Forest Fire Burn Area Prediction Regression Problem

CSE 6470 Final Project

gdriskill3 Driskill

December 2023

1 Introduction

Code can be found at `https://github.com/gdriskill/CSE6740-Project`

1.1 Problem overview

Forest fires play a crucial role in the natural life cycle of a forest; however, large uncontrolled fires pose threats to human safety and the environment [1]. It would be useful to have a system to quickly and reliably detect these fires in order to contain them and prevent excessive destruction. The traditional solution of having a manned lookout is subject to human error and requires a person constantly on duty. Machine learning offers possible ways to automate the detection. The aim of this project is to predict the burn area for a specific location on a certain day. This regression model would be useful to help officials predict how severe a forest fire is going to be and what resources need to be allocated for it.

1.2 Dataset

The data set being used for this project was originally from a study by Cortez & Morais [2], and was found in the UCI Machine Learning Repository [3]. The data comes from a forest in northern Portugal, in Montesinho natural park. It is comprised of 12 input features, including geographic data, temporal data, and meteorological data, and the target variable, area burned. There are 517 entries that will be used to test and train regression models.

2 Related Work

Cortez & Morais 2007 also used this data set to do a regression problem, and focused on feature selection. Their results found that SVM using only weather data performed best [2]. Other work in the field of forest fire detection with machine learning often use satellite images as input. For example, Sathishkumar et. al. use a learning without forgetting deep learning model to detect fire or smoke in images of forest [4] .

3 Proposed Solution Approach

Three regression algorithms will be applied to this data set, linear regression, support vector regression (SVR), and artificial neural network. Then, their performances will be compared with mean squared error and the generalization gap.

3.1 Linear Regression [5]

Linear regression the most basic type of solution for regression problems, so it was chosen in order to have a baseline to compare the more complicated methods to.
Consider the class of linear hypothesises:

$$\mathcal{H} = \{x \to \mathbf{w} \cdot \mathbf{\Phi}(x) + b : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

Where $\mathbf{\Phi} : X \to \mathbb{R}^d$ is a feature mapping for the input space X. The goal of linear regression is to find a hypothesis $h \in \mathcal{H}$ that minimizes the empirical mean square error. For a sample of labelled training data $S = \{z_i = (x_i, y_i)\}, i \in [m]$ the problem is:

$$min_{\mathbf{w},b} \frac{1}{m} \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{\Phi}(x_i) + b - y_i)^2$$

Let $\mathbf{X}$ be a matrix with columns $\mathbf{X_i} = (\Phi(x_i), 1)$ for $i \in [m]$, $\mathbf{W}$ be a column vector $\mathbf{W} = [w_1, ..., w_d, b]$ and $\mathbf{Y}$ be a column vector $\mathbf{Y} = [y_1, ..., y_m]$. The problem is equivalent to:

$$\min_{\mathbf{W}} \frac{1}{m} ||\mathbf{X}^T\mathbf{W} - \mathbf{Y}||^2$$

The solution for linear regression is then:

$$\mathbf{W} = \begin{cases} (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{Y} & \text{if } \mathbf{X}\mathbf{X}^T \text{ is invertible} \\ (\mathbf{X}\mathbf{X}^T)^{\dagger}\mathbf{X}\mathbf{Y} & \text{otherwise} \end{cases}$$

3.2 Support Vector Regression [5] [6]

Support vector regression (SVR) is a regression model I saw in the textbook but was not covered in class, so I wanted to explore it in this project.

SVR builds off the ideas of support vector machines used for classification problems to fit a hyperplane in a high-dimensional space to the data, which creates a ϵ width tube where points inside the tube are not penalized and points outside the tube are (ϵ-insensitive loss). This type of regression is useful if we do not care about errors that are less than ϵ.

Consider the same hypothesis set $\mathcal{H}$ used above in linear regression but with the feature mapping Φ corresponds to a PDS kernel K $k(x, x') =< \Phi(x), \Phi(x') >$. The kernel trick allows for describing non-linear relationships. The optimization problem for SVR is:

$$\min_{w,b,\xi,\xi'} \frac{1}{2}||w||^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i')$$

$$\text{subject to} \begin{cases} y_i - (\Phi(x) \cdot w + b) \leq \epsilon + \xi_i \\ (\Phi(x) \cdot w + b) - y_i \leq \epsilon + \xi_i' \\ \xi_i, \xi_i' \geq 0 \end{cases}$$

3.3 Artificial Neural Networks [7]

Neural networks are currently generating a lot of interest in news and literature, so through this project I would like to further explore this topic and get experience with implementation. Artificial neural networks are another method that allows for a non-linear model for regression problems. In this project, a feed forward neural network, one where the structure is an acyclic directed graph, will be used.

