Evaluating Supervised Machine Learning Techniques to Predict Forest Fires in Algeria

Dataset: Algerian forest fires

Divya Nandlal Sahetya, Vibhav Chitalia,
sahetya@usc.edu, vchitali@usc.edu

May 4, 2022

1 Abstract

In this project we work on evaluating different aspects of Machine Learning system design in order to solve classification task. We are using Algerian Forest Fires dataset [3] that has been further modified (reduced features, and samples) to make this problem more challenging. We approach this project with an opportunity to try several techniques involved in designing an ML system like, exploratory data analysis, feature engineering, feature scaling (standardization and normalization), feature selection and reduction, model selection by hyper-parameter tuning and cross-validation (also taking care when data has time-dependency). All of these we try on several supervised learning models while comparing it with the baseline models based on statistics. We explore effects of different feature scaling (StandardScaler vs MinMaxScaler), feature reduction (PCA vs Pearson correlation based), cross-validation (KFold vs TimeSeriesSplit) and other related aspects of ML pipeline in this project. We found that by trying simple approaches on each stage of ML pipeline significantly boosts the model performance as compared to baseline. Overall the experiments show that the Logistic Regression based classifier and SVM outperforms advanced techniques such as Decision Tree and Multi-layer Perceptron on the given dataset. Main reason being the complexity (d.o.f) of the models being larger or comparable to the dataset samples (leading to overfitting). Overall this project focuses on following the Occam's razor by trying simple techniques and make the model generalize well on test data.

2 Introduction

2.1 Problem Assessment and Goals

Forest fires are a major concern all over the world; each year, millions of hectares are lost [4]. Algeria is one of the countries affected by forest fires. With the use of Algerian dataset, we want to predict if there will be a fire in a future date.

This dataset [3] contains weather data from 2 regions in Algeria over the period of 3 months and the fire prediction is based on weather data collected from the regions. The provided dataset is already divided into train and test. The training dataset contains data points dated between June and including August from the Date feature and the test dataset contains the data points of September from the Date feature.

The dataset contains 11 features, all real valued with no missing values. They are:

- Date: Date when the data point was collected (day/month/year format).
- Temperature: Max. temperature of the day in degrees Celsius
- RH: Humidity

- Ws: Wind speed in km/h
- Rain: rain level in mm
- FFMC*: Fine Fuel Moisture Code
- DMC*: Duff Moisture Code
- DC*: Drought Code
- ISI*: Initial Spread Index
- BUI*: Buildup Index
- Classes: The label of the dataset (1: there was a fire, 0: not a fire)

The features can be broadly divided into date, weather factors, fuel moisture codes, fire behaviour indices.

- Weather factors: Temperature, Humidity, Wind and Rain.
- Fuel Moisture Codes: FFMC, DMC and DC
  - Fine Fuel Moisture Code (FFMC) denotes the moisture content surface litter and influences ignition and fire spread. It depends on temperature, rain, relative humidity, wind.
  - The Duff Moisture Code (DMC) is a numeric rating of the average moisture content of loosely compacted organic layers of moderate depth. It depends on temperature, rain, relative humidity.
- Fire Behaviour Indices: ISI, BUI
  - Initial Spread Index (ISI) is a score that correlates with fire velocity spread. It depends on FFMC, wind.
  - Buildup Index (BUI) is a numeric rating of the total amount of fuel available for combustion. It depends on DMC, DC.

We tried different data visualization methods to understand more about how each parameter relates to the occurence of Fire. Some of them includes Class based feature importance, Class distribution, Time Series (trend) analysis of each feature which are displayed in the figure 1.

In this project, we intend to investigate the approaches mentioned below, analyze them in detail, and compare their performances.

2.2 Approaches to be investigated

1. Trivial System
2. Nearest mean Classifier
3. Logistic Regression Classifier
4. K Neighbour Classifier
5. Support Vector Machine
6. Perceptron Classifier
7. Naïve Bayes Classifier

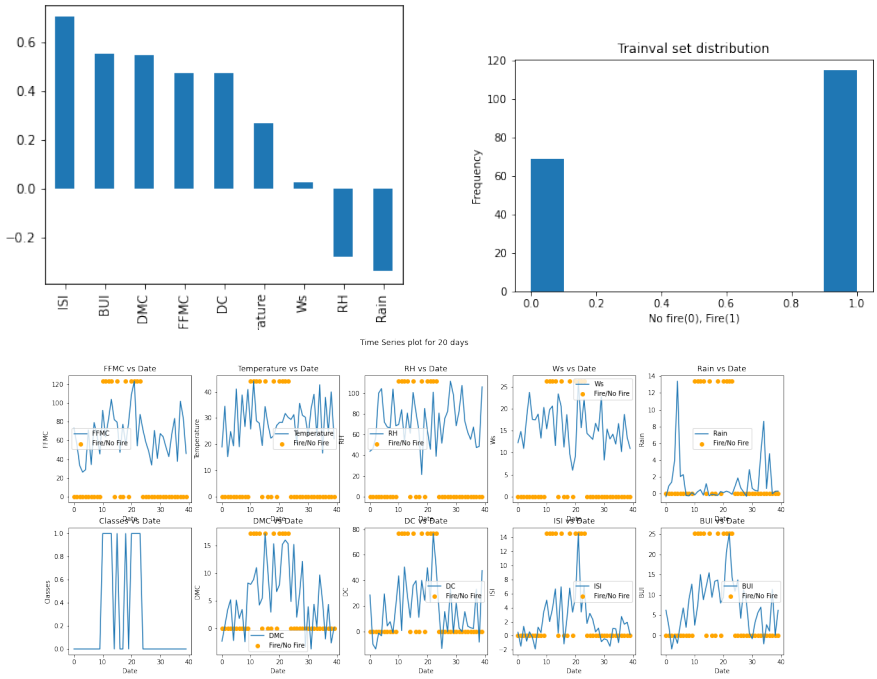


Figure 1
Class based feature importance, Class distribution, Time Series (trend) analysis of each feature

