Machine Learning Assignment 2 - Group: 4

Tutor: Iwan Budiman

Regression task performed on Forest Fires Dataset: A Latex report.

Anshu Kumar (490517666)
Sanna Nazir (490517677)
Samarth Sehgal (490528857)

November 2019

Contents

1 ABSTRACT **4**

2 INTRODUCTION **4**

2.1 Aim (Goals and motivation) 4

2.2 Dataset overview 5

2.2.1 Dataset Attributes 5

2.2.2 Visualize data in 2/3D (using PCA) 5

2.2.3 Display samples 6

3 PREVIOUS WORK **6**

4 METHODS **7**

4.1 Data Pre-processing techniques 7

4.1.1 Standardization 7

4.1.2 Encoding months and days 7

4.1.3 Changing the target variable 7

4.1.4 Feature selection 9

4.2 Regression algorithms 10

4.2.1 Linear Regression 10

4.2.2 Support Vector Regression 11

4.2.3 K-Nearest Regression 11

4.2.4 AdaBoost Regression 11

4.2.5 Decision Trees Regression 11

4.2.6 Random Forest Regression 12

4.2.7 Neural Network Regression (Using Adam Optimization) 12

5 EXPERIMENTATION **13**

5.1 Experiment setting 13

5.1.1 Outline 13

5.1.2 Performance Metrics 13

5.1.3 Experiment steps 14

5.2 Experiment Observations 14

5.2.1 Linear Regression 14

5.2.2 Support Vector Regression 14

5.2.3 K-Nearest Regression 15

5.2.4 AdaBoost Regression 15

5.2.5 Decision Trees Regression 15

5.2.6 Random Forest Regression 15

5.2.7 Neural Network Regression (Using Adam Optimization) 15

5.3 Hyper-parameter Tuning 16

6 DISCUSSION **16**

6.1 Discussion of Results 16

6.2 Final choice of algorithm and parameters 17

6.3 Personal Reflection 17

7 CONCLUSION **17**

8 BIBLIOGRAPHY **19**

9 APPENDIX **20**

9.1 Hardware and Software specifications 20

9.2 Contributions 20

9.3 Instructions for running the code 20

1 ABSTRACT

Globally, forest fires are on the rise. According to a 2017 Climate Science Special Report [1], forest fires have significantly increased in the recent decades. A significant task when dealing with such disasters is the identification of vulnerable areas and prediction of area of impact. For the purposes of this report, a regression task is being performed to predict the area affected by a forest fire given certain environmental conditions and geographic metrics. To train the regression model, the Forest Fires Data Set by the UCI Machine Learning repository has been used. The report explores the efficiency of various regression models on the dataset and analyzes the performance of each. The work is concluded with a reflection on the significance of the experiment and potential future work.

2 INTRODUCTION

The purpose of this report is to evaluate various regressors for the prediction of forest fire affected area. Each regressor is evaluated on industry standard metrics such as Root Mean Square Error (RMSE) and Negative Log Likelihood (NLL) of the predicted results versus the actual test results. The data used for performing this task is the data collected by Cortez and Morais [4] from the Montesinho Natural Park of Portugal. The same is publicly available on the UCI Machine Learning Repository.

2.1 Aim (Goals and motivation)

The aim and motivation behind this project may be divided into two major categories.

At the student level, the project is aimed at inculcating technical and analytical skills within the student. Each step of the project, starting with the choice of a problem statement to discussing and analyzing the best solutions for that problem, the project is designed to educate the student about various techniques available to deal with the same. The core focus being development of essential data analysis skills as well as associated business and analytical acumen.

At a macroscopic level, the project is aimed at understanding the significance of technological solutions to real-world problems. In case of the forest fire dataset, this problem is the creation of robust, efficient and effective techniques for controlling such disasters.

Global warming is a scientifically proven and physically experienced phenomenon of the 21st century. In the future, under a warmer climate, it is expected that there will be more severe fire weather, more area burned, more ignitions and a longer fire season [7]. To better understand allocation of resources for such instances, predicting the affected land is very important. To predict the same, environmental as well as geographic features are of paramount importance. Thus, with the exploration of this study, such a task can be achieved.

Microscopically, the key tasks of this project are: Selection of the dataset. Here, chosen dataset is Forest Fires Dataset. Perform experiments to analyze the performance of various regression techniques on the same to predict the burned/affected area due to a fire, given environmental and geographical parameters.

2.2 Dataset overview

The data used for performing this task is the data collected by Cortez and Morais [4] from the Montesinho Natural Park of Portugal. The same is publicly available on the UCI Machine Learning Repository.

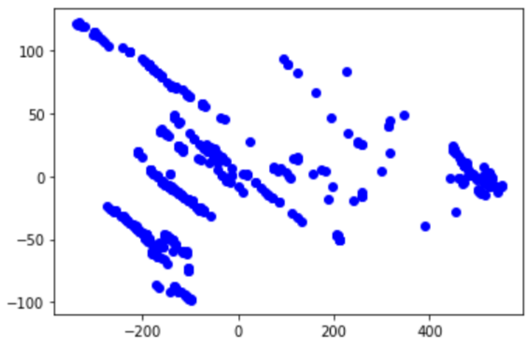
2.2.1 Dataset Attributes

The attributes of the dataset can be listed and described as under:

