenpfordt, Jonathan & Bulkeley, Marisa, 2018. "**The ENTSO-E Transparency Platform – A review of Europe’s most ambitious electricity data platform**," Applied Energy, Elsevier, vol. 225(C), pages 1054-1067.

Linear Regression : https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

SVM : https://scikit-learn.org/stable/modules/svm.html#svm-regression