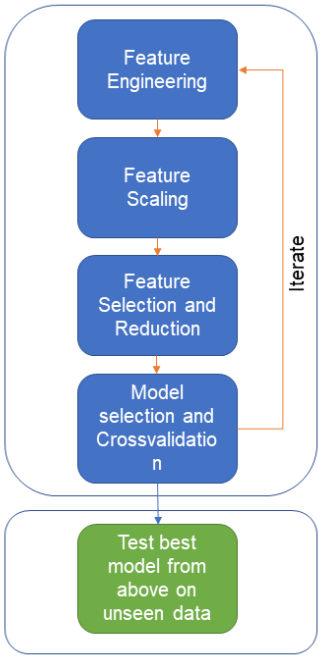


Figure 2
Overall ML pipeline used in the project

3 Approach and Implementation

3.1 Dataset Usage

The train data set consists of 184 data points and the test data set contains 60 data points.

Validation data is generated from train data set after the pre-processing step which is explained in Section 3.2 that includes Standardization/Normalization block and Section 3.3 that covers the feature engineering block using Cross Validation. The details of Cross Validation is covered in Section 3.5 which covers the training, classification, and model selection workflow. Overall process followed in this project is shown in the Figure 2

Train data set is passed to the Feature Engineering (Section 3.3) block where new features are generated. The orginal features in addition to the new added features are then passed to the preprocessing block where the data is standardized and/or normalized. The scaled data is then passed to Feature Selection/Feature Reduction (Section 3.4) block wherein the expanded feature space is reduced based on different parameters.

Once the input features are ready, it is then passed to the model selection and cross validation block, where in each classifier is trained with set of different hyper parameters on different cross-validation set and the best combination of the parameters performing is selected based on the (average) cross-validation accuracy. More details about the same can be found in algorithm 1 in Section 3.5

The test data set was used in the end after the model is selected and best hyper parameters are available

3.2 Preprocessing

Feature scaling is a technique used for standardizing the dataset's independent variables within a given range. In feature scaling, we place all of our variables in the same range and scale so that none of them dominate the others. We try 3 different scaling approaches in this project only StandardScaler, MinMaxScaler, StandardScaler + normalize (sample/row wise)

Classification Algorithm	Preprocessing
Distance-Based Algorithms	Feature Scaling is required as the range of features has the greatest impact on distance algorithms like KNN, K-means, and SVM. This is because they analyze distances between data points to estimate its resemblance.
Tree-Based Algorithms	Feature Scaling is not required as a decision tree is only splitting a node based on a single feature. This split on a feature is not influenced by other features.

Table 1
Feature Scaling

There are three ways to perform feature scaling in machine learning:

3.2.1 Standard Scaler

Standardization is a feature transformation that involves subtracting from the mean and dividing by the standard deviation [1]. This is commonly referred to as the Z-score. This is known as the Z-score. Scikit-Learn provides a transformer called StandardScaler for standardization. The StandardScaler assumes that the data is normally distributed within each feature and will scale them such that

the distribution is now centred around 0, with a standard deviation of 1. The mean and standard deviation are calculated for the feature and then the feature is scaled based on:

$$z = \frac{x - mean(x)}{stdev(x)} \tag{1}$$

3.2.2 Min Max Scaler

MinMax Scaler is used to transform features to be on a similar scale. The scales ranges from [0, 1] or [-1, 1].

$$z = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2}$$

where, x_{max} and x_{min} are the maximum and the minimum values of the feature respectively.

3.2.3 Normalizer

Normalize samples individually to unit norm. By default, L2 normalization is applied. Therefore, the values in a row have a unit norm. Alternatively, L1 normalization can be also applied.

$$x' = \frac{x}{size(x)} \tag{3}$$

where, size(x) is $||x||_1$ for l1 norm, $||x||_2$ for l2 norm, and $||x||_\infty$ for max norm,

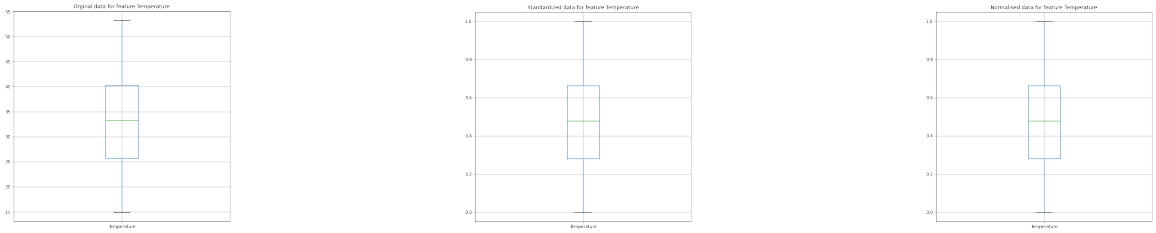


Figure 3
Comparing unscaled, normalized and standardized data

From Figure 3, we can observe that numerical features are centered on the mean with a unit standard deviation after Standardization and all the features have a minimum value of 0 and a maximum value of 1 after Normalization.