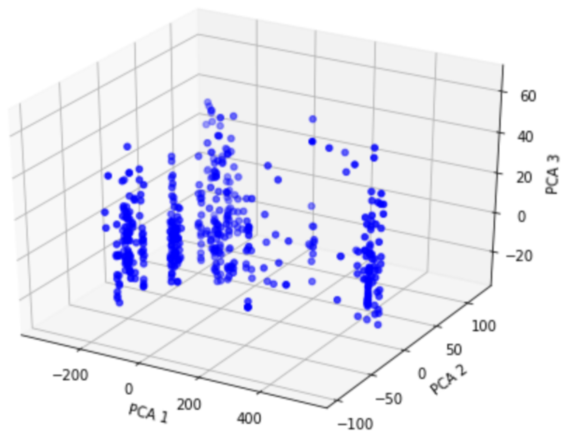
Attribute Name	Attribute Description
X	x-axis co-ordinate with reference to the park area
Y	y-axis co-ordinate with reference to the park area
Month	month of the year in which the fire took place
Day	Day of the week
FFMC	Fine Fuel Moisture Code(denotes the moisture content surface litter) [4]
DMC	Duff Moisture Code(moisture content of shallow organic layers)[4]
DC	Drought Code(moisture content of deep organic layers)[4]
ISI	Initial Spread Index(score that correlates with fire velocity spread)[4]
Temp	Temperature of the area (in Celsius)
RH	Relative Humidity within the area (in percentage)
Wind	Wind speed in the area (in km/hr)
Rain	Rainfall in the area (in mm)
Area	Total burnt area (in ha)

Table 1: Attribute Descriptions

2.2.2 Visualize data in 2/3D (using PCA)



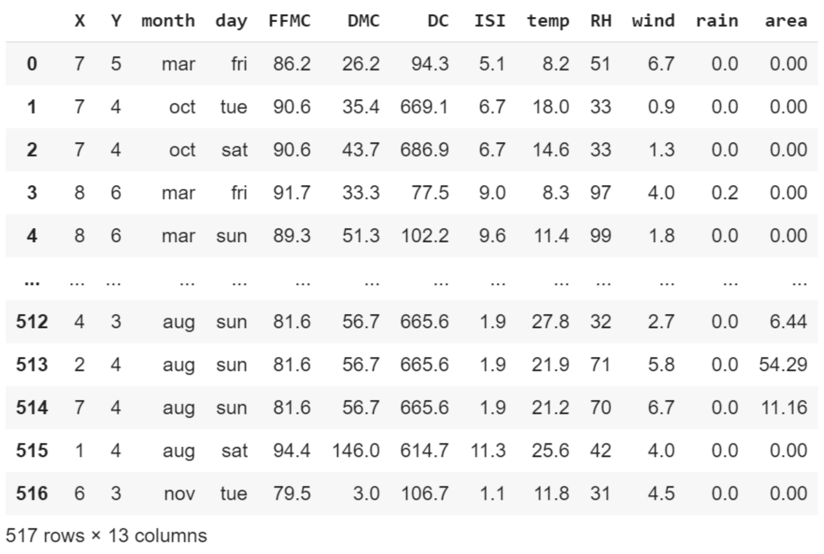
(a) 2D



(b) 3D

Figure 1: Data visualization in two and three dimensions

2.2.3 Display samples



	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.00
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.00
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.00
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.00
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...
512	4	3	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44
513	2	4	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	54.29
514	7	4	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	11.16
515	1	4	aug	sat	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	0.00
516	6	3	nov	tue	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00

517 rows × 13 columns

Figure 2: Glimpse of the dataset

3 PREVIOUS WORK

Cortez and Morais [4] studied the effects of different factors on the forest fires in their research paper “A Data Mining Approach to Predict Forest Fires using Meteorological Data.” They have used different machine learning algorithms like Support Vector Machines (SVMs) and Random Forest to predict data. The have also used distinct feature selection setups in order to get the best results. Benchmark results of 63.71 Root Mean Square Error (RMSE) and 12.71 Mean Absolute Deviation (MAD) were obtained using the SVM technique combined with four features, namely temperature, relative humidity, rain and wind. A conclusion was drawn proposing the use of a Data Mining approach that in turn uses the meteorological data to detect forest fires. Castelli, Vanneschi and Popovič also tried to predict the burned areas of forest fires using artificial intelligence in their paper [3]. They used different algorithms and pre-processing techniques, including SVM, random forest and Neural Networks. While the best result of 12.9 MAE was achieved using Genetic Programming algorithm, the Neural Network was able to achieve only an MAE of 33.8 on test set. A similar approach can also be found in Babu, Swetha, Charitha and Stephen’s paper [10] where in they used SGD, Decision Tree, Naïve Bayes, SVM and Random Forest, achieving a maximum accuracy of 62.82 % using decision trees.

Our project tries to incorporate different machine learning (ML) and deep learning (DL) algorithms in a quest to achieve better results. We have used different pre-processing techniques that have never been used to pre-process this data before. We have also used different feature selection methods in order to train our model on the features that really matter and save both computational time and overfitting. Furthermore, GridsearchCV is utilized for hyper-parameter estimation. These methods, combined with different ML and DL algorithms, give a unique approach to the forest fire problem that we are trying to solve.

4 METHODS

4.1 Data Pre-processing techniques

"Pre-processing data is an essential step to enhance data efficiency [2]". In lay man terms, data pre-processing is the extraction of valuable data in an understandable format from raw data. Real world data is often biased, raw, incomplete and inconsistent, thus resulting in poorly performing machine learning (ML) and deep learning (DL) models. Often, the complexity and the computation time of the ML or DL algorithm is increased due to the presence of correlated features found in unprocessed data. To mitigate such issues, data pre-processing is done, producing cleaner data, resulting in better, faster and more accurate results.

Some of the pre-processing techniques used in our project are explained below:

4.1.1 Standardization

Standardization is a feature scaling technique used to confine values within a specific range so as ensure that data does not flow out of bounds or tend to zero. Various methods of standardization can be used such as Min-Max scaler, Max Absolute scaler, Z-score Normalizer etc.

Though all these methods are equally good, we have used Z-score Normalizer for our project. This is because it is one of the most popular scaling methods. Since our data varies from zero to large values, scaling it using Z-score Normalizer seemed like the right choice. The method makes use of the mean and variance of each feature in the dataset.

Put mathematically, a z-score is defined as:

$$z = \frac{x_i - x_{mean}}{\sigma}$$

where:
x_i is a data point
x_{mean} is the mean of the data points across a feature
σ is the variance of that feature.

4.1.2 Encoding months and days

Real life data may include string type data, like months written in words (January, February, etc). Such values need to be converted into a type understandable by our regressor. To accomplish this, we have mapped the words to numbers. For example, January is converted to 1, February to 2, and so on. We used this method in our project because we are applying regressor models and we wanted to maintain consistency in our data so as to achieve best results.

