VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A PROJECT REPORT **(15CSP85)** ON

"PREDICTING FOREST FIRES WITH SPARK MACHINE LEARNING"

Submitted in Partial fulfillment of the Requirements for the Degree of

Bachelor of Engineering in Computer Science & Engineering

By

HARITA A (1CR16CS055)

K MADHUREKAA (1CR16CS063)

KP NEETHU (1CR16CS064)

KAJOL R SINGH (1CR16CS065)

Under the Guidance of,

Dr. Pushpa Mohan

Assoc. Prof, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled **"PREDICTING FOREST FIRES WITH SPARK MACHINE LEARNING"** carried out by **Ms. HARITA A**, 1CR16CS055, **Ms. K MADHUREKAA**, 1CR16CS063, **Ms. KP NEETHU**, 1CR16CS064, **Ms. KAJOL R SINGH**, 1CR16CS065, bonafide students of CMR Institute of Technology, in partial fulfillment for the award of Bachelor **of Engineering** in Computer Science and Engineering of the Visveswaraiah Technological University, Belgaum during the year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Dr. Pushpa Mohan	**Dr. Prem Kumar Ramesh**	**Dr. Sanjay Jain**
Assoc. Prof	**Professor & Head**	**Principal**
Dept. of CSE, CMRIT	**Dept. of CSE, CMRIT**	**CMRIT**

External Viva

Name of the examiners	Signature with date
1.	________________
2.	________________

DECLARATION

We, the students of Computer Science and Engineering, CMR Institute of Technology, Bangalore declare that the work entitled "**PREDICTING FOREST FIRES WITH SPARK MACHINE LEARNING**" has been successfully completed under the guidance of Prof. Dr. Pushpa Mohan, Computer Science and Engineering Department, CMR Institute of technology, Bangalore. This dissertation work is submitted in partial fulfillment of the requirements for the award of Degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2019 - 2020. Further the matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place: Bangalore

Date: 5th May 2020

Team members:

HARITA A (1CR16CS055) __________________

K MADHUREKAA (1CR16CS063) __________________

KP NEETHU (1CR16CS064) __________________

KAJOL R SINGH (1CR16CS065) __________________

ABSTRACT

Forest fires have become one of the most frequently occurring disasters in recent years. The effects of forest fires have a lasting impact on the environment as it leads to deforestation and global warming, which is also one of its major causes of occurrence. Forest fires are dealt by collecting the satellite images of forest and if there is any emergency caused by the fires then the authorities are notified to mitigate its effects. By the time the authorities get to know about it, the fires would have already caused a lot of damage. Fast detection and accurate estimation of forest fire burned areas can help us to effectively control damage. Based on the previous literature survey, SVM is the most efficient machine learning algorithm for predicting small forest fires at better accuracy which eventually lead to bigger fires. Due to high computation and high storage requirements the training time required is more, hence to improve the speed of computation parallel computing is required. Our model uses Parallel SVM to train the meteorological data. Forest weather indexes along with basic weather parameters have to be considered to predict the occurrence of forest fires. Our findings indicate that Parallel SVM gives higher accuracy and less computational time as compared to Linear SVM.

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing me a platform to pursue my studies and carry out my final year project

I have great pleasure in expressing my deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore, for his constant encouragement.

I would like to thank **Dr. Prem Kumar Ramesh,** HOD, Department of Computer Science and Engineering, CMRIT, Bangalore, who has been a constant support and encouragement throughout the course of this project.

I consider it a privilege and honor to express my sincere gratitude to my guide
Dr. Pushpa Mohan, Assoc. Prof, Department of Computer Science and Engineering, for the valuable guidance throughout the tenure of this review.

I also extend my thanks to all the faculty of Computer Science and Engineering who directly or indirectly encouraged me.

Finally, I would like to thank my parents and friends for all their moral support they have given me during the completion of this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviations	**Full Forms**
SVM	Support Vector Machine
FFMC	Fine Fuel Moisture Code
DMC	Duff Moisture Code
DC	Drought Code
FWI	Forest Fire Weather Index
PCA	Principal Computation Analysis
KNN	K- Nearest Neighbors
MPI	Message Passing Interface
PSO	Particle swarm optimization
CNN	Cascade correlation network
MPNN	Multilayer perceptron neural network
PNN	Polynomial neural networks

CHAPTER 1

INTRODUCTION

Forest fires have become one of the most frequently occurring disasters in recent times which cause destruction to large acres of forests. One of the main reasons behind the occurrence of forest fires is global warming which is one of the major causes of increase in average temperature of the earth. The other reasons are due to lightning, thunderstorms, and human negligence. Forest fires can lead to deforestation which has a lot of negative impact on human society. It is reported that for the last decade, each year, more than 100,000 in all countries. Early detection of forest fires is very important in fighting against fires. Spread features of forest fires show that, in order to put out a fire without causing any permanent damage in the forest, the fire fighter center should be aware of the threat at most 6 min after the start of the fire.

Forest fires prediction combines weather factors, terrain, and dryness of flammable items, types of flammable items, and ignition sources to analyze and predict the combustion risks of flammable items in the forest. There are many techniques that the authorities use to detect forest fires, satellite detection is widespread among worldwide authorities. Few forest authorities use human observers as detectors and reporters of forest fires.

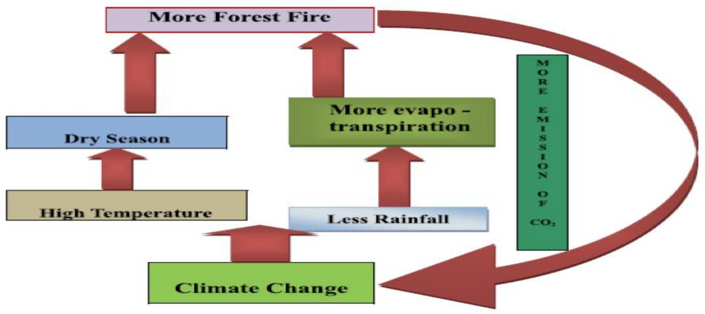


Fig 1.1: Data Flow Diagram

Fig 1.1 displays a data flow map for forest fires and climate change worldwide. In this article, we compare the various methods used in related research. In each of the methods used, there are different advantages and disadvantages. Data mining and machine-learning techniques may provide an effective mitigation strategy, where forest-related data can be used to predict forest fires. Owing to changes in earth's orbital parameters, solar intensity, and atmospheric composition, the environment and associated weather are not continuous.

Our atmosphere has warmed up in recent years due to changes in human behavior in the air in radioactively active gases (carbon dioxide, methane, etc.). Such warming is likely to have a profound impact on fire activity in the forest zone.

Based on weather altering factors, we restrict our research to forest fires. From recently on, it appears that they greatly impact the frequency of wildfires. By analyzing these causes, we are developing a stronger and more efficient way of predicting forest fires. The IMD (Indian Meteorological Department) dataset is split into training data and test data. The training data were used in the supervised learning model. This model will now forecast forest fires on the basis of the input test results. For testing various test data to forecast the occurrence of forest fires a user interface is created.

There are many Machine Learning models used to predict forest fires and alerts the forest department. Some of the techniques used to detect these fires are logistic regression, random forest, K-Nearest Neighbor, Support Vector Machine, Artificial Neural Networks, Gradient Boosting, Bagging, etc. There are various advantages and disadvantages in each of the methods used. From our literature survey we concluded that SVM is the most suitable technique which can be used. SVM is not only a good classification algorithm but also a good regression algorithm. SVM gives accurate results in case of small fires which eventually lead to large fires. Therefore SVM is the most efficient method for forest fire prediction.

SVM for large datasets produce a large number of support vectors which increase the computation time and storage requirements. Therefore SVM is implemented using parallel computing, in our case parallel SVM. Splitting the dataset into sub datasets and using subSVMs to evaluate each dataset to filter the support vectors from non-support vectors so that the last global sub-SVM will have only one set of support vector which is easy to classify and because we use many sub-SVMs simultaneously, computation speed is increased and accuracy of prediction also is increased.

1.1 Relevance of the Project

Data mining and machine learning techniques can provide an efficient prevention approach where data associated with forests can be used for predicting the occurrence of forest fires. Fire activity is strongly influenced by four factors – weather/climate, fuels, ignition agents and human activities. Climate and the associated weather are not constant due to changes in the earth's orbital parameters, solar output and atmospheric composition.

Recently, our climate has been warming as a result of increases of radioactively active gases (carbon dioxide, methane etc) in the atmosphere caused by human activities. Such warming is likely to have a profound impact on fire activity in the forest zone.

We are limiting our study to predict forest fires based on weather altering factors. As of recent, they seem to be affecting the occurrence of wildfires greatly; we aim to devise a more accurate and faster method to predict forest fires by studying these factors.

1.2 Problem Statement

To predict the occurrence of forest fire using Parallel Support vector machine algorithm to provide accurate and faster results by calculating forest weather index for basic weather parameters which is an important factor to determine the intensity of the wildfire.

1.3 Objectives

- To create a machine learning model
- It is used to predict forest fires
- It uses meteorological data
- It uses support vector machine (SVM) technique for better accuracy for small forest fires
- It uses parallel computing (parallel SVM) for high speed computation
- To create a user interface for predicting forest fires and display an alert message to authorities like the Ministry of Environment, Forest, Climate Change (MOEFCC).

1.4 Scope of the Project

The goal of this project is to predict forest fires using machine learning algorithms. This project uses SVM to provide better accuracy. It also uses parallel computing for high speed computation.

The dataset provided by IMD (Indian Meteorological Department) will be divided into training data and test data. The training data is fed into the model for supervised learning. This model should now predict forest fires based on the test data given as input. A user interface is developed for testing different test data to predict the occurrence of forest fires.

1.5 Methodology

Parallel Computing

- Traditional computation instructions in a queue pass through the processor and processor does the processing and get the processed data
- The problem with this type of approach is we can only make the processor faster to limit i.e., there is a frequency limit, heat.
- To increase the speed of processing we use two processors which divide the instructions among them.

- Two processors running in parallel dividing the instructions among them (workload distributed)

Support Vector Machine

- It is a supervised learning algorithm for classifying data
- Uses hyperplane to separate data to segments where each segment has one kind of data
- Classification based on features of the data
- SVM is used for regressions as well
- SVM kernel function for classifying non linear data
- Plot the data as nodes on n dimensional space and draw hyperplane
- Closest points to the hyperplane are called support vectors and have max distance from the hyperplane

Parallel SVM

- SVM is taken as a most efficient classification and regression model.
- Classical SVM model is difficult to analyze large scale practical problems.
- Parallel SVM can improve the computation speed greatly.
- In our project we plan on handling large datasets using Apache Spark and use parallel SVM for more accuracy with better computation speed.

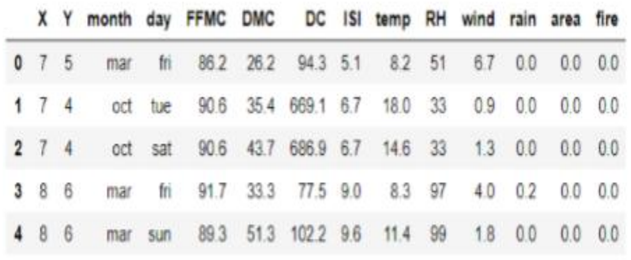
CHAPTER 2

LITERATURE SURVEY

Forest fires are managed by weather data collection or by the processing of satellite forest photos. Rapid detection can help us control damage effectively. Earlier work has shown that forest fires can be caused for many causes and it is therefore very important to accurately make predictions. Many machine learning techniques were applied and compared in order to construct a prediction model. The SVM model provides reliable forecasts for the use of the environment in small forest fires, which inevitably lead to larger fires. When forecasting forest fires using weather information, the forest weather predictor along with basic weather parameters must be taken into account. We show that through parallel computation with vector support engines the accuracy of this predictive model can be enhanced.

2.1 Supervised and Ensemble Machine learning algorithm

To analyse the data in datasets and how each feature affects the forest fires and some of the techniques used for solving regression and classification problems used in this study are discussed. The dataset includes: X and Y axes special park coordinates, Fine Fuel Moisture Code (FFMC), Duff Moisture Code (DMC), Drought Code (DC) and Initial Spread Index (ISI).The other characteristics used are temperature, relative humidity, wind speed, rain outside and the burnt forest region. All these features were collected from January to December on all days for a whole year.



	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	fire
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0	0.0

Fig 2.1: Sample Dataset

Fig 2.1[1] represents a sample (first 10 rows) of the UCI forest fire dataset. In this paper, there are two graphs plotted, variance of feature with respect to fire and variance of feature with respect to each month. The variance calculated for rain is 0.09 which is close to 0; therefore it does not affect the model. From the dataset, we inferred that, the number of forest fires that happened is 270 and not happened is 247. A bar graph is plotted showing the frequency of occurrence of forest fire with respect to each month. It was observed that forest fires were frequent in the month of August and September and less frequent in May and April and in the other months it is sporadic. The normalization techniques used are Principal computing analysis (PCA), Feature Scaling (Min Max Scaling) and label encoding. The ML techniques used are Logistic regression, Bagging, SVM, KNN, Boosting and Random Forest.

When PCA was applied, Logistic regression had the highest classification rate of 68%. When PCA was not applied, Gradient Boosting had the highest classification rate of 68%. SVM is best for small fires which lead to larger fires.

2.2 Soft computing approaches

To compare the various algorithms under Artificial Neural Networks and use Soft Computing Techniques to find the best algorithm for forest fire prediction. The aim of this paper is to examine the quality of five SC techniques to determine the best and most effective forest fire predictor. These methods include cascade correlation network (CCN), multilayer perceptron neural network (MPNN), polynomial neural network (PNN), radial basis function (RBF) and support vector machine (SVM). The data collected from the UCI machine learning database, which took 517 different entries for montesinho natural park (MNP) at different times.

