Assignment 2 - Group: 29

Tutors: The Canh Dinh, Fangzhou Shi

Group members : Gautam Radhakrishnan Ajit (grad0149), Lavanshu Agrawal (lagr7305), Renil Austin Mendez (rmen0735)

1. ABSTRACT

Forest fires have been a cause for concern for environmentalists. Predicting the probability of their occurrence could not only help save the ecosystems, but also help the authorities better allocate their resources efficiently. By taking a closer look into those factors that have share a correlation with the occurrence of forest fires, appropriate regression models could be fit and trained to achieve the same. The Forest Fires dataset focuses on the real-world data concerning the past happenings of forest fires in the north-eastern region of Portugal, with the area of the affected region representing the severity of the fire. Popular regression models were fit and fine-tuned to boost the performance and reduce the mean squared error. The five models used include SVM (linear, radial and polynomial of degree 2), XGBoost regression, Random Forest, linear regression, and kNN models. Overall, Random Forest classifier produced the least mean squared error and, arguably, performed the best after hyper-parameter tuning.

2. INTRODUCTION

The Forest Fires dataset consists of 517 observations with 12 feature columns and 1 dependent feature (Area), which is what the model plans to estimate on the test set. These features include metrics that measure the fire velocity speed, the fire intensity, relative humidity in the atmosphere, temperature, the impact of rain in the region, and the dates on which past accidents took place. These factors contribute to the extent of damage the forest fires cause, and thus help in boosting the predictive power of the models.

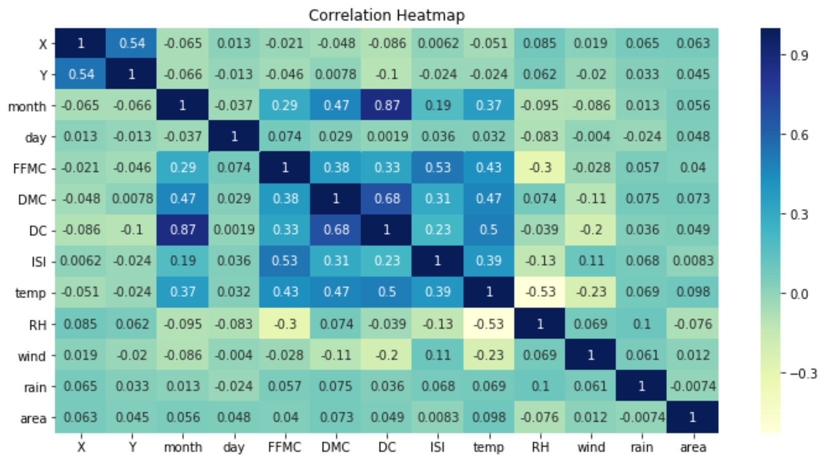


Fig 1. Correlation heatmap of the features in the dataset

From *figure 1*, it can be seen that there is not a significant direct relationship between the 12 features and the 'Area' variable. However, there are interdependencies between the features among themselves. The features that show such relationships are mostly related to weather and temperature. It can also be seen that the date attributes, especially month, could also potentially boost the performance upon further analysis.

Although the dataset details the occurrence of forest fires limited to the north-eastern region of Portugal, the factors that contribute to the spread and intensity of these fires are relevant to other cases as well, and could positively influence the way such cases are dealt with by the authorities. Relocation procedures could be followed to ensure that the flora and fauna of those regions where these fires are likely to break out can be saved. Moreover, this could help the people involved save a lot of time and money by streamlining the entire process, on a global scale.

3. PREVIOUS WORK

Apart from the small set of training samples that are available for training the different models, another challenge this dataset poses is that, in terms of the distribution of the target variable (Area), 247 samples correspond to fires that impacted negligible areas of forest fire. Since this could limit the potential of modeling as the data is biased, the areas were transformed using a logarithmic function, similar to the preprocessing method used in the research paper. Furthermore, the date features have been label encoded.The distribution before and after the transformation is visually represented in figure2 below:

