6/2/2020

Project 2: Predict Energy Use

Computational Machine Learning Assignment 2

COSC2793

SARTHAK SIRARI | S3766477

MASTER OF DATA SCIENCE | MC267
COMPUTER SCIENCE & IT, SCHOOL OF SCIENCE
RMIT UNIVERSITY

INTRODUCTION

In this machine learning project, we were given a real-world Energy Usage dataset which contains several factors, ranging from temperature and humidity in different rooms within a house inside a building and outside the building, which were recorded with a ZigBee wireless sensor network; plus other factors like pressure, wind speed, visibility etc. were also recorded from the nearest weather station at Chievres Airport (Belgium), that determine the energy use of light fixtures and appliances in the house. Apart from that there is a timestamp given with each record; plus two other random factors were added to this dataset for regression models test and filtering attributes that are non-predictive. We used different regression algorithms to determine how to train our model. We applied several machine learning tasks to train different models with the training data, predict and test our fitted model with the testing data and finally predict the values from the unseen data.

WORKFLOW

We started the task by importing all the necessary libraries that will be used in this task. We then, imported the data and made the timestamp as an index, extracted the time series features and created new features (like hour, month etc.) and started with the data exploration step. After exploring the data, we concluded that the data needs to be scaled, linear regression should not be applied as there was no linear dependence of features with the target element and decided on the regression models, which could best fit for the given dataset. Before training our machine learning model, we first scaled the dataset, checked which and how many features must be selected, then applied feature selection based on the results of the analysis, then applied the hyperparameter tuning to the selected regression models, then fit the model using the hyperparameters selected from the results of its tuning and then applied the selected models to our dataset. In order to provide an ultimate judgement of the "best" model that we would use, we performed the comparison of Mean Squared Error and R2 Score of each model and we checked which model gives the best predictions on the unseen data. After comparing all the models, we concluded our final judgement on which machine learning is the best fit for the given data set. The detailed workflow is described further.

1. Data Analysis

The following part contains all the steps taken to conclude the data exploration.

- In the data exploration part, after importing the data, we **used the timestamp** provided in the 'date' column **as the index** for the data frame.
- We first **split the data** into two parts, i.e. 85% train (containing all the data which is used to train and analyse the machine learning model) and 15% unseen (containing the data which will only be used for evaluation of the trained model). This split is done because we need to check how well our fitted models work with the unseen data.
- Then we **sorted all the values in the data frame** "train" by the index column, i.e. "date". This is done because we wanted to see how often each record was recorded, to conclude if resampling of data is needed.
- From the sorted data frame, we observed that the records were recorded every 10 minutes. However, we do not need these many entries because there is not much difference in any of the features for consecutive records. Hence, we decided to perform **hourly resampling** because we observed that there were significant changes in the observations for many consecutive hours. This is mainly because if we observe in our real lives, for example, the morning hours we consume more electricity as we prepare breakfast and get ready for school/work using many appliances, then in the afternoon mostly people are outside of their homes, so electricity consumption is low, then in the evening after we return from

school/work, we again consume more electricity as use appliances for preparing dinner or recreational activities and finally in the night time, since mostly people are sleeping, so lesser consumption of electricity is observed. Therefore, we can say that energy consumption depends on an hourly basis, hence we did the hourly resampling.

