CM50264 – Machine Learning 1

Lab 5 / Project

1. What exploration of the data set was conducted?

Several methods have been applied for exploring the provided data set. Firstly, I checked to see if there were any missing entries in the dataset given that these might cause problem to the algorithms. Also, I have used the describe() method on the dataset to see the statistics on each of the columns to observe the general state of the features. These statistics include mean, standard deviation, min, max, and as shown in Figure 1.

	year	jan rain	jan mean min temperature	jan mean max temperature	feb rain	feb mean min temperature	feb mean max temperature	mar
count	31744.000000	31744.000000	31744.000000	31744.000000	31744.000000	31744.000000	31744.000000	31744.00
mean	1998.000000	82.690023	9.031864	19.652315	74.055229	10.078598	20.981471	81.62
std	8.944413	122.161265	12.953803	13.512101	97.192158	12.255713	12.702396	93.78
min	1983.000000	0.000000	-41.200000	-23.800000	0.000000	-38.100000	-18.800000	0.00
25%	1990.000000	5.700000	-0.500000	7.400000	7.100000	0.400000	9.600000	13.20
50%	1998.000000	37.800000	14.100000	26.000000	38.100000	15.100000	26.400000	50.70
75%	2006.000000	117.000000	19.800000	31.000000	106.900000	20.500000	31.300000	121.10
max	2013.000000	3476.000000	32.400000	40.200000	1009.000000	32.900000	46.500000	1102.40

8 rows × 38 columns

Figure 1 - Dataset Statistics

Furthermore, I have made several visualizations using the features of the dataset when comparing to the continuing years and months. Firstly, by averaging the feature values by year, I found the changing in yield over the years as shown on Figure 2. It is noticeable that the yield was steadily increasing.

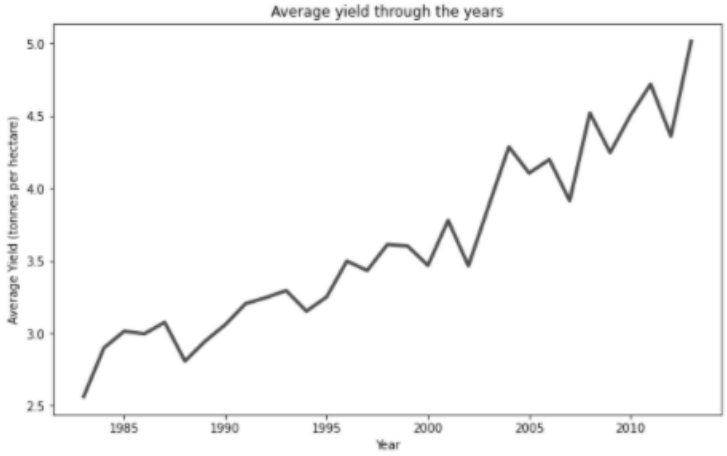

Figure 2 – Average Yield through the years Graph

Following the same method, I investigated the other features when comparing to the continuing years. Shown in Figures 3, 4, and 5 I observed the average changes in rainfall, minimum temperature, and the maximum temperature, per month, respectively.