The components of a feed forward neural network are an acyclic directed graph $G = (V, E)$, where V is a set of neurons and E is a set of edges connecting the neurons, a weight function $w : E \to \mathbb{R}$, and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$. The input to a neuron is a weighted sum of the outputs from neurons connected to it. The output of a neuron is the activation function applied to the input. The NN is organized in layers of neurons of depth T and width n, $V = \bigcup_{t=0}^{T} V_t$. The a neuron is connected to every to every neuron in the previous layer. Let $v_{t,i}, t \in [T], i \in [n]$ be the ith neuron in layer t, $o_{t,i}(\mathbf{x})$ and $a_{t,i}(\mathbf{x})$ be the output and input of $v_{t,i}$ when the NN has the input $\mathbf{x}$. The input and output for the neurons of the NN defined by $(V, E), w, \sigma$ are:

$$a_{t,i}(\mathbf{x}) = \sum_{r:(v_{t-1,r},v_{t,i})\in E} w(v_{t-1,r}, v_{t,i}) o_{t-1,r}(\mathbf{x})$$

$$o_{t,i}(\mathbf{x}) = \sigma(a_{t,i}(\mathbf{x}))$$

V_0 is the input layer, and has d neurons, where the inputs for the NN are $x \in \mathbb{R}^d$. V_T is the output layer, and in this project, will have one neuron that outputs the prediction for area burned.

The hypothesis class for a NN is $\mathcal{H}_{V,E,\sigma} = h_{V,E,\sigma,w} : (w : E \to \mathbb{R})$. The graph structure and activation function is fixed, and the learning problem is then finding the weights w that minimizes the loss. To learn the weights, the back propagation technique can be used, which involves multiple iterations with a forward pass where predicted values are calculated with the current weights, then a backward pass where the gradient of the error value are calculated, and finally the weights are updated.

3.4 Implementation

First, the data was preprocessed with a log transformation on the target data and scaling of all the data. The log transformation y = ln(x+1), which was suggest by Cortez & Morais, help counteract the heavily right skewed distribution. The data was scaled be in the range [0,1] because there was issues with value overflow when using the original data.

The Scikit-learn library was used create the linear regression and SVR. All the SVR kernels available in sklearn will be tested: linear, polynomial, rbf, and sigmoid.

PyTorch was used to create the ANN. A fully connected feed forward network with one hidden layer and a sigmoid activation function. This architecture

was used to match what is described in the universal approximation theorem. The UAT states that a neural network with one hidden layer and a sigmoidal function can approximate continuous function with arbitrary precision, if there are no constraints placed on the number of nodes or size of weights [8]. These assumptions are not met because constraints need to be put on the nodes and weights in order to implement the network. However, this architecture was still chosen for this project. The ANN was trained with stochastic gradient descent and MSE as the criterion.

Next, the hyper parameters for SVR, (C, epsilon, gamma), and ANN (hidden layer size, learning rate, epochs, batch size) were tuned using the grid search cross validation from Scikit-learn. The library Skorch was used to make the PyTorch NN compatible with Scikit-learn. Two rounds of the grid search were performed. The first round tested a wider range of values. The second, a more granular search was done on values closer to the result from the first round.

Finally, the MSE was calculated on the test and training data for each of the models. Scikit-learn was also used here.

4 Results and Analysis

Below is a summary of the results. Full results are in appendix.

	RMSE test	RMSE train	RMSE diff.
Linear regression	0.210	0.193	0.017
SVR - linear kernel	0.213	0.197	0.016
SVR - polynomial kernel	0.211	0.196	0.015
SVR - RBF kernel	0.211	0.196	0.015
SVR - sigmoid kernel	0.211	0.197	0.014
ANN	0.211	0.163	0.048

Linear Regression had best test error The linear regression model's test RMSE was lower than any other. One explanation for why basic linear regression could out perform the more sophisticated models is that the data naturally had a linear relationship. The r^2 statistic, or the coefficient of determination, was just 0.035. The r^2 is the proportion of variance in the dependent variable explained by the model, where values close to 1 indicate a good fitting model. The low r^2 suggests that the data does not actually have a linear relationship. Additionally, looking at the graphs comparing the individual features to area burned, it does not visually look like there are any strong linear relationships, leaving the reason for the linear regression's success unanswered (graphs are in appendix).