- After that, we performed **time series feature extraction** from the timestamp index and created the following features and their reasons: -
    - **Month**: Each month the climatic conditions are different, hence the energy usage will be different and this could help in predictions, e.g. in winters, we use more energy on heating appliances, whereas in summers, we use more energy in cooling appliances.
    - **Day**: Depends on the day of the month, we can help predict the household energy usage, e.g. many people who gets monthly salary at the starting of the month, tends to stay out more to spend their money, so the energy consumption at home is lesser during this period, which is vice-versa at the month end.
    - **Day of the Week**: Depending on the day of the week, household energy usage can be predicted as people stay out their homes more during weekdays and mostly stay at home during the weekends.
    - **Hour**: As explained in detail for hourly resampling in the last step how household energy consumption depends on the hour of the day, this feature can be used for energy consumption prediction.
- Now, we **split the data** into two parts, i.e. X (containing all the features which is used to train the machine learning model) and Y (containing only the target column, i.e. TARGET_energy).
- We, then, plot a **histogram** (shown in Fig. 1) for all the features which is used to train the machine learning model to check for several factors which may affect the accuracy of our model, such as range and skewness of each feature.
- The plotted histograms showed that most of the features had **skewed data**. The data is either skewed to the right or to the left. There is no sign of only one-sided skewness. Only T1, T8, rv1 and rv2 columns showed little to no skewness.
- Further, the plotted histograms also showed that almost all the features were scaled differently. The data in each feature was having **different ranges**.
- Then we moved ahead to check the **correlations** of the features amongst each other. To find the correlations, we plotted a **heatmap** (shown in Fig. 2) with correlation factors of each feature in our training dataset.
- The plotted heatmap displayed **significant correlations** amongst many features.
- Hence, we drew the conclusion to perform **Normalisation** on our training dataset before we start to train our model.
- We, then, plot the **histogram** (shown in Fig. 5) for the target column. It can be clearly seen from the plot that the Target is highly **skewed to the right**, this will impact the accuracy of our trained model.
- Now, plot a **boxplot** (shown in Fig. 3) for all the features which is used to train the machine learning model to check for outliers which may affect the accuracy of our model. **Numerous outliers** can be observed from the plot, mostly in T2, RH_5, Visibility features. These features will negatively impact our accuracy rate.
- We plot the **boxplot** (shown in Fig. 6) for the target column. It can be clearly seen from the plot that the Target has a **huge number of outliers**, this will negatively impact the accuracy of our trained model.
- Finally, we plotted the **scatter plot** (shown in Fig. 4) of each feature against the target column of our data frame. We can observe that there is **no linear relationship** between any of the features with the target element, this concludes that we should use regression techniques other than the Linear Regression.

2. Model Fitting

The following part contains all the steps taken into account to determine our ultimate judgement on the BEST model.

- Since we could only use the **regression algorithms** to train our machine learning model and only our features show no linear relationships with the target element, no type of Linear Regression should be used, hence this narrowed our choices.
- After data exploration, it was clear that the target value is a **continuous data**, so the following regression techniques were tried to fit the model: -
  - Decision Tree
  - Random Forest
  - Ada Boost
  - Gradient Boosting
  - Support Vector Regression
- Out of the above-mentioned algorithms, it was observed that **Random Forest Regressor** and **Support Vector Regression** provided better R2 Score and lower Mean Squared Error rate than other fitted algorithms, therefore, we went ahead by modelling these two algorithms.
- Neural network algorithms were not applied to our data set. This is done because neural network-based algorithms are used mainly for Classification problems (Image Classification, to be precise). Hence, it was not applied to our regression problem.
- We started by selecting the appropriate **Data Normalisation** or **Scaling** techniques. We applied the following techniques: -
  - Normalizer
  - StandardScaler
  - MinMaxScaler
  - RobustScaler
  - QuantileTransformer
  - MaxAbsScaler
- After applying each technique we concluded that **MinMaxScaler** provided the best accuracy rate for the machine learning model trained with the provided data.
- After scaling the data, we **split the data further** into testing and training datasets. We tried the following splits of data: -
  - 80% training and 20% testing data
  - 70% training and 30% testing data
  - 60% training and 40% testing data
- We chose the split to be **80% training and 20% testing** data because this split provided the best accuracy rate for machine learning models trained with the provided data.

Algorithm 1: Support Vector Regression

- **Feature Selection**: In this step, we used the Filter Method '**SelectKBest**()' method to identify the k-features with the highest score as this feature selection method, gave the best number of features to be selected because it resulted in creating a better accuracy model. As the 'score_func' hyperparameter for the method, we used 'f_regression' because this is the scoring mechanism which is used for regression problems. For the 'k' value, we started a loop from selecting one feature till selecting all features. Within this loop, for each value of 'k', we transformed the training data frame and used the SVR model with 'C' as '1000' (which is the Regularization parameter , should be kept positive) and 'gamma' as '1' (which is the Kernel coefficient) to get the cross validation score, because these hyperparameters gave the best accuracy rate for the SVR model. We used the '**cross_val_score**()' method to calculate the cross validation scores, keeping the 'cv' as '10' for using the ten-fold validation and the 'n_jobs' value was set to be '-1' to utilise every processors to run the jobs to get the results faster. All the scores are then plotted (shown in Fig. 7), and it showed that using **23 features** will give the best accuracy rate for the SVM algorithm.