Classification Algorithm	Preprocessing
Logistic Regression Classifier	Standard Scaler, Min Max Scaler, Standard Scaler and Normalizer
K Neighbour Classifier	Standard Scaler, Min Max Scaler, Standard Scaler and Normalizer
Support Vector Machine	Standard Scaler, Min Max Scaler, Standard Scaler and Normalizer
Perceptron Classifier	Standard Scaler, Min Max Scaler, Standard Scaler and Normalizer
Naïve Bayes Classifier	Standard Scaler, Min Max Scaler, Standard Scaler and Normalizer

Table 2
Feature Scaling

3.3 Feature Engineering

Feature engineering, also known as feature extraction, is the process of extracting features from raw data. When compared to sending merely raw data to a machine learning process, the motivation is to leverage these extra features to improve the quality of results from a machine learning process.

New features is generated on the whole dataset, that is training and test dataset based on the existing features. The new features reflects statistics of the original ones such as: average, min, max and median of temperature, rain, humidity and wind of the last 2 days. That is, we are trying to predict fires in future dates using past dates when collecting the statistics.

The dataset consists of two readings on the same date from the two regions. In such a scenario, we are considering the window of past two days (4 points) for the first statistic reading of the date. And for the second statistic reading we are considering the past two days data and the first reading of the present data (5 points).

For example, if we want to find the max temperature of the past 2 days of August 15, we will assign the max temperature of the dates between 13-14 August for the first reading of the feature of August 15 and the max temperature of the dates between 13-14 August and the first reading of 15 August.

The below Table 3 lists out the input and the new expanded features of the train and test dataset.

Dataset	Initial Features	Count	Expanded Features	Count
Train dataset, Test datset	Date, Temperature, RH, Ws, Rain, FFMC, DMC, DC, ISI, BUI	10	Date, Temperature, RH, Ws, Rain, FFMC, DMC, DC, ISI, BUI, max_Temperature, min_Temperature, mean_Temperature, median_Temperature, max_RH, min_RH, mean_RH, median_RH, max_Ws, min_Ws, mean_Ws, median_Ws, max_Rain, min_Rain, mean_Rain and median_Rain	26

Table 3
Feature Engineering

3.4 Feature dimensionality adjustment

Feature selection is the process of selecting a subset of relevant features whereas dimensionality reduction does not have to select a subset instead new synthetic features can be constructed using the linear combination of the originak ones and discarding the least important ones. For example, Principal Component Analysis (PCA).

3.4.1 Pearson Correlation:

A Pearson correlation is a value between -1 and 1 that shows how closely two variables are related linearly.

- A value closer to 0: weaker correlation
- A value closer to 1: stronger positive correlation
- A value closer to -1: stronger negative correlation
- 0: no correlation

Through correlation, each variable can be predicted from the other. The variables should be correlated with the target but should be uncorrelated among themselves. If two variables are correlated, one can be predicted from the other. Similarly, if two features are highly correlated, then the model only needs one of them, as the second one would not add any additional information.

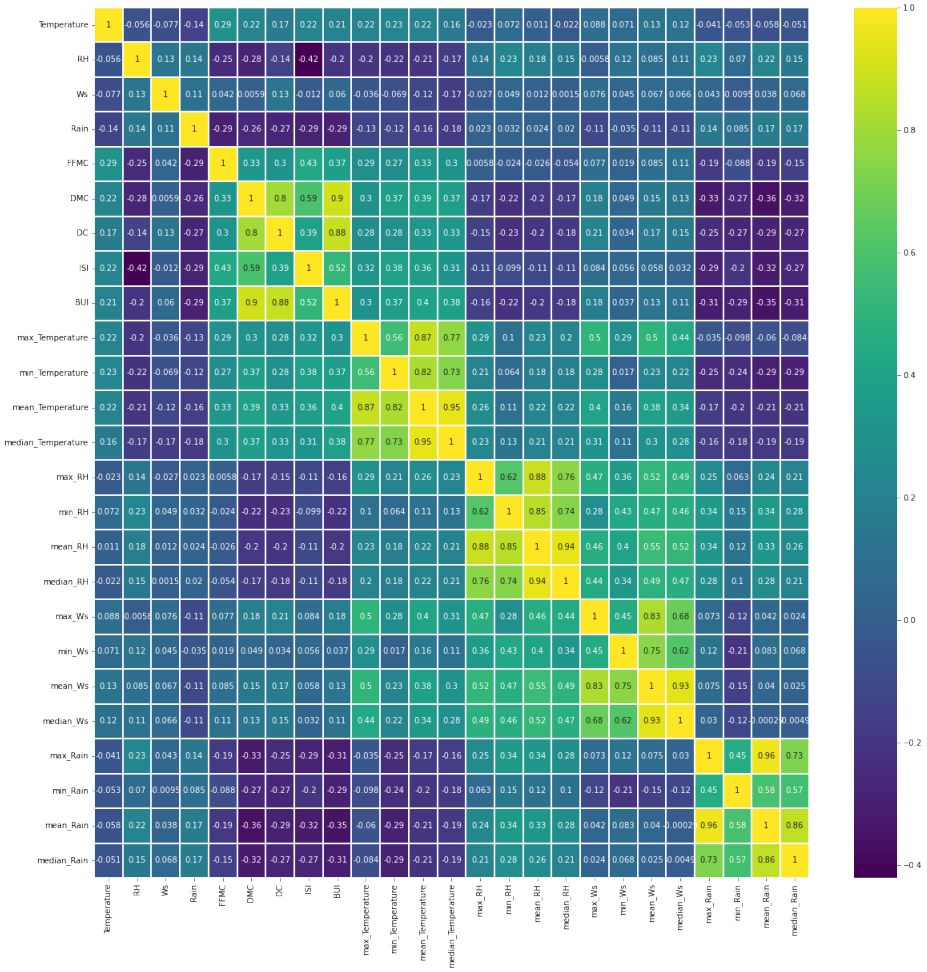


Figure 4
HeatMap of train data set with feature engineering

3.4.2 Principal Component Analysis (PCA):

PCA is a is an unsupervised algorithm and dimensionality-reduction method that finds set of uncorrelated features in the lower-dimensional space from a set of correlated features in the high-dimensional input space. These uncorrelated features are also called principal components which are perpendicular to each other and hence PCA is is an orthogonal linear transformation. The first component interprets the maximum variance from the original data.

