Prediction of Forest Fire

Data Spaces, A.Y. 2020/2021
Politecnico di Torino

Stefano Bergia s276124
`s276124@studenti.polito.it`

Abstract

Every year forest fire causes incalculable economic and ecological damage, as well as put in danger many human lives. Thus predicting such critical environmental disaster is essential to mitigate this threat. In this report a dataset containing data from an algerian smart sensor grid is analyzed using various techniques and algorithms in order to determine which is the most suitable (and under which circumstances) to predict the possible occurrence of a forest fire.

Contents

1. Introduction

The aim of this report is to analize the Algerian Forest Fires dataset available on UCI website $http://archive.ics.uci.edu/ml/datasets/Algerian + Forest + Fires + Dataset + +$ and applying some machine learning algorithms on this set of data in order to predict the risk of a forest fire in certain conditions.

The Dataset contains 243 record collected from two region of algeria defined by the following attrubutes:

- Date: date in which data for this record was gathered, all data refers to a period of time from june to september 2012 (this attribute is irrelevant in the context of this analisys so it will be ignored)
- Temperature: max temperature at noon in Celsius degrees: 22 to 42
- RH: Relative Humidity in %: 21 to 90
- Ws: Wind speed in km/h: 6 to 29
- Rain: mm of rain in a day: 0 to 16.8

The following attributes are indexes computed following the Fire Weather index standard and directly depend on previous attributes, a more detailed explanation of these attributes an be found here: $https://cwfis.cfs.nrcan.gc.ca/background/summary/fwi$.

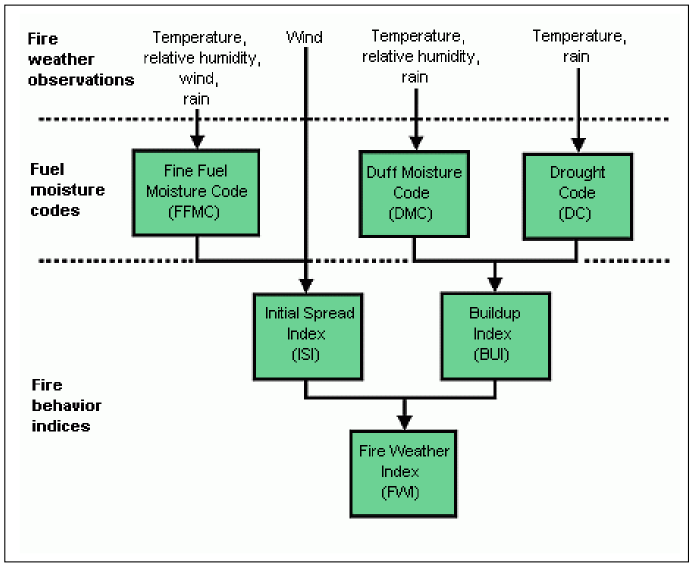

Fire weather observations
Temperature, relative humidity, wind, rain
Wind
Temperature, relative humidity, rain
Temperature, rain
Fuel moisture codes
Fine Fuel Moisture Code (FFMC)
Duff Moisture Code (DMC)
Drought Code (DC)
Initial Spread Index (ISI)
Buildup Index (BUI)
Fire behavior indices
Fire Weather Index (FWI)

Figure 1: FWI indexes dependencies

- FFMC: The Fine Fuel Moisture Code (FFMC) is a numeric rating of the moisture content of litter and other cured fine fuels. This code is an indicator of the relative ease of ignition and the flammability of fine fuel. From 28.6 to 92.5
- DMC: The Duff Moisture Code (DMC) is a numeric rating of the average moisture content of loosely compacted organic layers of moderate depth. This code gives an indication of fuel consumption in moderate duff layers and medium-size woody material. From 1.1 to 65.9
- DC: The Drought Code (DC) is a numeric rating of the average moisture content of deep, compact organic layers. This code is a useful indicator of seasonal drought effects on forest fuels and the amount of smoldering in deep duff layers and large logs. From 7 to 220.4
- ISI: The Initial Spread Index (ISI) is a numeric rating of the expected rate of fire spread. It is based on wind speed and FFMC. Like the rest of the FWI system components, ISI does not take fuel type into account. Actual spread rates vary between fuel types at the same ISI. From 0 to 18.5

- BUI: The Buildup Index (BUI) is a numeric rating of the total amount of fuel available for combustion. It is based on the DMC and the DC. The BUI is generally less than twice the DMC value, and moisture in the DMC layer is expected to help prevent burning in material deeper down in the available fuel. From 1.1 to 68
- FWI: The Fire Weather Index (FWI) is a numeric rating of fire intensity. It is based on the ISI and the BUI, and is used as a general index of fire danger throughout the forested areas of Algeria. From 0 to 31.1
- Classes: it's the target attribute, it canbe 'fire' or 'not fire'

The code was written in python on google colab platform taking advantage of the following libraries:

- numpy
- pandas
- plotly.express
- sklearn

The google colab notebook can be found here:
$https://colab.research.google.com/drive/1IEdckF3Oj97ZGG8YawYNtD7w_UBztwRa?usp = sharing.$

2. Data Exploration

Here some raw statistics are listed. 25% refers to 25 percentile (which is the value before which 25% of the sample are located), 50% to the median and 75% to the 75 percentile. it is already clear that features have different scales so data normalization will be required.

	Temperature	RH	Ws	Rain	FFMC	DMC	ISI	BUI
count	243.00	243.00	243.00	243.00	243.00	243.00	243.00	243.00
mean	32.15	62.04	15.49	0.76	77.84	14.68	4.74	16.69
std	3.63	14.83	2.81	2.00	14.35	12.39	4.15	14.23
min	22.00	21.00	6.00	0.00	28.60	0.70	0.00	1.10
25%	30.00	52.50	14.00	0.00	71.85	5.80	1.40	6.00
50%	32.00	63.00	15.00	0.00	83.30	11.30	3.50	12.40
75%	35.00	73.50	17.00	0.50	88.30	20.80	7.25	22.65
max	42.00	90.00	29.00	16.80	96.00	65.90	19.00	68.00

2.1. Class Distribution

Here it is analyzed how the target class value (fire/not fire) are distributed, with respect to both the whole dataset and per single attribute value

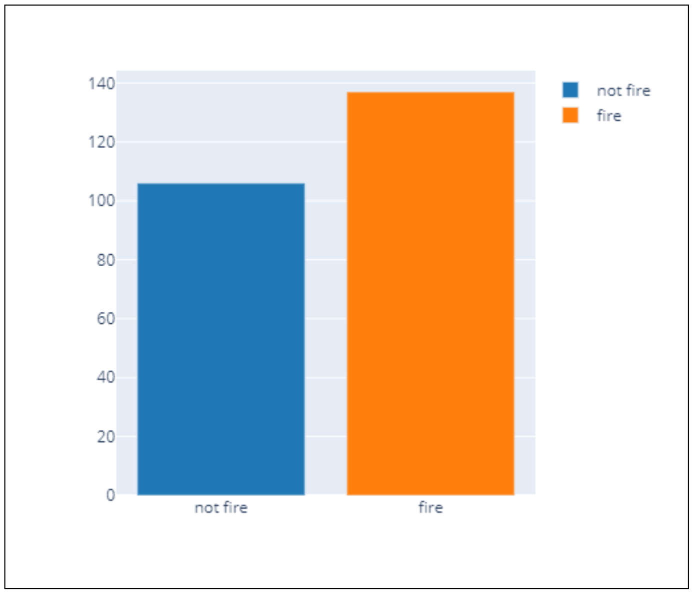


Figure 2: Target class distribution, Fire percentage is 56.38%

Since the two classes are quite balanced there is no need to use oversampling tecniques such as SMOTE.

Per attribute class distribution charts are listed in the following page. From these charts we can see that the number of fires for the Temperature, Relative humidity and wind speed features follow a sort of gaussian distribution, in particular the number of fires recorded is higher around the average value of these features (32.15 °C, 62% 15.5 Km/h). Rain Attribute seems not to be too informative because all of the fires were of course recorded with no (or very little) rain. Other parameters directly depends on these raw measurements, of course we expect them to be correlated and we have to decide which are the most useful in order to predict a fire. in particular FFMC and ISI seems to work with a "sort of threshold" which means that after a certain value all samples are categorized as fire. The Remaining features all seem to follow a similar distribution which is a gaussian skewed to the left but with different scales for each attribute.