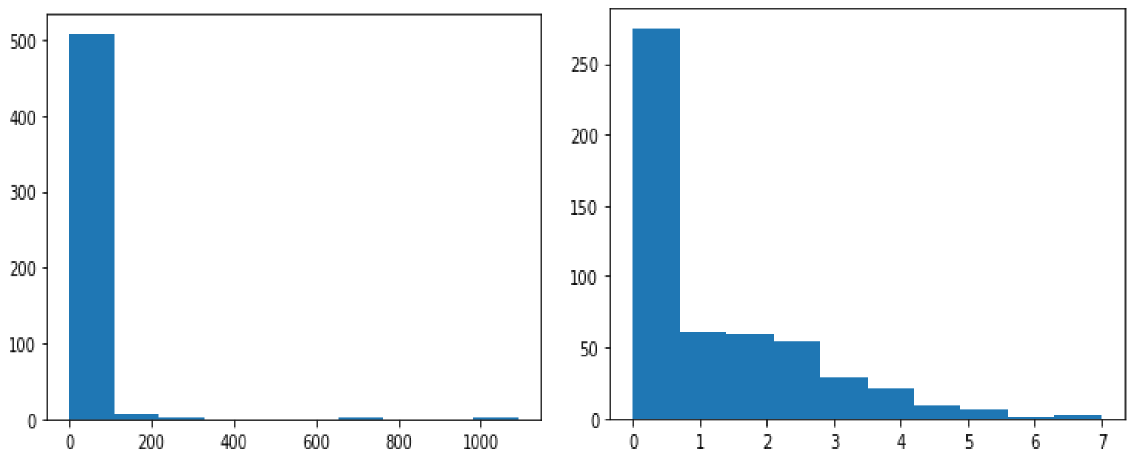


Fig 2. Distribution of 'Area' before (left) and after (right) log transformation

However, despite the similarity in the preprocessing steps, the approach to modelling is different. Instead of focusing on the four weather variables to fit the models like was done in the paper, the five models were trained using all the features available. The extent to which feature selection was done in the paper did not minimise the mean squared error corresponding to the fitting of the models used in this study. Computation costs were not penalised. The absence of missing values in the given data also encouraged the use of kNN, an algorithm not explored for this dataset. Additionally, Principal Component Analysis (PCA) was implemented and the dataset was transformed accordingly before training the linear regression model. The dimensionality reduction technique helped in minimising the mean squared error and the negative log likelihood further. (1)

4. METHODS

4.1 Random Forest

In terms of overall performance, random forest model produced the least error on the test set. An extension of decision trees, random forest algorithm ensures that the train data are randomly sampled to produce trees. This randomness minimises overfitting the training data and boosts the predictive power of the model, hence minimising the error. (2) Four of the important parameters were hyper-tuned to find the best fit.

Parameter	Description
N_estimators	Number of trees in the forest. Increasing the value will train the data better, but can also increase the computation time
Max_features	Maximum number of features that can be taken at each split. Default value is the square root of the total number of features.
Min_sample_split	Minimum number of samples to consider at each split. Could increase the computation cost and cause overfitting.
Max_depth	Maximum depth of each tree. Increasing the depth too much could overfit the training data and result in poor generalisation.

4.2 XGBoost Regressor

XGBoost is based on the concept of tree ensemble. After each iteration, the model tries to reduce the error made in the previous iteration by using the gradient boosting algorithm. XGBoost generally gives good results and is faster than most gradient boosting methods. Three approaches have been explored in this study - default parameters, optimised parameters after hyper-tuning, and using the *gblinear* linear model. (5) Apart from ‘max_depth’, and ‘n_estimators’ which are similar to the ones used by the random forest algorithm, two other parameters have also been tuned.

Parameter	Description
Learning_rate	Determines how fast or slow the algorithm finds the minimum error.
Subsample	Percentage of observations taken to build the tree

4.3 Support Vector Machines

Support Vector Machines map the training data to a higher dimension using non-linear mapping, so as to view the data points linearly in the higher dimension. The aim here is to find a function that differ from the actual value by a value ε for each point in the training data (hyper-plane). This function would later be used on the test set to make predictions. The objective function behind linear kernel is given below (3):

For linear model,

$|y-(x'\beta+b)| \leq \varepsilon$ *, where y is the actual 'area' given in the training set and x' is the predicted value of 'area'.*

For polynomial kernel with degree 2,

G(xj,xk) = (1 + xj'xk)^2 where xj'xk denotes the dot product.

For radial kernel,

K(xj,xk) = exp(-(||x-,xk||^2) / 2(sigma^2)) , where ||x-,xk||^2 indicates the Euclidean distance and sigma is a free parameter.

