Machine Learning 1: Lab 5

Part 1: What exploration of the dataset was conducted?

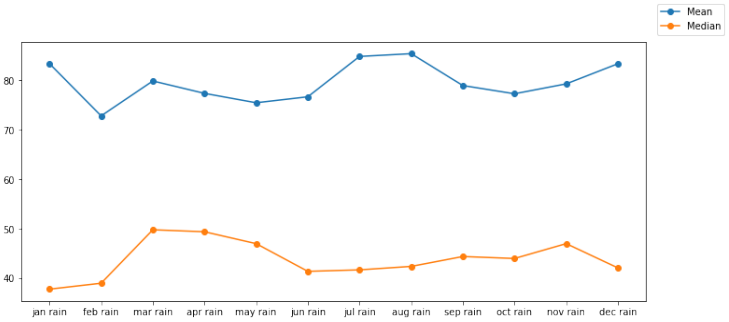
Basic Information: df.info(), a pandas method, tells us that there are 31,744 rows, 38 features (all of which are floats apart from year which is integers) and no null values.

Note: at this point I split the data into train/validate/test sets to avoid data leakage. More info about this in Part 2.

Centrality and Spread: df.describe() - another pandas method - sheds light on mean, median, standard deviation, min and max of each feature including the target. I use this to make plot 1A.

Plot 1A compares median and mean rain for each month. The gap between mean and median indicates outlier influences and further complexity.

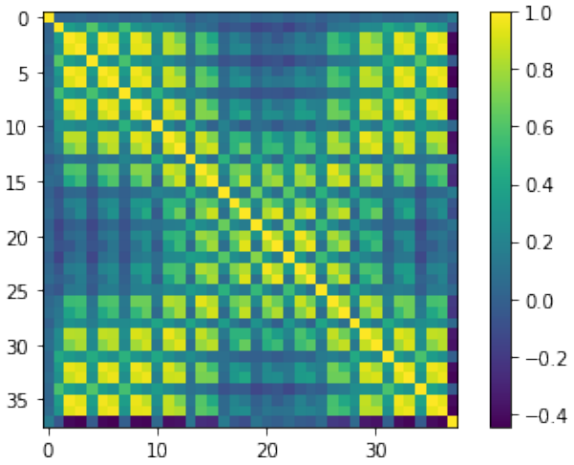
Plot 1A:



Plot 2A is a correlation heatmap of each of the 38 variables (where 0 is the first variable, year, and 37 is the last variable, yield). It suggests that for each month the average max and average min variables are highly correlated with one another. The brightness just above and below the downward diagonal suggests months close to each other are correlated (for temperature). The top-right and bottom-left brightness is because winter months are correlated but on opposite sides of the year. The final column suggests there is a negative relationship between winter month temperatures and crop yield.

[206]

Plot 2A:



Part 2: How was the dataset prepared?

The dataset has already been split into train/validate/test sets - with ratio 60:20:20 - in the previous section. The data was shuffled first then split. The validation set will be used for hyperparameter optimisation.

To reduce the strong multicollinearity between min and max temperatures in each month I replaced the two columns with one column that is an average of the min, max. The cost of doing this is a potential loss in explanatory information.

To reduce the outlier influence on the training data I removed rows that contained outliers. That is, the rows in which there was a column where the value was 4 times standard deviation away from the column average. This is not done on the year column. In total, 925 outlier rows were removed.

I chose to go 4 times standard deviation because this data will not have a singular point of centrality for each column given the data is from different parts of the world.

Finally, I standardised the data. Distance algorithms can underperform if the scales vary between different features.

[176]

Part 3 - How does a regression forest work?

A decision tree in a regression forest makes a prediction of the target variable based on the feature variables. Going deeper, depending the value of certain feature variables - i.e. if they are above or below a certain split point - narrows the prediction of the model.

The features and split points are decided in the training stage by iterating through all possibilities and selecting the 'optimal' split point. In the case of regression trees the optimal split point is the one with the highest 'variance reduction'.

