ID5059 Coursework Assignment 2
Group 2 Summary Report

ID 230031900, 200001408, 210003624, 240032316, 240031581

Overview

The goal of this machine learning project is to develop a predictive model for temperature in the UK based on a variety of meteorological features. The model should improve the weather forecasting of the UK-based meteorological agency. The data was sourced from the European Centre for Medium-Range Weather Forecasts, based on the dataset called ERA5. The dataset contains **13288920** observations and **13** features related to various meteorological factors that impact the temperature (t2m in Kelvins) and is structured by spatial and temporal patterns. We also were briefed that using nearby features can be useful for our task. We first investigated the dataset for potential correlations between the features and the temperature and then engineered it to be more easily interpretable by the models. We chose 5 models that we thought were likely to be well-suited to this data as well as attempting to ensure that the models covered a range of different approaches so that each model was interesting to investigate and did not overlap too much with the others.

Data exploration

It seemed reasonable to assume that the time of day would have an impact on the temperature reading as time of day is a good predictor of the Sun's location in the sky which has an obvious effect on the temperature. It is, of course, not a perfect predictor, as the entirety of the area covered by the data is part of the same time zone, which means that while all areas are displaying the same time the sun exposure of the area is not necessarily the same (the sun rises later in areas farther to the West). Figure The angle of the Sun, and resulting temperature increase, is therefore also determined by location at which the data is being collected, both in terms of longitude as discussed already, but also in terms of latitude which determines how much Sun the area is exposed to in conjunction with the broader time scale of when in the year the day in question falls. In order to gain a better understanding of how these changes were actually affecting the data, we created graphical representations of temperature fluctuation by time of day; temperature change over the year [Figure 1]; and temperature in each location as seen in [Figure 2].

Another influence on the temperature in a location at a given time is the cloud cover in the area. We can see from visualisations made of temperature variation with cloud cover [Figure 3] that, as expected, during the day the cloud cover affects the Sun exposure in an area with high cloud cover reducing temperatures, but at night the cloud cover is actually positively correlated with temperature which is likely due to the insulating effect that the clouds provide by absorbing and re-radiating IR produced by the ground heated during the day. However, the observed effect is quite slight compared to the previously discussed factors, so it is likely less significant in determining temperature. These features all share a common aspect: they are all at least partially abstractions of the Sun exposure that an area receives at a given time (and in the time recently preceding). Although this data was not available for us to analyse it seems clear that it would be a very good predictor of temperature if it could be included.

Precipitation is also a potential predictor of temperature. High quantities of precipitation are obviously associated with high levels of cloud cover which could suggest lower temperatures in the day as mentioned above, but the type of precipitation could also provide some clues as to the temperature as certain types of precipitation (e.g. snow) are only possible with lower temperatures. Plotting the spread of temperatures in which these precipitation types appear [Figure 4] allowed us to re-evaluate them in terms of this metric.

Our initial thought was that wind had the potential to be very significant as a predictor of temperature for a few reasons. Firstly, wind transports air from one location to another, and, if that air is of a different temperature to that of the destination, then the average temperature of the destination will necessarily change. This could theoretically occur not just between different locations by latitude and longitude but also between different altitudes by vertical wind shear. However, as the temperatures of the nearby locations, at any given time, are not reported in the test dataset it is not possible to use this information directly to influence prediction for the near future temperature of our destination. The direction the wind is coming from could provide a way to approximate the likely temperature change brought by the wind [Figure 5]; in the UK winds from the North and East tend to be cooler (as they are carrying air from the Arctic and the Atlantic), while winds from the South and West tend to be warmer (as they are carrying air from the Equator and Europe).

Surface pressure is also likely to be very closely associated with the temperature at the location as pressure and temperature are directly proportional [Figure 6].

Data cleaning and preprocessing

There was no missing data in the dataset, so it was not necessary to impute data. We then dropped the id column along with the u10 and v10 features as they had high overlap with u100 and v100 and the latter pair were better correlated with temperature, albeit only slightly. Ptype was replaced with a list of the precipitation types ordered by the temperature required for them to form. This allows the relationship between the type of precipitation and the temperature to be more easily interpreted by the model. The majority of the remaining features we were able to include as they were, only having to scale them to ensure they were standardised.

Feature engineering

The valid time feature was of limited use in the format it was in initially, so we decided to trim the year information as there is only one year present in the training data, and convert the day and month data to 'month of the year' and 'day of the year' features that can be included separately. The time of day can then be included as another feature as it contains separate information from the seasonal data. These are also converted into sinusoidal forms, so the model has a better understanding of the proximity of the start and end of the year (and day) in terms of their effect on temperature.

Initially, we engineered nearby spatial and temporal features based on temperature (t2m): for spatial we split the UK into 5 regions and calculated an average temperature for each as a benchmark for spa, for temporal we calculated temperature 1 hour ago for each observation. But after we discovered that test dataset does not contain t2m, we realized that we cannot apply this approach.

This issue also affects the temperature transfer caused by the wind but as mentioned earlier we attempted to get around this by creating a wind direction feature from the u and v components of the wind that would allow us to judge where the wind is coming from. We then tested to see how correlated this new feature was with temperature. Our findings were that it was actually less correlated with temperature than the v100 component of the wind but there is still some dependency on direction, so we decided to include both the components and the newly generated feature.

Elevation

Each longitude and latitude in this dataset have different elevations [Figure 7]. Higher elevations tend to have lower surface pressures [Figure 8] and at higher elevations the impact of low surface pressure is more pronounced on temperature [Figure 9]. Elevation impacts other determinants, e.g. higher elevations have lower wind speeds. Elevations can be easily imported from the Shuttle Radar Topography Mission (SRTM) dataset in R, but accessing it via API (e.g. from the OpenTopography API) in Python proved too difficult to implement in our pipeline. Because our models are capable of capturing interactions implicitly inferring elevation.

Models fitting and tuning

For temperature prediction we fitted 5 regression models. The performance measure for all the models we fitted is RMSE, which stands for Root Mean Square Error, a measure of how far our temperature predictions are from the actual temperatures, on average, with lower numbers indicating more accurate predictions. We trained models on training dataset and validated the results on unseen data on validation dataset.

Model 1: KNN Regressor

K-nearest neighbourhood (KNN) regressor model suits well the data's spatial and temporal structure for KNN's ability to predict temperature by identifying the most relevant historical observations from similar meteorological conditions across nearby locations and times.

Due to dataset's large size (10.6 million observations) we applied Principal Component Analysis (PCA) to reduce the dimensionality of data while preserving 95% of variance, so data is still representative for training the model. PCA revealed that historic local wind speed and precipitation are significant contributors to temperature prediction, confirming the value of engineered spatial and temporal nearby features. The initial model with default parameters performed RMSE = 0.0000 on train and 0.5139 on validation, suggesting that model is overfitting, meaning it memorizes the data perfectly rather than learning generalizable patterns.

After extensive hyperparameter tuning using GridSearch on a 5% of sample training data, we found that our initial configuration was nearly optimal, resulting in same performance as it was initially. Most probably, KNN was already robust enough for this prediction task so that it was less sensitive for parameters tuning, or our intuition for selecting parameters was confirmed. Also, we fed KNN with nearby features, which the model would do by default, with given nearby features we basically set the priorities for the model. KNN's trained on best hyperparameters achieved RMSE 0.5139 on validation and 0.5140 on test dataset, indicating strong predictive performance despite some overfitting. This means that this model predicts temperatures with average error of just 0.5139 of Kelvin unit, which is pretty accurate. Evaluation details in [Figure 10].