Two other kernels, polynomial regression and radial kernel, have been used in this assignment to study and compare the relative performances on this dataset. These kernels could be useful to find mappings that would otherwise be limited by a linear function.

To improve the computation time of the SVM model, Principal Component Analysis, a dimensionality reduction technique was applied. The theory behind PCA will be discussed in detail later.

4.4 k-Nearest Neighbor

k-Nearest Neighbor algorithm is mostly used on classification problems. However, it can be extended to regression as well, as was done in this study. 'K' indicates the number of neighbors the algorithm takes into consideration for minimising the mean squared error. The distance metric used here is Euclidean. By increasing the value of 'k', the algorithm becomes more conservative and produce results that are closer to the actual value in the training set. However, the decrease in the error metrics become marginal after a threshold value, and increasing the value even further could result in overfitting.

4.5 Linear regression model

Linear regression models are used to study the relationship between a dependent variable and two or more independent variables. Depending on the degree of the function, these models could be used to identify both linear and non-linear relationships. Apart from simple linear regression, polynomial regression of degree 2 has also been implemented in this study to compare and find the optimum model. Polynomial regression of degrees greater than 2 were found to be ineffective as they tend to lay more emphasis on outliers and thus drastically increased the error metrics.

4.6 Principal Component Analysis

Principal Component Analysis is a dimensionality reduction technique used to reduce the dimensionality of the training set. Depending upon the variance in the training set that needs to be captured, the number of components can be increased/decreased to capture more/less variance respectively. This helps in reducing the computation cost without sacrificing the results. For this study, PCA was implemented to capture 80% of the variance and the transformed training set was used to model the support vector machine models.

4.7 Preprocessing

Apart from the logarithmic transformation used to make the data more symmetric with respect to the area, the data was also scaled using the StandardScaler package in order to not let the features that are not important influence the data prediction. Although tree based models like Random Forest and XGBoost do not require scaling, the scaling made a difference for k-Nearest Neighbour model where hughe difference in variances could badly affect the performance of the model. The original dataset has no missing values, and thus, on that front, needs less preprocessing.

5. EXPERIMENTS AND DISCUSSION

10 fold cross validation was performed on the data to produce the evaluation metrics, which was later used for comparison analysis of the models. The cross validation process allocates 1 fold to the test data and 9 folds to the train data. Then the evaluation metrics such as Mean Square Error (MSE), Negative Log Likelihood (NLL), Root Mean Square Error (RMSE) and Mean Absolute Deviation (MAD) was calculated on the test data. After performing 10 iterations of predicting test data folds, the average of the above mentioned evaluation metrics were considered as the final result of the model for comparison.

Mean Square Error (MSE)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2$$

where N is the number of data points, f_i the value returned by the model and y_i the actual value for data point i.

Negative Log Likelihood (NLL)

$$l = \frac{n}{2} ln(2\pi\sigma^2) + \frac{\sum_{i=1}^{n}(y_i - mx_i - b)^2}{2\sigma^2}$$

*Where 'n' is the number of observations, yi is the actual value of the ith observation and (m*xi + b) is the predicted value.* (4)

5.1: Support Vector Machine – Regression

Different functions such as linear, polynomial and radial basis was tried on SVM to get the optimal model. The outcomes of their model is given below:

SVM Function	MSE	NLL	RMSE	MAD
linear	4662.7	644.1	53.3	16.5
Radial	5060.7	314.3	56.1	17.9
Polynomial (degree = 2)	6.48 e+78	786.2	8.052 e+38	1.1 e+38

From the above results, linear function performs well follows by radial basis function. Polynomial function is giving a poor predictive result. As we increase the polynomial degree, the model is going out the boundary and predicating infinity for some observations which makes it a bad choice for this dataset.

Principal component analysis was performed on the model. The image below shows the cumulative explained variance of different number of features on the linear SVM model.

The result below shows the performance of PCA on linear SVM model.

PCA Variance	MSE	NLL	RMSE	MAD
1.0	4662.7	644.1	53.3	16.5
0.8	4652.9	15502.7	53.2	16.5

0.7	4665.6	11723.2	53.3	16.5

The model performs slightly better when PCA is done with overall variance of 80 percentage.