A node before being split is 'parent' node. After being split, the rows go to their respective 'child' nodes. The variance of each group is calculated, and a weighted sum is taken. The variance reduction is the variance of parent node minus the weighted sum variance of the child nodes.

After a split is chosen and the child nodes are formed then these are also split using the same process until you reach the max depth - which is a hyperparameter and is a limit on the number of generations of parents you can have.

Prediction: When test data comes in, each row goes through the tree split points and the prediction is given by the mean of the final node, 'leaf', that the test point lands on.

A regression forest has multiple such trees that are trained using subsets of the whole training dataset. You can also limit the features that each tree is allowed to split on to improve the diversity of predictions. After the trees are trained - in order to predict a test data point, the data point is predicted on each tree then these predictions are averaged together to return a meta prediction.

[285]

Part 4: How does a Gaussian Process work?

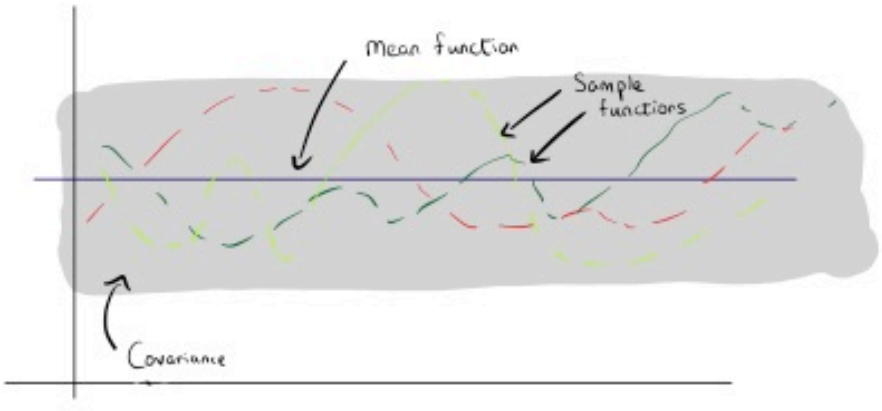
A gaussian process is a probability distribution over functions. Gaussian Processes use the paradigm of Bayesian Inference in that they model uncertainty and incorporate priors and posteriors. The prior belief (see 4A), which usually has mean function = 0, is updated to give us a posterior belief (see 4B) - by constraining the prior using the training data. That is, the training data presents function output for certain input values and the posterior belief's function space is constrained by saying all of the functions must intersect these data points.

The mean function crosses all of these data points. The standard deviation close to the training data points is narrow, whereas away from the data points the standard deviation (and confidence interval) is wider. Depending on the noise in the training data there is flexibility in training the model to adjust strictness of the constraint on the intersection points.

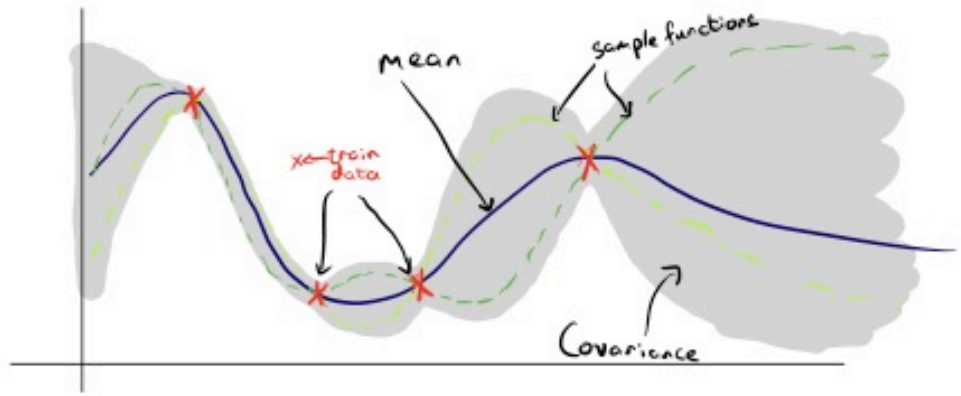
There is a further constraint that we use which is on the shape, such as smoothness, of the functions in the function space. This is defined by the kernel/covariance function. This flexibility is important otherwise the model would overfit. The mean function is used to make predictions for the test data.