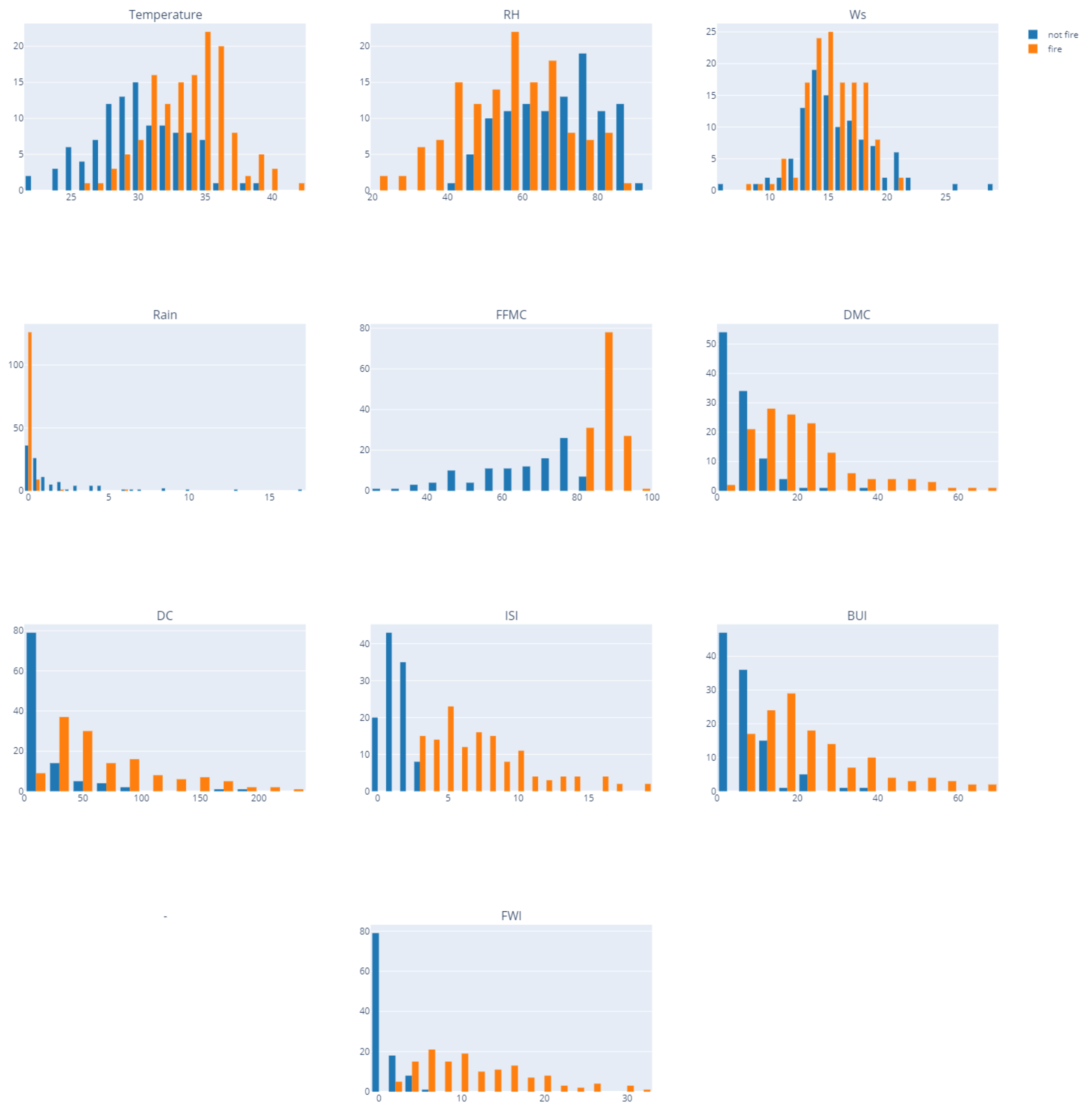


Figure 3: per attribute class distribution

2.2. Box plots

here the box plots for each feature are listed. A box plot visualizes the following statistics:

- median
- the first quartile (Q1) and the third quartile (Q3) building the interquartile range (IQR)
- the lower fence (Q1 - 1.5 IQR) and the upper fence (Q3 + 1.5 IQR)
- the maximum and the minimum value

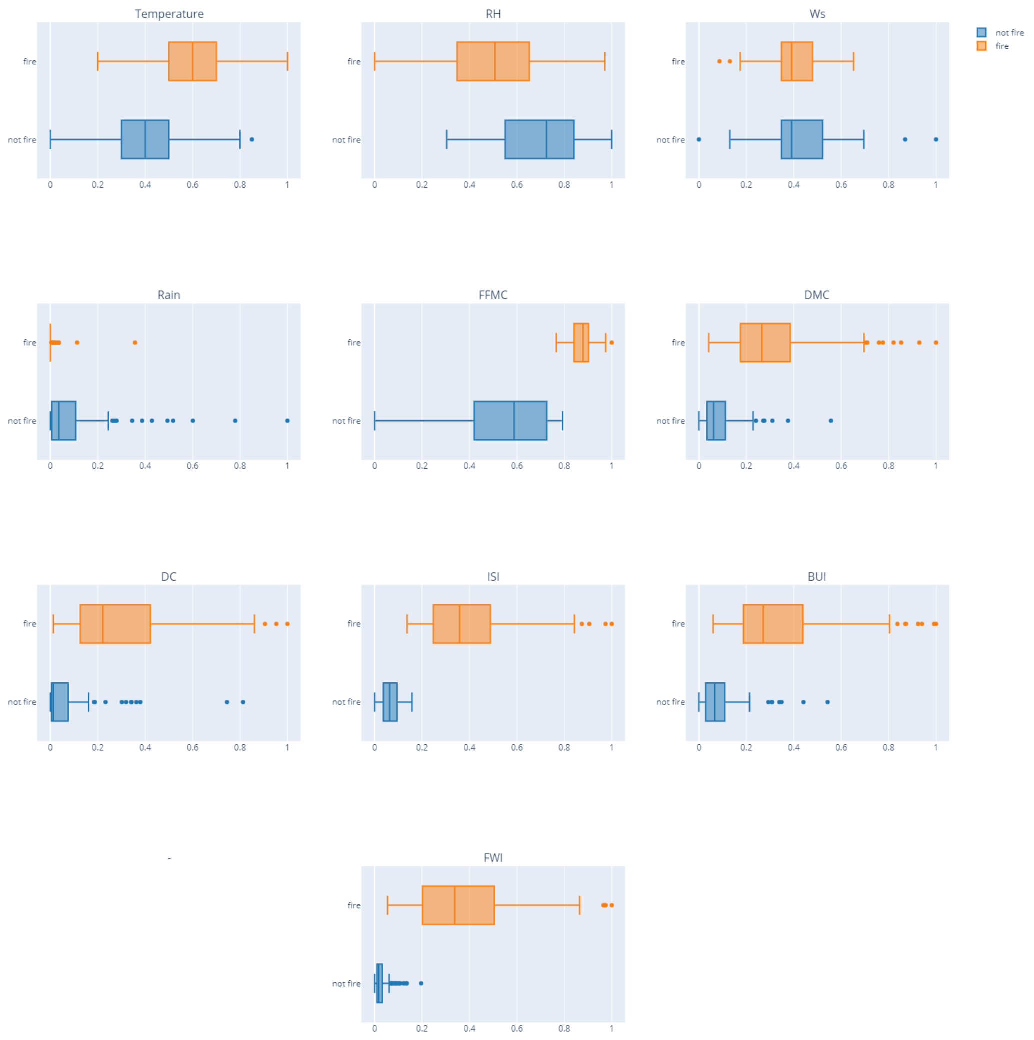


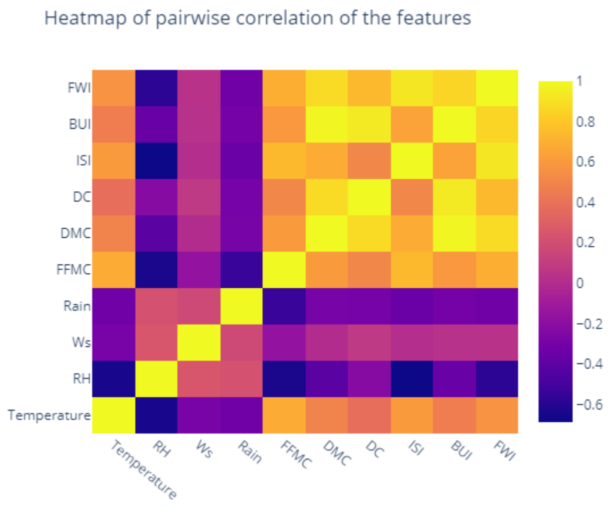
Figure 4: per attribute box plots

The boxplots highlight the previous observation. The first features (Temperature, Relative Humidity, and Wind Speed) are more overlapped, while others have a clear threshold between fire and not fire class. So it should be possible to apply dimensionality reduction techniques and work with less infromation.

