Summary Report: Temperature Prediction

ID5059 Knowledge Discovery and Datamining | Group Coursework | Group 19

Section 1: Introduction

This project tasked groups of students to create a machine learning model to predict the temperature based on several input features – this a problem which proves especially relevant in weather forecasting. The ERA5 dataset from the European Centre for Medium-Range Weather Forecasts is adapted for use into a dataset with 13,288,920 observations, and 13 features to be used to train the machine learning models. In the language of machine learning this is a supervised, offline, regression task.

This report breaks down the steps taken to create the final machine learning model into four sections: Introduction, where the goal of the project is described; Methods, where the procedures used in preprocessing and modelling are described; Results, where the key results from fitting the models are described, and lastly, Discussion, where the implications from the results are described, and conclusions are drawn.

Section 2: Methods

2.1: Data Exploration

Data exploration is the stage where the training data is analysed to discover underlying trends and distributions contained within the data that can be used to preprocess the data.

The full training dataset (train.csv) is loaded and split into a new training and validation set with an 80 / 20 split, and saved to ensure all models are trained on identical subsets of the data. Data exploration was conducted with the training dataset. The validation set was saved to be used for evaluating model performance.

The structure and quality of the data is assessed by checking for missing values which would hinder model performance. None are found so no steps are taken to deal with this at this stage.

The distribution of the response variable, “t2m” (the average surface temperature, in Kelvin) is examined. It showed an approximately normal distribution ranging primarily between 270K and 300K (-3°C to 27°C), with a peak between 282-285K (9-12°C). This aligns with expected UK climate patterns.

Next, the variables “id” and “valid_time” are dealt with. “id” is a unique identifier and must be dropped in order to stop our machine learning models from overfitting. “valid_time” which is a datetime variable representing when the temperature measurement was taken. The problem with the “valid_time” variable is that it is hard to work with, and although it may contain interesting insights (the time of day is certainly related to the temperature) they remain hard to access. Therefore, the feature is subdivided into five more illuminating features: “year”, “month”, “day”, “hour”, and “season”.

Now the temperature is plotted as a function of the other features. This allows us to view the relationship between the individual features and the temperature. For the majority of the features there is an obvious relationship with the temperature, however, the variable “tcc”, which represents the total cloud cover present at the time of the measurement, appears to be uniform with respect to the temperature. This may serve as an indication that the feature is irrelevant and should be discounted from further analysis. However, we have elected not to do this due to considerations about higher order time effects. Meaning that cloud cover influences the change in temperature i.e. a higher value of “tcc” means that the temperature changes less over time due to the clouds trapping the heat.

Geospatial analysis was particularly important, as weather patterns are inherently tied to location. Temperature maps across the UK revealed clear regional variations with generally warmer temperatures in southern England and cooler temperatures in northern Scotland. The seasonal analysis further showed how these patterns shift throughout the year, with more pronounced north-south temperature gradients in winter and summer compared to the rest of the seasons.

Correlation analysis identified meaningful relationships between temperature and other meteorological variables. Notably, surface pressure (“sp”) showed a moderate positive correlation with temperature, while wind components (“u10”, “v10”, “u100”, “v100”) displayed complex relationships that varied by location and season.

2.2: Data Preprocessing

Based on the data exploration, fundamental preprocessing pipeline principles (for consistency and validity of comparisons) were applied across all models:

1. Temporal Feature Extraction: Decomposed the `valid_time` datetime field into component features (year, month, day, hour) and derived a seasonal indicator to capture cyclical weather patterns.

2. One-Hot Encoding Application: This encoding was applied to features with complicated ordering. Specifically, one-hot encoding was applied to:

- Years: due to only two years of data being provided it made sense to split them accordingly. This data could have been kept as standard numerical values given a wider range of years.
- Seasons, Months, Days and Hours: due to their cyclical nature, i.e. we wouldn’t our model to learn that Dec (12) and Jan as (1) are numerically far apart even though they are actually very close to each other.
- Precipitation Type (ptype): due to the lack of inherent ordering in precipitation types.

3. Numerical Feature Processing: For continuous variables, applied median-based imputation to handle any missing values, followed by standardisation to ensure all features contributed equally to model training regardless of their original scales.