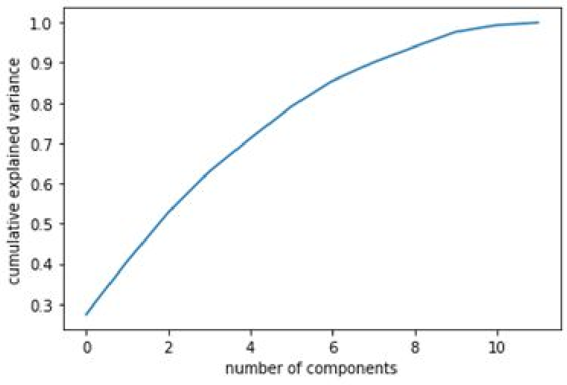


Fig 3. Cumulative variance explained vs. number of components (PCA)

Figure 3 visually represents the relationship between the number of components and the cumulative variance that is explained by that number. For this study, the components were chosen so as to capture 80% of the variance, and this was done by choosing the first six components, hence reducing the dimensionality from 10 to 6.

5.2 XGBoost Regression

Hyper-parameter tuning was performed on XGBoost to find the optimal parameter of the model that provides good predictive results. The main parameter under consideration for the tuning were learning rate, number of trees, maximum depth and subsample of the trees. The data was split into 70% of train data and 30% of test for the tuning process. The parameters which correspond to the least RMSE for both the train and test data is taken as the final parameter for the model. The images below show the results and parameters chosen for the XGBoost model.

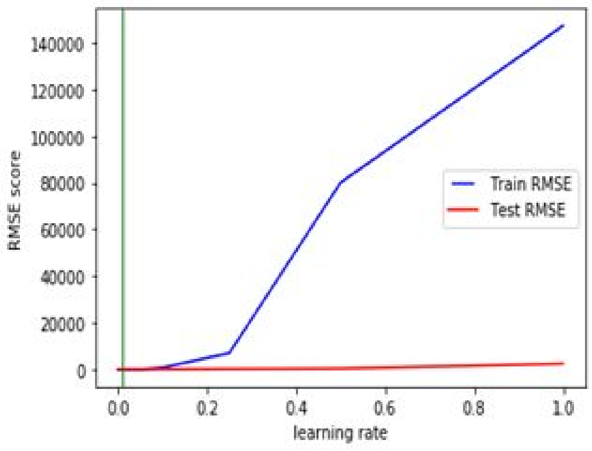


Fig 4.1 RMSE vs. learning rate

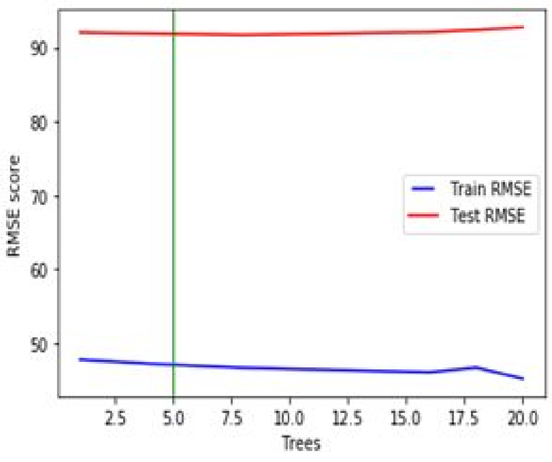
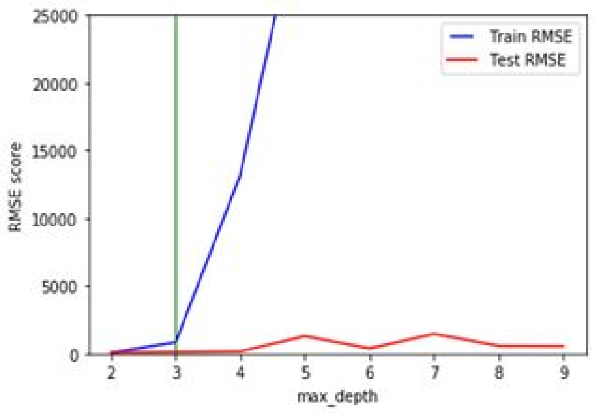


Fig 4.2 RMSE vs. number of trees

From figure 4.1 , it can be inferred that the learning rate should be kept small (ideally between 0 and 0.1) in order to make both the training error and test error at their respective minimum. XGBoost tree model does not perform as well as the linear model as there is no change in the test error which remains high despite the increase in the number of trees.(figure 4.2)



