**Project: Smoke Detection**

**By: Pratima Paudel**

**Dataset Link**: <https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset>

* Usability: 10.00
* Records: 62.6K
* Features: 14

**Task**: A collection of data points on various parameters related to air entities and our task is to realize when there is smoke according to the data passed. This is a **binary classification task** wherein we have to classify if there is smoke or not.

**Introduction**: Our project is about a smoke detection system that introduces a smoke detector to determine a fire or not. It helps to give a warning to people about the smoke fire, carbon monoxide, or other fire-related or general notification emergencies which are detected in their surroundings. Some of the classifiers that we used in this project are KNeighborsClassifier, SGDClassfier, LogisticRegression, RandomForestClassifeir, GradientBoostingClassifier, AdaBoostClassifier, etc. In our project for getting the result of smoke detection, we have done data exploration, data cleaning, data splitting, and feeding the data to various models, and plot results, and fine-tune model stacking.

described the accuracy given are, KNeighborsClassifier accuracy(99.81), similarly SGDClassfier(90.86), LogisticRegression(89.80), RandomForestClassifeir(99.99), GradientBoostingClassifier(99.99), AdaBoostClassifier(99.99).

**Experiments and Results**:

The smoke detection dataset contains 16 columns in total which are mentioned below in detail.

Feature Description:

The project consists of only numerical features; there are no categorical features. For example

1. UTC - The time when the experiment was performed.

2. Raw H2 - The amount of Raw Hydrogen present in the surroundings.

3. Temperature -Surroundings temperature in Celsius

4. Humidity - The air humidity during the experiment.

5. TVOC - Total Volatile Organic Compounds in ppb (parts per billion)

6. eCo2 - CO2 equivalent concentration in ppm (parts per million)

7. Raw Ethanol - The amount of Raw Ethanol present in the surroundings.

8. Pressure - Air pressure. Measured in hPa

9. PM1.0 - Particulate matter of diameter less than 1.0 micrometer .

10. PM2.5 - Particulate matter of diameter less than 2.5 micrometers.

11. NC0.5 - Concentration of particulate matter of diameter less than 0.5 micrometers.

12. NC1.0 - Concentration of particulate matter of diameter less than 1.0 micrometers.

13. NC2.5 - Concentration of particulate matter of diameter less than 2.5 micrometers.

14. CNT - Simple Count.

15. Fire Alarm - (Reality) If the fire was present then the value is 1 else it is 0.

16. Unnamed: index

This dataset contains 62630 data entries and there are no null values in any of the rows which we confirmed using the function "isnull" offered by the pandas library. After going through the data and all the features we can conclude that not all the features will be helpful to us in predicting the target variable which is "0" if no fire is detected and "1" if the fire is detected. The 3 features namely, "Unnamed: 0" ,"UTC" and "CNT" are of no particular use for this classification task as they are the index, the timestamp at which the sample was collected the sample counter, none of which helps in determining the target variable.

*Correlation matrix*:

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses. Key decisions to be made when creating a correlation matrix include: choice of correlation statistic, coding of the variables, treatment of missing data, and presentation.