Although linear regression had the best test error, all of the test errors were within 0.003, or 1.4%. One could argue that all the models essentially performed the same. One explanation for this is that all the models were performing similarly poorly. Forest fires are notoriously hard to predict because they are part of a complex system and data on key factors, such as vegetation, was not available. Another explanation is that the SVR and ANN I implemented were

not as optimal as they could have been. There is possible room for improvement in hyper paramter tuning and the use of deep NN.

ANN Overfitting The results showed that the ANN had the largest difference between the test and training error, along with the lowest training error. NN are susceptible to overfitting, as seen in this project, primarily due to the model's complexity. This vulnerability is amplified, especially with smaller data sets becuase the model tends to memorize the training data. Although the data set in this project consisted of 517 samples, which is not necessarily small, it falls into the intermediate range. This might explain why the ANN exhibited the most substantial difference between test and training RMSE, coupled with having the lowest training error.

To mitigate the overfitting issue, regularization techniques such as weight decay could have been employed. Incorporating weight decay is another potential route for enhancing the performance of the ANN in this project.

SVR Generalization Overall, all the SVR models with the various kernels had the smallest differences between test and training RMSE, indicating that the SVR model generalizes well. There is the following is a theorem on the stability based generalization bound for SVR: Assuming that $K(x, x) \leq r^2$ for all $x \in X$ and that L_ϵ(the epsilon insensitive loss used in SVR) is bounded by $M \geq$. For any $\delta > 0$ the following holds with probability $\geq \delta$

$$R(h_S) \leq \hat{R_S}(h_S) + \frac{r^2}{m\lambda} + (\frac{2r^2}{\lambda} + M)\sqrt{\frac{log(1/\delta)}{2m}}$$

This bound comes from the fact of SVR being a β stable algorithm. The proof for both of these can be found in Mohri et. al. This stability and associated generalization bound may contribute to the SVR's good generalization performance.

5 Conclusion

In conclusion, this project applied and compared multiple regression algorithms to predict forest fire burn areas, revealing unexpected success with a simple linear regression. The linear regression model resulted in the lowest test error, while the SVR models resulted in the best generalization. Future work could involve further hyper parameter tuning and exploration of deep neural network architectures to enhance SVR and ANN performance. Additionally, implementing regularization techniques may address overfitting concerns in the ANN. The SVR stability-based bound provides a theoretical foundation for its good generalization, warranting further investigation.

References

[1] "Wildfires: How They Form, and Why They're So Dangerous". In: *National Geographic Society* (2023).

[2] A. Cortez P. Morais. "A Data Mining Approach to Predict Forest Fires using Meteorologicl Data". http://www3.dsi.uminho.pt/pcortez/fires.pdf. 2007.

[3] *UCI Machine Learning Repository*. https://archive.ics.uci.edu/dataset/162/forest+fires.

[4] V.E. Sathishkumar, J. Cho, and M. Subramanian. "Forest fire and smoke detection using deep learning-based learning without forgetting". In: *Fire Ecology* (2023). DOI: https://doi.org/10.1186/s42408-022-00165-0.

[5] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2018.