From Figure 4, we can observe that the features are highly correlated and so PCA would help in reducing the set of correlated features to uncorrelated.

3.5 Training, Classification, and Model Selection

For model selection we begin with preprocessed (Feature engineered, Standardize and/or Normalized and Feature Selection or Reduction applied) data in the start. We have a HyperParameterTuning function that takes list of models hyperparameters and trainval data and returns a dictionary of best parameters for each model as its key and hyperparameter and avg accuracy on crossvalidation set. For CrossValidation strategy we have tried KFold and TimeSeriesSplit available in sklearn and also tried some similar implementation of our own. We used split size of 5 for the CrossValidation with a gap of 4 when possible to avoid information leak (however we observe that the as the KFold without gap is working well on test set possibly because standardization and PCA have a regularizing effect).

Algorithm 1 Algorithm (pseudocode) for Model selection and hyperparameter tuning

Require: Trainval and Test set is available with below preprocessing:

- Feature Engineering (expansion),
- Standardization and/or Normalization,
- Feature reduction (PCA, Pearson's correlation)

```
function HyperParameterTuning(model_list, hyperparameters, trainval_data, trainval_labels)
    best_params = {}
    for model in model_list do
        params_vs_score = {}
        for hp in hyperparameters do
            scores = []
            for train, val from CrossValidation do
                model.fit(train)
                curr_score = model.score(val)
                scores.append(curr_score)
            end for
            params_vs_score[hp] = Average(scores)
        end for
        best_params[model] = hp where params_vs_score has max(average accuracy)
    end for
end function
```

3.5.1 Trivial System

A system that outputs class assignments S1 or S2 at random with probability N_1/N and N_2/N, respectively; N_i' is the population of data points with class label Si, and N is the total population of data points, based on the training set. The probability summarizes the likelihood of an data belonging to each class label.

When an data reading has the class label **not a fire** (0), then the probability of the class labels 0 and 1 will be 1 and 0 respectively and when an data reading has the class label **fire** (1), then the

probability of class labels 0 and 1 will be 0 and 1 respectively.

- Class=0 (not a fire): P(class=0) = 1, P(class=1) = 0
- Class=1 (fire): P(class=0) = 0, P(class=1) = 1

This has been implemented using **Bernoulli distribution**:

- Class=0 (not a fire): P(class=1) = 0
- Class=1 (fire): P(class=1) = 1

Therefore, probability metrics summarizes how well the predicted distribution of class membership matches the known class probability distribution. From the below Table 4 we can observe that the test dataset predicted that there will be a fire with accuracy 48%. Figure 5 displays the confusion matrix of the trivial system where the number of True Positive is 12, True Negative is 17, False Positive is 25 and False Negative is 6.

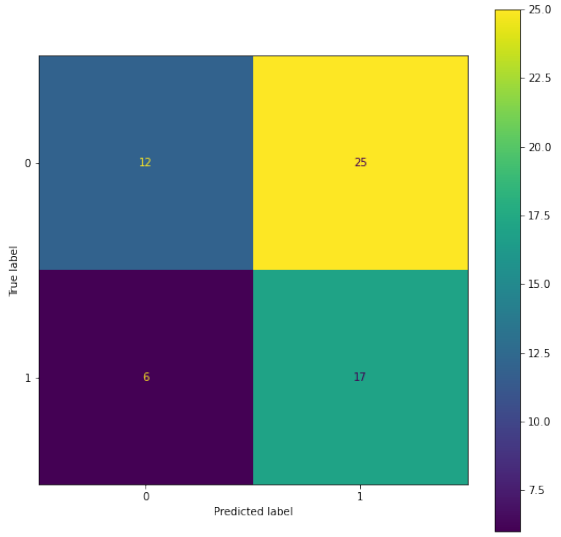


Figure 5
Trivial System Confusion Matrix

Class	Precision	Recall	F-score	Support
0	0.67	0.32	0.44	37
1	0.40	0.74	0.52	23
accuracy			0.48	60
macro avg	0.54	0.53	0.48	60
weighted avg	0.57	0.48	0.47	60

Table 4
Classification Report for Trivial System

- Bernoulli distribution was imported from scipy library [5].
- The classifier was coded using object-oriented approach that has api similar to sklearn's estimator (it has fit, predict and score methods).

3.5.2 Baseline system: Nearest Means Classifier

The Nearest Means classifier or Nearest Centroid classifier works on a simple principle, given a data point, the classifier assigns the class of the training sample whose mean is closest to it. The following are the steps used to classify input data using Nearest Means Classifier:

- During training, the mean for each target class is computed.
- After the training phase, for any given data point, say 'X'. The distances between the point X and each class' mean is calculated.
- The minimum distance is picked out of all the calculated distances. The mean to which the given point's distance is minimum, it's class is assigned to the given point.

From the Table 5 we can observe that the test dataset predicted that there will be a fire with accuracy 78%. Figure 6 displays the confusion matrix of the trivial system where the number of True Positive is 36, True Negative is 11, False Positive is 1 and False Negative is 12.