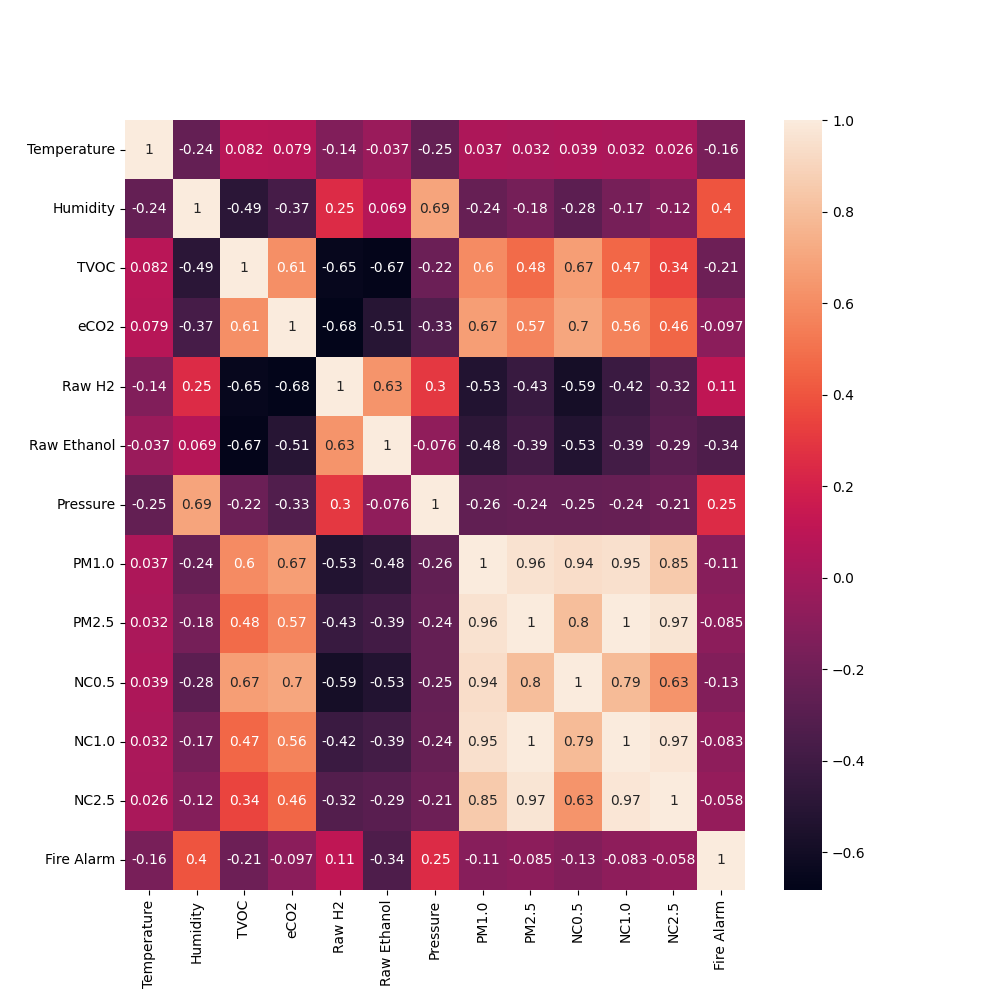
Typically, a correlation matrix is “square”, with the same variables shown in the rows and columns. I've shown an example below. This shows correlations between the stated importance of various things to people. The line of 1.00s going from the top left to the bottom right is the main diagonal, which shows that each variable always perfectly correlates with itself. This matrix is symmetrical, with the same correlation shown above the main diagonal being a mirror image of those below the main diagonal.

There are three broad reasons for computing a correlation matrix:

⦁ To summarize a large amount of data where the goal is to see patterns. In our example above, the observable pattern is that all the variables highly correlate with each other.

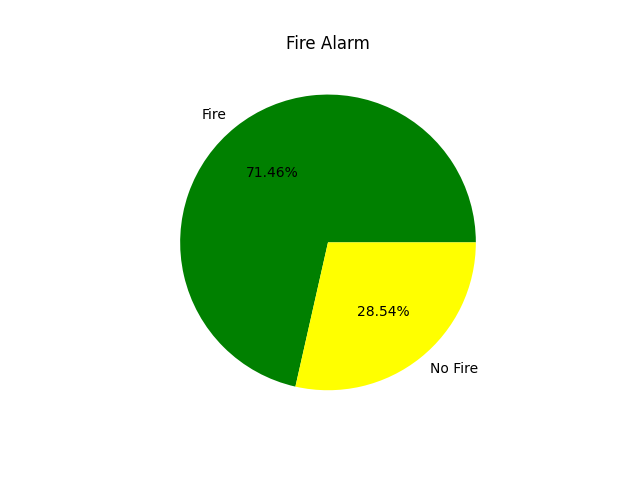
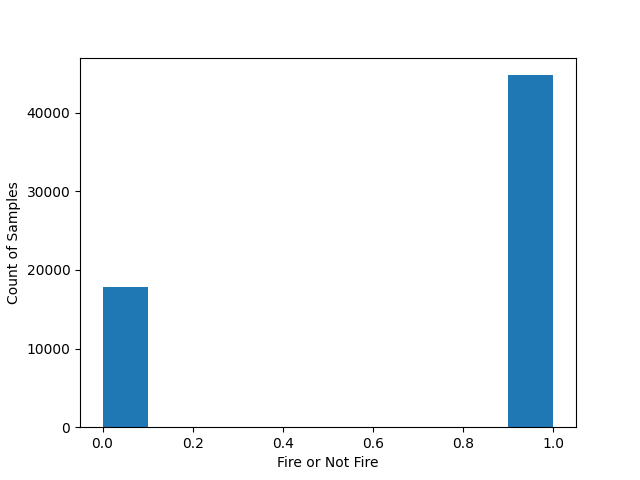
⦁ To input into other analyses. For example, people commonly use correlation matrices as inputs for exploratory factor analysis, confirmatory factor analysis, structural equation models, and linear regression when excluding missing values pairwise.