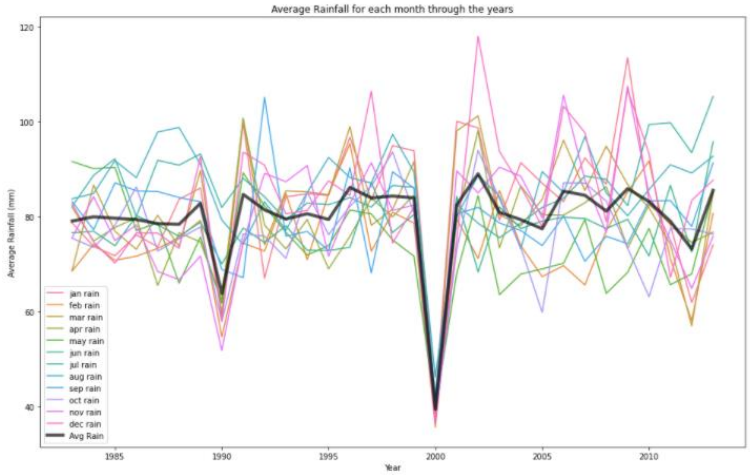

Figure 3 – Average rainfall for each month through the years Graph

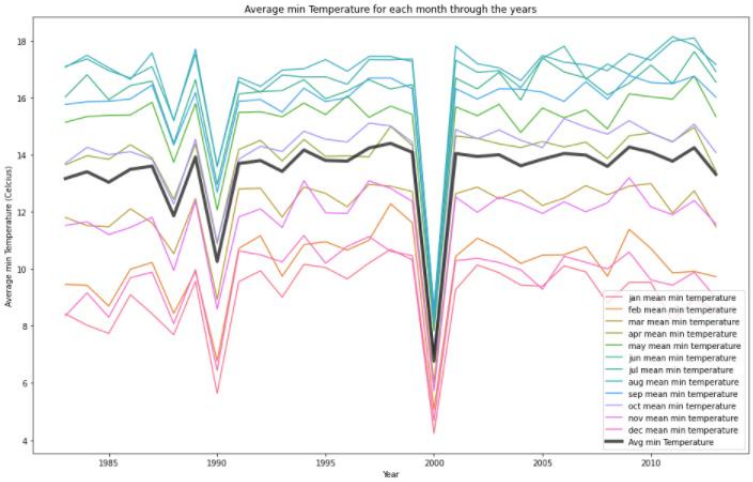

Figure 4 - Average min Temperature for each month through the years Graph

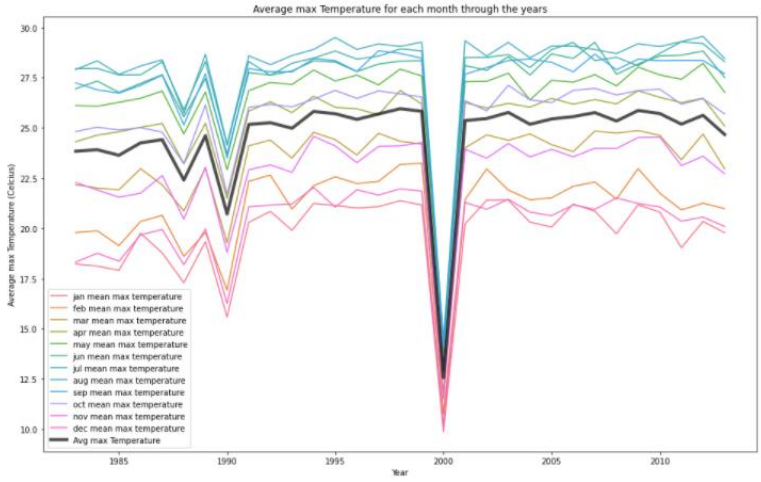

Figure 5 - Average max Temperature for each month through the years Graph

All 3 of these graphs shown in figures 3-5 have the same structure of results, meaning that they are correlated with each other. The issue is that although I could identify that some of the features are correlated, I was unable to identify a direct cause for the changing yield values.

2. How was the dataset prepared?

Firstly, I shuffled the dataset to prevent any bias during training and split the whole dataset into 70% training, 20% validation and 10% testing. Then I extracted the 'yield' column from all sets creating the target values.

I have included a validation set to use for hyperparameter tuning without it interfering with the results I ultimately got from the test set.

For feature engineering I chose to scale the data by standardizing it, resulting to the dataset columns being on the same scale with mean 0 and standard deviation 1. Since the features that are used have different scales and contribute to the analysis differently the process of standardization will remove any bias[1]. For standardizing each column subtracting its mean and divided by its standard deviation.

3. How does a regression forest work?

Regression forest works by getting several different decision trees which by themselves have high variability and combining them to make more accurate predictions. For each decision tree we use different bootstrapped samples using the method of bagging which can provide high predictive powers [2].