Detection of forest fires requires a variety of sensors distributed over a wide area to be deployed. Pre-processing the data set is the first stage. For this purpose, the principal component analysis (PCA) and particle swarm optimization (PSO) techniques were used to find the critical features and to segment the fire regions (clusters) respectively. Then, five predictors were applied to identify the best and suitable predictor that has the ability to predict forest fires.

The ANN methods were: Principal component analysis (PCA), Particle swarm optimization (PSO), Cascade correlation network (CCN), Multilayer perceptron neural network (MPNN), Polynomial neural networks (PNNs), Support vector machine (SVM).

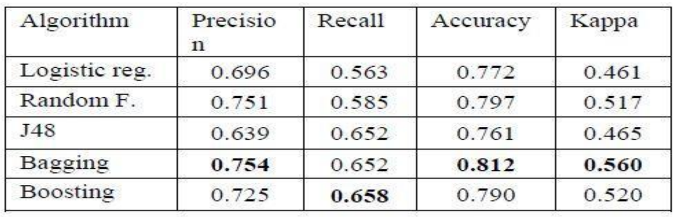
The results show that SVM predicts fire probability well. SVM has the smallest RMSE of 54.0, MSE of 2926.4, RAE of 10.5, and MAE of 2.656 and the highest IG of 2.656 in the testing stage. In general, the results indicate that SVM has the best prediction ability for forest fire compared to other selected SC methods.

2.3 Data mining techniques to predict Forest Fires

The reason for this study was to learn how to predict forest fires in Slovenia using different data mining techniques based on the forest structure GIS (geographical information system) and the weather prediction model - Aladin and MODIS. The data is divided into 3 groups: GIS data (geographic data, part of the land with forest, field, urban part, distance from roads, highways, railways, cities etc.), MODIS data with dependence from the day of the year (average temperature for specific quadrant for specific day, average net primary production for specific day), ALADIN data (temperature, humidity, sum energy, evaporation, speed, direction and course of the wind, transpiration etc.). The data were analyzed with several different data mining algorithms like: logistic regression, random forests, decision trees, bagging and boosting ensemble methods.

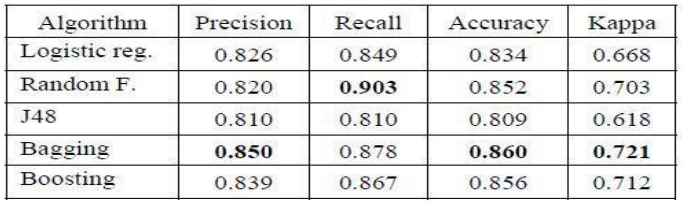
Performances of the experiments were measured in terms of precision, recall, accuracy and kappa statistics for each of the datasets respectively as shown in Table 2.1[3]. From the results we can conclude that Bagging of decision trees shows the best results in terms of predictive accuracy, precision and kappa statistics compared to the other algorithms.

Table 2.1: Performances in Terms of Accuracy, Precision and Kappa



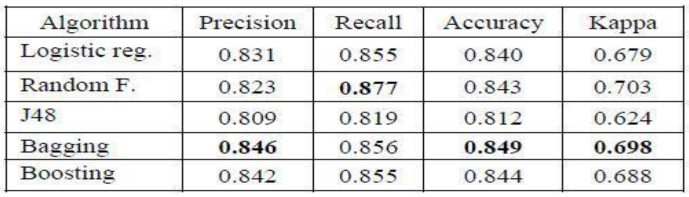
Algorithm	Precision	Recall	Accuracy	Kappa
Logistic reg.	0.696	0.563	0.772	0.461
Random F.	0.751	0.585	0.797	0.517
J48	0.639	0.652	0.761	0.465
Bagging	**0.754**	0.652	**0.812**	**0.560**
Boosting	0.725	**0.658**	0.790	0.520

Table 1 Performances of DM algorithms on Kras data



Algorithm	Precision	Recall	Accuracy	Kappa
Logistic reg.	0.826	0.849	0.834	0.668
Random F.	0.820	**0.903**	0.852	0.703
J48	0.810	0.810	0.809	0.618
Bagging	**0.850**	0.878	**0.860**	**0.721**
Boosting	0.839	0.867	0.856	0.712

Table 2 Performances of DM algorithms on Primorska data



Algorithm	Precision	Recall	Accuracy	Kappa
Logistic reg.	0.831	0.855	0.840	0.679
Random F.	0.823	**0.877**	0.843	0.703
J48	0.809	0.819	0.812	0.624
Bagging	**0.846**	0.856	**0.849**	**0.698**
Boosting	0.842	0.855	0.844	0.688

Table 3 Performances of DM algorithms on continental Slovenia data

2.4 Review on various approaches

This paper was a review on different papers to find out the advantages and disadvantages of existing techniques like:

Artificial Neural Network- An Artificial Neural Network (ANN) is inspired by the way biological nervous systems, such as the brain, process information.

Fuzzy logic- Fuzzy logic is an approach to computing based on "degrees of truth" rather than the usual "true or false" (1 or 0).

Image Processing- Image processing is a method to convert an image into digital form and perform some operations on it to extract some useful information from it.

Satellite Constellation- A group of artificial satellites with coordinated ground coverage, operating together under shared control.

Multisensor Data Fusion- Multisensor data fusion is the combining of sensory data from separate sources such that the resulting information is better than individual sources.

Intelligent system- An intelligent system is a computer system that uses infrared images along with ANN with additional information from sensors which can make decisions.

2.5 Data Mining Approach to predict burned area of Forest Fires

Five different DM techniques, e.g. Support Vector Machines (SVM) and Random Forests, and four distinct feature selection setups (using spatial, temporal, FWI components and weather attributes), were tested on recent real-world data collected from the northeast region of Portugal. The forest Fire Weather Index (FWI) is the Canadian system for rating fire danger and it includes six components: Fine Fuel Moisture Code (FFMC), Duff Moisture Code (DMC), Drought Code (DC), Initial Spread Index (ISI), Buildup Index (BUI) and FWI.

The proposed solution, which is based on SVM and requires only four direct weather inputs (i.e. temperature, rain, relative humidity and wind speed), is capable of predicting small fires. This procedure indicates that all weather conditions affect the model, with the outside temperature being the most important feature, followed by the accumulated precipitation.

2.6 Parallel SVM Based on MapReduce

This mode was to use a support vector machine for classification and regression. Large computation and storage requirements increase due to the increase in the number of training vectors, therefore parallel SVM is studied to increase the speed of computing.

The large dataset is first handled using mapreduce. Mapreduce is an efficient distributed computing model to process large scale data mining problems. Mapreduce is developed in software tools like Hadoop and Twisters. Hadoop and Twisters are open source mapreduce software. Mapreduce in Hadoop does not support iterative map reduce tasks. Twisters support both iterative and non-iterative map reduce and combine tasks. Many SVM software models have been developed like, libSVM, lightSVM, ls-SVM and so on. LibSVM is taken as the most efficient SVM model and widely applied in practice because of its excellent property. Using parallelization, training samples are divided into subsections. Each subsection is trained with a libSVM model. The non-support vectors are filtered with subSVMs. The support vectors of each subSVM are taken as the input of the next layer subSVM. The global SVM model will be obtained through iteration. This shows that parallel SVM along with MapReduce reduces the computation time.

2.7 Artificial neural networks and logistic regression to predict forest fire danger

A model based on the Galician region (north-western Spain) has been proposed and image MODIS was used to track the status and the acquiring land surface temperature (LST). The following have been found: LST 8 days, fire and year history. The LST is an important parameter because higher temperatures correlate with lower humidity, which enables vegetation to ignite In order to assess forest fire hazards from remote sensing and fire history data, artificial neural network (ANN) and logistics regression were used for this work. The land surface temperature and EVI (Enhanced Vegetation Index) were remote sensing inputs used. Different input combinations with logistic regression have been tested. In an artificial neural network, combinations of variables have been implemented and results obtained by the two techniques have been compared. Increased accuracy and recall in artificial neural networks compared to logistic regression. This description was helpful in determining maps of fire hazards that can prevent fires.

2.8 Forest Fire Danger Index Using Geo-Spatial Techniques

K V Suresh Babu [5] provides a comprehensive explanation of various worldwide indices of forest fire conditions, groups of forests and their areas, classification of different rates of hazards, fire accidents in various classes of land cover, level of fire danger of different types of property, categories of vegetation fire accidents, fire hazards for Uttarakhand vegetation categories. In addition, he has described different class and FWI rating systems, such as the Canadian Forest Fire Danger Rating System, the Canadian Fire Risk Index System and the National Fire Danger Rating System. The different forms of forest in India are also being studied. We plan to use his work to measure forest fire weather indicators and measure fire hazard rates thresholds to improve the performance and accuracy.

2.9 Mapping regional forest fire probability using artificial neural network

The Multi-Layer perceptron (MLP) techniques based on back propagation algorithms for data on physical, climate, anthropogenic and fire incident facts were explored by Onur Satir [9] et al. in the Mediterranean forest in Turkey. It was concluded that each area should be studied separately for exact fire risk maps in order to accurately track risks posed by forest fires with the most concise features being tree canopy covering, temperature and Digital Elevation Map (DEMs) and according to cause of fire, vegetation dynamic, climatic conditions and physical environment structures.

2.10 Summary

A comparison on the pros and cons of the different algorithms like Linear regression, Support vector machine, non-linear regression, Random forest, Artificial neural networks and parallel support vector machine that were used in the previous works have been discussed below in Table 2.2.

Table 2.2: Comparison of different Methodology with pros and cons

Algorithm	**Key Idea**	**Formulas**	**Pros**	**Cons**
Linear Regression	Method for linking independent variables with dependent variables and finds a linear relationship between them. It aims to find the value of Y such that the difference between the predicted value and the actual value is minimum. The cost function can be used to find the best values for and which provides the best fit line for the points.	The equation is of the form, Y= a +b X, Where Y is dependent variable and X is independent variable The cost function is given by, $C = \frac{1}{n}\sum_{i=0}^{n}(pred_i - Y_i)^2$	Simple and easy to implement	Most problems in the real world are not linear and are not realistic
Non-Linear Regression	The dependent variables act as a nonlinear part of model parameters and one or more independent variables. It typically generates a curve where Y is a random variable. It is more complex to develop because it uses multiple iterations.	The equation is of the form, $Y = f(X,\beta) + \varepsilon$ Transformed to intrinsically linear $Y = \theta1 + \theta2*1/X$	Well deals with small datasets and effectively uses information to predict unknown parameters	The most productive model research can be established because the experimental information does not have concrete cinematic values.

Random Forest	Many decision trees act as a group in which each individual decision tree gives a class prediction, and this is the final prediction for the class with the highest votes. There is low correlation between the trees/models and the predictions are more accurate	To make a prediction at a new point x then, f˜B(x) =1/BΣ (b1, B) (Tb(x)). Where Tb is the random forest tree and we take the average of B such trees.	Works well for large datasets and avoids overfitting problems by averaging or combining decision tree results.	Random forests are difficult to construct and timely
Artificial neural networks	Similar to how the brain has interconnected neurons and processes information, we have multiple interconnected elements that work together simultaneously to solve a problem. They are multi-layer connected nets. There are three layers, the input layer, hidden layer and output layer.	In general if there are n variables then the equation for ANN will be f(x)=b+w1·x1+w2·x2+...+wn·xn where w is the weights and x is the data points.	It is durable, versatile and can be used in complex designs. The result is obvious	Requires high data and the chances of over-fitting are high

Support Vector Machine	Every data component is drawn as one point in an n-dimensional space and a hyperplane can be defined to differentiate between the classes by maximizing its margin. Having maximum margin between the support vectors and hyperplane is important as this will ensure accurate predictions.	The hyperplane can be found using the equation: w.x+b = 0, where w is the normal vector to the hyperplane and x is the set of points. The width of the margin is (2/|w|)	Functions well in large spaces. It is efficient in memory because it support vectors which form part of a group of training vectors.	If the number of features is much greater than the number of samples, chances of over-fitting are possible. No memory and processing time Portable.
Parallel Support Vector Machine	Parallel methods for large scale classification are implemented to speed up the SVM process. Here the model has multiple SVMs working in parallel on divided datasets. Each SVM will produce support vectors which will be sent to another set of SVMs. This will continue till we get a refined set of support vectors which will not change further.	SVM is implicitly parallelized to get smaller SV sets. Parallelizing SVM with split dataset to get refined set of support vectors	Reduces memory from O(n2) to O(np/m) and improves computation time to O(np2/m) Where n is the number of instances, p is the reduces matrix and m is the number of machines	Reformulations are too memory intensive.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

SRS is a document that completely describes what the proposed software should do without describing how the software will do it. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time. Requirement is a condition or capability to which the system must conform. Requirement Management is a systematic approach towards eliciting, organizing and documenting the requirements of the system clearly along with the applicable attributes.

3.1 Functional Requirements

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality.

- To create a machine learning model to predict forest fires using SVM technique and LIBSVM model.
- To increase the computational speed we use parallel SVM.
- To create a user interface for predicting forest fires and sending an alert message to authorities like the Ministry of Environment, Forest, Climate Change (MOEFCC).

3.2 Non-Functional Requirements

Non- Functional Requirement defines the performance of a software system and how scalable and reliable the system is and it includes scalability, recoverability, availability, capacity etc.

- To give a user-friendly experience using the user interface to check and test the dataset.
- It can be used in real-world using real-time meteorological data.
- To make the application extremely scalable.
- To test the application and ensure maintainability on a timely basis.