4. ID Removal: Removed the ID field as it contained no predictive information.

The preprocessing approach employed scikit-learn pipelines to ensure consistent treatment of features across training

and testing datasets, preventing data leakage and enabling reproducible results. Although different models used different style of code to obtain processed data, the fundamental steps remained the same – consistency was inspected across all files.

2.3: Architecture Selection

Four regression methods were selected based on the varying levels of computational power available, and over a varying range of model complexity of models to assess for comparison:

1. Linear Regression:

Linear regression is a technique whereby a straight line is fitted to the data. When the model is trained it searches for the gradient and intercept of the line which fits the data best.

2. Gradient-Boosting:

This method works by building a series of simple prediction models (called "decision trees") one after another. Each new tree specifically focuses on correcting the mistakes made by all the previous models. This learning approach allows the system to gradually refine its temperature predictions with each iteration, making it particularly effective for complex weather patterns.

3. Random Forest:

It combines multiple decision trees through bagging (i.e. training the model on different random subsets of the data), thus it is capable of finding nonlinear relationships between input features and target variable [1], mixed data types (including time features), and dealing with multicollinearity (especially when we're doing nearby features).

4. Artificial Neural Network (ANN):

An artificial neural network is a machine learning technique that is inspired by the neurons in an organic brain. The popularity of neural networks has surged recently because of their ability to solve incredibly complex problems.

To prevent overfitting, when training, checkpoints are implemented which return the model in its optimal state. Due to the complexity and power of this type of model, the nearby features described in the following section are not applied to the neural network. It is suspected that the model will achieve good performance, even without the implementation of the nearby features.

2.4: Discussion of Nearby Features:

A key innovation in our approach to modelling was incorporating distinct methods of "nearby features" for different models:

1. Distance and Bearing Features:

Used in the Linear Regression model. Calculated the distance and directional bearing from each observation point to a central reference point. This established a consistent spatial reference frame that helped capture how temperature varies with distance and direction from central UK.

2. Time Lagging and Spatial Averages

Used in the Random Forest model. Given that the weather (temperature) is continuous and can be influenced by the last state, we thus applied lagged features in the random forest model to capture this temporal dependency [2], allowing it to memorise the past for a while and improve its temporal learning. We first selected key variables for the weather forecast: `tp`, `sp`, `u10`, `v10`. Then we constructed lag features for every spatial spot after ensuring each spot has sorted all the observed values by time.

Besides, spatial averages are applied in the random forest to work as spatial nearby features. We constructed a 3*3 grid for each spot and calculated the averages of `tp`, `sp`, `tcc` of all nine spots, allowing model to consider the nearby influences. At last, we combined lagged features and spatial averages together, dropped n/a values created during construction, and then trained model on that dataset.

3. Radius-Based Aggregation and Pressure Gradient Calculation:

Used in the Gradient-Boosting model. For each weather observation point, we calculated statistical aggregations (mean, minimum, maximum) of key meteorological variables from surrounding locations within a 0.25-degree radius (approximately 25km). This approach allowed the model to capture local weather patterns beyond individual measurement points.

Computed pressure gradients along latitude and longitude directions to quantify how rapidly atmospheric pressure changes across space. These gradients provide crucial information about potential air movement that drives weather changes.

2.5: Model Development and Validation Strategy

Every model incorporated hyperparameter tuning. We implemented k-fold cross-validation to ensure models parameters were robust. The tuning process systematically explored different parameter combinations to minimise prediction error while avoiding overfitting.

Using the best hyperparameters found, models are trained on the full training dataset to maximise learning from the given data. This methodical approach allowed for fair comparison between models while maintaining scientific rigor in the evaluation process.

2.6: Evaluation Methodology

To comprehensively assess model performance, multiple evaluation metrics were calculated. Root Mean Squared Error (RMSE): which measures the average magnitude of prediction errors in Kelvin, with lower values indicating better performance. R-squared (R^2): Quantifies the proportion of temperature variance explained by the model, with values closer to 1.0 indicating stronger predictive power. Mean Absolute Error (MAE): Provides the average absolute difference between predictions and actual temperatures in Kelvin, offering an intuitive measure of prediction accuracy.

Models are evaluated using these metrics and visual assessment of scatter plots showing the predicted temperatures against the actual temperatures, analysis of the residuals (the difference between the predicted and actual values). Feature importances are considered to gain comprehensive insights into model performance and behaviour.