[199]

Plot 4A (prior):



Plot 4B (posterior):



Part 5: Which additional algorithm did you choose and why?

I chose the K Nearest Neighbours Algorithm. KNN can be used for regression and classification. It is straightforward to digest and implement - thus saving time in creating it and improving the chances of correct implementation. It operates on the paradigm that data points 'close' to each other will have similar results. This holds true to a large extent in our data and situation. When the weather and rainfall levels are similar the crop yield will likely be similar. 'Closeness' and distance are defined how you like, Euclidean distance is what I use.

The KNN algorithm works as follows. A k, positive integer, is selected. The distance is found between the test data point and every training data point. There are different choices for distance that can be used such as Euclidean and Manhattan distance. These distances and respective data points are sorted in ascending order. The first k data points are selected (i.e. the k nearest to the test data). (In regression) These data point's mean output is calculated and used as the prediction for the test data. (In classification) The most frequent class is used as the prediction.

[190]

Part 6: What are the pros and cons of the algorithms

	Multilinear Regression	Random Forest	Gaussian Process	KNN
Interpretability	(+) Informs the relationship between features		(-) Can't interpret the effect of features using	
Uncertainty	(-) Maintain high certainty away from training data		(+) Uncertainty increases away from training data	(-) Maintain high certainty away from training data
Computation/ Prediction	(+) Once the model is trained it can be used for prediction cheaply		to take into account whole training data when making a prediction	
Implementation	(+) Easy to understand and quick to implement	(+) Easy to understand, time consuming because lots of code required	(-) Complex statistical theory and implementation	(+) Easy to understand and quick to implement

Part 7: Describe the toy problem used to validate the algorithm. Explain its design.

The design of the toy problem is as follows. There are training 500 data points. There is one feature and one target variable. Keeping the data to 2 dimensions means I can plot graphs that will be useful in diagnosing the algorithms. The x data is randomly generated between -5 and 5.

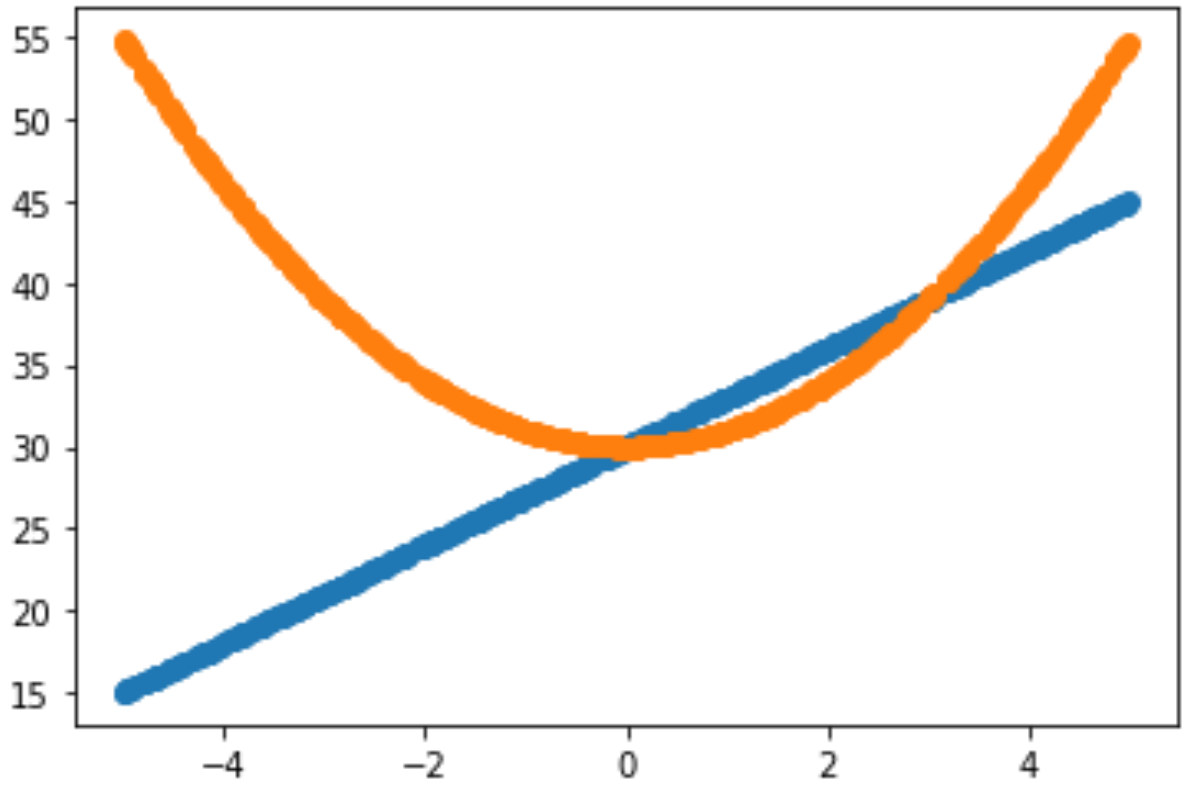
I separated the above to two toy problems. The first has a y that is generated by the function,