3.3 Hardware Requirements

- Processors: Intel i3,i5,i7
- Processor Speed: 3.00GHZ
- RAM: 4GB
- Storage: 50GB
- Monitor: 15inches
- Keyboard: Standard 102 keys
- Mouse: Standard 3 buttons

3.4 Software Requirements

- Operating System can be either Windows 7,8,9,10, XP
- Pandas, Numpy, Sklearn, Pyspark, Matplotlib modules in python
- Jupyter notebook
- Pyspark for implementing Parallel SVM, Binary and Multi class evaluator modules
- Visual studio Code
- Django

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

We have used Django web framework to build the UI for this predictor model. The implementation framework is discussed in detail in this section. Data set is collected from IMD to train and test the model. Based on our literature survey we will be using SVM to predict forest fires. SVM is best for small fires and parallel SVM model is implemented to make the predictions accurate and fast. A detailed explanation about the model and how it is implemented is discussed in the below section.

4.1 System Architecture

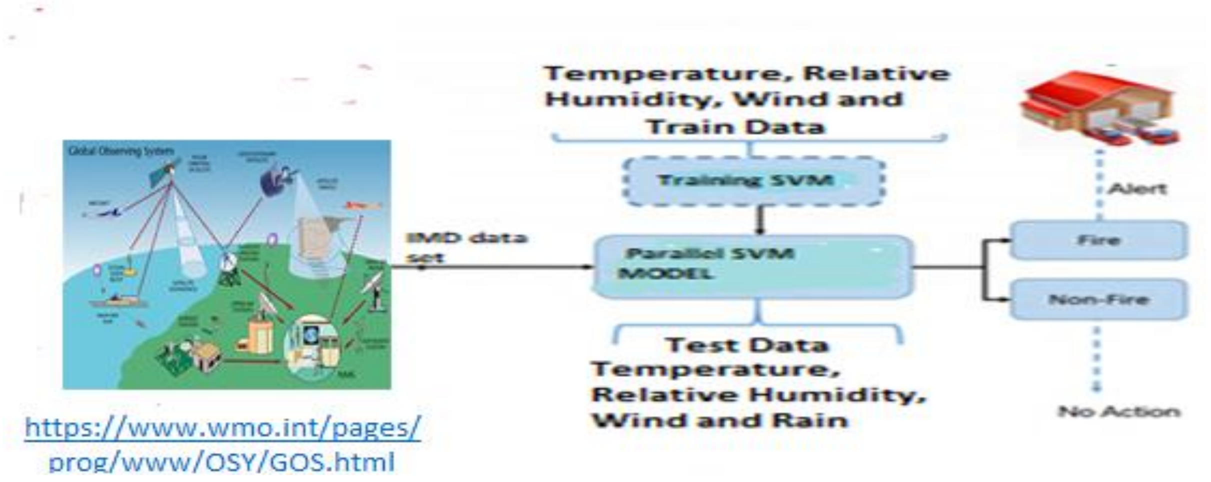

Global Observing System
https://www.wmo.int/pages/prog/www/OSY/GOS.html
IMD data set
Temperature, Relative Humidity, Wind and Train Data
Training SVM
Parallel SVM MODEL
Test Data Temperature, Relative Humidity, Wind and Rain
Alert
Fire
Non-Fire
No Action

Fig 4.1: System Architecture

As shown in Fig 4.1, the IMD dataset is fed as input to the Forest fire detection system implemented using Parallel SVM wherein the dataset contains only the basic attributes like Temperature, Relative humidity, Wind and Rain. Then the data is processed by calculating the Fire Weather Indices (FWI). After the data processing, the model is trained using the train data and for testing purpose, the test data is fed with the same attributes as the train data. The model produces a result based on whether Fire happened (Fire =1) or not. If the output contains Fire = 1, then an alert is sent to the forest department and other concerned departments to take the necessary actions.

4.2 Implementation Framework

Fig 4.2 is a web application using Django web framework. Like in a traditional data-driven website, a web application waits for HTTP requests from the web browser (or user). When a request is received the application works out what is needed based on the URL and possibly information in POST data or GET data. Depending on what is required it will perform tasks required to satisfy the request, in this case it will take the file uploaded and will run the programs in the background to predict forest fires. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template.

Django web applications typically group the code that handles each of these steps into separate files:

- **URLs:** A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data.
- **View:** A view is a request handler function, which receives HTTP requests and returns HTTP responses.
- **Forest Fire:** Model in Python that will run when the dataset is uploaded by the front end user.

- **Templates:** A template is a text file defining the structure or layout of a file (such as an HTML page), with placeholders used to represent actual content. A view can dynamically create an HTML page using an HTML template, populating it with data from our program. This acts as the client side.

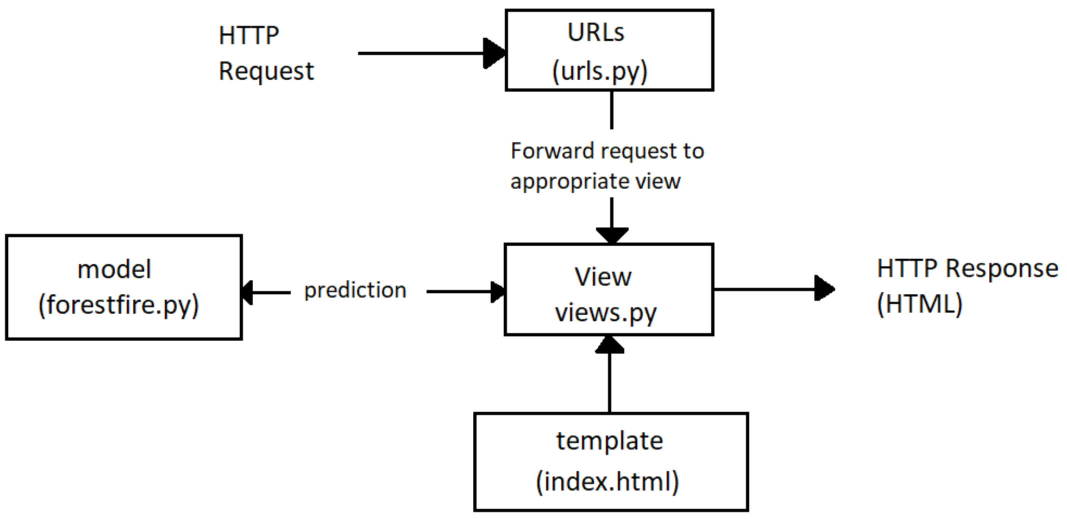


Fig 4.2: Django implementation Framework

4.3 Dataset

The dataset is collected from Indian Meteorological department (IMD). The four basic attributes considered are: Temperature, Wind, Rainfall and Relative humidity. From these basic parameters, fuel moisture codes like Ffmc (fine fuel moisture code), DMC (duff moisture code), DC (drought code) and fire behaviour indices like ISI (initial spread index), BUI (buildup index) and FWI (fire weather index) are calculated where the derived attributes are a function of the basic attributes as shown below:

- FFMC is a function of Temperature, Relative humidity, Rain and Wind.
- DMC is a function of Temperature, Relative humidity and Rain
- DC is a function of Temperature and Rain
- ISI is a function of ffmc, wind
- BUI is a function of ffmc, dmc
- FWI is a function of ISI and BUI

Code:

```
def main():

   import sys

   ffmc0= 85.0

   dmc0= 6.0

   dc0= 15.0

   my_csv_in = sys.argv[1]

   with open(my_csv_in, 'r') as f_in:

       print("opened")

       next(f_in)

       with open('/Users/krsingh/Desktop/datasets/testset3.csv', 'w') as f_out:


h=["Year","Month","Day","FFMC","DMC","DC","ISI","BUI","Temp","RH","Wind","Rain","FWI",
"Intensity","Fire"]

           hd=','.join(h)

           f_out.write(hd)

           f_out.write("\r")

           for line in f_in:

               l=line.rstrip().split(',')

               yr=l[0]
```

```
                mth=l[1]

                day=l[2]

                temp=float(l[3])

                rhum=float(l[4])

                wind=float(l[5])

                prcp=float(l[6])

                if rhum> 100.0:

                    rhum=100.0

                mth=int(mth)

                fwisystem= FWICLASS(temp,rhum,wind,prcp)

                ffmc = fwisystem.FFMCcalc(ffmc0)

                dmc = fwisystem.DMCcalc(dmc0,mth)

                dc = fwisystem.DCcalc(dc0,mth)

                isi = fwisystem.ISIcalc(ffmc)

                bui = fwisystem.BUIcalc(dmc,dc)


l=[str(yr),str(mth),str(day),str(round(ffmc,4)),str(round(dmc,4)),str(round(dc,4)),str
(round(isi)),str(round(bui,4)),str(round(temp,4)),str(round(rhum,4)),str(round(wind,4)
),str(round(prcp,4)),str(round(fwi,4)),str(intensity),str(fire)]

            fwi = fwisystem.FWIcalc(isi,bui)
```

These factors affect the occurrence of Forest fire in various ways:

- **Relative Humidity:** When the relative humidity is 40 percent, it means that the atmosphere contains 40 percent of the moisture that it could contain at that same temperature. When the humidity is high, it’s harder for the moisture to evaporate into the air. Consequently, high humidity acts like a damper on a stove. If the humidity is 100 percent or close to it, the fuel will not dry. On the other hand, the lower the relative humidity, the quicker the moisture will evaporate. The lower the relative humidity, the more readily a fire will start and burn; the more vigorously a fire will burn.

- **Temperature:** Air temperature has a direct influence on fire behaviour because of the heat requirements for ignition and continuing the combustion process. Forest fuels receive heat by radiation from the sun. As a result, less heat is required for ignition. The differential heating of the earth’s surface is the driving force behind most of the influences on the atmosphere. The sun emits short-wave energy rays (radiation). When striking a solid object such as trees or grass, it is warmed. Temperature is the most important weather factor affecting fire behaviour. Warm fuels will ignite and burn faster because less heat energy is used to raise the fuels to their ignition temperature. Fuels exposed to sunlight will be warmer than the fuels in shade. They will also be drier. For this reason, fuels not shaded by an overstory will generally be warmer and drier resulting in a more intense fire.
- **Wind:** Wind increases the supply of oxygen, which results in the fire burning more rapidly. It also removes the surface fuel moisture, which increases the drying of the fuel. Air pressure will push flames, sparks and firebrands into new fuel. By pushing the flames closer to the fuel in front of the fire, the fuel is preheated quicker because of the increased radiant heat discussed previously. More of the fuel becomes available for combustion since it is dryer and can reach ignition temperature quicker.
- **Rain:** Precipitation (rain or snow) has a direct and immediate effect on fuel moisture and relative humidity. Temperature usually drops as well and the winds become calm. When the atmosphere becomes saturated, precipitation usually occurs if more moisture is added. Precipitation will quickly dampen the surface of fuels to the point that fires cannot ignite and no wildfires will occur. Rain can prevent back-burning, making it harder to build control lines and lead to patchy burnt areas that can flare up again. For rain to extinguish the fires, it will require inches of steady falls over an extended period. As the vegetation greens-up, prescribed burning conditions may deteriorate. If, however, a winter drought occurs and continues into the spring, fires will readily burn on into the summer because of the larger amount of dead, dry fuel and low fuel moisture. These fires may be more difficult to control and do more damage due to burning deeper into the litter and

consuming larger size fuel. During long periods of dry weather, drought, moisture that is toward the centre of larger fuels and deeper in surface litter is able to work its way to the surface and evaporate into the dry atmosphere. As a result, a larger percent of the total fuel becomes available fuel; available to burn.

- **FFMC:** This index classifies the moisture content of litter and other cured fine fuels, like needles or twigs less than 1 cm in diameter. FFMC is representative of the top litter layer 1-2 cm deep and has a short-term memory, only reflecting weather conditions that have occurred over the past three days.
- **DMC:** This index indicates the moisture content of loosely compacted organic layers with a depth of 5-10cm. DMC fuels have a slower drying rate than FFMC fuels and DMC may be used in predicting the probability of fire ignition by lightning.
- **DC:** reflects the moisture content of compact organic layers, 10-20cm deep. It is an indicative long-term moisture condition and deep burning fires.
- **ISI:** this index combines FFMC and wind speed being a good indicator for fire spread.
- **BUI:** represents the total fuel available for the spreading of fire.

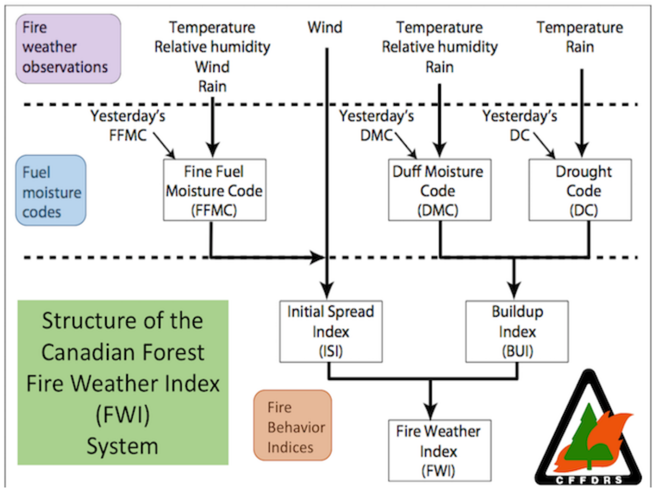


Fig 4.3: Structure of FWI System

4.4 Flowchart

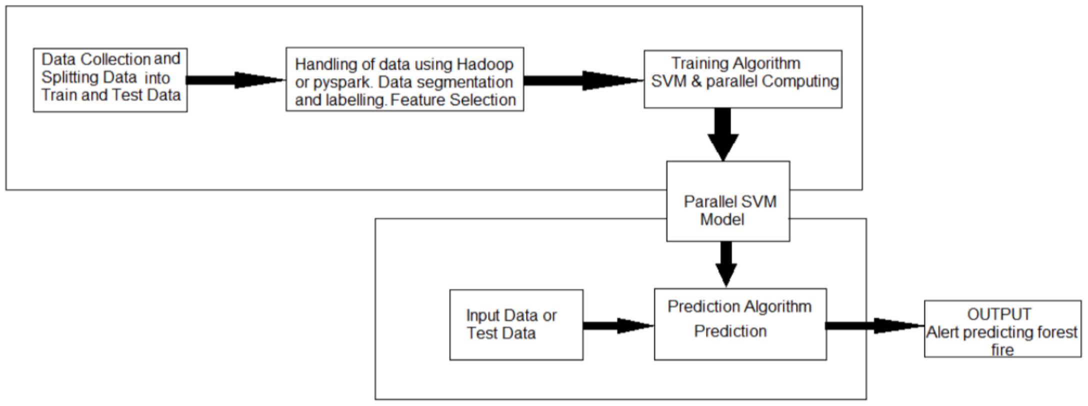

Data Collection and Splitting Data into Train and Test Data
Handling of data using Hadoop or pyspark. Data segmentation and labelling. Feature Selection
Training Algorithm SVM & parallel Computing
Parallel SVM Model
Input Data or Test Data
Prediction Algorithm Prediction
OUTPUT Alert predicting forest fire

Fig 4.4: Process Outline

- ➢ Data collected will be pre-processed by the model to calculate dependent attributes based on independent attributes and will be added to the dataset. The independent attributes are namely temperature, rain, rh and wind. The dependent attributes are the forest weather indices like FFMC, DMC, DC, ISI, BUI and FWI. The data is split into test and train data.