Section 3: Results

3.1: How to assess results

A machine learning model's performance is measured by its score on various statistical tests. The best performing model will be the one with the best scores. The model will have a score on the training set and the validation; in general, these scores will be different. A model that has a significantly worse score on the validation set than it does on the training set is likely overfitting to the training data, meaning that the model does not generalise well to unseen data, and therefore, not a good predictor. One of the goals of a successful machine learning project is to create a model with small scores on the training and validation datasets. The models' scores are calculated in two configurations: the first from the models run without nearby features, and the second the models run with nearby features. Including these versions makes the difference in model performance with/without nearby features obvious.

3.2: Results, and Nearby Feature Improvement

The validation scores from all models are included below. All the models tested which include nearby features, show an improvement in performance with the nearby features included. It is important to mention that the Gradient Boosting model uses basic standard hyperparameters different from the best ones found during hyperparameter search. The Linear Regression RMSE improved by 0.33%. The Gradient Boosting RMSE improved by 1.85%:

Linear Regression:

	RMSE	R^2	MAE
With nearby features:	2.3846,	0.7630,	1.7866
Without nearby features:	2.3924,	0.7614,	1.7950

Random Forest:

With nearby features:	0.8167	0.9722	0.5280
Without nearby features:	N/A	N/A	N/A

Gradient Boosting model:

With nearby features:	2.0295,	0.8281	N/A
Without nearby features:	2.0677,	0.8216	N/A

Neural Network:

With nearby features:	N/A	N/A	N/A
Without nearby features:	1.8413	0.8587	1.3874

These results consistently demonstrate that incorporating nearby spatial features yields modest but meaningful improvements in prediction accuracy across all models. The improvements were most pronounced in the ensemble models: Random Forest and Gradient Boosting, which have greater capacity to utilise complex feature interactions.

These metrics indicate that the Random Forest model provides the most accurate and consistent temperature predictions among all tested approached. For this reason, this is the final model that will be used for making predictions on the fully unseen dataset as it is predicted to generalise the best to new data.

The Gradient Boosting model achieved the second-best performance. This is a strong performance and indicates that out model should generalise well to new data.

The ANN followed. This was a somewhat unexpected result given the inherent complexity of ANNs – we would have expected it to outperform the other algorithms that were implemented. This result is likely due to the simplicity in structure of the ANN we chose to train. (which was selected based on the large amount of data for training and the computational power available).

The Linear Regression model showed the weakest performance. This result is expected given that the model employs the simplest regression technique from the ones selected – it isn't strong enough to learn best from the training data due to the complexity of the weather data. The underlying structure of the data is, therefore, not linear.

3.3: Model Fitting Analysis

To assess whether our models were overfitting or underfitting the training data, we compared training and validation metrics and examined prediction patterns (Figures 4).

For Random Forest, the training RMSE notably lower than the validation RMSE and the training R^2 is higher than the validation R^2. This gap indicates some degree of overfitting, which is common in tree-based ensemble methods due to their high capacity to capture training patterns. However, the strong validation metrics suggest that this overfitting is moderate, and the model still generalises well to unseen data.

Gradient Boosting shows signs of overfitting similar to Random Forest, but to a more managed degree due to regularisation parameters like learning rate and subsample ratio. From figures included in the associated Jupyter notebook the model demonstrates consistent performance across the temperature range with some deviation at extreme temperatures.

The ANN shows a balanced fit with similar training and validation metrics. The violin plot in Figure 5 indicates that error distributions are symmetric around zero, suggesting the model isn't systematically biased.

With nearly identical training and validation metrics, our Linear Regression model shows no signs of overfitting. However, its higher overall error rates suggest underfitting, as it can't capture the complex non-linear relationships in the data.

Random Forest utilising nearby features will be our model used for final predictions on unseen data. This decision was made given its strong performance metrics measured on the validation set – despite some mild overfitting, these scores indicate excellent generalisation to unseen data, indicating it should make consistently accurate predictions across the temperature range. Additionally, as shown in Figure 5, it demonstrated the most consistent error distribution with minimal bias and outliers.

Section 4: Discussion

4.1: Feature Importance Analysis

The feature importance analysis reveals valuable insights about which variables most significantly influenced temperature predictions. Feature importance analysis was not conducted on the Linear Regression model due to the fact that this model is unlikely to give accurate weather predictions, so deeper analysis was not sought after.