Depending on the number of features and the choice of features used we can find the best possible split on the feature that minimizes variance. Continuously breaking the tree down to decision nodes and creating leaf nodes when the stopping criterions have been reached.

Combining all of the trees that were created and we can make predictions using each tree and then taking their average to make a final prediction value.

Regression allows us to predict a continuous value for y based on the provided x values. Instead of classification where the prediction is only within a subset of expected outcomes. This is suitable for our task as we want to predict yield values ranging from 0 up to infinity.

4. How does a Gaussian process work?

Gaussian process is a non-parametric model which measures the similarity between the points that are created from the Kernel function to predict values on the given test set. It combines the distribution of the features in the dataset.

We define the Gaussian by obtaining its mean using a mean function and covariance matrix with a covariance sample (kernel) which are used to draw samples. This kernel function encodes all of the assumptions of similarities amongst the different data and features.

Using the predictive mean function as shown in Figure 6 with the RBF (Radial Basis Function) as the kernel Figure 7 we can get a solution with the mean value predictions to evaluate.

$$\mu_* = \mathbf{K}_{*\cdot}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y},$$

Figure 6 – Predictive Mean Function

$$k_{\mathrm{SE}}(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$$

Figure 7 – RBF Kernel

5. Which additional algorithm did you choose and why?

Scrubbing through several common regression machine learning algorithms [3], I had chosen to implement the k Nearest Neighbours algorithm. Its concept was very simple and intuitive to understand and as I was interested in seeing how well it performs in comparison to the other more complex algorithms.

The properties of KKN which I had found that made it suitable for the specific problem is that it performs well on scaled data which I have due to my future engineering method and that it can only work if there are no missing data which I had also previously checked. Additionally, it is non-parametric [4] thus, it makes no assumption on structure of the given dataset making it suitable for real world data. Lastly, KNN has lazy and instance-based learning meaning it has no training process.

6. What are the pros and cons of the algorithms?

Linear regression is both simple and computationally efficient, but its several disadvantages lie mostly on the fact that it expects that the prediction variables are linearly dependent. It expects that there is no correlation between the features, and it is greatly affected by outliners.

This is much different for random forest which can handle outliers very well and works well with non-linear data. It can handle missing data and even high dimensionality and provides variable importance helping in identifying the variable that impacts the model mostly. The main issues of random forest are that it is relatively slow and inefficient for large number of trees.

A common problem found in both Gaussian Process and k-Nearest Neighbours is the fact that they do not scale well in higher dimensional spaces and are both computationally expensive, requiring a lot of memory to run.

On the other hand, the advantages of Gaussian Process are that the predictions made interpolate observations, making it very accurate and that the predictive distribution it provides can account for the uncertainty of the output variables which can be very useful in the performance analysis stage. For KNN it can be also highly accurate with the benefit that it makes no assumptions about the data.

7. Describe the toy problem used to validate the algorithms, and explain its design?

The toy problem that I have chosen to use in order to validate the algorithms lies on a simple cubic equation with 2 variables or as also shown in Figure 8.

$$y = 5a^3 + b + 7$$

Figure 8

Variables a and b act as 2 features and impact the result for y. Each variable is a vector of 2000 randomly generated float values from 0 to 1. These 2 vectors are then placed in the equation giving us the known y values we will need. And plotting the two variables with their function result we get the graph shown on Figure 9.