- **Hyperparameter Tuning**: In this step, we used the '**GridSearchCV**()' as the method for identifying the best parameters for our SVR algorithm as it provided the parameters that gave the best accuracy rate for the SVR model. As the 'param_grid' parameter (which is the grid of parameters which will be applied to the provided model to identify the best parameters from), we gave 'C' as '[0.1, 1, 10, 100, 1000]', 'gamma' as '[1, 0.1, 0.01]' and 'kernel' as '['rbf']'. The best parameters returned were "'C': 1000, 'gamma': 1, 'kernel': 'rbf'". Then for hyperparameter tuning of the 'linear' and 'poly' kernel, we used the 'gamma' value as '['scale', 'auto']' and 'C' was kept the same. This returned with identifying "C = 100, gamma = 'scale', kernel = 'poly'" as the best parameters for fitting. The other parameters used for 'GridSearchCV()' were 'cv' as 3 to make the three-fold of data for cross validation, 'verbose' was kept as 'False', so that it does not gives any output while running the tasks and finally 'n_jobs' was kept as '-1' for the same reason as explained during feature selection.
- **Model Fitting**: Both the SVR model for kernels 'rbf' and 'poly' were fitted using the parameters provided by the hyperparameter tuning performed in the previous step.

Algorithm 2: Random Forest Regressor

- **Feature Selection**: In this step, we used the Filter Method '**SelectKBest**()' method to identify the k-features with the highest score as this feature selection method, gave the best number of features to be selected because it resulted in creating a better accuracy model. As the 'score_func' hyperparameter for the method, we again used 'f_regression' because this is the scoring mechanism which is used for regression problems. For the 'k' value, we started a loop from selecting one feature till selecting all features. Within this loop, for each value of 'k', we transformed the training data frame and used the RandomForestRegressor model with 'max_depth' as '100' (which is the max depth of the tree), 'max_features' as 'auto' (which is the no. of features to be considered while looking for the best split, 'auto' will ensure all the features are used), 'min_samples_leaf' as 3 (which is the min no. of samples required at a leaf node), 'min_samples_split' as 6 (which is the min no. of samples used to split an internal node) and 'n_estimators' as 100 (which is the number of trees in the forest) to get the cross validation score, because these hyperparameters gave the best accuracy rate for the RandomForestRegressor model. We used the '**cross_val_score**()' method to calculate the cross validation scores, keeping the 'cv' as '10' for using the ten-fold validation and the 'n_jobs' value was set to be '-1' to utilise every processors to run the jobs to get the results faster. All the scores are then plotted (shown in Fig. 8), and it showed that using **28 features** will give the best accuracy rate for the RandomForestRegressor algorithm.
- **Hyperparameter Tuning**: In this step, we used the '**GridSearchCV**()' as the method for identifying the best parameters for our RandomForestRegressor algorithm as it provided with the parameters that gave the best accuracy rate for the RandomForestRegressor model. As the 'param_grid' parameter (which is the grid of parameters which will be applied to the provided model to identify the best parameters from), we gave 'max_depth' as '[50, 100, 150]', 'max_features' as '['auto']', 'min_samples_leaf' as '[1, 2, 3]', 'min_samples_split' as '[2, 4, 6, 8, 10]' and 'n_estimators' as '[10, 50, 100]'. The best parameters returned were "'max_depth': 150, 'max_features': 'auto', 'min_samples_leaf': 3, 'min_samples_split': 4, 'n_estimators': 100". The other parameters used for 'GridSearchCV()' were 'cv' as 3 to make the three-fold of data for cross validation, 'verbose' was kept as 'False', so that it does not gives any output while running the tasks and finally 'n_jobs' was kept as '-1' for the same reason as explained during feature selection.
- **Model Fitting**: The RandomForestRegressor model was fitted using the parameters provided by the hyperparameter tuning performed in the previous step.

3. Model Evaluation

The following part contains all the steps taken to conclude the evaluation of our fitted models.