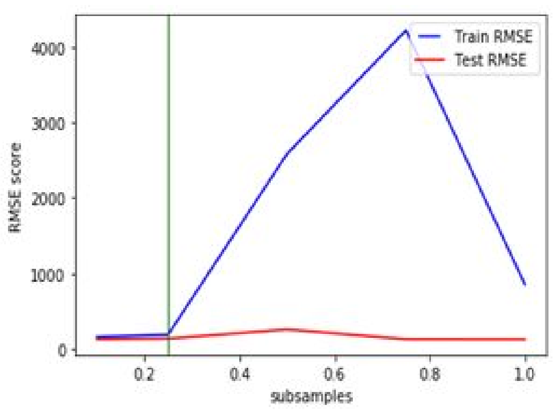


Fig 4.3 RMSE vs. maximum depth *Fig 4.4 RMSE vs. number of subsamples*

Like the learning rate, changes in both the maximum depth (figure 4.3) and the number of subsamples used for making the trees (figure 4.4) have negligible effect on the test error while the training error increases.

The table below shows the performance of the XGBoost tree based model before and after hyper-parameter tuning was done. It also shows the results of XGBoost linear function.

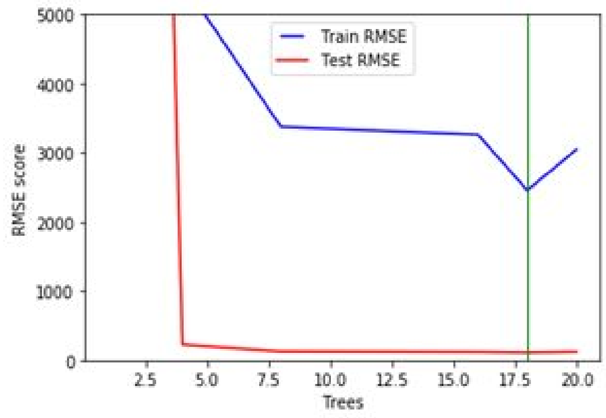
XGBoost Model	MSE	NLL	RMSE	MAD
Default XGBoost Package parameter Learning rate: 0.3 max_depth: 6 subsample: 1	2692631.1	331.7	609.54	130.4

Tuned hyper-parameters Learning rate: 0.001 Max_depth: 3 N_estimators: 5 Subsample: 0.25	4159.16	1178802022.5	46.6	13.0
XGBoost with linear function Booster: gblinear	4006.7	5085.1	46.4	18.6

XGBoost with linear function performs better than the tuned tree based models.

5.3 Random Forest

Similar to the XGBoost model, hyper-parameter tuning was done on random forest model to find the optimal parameters of the model. The main parameter under consideration for the tuning were maximum features, number of trees, minimum sample split and maximum depth of the trees. The images below shows the results and parameter chosen for the random forest model.



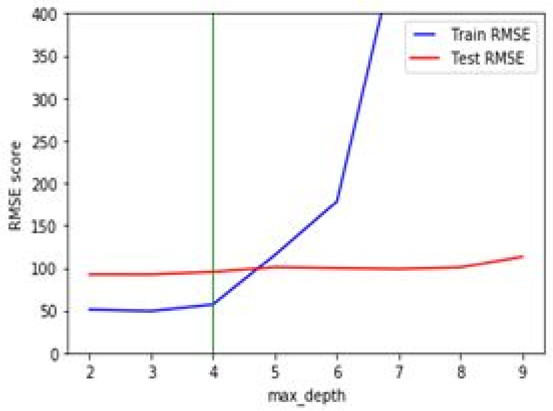
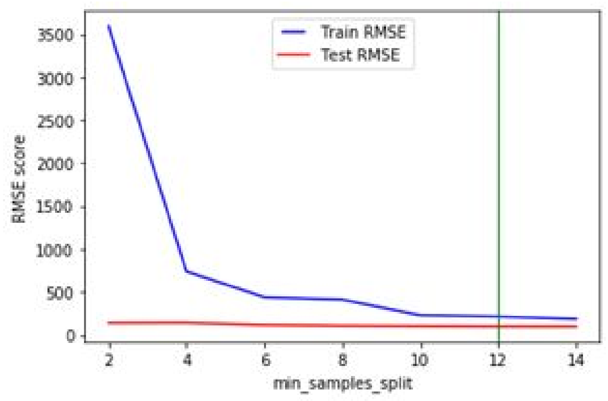


Fig 5.1 RMSE vs. number of trees

Fig 5.2 RMSE vs. maximum depth

Increasing the number of trees (figure 5.1) will train the data better, but it doesn't further minimise the error on the test set, which remains constant at around 8 trees. 18 trees minimises the errors on both the training set and the test set. Increasing the depth (figure 5.2) too much could overfit the training data, which is why after 4, the error on the training set increases. The test RMSE remains almost constant throughout.



