Assignment 3

Name: Smitesh Nitin Patil Class: 1CSD1

Student Id: 22223696

For this machine learning task, a regression problem was given, here I have documented my efforts in creating two regression models for predicting the target variable electricity consumption.

1: Machine Learning packages and their description

There are many open-source python packages readily available for developing the complete machine-learning pipeline. From data pre-processing to model evaluation metrics. For the machine learning task, the sci-kit learn[1] package was used. Sci-kit learn has sub-packages for many regression ML models like MLPRegressor, Linear Regression, Gradient-boosting regressor, etc. Also, it is relatively easy to evaluate your models as it has many evaluation metrics for regression models predefined in the library like RMSE(root mean squared error) and R2 error.

From the sklearn library, two regression models were used

1. K-nearest neighbors Regressor[2]
2. Random Forest Regressor[3]

1. K-nearest neighbors: KNN is a supervised learning algorithm, KNN utilizes distance measuring metrics to make predictions on samples. In KNN, each training sample is suspended in a vector space dimension. Its position depends on the independent variables or features. Each testing sample is then suspended in the same vector space. Consequently, N-nearest training samples closest to the testing sample are taken into consideration. In the case of classification. The testing sample is assigned the label that is the most frequent for its n-nearest neighbor. In the case of regression, we simply take the average of the n-nearest neighbors.

KNN is also called a lazy learner. A lazy learner generalizes the training data during the testing phase. In other words, during the training phase, KNN just stores the data, and regression is done during the test phase.

KNN model was created using the KNeighborsRegressor object in sklearn.neighbors package. As for the hyperparameters we used the value of n as 20 and Minkowski as the distance measuring unit. i.e., the model will select 20 nearest sample points for making predictions.

2. Random Forest Regressor: Random Forest is one of many ensemble methods. It derives from the decision trees; it uses many weak trees and combines them for better prediction.

Bagging is a process in which we subset our dataset for the many random decision trees we make, each tree would have its unique set of data. We do this because a single decision tree is susceptible to training data i.e., the tree formed is highly dependent on the order of data. After the decision trees are made, the individual decisions are aggregated to a single value in the case of regression it is the mean value. This process is called bootstrapping. So basically, bootstrapping = (bagging + aggregating).

A random forest model was created using the RandomForestRegressor object in sklearn.ensemble package. As for the hyperparameters we used the value of trees created(n\_estimators) as 200.

2 Methodology followed for two models and hyperparameter tuning:

Data Pre-processing: First, the dependent variable var2 is a categorical attribute, it needs to be one-hot encoded as numerical values. OneHotEncoder from sklearn.preprocessing was used to convert the var2 into Var\_A, Var\_B, and Var\_C. As the Var2 is no longer needed it is discarded along with the ID variable as it doesn’t relate to the value of electricity\_consumption. The DateTime column is converted to the index of the dataframe. Also, new columns with year, month, day, and hour were added from the DateTime column.

Chart, histogram

Description automatically generated

The above graph shows training data for the first month the gap is for the duration of 24th of the month to the end of the month on which we must make predictions from the testing data available in test.csv

Data were normalized for both train and train by using a standard scaler. Standard scaler scales the features such that the mean of the data is zero and the standard deviation is 1.

The hyper-parameters were selected based on how the model performed on validation data for each fold. After setting the hyper-parameter the one that gave the highest r-squared values was chosen. Hyper-parameters tuned for Random Forest Regressor were n\_estimators (number of estimators) and for K-Nearest Regressor it was n\_neighbors (number of neighbors)

Hyperparameter settings for the model:

K-Nearest Neighbour:

n\_neighbors = 20 # number of samples considered for predicting the output

metric = ‘Minkowski’ # distance measuring metrics

Random Forest Regressor:

n\_estimators = 100 # no of trees

max\_depth = 10 # used to set the depth of the estimators helps with overfitting

3. Implementing cross-validation for time-series data:

The cleaned data after pre-processing was passed to both the random forest and k-nearest neighbors, for hyper-parameter tuning 5-fold cross-validation was used. Especially, data was split by considering time-series. As the data is a time series we cannot use future data to make current forecasts. Hence, we used TimeSeriesSplit() function. In a time series split, we maintain the order of data while cross-validating the data, i.e., future data is not considered current predictions.Chart, bar chart

Description automatically generated Chart, bar chart

Description automatically generated

Image ref: https://medium.com/@Stan\_DS/timeseries-split-with-sklearn-tips-8162c83612b9

Underfitting of the model was checked by the cross-validation scores obtained while training the model on various hyper-parameter. Metrics RMSE and R2 were used to measure the model accuracy during the validation.

A model is said to be overfitting if it performs well during the training phase and badly on unseen data as the test.csv file provided doesn’t contain values of electricity consumption to check if overfitting is occurring. The best we could do is check if our models capture the trends of the data.

First Months Data

Chart, histogram

Description automatically generated Chart, histogram

Description automatically generated

Second Month Data

Chart

Description automatically generated Chart

Description automatically generated

The following two graphs are for the first month of the dataset. You can see the predictions in orange, the predictions for the K-nearest regressor seem to be fitting the trend much better whereas for the Random regressor it seems to be getting higher peaks thus overfitting seems likely here. However, for the second month, both models seem to be capturing the trend. It is hard to quantify overfitting without labeled testing data to cross-check.

4. Evaluation of models and conclusions

Performance metrics RMSE(Root mean squared error) and R2 error were used to evaluate the validation set from the data.

Root Mean Squared Error: It calculates the deviation of the predicted value from its original value . RMSE penalizes data points with a higher deviation from original results more than with a small deviation.

R-Squared: R-squared error or coefficient of determination is the ratio of the sum of squares residuals and the total sum of squares subtracted by 1. It gives the proportion of the dependent variable (electricity\_consumption) that is predictable from independent variables.

Text

Description automatically generated Text

Description automatically generated

K-Nearest Neighbours Random Forest Regressor

From the cross-validation scores obtained after hyper tuning the parameter, it can be concluded that models perform relatively similarly on their respective validation sets. Random Forest seems to be overfitting as it doesn’t capture the trend of the data that K-Nearest Neighbours seem to do well. However, it is hard to quantify if the results are overfitting as we don’t have test data labels to measure if the model is performing well.

References:

1. <https://scikit-learn.org/stable/>
2. <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html>
3. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>