⦁ As a diagnostic when checking other analyses. For example, with linear regression, a high amount of correlations suggests that the linear regression estimates will be unreliable.



From the above correlation plot we can infer that the features of "PM2.5", "NC1.0" and "NC2.5" are highly correlated to each other and therefore we can drop these columns as they provide no more information and can be considered as redundant data.

Next step is to get the count of the classes that we have as the target variable. We can easily achieve this by using matplotlib as well as seaborn libraries.



According to both the plots, we can see that our dataset is highly skewed towards the "Fire" class which is represented as 1 in our dataset. In fact, the value count of class 1 is 44757 and the value count of class 0 is 17873 which is a 2.5:1 ratio for the classes. We will try to bridge this gap between classes with the help of a method called SMOTE.

*Splitting the dataset into training and testing set:*

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

⦁ Train Dataset: Used to fit the machine learning model.

⦁ Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model. This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values.

The scikit-learn Python machine learning library provides an implementation of the train-test split evaluation procedure via the train\_test\_split() function. We used the same function for our dataset with a 70% : 30% split, 70% for the training set and 30% for the testing set.



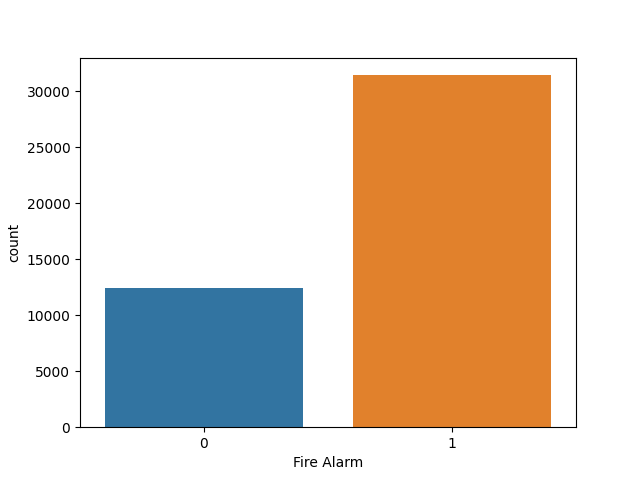
This is the shape of our train and test splits.

*SMOTE (Synthetic Minority Oversampling Technique):*

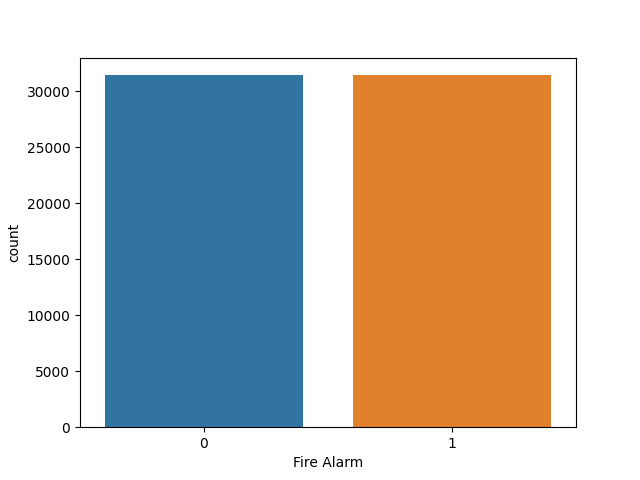
A problem with imbalanced classification is that there are too few examples of the minority class for a model to effectively learn the decision boundary. One way to solve this problem is to oversample the examples in the minority class. This can be achieved by simply duplicating examples from the minority class in the training dataset prior to fitting a model. This can balance the class distribution but does not provide any additional information to the model.