The Gradient Boosting model identified temporal features - specific days of the month and hours of the day - as the most important predictors. Notably, several of the "nearby" features that were engineered appeared among the top 20 features. This confirms that the spatial context approach added meaningful predictive information to the model.

The ANN feature importance reveals a different pattern. As feature indices are hard to interpret directly, the high weights of multiple features suggest that the ANN was using a broad range of input variables in its predictions.

The Random Forest model feature analysis faced a technical issue, due to the size of the model (16 GB) and unfortunately the important features analysis was not conducted.

4.2: Exploration of Alternative Approaches

We explored several different nearby features that were not implemented in the final models:

1. K-means Clustering of Geographical Areas: Clustering locations based on temperature and geographical coordinates to identify natural climate zones. This approach risked data leakage by incorporating the target variable in the clustering process.
2. Temporal Sequence Embedding: Creating embeddings from sequences of past temperatures at each location to capture recurring temporal patterns. This required significant computational resources and risked overfitting to historical patterns.

4.3 Model Training Challenges

Each model presented unique challenges during development: the Random Forest model, while delivering the best performance, was the most resource-consuming to train. The model frequently stalled during fine-tuning, even after reducing the parameter search space and number of iterations. This is a known limitation of tree-based ensemble methods with large datasets and many features. Additionally, at the last stages of this project we discovered that a seed for reproducibility of results was not used correctly within the Random Forest Model and the actual notebook has a slightly different scores output.

The Gradient Boosting model required 112,846.80 seconds (approximately 31 hours) to train - an enormous time investment. While it delivered decent results (second-best performance), the return on this investment was questionable compared to the more efficient Random Forest.

The Neural Network demonstrated an unexpected performance gap. Despite the theoretical advantages of neural networks for capturing complex patterns, it ranked third in our comparison. We had expected it to outperform the tree-based methods, but its requirement for extensive hyperparameter tuning (while our project only used a dataset sample with limited gird search) and potential sensitivity to feature scaling may have limited its effectiveness.

The Linear Regression model was straightforward to implement but performed poorly relative to the other models. Its inability to capture non-linear relationships between variables and temperature resulted in systematic prediction errors, particularly at extreme temperatures.

4.4: Future Improvements

Several prospects that could be explored to enhance the prediction performance:

1. Enhanced computing resources: A more powerful computational infrastructure would enable more comprehensive hyperparameter tuning, particularly for the resource-intensive Random Forest and Gradient Boosting models. Our fine-tuning was conducted on sample sets (10% of data); using the full dataset could yield better parameter optimisation.
2. Advanced neural network architectures: Implementing a more sophisticated neural network architectures with proper regularisation (dropout, batch normalisation) and optimisation strategies (learning rate scheduling, early stopping) could potentially improve performance. Recurrent neural networks (RNNs) or convolutional neural networks (CNNs) might better capture the spatial and temporal patterns in the data.
3. Ensemble approaches: Combining predictions from multiple models could potentially improve accuracy further. A weighted ensemble of the Random Forest, Gradient Boosting and Neural Network models might

achieve higher prediction accuracy by leveraging the strengths of each approach.

4. Feature engineering refinement: Further exploration of interaction terms and derived features, particularly those capturing seasonal temperature patterns, could enhance model performance. The strong influence of temporal features suggests that more sophisticated time-based features could be beneficial.
5. Geospatial modelling: Incorporating more sophisticated geospatial techniques, such as geographically weighted regression, could better capture the spatial correlation in temperature patterns.

Given the strong performance of the Random Forest model despite limited hyperparameter tuning, this approach appears to have the most potential for further optimisation, particularly if computational constraints can be addressed.

4.5: Conclusions

The goal of this group project was to create a machine learning model to predict the temperature. Of the four machine learning models presented, the Random Forest proved to be the most effective. Based on the results obtained we can determine that the Random Forest model is most suitable for real-world application in weather prediction.

Additionally, the methods implemented to engineer nearby features have all improved the performance of each of the machine learning models that they were applied.

