**COSC2793 Computational Machine Learning**

**Assignment 2 – Project 2: Predict Energy Use**

#### Data Preparation

##### Read Data & Initial Data Inspection

* Starting from loading libraries, unzip the data zip file, and retrieve the data from the CSV file.
* Check the data shape and info because it helps us understand the structure and content of the dataset (e.g., all the features we found are numerical attributes, except for one object type, where we converted to datetime).

##### Data Cleaning

* Verified that there are no missing or duplicated values to secure the data is clean, it ensures the accuracy of the analysis and models.
* Reviewed column names for future reference because it helps in identifying and manipulating specific columns later.

#### Data EDA

##### Time Series Encoding and Feature Addition

* Encoded the 'date' column and set it as the index because it allows us to work efficiently with time-series data.
* Added year, month, weekday, day, and hour features because these time-related features can reveal patterns and trends (e.g., how energy consumption varies by hour or day).

##### Overall Distribution Analysis

* Upon overall exploration, histograms are generated to help understand the distribution, skewness, outliers contribution, and the range of the data (e.g., identifying what variables need to be transformed).
* Separated data into temperature, humidity, and other categories because it helps in the focused analysis on specific aspects, many temperature-related features show a linear relationship with ‘TARGET\_energy’ attribute. As the temperature value increases, the energy consumption tends to raise, while Humidity-related features show a weaker correlation with ‘TARGET\_energy’ attribute compared to temperature features in general. Hence, the scatter plots are more dispersed. In addition, outliers are more prevalent in humidity data, which we will address later to improve data quality and analysis accuracy. *(Figure 1, Figure 2)*
* Calculated and visualized the skewness because it indicates how much the data deviates from a normal distribution, which is important for applying the right transformations to the models. All columns are calculated and sorted based on the absolute values of their skewness to identify the most skewed features.
* Created box plots to identify the outlier contributions because the outliers can be a significant factor to influent the performance of models which needs to be addressed. *(Figure 3)*

##### Relationship Analysis

* Generated a heatmap to show correlations between different features because it helps identify which variables are strongly related, aiding in feature selection. *(Figure 4)*
* Plotted average TARGET\_energy consumption across different months and hours because it reveals time-based consumption patterns, useful for predicting energy usage. *(Figure 5, Figure 6)*
* Mapped and visualized TARGET\_energy consumption across different weekdays using box plots because it shows how consumption varies by day, highlighting patterns that can be leveraged in forecasting. *(Figure 7)*
* Dropped columns by checking their unique count and low relation from previous analysis, e.g.yy refers to year and weekday refers to the number of days.

#### Feature Transformation

##### Outlier Removal

* Carry out the outlier removal process to reduce data variability and prevent overfitting in the model, this step improves data integrity and model accuracy. We calculated and defined the lower and upper bounds of the data frame referred to IQR, and recursively replaced the current bounds with the latest limits by using a function. After this, outliers should be less. *(Figure 8)*
* We created boxplot before and after to see how well the outlier removal is done, and it has been successful. We can continue using the filtered dataset.
* We also created a HeatMap plot to see the correlations between data pairs. Based on the plot result, we removed ‘rv1’, ‘rv2’, ‘dd’ columns as the correlations between these columns and the target column are extremely low has nearly no impact. *(Figure 9)*

##### Power Transformation

* Check the feature skewness to determine the distribution by separating the data into symmetrical and skewed categories with a skewness threshold (0.5). Handling skewed data helps in preprocessing and improving the model performance.
* Then we applied PowerTransformer to the skewed features to make their distribution more normal because the transformation stabilizes the variance and minimizes the skewness. The transformed dataset is ready for modeling.