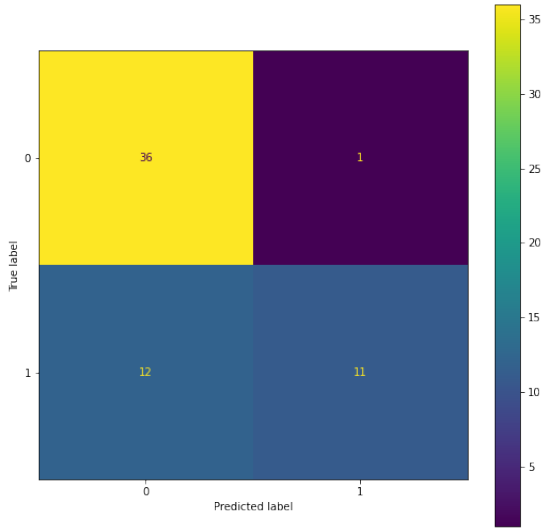


Figure 6
Nearest Means Classifier Confusion Matrix

Class	Precision	Recall	F-score	Support
0	0.75	0.97	0.85	37
1	0.92	0.48	0.63	23
accuracy			0.78	60
macro avg	0.83	0.73	0.74	60
weighted avg	0.81	0.78	0.76	60

Table 5
Classification Report for Nearest Means Classifier

- The Nearest Means Classifier was coded using object-oriented approach that has api similar to sklearn's estimator (it has fit, predict and score methods).

3.5.3 Logistic Regression Classifier

Logistic Regression can be used as a classification algorithm when the value of the target variable is categorical in nature, that is when it has a binary output or belongs to one class or another, or is either a 0 or 1.

Sigmoid function is used to map the predicted values to probabilities as it maps real value into another value between 0 to 1.

- Using Cross Validation, the Logistic Regression Classifier is tried with various combination of hyper parameters to give the best accuracy.
- The **Parameters** are: Penalty (norm of the penalty): ['l1','l2'], C (Inverse of Regularization strength): [1,1.5,2].
- The best hyper parameter if this classifier was found using k-fold cross validation and the hyper parameter values are, penalty: **l2**, C: **1.5** as per Table 6.

Scaling	Feature Reduction	features count	C	accuracy*
Standard	-	25	1	0.89
Standard & Normalizer	-	25	1.5	**0.90**
MinMax	-	25	1.5	0.89
Standard	PCA	5	2.0	0.89
Standard	PCA	4	1.0	0.85
Standard	Pearson's Correlation	11	1.0	0.84

Table 6
Hyper parameter tuning for Logistic Regression Classifier
* Accuracy is average accuracy on all the k (5) folds.

- Logistic Regression Classifier was implemented using sklearn function LogisticRegression [6].

3.5.4 K Neighbors Classifier

In k-NN classification, the output is decided based on the class votes of its neighbors. A datapoint is classified by a plurality vote of its neighbors, and it is assigned to the class most common among its k nearest neighbors. The steps involved in KNN:

- The train data set is loaded and k is initialised.
- For each sample in data, the euclidean distance between each of its neighbours and current sample is calculated and stored.
- The stored list of distances is sorted in ascending order and the first k entries, labels is fetched and the mode is obtained. This final label is used to classify the given sample.
- Using Cross Validation, the K Neighbors Classifier is tried with various combination of hyper parameters to give the best accuracy.
- **Parameters**: n_neighbors (Number of neighbors): [3,4,5,6], weights (Weight function used in prediction): ['uniform', 'distance'] was considered.
- The best hyper parameter if this classifier was found using k-fold cross validation and the hyper parameter values are, n_neighbors is **4**, weights is **distance** as per Table 7.

Scaling	Feature Reduction	Features count	n_neighbours	weight	Accuracy*
Standard	-	25	5	uniform	0.82
Standard & Sample Normalizer	-	25	6	uniform	0.84
MinMax		25	4	uniform	0.84
Standard	PCA	5	4	distance	**0.86**
Standard	PCA	4	5	uniform	0.84
Standard	Pearson's Correlation	11	6	uniform	0.77

Table 7
Hyper parameter tuning for K Neighbors Classifier
* Accuracy is average accuracy on all the k (5) folds.

- kNN Classifier was implemented using sklearn function KNeighborsClassifier [7].

3.5.5 Support Vector Classifier (SVC)

The Support Vector Machine classifier has an advantage of it working on a subset of points which makes it memory efficient. With an appropriate kernel function SVM is capable of modeling non-linear decision boundaries. However, when number of features are comparable with number of samples it can lead to overfit and may require regularization.

- Using Cross Validation, the SVC is tried with various combination of hyper parameters to give the best accuracy.
- **Parameters**: kernel (kernel type): ['linear', 'poly', 'rbf'], C (Regularization parameter): [1, 10, 100], gamma(Kernel coefficient for 'rbf', 'poly' and 'sigmoid'): [1e-3,1e-4,auto] was considered.
- The best hyper parameter if this classifier was found using k-fold cross validation and the hyper parameter values are, kernel is **linear**, C is **1**, gamma is **auto** as per Table 8.
- Support Vector Classifier was implemented using sklearn function SVC [8].

Scaling	Feature Reduction	Features count	kernel	C	gamma	Accuracy*
Standard	-	25	rbf	10	0.001	0.88
Standard & Sample Normalizer	-	25	linear	1	0.001	0.89
MinMax		25	linear	1	0.001	0.89
Standard	PCA	5	linear	1	auto	**0.90**
Standard	PCA	4	linear	1	auto	0.86
Standard	Pearson's Correlation	11	linear	10	auto	0.86

Table 8
Hyper parameter tuning for Support Vector Classifier
* Accuracy is average accuracy on all the k (5) folds.