4.6 References

[1] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

[2] Scikit-learn developers. (2022). Time series prediction using lagged features. Scikit-learn. https://scikit-learn.org/stable/auto_examples/applications/plot_time_series_lagged_features.html

Appendix

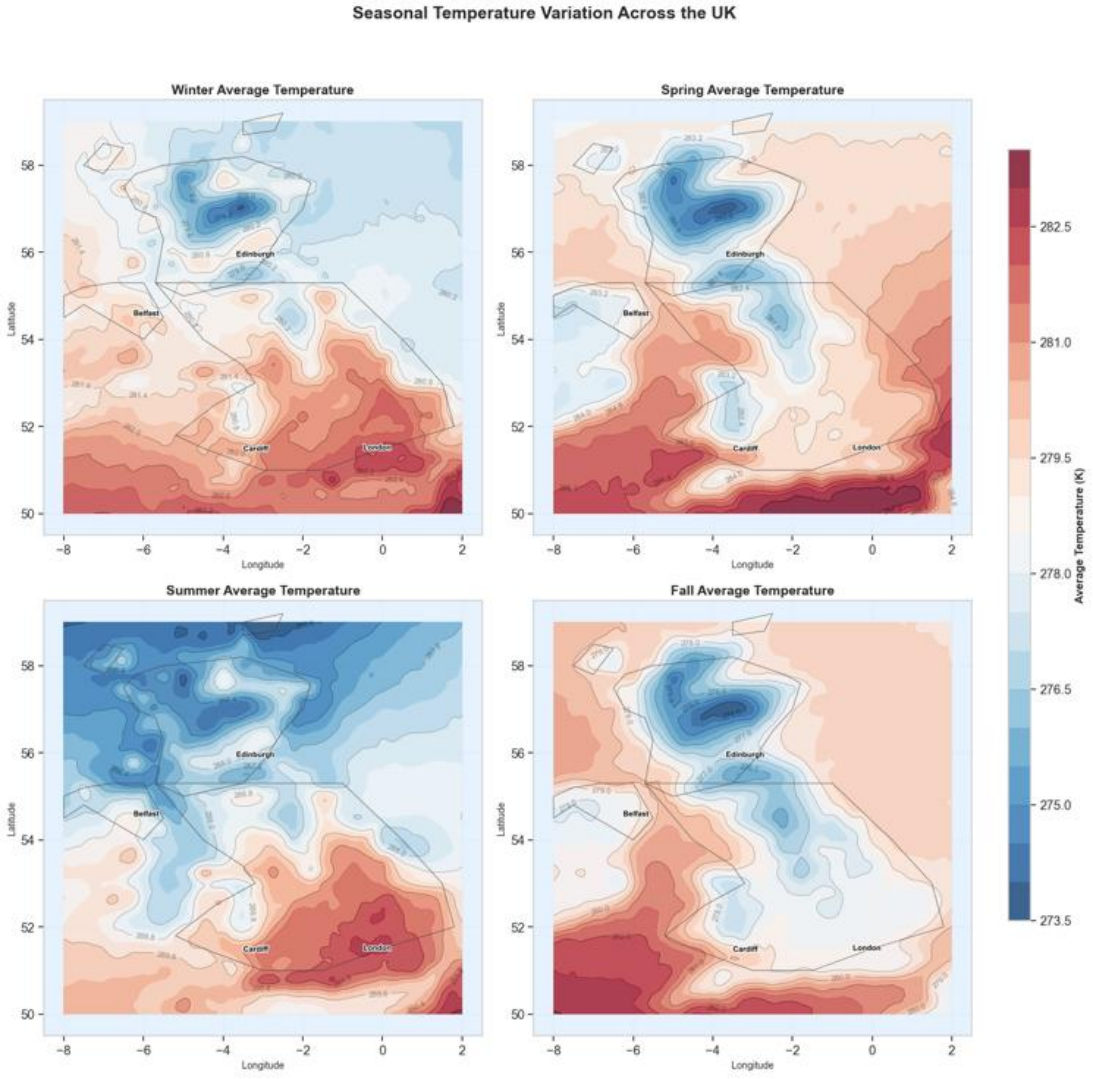

Figure 1: Plot showing the average seasonal temperature distribution throughout the UK. Created during data exploration.