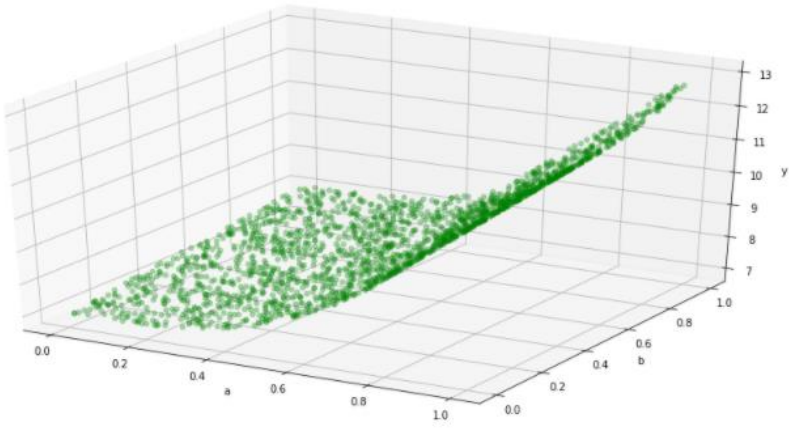

Figure 9 – Toy Problem $y=5a^3+b+7$ Graph

Again, following the same preparation methods as before but without using a validation set as I only wanted to observe the results and not tune a hyperparameter.

The expected failures and successes that I expected to find lied on the fact that I used a cubic function for y=f(x). This I expected to impact the performance of the linear regression model significantly as it completely messes up with the expectations of the algorithm about linearity.

Also, I expected to observe very good performance for the other 3 algorithms as they do not depend on the linearity on the data.

8. What evidence of correct, or incorrect, implementation did the toy problem provide?

With the results gathered I saw the exact things I was aiming for. Using some standard hyperparameter values that I have found during my testing process I saw great results for the Random Forest, Gaussian process and KNN algorithms with very low Mean Squared error (< 0.04) values and very high R Squared score (> 0.99).

The Linear Regression algorithm, as expected did not perform well in the toy problem, with an r2 score of only 0.84 and MSE over 0.3. Since I have used a cubic equation, the Linear regression model's failure was as expected, thus the implementation of the toy problem was a successful indicator on how the models are expected to perform.

9. How were the hyperparameters optimised?

For linear regression we have no hyperparameters to optimize but for the other 3 algorithms I chose the main parameters that the algorithms lie on.

For each algorithm instead of testing a combination of values for the hyperparameters which would have taken a substantial amount of time I chose to optimize the hyperparameters individually.

To test for overfitting and underfitting issues I got the mean squared error (MSE) metric on both the training set and validation set to see how the model performance changes on different hyperparameter values.

Random Forest

For Random Forest I had to make choices on which hyperparameters to tune because it had plenty. I chose to optimize number of trees, the maximum depth and the maximum number of features used.

For optimizing the **number of trees**, I choose to use a large range of values [2,3,4,5,6,7,8,9,10,11,12,13] with a default value for max depth of 10 and the square root for max features it gave me the results shown on Figure 10. It shows that the MSE decreases up to the point of having 7 trees, which after we see almost no significant improvements on both training and validation set thus my choice of optimal value is **7**. Here I chose this value not as at in terms of overfitting or underfitting but in relation to performance versus runtime, as runtimes scales linearly with the number of trees.

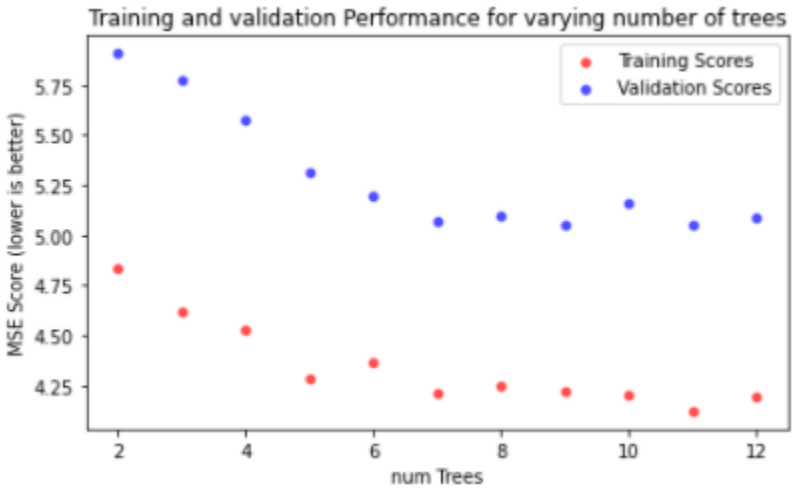

Figure 10 – Training and validation performance for varying number of trees

Using the optimal value 7 for number of trees and again using square root for max features I then tested several maximum depth values [4,6,8,10,12,14,16,18,20]. From Figure 11 it is observed that max_depth improves the performance for both sets but at different rates. After max_depth 12 the MSE for validation stops decreasing.