3.5.6 Perceptron

Perceptron is an supervised learning algorithm that iteratively tries to learn the weights (or coefficients) that effectively solves binary classification problem. Since our problem is also one of a binary case, of identifying if forest fire would occur or not. We thought of trying this classifier in our list of models to choose from.

- Using Cross Validation, the Perceptron is tried with various combination of hyper parameters to give the best accuracy.
- **Parameters**: penalty (The penalty (regularization term) to be used): ['l2'], shuffle : true was considered.
- The best hyper parameter if this classifier was found using k-fold cross validation and the hyper parameter values are, penalty is **'l2'** as per Table 9.

Scaling	Feature Reduction	features count	Accuracy*
Standard	-	25	0.85
Standard & Sample Normalizer	-	25	**0.88**
MinMax		25	0.86
Standard	PCA	5	0.74
Standard	PCA	4	0.64
Standard	Pearson's Correlation	11	0.71

Table 9
Hyper parameter tuning for Perceptron
* Accuracy is average accuracy on all the k (5) folds.

- Perceptron was implemented using sklearn's Perceptron estimator [9].

3.5.7 Gaussian Naive Bayes Classifier

When all the other models discussed so far follows a discriminative approach, Gaussian Naive Bayes (or any version of Naive Bayes) classifier follows a generative approach. Bayes theorem underlies at the heart of this algorithm where a an assumption that each feature is independent of the other and posterior is computed using the prior and likelihood. Since some the features are observed to be normal in nature we try Gaussian Naive Bayes approach in this project.

- Using Cross Validation, the Gaussian Naive Bayes is tried with various combination of hyper parameters to give the best accuracy.
- **Parameters**: default parameters was considered.
- The best hyper parameter if this classifier was found using k-fold cross validation and the hyper parameter values are, n_neighbors is **4**, weights is **distance** as per Table 12.

Scaling	Feature Reduction	features count	Accuracy*
Standard	-	25	0.83
Standard & Sample Normalizer	-	25	0.83
MinMax		25	0.83
Standard	PCA	5	0.77
Standard	PCA	4	0.79
Standard	Pearson's Correlation	11	**0.84**

Table 10
Hyper parameter tuning for Gaussian Naive Bayes Classifier
* Accuracy is average accuracy on all the k (5) folds.

- Gaussian Naive Bayes Classifier was implemented using sklearn function GuassianNB [10].

4 Analysis: Comparison of Results, Interpretation

4.1 Logistic Regression Classifier

- Preprocessing : Standard Scaler followed by Normalizer
- Feature Engineering : The total number of features after feature engineering is 25.
- Feature Reduction : None.
- Cross Validation : KFold with k = 5
- Hyper parameters : 'penalty': 'l2', 'C': 1.0

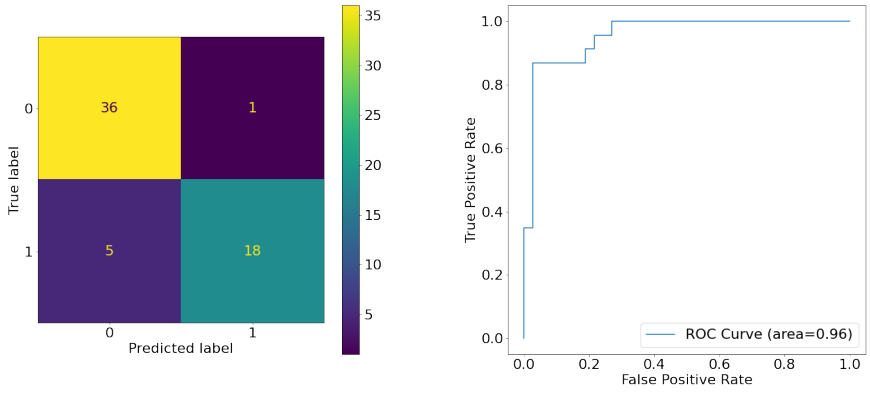


Figure 7
Logistic Regression Classifier Confusion Matrix and ROC curve

Dataset	Trivial System Accuracy	Nearest Means Accuracy	Logistic Regression Classifier Accuracy
Test	0.48	0.78	0.90

Table 11
Accuracy comparison of test data set of classifier with baseline and trivial model

Class	Precision	Recall	F-score	Support
0	0.90	0.95	0.92	37
1	0.90	0.83	0.86	23
accuracy			0.90	60
macro avg	0.90	0.89	0.89	60
weighted avg	0.90	0.90	0.90	60

Table 12
Classification Report for Logistic Regression Classifier

4.2 K Neighbors Classifier

- Preprocessing : Standardization
- Feature Engineering : The total number of features after feature engineering is 25.
- Feature Reduction : PCA with n_components 5.
- Features count reduced to 5 after PCA.
- Cross Validation : KFold with k = 5
- Hyper parameters : 'n_neighbors': 4, 'weights': 'distance'

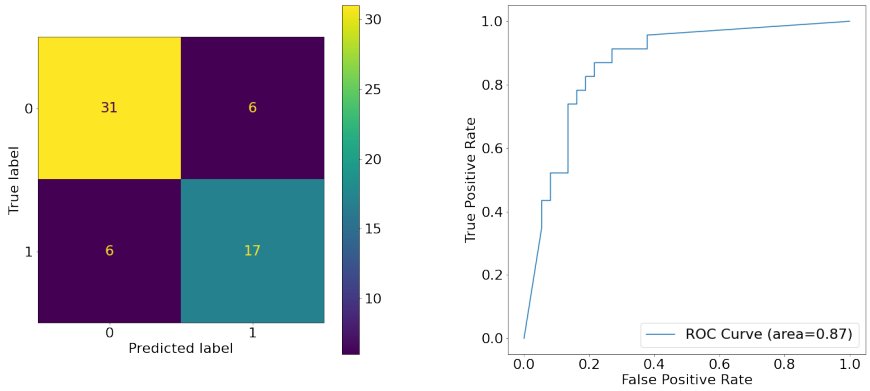


Figure 8
K Neighbors Classifier Confusion Matrix and ROC curve

Class	Precision	Recall	F-score	Support
0	0.84	0.84	0.84	37
1	0.74	0.74	0.74	23
accuracy			0.80	60
macro avg	0.79	0.79	0.79	60
weighted avg	0.80	0.80	0.80	60