2.3. Correlation matrix

In order to investigate the pair-wise correlations between two variables X and Y, it is usefull to use the Pearson correlation. Let X,Y be the standard deviation of X,Y and $cov(X,Y) = E[(X\text{-}E[X])(Y\text{-}E[Y])]$. Then we can define the Pearson correlation as the following: $\rho X, Y = \frac{cov(X,Y)}{\sigma X \sigma Y}$.
To visualize these correlations we use a heatmap plot, in which high correlations are coloured more to the yellow and lower ones more to the violet.

A value close to one means that the two features can be associated to a linear correlation, meaning that if the average of the samples from one feature increases also the average of the sample from the other distribution increases. The opposite if the correlation is close to -1.



As already implied by the relation between the FWI indexes and the raw data, we can see that there is an highly positive correlation between the indexes and the temperature as well as between the indexes with each other. We can also see that the indexes are negatively correlated with the relative humidity. This suggest that it will be possible to reduce dimensionality of the problem by using or ignoring the fwi indexes given that many features seems to model the same aspect of the event.

2.4. Data Normalization

Another thing to notice is the fact that the features have different ranges, so before proceeding it's better to apply min max noralization in oder to have all features in the same scale. Min max normalization is applied because in the original dataset all values are positive and it is necessary to keep this property after normalization. Min Max normalization has the following formulation: $X = \frac{(X - min(X))}{(max(X) - min(X))} \; \forall \; X \epsilon Features$

2.5. Split in training and testing set

to create a common baseline between all models, an hold out stratified testing set is created by removing 20% of the record from the dataset keeping the relative proportion for fire/not fire labels. All algorithms from now on will be trained on the same training set and tested on the same testing set.

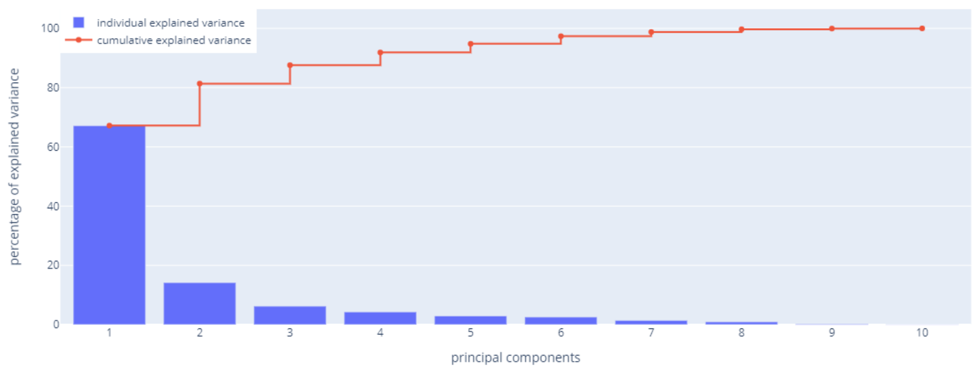
3. Principal Component Analysis

Principal Component Analysis is a linear transformation algorithm whose purpose is to project data sampled in an high dimensional space onto a lower dimensional space, so that they can be represented in an easier and more meaningful way and avoid the so called "Curse of Dimensionality".
The Curse of dimensionality affects distances between points in high dimensional spaces. The greater the number of dimensions (features) the less meaningful distance measures become. This happens because point density in n-dimensional spaces gets lower and lower as the number of dimension increases and if we fix a certain n-dimensional Volume and a density, the number of points in the volume needed to keep that density grows exponentially.
The output of this procedure are the so called "principal components", one for each dimension of the target space. The principal components are linear combinations of original features and are orthogonal with respect to each other, as together they generate the target space. Each PC is composed by two element, a loading (which is a versor) and a score which represents the amount of variance along the direction of the loading. If we rank all principal components by their variance and take the first n (where n is the dimension of the target space) we obtain the base vectors of the target space.
The PCA algorithms works as follows:

- center data by subtracting to each attribute the average value for that features
- compute the sample-covariance matrix, a dxd symmetrical matrix (d is the original number of features) in which each cell i,j of the matrix contains the covariance between feature i and feature j
- Compute eigenvectors and eigenvalues of the sample covariance matrix (solve $det(\Sigma – \lambda I) = 0$) to find the eigenvalues and then plug them in the equation $det(\Sigma – \lambda I)V = 0$ to find the corresponding eigenvectors)
- Sort eigenvectors with respect to their eigenvalues
- build the projection matrix W by putting the top n (n dimension of the target space) eigenvectors into the columns of W
- transform the dataset X by multiplying it with W (W is a projection matrix)

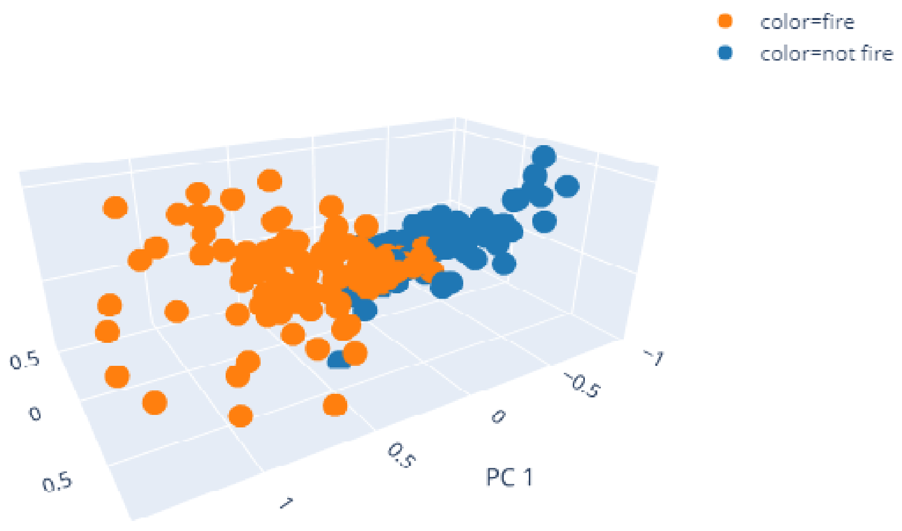
Two important measures with respect to PCA are the individual explained variance and the cumulative explained variance: the individual explained variance is the ratio between the variance of that principal component and the total variance, the cumulative explained variance for PCi is the sum of all previous PC explained variance up until i (included). The higher the cumulative explained variance the better is our representation because it will introduce fewer errors (projecting on a lower dimensional space means that points that originally are far apart could become closer and possibly be classified, by algorithms applied on projected data, with the same label and this could be an error)