##### Data Scale

* Apply StandardScaler to standardize the concatenated data to have a mean of 0 and standard deviation of 1 to ensure all features contribute equally to the model and improve the training performance. This additional part can be referenced later in some models when needed as their accuracy is not high enough. *(Figure 10)*

#### Model Development and Evaluation

##### Model 1: Decision Tree

* Splitting Data with the pre-processed transformed data into Train/Test model, and then apply the feature scale using StandardScaler for Train/Test/Validation data. The histogram graph comparisons distributions are presented. *(Figure 11, Figure 12)*
* We plotted decision tree model to see the distributions. We could see that the data splitting is bad, and the entropy values are all high, indicating impurity of the dataset. F1-Score was also calculated based on the model, which was lower than 0.1 for both training and validation datasets, revealing poor performance.
* We tend to try to improve the model by applying Hyper parameter tuning, and then we chose to use DecisionTreeClassifier to get the best performance. We attempted to get the best ‘f1\_macro’ score with its optimized depth number and estimations by setting the parameter to be (2, 400, 50) implies from range 2 to 400, increase depth in unit of 50. It gives a score result 0.2 which is not satisfying, therefore, we moved to explore the next model.

##### Model 2: Regression Model – Linear Regression

* We trained this regression model, which is sensitive to feature scales from discussed in class, using the standardization to scale the training data in the ratio of 8:1:1 for training, validation, and test, respectively.
* We calculated RMSE and R² score on the validation set to evaluate its accuracy, while R2 score tells the correlation between x and y values, and RMSE quantifies the difference between them.
* In this case, the model's predictions are off by about 37.10 units from the actual values. Approximately 28.23 percent of the variance in the dependent variable (target) is predictable from the independent variables used in the model. This poor performance is reasonable, as time series data rarely follows a linear pattern based on our prior exploration. Hence, we moved to explore the next model.

##### Model 3: Regression Model 2 – Polynomial Regression (2,3,4)

* As a variant model from linear regression, we used the same scaled data to test polynomial regression with the degrees of 2, 3, and 4. This approach provides an initial comparison with the linear Model.
* The results show that the model with degree 3 is more reliable than t degrees 2 and 4, with the R² score approximately 15 percent higher than that of degree 2 and a lower RMSE by about 6. *(Figure 13)*
* We plotted the predictions versus the actual values with degrees 2 and 3 to investigate further. Degree 4 attribute was excluded from the plot due to its poor performance, indicated by extremely high RMSE and negative R² scores, which implies the model is more overfitting. *(Figure 14, Figure 15)*
* The residuals were plotted to assess the spread of prediction errors for each degree of the polynomial regression. For the model with degree 2, the residuals appear scattered with noticeable variance, suggesting a moderate level of prediction. In contrast, for degree 3 model, the residuals are more concentrated around the zero line, indicating fewer large errors so that a potential better fit to the data performance. This suggests that the degree 3 model exhibits a better balance between bias and variance, fitting the data more accurately than the model with degree 2. Despite the improvement shown with degree 3 model, the accuracy still needs to be improved, suggesting exploring alternative models or techniques to enhance predictive performance further.
* Combined with the accuracy score, we finally decided to use the degree 3 transformed data. However, it is still not good enough which leads to another model exploration.

##### Model 4: Baseline Model 1 – Random Forest Regression

* Through research, we found that Random Forest Regression is frequently used in various domains due to its ability to handle datasets containing continuous variables, manage complex data structures, and mitigate overfitting. That is the main reason why we chose this model for its high performance in general. *(Figure 16)*
* Using the polynomial degree 3 transformed dataset, we initially set ‘n\_estimators’ to 20 and ‘random\_state’ to 42 for our tests. ‘n\_estimators’ determines the number of trees in the forest, and ‘random\_state’ ensures reproducibility.
* Again, we evaluated the model using RMSE and R² score for consistent comparison. Surprisingly, the R² scores for both validation and test data exceeded 75 percent, with only a 0.3 percent variance between them. This indicates the improvement of the model's reliability.
* From previous training in different models, we eventually determined to deepen learning this model, hence, we will use polynomial linear model as our baseline model. Although the difference of RMSE and R2 score between validation and test data is minor, a Diagnostic Plot for these two datasets would be helpful to guarantee the unnecessary of regularization.