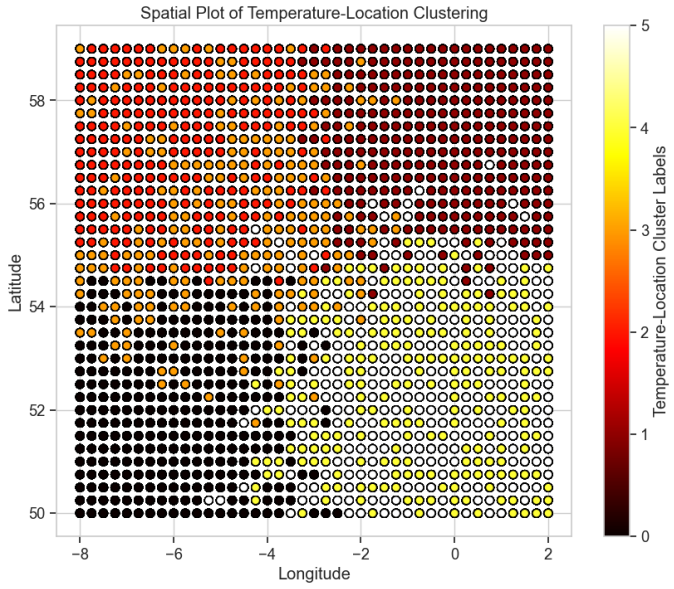

Figure 2: Plot showing the temperature clustered with latitude and longitude. Comes from running K-Means algorithm during data exploration. Number of clusters determined by silhouette score.

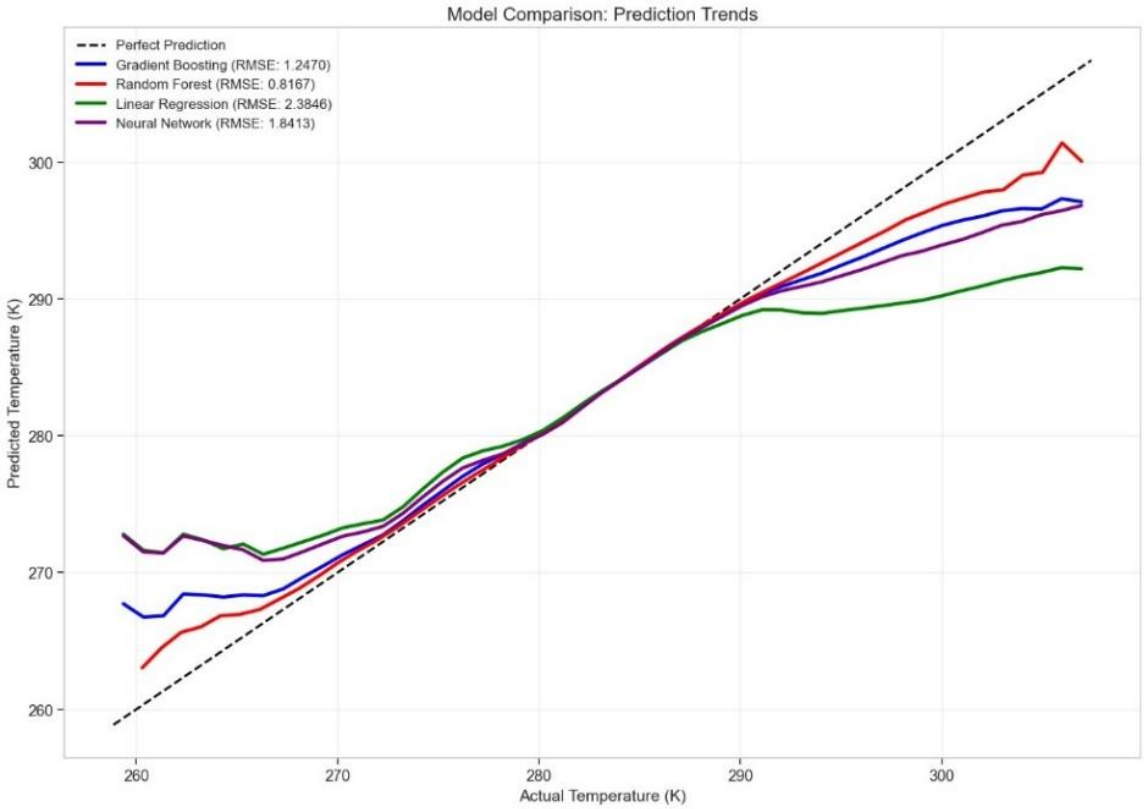

Figure 3: Showing the trendlines associated with the models from each chosen architecture.