$$y_1 = 3x + 30$$

The second toy problem generates the y values using,

$$y_1 = x^2 + 30$$

There is no noise term because I want to see if the algorithms have the potential to be extremely accurate. For the test data, I will generate more random x data between -5 and 5 and use that to see how well the algorithms perform for each toy problem.

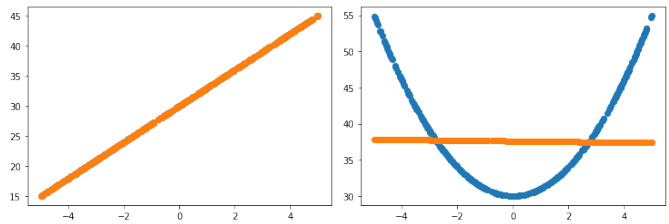


[131]

Part 8: What evidence of correct or incorrect implementation did the toy problem provide.

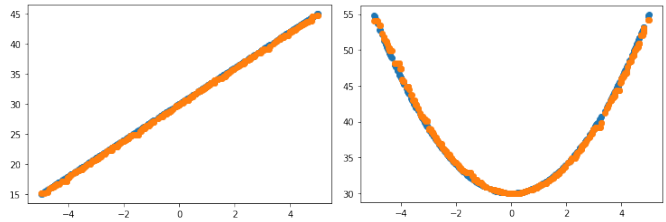
Blue = Actual, Orange = Prediction
Left = Toy Problem 1 (Linear), Right = Toy Problem 2 (Quadratic)

Multi Linear Regression -



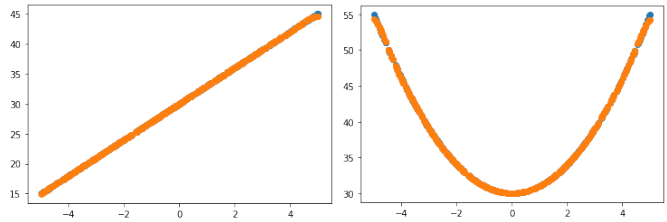
As expected, this algorithm cannot model non-linear behaviour.

Random Forest -

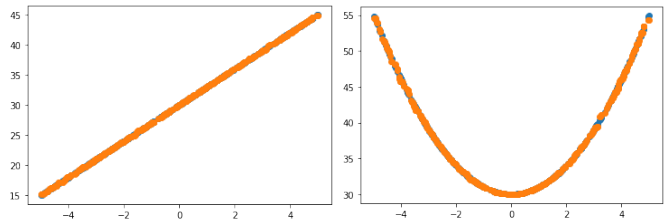


The random forest is not as sharp as the Multi-Linear for toy 1. Can manage problem 2 however near the edges the performance drops slightly

Gaussian Process -



K-Nearest Neighbours -



KNN and GP can handle both toy problems without error. Suggesting that for non-complex relationships the KNN and GP work as expected.