- ➢ The training data is hardcoded with the model to train the model while the test data is the data that the user uploads to the predictor to check forest fire alerts for uploaded data. As the data is very large we and the number of support requests created are too huge which will affect the efficiency and accuracy of the system, we will use parallel processing for big data using Spark.
- ➢ The python module pyspark is used and rdd data frames are used for data segmentation and labeling. This processed data is then fed to the trained model to predict forest fires with the uploaded test dataset.
- ➢ The output of this model is a table of alerts generated for the uploaded data set. We also have an alerting system that can be used to send alerts to respective departments in areas of data uploaded.

CHAPTER 5

IMPLEMENTATION

5.1 Support Vector Machine

Support vector machine is a supervised learning algorithm which can be used for classification and regression. In SVM, the data is represented in an n-dimensional space where it can predict whether a new training example falls into the same category or a different category. The main aim of SVM is to find a hyperplane in the n- dimensional space that can clearly classify the data points. There are several potential hyperplanes which could be chosen to distinguish the two classes of data points. But the ideal hyperplane is the one that maximizes the margin i.e. the maximum distance between data points of both classes as shown in Fig 5.1. Hyperplanes are boundaries for decision making which help to distinguish data points. Data points which fall on either side of the hyperplane can be attributed to various classes. The dimension of the hyperplane depends upon the number of features.

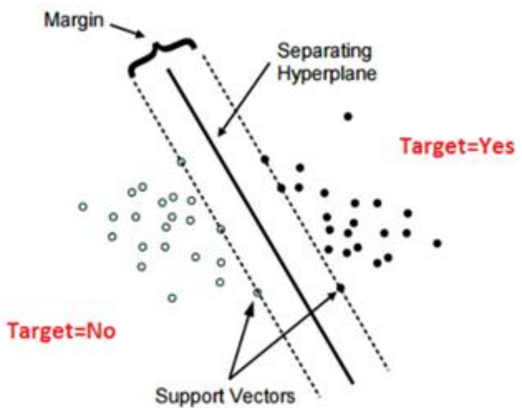


Fig 5.1: Support Vector Machine

As shown in Fig 5.1, Support vectors are the data points that are closer to the hyperplane. They influence the position and orientation of the hyperplane. These support vectors play a major role in classifying the data points to different classes.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane like in Fig 5.2. It becomes difficult to imagine when the number of features exceeds 3.

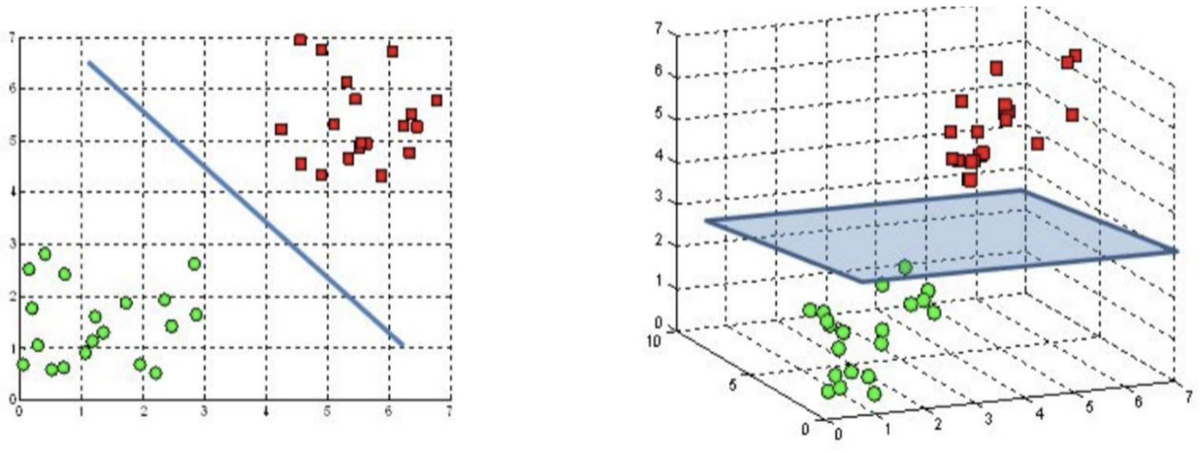


Fig 5.2: Hyperplanes in 2D and 3D

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

A major disadvantage of SVM is that it does not perform well when the training set is large in size as the storage and compute requirements increase with increase in the training vectors. Hence, Parallel SVM can be used in order to reduce the computational time and improve the performance.

5.2 Parallel Support Vector Machine

The parallel SVM is based on the Cascade SVM model. The SVM is trained through partial SVMs. Each subSVM is used as a filter which takes us towards the global optimum. Using the Parallel SVM model, large scales of data can be divided into smaller, independent sub data. The former subSVMs support vectors are used as the input to the later subSVMs as shown in Fig 5.3.

The support vectors of two SVMs are combined into one set and sent as input to a new SVM. This process is continued until a single set of vectors is left which is the global optimum. The size of the training set is reduced in each subsequent stage. Each subSVM is trained with a SVM model like libSVM, lightSVM etc. Most efficient SVM model is libSVM.

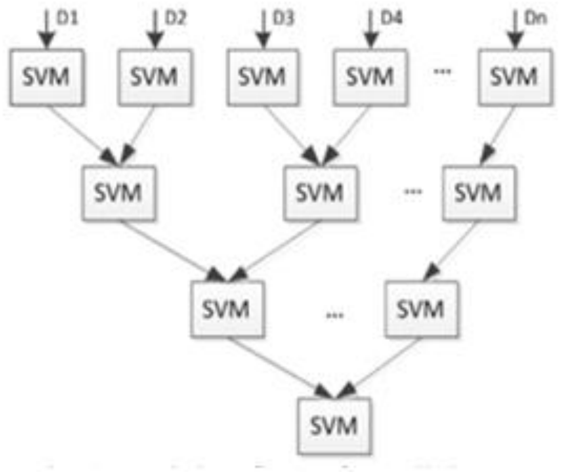


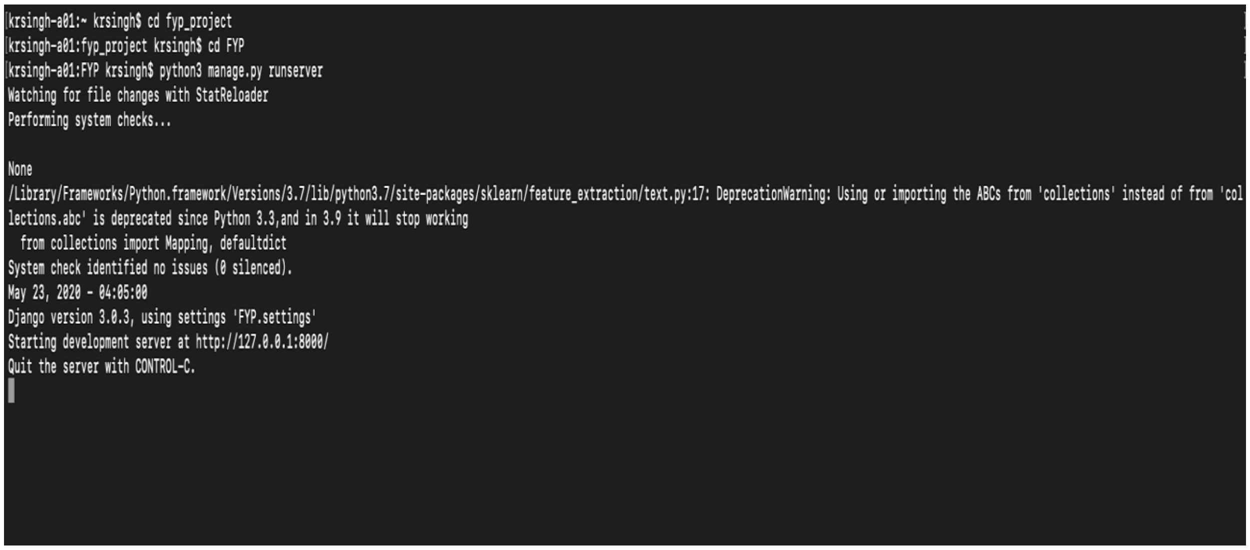
Fig 5.3: Parallel Support Vector Machine

(Figure taken from paper)

Explicit parallelization approaches parallelize the computation within each iteration as well as parallelizing kernel computations. Parallelize reduces the number of working sets for the model thereby speeding up the testing process and making it more efficient in predictions. Parallelizing the SVM is a method where the huge dataset of 30 years data will be divided into smaller datasets and is fed into multiple SVMs simultaneously that outputs a set of support vectors along with the another set of support vectors are fed into another SVM. This is done until we get one set of support vectors that have maximum margin from the hyperplane which will make this model more reliable and will predict with better accuracy and efficiency. To achieve this we will use spark for big data computing and rdd datasets for parallel computing in our program. Forest fires could be due to multiple reasons; it is very difficult to make a prediction system. There could be multiple reasons for forest fires, what we want to achieve using our model is a reliable system that could predict correctly based on weather conditions if fire occurs and also the intensity of fire.

5.3 Front-end and Back-end implementation details

The **python manage.py run server** command is executed in the Django project directory as shown in Fig 5.4 and the page appears in the localhost URL.



```
[krsingh-a01:~ krsingh$ cd fyp_project
[krsingh-a01:fyp_project krsingh$ cd FYP
[krsingh-a01:FYP krsingh$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

None
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/sklearn/feature_extraction/text.py:17: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from 'col
lections.abc' is deprecated since Python 3.3,and in 3.9 it will stop working
  from collections import Mapping, defaultdict
System check identified no issues (0 silenced).
May 23, 2020 - 04:05:00
Django version 3.0.3, using settings 'FYP.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Fig 5.4: Command Prompt

Sending request to the right Url (url.py)

A URL mapper is typically stored in a file named urls.py. In the example below, the mapper (urlpatterns) defines a list of mappings between routes (specific URL patterns) and corresponding view functions. If an HTTP Request is received that has a URL matching a specified pattern then the associated view function will be called and passed the request.

Code

```
urlpatterns = [
   re_path(r'^admin/', admin.site.urls),
   re_path(r'^$', views.button),
   re_path(r'^output',views.output, name='script'),
   re_path(r'^external',views.external),
   re_path(r'^alert/',views.alert, name="alert"),
   re_path(r'^prediction/',views.prediction, name="prediction"),
   re_path(r'^validation/',views.validation, name="validation"),
   re_path(r'^predictor/',views.predictor, name="predictor")
]
urlpatterns+=staticfiles_urlpatterns()
```

The first argument to both methods is a route (pattern) that will be matched. The path() method uses angle brackets to define parts of a URL that will be captured and passed through to the view function as named arguments.

Handling the requests (view.py)

Views are the heart of the web application, receiving HTTP requests from web clients and returning HTTP responses. In between, they marshall the other resources of the framework to access databases, render templates, etc.

Code

```
from subprocess import run, PIPE ,Popen, call,check_output
def button(request):
   return render(request,'index.html')
def output(request):
   return render(request,'index.html')
```

```
def external(request):
   file=request.FILES['myfile']
   f="/Users/krsingh/Desktop/datasets/"+str(file)

output=Popen(['python3','/Users/krsingh/fyp_project/FYP/FYP/forestfire.py',f],stdout=P
IPE,universal_newlines=True)
   l=[]
   for line in output.stdout.readlines():
       l.append(line)
   print(len(l))
   return render(request, 'index.html',
{'Accuracy_LSVM':l[1],'Precision_LSVM':l[2],'Recall_LSVM':l[3],'Time_LSVM':l[4],'Accur
acy_PSVM':l[5],'Precision_PSVM':l[6],'Recall_PSVM':l[7],'Time_PSVM':l[8]})

def alert(request):
   return render(request,'pg2.html')
def prediction(request):
   return render(request,"prediction.html")
def validation(request):
   return render(request,"validation.html")
def predictor(request):
   return render(request,"index.html")
```

Defining model(foresfire.py)

The definition of the model is independent of the underlying database and in our model it runs the predictor algorithm when the test dataset gets uploaded on the predictor page of UI. The view external() gets called when the file gets uploaded and the model predicts the results for the dataset uploaded.

Code