3.1. Explained variance



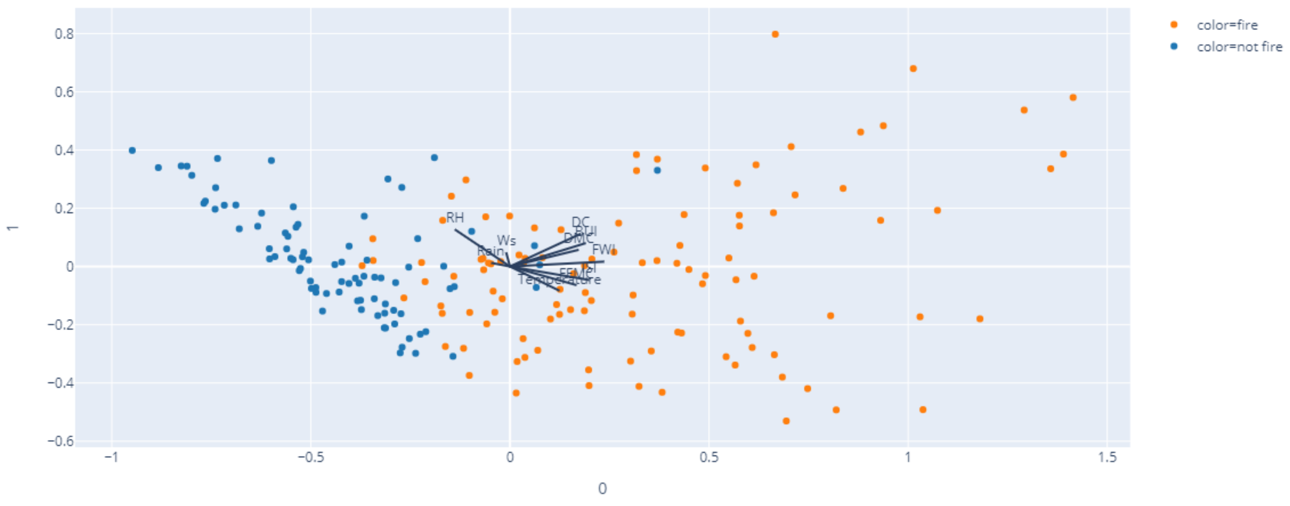
As expected, applying PCA we obtained that most of the variance is explained by the first principal components, in particular the first 2 together accounts for the 81,36% of the total variance. That’s because most of the parameters are correlated to each other and they basically represent the same aspect of the event, just from different point of view.

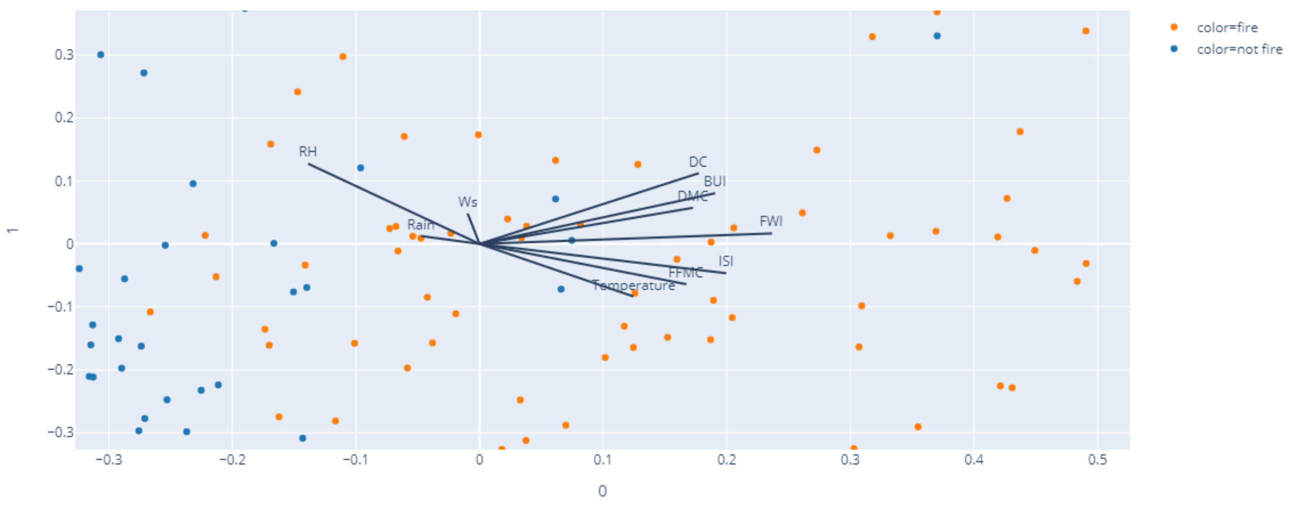
3.2. Scatter Plot with 3 principal components



As we can see from the scatterPlot, projecting data on a 3d space generated by the first 3 principal component yields quite good results. The two sets (fire/not fire) are easily distinguishable, leaving only an uncertainty region when the two touches which amounts to about 13% of the data. A linear classifier with an appropriate margin should perform quite good on this space (SVM will be discussed later). The total explained variance of the first 3 principal component is 87.59%

3.3. Biplot with first 2 components





The previous observations still hold even with only in two dimensions. The two sets are still quite separable, so it makes sense to use just the first two principal components.

Moreover if we look at the loading vectors we can see that the closest features to the principal components are FWI (close to PC0) and Ws (close to PC1), this means that these two features are determined for the most part to only one principal component, meaning that they probably represent a really similar concept. This was to be expectd as FWI is a measure that aggregates all raw features as explained before.

4. Model Selection

Data exploration yielded some interesting results. In particular correlation matrix showed that FWI measure are all correlated with each other, so it is reasonable to ignore FWI attribute and use only the raw features.

On the contrary PCA seems to suggests that refined features make a better job at explaining variance and therefore it can be interesting to use principal components in order to solve classification task.

In the following paragraphs various models are presented, for each of the three scenarios will be analyzed:

- keep only raw features (temperature,RH,Ws,Rain) and drop the others (Scenario 1)
- project data in 2d -principal component space (Scenario 2)
- use all the features (Scenario 3)

All models will be trained on the same training set and compared computing the accuracy score on the same test-set and the confusion matrix.

the accuracy score is the ratio between the amount of test samples correctly labeled and the total amount of samples.

The confusion matrix contains the following four measures:

- TP = True positive -> number of samples for which the prediction is positive and the true label is positive
- FP = False positive -> number of samples for which the prediction is positive but the true label is negative
- TN = True negative -> number of samples for which the prediction is negative and the true label is negative
- FN = False negative -> number of samples for which the prediction is negative but the true label is positive

In Forest fire scenario it is of course important to have an high TP value (fire correctly predicted) as well as a FP value as low as possible (this would correspond to a fire not predicted when it should have)

Confusion matrix values can be aggregated through the f1-score, which is given by the following equation:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{precision + recall}$$

where :

- $precision = \frac{\text{TP}}{\text{TP+FP}}$
- $recall = \frac{\text{TP}}{\text{TP+FN}}$

f1-score has a maximum value of 1, which corresponds to a perfect classifier.

The accuracy score is not only used to compare different models but also so select hyperparameters for the models. This is done trying different parameters though grid-search and applying k-fold cross validation for each combination of parameters).

k-fold cross validation is a technique that tries to validate a model (i.e. evaluate it's performance and select the best hyperparameters) without "wasting" useful data for training fixing a part of the training set (like in the hold-out validation technique). K-fold splits the training set in k sets of equal size. Then the model is trained on k-1 sets and the accuracy score is computed on the remaining set. This is repeated iteratively until all the sets have been used for validation once. The final Accuracy score is obtained computing the average of all scores computed on the validation sets.

5. Logistic Regression

Logistic regression is one of the simplest machine learning algorithms. It's purpose is to fit a linear combination of the input features and pass it through the sigmoid function that squishes all the valued in a range between 0 and 1. Despite being a regression problem it is mostly used for binary classification tasks because sigmoid functions converges really fast either to 0 or 1, with a really steep zone centered in 0. So a threshold can be set in the center value (that will always be 0.5) and classify everything on the right as class1 and everything on the left as class0.

we can compute the probability of labeling a feature vector x as class1 can be computed as:

$$P(Y = 1|x_i) = \frac{\exp(\beta_0 + x_i^T \boldsymbol{\beta})}{1+\exp(\beta_0 + x_i^T \boldsymbol{\beta})} = \frac{1}{1+\exp(-(\beta_0 + x_i^T \boldsymbol{\beta}))}$$

where β_0, $\boldsymbol{\beta} = (\beta_1, ..., \beta_n)^T$ are the model parameters.

