Using Machine Learning to Predict the Energy Output of a Combined Cycle Power Point

Xuyang Fang

I. Introduction

A Combine Cycle Power Plant (CCPP) is an ensemble of heat engines that are used to efficiently generate electrical power [1]. To ensure the economic and operational efficiency of a CCPP, it is important for engineers to accurately simulate its energy output [2].

Traditional techniques that are used to predict the energy output of a CCPP are usually based on systems of Partial Differential Equations (PDEs) [3]. However, due to the difficulties of solving these PDEs, either numerically or analytically, engineers and researchers are turning their attention to techniques in the realm of Machine Learning [3].

In 2014, Tufekci et al released the CCPP dataset [2]. The dataset contains 9568 data points in total. These data were collected as hourly averages from a confidential CCPP out of 674 days between 2006 and 2011. During those 674 days the CCPP was fully operational without any anomalies. Each data point in the dataset contains 5 variables. They are Ambient Temperature (AT), Atmospheric Pressure (AP), Relative Humidity (RH), Vacuum (Exhaust Steam Pressure, V) and Full Load Electrical Power Output (P_E). The dataset contains no duplicates or missing values.

The aim of this coursework is to use ML techniques to predict target P_E given feature AT, AP, RH and V in the CCPP dataset. The objectives of this coursework are the following:

1) Determine the type of the ML models to use, i.e., whether models from Classification, Regression or Clustering are the most suitable to achieve the aim.
2) Determine the metrics that would be used to evaluate the performance of the models.
3) Choose a baseline model.
4) Determine the specific ML models to use.
5) Train those models and optimize their performance via hyperparameter tuning.
6) Design then conduct a testing strategy that would show the robustness of the models.

For the rest of this section, the execution of objective 1) will be discussed. The execution of the rest of the objectives will be covered in the Method section. All results will be displayed in the Results section. The results will then be further evaluated in the Analysis section. All code for this coursework was written in a Python Jupyter notebook using packages such as sklearn, scipy and matplotlib.

A. Select The Type Of Algorithms

The type of algorithms to use for this problem was determined by inspection on the CCPP dataset given. First of all, every single data point in the dataset contains the target PE, which is also the value that the aim is trying to predict. Therefore, the most straightforward way to solve this problem using ML is to:

- assume that there is some true function f such that $f : (AT, V, AP, RH) \rightarrow PE$.
- find a hypothesis function h that best approximates unknown true function f

which is known as the the task of Supervised Learning [4]. Therefore, Clustering algorithms can be ignored since they are the most commonly used for Unsupervised Learning [4]. Since PE is a continuous variable, rather than one of a finite set of values, it is a Regression problem [4].

II. Methods

This section outlines the executions that were carried out to achieve the objective 2) to 6).

A. Select Performance Evaluation Metrics

Following section I.A, for regression models/regressors, typical performance evaluation metrics used are the Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error

(RMSE) and Residual Squared Error (R2). MAE calculates the unweighted average of the absolute distance between predicted and actual target values. MSE calculates the unweighted average of the distance squared between the predicted and actual target values. RMSE is the squared-root value of MSE. R2 measures how well the model prediction explains the variance of the actual target values.

For this coursework, MAE, MSE and R2 are chosen as the evaluation metrics. This is because all 3 quantities measure different aspects of the performance of the models. MAE measures the overall error of the model whilst MSE penalises the outliers more than MAE (due to the squaring of the error). R2 measures the overall goodness-of-fit of the model prediction. RMSE is not chosen as it also penalises the outliers similar to MSE.

B. Model Testing Strategy

To prove the robustness of the model, it must be perform *well* on test data that it has *never* seen before. First, this means that the data that are used for testing must never be used for training. Second, the training set must be large enough so that the most optimal model with the most optimal hyperparameters can be found. Therefore, I have split the dataset into a training set and a test set with 90-10 split.

It is important that the data distributions of the variables in the training set is consistent with those of the test set. This is to ensure that that the model should generalise well on the test set after it has been trained with hyperparameter tuning on the training set. To ensure the consistency between the distributions of target variables between the training and test set, the target variable was first put into 20 bins. Then the dataset was split by applying the 90-10 train-test shuffle split on every single bin. For more details of this please read stratified k-fold section on sklearn user guild on cross validation [5]. However, this process cannot ensure that the distribution of features between the training and test set are consistent.

C. Model Training Strategy

1) Hyperparameter Tuning: In ML, hyperparameters are the configurations of a ML model that have to be set before training starts. Different choices of hyperparameters can lead to drastic change of the performance of the model on the test set. Therefore, it is important to select the most optimal values of hyperparameters to find the model that performs/generalises the best on the test set (according to the evaluation metrics). However, the test set should never be used to for hyperparameter tuning as this would lead to data leakage [4], which over estimates the performance of the model. To avoid peeking, the training data (selected according to Section II.B) should be further split into a train and validation set, which will be used for all hyperparamter tuning procedures.