[86]

Part 9: How were the hyperparameter optimised

Each Algorithm uses a validation set to optimise hyperparameters. The evaluation statistics for each hyperparameter selection is calculated on the validation set. A sample of all hyperparameters are tried. The hyperparameters that yield the best evaluation statistics on the validation set are selected to use on the test set.

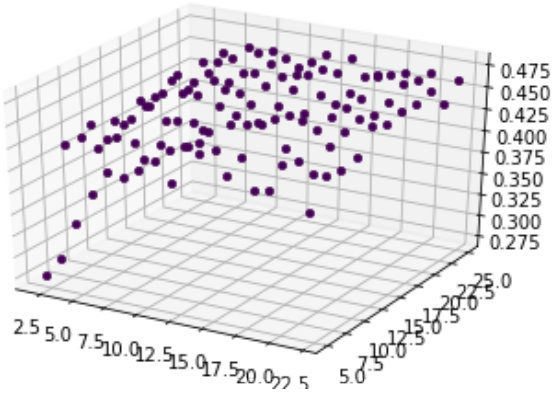
Due to a lack of computing resources and in the interest of saving time a sample between a range of the possible hyperparameter values is selected to optimise over. An example of this process is shown for the Random forest Algorithm below. But first, the hyperparameters in each algorithm are;

Multi-Linear: Number of iterations and Learning rate
Random Forest: # of trees, max. depth, # of samples and features each tree trains on
Gaussian Process: Kernel parameters ('l' for smoothness, 'sigma f' for kernel noise, 'sigma y' for noise in training data)
KNN: K value (# of neighbours used for prediction)

Hyperparameter Optimisation in the Random Forest:
Sample output:

```
Trees: 5, Depth: 2 - MAE: 0.636, MSE: 0.723, R Squared: 0.277
Trees: 5, Depth: 4 - MAE: 0.552, MSE: 0.567, R Squared: 0.433
Trees: 5, Depth: 6 - MAE: 0.524, MSE: 0.540, R Squared: 0.460
Trees: 5, Depth: 8 - MAE: 0.526, MSE: 0.547, R Squared: 0.453
Trees: 5, Depth: 10 - MAE: 0.539, MSE: 0.567, R Squared: 0.433
Trees: 5, Depth: 12 - MAE: 0.560, MSE: 0.587, R Squared: 0.413
Trees: 5, Depth: 14 - MAE: 0.526, MSE: 0.550, R Squared: 0.450
Trees: 5, Depth: 16 - MAE: 0.546, MSE: 0.567, R Squared: 0.433
Trees: 5, Depth: 18 - MAE: 0.552, MSE: 0.579, R Squared: 0.421
Trees: 5, Depth: 20 - MAE: 0.532, MSE: 0.545, R Squared: 0.455
Trees: 5, Depth: 22 - MAE: 0.554, MSE: 0.591, R Squared: 0.409
Trees: 7, Depth: 2 - MAE: 0.613, MSE: 0.715, R Squared: 0.285
```

Plot of R-Squared against # of trees and max depth:



It is not the best graph to decide from, however using the explicit output data in the notebook I determined that 21 trees and a max depth of 18 is optimal for the range that I considered. Similar approach is used for the other algorithms, can be found in the notebook. [219]

Part 10 - What results are obtained by the Algorithms

The evaluation metrics I considered are the mean absolute error, mean squared error and r-squared. These are all standardised however.