- After training each model, we predicted the target values of the **testing** data and recorded the following **scores** to **evaluate** each model: -
  - Mean Squared Error (lowest is the best)
  - R2 Score (highest is the best)
- We make the predictions using the respective models on the **training** data.
- Over the **unseen** data that we set aside at the beginning, we perform the feature selection use for the respective models. We, then, make the predictions using the respective models on the **unseen** data.
- Mean Squared Error and R2 Score are stored for the prediction of **testing, training and unseen** data of SVC and RandomForestRegressor models (shown in Fig. 9). Overfitting or Underfitting of the data can be observed from this table.

Ultimate Judgement

- From the table (shown in Fig. 9), it can be observed that the SVC RBF model has R2 score of **testing** as 0.40, whereas, the Random Forest Regressor model has R2 score of **testing** as 0.42. This indicates that Random Forest Regressor must be used as the BEST model, as its R2 score is higher.
- Now in the same table, we can see that the SVC RBF model has R2 score of **training** as 0.62, whereas, the Random Forest Regressor model has R2 score of **training** as 0.83. This indicates that there is a huge difference in the training and testing data prediction results, showing that our models **do not generalise** well. Since the training prediction scores are more in both the model, we conclude that these both model as **Overfitting**. Although the difference in training and testing data R2 scores is less in the SVC RBF model, hence this indicates that we must use the SVC RBF model as the BEST model.
- Viewing the table again, it can be observed that the SVC RBF model have R2 score of **unseen** as 0.36, whereas, the Random Forest Regressor model has R2 score of **testing** as 0.33. This indicates that SVC RBF must be used as the BEST model, as its R2 score is higher.
- Same pattern was observed for **testing, training and unseen** data in the Mean Squared Error for both the models.
- Even though the accuracy of the Random Forest Regressor model for testing data was more and error rate was lesser, SVC RBF model showed better accuracy and lower error rate for unseen data. Plus, the latter model also showed lower overfitting characteristics than the former model.

BEST MODEL: After observing all the facts and keeping all the above factors in mind, we conclude **Support Vector Regression** having '**C**' as **1000**, '**gamma**' as **1** and '**kernel**' as '**rbf**' to be our BEST MODEL that we would use and recommend in the real-world setting with the provided dataset, which uses 23 features with best cross validation scores and was normalised using **MinMaxScaler** technique.

SUMMARY

We performed a thorough examination of the regression machine learning algorithms in different settings, explored data to remove skewness and correlations to find the model that should be used for the provided dataset. After performing the feature selection, hyperparameter tuning, k-fold cross validations, testing possible regression algorithms, and evaluating them by checking the Overfitting/Underfitting and comparing the Mean Squared Error and R2 Scores of each of the models (shown in Fig. 9), the report concludes that **Support Vector Regression** having '**C**' as **1000**, '**gamma**' as **1** and '**kernel**' as '**rbf**', as the BEST MODEL for predicting the target value in the provided dataset, which was normalised using **MinMaxScaler** technique.

APPENDICES

Figure 1: Histogram of all the participating features of machine learning model

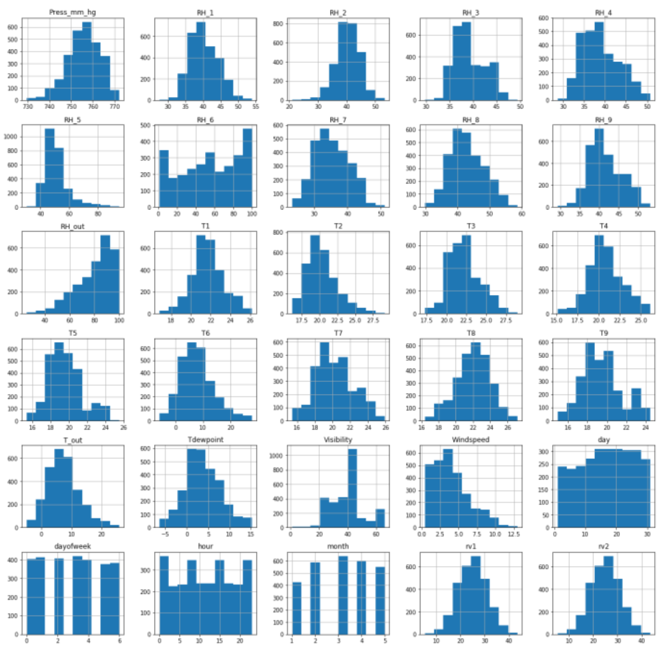

Figure 2: Heatmap showing the correlations amongst the selected feature in the dataset