#### Judgement and Independent Evaluation of the Baseline Model 1

##### RF Regularisation and Diagnostic Analysis

* When considering the necessity of regularisation, defining the gap between the validation and test datasets is crucial. Given the minor differences in RMSE and R² scores between the validation and test datasets, we performed a detailed diagnostic analysis.
* The distribution of residuals for both datasets was plotted to ensure they followed a similar pattern. The results confirmed that approximately 99% of the patterns overlapped, supporting the initial assessment of model performance. *(Figure 17)*
* Additionally, we visualized the first 100 predicted values against the actual values for both validation and test datasets. This visualization revealed that the validation predictions exhibited slight overfitting compared to the test data, indicating a potential need for hyper-parameter tuning. *(Figure 18)*
* Despite these observations, the diagnostic plots later confirmed that the predictions closely follow the actual values, reinforcing the model's stability. Consequently, these findings support the decision to use the Random Forest Regression model as our baseline, while highlighting areas for potential refinement through hyper-parameter tuning to address any overfitting tendencies. *(Figure 19, Figure 20)*
* As a part of our model verification process, we retrained and evaluated our baseline model using the 5-fold cross-validation method. We employed a Random Forest Regressor with 20 trees and a fixed random seed to ensure reproducibility. The results from our cross-validation and performance metrics calculation are consistent with our previous calculations, affirming the model's performance stability. *(Figure 21)*
* Building on our initial cross-validation (CV) steps, which provided consistent and reliable performance metrics, we further optimized the Random Forest Regression model using Grid Search with cross-validation to fine-tune its hyperparameters. A parameter grid with varying n\_estimators values and a fixed random\_state was defined. By employing GridSearchCV with 5-fold cross-validation, we systematically trained the model on polynomially transformed training data. The optimal hyperparameters were identified based on the best cross-validated score, resulting in a notable improvement in accuracy from 75% to nearly 82%. This enhancement demonstrates the efficacy of hyperparameter tuning in refining model performance and underscores the robustness of our approach. The final evaluation confirmed the model's increased accuracy, mean absolute error reduction, and higher R² score on the test set, validating the effectiveness of our optimization strategy. *(Figure 22)*

##### 5.2 Research Method 1: RFE Feature Selection To refine our Random Forest Regression model for time series data, we employed Recursive Feature Elimination (RFE) to select the most significant features. Unlike Principal Component Analysis (PCA) or mutual information criteria, RFE recursively removes the least important features and builds the model iteratively, aligning well with the decision-tree-based nature of Random Forest.

* We first transformed the training data into a DataFrame and defined a base Random Forest model with 20 trees. Using RFE, we aimed to select the top 10 features. We performed walk-forward validation with 5-fold TimeSeriesSplit, fitting RFE on the training data for each fold, selecting the most important features, and training the Random Forest model on these features.
* However, the process proved to be computationally intensive, resulting in a lengthy running time.

##### 5.3 Research Method 2: RandomizedSearchCV

* According to the performance and insights from previous work, we conducted an advanced hyperparameter tuning process to further optimize our Random Forest Regression model. Initially, we used RandomizedSearchCV to explore a wide range of parameters, including n\_estimators, max\_features, max\_depth, min\_samples\_split, min\_samples\_leaf, and bootstrap. This step identified a promising set of parameters, which were then fine-tuned using GridSearchCV. By narrowing down the parameter grid based on the results from the randomized search, we conducted a more focused optimization. The refined model showed a significant improvement, with the accuracy increasing over 85%.
* Inspiration for this process came from the detailed guide by Will Koehrsen, published in Towards Data Science, which emphasizes the crucial role of hyperparameter tuning in machine learning. Koehrsen’s work illustrated that while gathering more data and feature engineering usually yield the greatest improvements, hyperparameter optimization is essential when these avenues are exhausted. The use of Scikit-Learn’s tools, such as RandomizedSearchCV and GridSearchCV, provides an efficient way to enhance model performance by systematically exploring and refining hyperparameter combinations. By following this method, we were able to achieve a significant performance boost, demonstrating the power of well-executed hyperparameter tuning .