Multi-Linear Regression -

Mean Absolute Error = 0.582
Mean Squared Error = 0.686
R-Squared = 0.314

Random Forest -

Mean Absolute Error = 0.510
Mean Squared Error = 0.533
R-Squared = 0.467

Gaussian Process -

Mean Absolute Error = 0.417
Mean Squared Error = 0.388
R-Squared = 0.612

KNN -

Mean Absolute Error = 0.397
Mean Squared Error = 0.378
R-Squared = 0.622

[83]

Part 11 - How fast do the algorithms run and how fast could they run

n = number of training points
d = number of dimensions
k = number of prediction points
t = number of decision trees

Multi-Linear Regression -

Training Speed: 24.53 seconds [19,046 training points]
Prediction Speed: 0.00 seconds [6,349 test points]
Time Complexity: O(n*d) [11]

Random Forest -

Training Speed: 1 Minute 11.45 Seconds [19,046 training points]
Prediction Speed: 0.23 Seconds [6,349 test points]
Time Complexity: O(n*log(n)*d*t) [11]

Gaussian Process -

Training Speed: Everything is processed during the prediction
Prediction Speed: 7 Minutes 54.23 Seconds [19,046 training points, 6,349 test points]
Time Complexity: $O(n^3)$ [from the lecture notes]

KNN -

Training Speed: Everything is processed during the prediction
Prediction Speed: 56.87 Seconds [19,046 training points, 6,349 test points]
Time Complexity: The KNN loops through every training observation (n data rows), computes the distance between all training set observations and the test observation. It then sorts these distances. Thus, we have O(p*n*d) + O(p*n) [second part is for sorting]

[11] Paritosh Kumar. *Computational Complexity of ML Models*. [24/12/2019] https://medium.com/@paritoshkumar_5426/time-complexity-of-ml-models-4ec39fad2770 [Accessed 19/01/2021]

Part 12 - Which algorithm would you deploy and why?

I would deploy the K Nearest Neighbours Algorithm for the following reasons;
1. Accuracy: It provided the best evaluation metrics out of the 4 algorithms (cf Part 10)

2. Live Updates: The model can be updated with new training data by simply appending the new data to the training set used by the model.

3. Implementation and debugging speed: The KNN algorithm was super quick to figure out and implement. The assumption is; that the simplicity allows for debugging, updating, changing and repurposing with ease and accuracy. Less prone to self-defeating errors.

Drawbacks of the KNN are that;
It is not a parametric algorithm, so it takes significantly longer time to make predictions compared to random forests. Considering the circumstances - yearly crop yield prediction - the algorithm will most likely not be used in environments that require instant predictions.

Reasons for not deploying the others:
ML: Cannot handle non-linear relationships
Random Forest: R-Squared >13% lower than either KNN and GP
Gaussian Process: Not optimised properly due to size and lack of computing power available [175]

Part 13 - How could the best algorithm be improved further

The best algorithm in this case is the KNN. Some modifications include (a) adding weights and (b) applying Principal Component Analysis to the dataset

(a) The Weighted KNN algorithm places more value on the 'opinions' of the data points that are closer to the test point. This may improve the performance of the algorithm when dealing with test points that are not similar to any of the training data. In the situation where the k or k-1 neighbour is very different, a weighted knn's prediction isn't skewed to the distant value by much - which is what we might want.

(b) The KNN is based on distance measurement. If there are many multicollinear features in the model, then the distance metric is biased towards the multicollinear features ('crowded out'). PCA reduces dimensionality and multi-collinearity. Thus, the KNN will likely perform better on a PCA reduced dataset.

[146]

Part 14 - If you were to try another algorithm, then which one and why?

I would try a Neural Network. Neural Networks are flexible models that adapt to the shape of the data. So, it can dynamically pick the type of regression in the case that the relationship between predictors and target is not obvious. Since Neural Networks can have many layers, they can be useful for models with a lot of complexity.

Neural Networks have been applied to variety of datasets (visual, audio, etc.). They are dynamic and applicable to a lot of circumstances. For the selfish reason of being a better equipped data scientist I would want to have a Neural Network under my belt.