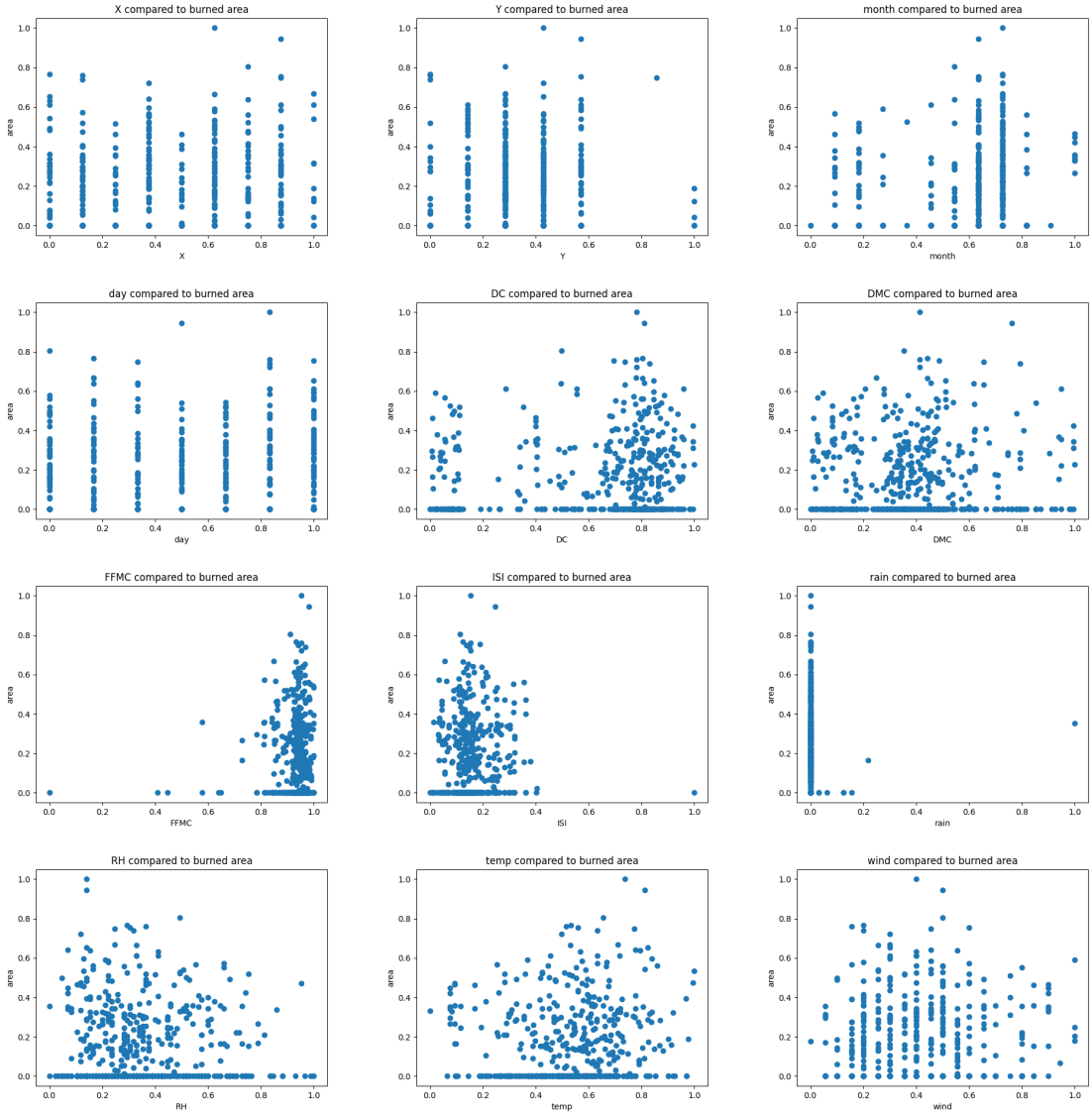
[6] B Smola A. J. Scholkopf. "A tutorial on support vector regression". In: *Statisics and Computing* (2004). DOI: https://doi.org/10.1023/b:stco.0000035301.49549.88.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2008.

[8] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* (1989). DOI: https://doi.org/10.1007/bf0255127.

6 Appendix

6.1 Graphs comparing individual features to area burned



6.2 Linear Regression Results

—-Linear regression—
Test MSE 0.04399206158213341
Train MSE 0.037262378891467864

6.3 SVR 1st Grid Search

——SVR: linear kernel——
Paramters for grid search:
{ 'C': [0.01, 0.1, 1, 10, 100],
  'epsilon': [0.0004728987782239752,
              0.004728987782239752,
              0.04728987782239752,
              0.4728987782239752]}

Best Parameters: {'C': 1, 'epsilon': 0.04728987782239752}

Test MSE: 0.05186879170114345
Train MSE 0.04439820894257932

——SVR: poly kernel——
Parameters for grid search:
{'C': [0.01, 0.1, 1, 10, 100,
  'degree': [2, 3, 4],
  'epsilon': [0.0004728987782239752,
              0.004728987782239752,
              0.04728987782239752,
              0.4728987782239752]}