Model 2: Random Forest Regressor

For the second model, we chose a Random Forest Regressor, an ensemble method that averages the outputs of multiple decision trees. This approach is particularly well-suited to our system, as it effectively handles complex, non-linear relationships between environmental features while minimizing the impact of less informative variables. Although individual trees are prone to overfitting, combining their outputs significantly reduces this risk.

Initially, we attempted a Grid Search to tune the model's hyperparameters. However, due to the relatively long training time of Random Forests and the large size of our dataset, we opted instead for a Randomized Search on a representative subset. Despite exploring a wide range of parameter combinations, our original hyperparameters yielded the best results. With the Random Forest we are also helpfully able to assess the importance of each of our variables in the final value from the forest. This let us see that the microscopic annual features (sin/cos day) were by far the largest factors in prediction followed by other time features and location. It also let use see that our engineered historic values were of little significance in predicting weather.

Overall, the Random Forest achieved a competitive RMSE of 0.5228 on the validation set, performing well compared to our other models. While the model likely exhibits some overfitting—given the dense temporal and spatial nature of the data and the necessity of including the full annual cycle of data in training—this effect is likely mitigated by the ensemble nature of the model. As such, the Random Forest is a strong candidate for temperature prediction in this context. Model evaluation details in [Figure 11].

Model 3: XGBoost

We chose XGBoost as a candidate model for predicting temperature on this high-resolution weather dataset due to its strong performance with structured data and ability to capture complex nonlinear relationships. XGBoost supports regularization to reduce overfitting, and scales well with large datasets — both of which make it well-suited for the granular, feature-rich nature of weather data. Its flexibility and speed also make it ideal for efficient hyperparameter tuning and iterative experimentation.

We used a randomised search (as opposed to a grid search, prioritising computational efficiency) to find the optimal combination of hyperparameter values. The results of the randomised search were as follows: learning rate = 0.17, max depth = 3, number of estimators = 261, L1 regularization (α) = 0.38, L2 regularization (λ) = 0.49, sub-sample = 0.78. The latter 3 hyperparameters (α, λ and sub-sample) are to prevent overfitting. See section below for details on regularization.

The results of this model - where the training cross-validation RMSE (3.4) is significantly higher than the validation RMSE (1.6) - are likely due to a data leakage issue introduced by the way the validation set was constructed. The dataset is time-ordered, but we chose to use every 5th data point to construct the validation set, causing the training and validation sets to be interleaved. This inadvertently allows the model to train on data that is immediately adjacent in time to the validation points (this data is also very high-resolution data), providing it with information that would not be available in a real-world forecasting scenario. As a result, the model performs unusually well on the validation set, leading to an artificially low RMSE. In contrast, the cross-validation within the training data preserves temporal separation between folds, making it a more honest reflection of the model's true generalization performance, hence the higher RMSE. This discrepancy highlights the importance of preserving temporal integrity in train/validation splits for time-dependent data. It could have been better to hold out a contiguous block from the end of the dataset as validation, letting us predict only on future data, and using only past data to train. However, we only have a year of data, and leaving out a chunk at the end, like December, for example, would mean not exposing our model to a full cycle of weather patterns. We were also constrained in this specific project by not being able to shuffle, as we were working across software programmes and would not have been able to transfer a random seed. So, all this considered, we have taken the decision not to use XGBoost for our predictions and instead go forward with a model that is more robust to this validation set issue. Evaluation details in [Figure 12].

Model 4: Elastic Net

For the fourth model we selected Elastic Net Regression which is a combination of two other regression models: Ridge and Lasso; both models are regularisation techniques to combat the potential for overfitting in ordinary linear regression. The two models differ, however, in the way in which they go about the regularisation process. Ridge regression is more useful in cases where the features are well correlated which can lead to poor feature selection in normal linear regression. It achieves this by applying a penalty to large regression coefficients thereby reducing their relative weighting. Lasso regression is used to remove irrelevant features from consideration to simplify the model and prevent overfitting. These methods, of course, do have the potential to bias the model if the features that are penalised are important to the overall understanding of the relationships present in the data. To best reduce the potential for bias to be introduced by the application of either one of these models, elastic net regression is employed. This works by combining the two regularisations, with a weighting that determines which one of them affects the penalties applied to a greater degree. For datasets that have many important features, the weighting tends to lean towards ridge regression, and, by contrast, when the data has only a few important features with several less relevant features, lasso regression is favoured.

After tuning the model to determine the best hyperparameters, the results were an L1_Ratio of 0.1 and an Alpha of 0.01. The low L1_Ratio value suggests that the model is better fit by ridge regression than lasso regression meaning that there is likely a large proportion of the features that are significant. The moderate Alpha value suggests that it is necessary to apply some penalty to the data to prevent overfitting but that it is not necessary to drastically simplify the model. We then attempted to use the model to predict temperature values for the validation set which produces a metric by which we can gain information as to the model's effectiveness. This model also produced a lower RMSE for the validation set than it did for the training set, similar to the XGBoost model discussed earlier although not as pronounced. The model did not perform as well as the other models that we attempted, however, and so the decision was made to move away from it when attempting to predict the test data. This was probably an expected result due to the relative simplicity of the Elastic Net model compared with the other models and the complexity of the data that we were provided. Evaluation details in [Figure 13].

Model 5: Neural Net

Model Description

A neural network is a computational model that is inspired by the way biological neural networks in the human brain work. It consists of "layers" of interconnected "neurons" that learns patterns. Each neuron has a "weight"

that determines its importance in the network, and a "bias" that adjusts the output of the neuron. These weights and biases are set randomly but are adjusted during the training process to minimise a loss function.
A "layer" consists of a set of neurons that are connected to a previous layer. The first layer is the "input layer" which takes in the data, and the last layer is the "output layer" which produces the final predictions. There can be one or more "hidden layers" between these two that learn complex patterns in the data, with hidden layers closer to the output layer learning more complex patterns than those closer to the input layer.

Application to Temperature Prediction
Our data contains many complicated and non-linear relationships, is "high-dimensional" (lots of dependent variables), and lots of rows.
We apply a non-linear transformation to data as it passes between layers to learn non-linear relationships, whilst the structural nature of the network allows the network to learn simple patterns in early layers and build on their complexity in later layers to learn our complex relationships. We chose four different network architectures with increasing numbers of hidden layers to capture increasingly complex patterns in the data. The network copes with our high-dimensional data because early layers learn which features are important.
Exposing a neural network to more data increases the complexity of the patterns it can learn but increases the risk of learning noise in the data rather than the underlying pattern. We use "early stopping" to combat this, where we stop training if gains in accuracy from further training are too small.

Failure of Neural Net
The accuracy of our neural nets increased with complexity, except for the five-layer network which failed to learn any patterns. Our chosen net with three hidden layers failed to outperform KNN regression.
Our "early stopping" was likely too aggressive, so our networks was not trained for long enough to learn the patterns in the data. I set such a high threshold because I was worried about overfitting given the complexity of the data. Dropout involves randomly "dropping" a fraction of neurons (and their incoming and outgoing connections) on each training update to force the network to learn more generalisable feature and prevents overfitting. Weight decay adds a penalty term to discourage the network from relying on any single connection too heavily. Both these techniques reduce overfitting and could enable me to reduce the early stopping threshold to improve learning. Unfortunately, I discovered this problem too late; I had already trained all models, and I did not have computational resources to retrain all models using these techniques in time.
Evaluation detailes in [Figure 14].