There are many methods to avoiding overfitting and underfitting in Neural Networks including; retraining, training multiple neural networks in parallel, early stopping, adding neuron layers, 'dropouts', etc. Given the volume of data, overfitting is a higher risk, and having an algorithm that can handle that better is valuable.

In a research paper [14.1], multiple linear regression, projection pursuit regression and neural networks were used to predict crop yield and they found that the neural network outperformed the other two.

[14.1] Drummond, S. T., Sudduth, K. A., Joshi, A., Birrell, S. J., and Kitchen, N. R. (2003). Statistical and neural methods for site specific yield prediction. Link: https://elibrary.asabe.org/abstract.asp??JID=3&AID=12541&CID=t2003&v=46&i=1&T=1

Part 15 - Are the results good enough for Real World use?

Situation 1 (in detail): The context that this algorithm might be used in is commodities trading. Commodities, such as crop yields, are commonly sold as securities on trading markets. Farmers will presell their following years crops as securities that are bought by traders for a fixed price. The farmer put for sale of their crops for the following year on sale for a fixed price. If the yield is low the trader who bought the security loses out and if the yield is high, then he takes home the profit. The farmers pass on the risk and reward of crop yield levels to commodity traders.

Thus, these commodities traders/institutions will need to decide if it is a worthwhile investment to purchase the security and bear the risk of low yields. When making such a decision they may want to incorporate crop yield predictions and uncertainty into their calculations. Unfortunately, a 62% R-Squared is significantly inaccurate.

Considering the tight margins that traders have to make their profits, this prediction would leave too much to chance for it to add any real value in this context. If crop yields are a unit lower than projected (which is likely when the r squared is so low), the traders take a big hit on profits.

Further situations that require yield predictions include [15.1]: Policy makers when making export/import decisions for national food security, "Seed companies need to predict the performances of new hybrids in various environments to breed for better varieties", and farmers can use the information to make their own financial decisions.

[259]

[15.1] Crop Yield Prediction using Deep Neural Networks [22 May 2019] *https://www.frontiersin.org/articles/10.3389/fpls.2019.00621/full* [Accessed 18/01/2021]

Part 16 - How could this solution fail?

Extrapolated Predictions: When a KNN algorithm tries to predict a data point that is not similar to any training data then it will face trouble. If the k nearest neighbours are very different to the test point then the prediction will likely be off by a lot. This would be okay if the KNN also provided uncertainty with its predictions like a gaussian is able to do. But KNN doesn't provide uncertainty, this issue could lead to failure.

"No Free Lunch": The model is a simplification of reality. In making the simplifications we are discarding information, noise, fluff or whatever else. We are focusing on the signals, the important information. However, it may be the case that what is actually important is different to what the model focuses on. If there are significant changes happening in the background that the model doesn't capture with its limited vision, then it can lead to failure. What if there is a pandemic that means farmers can't reach the farm to plant crops, the weather will be normal, but yields will be different to predictions.

Part 17 - What improvements could be made to the dataset

Features that, if added, could improve performance include:

- *Location Data* - including a metric for location data could capture the forces at play in the environment, all of which you couldn't track. For example, if there are two farms in very different locations but with similar temperature and rain levels, there will be factors that make the data point to predict more similar to the farm that is geographically closer to it.

- *Crop type* - the behaviours within a crop class will be similar. Including a factor variable for crop type would probably improve prediction - assuming the current data set isn't already of just one crop type.

- *Crop genotype* - in predicting crop yields, the genotype of the crops are important features that are sometimes used in and of themselves to predict crop yields. [17.1]

- *Soil data* - information about the soil plays a part in the quality and quantity of crops. Soil information that could improve performance: "percentage of clay, silt and sand, available water capacity (AWC), soil pH, organic matter (OM), cation-exchange capacity (CEC), and soil saturated hydraulic conductivity (KSAT)"[17.1].

[187]

[17.1] Crop Yield Prediction using Deep Neural Networks [22 May 2019] *https://www.frontiersin.org/articles/10.3389/fpls.2019.00621/full* [Accessed 18/01/2021]