An improvement on duplicating examples from the minority class is to synthesize new examples from the minority class. This is a type of data augmentation for tabular data and can be very effective. Perhaps the most widely used approach to synthesizing new examples is called the Synthetic Minority Oversampling TEchnique, or SMOTE for short. This technique was described by Nitesh Chawla, et al. in their 2002 paper named for the technique titled “SMOTE: Synthetic Minority Over-sampling Technique.”

SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found (typically k=5). A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.



This is the countplot of our dataset before applying SMOTE technique on it. We can clearly see the imbalance of our dataset in the plot and we will try to balance this in the next step. SMOTE is already applied in the imblearn library and we are going to use the same for our requirement. We are going to create new samples for the minority class instead of trimming the majority class and risking losing valuable information.



The above countplot is after we have applied the SMOTE and balanced our dataset. We can see that our classes are balanced now with no skew to either class.

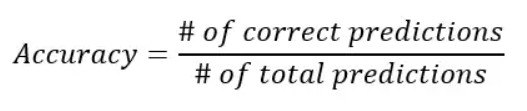
The Seaborn Pairplot function allows the users to create an axis grid via which each numerical variable stored in data is shared across the X- and Y-axis in the structure of columns and rows. We can create the Scatter plots in order to display the pairwise relationships in addition to the distribution plot displaying the data distribution in the column diagonally.

The pairplot() function can also be used to showcase the subset of variables, or we can plot different types of variables on rows and columns. We have used the pairplot() function to visualize the plot with the kind parameter set to the value 'kde'.

*Model Comparison:*

We have trained 11 different models in total and the comparison between their performance is made on the basis of their accuracy score, f1 score and AUC (area under the curve) score.

Accuracy is a metric for classification models that measures the number of predictions that are correct as a percentage of the total number of predictions that are made.

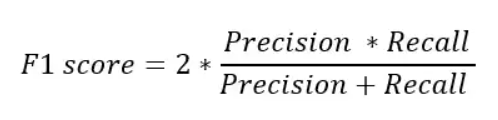


Accuracy is a useful metric only when you have an equal distribution of classes on your classification. This means that if you have a use case in which you observe more data points of one class than of another, the accuracy is not a useful metric anymore.

The F1 score is defined as the harmonic mean of precision and recall.

As a short reminder, the harmonic mean is an alternative metric for the more common arithmetic mean. It is often useful when computing an average rate. In the F1 score, we compute the average of precision and recall. They are both rates, which makes it a logical choice to use the harmonic mean. The F1 score formula is shown here:

This makes that the formula for the F1 score is the following:



Since the F1 score is an average of Precision and Recall, it means that the F1 score gives equal weight to Precision and Recall:

⦁ A model will obtain a high F1 score if both Precision and Recall are high

⦁ A model will obtain a low F1 score if both Precision and Recall are low

⦁ A model will obtain a medium F1 score if one of Precision and Recall is low and the other is high

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

⦁ True Positive Rate

⦁ False Positive Rate

True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:

TPR = TP / TP+ FN

False Positive Rate (FPR) is defined as follows:

FPR = FP / FP+TN

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

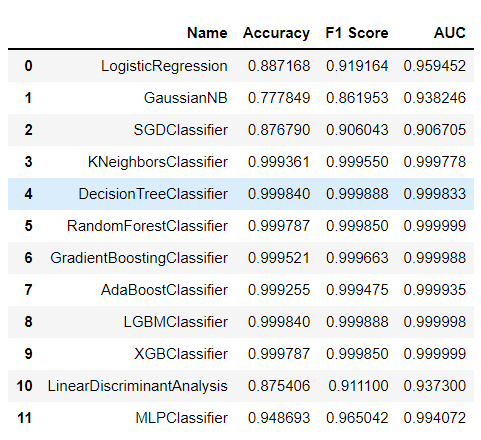
AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1). AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