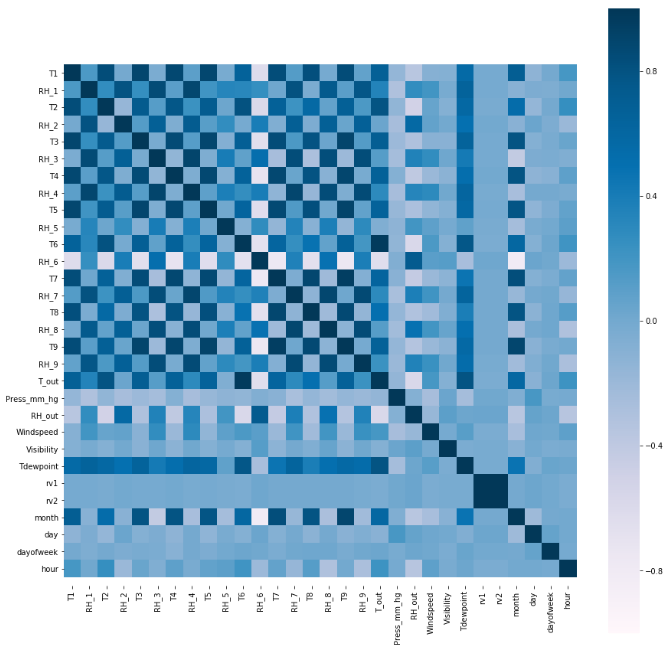

Figure 3: Boxplot of all the participating features of machine learning model

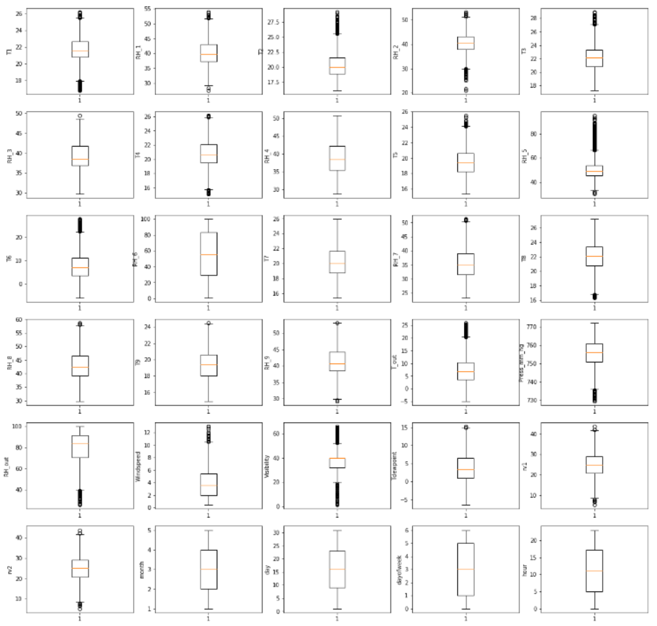

Figure 4: Scatterplot of all the participating features v/s target of machine learning model