#### 6. Research and Evaluation of Published Works

##### 6.1 Build Models

* For project 2, we are required to conduct an independent evaluation on researching several published works that used this dataset. Out of a various of searching, we found that LSTM Model and Boosting Model are widely used, and also have outstanding results.
* We created these two models by learning the method from the published works with our dataset.

**Research Model 1: LSTM,baseline model 2**

* This is a neural network based algorithm, therefore, we are curious to investigate into the model learning. Firstly, load the relevant libraries. Then define the function to convert series to supervised learning dataset for model training, this is for the predicting on time-series sensitive dataset. Followed by it is to normalize the features by using MixMaxScaler. Transform the time-series data into a supervised learning model using the function before.
* ‘reframed’ data set is the filtered data with some redundant columns dropped depending on the checks on the important ‘values’ dataset size. Then the input feature ‘X’ and output feature ‘Y’ are defined as subsets of these values.
* Split Data into training and testing sets, and then reshape ‘X’ and ‘Y’ into 3D arrays with dimensions ‘[samples, timesteps, features]’ as required by LSTM.
* Fit and train the LSTM model with some parameter settings. We can get the predict results in forms of R2 Score and plot evidence. The plot shows two lines, one line represents the true values, and another represents the LSTM model’s predictions.
* For achieving the best performance of the LSTM model, we tried multiple hype-parameter tuning adjustments such as adding layers and modifying the parameter, and believed that three layers and 100 epochs give the best results, which shows as the figure in the code output.

**Research Model 2: Gradient Boosting**

* The second model is also popular, we have found this on various resources. Firstly, we created the boosting model by learning from their methods using the transformed data from before. Then we decided to implement PCA procedure to enhance the performance as it is not ideal.
* For PCA implementation, necessary libraries are imported, and the scaled data (referring to 3. Feature Transformation) is fitted with PCA model. It helps in reducing the complexity of the data.
* Assign PCA-transformed data to ‘x’ and ‘TARGET\_energy’ to ‘y’ and then conduct train/test splitting.
* At last, we built the Gradient Boosting model with PCA and trained it with the training data. We get the prediction results in the forms of MSE and R2 Score.

##### 6.2 Comparison between LSTM and Gradient Boosting

* First of all, LSTM model is a type of neural network based algorithm, while Gradient Boosting is an ensemble learning method. Secondly, LSTM modes is primarily used for time-series model forecasting, and Gradient Boosting model mainly focuses on classification and regression task.
* Compare the results between LSTM and Gradient Boosting models, which can be interpreted in the cast of their R2-Scores. The LSTM model can achieve and beyond a score of 80 percent, while Gradient Boosting model stays around 65 percent. Hence, LSTM model is obviously more suitable for the energy prediction than Gradient Boosting model.
* Conclusion: A higher R2-Score for the LSTM model suggests better performance than Gradient Boosting model, this is due to the nature of the project dataset. The energy dataset is time-series based data where LSTM is designed to handle sequences and time dependencies. Also, for more complex multivariate time series interactions, LSTM model can handle better with how the time- series data influence the target variable. Furthermore, LSTM model can work directly with raw time-series data, while gradient boosting requires significant features to transform the data into a suitable format, which may potentially lead to suboptimal performance due to missing features.

##### 6.3 Comparison between Our Work and Existing Work

* Until now, the final outstanding models are presented – Random Forest model from our baseline model and LSTM model from the researching evaluation. We carried out a comparison of each model in details.
* Regarding the Random Forest Model, the simple model achieves a R2 Score of 75 percent for both validation and testing datasets, and it goes above 80 percent after applying hyper-parameter tuning. Same with LSTM, the original try gives a R2 Score of 76 percent for both training and testing datasets, and it also pushes above 80 percent when the appropriate parameters are chosen. This means that both models are successful and can be taken as the baseline model for training and prediction. *(Figure 21)*
* Despite the performance, time consumption is another key point to consider about. LSTM finishes training in minutes while Random Forest might end cost up to 80 hours. Random Forest consists of many individual decision trees where each tree is trained independently on a random subset of the data, this leads to heavy computational loads. LSTM is a type of recurrent neural network that designed to handle sequence data and capture time-series dependencies, this is more efficient as it models temporal dependencies directly.
* Based on our research and multiple testings, the conclusion is that LSTM model is the best baseline model in overall performance considering the accuracy and time-efficiency.