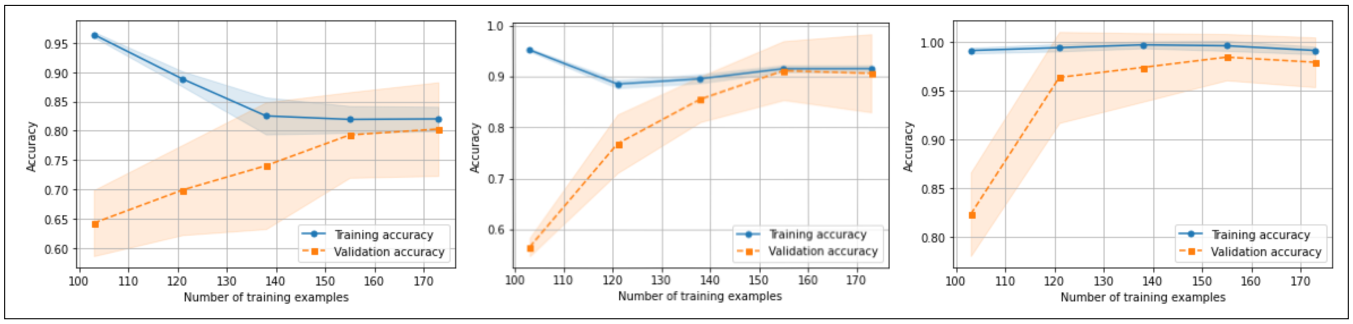


Figure 5: learning Curve of Logistic regression for scenario 1(raw features), scenario 2(PCA), scenario 3 (All features)

Despite being a simple model, logistic regression works quite well for classification on this dataset. The first scenario on the left (training only on "raw features") is the worst as it reaches a validation accuracy of 81%, the middle scenario refers to training on first two principal components and a validation accuracy of 91% is reached. Finally the scenario on the right refers to training done on all features and a 98% validation accuracy is reached. When tested on the common testing set the three models reach an accuracy of 84%,92% and 94% respectively. Below The confusion matrix for the three scenarios are showed, they of course reflect the accuracy results just discussed.

Raw Features	predicted not fire	predicted fire
true not fire	0.77	0.23
true fire	0.11	0.89

PCA	predicted not fire	predicted fire
true not fire	0.95	0.04
true fire	0.11	0.89

All Features	predicted not fire	predicted fire
true not fire	0.95	0.04
true fire	0.07	0.93

The hyperparameters to set for logistic regression are the regularization type (l1 or l2) and C, which is the incerse of the regularization strength. The Best configurations for the three scenarios are respectively: $l1/C = 10, l2/C = 1, l1/C = 10$

6. K Nearest neighbours

K Nearest neighbours is a non linear classification algorithm. It classifies new samples according to the K nearest neighbours in the training set. When a new samples has to be classified, the distance between the new sample and every training sample is computed. Then the K closest samples are selected (k is an hyperparameter) and the new sample is labeled as the class that is most present in the k training samples. Another hyperparameter to set is the distance metrics (here minkowski distance is used, where the power parameter p is selected though grid search), and the distance penalization (all k votes can wight the same or can have a penalization factor which grows with the distance with respect to the new sample)

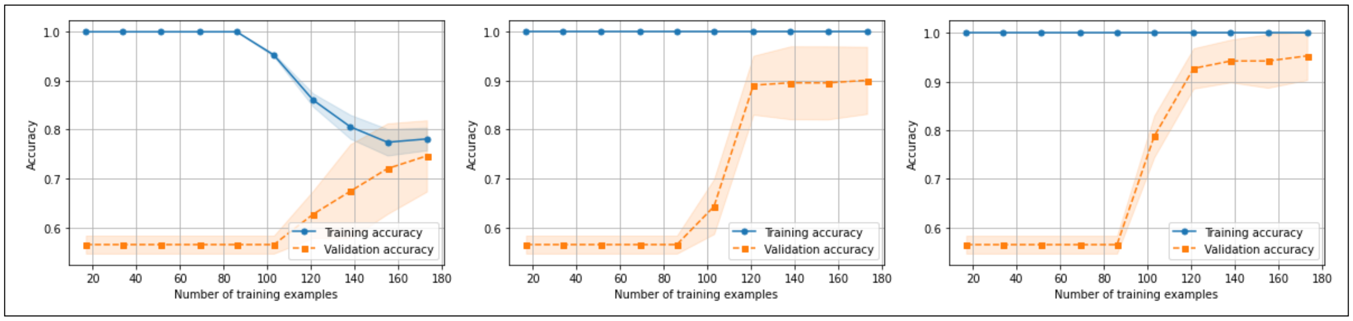


Figure 6: learning Curve of knn for scenario 1 (raw features), scenario 2(PCA), scenario 3(all features)

Knn has slightly worse performance with respect to Logistic regression, but it reaches quite high accuracy anyway. The first scenario on the left (training only on "raw features") is again the worst as it reaches a validation accuracy of 77%, the middle scenario refers to training on first two principal components and a validation accuracy of 90% is reached. Finally the scenario on the right refers to training done on all features and a 95% validation accuracy is reached. When tested on the common testing set the three models reach an accuracy of 76%,92% and 96% respectively. Below The confusion matrix for the three scenarios are showed, they reflect the accuracy results just discussed.

Raw Features	predicted not fire	predicted fire
true not fire	0.77	0.23
true fire	0.25	0.75

PCA	predicted not fire	predicted fire
true not fire	0.95	0.04
true fire	0.11	0.89

All Features	predicted not fire	predicted fire
true not fire	0.95	0.04
true fire	0.04	0.96

The hyperparameters to set for knn are the number of neighbours k,the penalization of the distance measure ('uniform' if all the distances have the same weight, 'distance' if farthest neighbours weight less than the closest one) and the power of the minkowski distance p. The Best configurations for the three scenarios are respectively: $k/uniform/2, 15/distance/2, 5/1/distance$

7. Support vector Machine

Support vector machines are a linear binary classification algorithm that tries to find an hyperplane to separate the data by maximizing the margin between the hyperplane and the two sets of data themselves. The term "support vector" refers to the the training points that lies on the margins. Indeed solving the optimization problem for the margin maximization shows that the contribution given by points that don't lie on the margin is null; only the samples on the margin plays a role in finding the solution and that's why they are called support vector. This makes the SVM really robust with respect to outliers.

The standard approach to SVM is also called "Hard margin SVM" as it doesn't allow for any misclassification of training data. In real world problems this is too restricting though, as perfect linear separability it's really uncommon due to noise and outliers. To solve this problem there exists a variation of the SVM model called "Soft margin SVM". Soft margin SVM introduces a new learnable parameter called "slack variable" which allows some points to violate the hard margin constraint so that some data can be classified wrongly. Each slack variable is weighted by an hyperparameter (usually called C) which is used to define how wide the incertainty area should be (the greater the value of C, the thinner the soft margin, the hard margin SVM can be see as a soft margin SVM with infinite C) This of course introduce a bit of uncertainty in the model but makes it much more flexible and applicable to real world problems.

Most of real world problems are not linearly separable, Nevertheless SVM can still be applied by mapping samples data to an higher dimensional space (through a mapping function) in which they can be separated linearly, and then projecting them back on the original space after the separation hyperplane has been found on the higher dimensional space. Finding a good feature mapper is not an easy task at all though. Luckily, given that in SVM optimization problem doesn't require the whole feature mapping expression but only the inner products of the feature map. The inner product of the feature mapping function is called "kernel" and is defined as $K(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2)$. So to perform the feature mapping it is enough to substitute every inner product in the original SVM with the kernel function, this is called "kernel trick"

For this report we analyze two kernels:

- linear kernel: $K(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2$ which is the standard setting without mapping
- radial basis function (rbf) kernel:$K(\mathbf{x}_1, \mathbf{x}_2) = exp(-\gamma(\|\mathbf{x}_1 - \mathbf{x}_2\|^2))$

where gamma is an hyperparameter which determines the area of influence for each point, low gamma means great area of influence

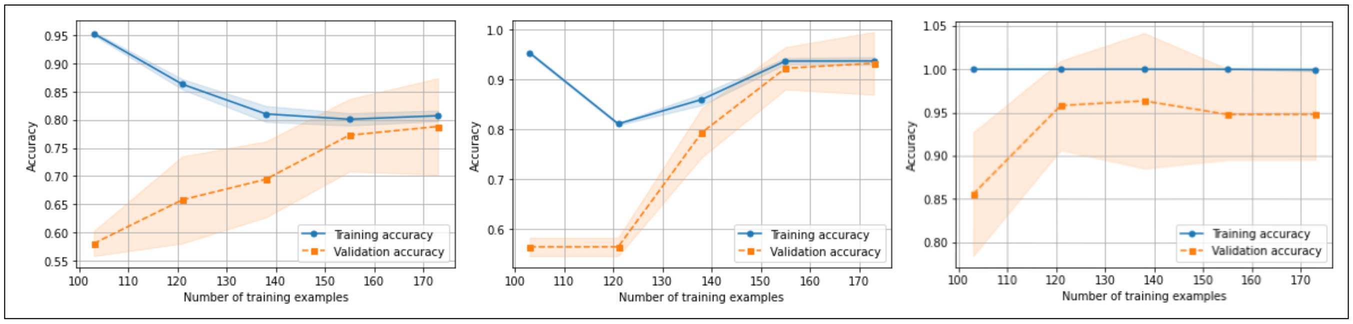


Figure 7: learning Curve of svm for scenario 1 (Raw features), scenario 2 (PCA), scenario 3 (All features)

The SVM has quite good results. The first scenario on the left (training only on "raw features")reaches a validation accuracy of 80%, the middle scenario refers to training on first two principal components and a validation accuracy of 91% is reached. The last scenario on the right refers to training done on all features and a 96% validation accuracy is reached. When tested on the common testing set the three models reach an accuracy of 84%,92% and 96% respectively.