As the term of overfitting describes, we can get good performance on the train data but poor generalization on the validation data. So, my choice for the optimal value is based on the best result of the validation scores and also where we have the least generalization gap. Thus, for **max depth** I chose **12** for my optimal hyperparameter value.

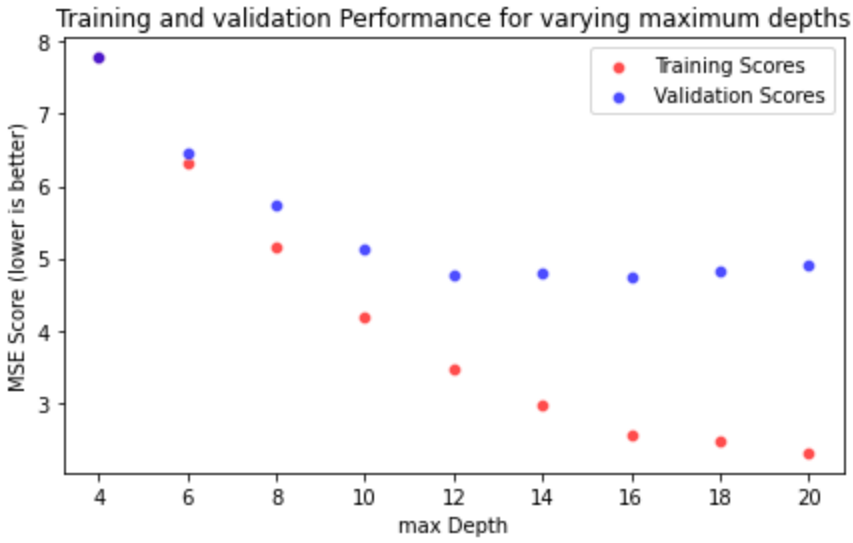

Figure 11 - Training and validation performance for varying maximum depths

With values found for both number of trees and maximum depth hyperparameters I needed to choose what **maximum number of features** to use. I had 2 options to choose from which were, using all of the 37 features or using the square root which results to 6 randomly chosen features.

With the results shown in Figure 12 I saw that using all of the features has slight improvement for both the training and validation set.

But comparing that it took 11 minutes for the algorithm using all of the features and only 2 minutes with the algorithm using the square root of features. The small difference in MSE is not justified for that amount of runtime increase thus, the optimal choice for the number of features used was the **square root**.

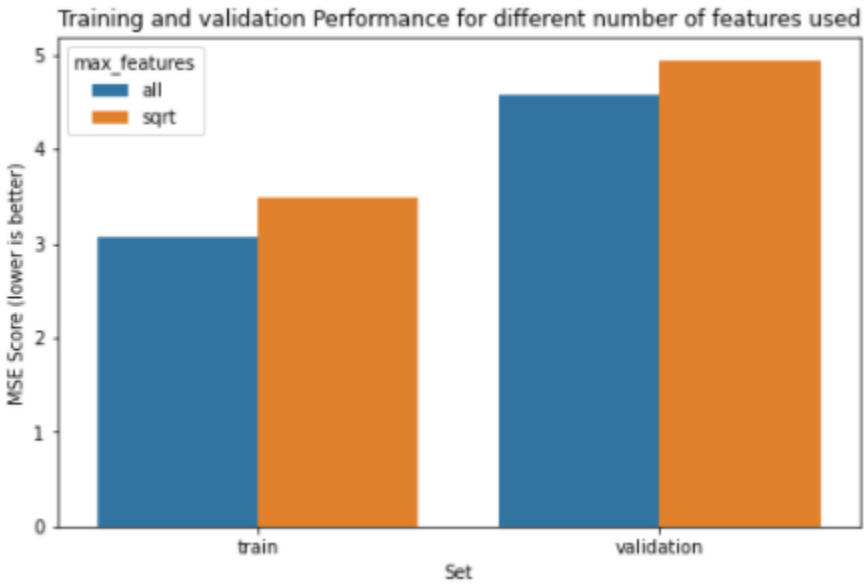

Figure 12 – Training and validation performance for different number of features used