In this coursework, hyperparameter tuning is performed via the grid search method. If a model has multiple hyperparameters, then for each one of them, pick several arbitrary values. After that, find the combination of hyperparameter values that leads to the best performance on the validation set.

2) K-Fold Cross Validation: Splitting the training data into one train set and one validation set is that has several disadvantages. If the train set is too small (say 10%), it may lead to a poor estimation of the actual performance of the model [4]. If the test set is too big (say 50%) then the train set is too small to find the optimal model [4]. In order to efficiently use the training data whilst reduce the luck factor, the training data were split into 5 folds like Fig. 1. The training of one model with one hyperparameter combination is divided into 5 rounds. For each round, the model will be trained on the green folds in each split then validated on the blue fold. The MAE, MSE and R2 on the green folds and the blue fold will be computed. Then the model will be reset to its initial state with the same hyperparameter combination before the next round of training starts. Finally, after 5 rounds of training is over, the MSE, MAE and R2 will be computed and used as an estimation of the performance of that model.

D. Selecting a Baseline Model

Selecting a baseline model for this coursework has two purposes. First, it provides a performance benchmark based on very crude assumptions of the data. Second, it serves as a "sanity check" tool such that, if the performance of a more complex model cannot beat the baseline performance, then it may indicate that there is a bug in the more complex model, or the dataset is very imbalanced.

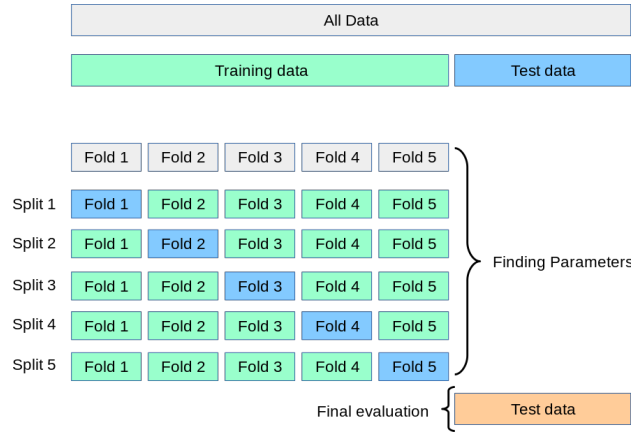

Fig. 1: Schematic Diagram of Cross Validation. Image adapted from sklearn documentation [6].

The choice of the baseline model for this coursework is the sklearn.dummy.DummyRegressor with strategy 'mean'. This regressor assumes that the target variable is only dependent to its own distribution whilst completely dependent from all of the presented features.

E. Model Selection

According to [4], to find the best model, it is better to start with simpler ones that will most likely underfit the data, gradually increase the complexity of the following choices of models, until the error on the train set plateaus. Then finally, pick out the model with the best validation score.

For this coursework, I choose to start with the DecisionTreeRegressor from sklearn as it is a simple model with great interpretability (can be rewritten into a series of rules). In order to reduce the overfitting and increase the overall performance, I choose to move on towards the RandomForestRegressor from sklearn. As an ensemble method, a random forest makes predictions based on the averages of multiple tree within itself. Therefore, it should be a more powerful model than a single decision tree.

III. Results & Analysis

The validation score of the baseline model is the following: R2=0.0, MAE=14.8, MSE=291.3. This was beaten by every single model presented in Fig. 2 and 3. The best result of Fig. 3 beat Fig.2, which shows a forest of deep trees can reduce the overfitting of each tree within the forest which leads to an increase in performance.