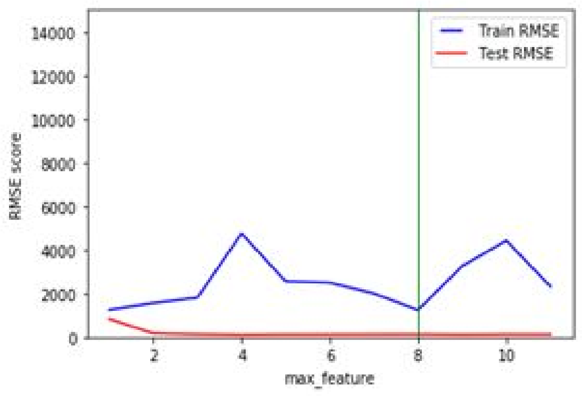


Fig 5.3 RMSE vs. min_samples_split   *Fig 5.4 RMSE vs. max_features*

Minimum number of samples to consider at each split (figure 5.3). As the value increases, the training error decreases, minimising at 12. The max_feature (figure 5.4) indicates the maximum number of features to be considered for each split.

The table below shows the performance of the random forest model before and after hyper-parameter tuning was done.

RF Model	MSE	NLL	RMSE	MAD
Default RF parameter N_estimators: 10 Min_sample_split: 2 Max_feature: √total features	341680.6	368.5	490.4	126.1
Tuned hyper-parameters N_estimators: 18 Max_features: 8 Min_sample_split: 12 Max_depth: 4	4557.2	675.3	54.8	23.2

Tuned random forest performs eight times much better than the non-tuned model while comparing the MSE.

5.4 Linear regression

Linear and non-linear models like polynomial regression of degree 2 were implemented. The results of the model is as follows:

Model	MSE	NLL	RMSE	MAD
Linear regression	4405.6	662.8	51.8	21.4
Polynomial regression (degree = 2)	1.7e+181	1470.0	1.3e+90	1.8e+89

Linear model performs better than non-linear model. Polynomial regression with degree greater than 2 goes out of boundary and tends to predict infinity for some cases.

Principal component analysis was performed on the model. The result below shows the performance of PCA on linear regression model.

PCA Variance	MSE	NLL	RMSE	MAD
1.0	4405.6	662.8	51.83	21.4
0.8	4397.3	1856.1	51.80	21.3
0.7	4428.9	4418.9	51.9	21.4

The linear model performs slightly better when PCA is done with overall variance of 80 percentage.

5.5 K-Nearest Neighbors (kNN)

KNN model was implemented with different number of neighbors and their corresponding error on the validation data was observed.

.The impact on the results of the KNN model as the number of neighbors increase is shown in the table

KNN Neighbors (N)	MSE	NLL	RMSE	MAD
50	4034.4	2802.5	46.7	18.9
100	4017.4	6408.9	46.4	18.6
150	4021.1	10008.5	46.2	18.1
200	4028.0	20688.0	46.2	18.0

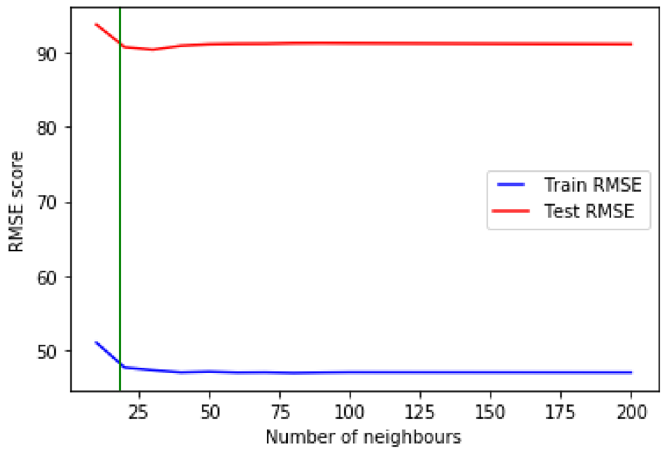


Fig 6 RMSE vs. number of neighbors

According to the KNN model results, MSE doesn't change much after 20 number of neighbors (according to the KNN plot shown above). Though not much difference in the results, the best observation according to MSE score was found to be KNN with 150 number of neighbors.