Table 13
Classification Report for K Neighbors Classifier

Dataset	Trivial System Accuracy	Nearest Means Accuracy	K Neighbors Classifier Accuracy
Test	0.48	0.78	0.80

Table 14
Accuracy comparison of test data set of classifier with baseline and trivial model

4.3 Support Vector Classifier (SVC)

- Preprocessing : Standard Scaler
- Feature Engineering : The total number of features after feature engineering is 25.
- Feature Reduction : PCA with n_components 5.
- Features count reduced to 5 after PCA.
- Cross Validation : KFold with k = 5
- Hyper parameters : 'kernel': 'linear', 'C': 1, 'gamma': 'auto'

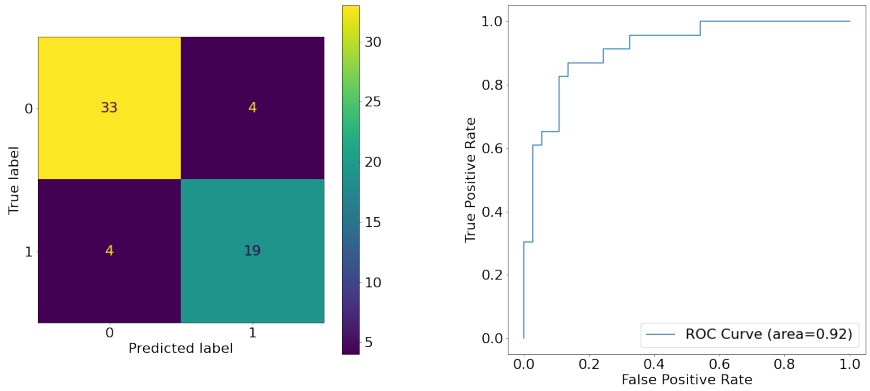


Figure 9
Support Vector Classifier (SVC) Confusion Matrix and ROC curve

Class	Precision	Recall	F-score	Support
0	0.89	0.89	0.89	37
1	0.83	0.83	0.83	23
accuracy			0.87	60
macro avg 0.86	0.86	0.86	60	
weighted avg 0.87	0.87	0.87	60	

Table 15
Classification Report for Support Vector Classifier (SVC)

Dataset	Trivial System Accuracy	Nearest Means Accuracy	SVC Accuracy
Test	0.48	0.78	0.8666666666666667

Table 16
Accuracy comparison of test data set of classifier with baseline and trivial model

4.4 Perceptron

- Preprocessing : Standard Scaler and Normalizer
- Feature Engineering : The total number of features after feature engineering is 25.
- Feature Reduction : None
- Cross Validation : KFold with k = 5
- Hyper parameters : 'penalty': 'l2', 'shuffle': False

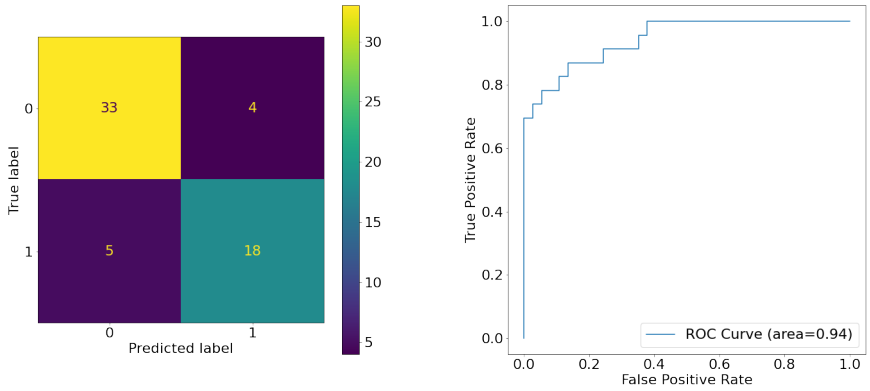


Figure 10
Perceptron Confusion Matrix and ROC Curve

Class	Precision	Recall	F-score	Support
0	0.87	0.89	0.88	37
1	0.82	0.78	0.80	23
accuracy			0.85	60
macro avg	0.84	0.84	0.84	60
weighted avg	0.85	0.85	0.85	60

Table 17
Classification Report for Perceptron

Dataset	Trivial System Accuracy	Nearest Means Accuracy	Perceptron Accuracy
Test	0.48	0.78	0.85

Table 18
Accuracy comparison of test data set of classifier with baseline and trivial model

4.5 Gaussian Naive Bayes Classifier

- Preprocessing : Standard Scaler
- Feature Engineering : The total number of features after feature engineering is 25.
- Feature Reduction : Pearson's Correlation with threshold of 0.5.
- Features count reduced to 11 after Pearson's Correlation.
- Cross Validation : KFold with k = 5
- Hyper parameters : default