4.1.3 Changing the target variable

The target variable for this regression task i.e. area, has a highly skewed distribution (See figure 4a). While small fires cover small areas, large fires can cover hundreds of hectares of land. Thus, creating a large range of values for the area variable. Such skewed distribution can cause bad regression results. To overcome such a problem, we employed the logarithmic conversion technique on the target variable (See figure 4*b*)

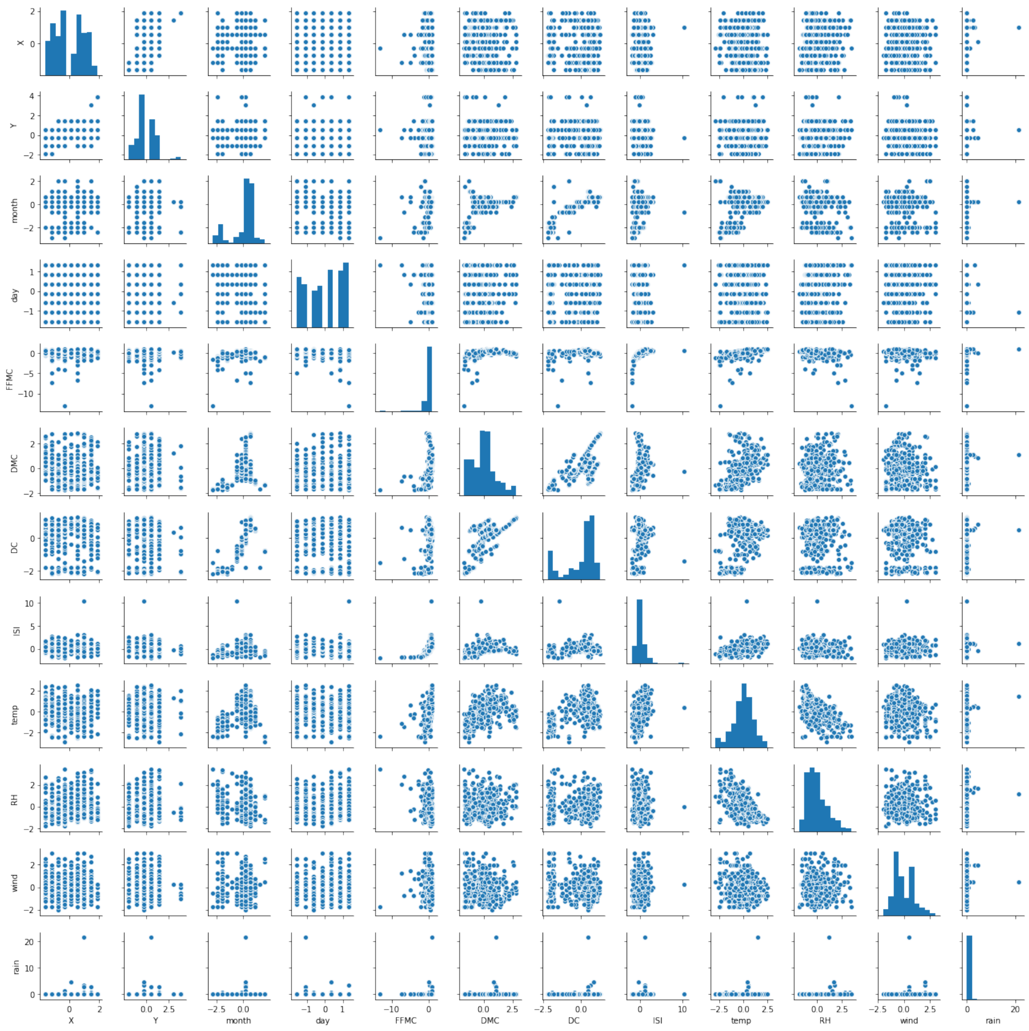
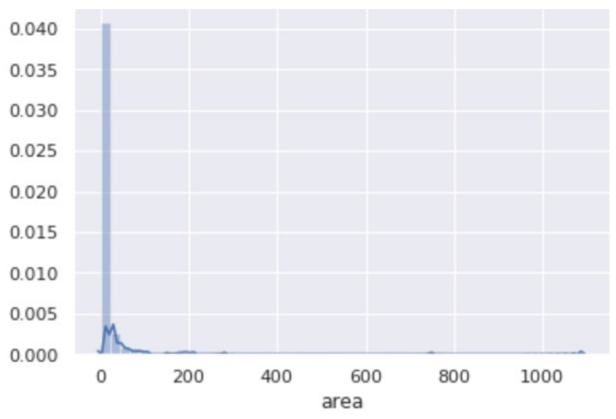
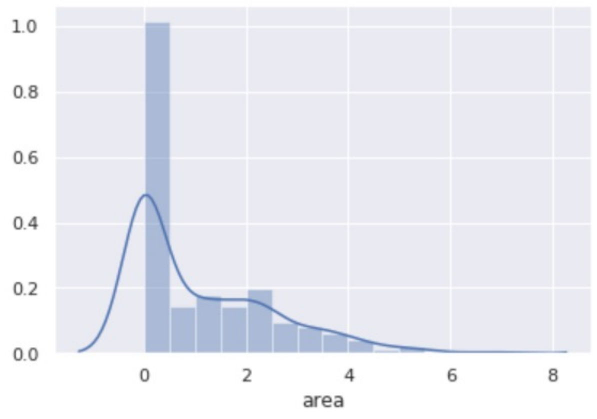


Figure 3: Visualizing data distribution after standardization using Seaborn



(a) Before log conversion



(b) After log conversion

Figure 4: Pre-processing of target variable

4.1.4 Feature selection

1. **ExtraTrees Classifier:**
The ExtraTrees Classifier, or the Extremely Randomized Trees Classifier, is a type of ensemble learning technique similar to a Random Forest Classifier. However, the difference is that in ExtraTrees Classifier the manner of construction of decision trees is different. In this technique, the results of all the different co-related decision trees are collected together in what is called a "forest", and the classification result is outputted. In order to perform feature selection using this technique, a mathematical value called "Gini Importance" is calculated for each feature during the construction of the tree. Each feature is then sorted according to its Gini importance and the top k features are selected.