```
   import pyspark
   from pyspark.sql import SparkSession
   from pyspark.ml import Pipeline
   from pyspark.ml.feature import StringIndexer
   from pyspark.ml.feature import VectorAssembler
   from pyspark.ml.evaluation import MulticlassClassificationEvaluator
   from pyspark.ml.feature import QuantileDiscretizer
   #import Spark and MLlib packages
   from pyspark import SparkContext, SparkConf
   from pyspark.mllib.regression import LabeledPoint
   from pyspark.mllib.classification import SVMWithSGD, SVMModel
   from pyspark.mllib.classification import LogisticRegressionWithLBFGS
   #import data analysis packages
   import numpy as np
   import pandas as pd
   import sklearn
   from pandas import Series, DataFrame
   from sklearn import svm
   from sklearn.svm import SVC
   from sklearn.cross_validation import train_test_split
   from sklearn import metrics
   from numpy import array
   from timeit import default_timer as timer
   #import data visualization packages
   import matplotlib.pyplot as plt
   import seaborn as sns
   sns.set_style('whitegrid')
   import random
   dataframets= pandas.read_csv(r"/Users/krsingh/Desktop/datasets/testset3.csv")
   dataframetr =
pandas.read_csv(r"/Users/krsingh/Desktop/datasets/newtraintdata7.csv")
   featuredatatr=dataframetr.iloc[:, :14]
   targetvaluestr=dataframetr.iloc[:,14:]
   featuredatats=dataframets.iloc[:, :14]
   targetvaluests=dataframets.iloc[:,14:]
   x_train=featuredatatr
   x_test=featuredatats
   y_train=targetvaluestr
   y_test=targetvaluests
# SVM regularizaion parameter
   C = 1.0
   svc = svm.SVC(kernel = 'linear', C=C).fit(x_train, y_train)
   start = timer()
   predicted = svc.predict(x_test)
   end =timer()
   predicted = svc.predict(x_test)
expected = y_test
   from pyspark.ml import Pipeline
   from pyspark.ml.feature import StringIndexer
   from pyspark.ml.feature import VectorAssembler
   from pyspark.ml.evaluation import MulticlassClassificationEvaluator
   from pyspark.ml.feature import QuantileDiscretizer
   from pyspark.ml.classification import LinearSVC
   from pyspark.ml.evaluation import BinaryClassificationEvaluator
   spark = SparkSession \
       .builder \
       .appName("Spark ML example on data ") \
       .getOrCreate()
   datatrain= "/Users/krsingh/Desktop/datasets/newtraintdata7.csv"
   dftr = spark.read.csv(datatrain,header = 'True',inferSchema='True')
   datatest= "/Users/krsingh/Desktop/datasets/testset3.csv"
   dfts = spark.read.csv(datatest,header = 'True',inferSchema='True')
   featuretr = VectorAssembler(
       inputCols=[x for x in dftr.columns],
       outputCol='features')
   feature_vector_tr= featuretr.transform(dftr)
   featurets = VectorAssembler(
       inputCols=[x for x in dfts.columns],
       outputCol='features')
   feature_vector_ts= featurets.transform(dfts)
   trainingData=feature_vector_tr
   testData=feature_vector_ts
   from pyspark.ml.classification import LinearSVC
   from pyspark.ml.evaluation import BinaryClassificationEvaluator
   svm = LinearSVC(labelCol="Fire", featuresCol="features")
   svm_model = svm.fit(trainingData)
   start1=timer()
   svm_prediction = svm_model.transform(testData)
```

Rendering data using templates (index.html)

Templates are often used to create HTML and code snippets show what the HTML template called by the render() function in the previous section might look like.

Code

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
<title>Forest_fire_predictor</title>
</head>
<link rel="stylesheet" href="{% static 'FYP.css' %}">
<body style="background-image: url({% static 'wall.jpg' %}); background-size:100%;
          background-repeat: y-repeat;">

<header>
<div class="wrapper" id="nav">
   <div><h1 style="color :white; background-color: black; padding-top: 10px; padding-
bottom: 10px; padding-left:15px;text-align:left">Forest Fire Prediction System</h1>
   <ul class="nav-area">

               <li><a href="{% url 'predictor' %}">PREDICTOR</a></li>&nbsp;
               <li><a href="{% url 'alert' %}">ALERTS</a></li>&nbsp;
               <li><a href="{% url 'prediction' %}">INTENSITY</a></li>&nbsp;
               <li><a href="{% url 'validation' %}">VALIDATION</a></li>&nbsp;
       </ul><div>
<div>
</header>
<div class="filechoose" style="background: black; opacity: 0.8; box-shadow: 0 4px 8px
0 rgba(0,0,0,0.2); border-radius: 30px; margin-left: 500px; width: 450px; height:
150px;">
   <form action='/external/' enctype="multipart/form-data" method="post"
enctype="multipart/form-data" style="color: white; margin-top: 50px; margin-left:
50px; padding-top: 30px;">
     <label for="myfile" style="font-size: 20px; ">Select a file: &nbsp; </label>
     {% csrf_token %}{{ form.as_p }}
     <input type="file" id="myfile" name="myfile" ><br><br>
     {{Accuracy}}
     <input type="submit" style="border: 1px solid black; border-radius: 10px;
padding: 10px 20px 10px 20px; margin-left: 140px; margin-top: 10px; font-weight:
bold;">
   </form>
```

```
</div>
<div style="margin-top: 50px; padding-top: 10px; border: 2px 2px 2px 2px; border-
style: solid;border-color:#6B8E23; background-color: black;">
           <h2 style="font-size: 40px;margin-left: 650px;color: white;
">ALGORITHMS</h2>
       </div>
       <div class="row">
           <div class="column" onclick="openTab('b2');" style="background: black;
opacity: 0.8; box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2); border-radius: 30px; height:
315px;">
<img src="{% static 'svm.gif' %}" style="border-radius: 25px; width: 60%; height: 75%;
margin-top: -5px;">
               <h2 style="color: white; font-size: 25px; font-style: italic; opacity:
1;"><b>LINEAR SVM</b></h2> <br>
           </div>
           <div class="column" onclick="openTab('b3');" style="background: black;
opacity: 0.8; box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2); border-radius: 30px;height:
317px;">
               <img src="{% static 'ksvm.gif' %}" style="border-radius: 25px; width:
50%; height: 70%; margin-top: 5px;">
               <h2 style="color: white; font-size: 25px; font-style: italic; opacity:
1;"><b>PARALLEL SVM</b></h2> <br>
           </div>
       </div>
       <div id="b2" class="containerTab" style="display:none;background:black;
opacity: 0.8;">
           <span onclick="this.parentElement.style.display='none'" class="closebtn"
style="cursor: pointer;">&times;</span>
           <div style="margin-top: 10px;"><center style="color: white; font-size:
15px; font-style: italic; ">Accuracy of Linear SVM: {{Accuracy_LSVM}} <br>
             Precision of Linear SVM: {{Precision_LSVM}} <br>
             Recall of Linear SVM: {{Recall_LSVM}} <br>
             Execution time taken by Linear SVM: {{Time_LSVM}} <br>
             </center>
       </div>
       </div>
       <div id="b3" class="containerTab" style="display:none;background:black;
opacity: 0.8;">
           <span onclick="this.parentElement.style.display='none'" class="closebtn"
style="cursor: pointer;">&times;</span>
           <div style="margin-top: 10px;"><center style="color: white; font-size:
15px; font-style: italic; "> Accuracy of Parallel SVM: {{Accuracy_PSVM}}<br>
             Precision of Parallel SVM: {{Precision_PSVM}} <br>
             Recall of Parallel SVM: {{Recall_PSVM}} <br>
             Execution time taken by Parallel SVM: {{Time_PSVM}} <br>
</center></div>
       </div>
       <script>
function openTab(tabName) {
           var i, x;
           x = document.getElementsByClassName("containerTab");
           for (i = 0; i < x.length; i++) {
             x[i].style.display = "none";
           }
           document.getElementById(tabName).style.display = "block";
       }       </script>
```

The home page of our forest fire prediction system is shown in Fig 5.3 which contains four tabs: Predictor, Alerts, Intensity and Validation. The dataset can be uploaded and the results of two algorithms can be seen.

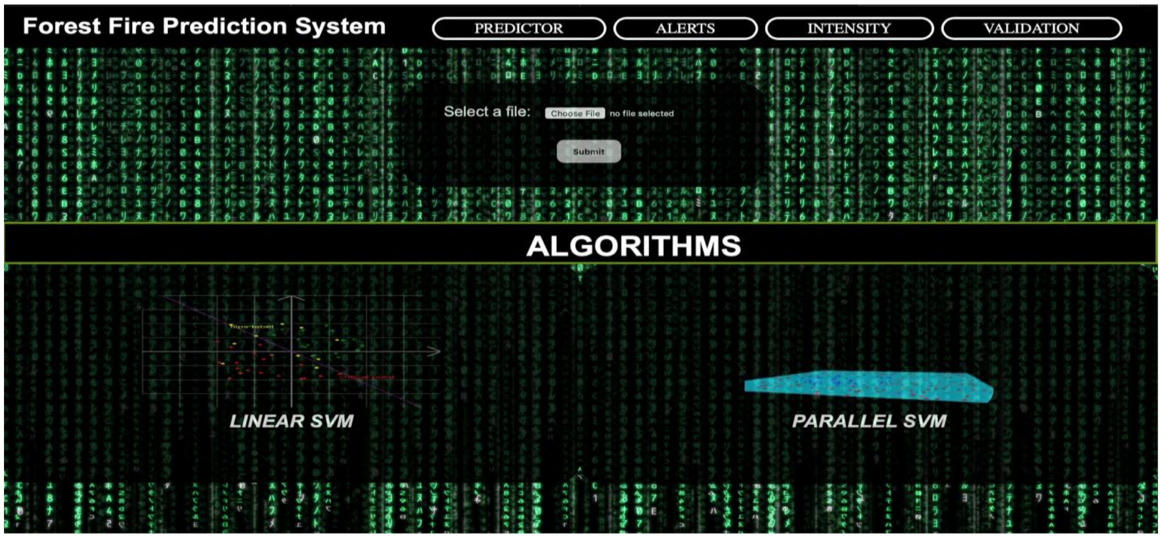


Fig 5.5: User Interface

Triggered Alerts

The model has an automated alert system that gets triggered whenever the predictions are made. The alerts are in tabular form and give the month day details for when the fire is predicted and also the intensity of the fire based on the thresholds applied on the weather indices.

Code

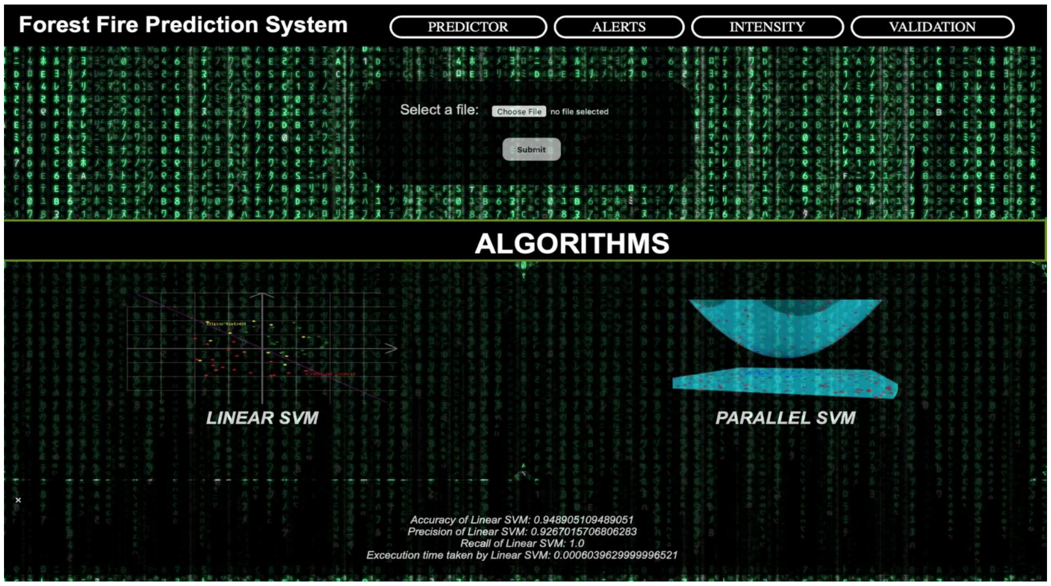
```
from email.mime.text import MIMEText
   from email.mime.multipart import MIMEMultipart
   import smtplib
   fromaddr = "ForestfireAlerts.gmail.com"
   toaddr = ["whom so ever specified"
   for i in range(len(toaddr)):
       html = open("/Users/krsingh/fyp_project/FYP/FYP/template/pg2.html")
       msg = MIMEMultipart()
       msg['From'] = fromaddr
       msg['To'] = toaddr[i]
       msg['Subject'] = "Fire Alerts Report"
       part2 = MIMEText(html.read(), 'html')
       msg.attach(part2)
       debug = False
       if debug:
           print(msg.as_string())
       else:
           server = smtplib.SMTP('smtp.gmail.com',587)
           server.starttls()
           server.login("forestfire.alerts@gmail.com", "**********")
           text = msg.as_string()
           server.sendmail(fromaddr, toaddr[i], text)
           server.quit()
```

CHAPTER 6

RESULTS AND DISCUSSION

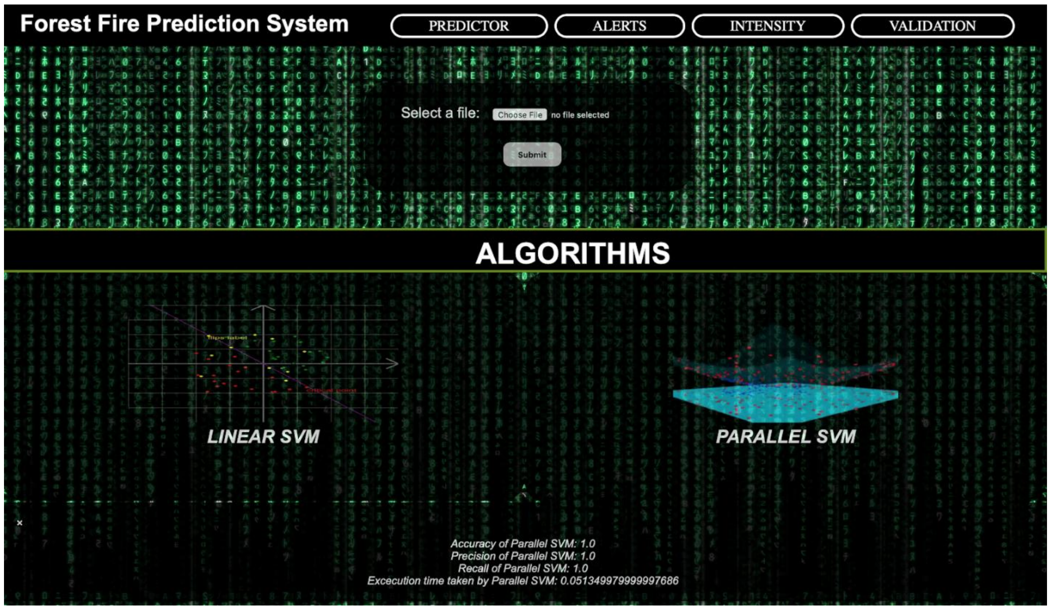
The results are divided into 4 tabs: Predictor, Alerts, Intensity and Validation.