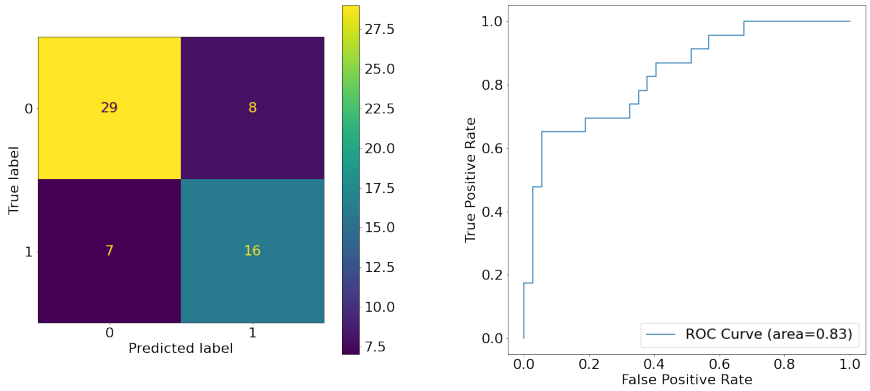


Figure 11
Gaussian Naive Bayes Classifier Confusion Matrix and ROC curve

Class	Precision	Recall	F-score	Support
0	0.81	0.78	0.79	37
1	0.67	0.70	0.68	23
accuracy		0.70	60	
macro avg 0.74	0.74	0.74	60	
weighted avg 0.75	0.75	0.75	60	

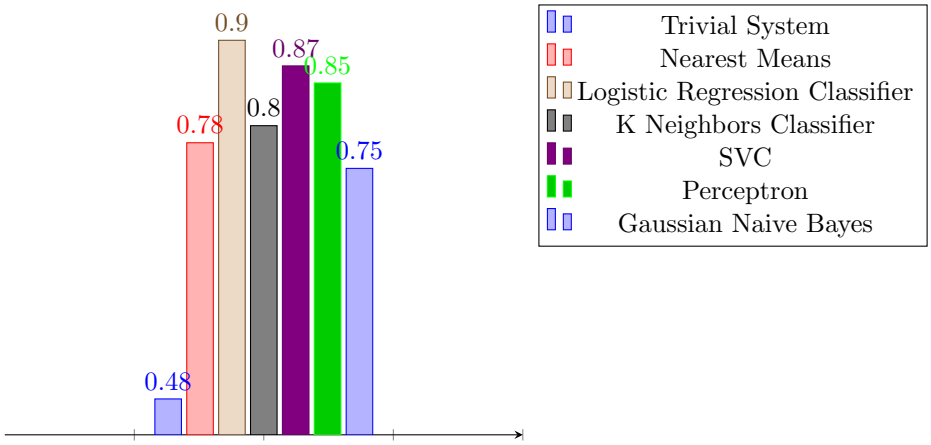
Table 19
Classification Report for Gaussian Naive Bayes Classifier

Dataset	Trivial System Accuracy	Nearest Means Accuracy	Gaussian Naive Bayes Accuracy
Test	0.48	0.78	0.75

Table 20
Accuracy comparison of test data set of classifier with baseline and trivial model

Figure 4.5 shows the comparison of various model and classifier used in this project. We can observe that Logistic Regression Classifier has a accuracy of 90% followed by SVM with an accuracy of 87%.

Accuracy comparison for the approaches implemented



We also tried other models that we have not listed in this report. They are Ridge Regressor, Multilayer Perceptron, Decision Tree Classifier, GradientBoosting methods on the same dataset. We used sklearn's available api for their implementation. Results of it can be accessed at [11]

5 Libraries

The libraries used for this project include pandas, sklearn and numpy.

- **Classifier**:
  - Self: K Means Classifier, trivial system classifier
  - sklearn: Logistic Regression Classifier, K Neighbors Classifier, Support Vector Classifier (SVC), Perceptron, MLP Classifier, Ridge Classifier, Gaussian Naive Bayes Classifier, Decision Tree Classifier, Gradient Boosting Classifier
- **Cross Validation**:
  - Self: Cross Validation loop was written and tested which gave similar results to that of sklearn.
  - sklearn: KFold, TimeSeriesSplit

6 Contributions of each team member

Both of us contributed equally to the code and report for this project. It was a collaborative work where each one of us learnt by solving the doubts we had and also new skills were gained.

One reference system and two classifiers was tried by each member.

7 Summary and conclusions

It was good experience to work on an open problem like this where we had a chance to explore different approaches at each stage of ML pipeline. We tried different Standardization techniques and found that simply by standardizing the features can provide a boost to the model performance in comparison to the baseline model. We also found that for this particular problem set, PCA can provide substantial reduction in the number of features (n_components) and still allow model to work well on test set. As mentioned in the start applying simple things in the each stage of the pipeline can lead to a overall simple model but still perform better with less number of samples in the dataset. Our best performing model was Logistic Regression based classifier which was passed input from the standardized and feature reduced (using PCA) data.

References

[1] *Standard Scaler*, available at `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler`

[2] Vitor Cerqueira, et al., *Combining Boosted Trees with Metafeature*, in Advances in Intelligent Data Analysis XV: 15th International Symposium, Stockholm, 2016.

[3] *Dataset information* `https://archive.ics.uci.edu/ml/datasets/Algerian+Forest+Fires+Dataset++#`

[4] `https://link.springer.com/chapter/10.1007/978-3-030-36674-2_37`

[5] *Bernoulli*, available at `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.bernoulli.html`

[6] *Logistic Regression*, available at `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

[7] *K Neighbours Classifier*, available at `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`

[8] *SVC*, available at https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[9] *Perceptron*, available at https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html

[10] *GuassianNB*, available at https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[11] *ExperimentalResults*, available at https://drive.google.com/drive/folders/1OWtL0rUM_Qn-v4bjADB-He2PuokGeZSt?usp=sharing