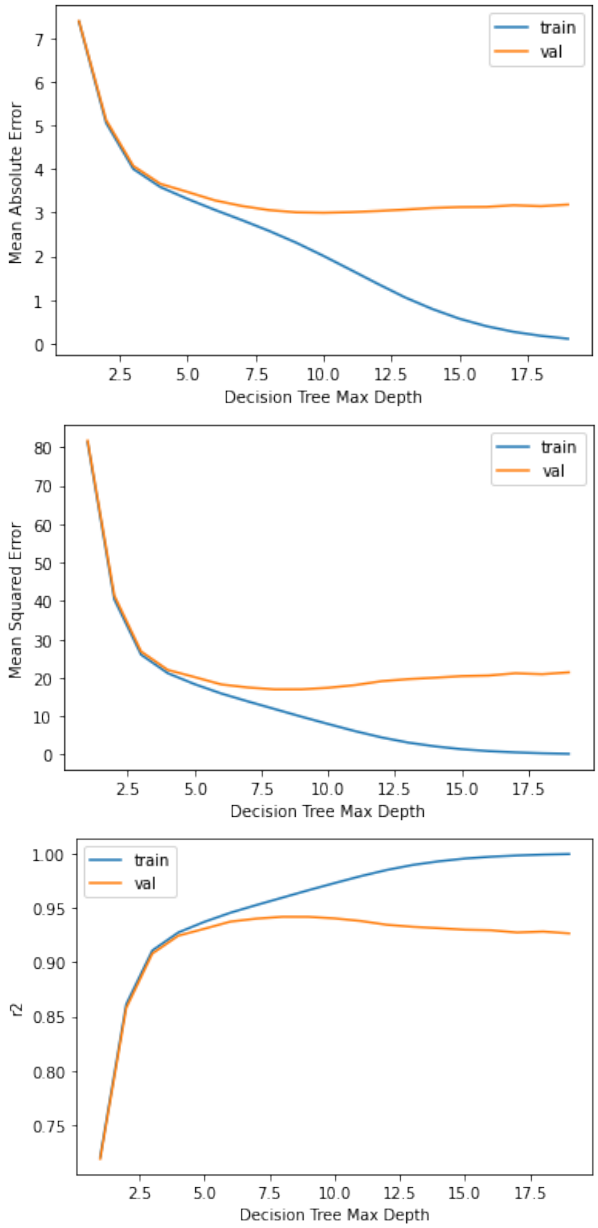

Fig. 2: Plots of the max depths of a decision tree (with no pruning) against its evaluation metrics. Best score: MAE=3.00, MSE=17.0, R2=0.94, at max depth=10, 8 and 8 respectively.

The best performance on the validation set, MAE=2.48, MSE=12.0 and R2=0.96 is achieved by a random forest with max depth=19, no pruning with 20 trees. This model is chosen to be the only model that is allowed to make a prediction on the test set. On the test set, it achieved R2=0.96, MSE=12.4 and MAE=2.41, which is consistent with the validation results.

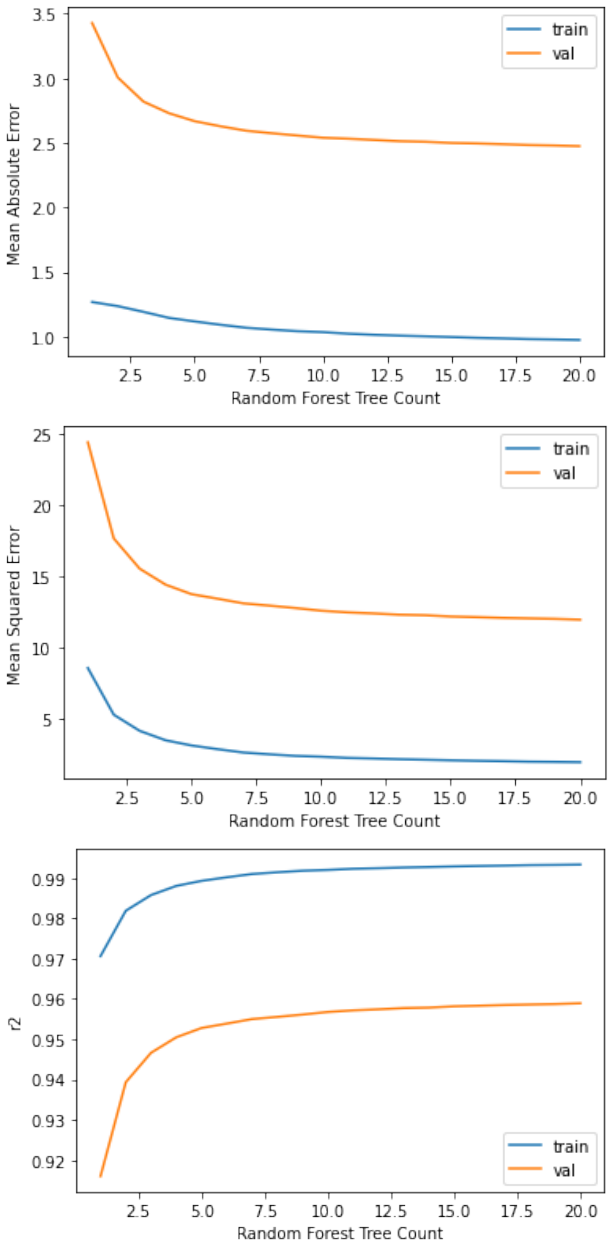

Fig. 3: Plots of no. of trees in a random forest (max depth=19, no pruning) against evaluation metrics. Best score: MAE=2.48, MSE=12.0, R2=0.96 with tree count=20

REFERENCES

[1] G. G. Power. Combined cycle power plant: how it works. Https://www.ge.com/gas-power/resources/education/combined-cycle-power-plants.

[2] P. Tüfekci, "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods," *International Journal of Electrical Power & Energy Systems*, vol. 60, pp. 126–140, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0142061514000908

[3] U. Kesgin and H. Heperkan, "Simulation of thermodynamic systems using soft computing techniques," *International Journal of Energy Research*, vol. 29, no. 7, pp. 581–611, 2005. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/er.1095

[4] S. J. Russell and P. Norvig, *Artificial Intelligence: a modern approach*, 3rd ed. Pearson, 2009, ch. 18.2.

[5] Sklearn cross validation. Https://scikit-learn.org/stable/modules/cross_validation.html#stratification.

[6] Cross-validation: evaluating estimator performance. Https://scikit-learn.org/stable/modules/cross_validation.html#k-fold.