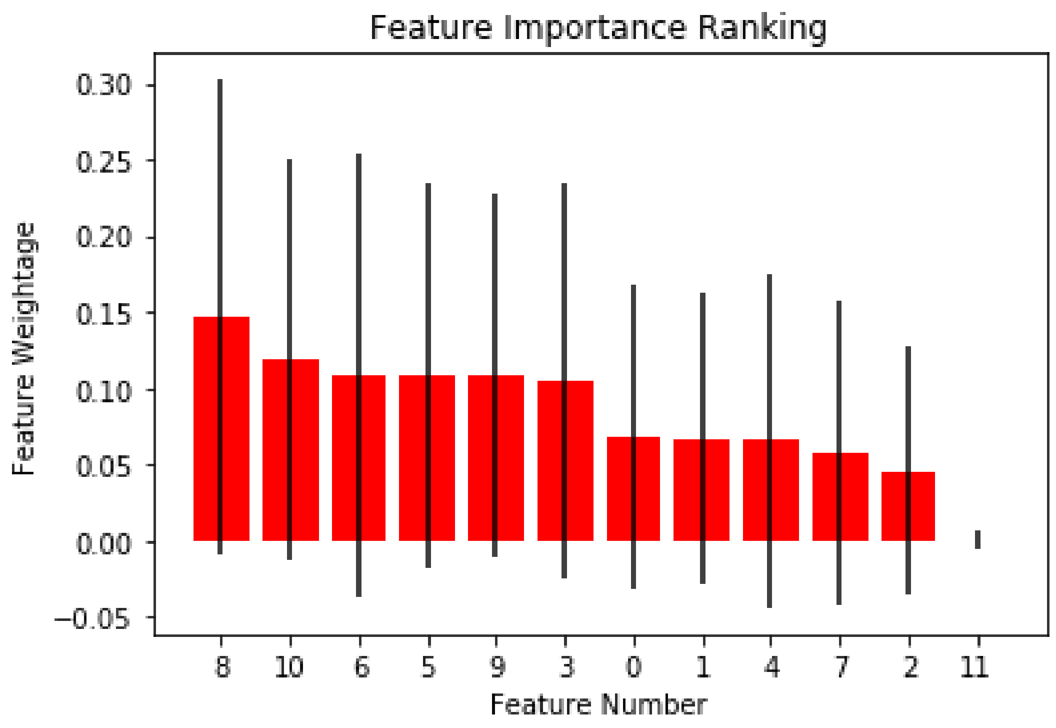


Figure 5: Feature ranking generated by ExtraTrees Classifier

We used this technique for feature importance in our program because due to its high amount of randomization (unlike random forests), we were able to select the best features suitable for our algorithms, reducing our calculation times by almost 1.6x times.

2. **Principal Component Analysis:**
Principal Component Analysis, or PCA, is a very popular and reliable dimension reduction technique used for large datasets. "The description of Principal Component Analysis is made by means of the explanation of eigenvalues and eigenvectors of a matrix" [3]. In this technique a data matrix of the dataset is created and the eigen values and eigen vectors are calculated using the covariance matrix. The spread of the data along its principal components is then understood using these eigen vector pairs, and only the features that give significant contribution to the spread are chosen. The PCA technique is based on the creation of a covariance

matrix and consequent eigen value decomposition of the same:

Covariance matrix of a matrix C with n rows and n columns may be defined as:

$$C^{n\times n} = (c_{i,j}, c_{i,j} = cov(Dim_i, Dim_j))\ [5]$$

We used this technique in our project because PCA is one of the most used and reliable technique for data pre-processing and has proven impact on algorithm improvement.

4.2 Regression algorithms

Regression algorithms are used to define a mathematical relationship between an independent/target variable and one or many dependent/predictor variables. Essentially, the key task is predicting the value of a given continuous target variable based on input features.

For the purpose of our project, we have chosen a wide variety of regression algorithms to cover variety of scenarios within the dataset. Reasons for the same may be discussed as under:

- Linear Regression is the classical choice for regression problems and thus, was included in the experiment process. It is chosen for its simplicity, ease of understanding, ubiquity and wide scientific acceptance.
- Support Vector Regression technique was employed for its ease of generalizing capabilities and high prediction accuracy.
- K-Nearest Regression technique is employed for its novelty, usability for non-linear data and relatively good accuracy.
- Decision Trees Regressor Method was used because of its comprehensive nature in tracing each outcome to its end. Thus, considering all possible outcomes for the data.
- Random Forest technique was employed to understand the impact of ensemble algorithms on our dataset. It's known advantages of being reducing variance and over-fitting (such as in decision trees) were also consider while choosing the same.
- Adaboost algorithm was chosen to understand the difference of a boosting technique on the dataset (as compared to the bagging technique of Random Forest). Furthermore, Adaboost considers the combination of features by itself, something which would be manual in techniques like Linear Regression.
- Neural Networks were chosen for their ability to detect all possible interactions between the variables with least statistical pre-processing. We also wanted to analyze the success of a neural network over a regression task.

4.2.1 Linear Regression

Linear regression technique maps a simple linear equation between the attributes and the target variables. Each attribute is assigned a weight and the same is employed to predict the value of a new data point. This model can be represented mathematically, as simply as:

$$Y = mx + c$$

4.2.2 Support Vector Regression

Support Vector Machines are popularly used for classification tasks. They create hyperplanes in high dimensional space separating the various classes. However, they can also be used for regression tasks by slight modification of the algorithm. This method is known as the Support Vector Regression.