#### 7. Challenges

* One critical challenge that we faced was the inability to fully execute the entire code file in google Colab, which takes almost one day to complete execution. Testing each model from the notebook file to capture the highest accuracy, we had to separate them into multiple file copies and ran specific sections, it is indeed time consuming.

A close up of a text

Description automatically generated

*Figure 1 Figure 2*

Attempt to fix this, we used both Google Colab and Jupyter Notebook for execution, we ran shorter tasks on Colab as its cloud RAM gets to run out, and complex process such as Random Forest on Jupyter Notebook. This unexpected heavy load process is frustrating for beginners.

* Searching for resources on optimizing features, scaling options, hyperparameter tuning, and parameter assignment beyond classroom materials is also a challenge for us, this requires a lot self-learning outside of class teaching.
* Conducting comparative analysis to evaluate different models and understand their metrics, strengths, and weaknesses. Each model has its own metrics and standards of data input, learning and analyzing all of them to select the best model took us loads of time, as well as training and tuning process. Only after understanding the graphs, logics and results, the strengths and purposes of the models can be interpreted.

A black text on a white background

Description automatically generated

A black text on a white background

Description automatically generated

A white background with black text

Description automatically generated

A close up of a text

Description automatically generated

*Figure 1 Figure 2*

#### Appendices

A grid of red lines

Description automatically generatedA screenshot of a graph

Description automatically generated

*Figure 1 Figure 2*

A group of blue and black boxes

Description automatically generated

*Figure 3*

A colorful grid with different shades of squares

Description automatically generated with medium confidenceA colorful grid with numbers

Description automatically generated with medium confidence

*Figure 4 Figure 9*

A line graph with numbers

Description automatically generated A line graph with numbers and a line

Description automatically generated

*Figure 5 Figure 6*

A graph of energy efficiency

Description automatically generated with medium confidence

*Figure 7*

**A chart of blue and white lines

Description automatically generated with medium confidence**

*Figure 8*

A screenshot of a graph

Description automatically generated

*Figure 10*

A graph of a graph

Description automatically generated A graph with blue and white bars

Description automatically generated with medium confidence

*Figure 11 Figure 12*

A screenshot of a computer

Description automatically generated

*Figure 13*

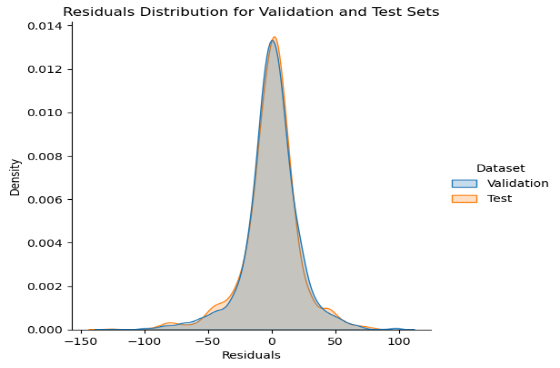
A graph of a graph with green and blue dots

Description automatically generated A graph of a graph with numbers and a chart

Description automatically generated with medium confidence

*Figure 14 Figure 15*

A black numbers and a white background

Description automatically generated

*Figure 16 Figure 17*

A graph of a graph

Description automatically generated with medium confidence

*Figure 18*

A barcode with blue and orange lines

Description automatically generatedA barcode with blue and orange lines

Description automatically generated

*Figure 19 Figure 20*

文本

描述已自动生成图片包含 文本

描述已自动生成

*Figure 21*

A graph of blue and orange bars

Description automatically generated