Conclusions and Discussion

Our evaluation of multiple models (KNN, Random Forest, XGBoost, Elastic Net, and Neural Network) showed that K-Nearest Neighbors (KNN) performed best with the lowest RMSE, closely followed by Random Forest with almost identical results.

While KNN worked well with our engineered nearby features, we think combining both approaches might give us the best results. We suggest using KNN as a first step to process the spatial and temporal relationships in the weather data, helping to identify patterns across nearby locations and recent time periods. KNN is naturally good at finding similarities between locations with similar weather conditions and can effectively leverage the local patterns we engineered. Then, we could feed these KNN-processed features into a Random Forest model for making the final temperature predictions.

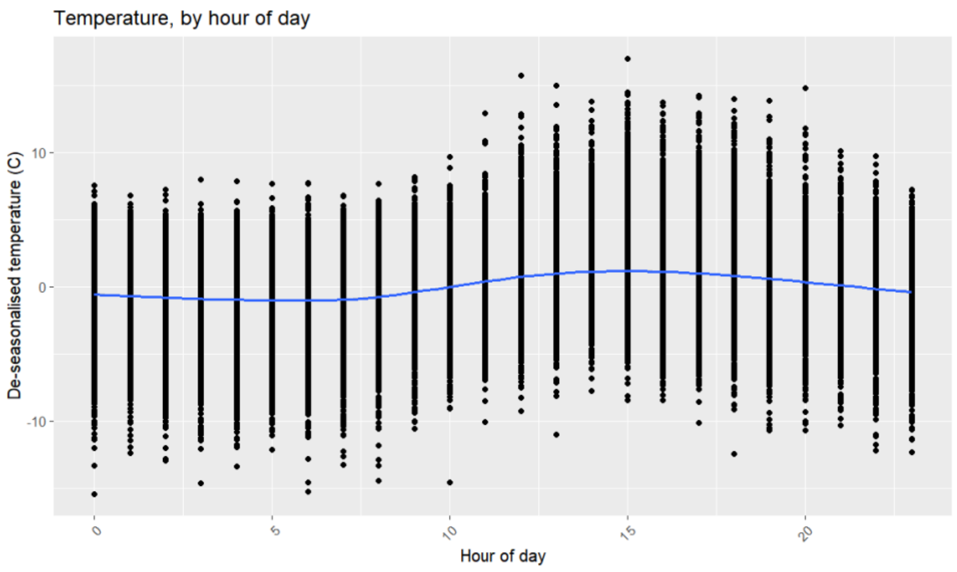
This two-step approach would take advantage of what each model does best - KNN is great at handling geographic and temporal data and finding similar weather patterns across locations and time, while Random Forest is good at handling complex relationships between different variables. Together, they would likely give us more reliable temperature forecasts.

Interestingly, while PCA identified historic local wind speed and other nearby features as among the most significant contributors for temperature prediction, the other models all identified these features as being relatively unimportant. To test the effects of the PCA results on the eventual outcome of the KNN model it would be necessary to train it without using the results of the PCA, which is impossible in the timeframe with the computational power we have available.

A potential improvement that could have been made during the feature engineering section might be to use a DBSCAN to cluster areas of similar temperature and then include the mean temperatures of these regions in 2018 as a feature in the dataset. This would provide the model some information as to the usual temperatures in various locations when predicting the temperature on the 2019 data (assuming we included this information as a feature in the 2019 test set). However, we ended up not including this information in our model despite having identified it as a potential improvement in the data exploration due to a miscommunication as to the viability of including temperature data in our models. If we had more time, it would be interesting to see how our model performance would be affected by the inclusion of this data.

Appendix
Data Exploration

Figure 1Valid Time:



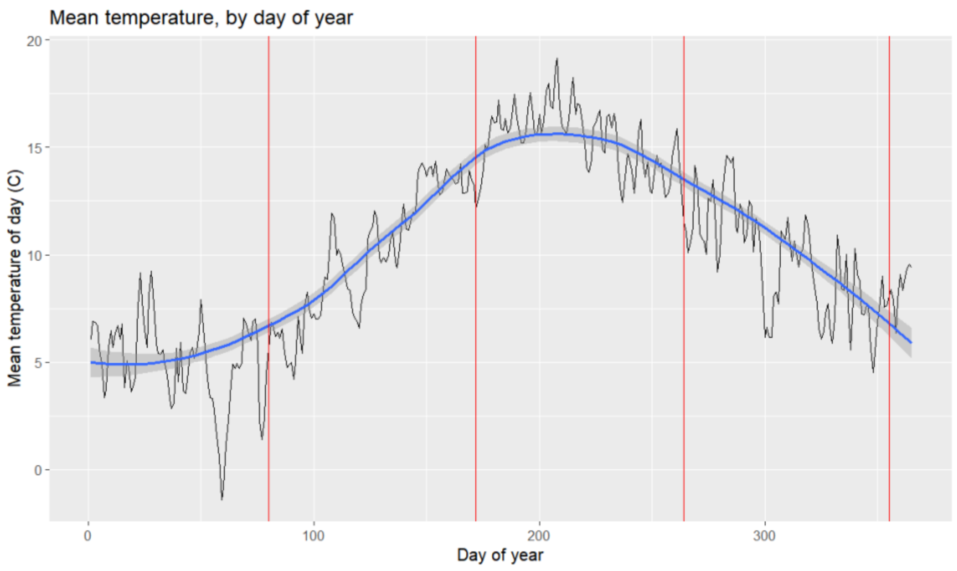
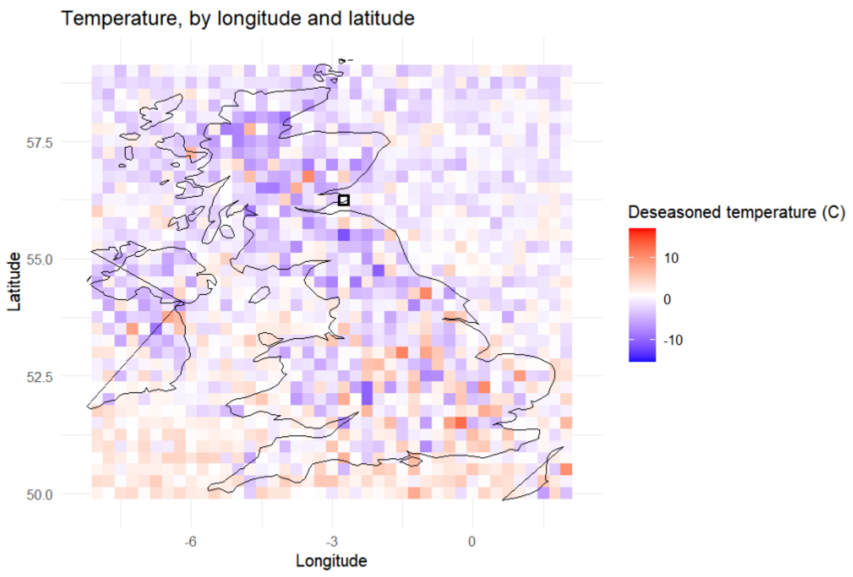