- **Predictor Tab:**
    - A comparison of Linear SVM and Parallel SVM is shown.
    - Both algorithms are trained to predict forest fires. The accuracy of prediction, precision, and recall values are displayed in the results.
    - The estimated time required for the algorithms to run is also displayed.
    - The results show a slight variation in the displayed values, thus proving the higher efficiency of parallel SVM compared to linear SVM.
    - The accuracy, precision, and recall values for parallel SVM is higher than SVM.
    - Fig 6.1 (a) shows the results of Linear SVM and Fig 6.1 (b) shows the results of Parallel SVM.



```
Accuracy of Linear SVM: 0.948905109489051
Precision of Linear SVM: 0.9267015706806283
Recall of Linear SVM: 1.0
Excecution time taken by Linear SVM: 0.0006039629999996521
```

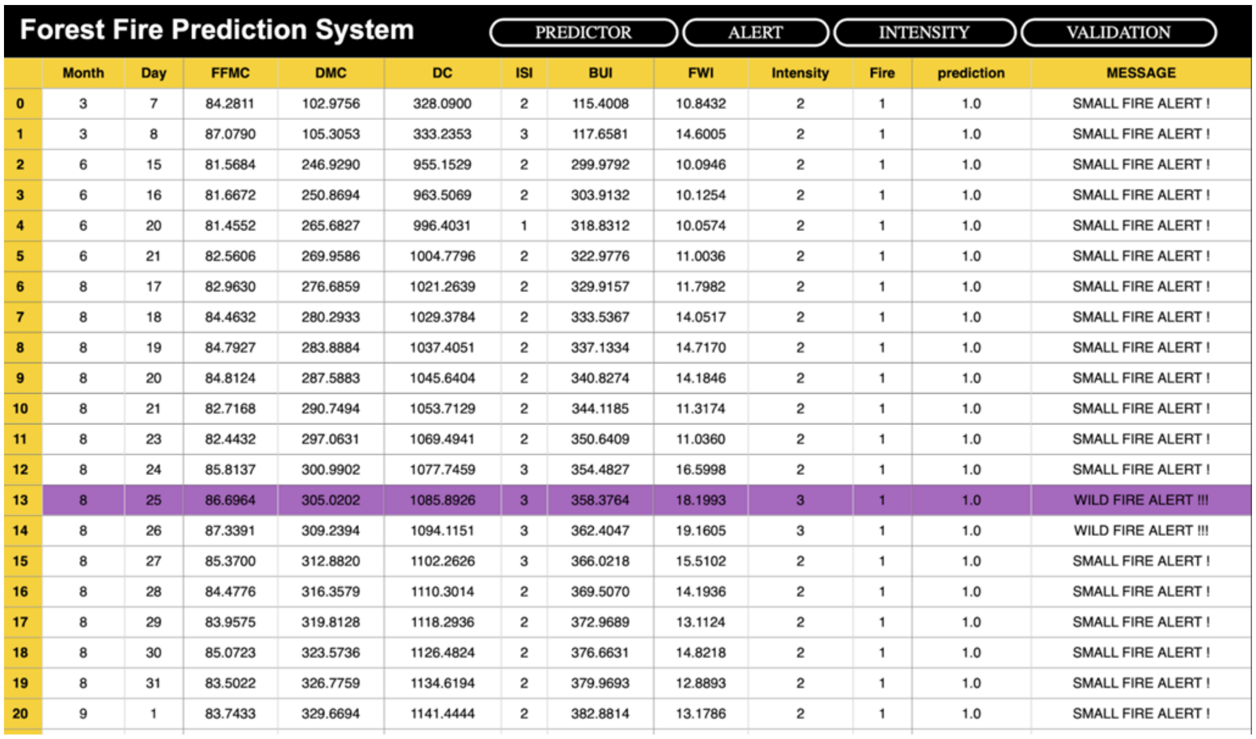
Fig 6.1 (a): Linear SVM Results



```
Accuracy of Parallel SVM: 1.0
Precision of Parallel SVM: 1.0
Recall of Parallel SVM: 1.0
Excecution time taken by Parallel SVM: 0.051349979999997686
```

Fig 6.1 (b): Parallel SVM Results

- **Alerts tab:**
  - ➢ In the second tab, a table of the data values is displayed.
  - ➢ These data values correspond to the entries where the occurrence of fire was predicted as 1.
  - ➢ A column displaying the intensity of the fire is also displayed.
  - ➢ Fig 6.2 shows the Alert tab showing the predicted fire days.
- **Intensity Tab:**
  - ➢ A graph showing month wise occurrence of fire is plotted as shown in Fig 6.3.
- **Validation Tab:**
  - ➢ A graph showing the predicted fire graph is plotted as shown in Fig 6.4 (a).
  - ➢ A graph showing the validation of our model with predicted and actual is plotted as shown in Fig 6.4 (b).



	Month	Day	FFMC	DMC	DC	ISI	BUI	FWI	Intensity	Fire	prediction	MESSAGE
0	3	7	84.2811	102.9756	328.0900	2	115.4008	10.8432	2	1	1.0	SMALL FIRE ALERT !
1	3	8	87.0790	105.3053	333.2353	3	117.6581	14.6005	2	1	1.0	SMALL FIRE ALERT !
2	6	15	81.5684	246.9290	955.1529	2	299.9792	10.0946	2	1	1.0	SMALL FIRE ALERT !
3	6	16	81.6672	250.8694	963.5069	2	303.9132	10.1254	2	1	1.0	SMALL FIRE ALERT !
4	6	20	81.4552	265.6827	996.4031	1	318.8312	10.0574	2	1	1.0	SMALL FIRE ALERT !
5	6	21	82.5606	269.9586	1004.7796	2	322.9776	11.0036	2	1	1.0	SMALL FIRE ALERT !
6	8	17	82.9630	276.6859	1021.2639	2	329.9157	11.7982	2	1	1.0	SMALL FIRE ALERT !
7	8	18	84.4632	280.2933	1029.3784	2	333.5367	14.0517	2	1	1.0	SMALL FIRE ALERT !
8	8	19	84.7927	283.8884	1037.4051	2	337.1334	14.7170	2	1	1.0	SMALL FIRE ALERT !
9	8	20	84.8124	287.5883	1045.6404	2	340.8274	14.1846	2	1	1.0	SMALL FIRE ALERT !
10	8	21	82.7168	290.7494	1053.7129	2	344.1185	11.3174	2	1	1.0	SMALL FIRE ALERT !
11	8	23	82.4432	297.0631	1069.4941	2	350.6409	11.0360	2	1	1.0	SMALL FIRE ALERT !
12	8	24	85.8137	300.9902	1077.7459	3	354.4827	16.5998	2	1	1.0	SMALL FIRE ALERT !
13	8	25	86.6964	305.0202	1085.8926	3	358.3764	18.1993	3	1	1.0	WILD FIRE ALERT !!!
14	8	26	87.3391	309.2394	1094.1151	3	362.4047	19.1605	3	1	1.0	WILD FIRE ALERT !!!
15	8	27	85.3700	312.8820	1102.2626	3	366.0218	15.5102	2	1	1.0	SMALL FIRE ALERT !
16	8	28	84.4776	316.3579	1110.3014	2	369.5070	14.1936	2	1	1.0	SMALL FIRE ALERT !
17	8	29	83.9575	319.8128	1118.2936	2	372.9689	13.1124	2	1	1.0	SMALL FIRE ALERT !
18	8	30	85.0723	323.5736	1126.4824	2	376.6631	14.8218	2	1	1.0	SMALL FIRE ALERT !
19	8	31	83.5022	326.7759	1134.6194	2	379.9693	12.8893	2	1	1.0	SMALL FIRE ALERT !
20	9	1	83.7433	329.6694	1141.4444	2	382.8814	13.1786	2	1	1.0	SMALL FIRE ALERT !

Fig 6.2: Predicted Fire Days

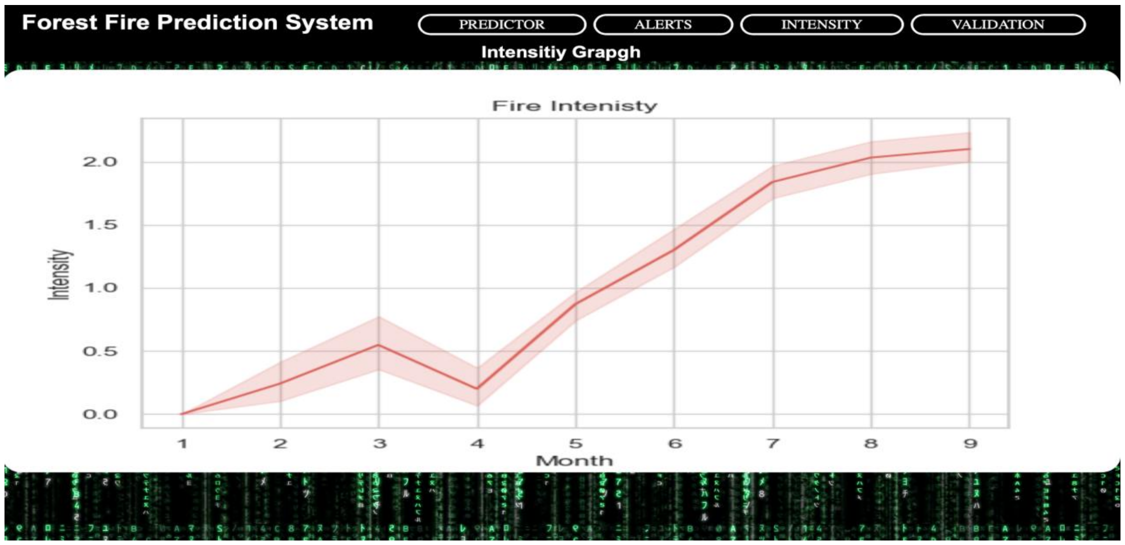
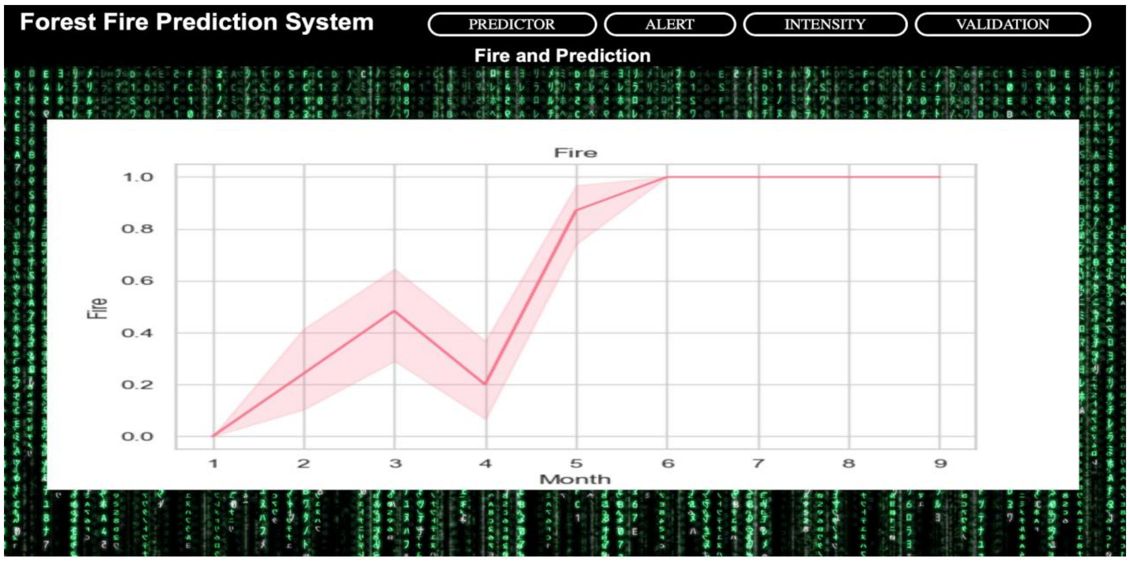