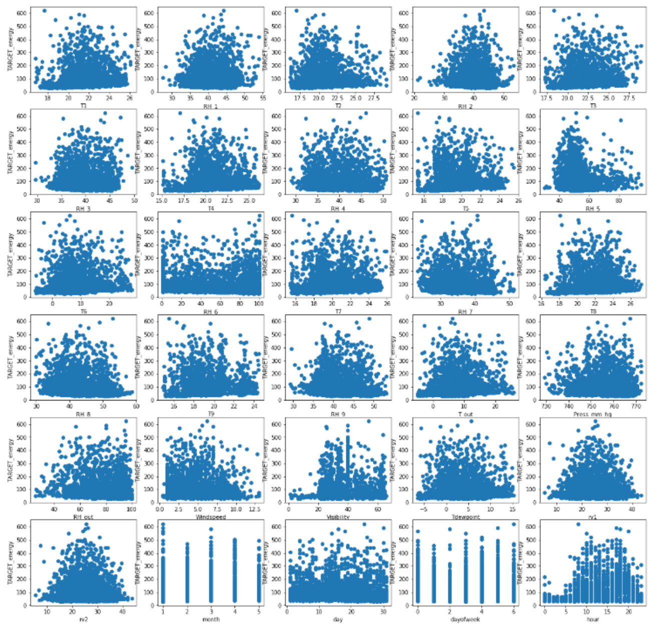

Figure 5: Histogram of the target of machine learning model

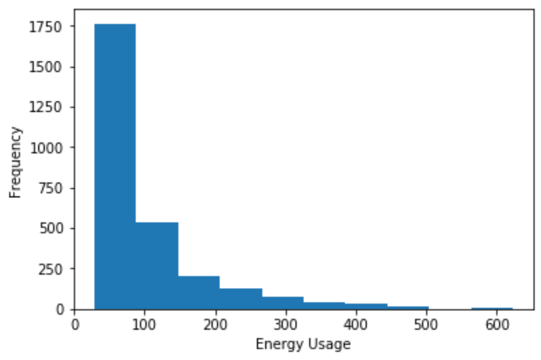

Figure 6: Box of the target of machine learning model

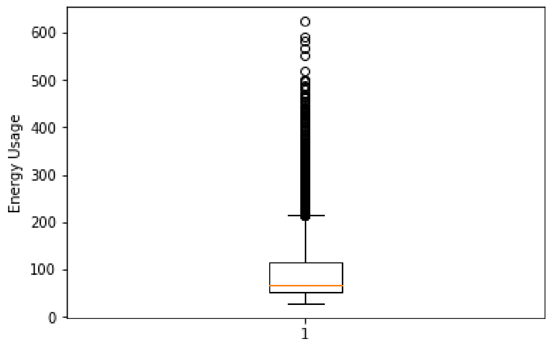

Figure 7: Plot showing the accuracy rate w.r.t. the number of features (SVR Feature Selection)

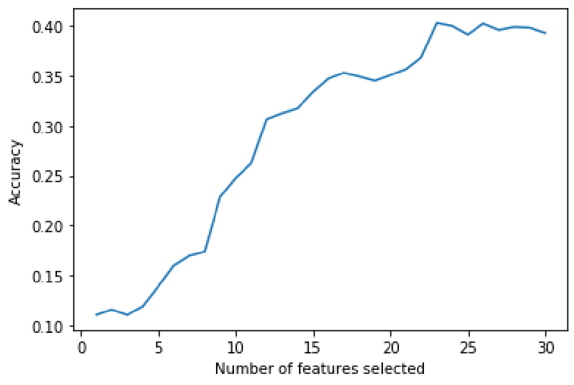

Figure 8: Plot showing the accuracy rate w.r.t. the number of features (RandomForestRegressor Feature Selection)

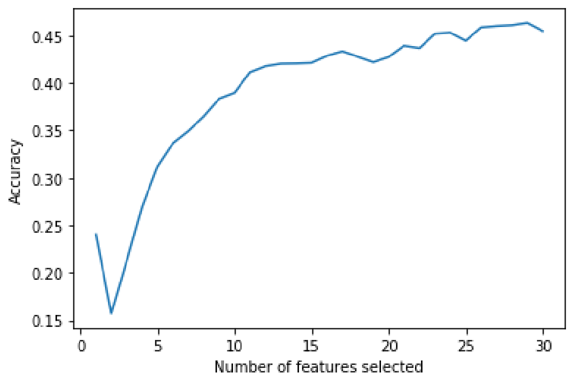

Figure 9: Table containing the Mean Square Error and R2 Score of different regression models

Regression	MSE (Testing)	MSE (Training)	MSE (Unseen)	R2 Score (Testing)	R2 Score (Training)	R2 Score (Unseen)
SVC RBF	3959.203903	2620.277081	4475.95739	0.403860	0.624614	0.365825
Random Forest Regressor	3806.098731	1153.559067	4689.32778	0.426913	0.834739	0.335594