Figure 2 Location:



Temperature, by longitude and latitude
Latitude
57.5
55.0
52.5
50.0
-6
-3
0
Longitude
Deseasoned temperature (C)
10
0
-10

Figure 3 Total Cloud Cover:

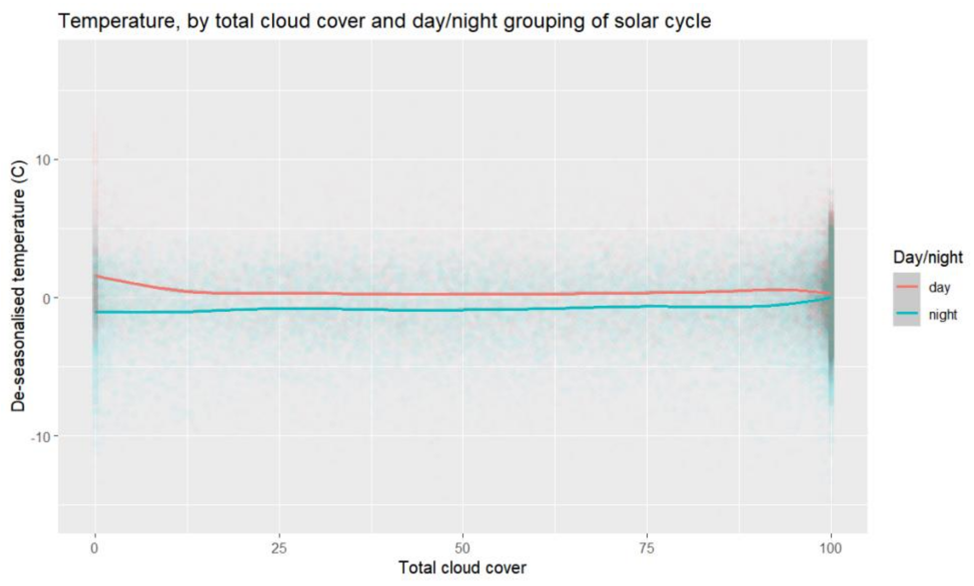
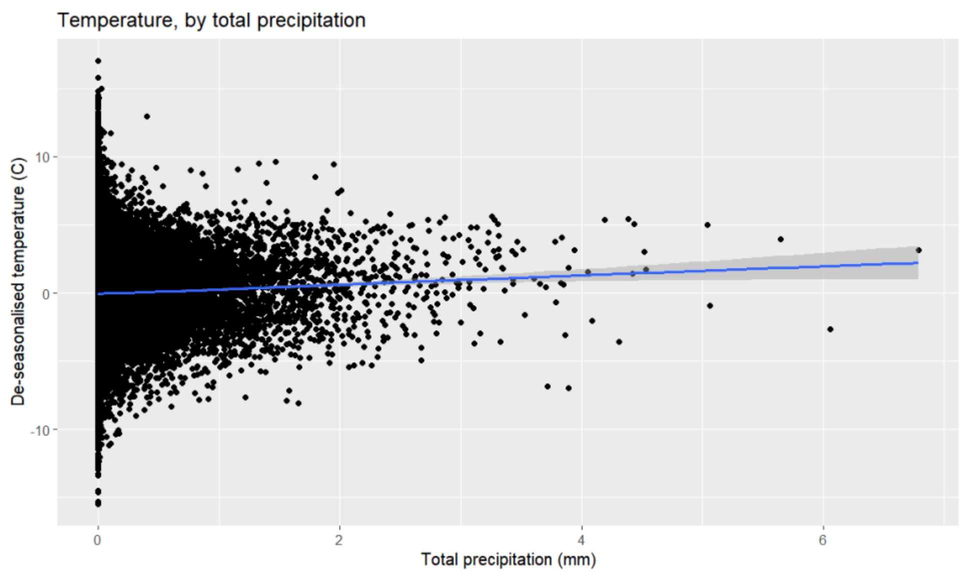


Figure 4 Precipitation:



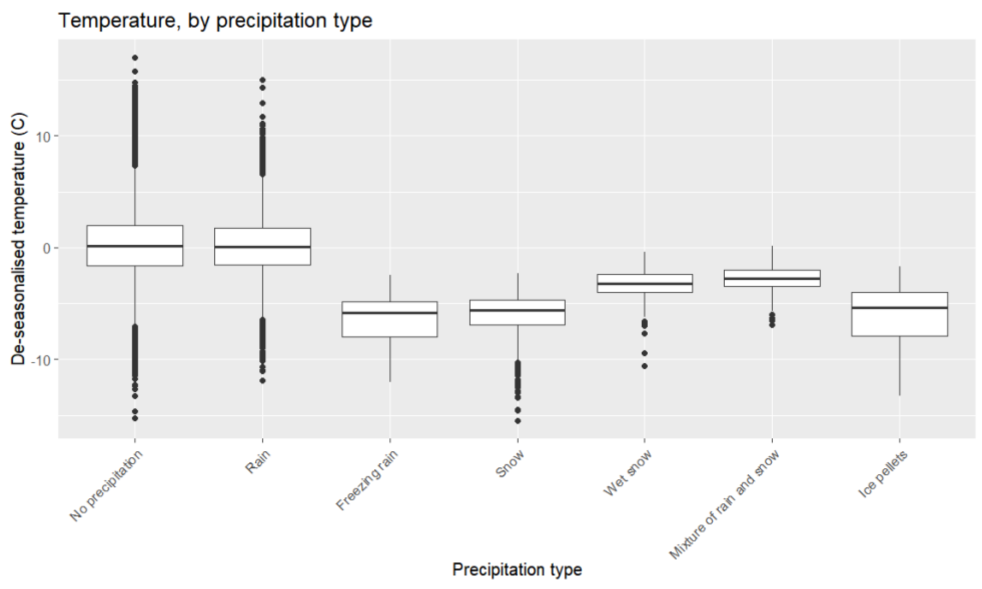
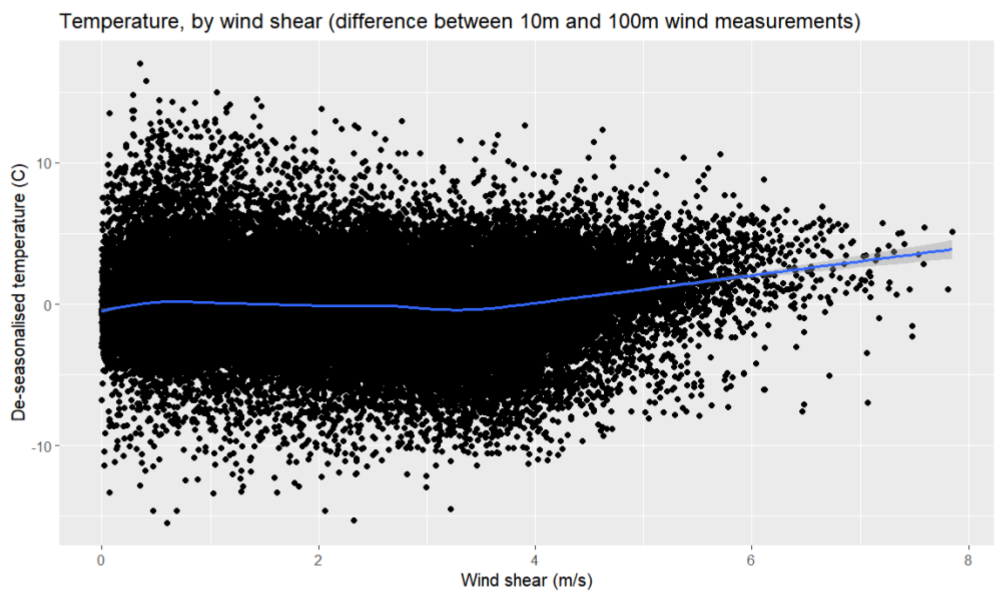