Fig 6.3: Fire Intensity vs. Month



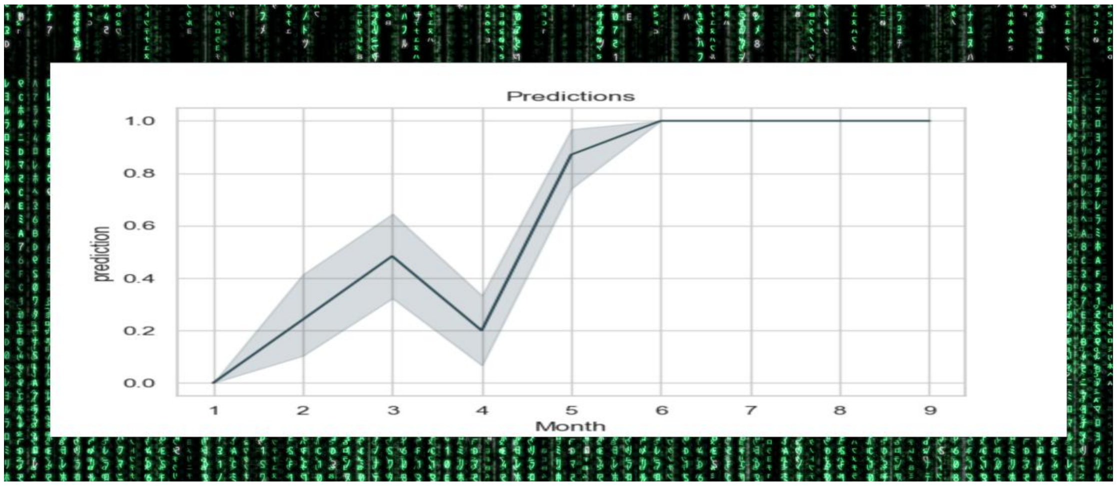
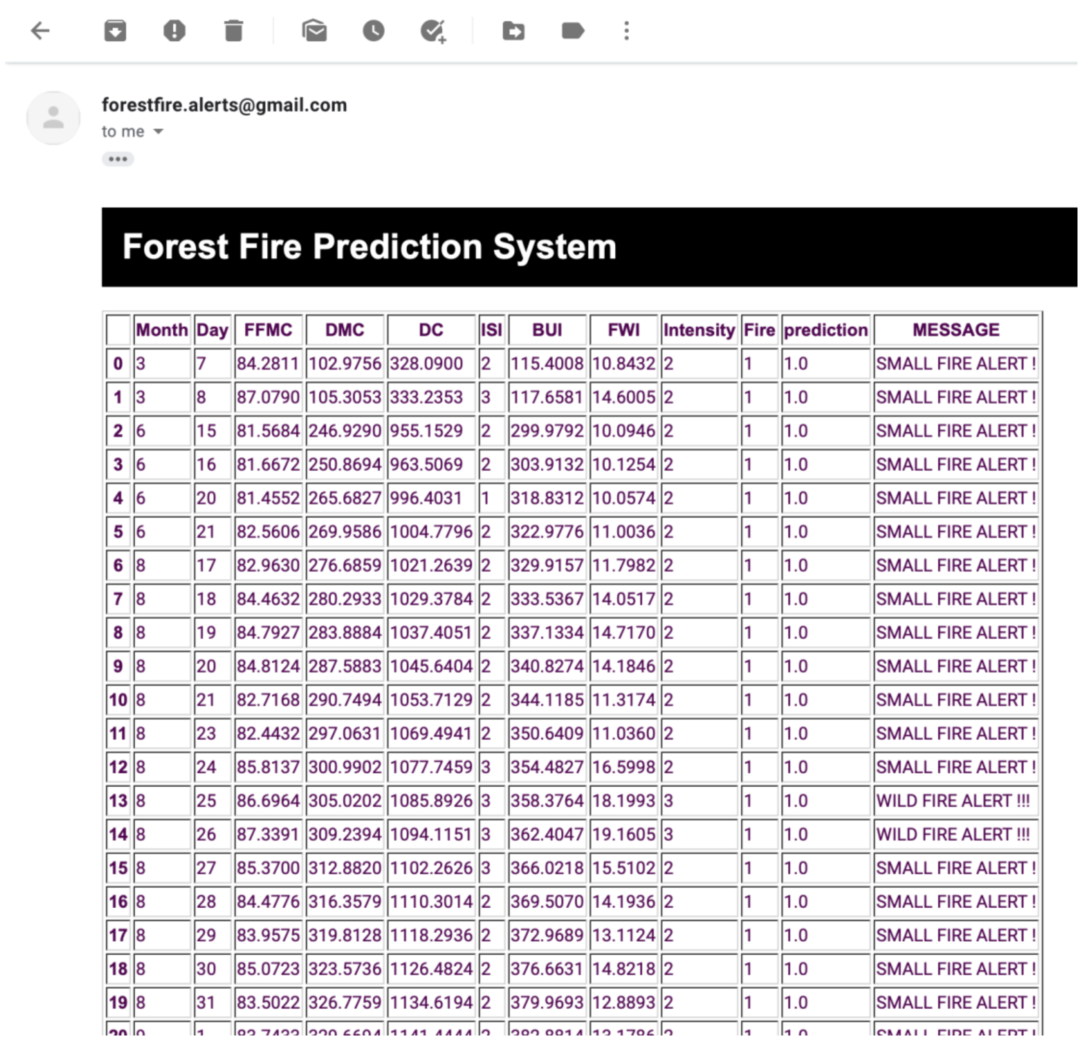


Fig 6.4 (a) and (b): Validation Predictions with Actual Fire Possibilities

- Finally, an automated alert system sends the generated forest fire report via Email as shown in Fig 6.5.



forestfire.alerts@gmail.com
to me

Forest Fire Prediction System

	Month	Day	FFMC	DMC	DC	ISI	BUI	FWI	Intensity	Fire	prediction	MESSAGE
0	3	7	84.2811	102.9756	328.0900	2	115.4008	10.8432	2	1	1.0	SMALL FIRE ALERT !
1	3	8	87.0790	105.3053	333.2353	3	117.6581	14.6005	2	1	1.0	SMALL FIRE ALERT !
2	6	15	81.5684	246.9290	955.1529	2	299.9792	10.0946	2	1	1.0	SMALL FIRE ALERT !
3	6	16	81.6672	250.8694	963.5069	2	303.9132	10.1254	2	1	1.0	SMALL FIRE ALERT !
4	6	20	81.4552	265.6827	996.4031	1	318.8312	10.0574	2	1	1.0	SMALL FIRE ALERT !
5	6	21	82.5606	269.9586	1004.7796	2	322.9776	11.0036	2	1	1.0	SMALL FIRE ALERT !
6	8	17	82.9630	276.6859	1021.2639	2	329.9157	11.7982	2	1	1.0	SMALL FIRE ALERT !
7	8	18	84.4632	280.2933	1029.3784	2	333.5367	14.0517	2	1	1.0	SMALL FIRE ALERT !
8	8	19	84.7927	283.8884	1037.4051	2	337.1334	14.7170	2	1	1.0	SMALL FIRE ALERT !
9	8	20	84.8124	287.5883	1045.6404	2	340.8274	14.1846	2	1	1.0	SMALL FIRE ALERT !
10	8	21	82.7168	290.7494	1053.7129	2	344.1185	11.3174	2	1	1.0	SMALL FIRE ALERT !
11	8	23	82.4432	297.0631	1069.4941	2	350.6409	11.0360	2	1	1.0	SMALL FIRE ALERT !
12	8	24	85.8137	300.9902	1077.7459	3	354.4827	16.5998	2	1	1.0	SMALL FIRE ALERT !
13	8	25	86.6964	305.0202	1085.8926	3	358.3764	18.1993	3	1	1.0	WILD FIRE ALERT !!!
14	8	26	87.3391	309.2394	1094.1151	3	362.4047	19.1605	3	1	1.0	WILD FIRE ALERT !!!
15	8	27	85.3700	312.8820	1102.2626	3	366.0218	15.5102	2	1	1.0	SMALL FIRE ALERT !
16	8	28	84.4776	316.3579	1110.3014	2	369.5070	14.1936	2	1	1.0	SMALL FIRE ALERT !
17	8	29	83.9575	319.8128	1118.2936	2	372.9689	13.1124	2	1	1.0	SMALL FIRE ALERT !
18	8	30	85.0723	323.5736	1126.4824	2	376.6631	14.8218	2	1	1.0	SMALL FIRE ALERT !
19	8	31	83.5022	326.7759	1134.6194	2	379.9693	12.8893	2	1	1.0	SMALL FIRE ALERT !
20	9	1	83.7433	330.6694	1141.4444	2	382.8814	13.1786	2	1	1.0	SMALL FIRE ALERT !

Fig 6.5: Forest Fire Prediction Report sent as Alert

CHAPTER 7

TESTING

7.1 Performance of the model

The performance of the two algorithms, Linear SVM and Parallel SVM are tested on the basis of four parameters which are Accuracy, Precision, Recall and Execution time.

- **Accuracy:**

Accuracy is the number of correctly predicted data points out of all the data points. More formally, it gives us an idea about how well our model predicted the occurrence of forest fires.

Accuracy = (True Positive + True Negative)/ (True Positive + False Positive + True Negative + False Negative)

- **Precision:**

Precision means the percentage of your results which are relevant i.e. ratio of all correct positive classifications to the total no. of positive classifications.

Precision = True Positive/ (True Positive + False Positive)

For our model, it tells us when the model predicted TRUE i.e., the forest fire will happen and how often it was right.

Recall:

Recall refers to the percentage of total relevant results correctly classified i.e., this is the case when in reality, fire has actually happened and how often did our model get it right.

Recall = True Positive/ (True Positive + False Negative)

- **Execution time:**

The time taken by the model to make predictions is considered as the execution time for the model. We use the timer () function to calculate the time taken to predict.

Observation:

The performance of the system was tested for different datasets using LSVM and PSVM. It was observed that PSVM was more efficient and had better accuracy compared to LSVM. The results were stable for PSVM compared to LSVM. Therefore we conclude that the predictions made by PSVM are more reliable to predict forest fires. Linear SVM has lower accuracy when the dataset is too huge and becomes less reliable while parallel SVM filters the most optimal support vectors and gives a more reliable model to predict forest fires. The alerts generated are based on the Parallel SVM model.

Code for LSVM Performance

```
from sklearn.metrics import classification_report, confusion_matrix

lsvmaccuracy = metrics.accuracy_score(expected,predicted)

lsvmprecision=metrics.precision_score(expected, predicted)
lsvmrecall=metrics.recall_score(expected, predicted)

print(lsvmaccuracy)

print(lsvmprecision)

print(lsvmrecall)

print(end-start)
```

Code for PSVM Performance

```
from sklearn.metrics import classification_report, confusion_matrix

print(metrics.precision_score(l, y_pred))

print(metrics.recall_score(l, y_pred))

print(end1-start1)
```

7.2 Comparison between Linear SVM and Parallel SVM

Parameters	Linear SVM	Parallel SVM
Accuracy	0.94	1.0
Recall	0.93	1.0
Precision	1.0	1.0
Execution time(in s)	0.0006	0.051

Table 7.1: Comparison between Linear SVM and Parallel SVM

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

Forest fires may happen because of many causes, accurate predictions are our target. Using weather data and analysing the forest weather indices for training our model is one of the best methods. Our work shows that the easiest way to predict forest fires is by using Support Vector Machines. For small fires, which ultimately trigger bigger fires, there are more reliable tests. But support for large data sets on Support vector machines would have several support vectors to minimize accuracy. We need accurate and reliable data. It is critical. Via parallel computation, we can boost the support vectors. We conclude that by modifying the algorithm and using parallel calculation, performance and calculation time are improved. There is however, a higher memory requirement and an increase in computational time. For this, we make use of the Apache Spark framework.

For implementing our model we have used the django framework for our user interface and integrated the prediction algorithm as the django model. This prediction model makes use of pyspark which is the Python API written in python to support Apache Spark. Apache Spark is a distributed framework that can handle Big Data analysis. Spark is basically a computational engine that works with huge sets of data by processing them in parallel and batch systems. Using these modules we have implemented parallel SVM that uses the weather data and uses parallel computing to predict forest fires. This way we want to make our model more efficient and reliable. We have an additional alerting system that can be used to alert a specific department by just uploading the weather data of a particular station. This way we can help the managing forest fires before it destroys the whole forest. This makes prevention easier by predicting forest fires easily.

8.2 Future scope

Forest fires cause deforestation and land burning. Forest fires are considered socially and economically unwelcome as they burn large amounts of land and may require some time to recover it. Thus it is important to prevent forest fires. Our model is used to predict forest fires. It has an accuracy of 0.99. Our model can be improved by making it live data processing, on site predictions, making it sensor based etc. In the future we hope to improve the speed and accuracy of the working model.

REFERENCES

[1] Predicting forest fires using supervised and ensemble machine learning algorithms by R. Rishickesh, A. Shahina, A. Nayeemulla Khan

[2] Assessing the suitability of soft computing approaches for forest fire prediction by Samaher Al_Janabi a, Ibrahim Al_Shourbaji b, Mahdi A. Salman a

[3]Learning to predict forest fires with different data mining techniques by Daniela Stojanova, Panče Panov, Andrej Kobler, Sašo Džeroski, Katerina Taškova

[4] Various approaches in Forest Fire Detection by Parul Mohindru, Vikshant Khanna, Rajdeep Singh

[5] A Data Mining Approach to Predict Forest Fires using Meteorological Data by Paulo Cortez and An´ıbal Morais

[6] Study on parallel SVM based on Mapreduce by Zhanquan Sun1 , Geoffrey Fox

[7] Bisquert M., Caselles E., Sánchez J.M., Caselles V. "Application of artificial neural networks and logistic regression to the prediction of forest fire danger in Galicia using MODIS data",July 2012 International Journal of Wildland Fire 21(8):1025-1029 DOI: 10.1071/WF11105

[8] Satir, Onur, Suha Berberoglu, and Cenk Donmez. "Mapping regional forest fire probability using artificial neural network models in a Mediterranean forest ecosystem." Geomatics, Natural Hazards and Risk 7.5 (2016): 1645-1658.