Raw Features	predicted not fire	predicted fire
true not fire	0.77	0.23
true fire	0.25	0.75

PCA	predicted not fire	predicted fire
true not fire	0.95	0.04
true fire	0.11	0.89

All Features	predicted not fire	predicted fire
true not fire	0.95	0.04
true fire	0.04	0.96

The hyperparameters to set for SVM are the kernel function,the "size" of the margin C and for the rbf kernel also the gamma parameter explained before. The Best configurations for the three scenarios are respectively: $rbf/C = 100/gamma = 0.1, rbf/C = 1/gamma = 0.1, rbf/C = 100/gamma = 1$

8. Decision Tree

Decision tree performs classification through a greedy top-down approach. At first a feature is selected and the training set is split in two subset according to a threshold value selected in the domain of the selected attribute. The attribute is selected so that a performance measure is optimized. Then the process is repeated again on the two subset until either the maximum depth of the tree is reached (this value is an hyperparameter) or optimization all the branches is above a certain threshold (also an hyperparameter). To perform classification on a testing sample the tree is traversed from the top to the bottom, going on the right or left branch according to the feature and the threshold of that node with respect to the value that feature has for the sample data to classify. When a leaf is reached the sample is labeled according to the majority class in that leaf. The following optimization measures are considered:

- Gini index: $I_G(p) = \sum_{i=1}^{j} p_i(1-p_i)^2$ where j is the number of classes and p_i is the fraction of samples assigned to class with respect to the total number of sample for the subset. it measures the total variance across all the classes for a subset. The purer a split, the better job it does in separating the classes.
- Information gain: $IG = H(p) - \sum_{i=1}^{k} \frac{n_i}{n} H(i)$ where $H(p)$ is the cross entropy defined as $H(t) = -\sum_{i=1}^{j} p_i \log_2 p_i$, with n, n_i the number of samples in the subset and the number of sample of class i in that subset respectively

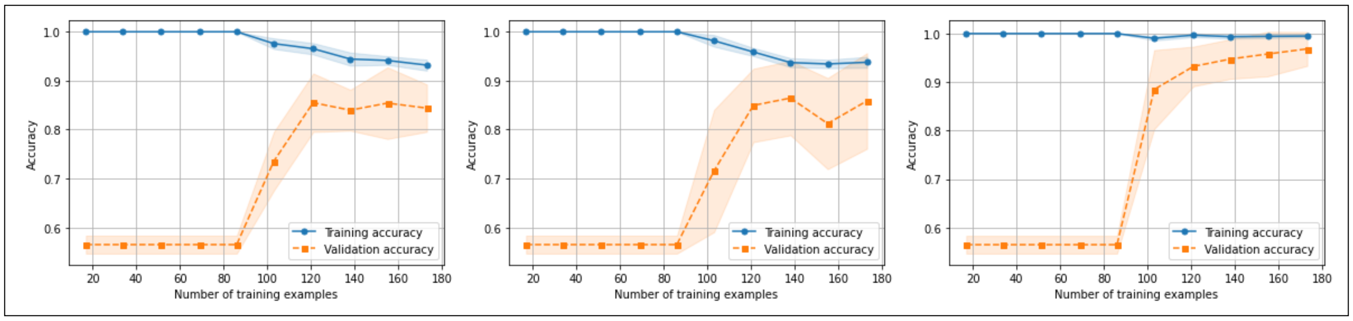


Figure 8: learning Curve of decision tree for scenario 1 (Raw features), scenario 2 (PCA), scenario 3 (All features

The performance of decision three are the following. The first scenario on the left (training only on "raw features")reaches a validation accuracy of 86%, the middle scenario refers to training on first two principal components and a validation accuracy of 86% is reached. The last scenario on the right refers to training done on all features and a 97% validation accuracy is reached. When tested on the common testing set the three models reach an accuracy of 80%,80% and 94% respectively.

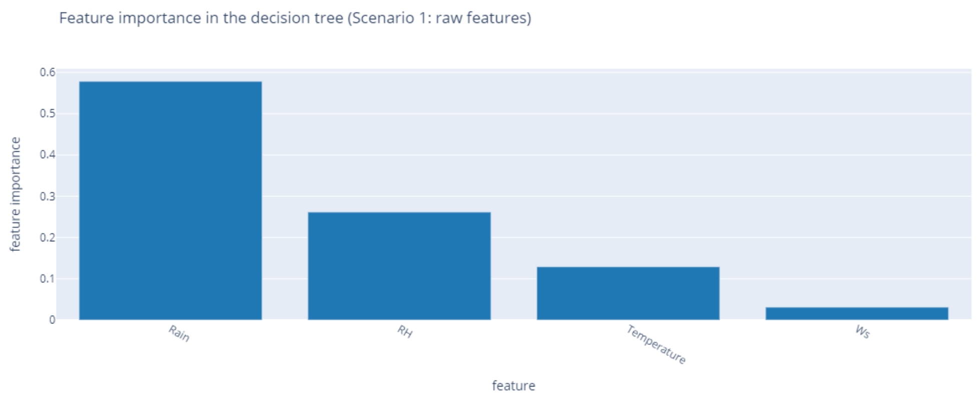
Raw Features	predicted not fire	predicted fire
true not fire	0.68	0.32
true fire	0.11	0.89

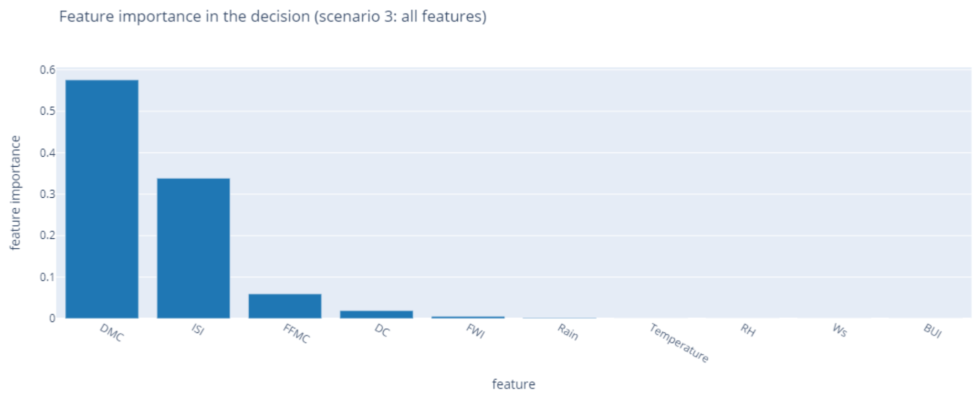
PCA	predicted not fire	predicted fire
true not fire	0.82	0.18
true fire	0.21	0.79

All Features	predicted not fire	predicted fire
true not fire	0.86	0.14
true fire	0.00	01.00

The hyperparameters to set for Decision tree are the criterion for the split (gini or entropy),the maximum depth of the tree and the minimum number of samples each split must have, this last parameter is needed to prevent the model to overfit by creating many small sets with perfect purity . The Best configurations for the three scenarios are respectively: $criterion : entropy/maxdepth : 25/minsamplesplit : 10, criterion : entropy/maxdepth : 50/minsamplesplit : 14, criterion : gini/maxdepth : 15/minsamplesplit : 6$

The following charts show the feature importance for the first and third scenario (it isn't very informative to do this analysis on the pca scenario with just 2 principal components) As expected the rain Parameter is really important for the first scenario given that all of the "fire" sample were recorded with a rain value of zero. For the third scenario we can observe that the FWI indexes plays an important role in the decision process. Moreover recalling the data exploration done before it is interesting to point out that the most important features are the indexes that had a clear threshold between "fire" and "not fire". The model was able to identify this thresholds and use them right away to output purer sets, thus resulting in a really accurate classification.