Figure 5 Wind:



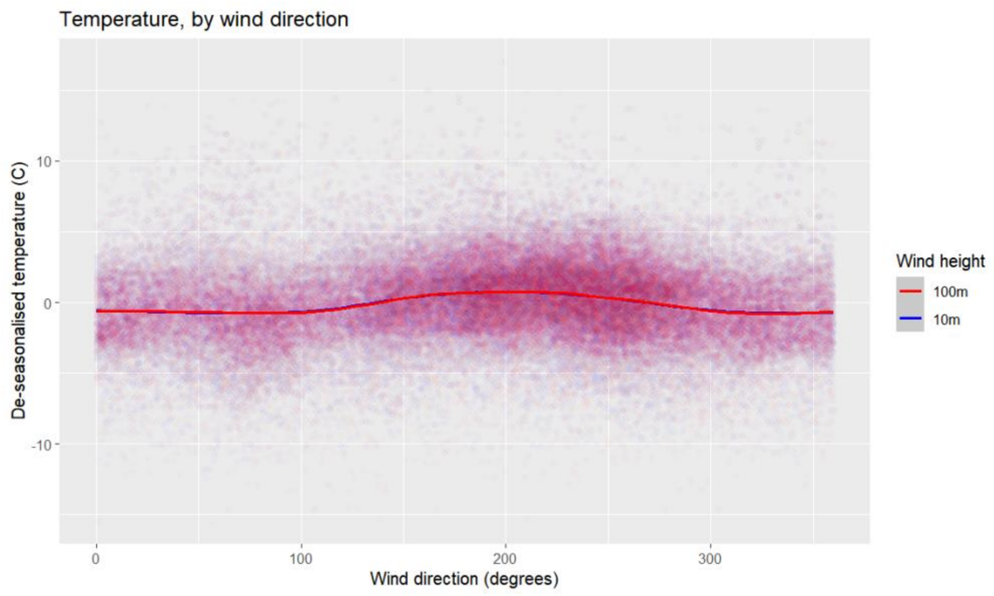


Figure 6 Surface Pressure:

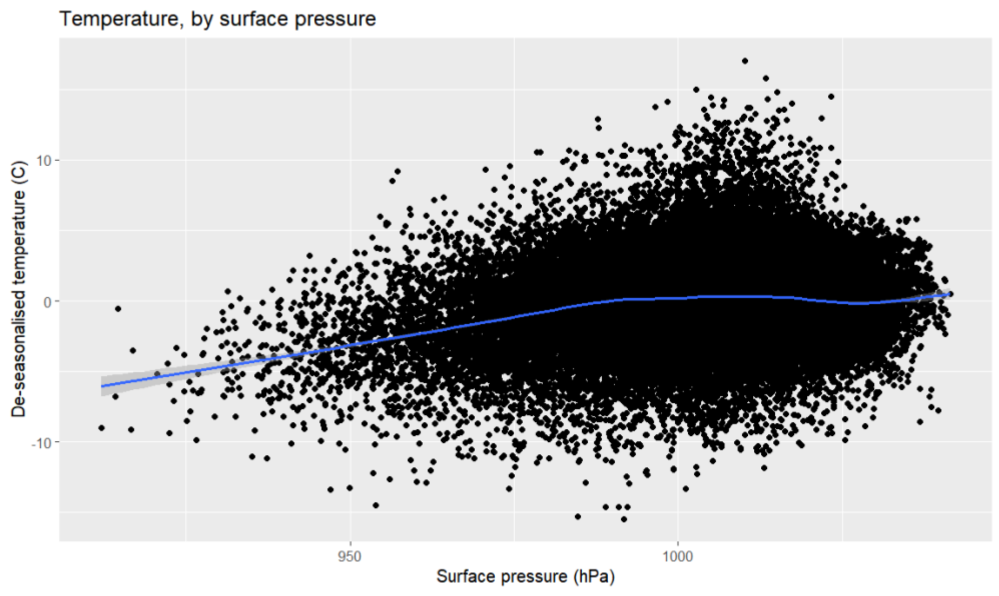


Figure 7 Elevation map of UK

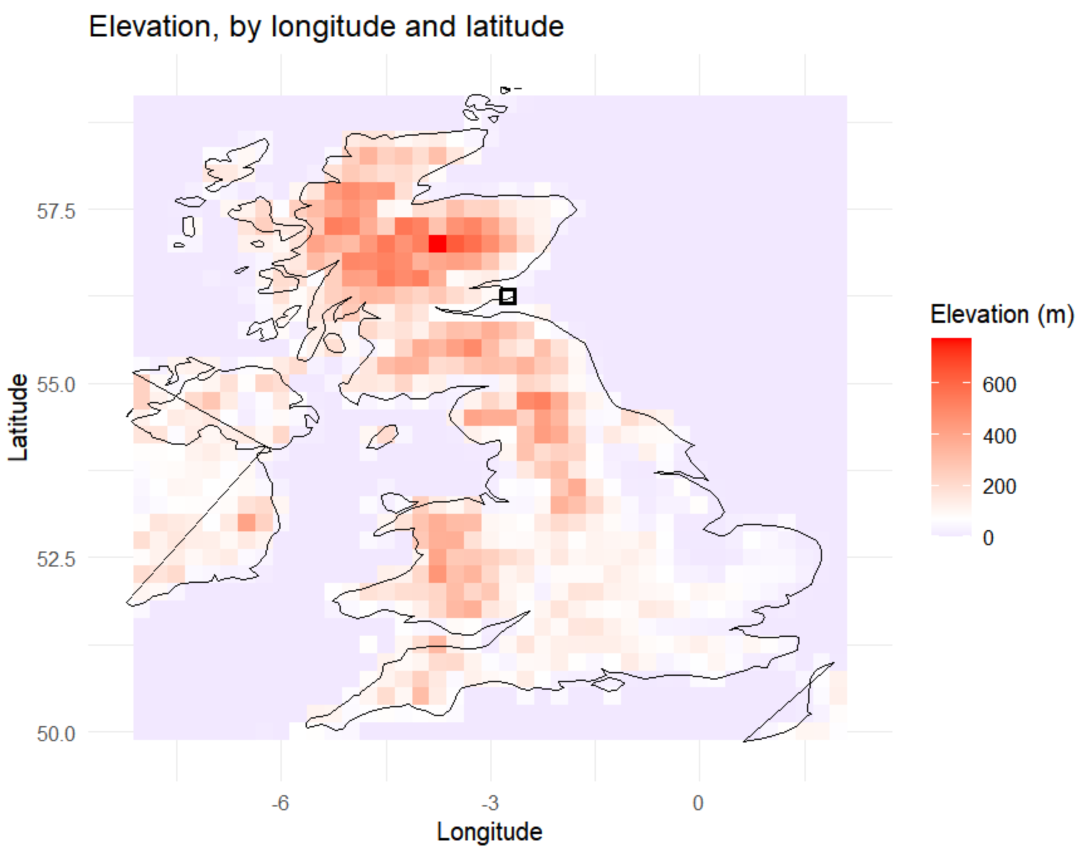


Figure 8 Elevation and surface pressure

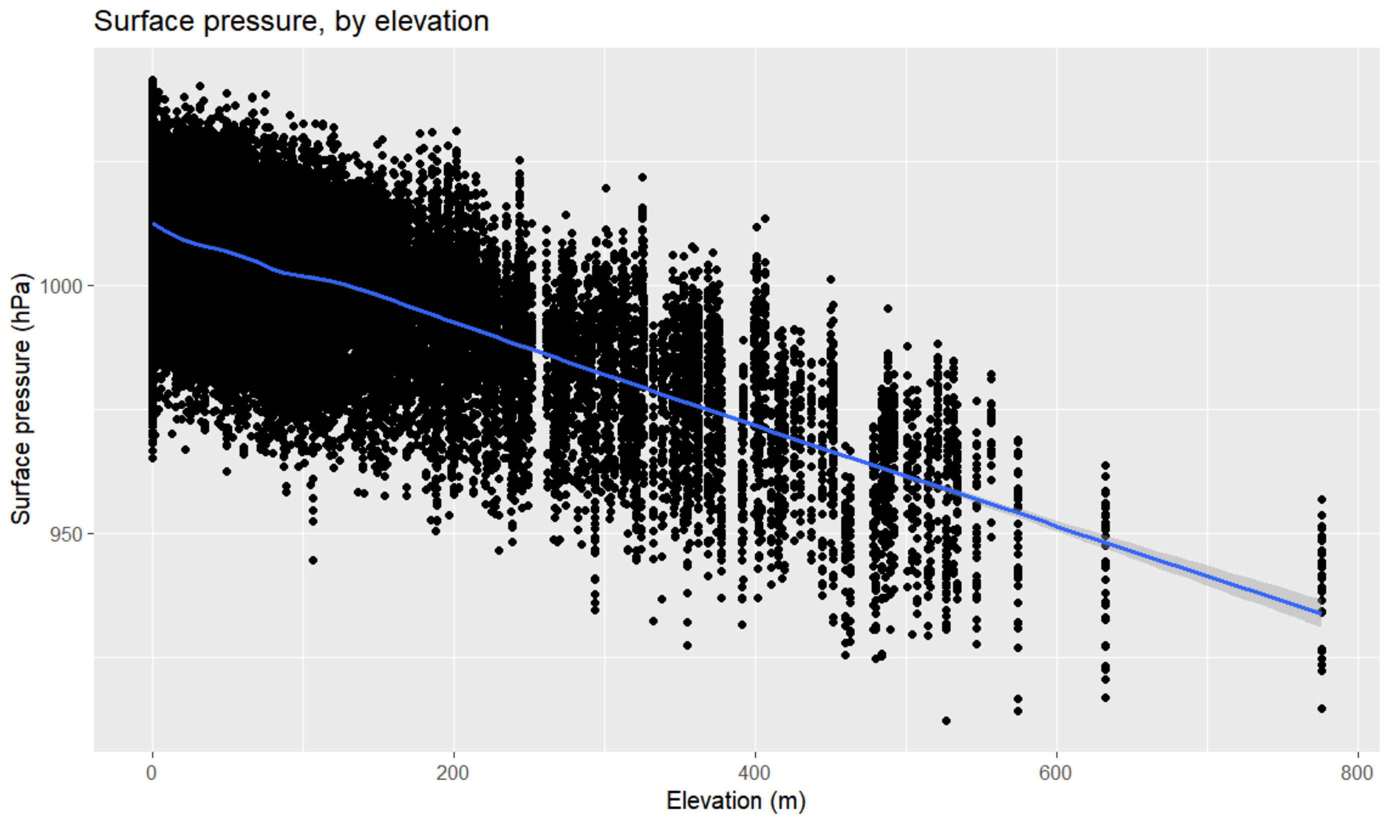


Figure 9 Elevation, surface pressure and temperature

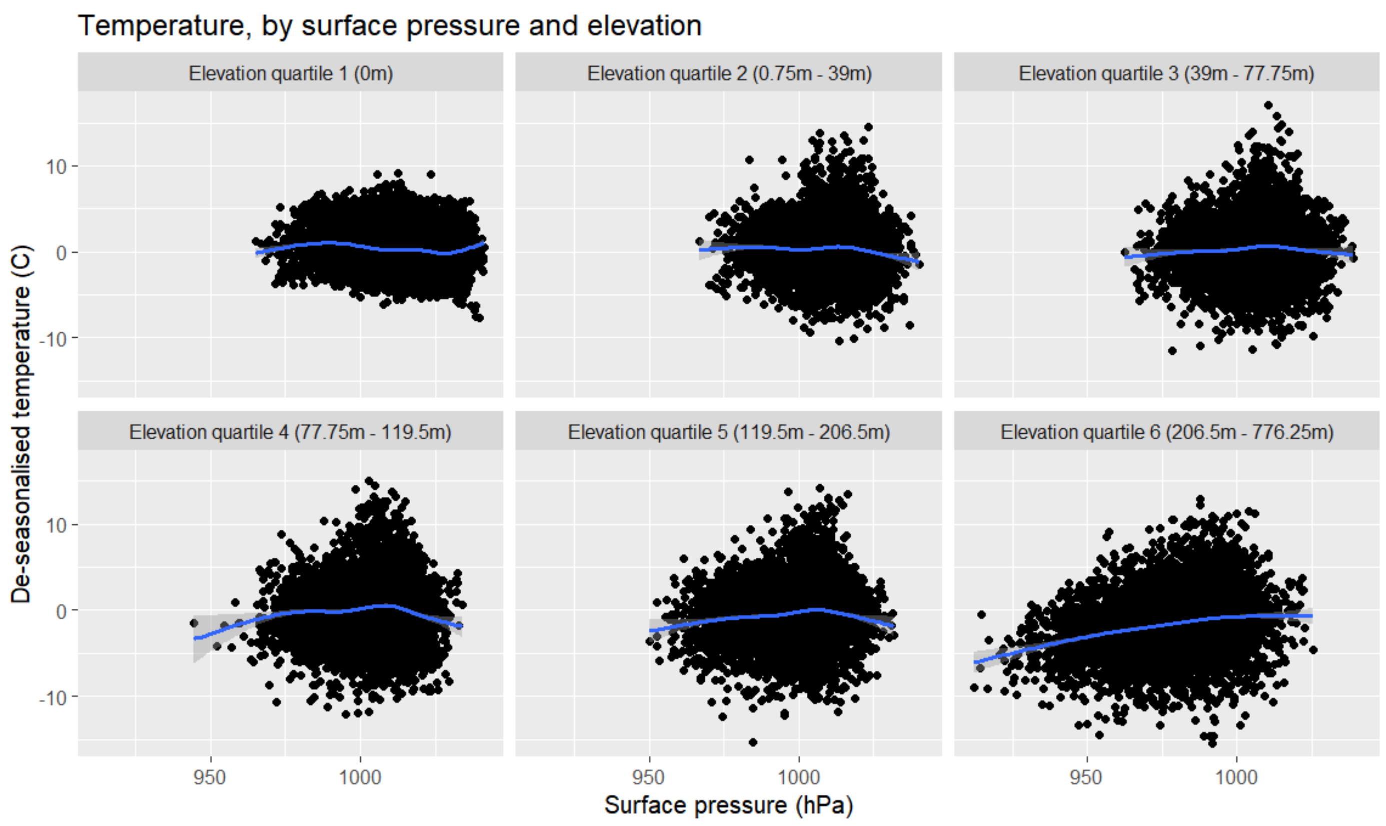