This conversion to a regression algorithm is achieved by means of a loss function defined as:

$$Loss = \begin{cases} 0 & \text{if } \|y - f(x)\| \leq \varepsilon \\ \|y - f(x)\| - \varepsilon & \text{otherwise} \end{cases}$$

This defines an ε tube so that if the predicted value is within the tube the loss is zero, while if the predicted point is outside the tube, the loss is the magnitude of the difference between the predicted value and the radius ε of the tube.[6]

4.2.3 K-Nearest Regression

K-Nearest Neighbours is a popular technique for classification tasks. However, the same can be used for regression tasks as well. This is achieved by storing all possible classes/cases during the task and then, using some similarity technique to calculate the continuous prediction variable.

Similarity techniques used can be of multiple times. Some of the popular one's are Euclidean distances, Hamming distances, Manhattan distances etc. For the purpose of this project, we have employed the most popular Euclidean measurement technique.

$$D = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

4.2.4 AdaBoost Regression

AdaBoost algorithm was first introduced as a boosting technique for other regression algorithms [8]. As per scikit-learn library, AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction [11]. One of the major advantages of using such an algorithm is that it is easier to train that neural networks while being non-parametric at the same time.

The loss function for this algorithm is a cumulative function calculated across the left and right splits of the tree. Mathematically, it may be represented as (where, 'L' is for left and 'R' is for right):

$$Total\ Squared\ Error = \sum_{i:x \leq x_s} (y_i - \bar{y_L})^2 + \sum_{i:x > x_s} (y_i - \bar{y_R})^2 \text{ [5]}$$

4.2.5 Decision Trees Regression

Decision Trees Regression employs the tree branching technique for regression. A tree structure is created with decision nodes and leaf nodes. The decision nodes represent attributes while the leaf nodes represent final values for the attributes tested. The first node where branching occurs is called the root node and signifies the most important attribute for decision making.

To describe the homogeneity of the attributes divided within a branch of the tree, the 'Standard Deviation' formula is used. Mathematically, it may be represented as:

$$S = \sqrt{\frac{\sum(x-\bar{x})^2}{n}}$$

4.2.6 Random Forest Regression

The essence of the Random Forest technique is to build multiple trees in randomly selected sub spaces of the feature space [9]. Trees in, different sub-spaces generalize their classification in complementary ways, and their combined classification can be monotonically improved [9]. Simply put, the Random forest technique is an ensemble algorithm that combines the outputs of various different trees (each trained on a sub-sample of the training sample/dataset) by means of averaging to give a final output. This result is less prone to over-fitting.

To calculate the split at each node, the usual Mean Squared Error method is used. Mathematically, it may depicted as:

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

4.2.7 Neural Network Regression (Using Adam Optimization)

For the purpose of this project, we have designed a four layer neural network. The activation function used at each level is the "ReLu" Activation function. For optimizing the network, we have used the 'Adam Optimizer.'

ReLu is an abbreviation for 'Rectified Linear Unit.' It is one of the most popular activation functions used in neural networks these days. This is because of its simplicity, ease of computation and efficiency. Mathematically, it can be depicted as:

$$y = max(0, x)$$

The 'Adam' optimizer is an iterative optimization technique for deep neural networks that is an extension of the classical stochastic gradient descent technique. It was first introduced in 2015 and has gained wider acceptance owing to its straightforwardness, less memory requirements and computational efficiency.

NOTE: Since the dataset used in this project is very small, accuracy issues were being faced. To overcome this issue, splitting of the data into test and train sets was not done (only specifically for this regressor). Instead, k-fold cross validation was used to judge the parameters of the regressor

5 EXPERIMENTATION

5.1 Experiment setting

5.1.1 Outline

For this regression task, dataset used is the 'Forest Fires' Dataset publicly available on the UCI Machine Learning repository [4]. The target variable is the 'area' attribute within the original dataset. Area is being measured in hectares.

5.1.2 Performance Metrics

The two key performance metrics used to measure the performance of each regression algorithm are:

1. **Root Mean Square Error (RMSE) :**
   Root mean square error is defined as the deviation of the predicted values from the actual values. Lesser the RMSE value, better the algorithm. **For the purpose of the project, RMSE benchmark has been defined at 63.7 (by the instructor).**
   Mathematically, Root Mean Square Error is defined as the square root of the MSE (Mean Squared Error) where MSE may be defined as:

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

   Thus, root mean squared error may be defined as:

   $$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y_i})^2}$$

2. **Negative Log Likelihood (NLL) :**
   NLL is the negative of Log Likelihood. Minimizing the NLL is same as maximizing the log likelihood. The NLL function produces a high value when the value of output layer are evenly distributed and low. For minimizing the NLL, the output layer should match the expected values. No specific benchmarks have been set for the purpose of this project.

   Mathematically, it is given as:

   $$L(b_0, b_1, s^2) = -\frac{n}{2}\log 2\pi - n\log s - \frac{1}{2s^2}\sum_{i=1}^{n}(y_i - (b_0 + b_1 \times x_i))^2$$

3. **Mean Absolute Deviation (MAD) :**
   Mean Absolute Deviation is a method to define the variation in a dataset. It is calculated by the distance between the mean of a dataset with each corresponding data point. Greater the MAD, more spread out the data is. Lesser the MAD, more clustered the data is. **For the purpose of our project, MAD benchmark has been defined at 12.71 (by the instructor).**
   Mathematically, it is defined as:

   $$MAD = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y_i})$$

5.1.3 Experiment steps

The steps performed within our project are as follows:

1. Cleaning of the dataset including changing required variables.
2. Selection of features from the original dataset. This has been divided into three techniques:
   (a) Training models with all features i.e. no feature selection done.
   (b) Feature Selection done by ExtraTrees Classification Method. *(Number of features selected is dependent on the value of feature importance generated by the method. For our project, only features that have a high importance i.e. value above 0.1 are taken. This results in selection of top 6 features.)*
   (c) Feature Selection done by Principal Component Analysis Method. *(Number of features selected depends on the information retention percentage. For our project, we have chosen to retain 90% of the information from total features. This results in selection of first 8 features)*
3. Choose the final algorithm and feature selection method.
4. Perform hyper-parameter tuning on the same to analyze the impact of such changes.