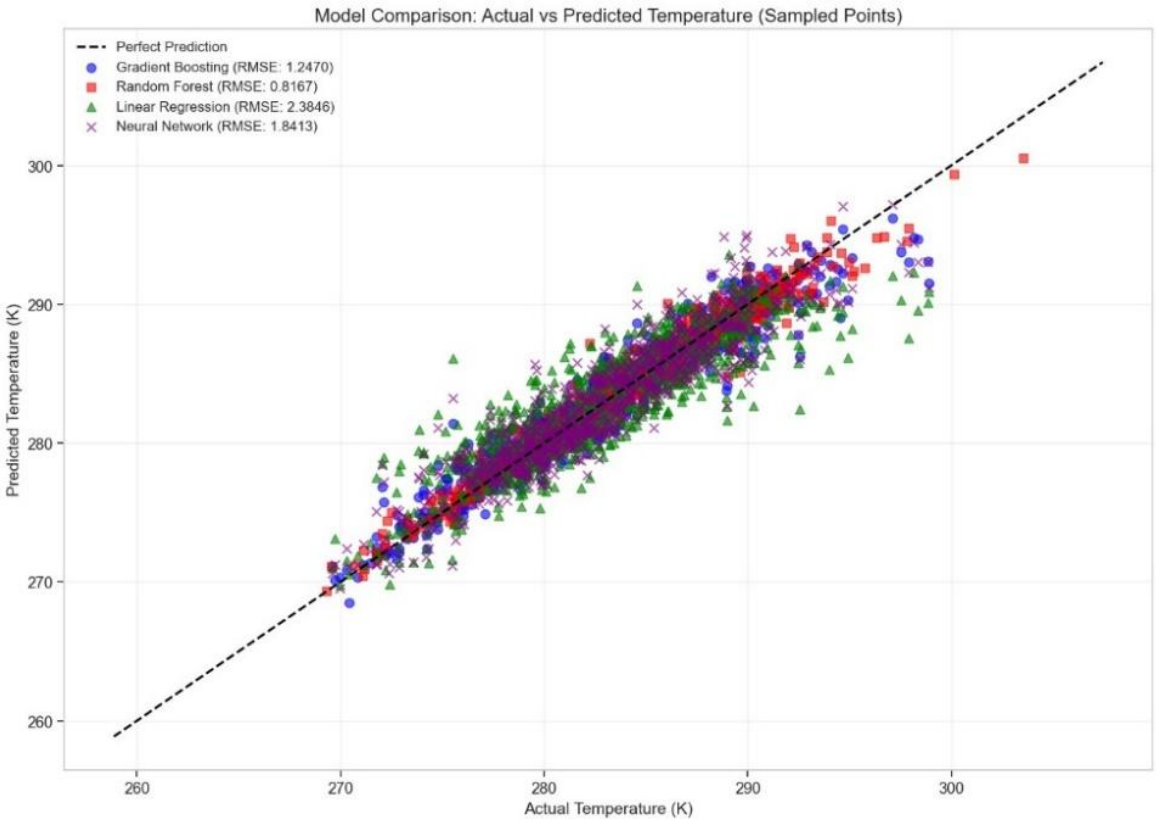

Figure 4: Plot showing a sample of the distribution of predictions drawn from each model.