Gaussian Process

For Gaussian Process, the 2 hyperparameters that I chose to tune is the noise represented as σ^2 in Figure 6 and the length_scale represented by l^2 in Figure 7. For the number of batches that I chose to split the training set was dependent on the limitations of my laptop as doing large batches caused "out of memory" issues. Thus, I split the training set to 5 batches and got the combined average from all 5 μ vectors.

Optimizing **length_scale**, I set the value for noise as 1 and tested on varying values of length_scale [1,2,3,4,5,6,7,8]. Shown in Figure 13 the MSE for both sets decreases up until the length scale is at 6 which after the MSE begins climbing back up thus, the optimal value should be **6**.

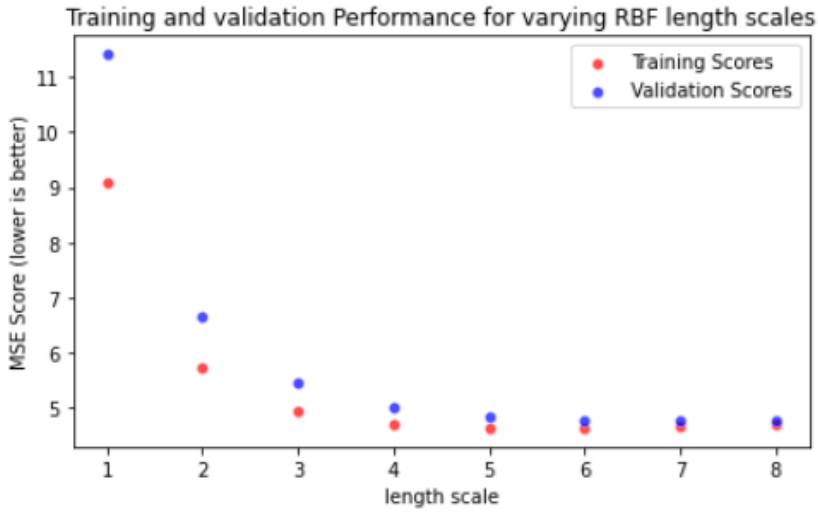

Figure 13 – Training and validation performance for varying RBF length scales

Using the value 6 for the length_scale I now needed to find the optimal value for noise. Testing on the set of values [0.1,0.5,0.75,1,1.5,2,3,4] we can observe from Figure 14 that noise has a negative impact in the performance of both sets thus the optimal value should be the lowest which is **0.1.**