Figure 10 Model 1 KNN Actual vs Predicted

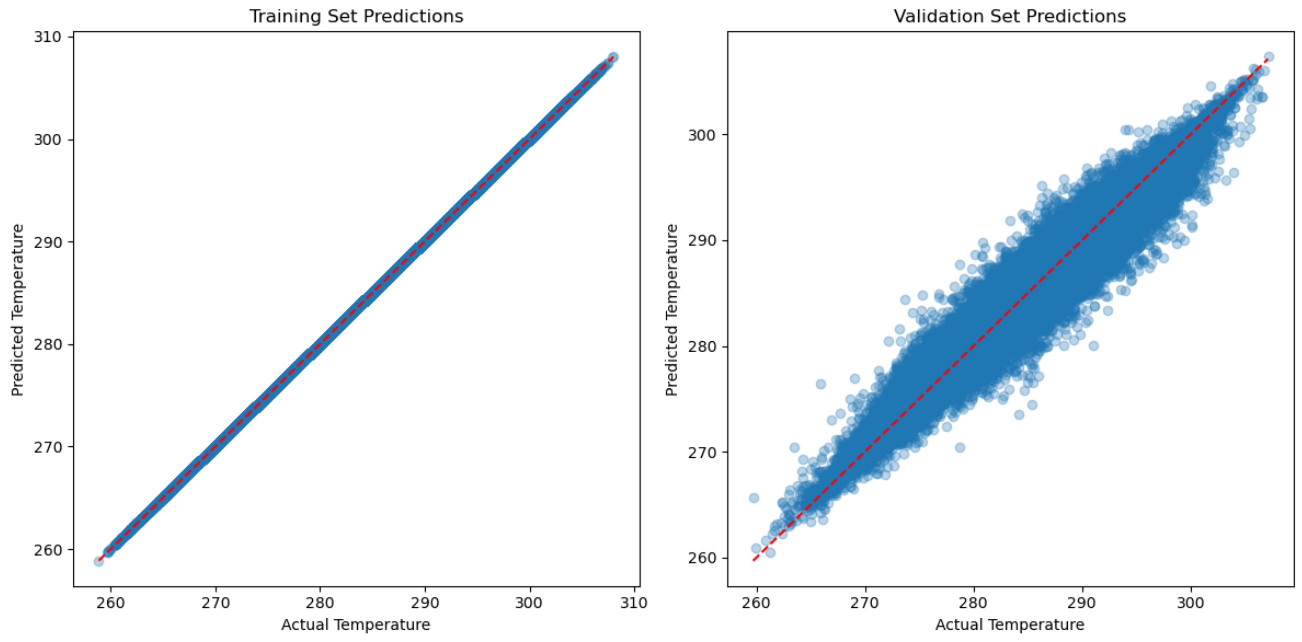


Figure 11 Model 2 Random Forest model

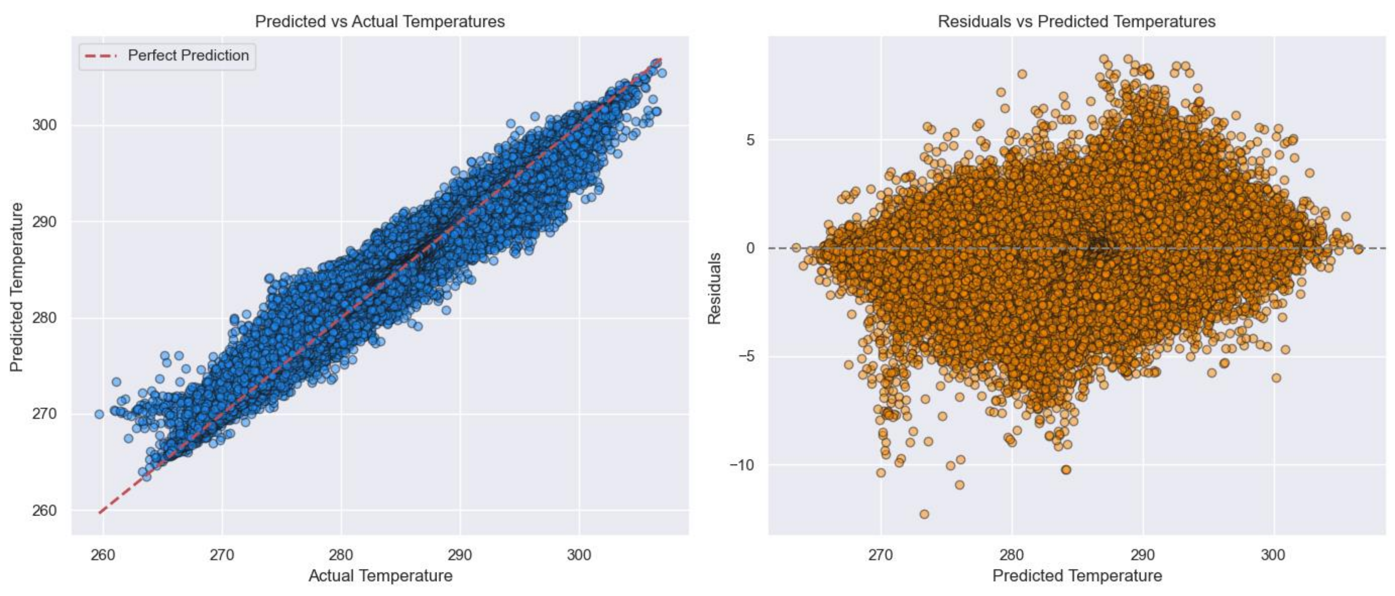


Figure 12 Model 3 XGBoost: Actual vs Predicted

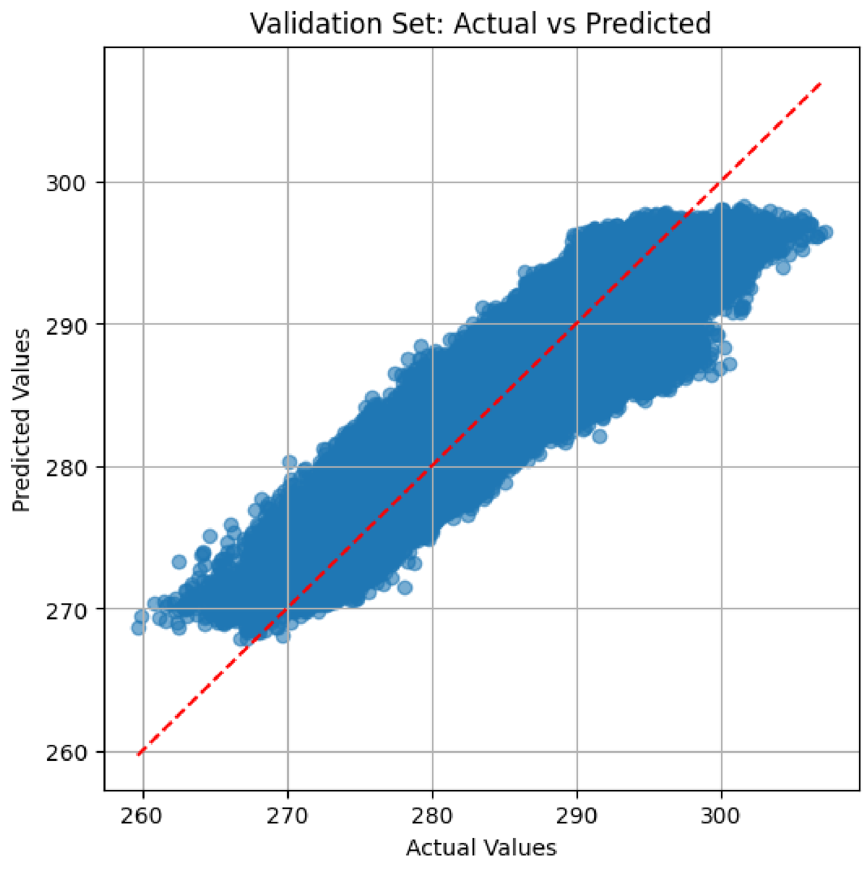


Figure 13 Model 4 Elastic Net: Actual vs Prediction