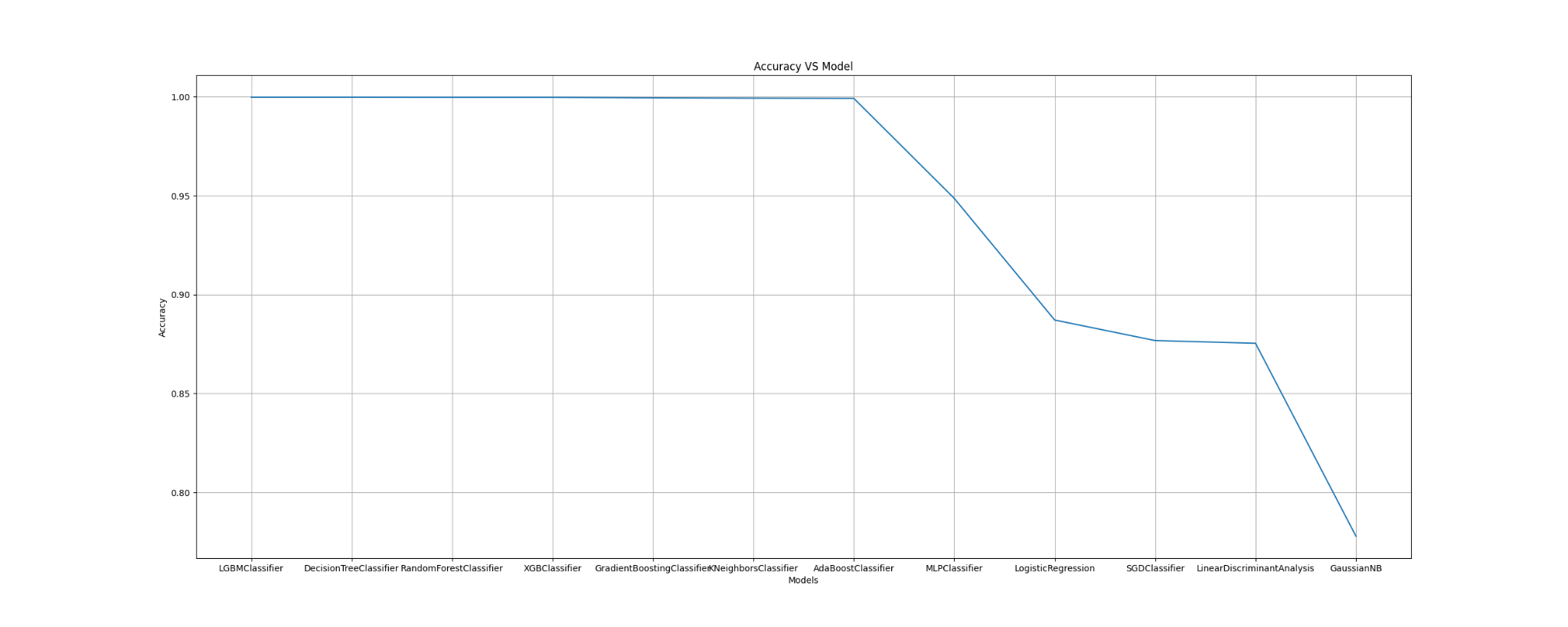
AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

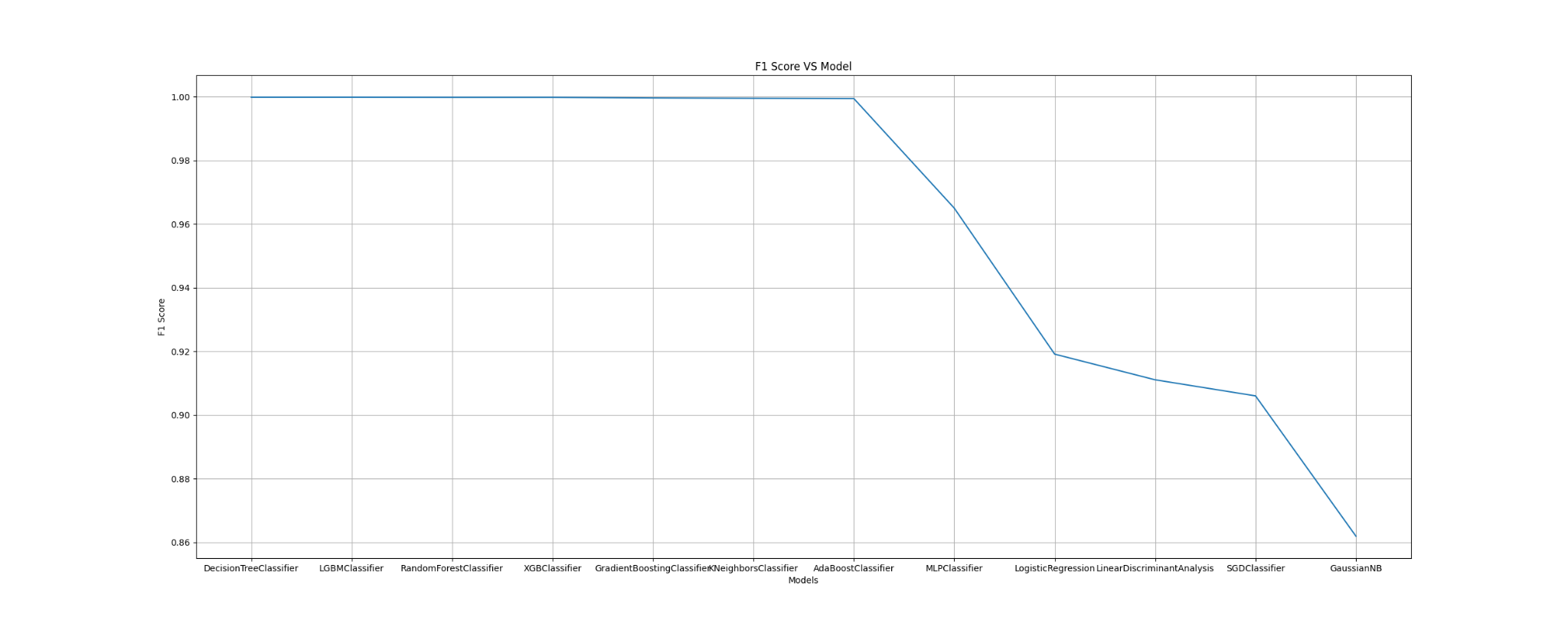
The models that we are going to take a look at are :