Raw Features	predicted not fire	predicted fire
true not fire	0.73	0.27
true fire	0.07	0.93

PCA	predicted not fire	predicted fire
true not fire	0.86	0.14
true fire	0.11	0.89

All Features	predicted not fire	predicted fire
true not fire	0.96	0.04
true fire	0.04	0.96

9. Random Forest

Random forest are an evolution of decision tree, they are an ensamble method meaning that many decision tree are trained (with the usual algorithm) and then the final labelling is done through majority voting. The trees are trained on a unique dataset of fixed size created by random sampling with replacement from the training dataset. This technique is called Bagging and it's used to reduce variance and as a consequence make the trees more accurate.

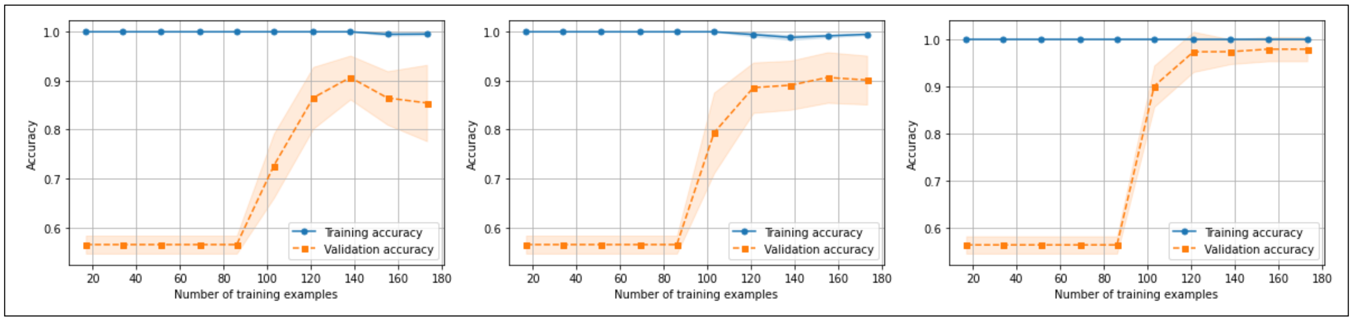
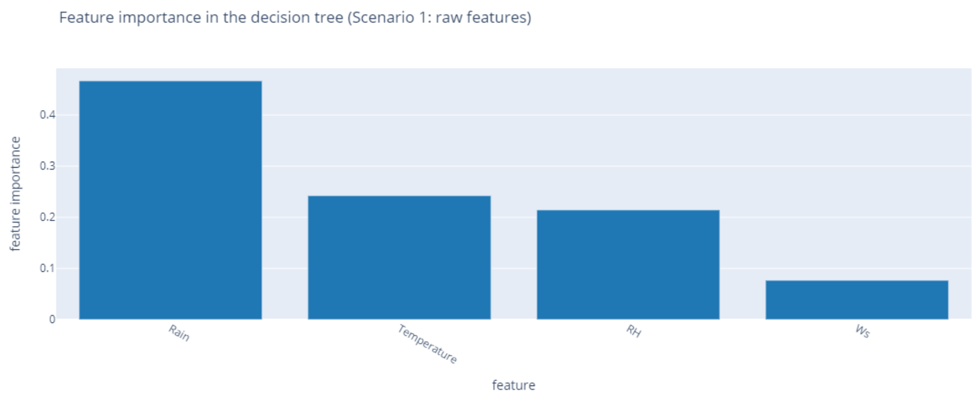


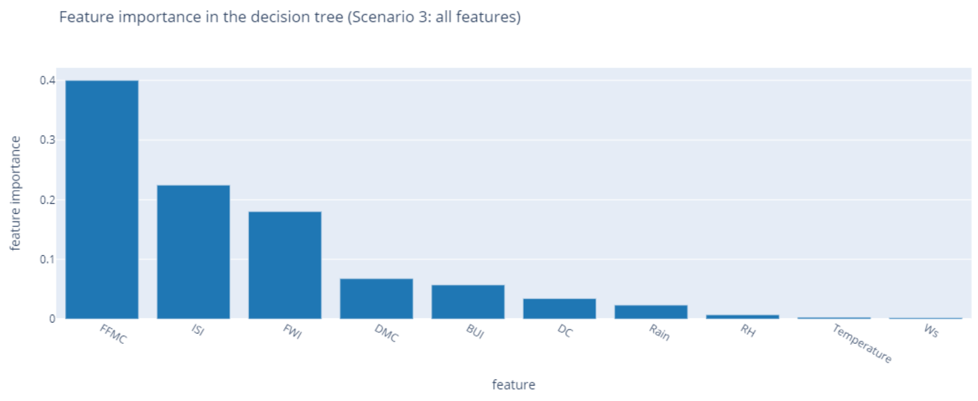
Figure 9: learning Curve of random forest for scenario 1, scenario 2, scenario 3

The performance of random forest are really good. The first scenario on the left (training only on "raw features")is associated with the highest accuracy for this setting equals to 87%, the middle scenario refers to training on first two principal components and a validation accuracy of 91% is reached. The last scenario on the right refers to training done on all features and a really high 98% validation accuracy is reached. When tested on the common testing set the three models reach an accuracy of 84%,88% and 96% respectively

The hyperparameters to set for Random forest are the same of decision tree with the addition of the number of tree in the forest as a new parameter. The Best configurations for the three scenarios are respectively: criterion:entropy/max depth:50/n estimators:50, criterion:gini/max depth:25/n estimators:50, criterion:gini/max depth:15/n estimators:30

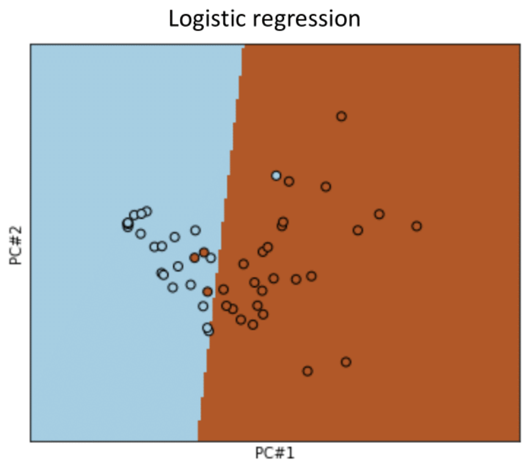
As far as Feature importance is concerned the observation pointed out in decision trees paragraph still holds. One interesting thing to note here is that feature importance seems to be a bit more distributed among more features, especially in the scenario with all the features. A possible explanation of this could be that bagging reduces the amount of data available for the single decision tree, so it forces it to use all the features at its disposal to perform the classification.

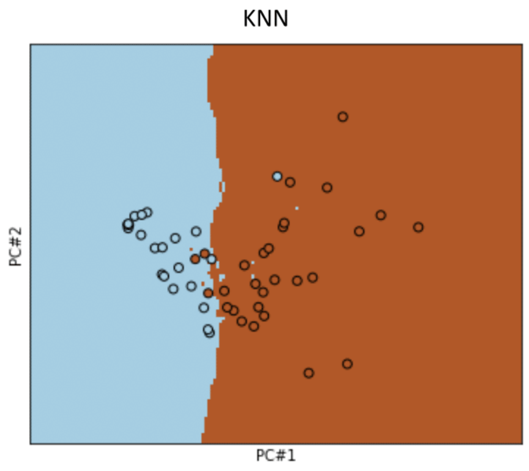

Feature importance in the decision tree (Scenario 1: raw features)
feature importance
0
0.1
0.2
0.3
0.4
Rain
Temperature
RH
Ws
feature