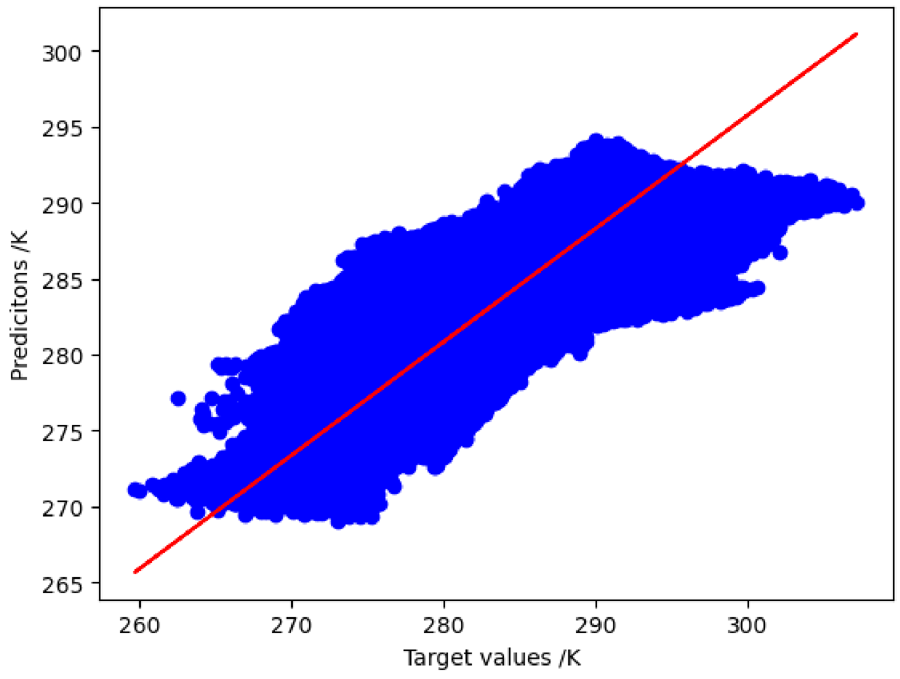
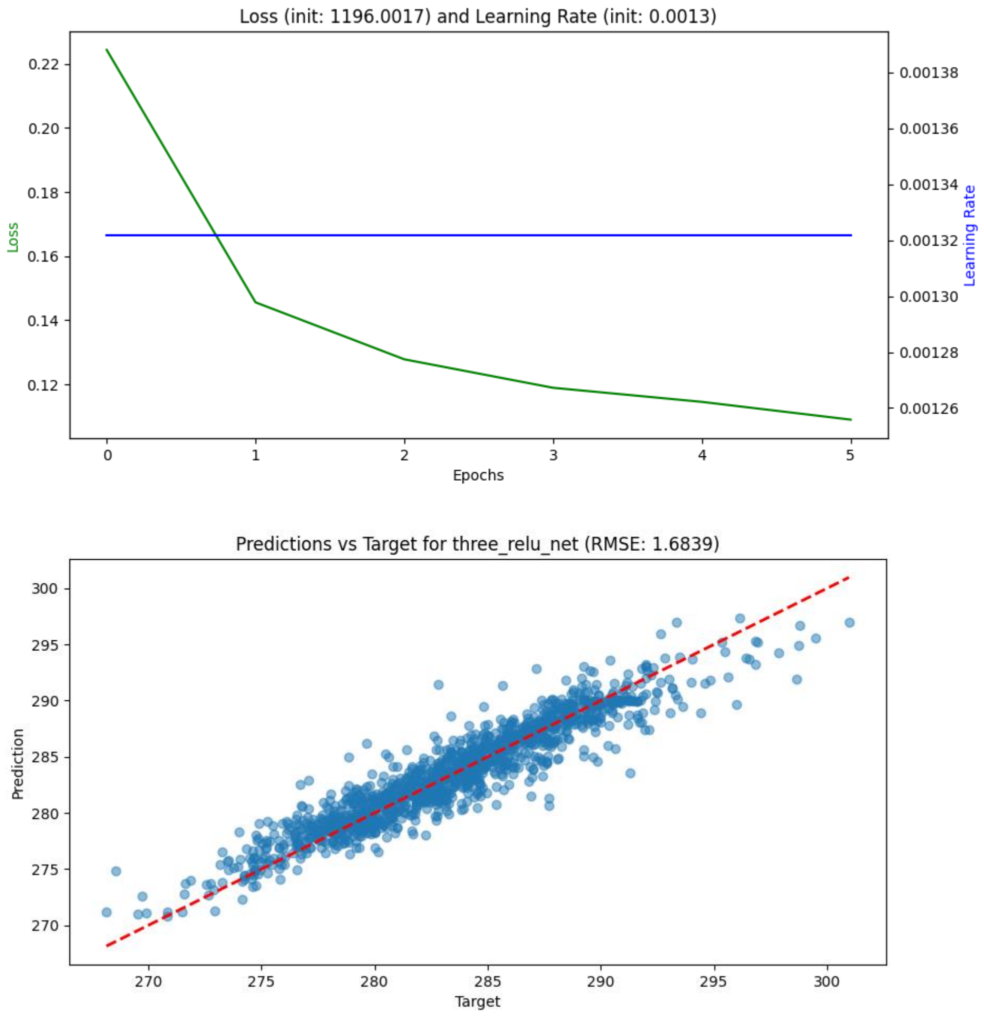


Figure 14 Model 5 Neural Networks


Loss (init: 1196.0017) and Learning Rate (init: 0.0013)
Loss
Learning Rate
Epochs
Predictions vs Target for three_relu_net (RMSE: 1.6839)
Prediction
Target