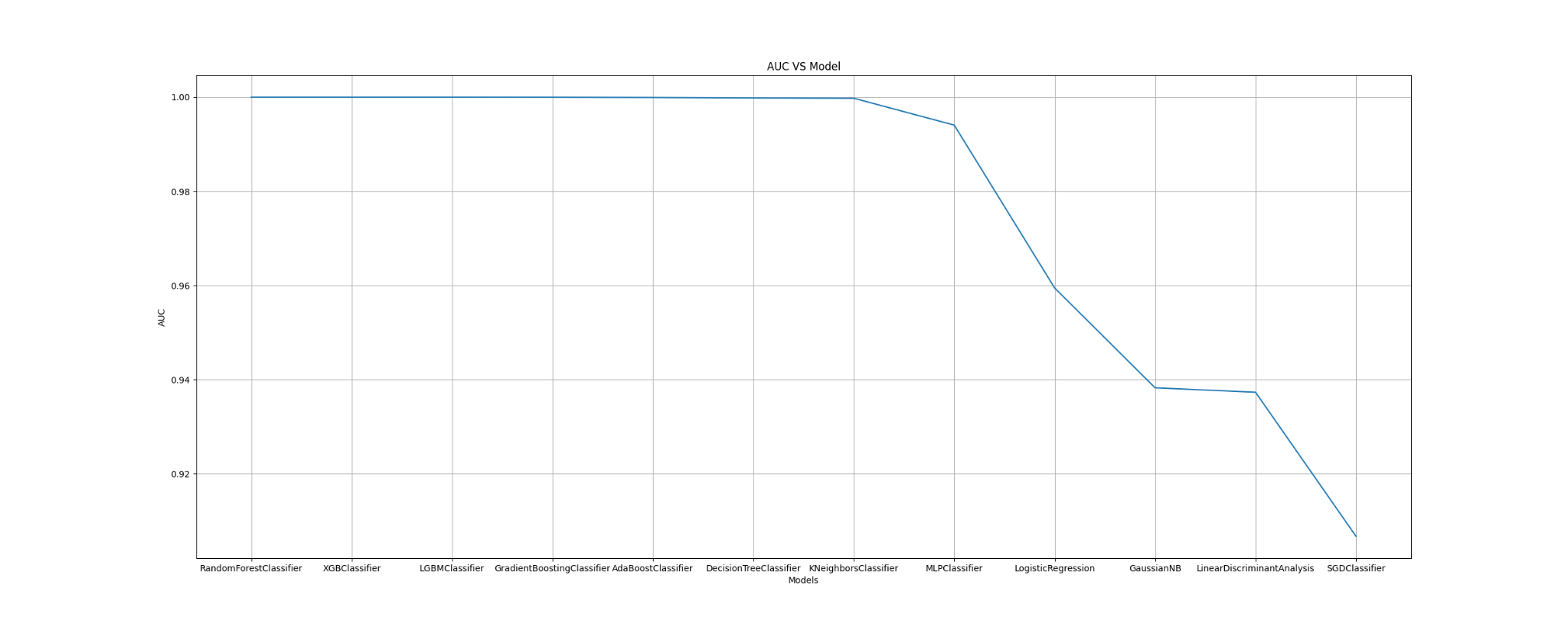
* The logistic regression model is the statistical analysis that predicts the dependent data variable by analyzing the relationship between one or more existing independent variables. It predicts binary outcomes such as yes on no.
* The Gaussuian Naïve Bayes is a variant of Naïve Bayes that follow the Gaussian normal distribution and supports continuous data. It is the classification technique which has high functionality and is used to find when the dimensionality of inputs is high.
* SGD Classifier is a linear classifier (svm, logistic regression) optimized by the SGD which implements a plain stochastic gradient descent learning routine and supports the different loss functions and penalties for the classification.
* The KNeighborsClassifier looks for the nearest neighbors. It is also called KNN supervised learning which uses proximity to make predictions or classifications about the grouping of an individual data point.
* DecisionTreeClassifier: It is the flowchart-like structure where each internal mode denotes the rest on an attribute and the individual branch represents the outcome of the rest and each leaf node. It has the ability to use different feature subsets and decision rules at different stages of classification.
* Randomforestclassifier: It is an ensemble method that can be used for a variety of tasks including regression and classifications. It is made up of large numbers of small decision trees and called estimators which produce their own predictions.
* GradientBoostingClassifier; This algorithm allows the optimization of arbitrary differentiable loss functions. It builds an additive model in a forward stage-wise fashion.
* AdaBoostClassifier: It is one of the algorithms of boosting which is also called Adaptive boosting. It helps to combine multiple weak classifiers to increase the accuracy of classifiers. Basically, it sets the weights of the classifier and trains the data sample in each iteration such that it ensures accurate predictions of unusual observations.
* LGBMClassifier: The light gradient boosting machine is a framework and variant of gradient boosting which is based on decision tree algorithms. It helps to reduce memory usage and can increase efficiency.
* XGBClassifier: The eXtreme Gradient Boosting is an implementation of gradient boosted decision trees designed for speed and performance.
* LinearDiscriminantAnalysis: Linear discriminant analysis is the Type of liner combination which predicts the modeling of algorithms for multi-class classification which can be used as a dimensionality reduction technique, providing a projection of training dataset