Best Parameters: {'C': 0.1, 'degree': 2, 'epsilon': 0.04728987782239752}

Test MSE: 0.050523288135232554
Train MSE 0.043608182785543174

——SVR: rbf kernel——
Parameters for grid search:
{'C': [0.01, 0.1, 1, 10, 100],
  'epsilon': [0.0004728987782239752,
              0.004728987782239752,
              0.04728987782239752,
              0.4728987782239752],
  'gamma': ['auto', 'scale']}

Best Parameters: {'C': 10, 'epsilon': 0.04728987782239752, 'gamma': 'auto'}

Test MSE: 0.05046289208299632
Train MSE 0.043707627664038864

——SVR: sigmoid kernel——

Parameters for grid search:
{'C': [0.01, 0.1, 1, 10, 100],
  'epsilon': [0.0004728987782239752,
              0.004728987782239752,
              0.04728987782239752,
              0.4728987782239752],
  'gamma': ['auto', 'scale']}

Best Parameters: {'C': 1, 'epsilon': 0.04728987782239752, 'gamma': 'auto'}

Test MSE: 0.05234198561482971
Train MSE 0.04481329458044758

6.4 SVR 2nd Grid Search

—SVR: linear kernel—
Parameters for grid search:
{'C': [0.3, 0.6, 1, 2.25, 4.5, 6.75], 'epsilon': [0.10403773120927454,
              0.21280445020078884,
              0.3168421814100634,
              0.04728987782239752,
              0.14186963346719256,
              0.2837392669343851]}

Best Parameters: {'C': 0.3, 'epsilon': 0.14186963346719256}

Test MSE: 0.045306358775313686
Train MSE 0.03886889294371659

—SVR: poly kernel—
Parameters for grid search:
{'C': [0.03, 0.06, 1, 0.225, 0.45, 0.675],
  'degree': [2],
  'epsilon': [0.10403773120927454,
              0.21280445020078884,
              0.3168421814100634,
              0.04728987782239752,
              0.14186963346719256,
              0.2837392669343851]}

Best Parameters: 'C': 0.06, 'degree': 2, 'epsilon': 0.14186963346719256

Test MSE: 0.04447896309775245
Train MSE 0.03852075166332333

—-SVR: rbf kernel—-
Parameters for grid search:
{'C': [3, 6, 10, 22.5, 45, 67.5],
  'epsilon': [0.10403773120927454,
              0.21280445020078884,
              0.3168421814100634,
              0.04728987782239752,
              0.14186963346719256,
              0.2837392669343851],
  'gamma': ['auto']}

Best Parameters: {'C': 3, 'epsilon': 0.14186963346719256, 'gamma': 'auto'}

Test MSE: 0.044366771804593946
Train MSE 0.03851312080834996

—-SVR: sigmoid kernel—-
Parameters for grid search:
{'C': [0.3, 0.6, 1, 2.25, 4.5, 6.75],
  'epsilon': [0.10403773120927454,
              0.21280445020078884,
              0.3168421814100634,
              0.04728987782239752,
              0.14186963346719256,
              0.2837392669343851],
  'gamma': ['auto']}

Best Parameters: {'C': 2.25, 'epsilon': 0.14186963346719256, 'gamma': 'auto'}

Test MSE: 0.04466636117667241
Train MSE 0.03880634751225186

6.5 ANN 1st Grid Search

—-ANN—-
Parameters for grid search:
'batch_size': [10, 20, 30, 40], {'batch_size': [10, 20, 30, 40],
  'max_epochs': [50, 100, 150],
  'module__hidden_size': [6, 12, 25],
  'optimizer__lr': [0.001, 0.01, 0.1, 0.2]}

Best Parameters: {'batch_size': 40, 'max_epochs': 50, 'module__hidden_size': 6, 'optimizer__lr': 0.001}

Test MSE: 0.04624458
Train MSE 0.03658771049231291

6.6 ANN 2nd Grid Search

—-ANN—-
Paramters for grid search:
{'batch_size': [35, 40, 45, 50],
  'max_epochs': [40, 45, 50],
  'module__hidden_size': [4, 5, 6],
  'optimizer__lr': [0.0001, 0.005, 0.001]}

Best Parameters: {'batch_size': 50, 'max_epochs': 40, 'module__hidden_size': 6, 'optimizer__lr': 0.001}

Test MSE: 0.044606548
Train MSE 0.026674195053055884