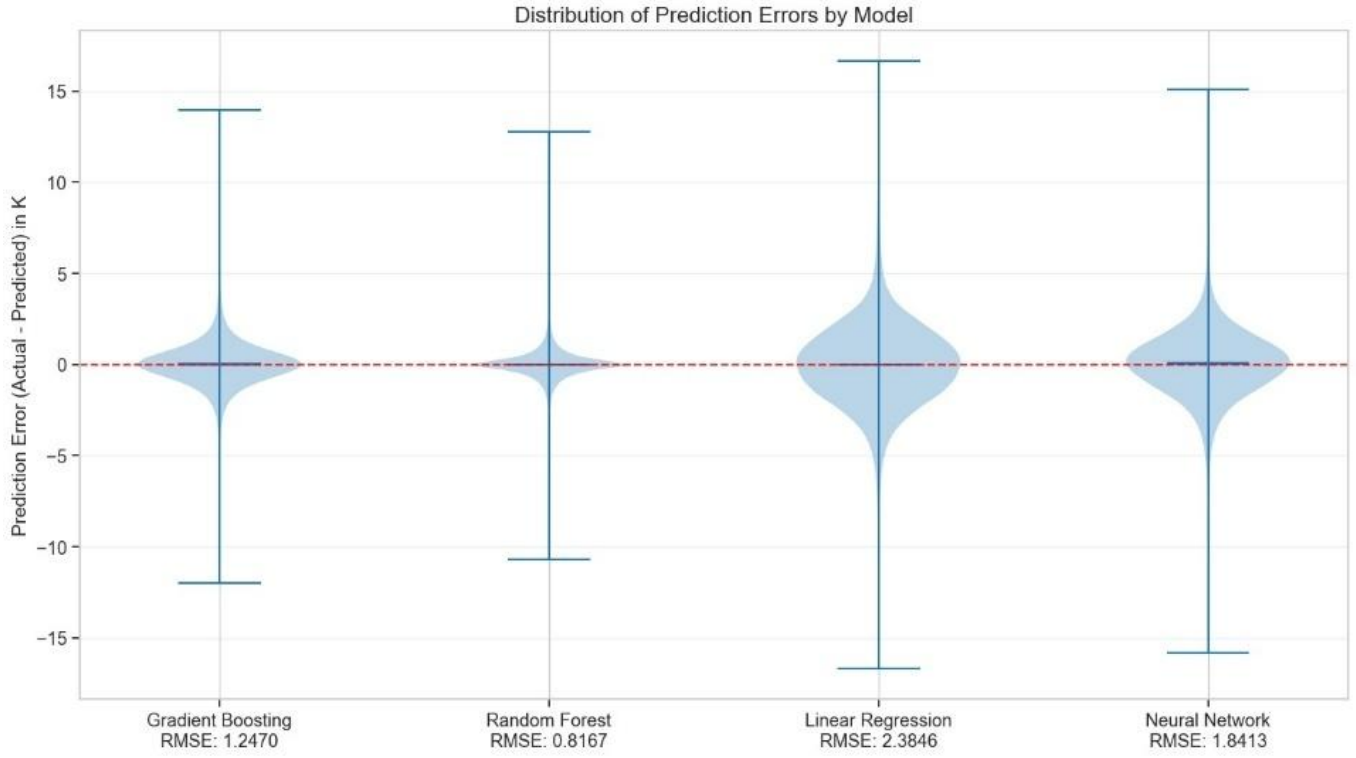

Figure 5: Plot showing the distribution of RMSE errors for each model architecture. Used to draw conclusions about model architecture performance.

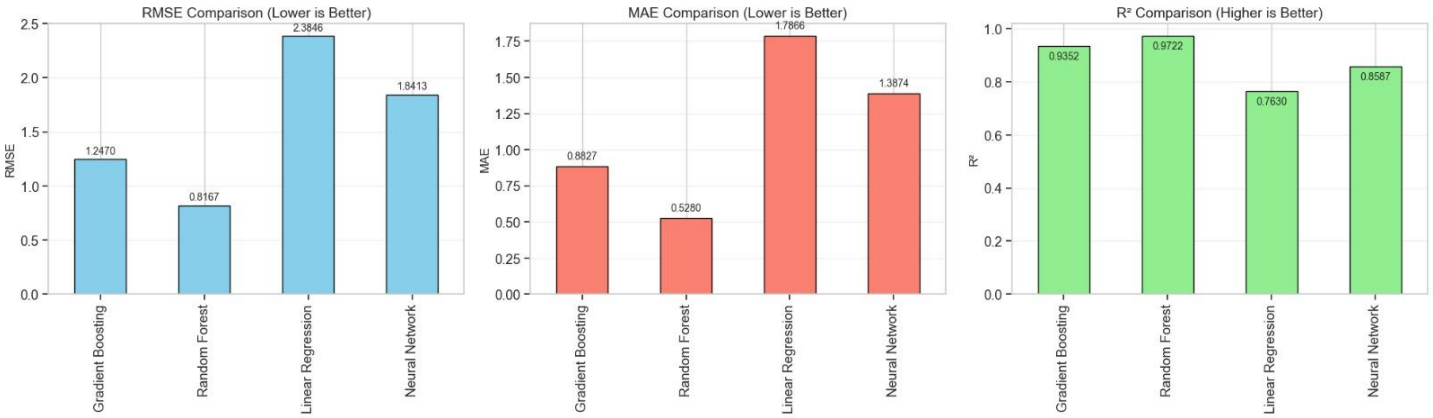

Figure 6: Plot showing the validation score for each model.