The performance metrics that we discussed earlier are used to evaluate the above mentioned models. We will discuss the performance of the models now:







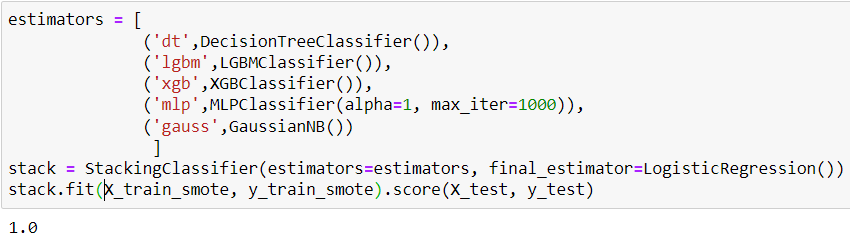


From the above table we can conclude that the best performing models on our dataset are KNeighbors classifier, Decision tree, random forest, gradient boosting, adaboost, LGBM and XGB classifiers which all have close to 100% percent accuracy.

We tried to stack models to improve the performance. Stacking Classifier is a stack of estimators with a final classifier.

Stacked generalization consists in stacking the output of an individual estimator and using a classifier to compute the final prediction. Stacking allows us to use the strength of each individual estimator by using their output as input of a final estimator.

Stacking is an ensemble learning technique to combine multiple classification models via a meta-classifier. The individual classification models are trained based on the complete training set; then, the meta-classifier is fitted based on the outputs -- meta-features -- of the individual classification models in the ensemble. The meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.



**Conclusion:**

To conclude we displayed all the results such as accuracy, f1 score, auc score and classification report. We also did 5-fold cross validation on the trained stacking classifier.

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