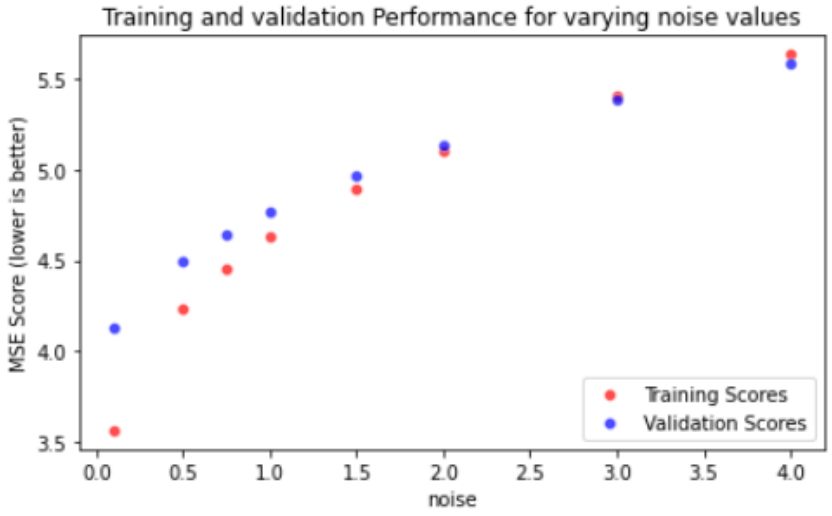

Figure 14 – Training and validation performance for varying noise values

KNN Regression

The one hyperparameter to optimize for KNN is the **k for the number of neighbours**. With the set for k values [1,2,3,4,5,6,7,8,9,10] I got the result shown in Figure 15, which I observed that for increasing k values both sets converged to the same value for MSE. But my choice of optimal value is based on the best value seen in the validation set before it all evens out, so that would be the value **5**.

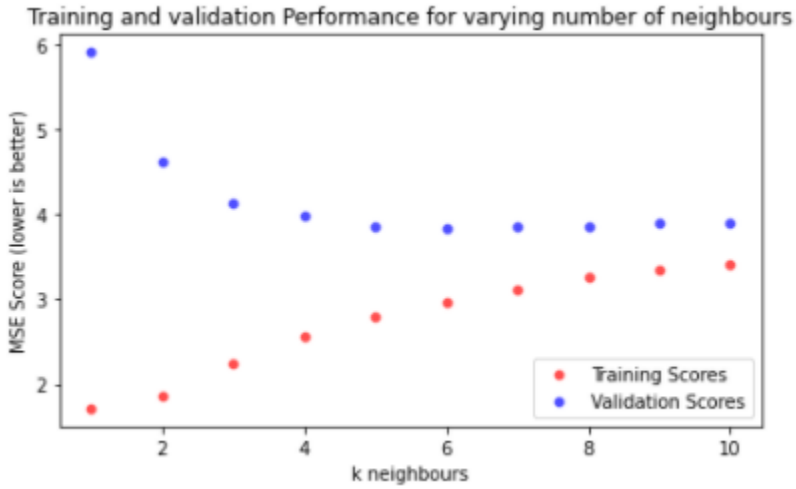

Figure 15 – Training and validation performance for varying number of neighbours

10. What results are obtained by the algorithms?

With the optimal hyperparameter values set for all of the algorithms found in the previous section I tested each one using the test set which I had kept at the beginning. As there was no issue of 'peeking' the test set is a clear indication of the actual performance on the algorithms.

Algorithm	Mean Squared Error (3dp) (lower is better)	R2 Score (3dp) (higher is better)	Runtime (seconds)
Linear Regression	9.268	0.325	0.009
Random Forest Regression	5.230	0.619	104
Gaussian Process	4.669	0.661	60
KNN Regression	4.598	0.665	37

11. How fast do the algorithms run and how fast could they run?

Gaussian and KNN only have an evaluation process.

f = number of features = 37
n = number of rows (training set) = 22221
t = number of rows (test set) = 3174
nTrees = number of trees = 7

Algorithm	Big-O Complexity (Training)	Big-O Complexity (Prediction)	Big-O Values	Runtime (seconds)
Linear Regression	$O(f^2*(n+f))$	O(f)	O(32142034)	0.009
Random Forest Regression	$O(nTrees*n^2*\sqrt{f})$	O(nTrees*f)	$O(2.1025x10^{10})$	104
Gaussian Process	-	$O(n^3)$	$O(1.0972x10^{13})$	60
KNN Regression	-	O(n*f*t)	O(2609589798)	37

Linear: $(38^2*(22221+38))+38=32142034$

Random Forest: $(7*22221^2*\sqrt{37})+(7*37)=2.1025x10^{10}$

Gaussian Process: $22221^3=1.0972\ x10^{13}$

KNN: 22221*37*3174=2609589798

Calculating the Big-O values for each algorithm, it is obvious that Gaussian Process has the most complexity, with Random Forest, KNN and Linear regression following.

But from Runtime we can observe that Big-O is not a direct indication for how the runtime behaves, as although Gaussian Process has higher complexity than Random Forest it was still 73% faster when measuring time taken on both training and predicting.