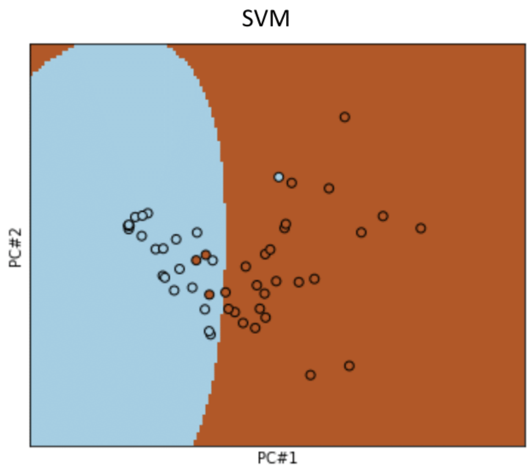

Feature importance in the decision tree (Scenario 3: all features)
feature importance
0
0.1
0.2
0.3
0.4
FFMC
ISI
FWI
DMC
BUI
DC
Rain
RH
Temperature
Ws
feature

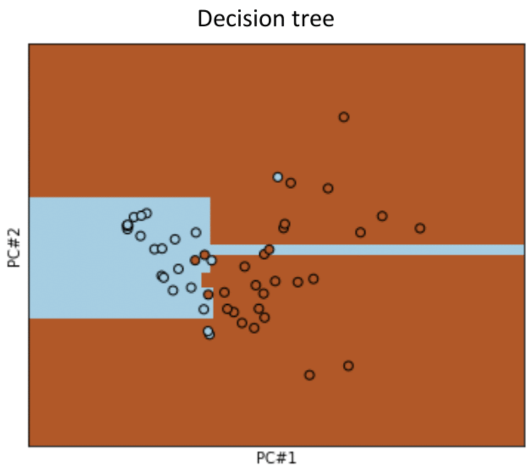
10. Decision Boundaries

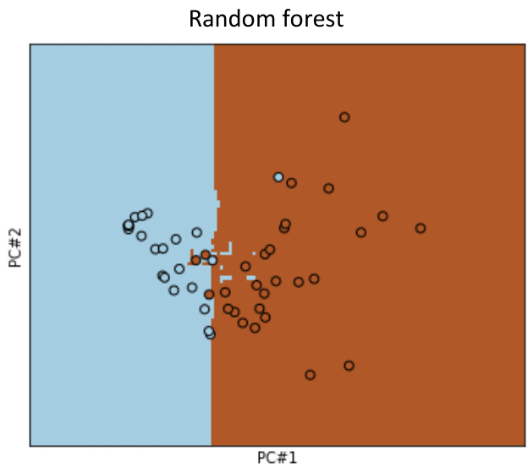
It is interesting to take a look at decision boundaries for the various models implemented. Of course this is possible to do in a meaningful way only for the PCA scenario as it is easy to plot a 2d chart.





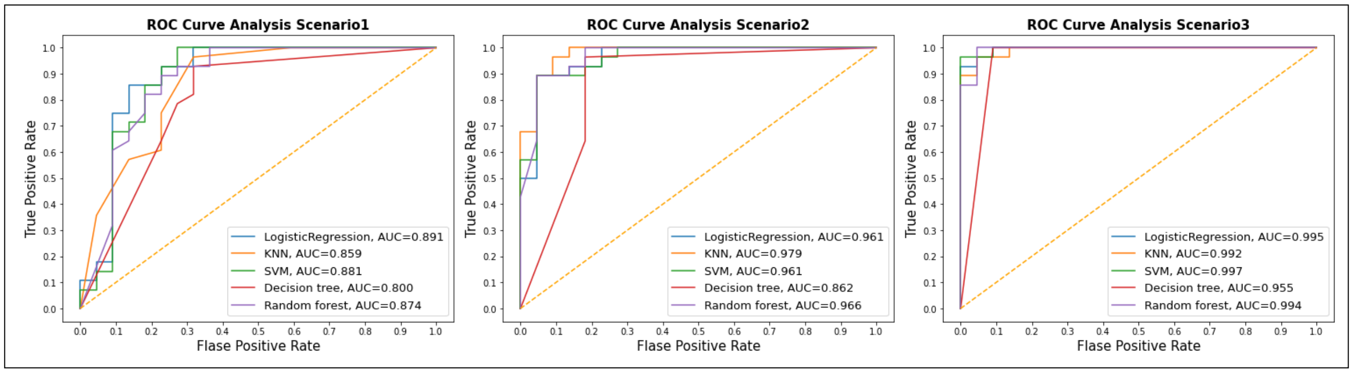






11. Comparison

To compare the various model it is useful to plot the Receiver operating characteristic (ROC) curve. This plot studies the relation between the false positive rate (x axis) and the true positive rate, or recall (y axis). The important measure in this scenario is the Area under the curve (AUC); a model gets better the closer this area gets to 1. An area of one means that no matter how many false positive are found, the "true" samples will always be classified correctly, while an area on 0.5 means that the classifier doesn't apply any "reasoning" and simply assigns labels randomly. An area under 0.5 doesn't really makes sense as it would mean that the classifier actively tries to misclassify samples. The following are the roc curves for all the models grouped by the three scenario.



From the three plots it's clear that the worst scenario is the first one with only the raw features while the best one is the third one using all features meaning that the fwi indexes are not a redundancy but actually helps predicting fires. The second scenario of the PCA has better performance than the first one and worse than the third but they are still quite good. PCA scenario is interesting because classification task with only two features (the principal components) it's much simpler and faster than the classification on all the features. So it could be a viable option if it was necessary to perform real time prediction directly from the sensors gathering the data (they of course wouldn't need to compute every time the projection matrix)
Considering each scenario separately it is clear that models don't perform the same an all scenarios. For the first one the better model seems to be the Logistic regression, for the second one KNN and for the third one random forest.

To better evaluate the models some performance measures have been computed on the common testing set and have been put in table form for each scenario.

Raw features	**accuracy**	**precision**	**recall**	**f1**
LogisticRegression	0.840	0.833	0.893	0.862
KNN	0.760	0.808	0.750	0.778
SVM	0.840	0.833	0.893	0.862
Decision tree	0.800	0.781	0.893	0.833
Random forest	0.840	0.812	0.929	0.867

For Scenario One the models with the highest accuracy are the logistic regression, the SVM and the random forest with an accuracy of 84% but among these the highest precision is reached only by logistic regression and SVM. The recall measure is really important because the highest it is, the fewer false negative there are (in our scenario a false negative would be a fire not detected, so a mistake could have dangerous consequences). The model with the highest recall is the random forest and has a consequence also its f1 score is the highest

PCA	accuracy	precision	recall	f1
LogisticRegression	0.920	0.962	0.893	0.926
KNN	0.920	0.962	0.893	0.926
SVM	0.920	0.962	0.893	0.926
Decision tree	0.800	0.846	0.786	0.815
Random forest	0.880	0.893	0.893	0.893

For Scenario Two the models with the highest accuracy are the logistic regression,the Knn and the SVM with an accuracy of 92%. It is interesting to note that with only two features the simplest models reach and higher accuracy. The highest precision and recall are reached by KNN, SVM and random forest. Overall model trained on PCA always outperform models trained on only raw features.

All features	accuracy	precision	recall	f1
LogisticRegression	0.940	0.963	0.929	0.945
KNN	0.960	0.964	0.964	0.964
SVM	0.960	0.964	0.964	0.964
Decision tree	0.940	0.903	1.000	0.949
Random forest	0.960	0.964	0.964	0.964

For Scenario Three all the models reach a really high accuracy, so it is not really possible to determine an absolute best. The same holds for precision and recall. Surprisingly the decision tree reaches a perfect recall with no false positive, but this is probably due to the samples in the testing set, if we were to change them it will probably change. Again the f1 score doesn't seem to clearly discriminate between models but considering also the AUC from the roc curves the random forest model seems to be the better one. Random forest is also the most complex model to implement so if this classifier were to be used on a system with low computational power aso other models should be considered.

12. Conclusion

In this report a dataset of algerian forest fire has been analyzed to understand which features are the most relevant for forest fire detection and which models can be applied in order to predict such an event. Through a series of experiments and evaluation it was found that FWI indexes are not simply an aggregation of raw features collected by the sensors but they actually introduce some insight on the problem allowing the models to perform better. It was also shown that applying PCA on this problem allows to simplifying it without loosing too much on performance an therefore it should be taken into consideration in those settings when computational power and storage capabilities are critical factors, such as a smart sensor directly on the field. In conclusion it seems to be possible to apply various machine learning models and algorithm to predict the possible outburst of a forest fire, even implementing more models at the same and use a majority voting system could result in even better and more robust performance.