5.2 Experiment Observations

Following the above experiment steps, observations are recorded as below. These observations have been recorded after the parameter tuning of each algorithm and selection of the respectively best parameters.

5.2.1 Linear Regression

Feature Selection Technique	RMSE	NLL	MAD
Without Selection	67.73	1657.57	13.71
By ExtraTrees Classifier	67.73	1657.57	13.71
By Principal Component Analysis	67.73	1657.57	13.71

Table 2: Linear Regression Observations

5.2.2 Support Vector Regression

Feature Selection Technique	RMSE	NLL	MAD
Without Selection	68.95	443944.08	0.92
By ExtraTrees Classifier	68.95	252976.94	0.92
By Principal Component Analysis	68.93	252976.94	0.86

Table 3: Support Vector Regression Observations

5.2.3 K-Nearest Regression

Feature Selection Technique	RMSE	NLL	MAD
Without Selection	86.01	963.36	22.99
By ExtraTrees Classifier	86.04	963.31	22.99
By Principal Component Analysis	86.01	963.36	22.99

Table 4: K-Nearest Regression Observations

5.2.4 AdaBoost Regression

Feature Selection Technique	RMSE	NLL	MAD
Without Selection	70.89	1797.79	12.01
By ExtraTrees Classifier	70.89	1797.79	12.01
By Principal Component Analysis	65.16	2272	9.63

Table 5: AdaBoost Regression Observations

5.2.5 Decision Trees Regression

Feature Selection Technique	RMSE	NLL	MAD
Without Selection	71.56	1596.94	11.19
By ExtraTrees Classifier	71.56	1596.94	11.19
By Principal Component Analysis	71.56	1596.94	11.19

Table 6: Decision Trees Regression Observations

5.2.6 Random Forest Regression

Feature Selection Technique	RMSE	NLL	MAD
Without Selection	70.55	1266.87	9.84
By ExtraTrees Classifier	70.55	1266.87	9.84
By Principal Component Analysis	70.55	1266.87	9.84

Table 7: Random Forest Regression Observations

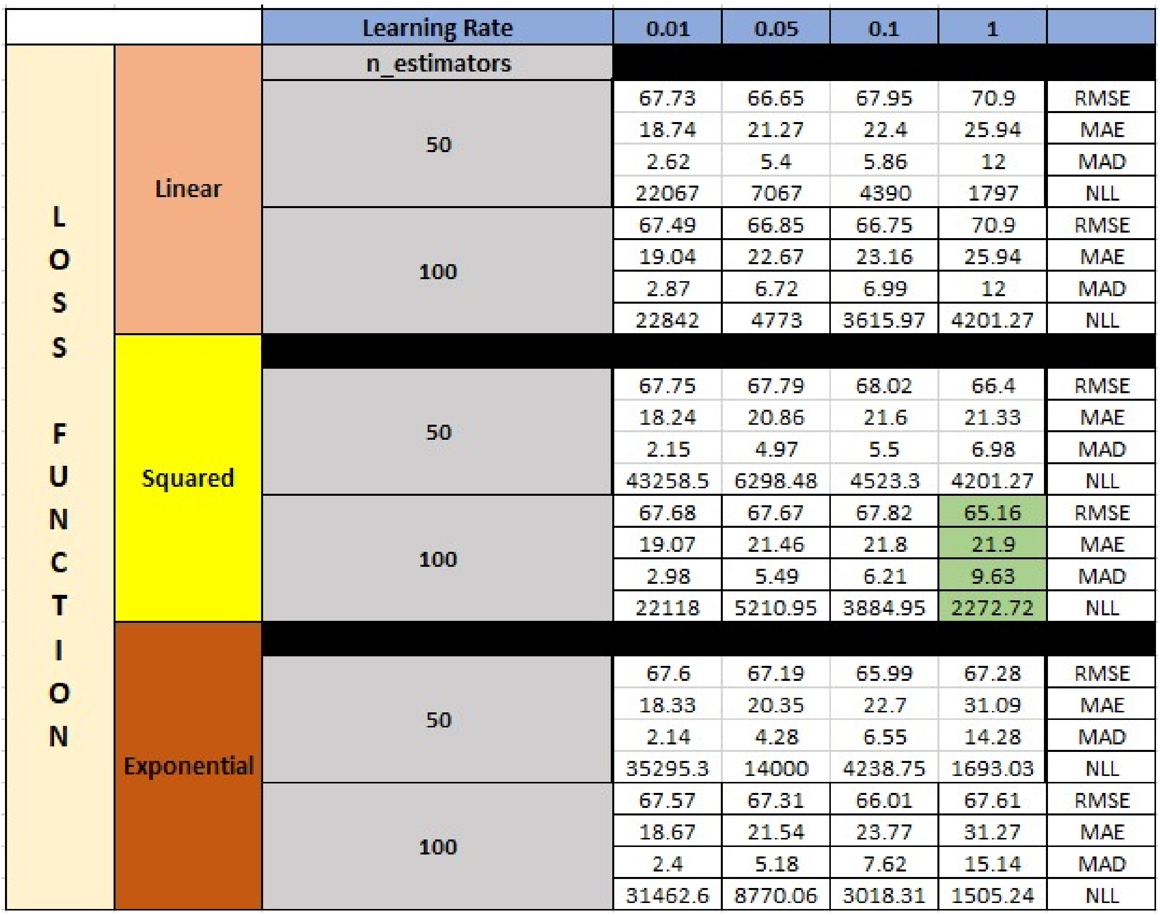
5.2.7 Neural Network Regression (Using Adam Optimization)

Feature Selection Technique	RMSE	NLL	MAD
Without Selection	89.18	11249.72	8.36
By ExtraTrees Classifier	89.18	11249.72	8.36
By Principal Component Analysis	89.18	11249.72	8.36

Table 8: Neural Network Regression Observations

5.3 Hyper-parameter Tuning

Based on the above observations, we have chosen the 'AdaBoost Regressor' with 'Principal Component Analysis' for feature selection as our final algorithm (detailed discussion as to why in next section). While hyper-parameter tuning was done for all algorithms, space constraints have led us to discussing only the best performing algorithm's tuning. For the AdaBoost regressor, we have performed the hyper-parameter tuning. The observations of the same are recorded as below:



		Learning Rate	0.01	0.05	0.1	1	
LOSS FUNCTION	Linear	n_estimators					
		50	67.73	66.65	67.95	70.9	RMSE
			18.74	21.27	22.4	25.94	MAE
			2.62	5.4	5.86	12	MAD
			22067	7067	4390	1797	NLL
		100	67.49	66.85	66.75	70.9	RMSE
			19.04	22.67	23.16	25.94	MAE
			2.87	6.72	6.99	12	MAD
			22842	4773	3615.97	4201.27	NLL
	Squared	50	67.75	67.79	68.02	66.4	RMSE
			18.24	20.86	21.6	21.33	MAE
			2.15	4.97	5.5	6.98	MAD
			43258.5	6298.48	4523.3	4201.27	NLL
		100	67.68	67.67	67.82	65.16	RMSE
			19.07	21.46	21.8	21.9	MAE
			2.98	5.49	6.21	9.63	MAD
			22118	5210.95	3884.95	2272.72	NLL
	Exponential	50	67.6	67.19	65.99	67.28	RMSE
			18.33	20.35	22.7	31.09	MAE
			2.14	4.28	6.55	14.28	MAD
			35295.3	14000	4238.75	1693.03	NLL
		100	67.57	67.31	66.01	67.61	RMSE
			18.67	21.54	23.77	31.27	MAE
			2.4	5.18	7.62	15.14	MAD
			31462.6	8770.06	3018.31	1505.24	NLL

Figure 6: Hyper-parameter Tuning of AdaBoost Algorithm.

As can be seen in the above observations, AdaBoost Regressor performs best for a 'squared' loss function with learning rate 1 and n_estimators 100. The increased performance can be attributed to variations in dataset.

6 DISCUSSION

6.1 Discussion of Results

According to the observations above, the best performing regressors are the Linear regressor, Support Vector Regressor, Random Forest Regressor and the AdaBoost Regressor, based only on the RMSE values of the same. The worst performing based on the RMSE values are the Neural Network Regressor and the K-Nearest Regressor. On further inspection using the MAD and NLL metrics for performance measurement, the best performing algorithms come out to be Linear Regressor and the AdaBoost Regressor.

In terms of pre-processing techniques, the only impact created is in running time of the algorithms. With such a small dataset, feature extraction does not result in major differences created in the performance metrics. As can be seen in the observations.

6.2 Final choice of algorithm and parameters

The final choice of algorithm was based on the novelty of the approach, performance across metrics as well as the scope of future enhancement of the algorithm. While the best performing algorithm was of course, linear regression, we did not choose the same for its classical existence, rigidity in innovation, proneness to over-fitting as well as lack of tuning of algorithms.

We choose AdaBoost regressor with PCA as our final algorithm because it gives performance comparable to the industry benchmarks. It is a novel bagging technique that is non-parametric like neural networks, but not as complex. It takes less training time and gives effectively high results. Furthermore, by tuning the hyper-parameters of the same, we were able to achieve results sometimes higher than even linear regression. PCA was chosen as the pre-processing technique because it reduces the dimensionality and thus, computational time for the algorithm. While the current dataset is too small to experience the impact of the same, such a dimensionality reduction technique is proven to be highly effective for larger datasets.

6.3 Personal Reflection

The results obtained from the implementation of this project may be divided into two parts, qualitative and quantitative.

Quantitatively, the implementation of a wide variety of regression algorithms on this dataset has given us the skills required for a data scientist. It has allowed us to understand the grass-root working of various algorithms, their advantages and dis-advantages as well as accepting the trade-offs while selecting a certain algorithm over the other. Further, the opportunity to understand the previous work done in the field enlightened our minds towards understanding the significance of the problem, the existing solutions for the same and how we can create our own unique solution while pushing our intellectual capabilities.

Qualitatively, by analyzing the impact of forest fires, we have understood the degree of severity of this problem and the need for a resolution. We can also empathize with the growing problem of forest fires in the world owing to global warming. By executing this exercise, better resource allocation systems can be designed for such disasters saving lives as well as money.

7 CONCLUSION

After extensive experimentation and analysis, we conclude that for the purpose of our project, the AdaBoost Algorithm with pre-processing is the best choice. As seen within the observations, the RMSE for this algorithm ($\tilde{6}5$) is close to the industry benchmarks ($\tilde{6}3$). Furthermore, the other measurement metrics i.e. NLL and MAD are also significantly better for the AdaBoost algorithm than the other algorithms tested. In terms of the industry benchmark for MAD ($\tilde{1}2$), AdaBoost algorithm is giving almost exact same MAD value.

As described before, hyper-parameter tuning of the AdaBoost algorithm is enhancing the performance of the same. With the best case scenario providing results better than linear regression. By employing some data cleaning and modification techniques, we believe this performance metric can be further improved and may exceed the industry benchmark as well.

Reflecting on the future work for such a project, we believe extensive data collection is required. The current dataset available for the problem statement is very small (517×13) for the purposes of a data mining approach. While some amount of accuracy and efficiency is achieved using the current dataset, it cannot be enhanced significantly without the introduction of more data for the training algorithm. Furthermore, various methods for cleaning or modification of the dataset can also be employed. One method that may be employed is the binning of related X and Y co-ordinates. By creating 3x3 grids within the 9x9 original grid of the park area, identification of affected area can be done in a better manner. It would also remove the co-related parameters of individual X and Y and eliminate the doubt of algorithm considering the co-ordinates as continuous variables (instead of categorical). Another method that can be employed is the conversion of the continuous rain attribute to a Boolean attribute (rain or no rain). Similarly, binning of months into seasons could also be done. Thus, we propose collection of further data as well as cleaning and modification of the existing data as a further step to this project.

8 BIBLIOGRAPHY

References

[1] *Climate science special report: fourth National Climate Assessment.* U.S. Global Change Research Program, 2017.