12. Which algorithm would you deploy and why?

Having the option to deploy one algorithm I would choose k-Nearest Neighbours as it has shown to perform better than every other algorithm for this specific problem having the least error as observed from the MSE metric with the 2nd best runtime following linear regression. As it was also previously stated KNN is suitable for real world applications as it is nonparametric making it the optimal choice amongst the algorithms I had tested.

13. How could the best algorithm be improved further?

Further modifications that may improve the accuracy even more can with the addition of weights. Weighted KNN is studied that it performs better on regression problems. Possible way of implementation is using inverted distance as weight making the values of the closest neighbours having a bigger impact on the predicted value.

14. If you were to try another algorithm then which one and why?

Having the option to implement and test another algorithm I would choose Elastic Net as it has the benefits of both Ridge Regression and Lasso regression. Combining Lasso's feature selection with Ridge Regression's grouping effect in combining features with high collinearity we can get the best of both worlds.

In our real world task since the features of the dataset have high collinearity this kind of algorithm would be suitable, and especially since it is a regression problem where Elastic Net excels at.

15. Are the results good enough for real world use?

The context in where these algorithms are used is described as in, predicting crop yields given climate data. With the algorithms that were implemented and the results that were gathered they are insufficient for real world use.

Using a dataset with over 30000 entries and the best result with get has an R-squared score of only 0.665 is unsuitable for real world applications. These sub-par results may have been due to poor algorithmic choices or limitations of the features in the dataset, but it shows that the predictability power lacks significantly.

If in a real world application such as farmers using these machine-learning methods, assuming they will use the same dataset, and our implementation of the algorithms that were used, to make predictions on their future yields would be inadequate. Providing farmers with appropriate predictions on their future yields can help them make informed management and financial decisions thus giving them poor predictions can have a significant negative effect to them.

16. How could this solution fail?

The 2 big issues of the K-Nearest Neighbours algorithm in relation to the data is that it cannot work if there are missing data in the dataset and it needs the data to have the same scale.

Any missing data would result for the algorithm to fail as it will have no values to compare in finding the Euclidean distance. A way to overcome that is by filling in the empty data by using the average of their respective data columns.

Also, it needs the data to be normalized or standardized. Any feature engineering that results in the data having the same scale is significant in the overall performance of the KNN algorithm.

If any of these 2 issues have not been dealt with is enough for the solution to fail!

17. What improvements could be made to the data set?

An issue that I have noticed on the dataset is that there where many duplicate rows with different yield values. Many instances of every feature having matching values with completely different yield results may have cause a degradation in all training, validation, and testing results.

A feature that can be added which can help better predict the yield values is the amount of sunlight an area gets. Sunlight has a direct correlation on yield numbers. “A general rule of thumb is that 1% more light will give you a similar percentage increase in plant growth, resulting in a 1% higher yield.”[5] . Since we have a feature that is investigated and proved to be very important in yield results, recording it in our dataset is crucial as it will help us better predict the expected yield results.

References:

[1] Team, T. and Team, T., 2021. *How, When, And Why Should You Normalize / Standardize / Rescale Your Data?*. [online] Towards AI — The Best of Tech, Science, and Engineering. Available at: https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff [Accessed 16 January 2021]

[2] Medium. 2021. *Building A Random Forest From Scratch*. [online] Available at: https://towardsdatascience.com/building-a-random-forest-classifier-c73a4cae6781 [Accessed 16 January 2021]

[3] Codes), C., 2021. *Commonly Used Machine Learning Algorithms | Data Science*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/ [Accessed 16 January 2021]

[4] Brownlee, J., 2021. *K-Nearest Neighbors For Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/ [Accessed 16 January 2021].

[5] Farmer's Weekly. 2021. *Light And Its Effects On Plant Growth*. [online] Available at: https://www.farmersweekly.co.za/agri-technology/farming-for-tomorrow/light-effects-plant-growth/ [Accessed 16 January 2021].