[9] K V Suresh Babu, Thesis on "Developing Forest Fire Danger Index Using Geo-Spatial Techniques", IIT Hyderabad Date: 2019-03-09. Report no: IIIT/TH/2019/21.

[10] Zhu, Kaihua, Hao Wang, Hongjie Bai, Jian Li, Zhihuan Qiu, Hang Cui, and Edward Y. Chang. "Parallelizing support vector machines on distributed computers." In Advances in Neural Information Processing Systems, pp. 257-264. 2008.

[11] Guo, W., Alham, N. K., Liu, Y., Li, M., & Qi, M. (2016). A resource aware MapReduce based parallel SVM for large scale image classifications. *Neural Processing Letters*, *44*(1), 161-184.

[12] "Pandas Dataframe: Plot Examples with Matplotlib and Pyplot" , https://queirozf.com/entries/pandas-dataframe-plot-examples-with-matplotlib-pyplot

[13] "Apply CSS class to Pandas DataFrame using to_html", https://stackoverflow.com/questions/50807744/apply-css-class-to-pandas-dataframe-using-to-html/50939211

[14] "Machine Learning with Text in PySpark ", https://datascience-enthusiast.com/Python/PySpark_ML_with_Text_part1.html

[15] "django spark - event sourcing and handling" , https://pypi.org/project/django-spark/

[16] "Interpreting The Canadian Forest Fire Weather Index (Fwi) System" ,http://www.dnr.state.mi.us/WWW/FMD/WEATHER/Reference/FWI_Background.pdf

[17] "Identifying the Threshold of Dominant Controls on Fire Spread in a Boreal Forest Landscape of Northeast China",

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0055618

[18] “Classification and regression” , https://spark.apache.org/docs/2.2.0/ml-classification-regression.html#linear-support-vector-machine

[19] “How to extract weather data from Google in Python” ,

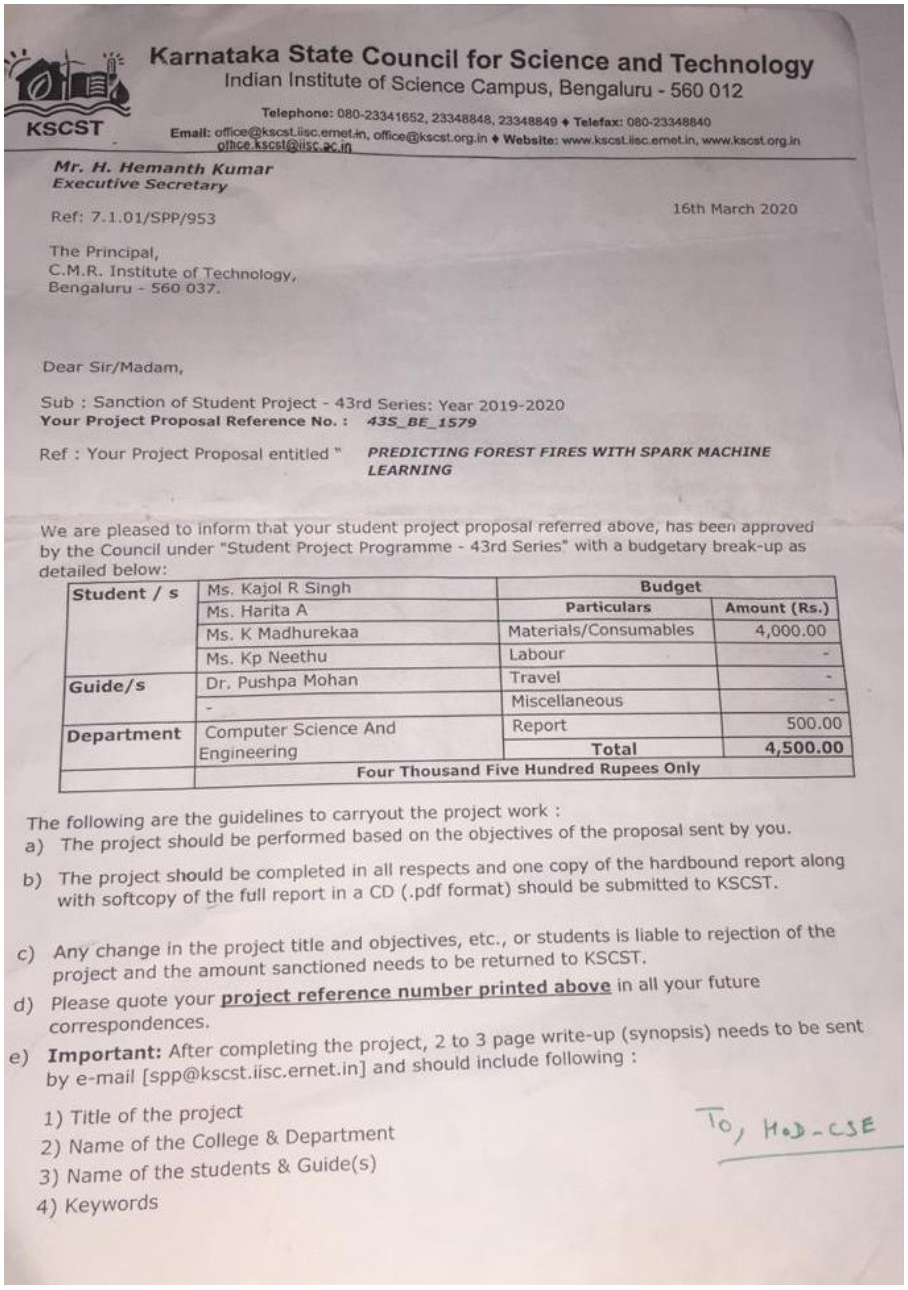
https://www.thepythoncode.com/article/extract-weather-data-python

[20] “Case studies in fire safety” ,

https://www.sciencedirect.com/science/article/pii/S2214398X16300127

APPENDIX- A

Our Project was selected for **KARNATAKA STATE COUNCIL FOR SCIENCE AND TECHNOLOGY.**



Karnataka State Council for Science and Technology
Indian Institute of Science Campus, Bengaluru - 560 012

KSCST

Telephone: 080-23341652, 23348848, 23348849 ♦ **Telefax:** 080-23348840
Email: office@kscst.iisc.ernet.in, office@kscst.org.in ♦ **Website:** www.kscst.iisc.ernet.in, www.kscst.org.in
office.kscst@iisc.ac.in

Mr. H. Hemanth Kumar
Executive Secretary

Ref: 7.1.01/SPP/953

16th March 2020

The Principal,
C.M.R. Institute of Technology,
Bengaluru - 560 037.

Dear Sir/Madam,

Sub : Sanction of Student Project - 43rd Series: Year 2019-2020
Your Project Proposal Reference No. : ***43S_BE_1579***

Ref : Your Project Proposal entitled " ***PREDICTING FOREST FIRES WITH SPARK MACHINE LEARNING***

We are pleased to inform that your student project proposal referred above, has been approved by the Council under "Student Project Programme - 43rd Series" with a budgetary break-up as detailed below:

		Budget	
Student / s	Ms. Kajol R Singh	**Particulars**	**Amount (Rs.)**
	Ms. Harita A	Materials/Consumables	4,000.00
	Ms. K Madhurekaa	Labour	-
	Ms. Kp Neethu	Travel	-
Guide/s	Dr. Pushpa Mohan	Miscellaneous	-
	-	Report	500.00
Department	Computer Science And Engineering	**Total**	**4,500.00**
	Four Thousand Five Hundred Rupees Only		

The following are the guidelines to carryout the project work :

a) The project should be performed based on the objectives of the proposal sent by you.

b) The project should be completed in all respects and one copy of the hardbound report along with softcopy of the full report in a CD (.pdf format) should be submitted to KSCST.

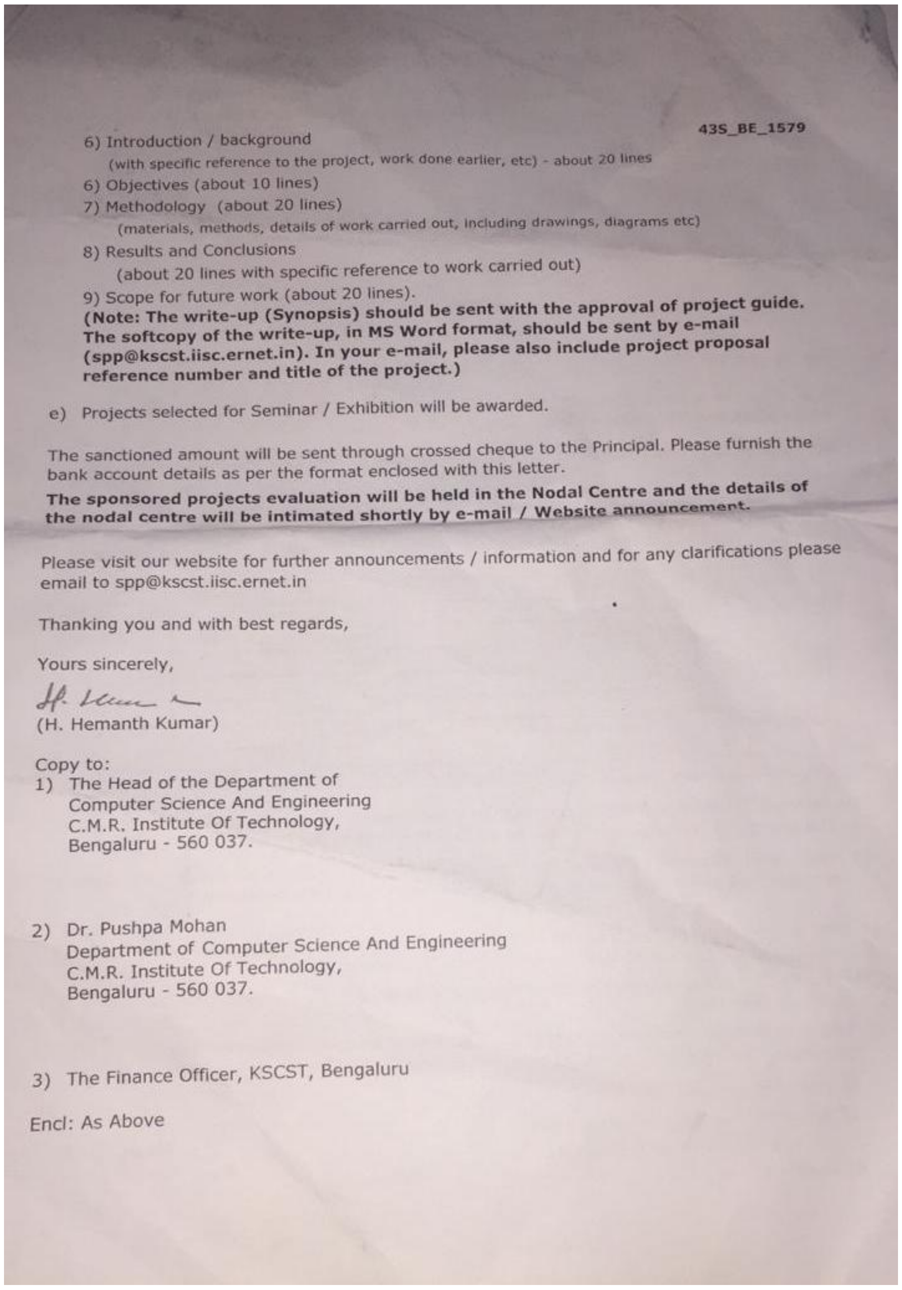
c) Any change in the project title and objectives, etc., or students is liable to rejection of the project and the amount sanctioned needs to be returned to KSCST.

d) Please quote your **project reference number printed above** in all your future correspondences.

e) **Important:** After completing the project, 2 to 3 page write-up (synopsis) needs to be sent by e-mail [spp@kscst.iisc.ernet.in] and should include following :

1) Title of the project
2) Name of the College & Department
3) Name of the students & Guide(s)
4) Keywords

To, HoD-CSE



43S_BE_1579

6) Introduction / background
(with specific reference to the project, work done earlier, etc) - about 20 lines
6) Objectives (about 10 lines)
7) Methodology (about 20 lines)
(materials, methods, details of work carried out, including drawings, diagrams etc)
8) Results and Conclusions
(about 20 lines with specific reference to work carried out)
9) Scope for future work (about 20 lines).
(Note: The write-up (Synopsis) should be sent with the approval of project guide. The softcopy of the write-up, in MS Word format, should be sent by e-mail (spp@kscst.iisc.ernet.in). In your e-mail, please also include project proposal reference number and title of the project.)

e) Projects selected for Seminar / Exhibition will be awarded.

The sanctioned amount will be sent through crossed cheque to the Principal. Please furnish the bank account details as per the format enclosed with this letter.
The sponsored projects evaluation will be held in the Nodal Centre and the details of the nodal centre will be intimated shortly by e-mail / Website announcement.

Please visit our website for further announcements / information and for any clarifications please email to spp@kscst.iisc.ernet.in

Thanking you and with best regards,

Yours sincerely,

[illegible]
(H. Hemanth Kumar)

Copy to:
1) The Head of the Department of
Computer Science And Engineering
C.M.R. Institute Of Technology,
Bengaluru - 560 037.

2) Dr. Pushpa Mohan
Department of Computer Science And Engineering
C.M.R. Institute Of Technology,
Bengaluru - 560 037.

3) The Finance Officer, KSCST, Bengaluru

Encl: As Above