[2] Suad A Alasadi and Wesam S Bhaya. Review of data preprocessing techniques in data mining. *Journal of Engineering and Applied Sciences*, 12(16):4102–4107, 2017.

[3] Mauro Castelli, Leonardo Vanneschi, and Aleš Popovič. Predicting burned areas of forest fires: an artificial intelligence approach. *Fire ecology*, 11(1):106–118, 2015.

[4] Paulo Cortez and Aníbal de Jesus Raimundo Morais. A data mining approach to predict forest fires using meteorological data. 2007.

[5] Harris Drucker. Improving regressors using boosting techniques. In *ICML*, volume 97, pages 107–115, 1997.

[6] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[7] M. D. Flannigan, B. D. Amiro, K. A. Logan, B. J. Stocks, and B. M. Wotton. Forest fires and climate change in the 21st century. *Mitigation and Adaptation Strategies for Global Change*, 11(4):847–859, Jul 2006.

[8] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[9] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[10] T Niranjan, D Swetha, V Charitha, and AJ Stephen. Predicting burned area of forest fires. *IRJCS:: International Research Journal of Computer Science*, 6:132–136, 2019.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

9 APPENDIX

9.1 Hardware and Software specifications

The code was created using Google Colabratory Open Source Computing Platform. However, we used a laptop for running the same online, which has below features:

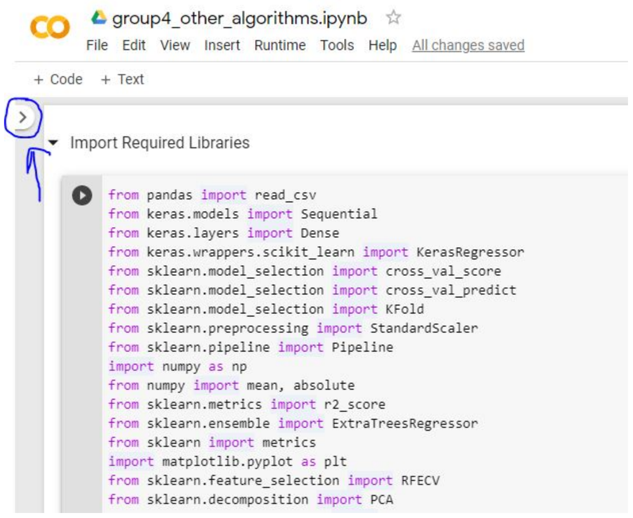
- Processor: Intel i7
- OS: Windows 10
- RAM: 8 GB
- GPU: 4 GB
- Model: HP Pavilion 360

9.2 Contributions

Each of the members contributed to the project equally. Two algorithms each were designed by each member and division of the report content was done on the same lines.

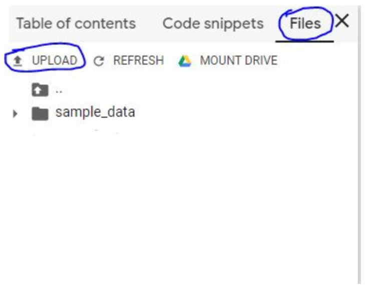
9.3 Instructions for running the code

1. Open the .ipynb notebook in Google Colab.
2. Upload the given files using the steps below
   - Click on side arrow


group4_other_algorithms.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Import Required Libraries

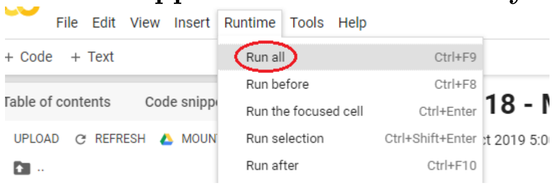
```
from pandas import read_csv
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
import numpy as np
from numpy import mean, absolute
from sklearn.metrics import r2_score
from sklearn.ensemble import ExtraTreesRegressor
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.feature_selection import RFECV
from sklearn.decomposition import PCA
```

   - On the side bar, click on Files option and click on upload

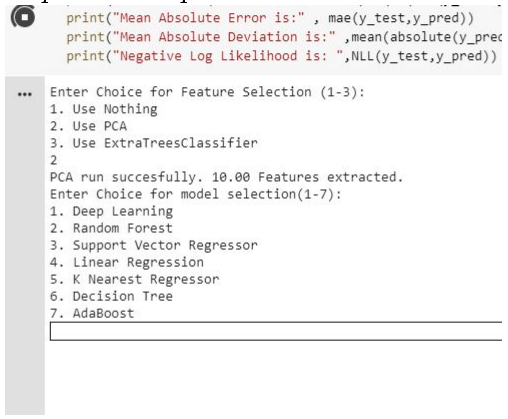


- Select the files to be uploaded and press ok.

3. Now from the menu above, click on "Runtime" and then click "Run All". Alternatively, you can press Ctrl+F9 (or Ctrl+Fn+F9) on your keyboard. **Please Note, it is necessary to run all snippets of code on each try to ensure no error is encountered.**


File Edit View Insert Runtime Tools Help
+ Code + Text
Run all Ctrl+F9
Run before Ctrl+F8
Run the focused cell Ctrl+Enter
Run selection Ctrl+Shift+Enter
Run after Ctrl+F10
Table of contents
UPLOAD REFRESH

4. The program will ask for a user input. Enter required input. For the best algorithm file, this step is not required.



```
print("Mean Absolute Error is:" , mae(y_test,y_pred))
print("Mean Absolute Deviation is:" ,mean(absolute(y_prec
print("Negative Log Likelihood is: ",NLL(y_test,y_pred))
```

```
Enter Choice for Feature Selection (1-3):
1. Use Nothing
2. Use PCA
3. Use ExtraTreesClassifier
2
PCA run succesfully. 10.00 Features extracted.
Enter Choice for model selection(1-7):
1. Deep Learning
2. Random Forest
3. Support Vector Regressor
4. Linear Regression
5. K Nearest Regressor
6. Decision Tree
7. AdaBoost
```

5. Wait for a few minutes for the algorithm to run and an output file will be generated displaying the required performance metrics for the code.