5.6 Model Comparison

Model	Parameters used	MSE	NLL	RMSE	MAD
SVM (with PCA)	Kernel: linear	4652.9	11569.6	53.2	16.5
Linear regression (with PCA)	_	4397.3	1856.1	51.80	21.3
KNN	Number of neighbors: 150	4021.1	10008.5	46.2	18.1
Random Forest	N_estimators: 18 Max_features: 8 Min_sample_split: 12 Max_depth: 4	4557.2	675.3	54.8	23.2
XGBoost	Booster: ‘gblinear’	4006.7	5085.1	46.4	18.6

6. CONCLUSION

Based on the results above, with hyper-parameter tuning, XGBoost gives the least mean squared error and the negative log likelihood, followed by kNN. The interdependencies among the features could have hindered the ability of the random forest regressor to find the best split, which must be why the model is performing poorly. The data does not seem to be nonlinear as SVM works best with the linear kernel, compared to polynomial and radial kernels. SVM also produces a decent NLL score,comparable to the other models.Taking the benchmark as 12.71±0.01 MAD (63.7±0.0 RMSE), the XGBoost model is able to minimise the RMSE further down to 46.4. The five models, with the optimum parameters, are compared with respect to the mean squared error, and are visually summarised below:

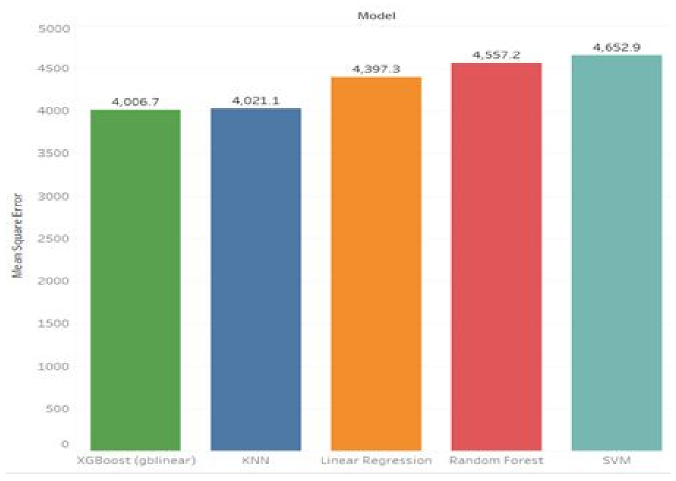


Fig 7. Mean squared error vs. model comparison

A major limitation of this dataset is that it is biased toward the occurrence of small forest fires. Even after the logarithmic transformation, outliers (large forest fires) have drastically increased the error margins, particularly for XGBoost regressor. More data points and related features should be collected and studied in the future to aid in resource planning of the authorities concerned and to set up proactive systems to reduce the extent of damage.

7. REFERENCES

1. Cortez P, Morais A. A Data Mining Approach to Predict Forest Fires using Meteorological Data. In Neves J, Santos MF and Machado J Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, December, Guimaraes, Portugal, pp. 512-523, 2007. APPIA, ISBN-13 978-989-95618-0-9.
2. Fraj MB. In Depth: Parameter tuning for Random Forest. [Internet] Medium. 2017 [cited 2019 Nov 4]; Available from: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-random-forest-d67bb7e920d
3. MathWorks. Understanding Support Vector Machine Regression. [Internet] 2019 [cited 2019 Nov 4]; Available from: https://au.mathworks.com/help/stats/understanding-support-vector-machine-regression.html
4. Rahman IU. Linear Regression: A Maximum Likelihood Approach. [online] Good Audience. 2019 [cited 2019 Nov 2]; Available from: https://blog.goodaudience.com/linear-regression-a-maximum-likelihood-approach-760263d8d395
5. Revert F. Fine-tuning XGBoost in Python like a boss. [online] Towards Data Science. 2018 [cited 2019 Nov 5]; Available from: https://towardsdatascience.com/fine-tuning-xgboost-in-python-like-a-boss-b4543ed8b1e