Classification Model on MODIS Active Fire Data- India

This code helps us predict the type of forest fire detected by MODIS using Classification algorithms.

MODIS (or Moderate Resolution Imaging Spectroradiometer) is a key instrument aboard the Terra (originally known as EOS AM-1) and Aqua (originally known as EOS PM-1) satellites.

Forest, bush, or vegetarian fire, can be described as any uncontrolled and non-prescribed combustion or burning of plants in a natural setting such as a forest, grassland. In this notebook, we are trying to predict the type of forest fire which can be a hurdle considering the vast vegetation and forest area in different parts of the world with the help of **Active Fire Data** provided by NASA.

This report highlights the ML algorithms used and it's analysis, while in the end we discuss the best suited algorithm for the prediction.

About the data:

MODIS Active Fire Data of India year 2021:

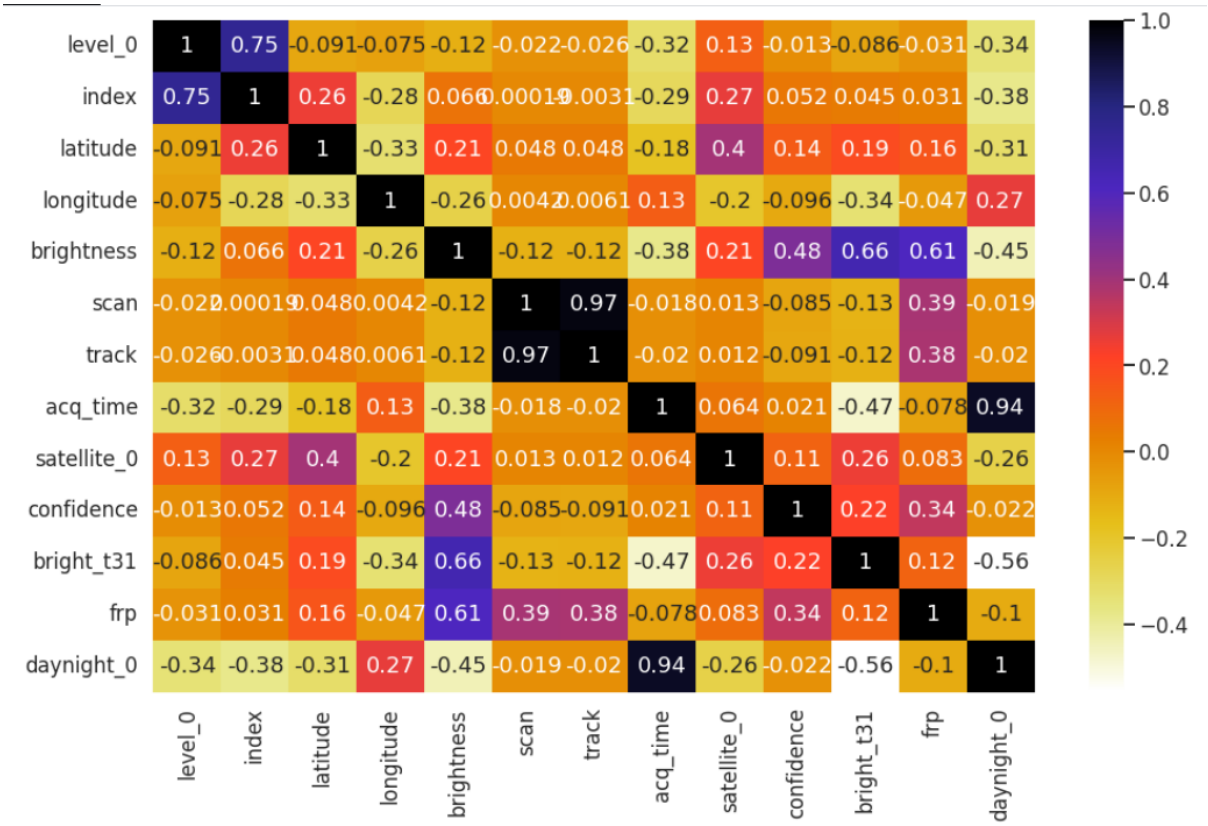
It consists of:

- **`latitude`:** Latitude of the fire pixel detected by the satellite *(degrees)*
- **`longitude`**: Longitude of the fire pixel detected by the satellite *(degrees)*
- **`brightness`:** Brightness temperature of the fire pixel *(in K)*
- **`scan`:** Area of a MODIS pixel at the Earth's surface *(Along-scan: ΔS)*
- **`track`:** Area of a MODIS pixel at the Earth's surface *(Along-track: ΔT)*
- **`acq_time`:** Time at which the fire was detected
- **`satellite`:** Satellite used to detect the fire. Either Terra(T) or Aqua(A)
- **`instrument`**: MODIS (used to detect the forest fire)
- **`confidence`:** Detection confidence *(range 0-100)*
- **`bright_t31`:** Band 31 brightness temperature of the pixel *(in K)*
- **`frp`:** Fire radiative power *(in MW- megawatts)*
- **`daynight`:** Detected during the day or night. Either Day(D) or Night(N)
- **`type`:** Inferred hot spot type:
  - `0= presumed vegetation fire`
  - `1= active volcano`
  - `2= other static land source`
  - `3= offshore`

We will be predicting the **Type** feature.

After some preprocessing, we split the dataset using **Train-Test split** of sklearn with 80% training and 20% for testing.

Correlation between the features:



Machine Learning Algorithms:

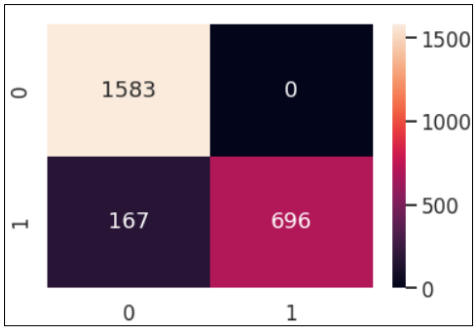
We make the use of 3 classification algorithm:

- Logistic Regression
- KNN
- XG Boost

Logistic Regression:

Here, we use the *liblinear solver* with *l2 regularization* which doesn't make much of a difference without it, but it gives us lesser computational complexity.

Confusion Matrix:



The confusion matrix for Logistic Regression depicts the following:

- No type 0s are predicted incorrectly. (Hence recall is 1.0 for type 0)
- 167 type 1s are predicted as 0, which is almost 20% of the type 2s predicted incorrectly.
- This is due to the lesser number of type 2s than type 0s.

Overall classification report:

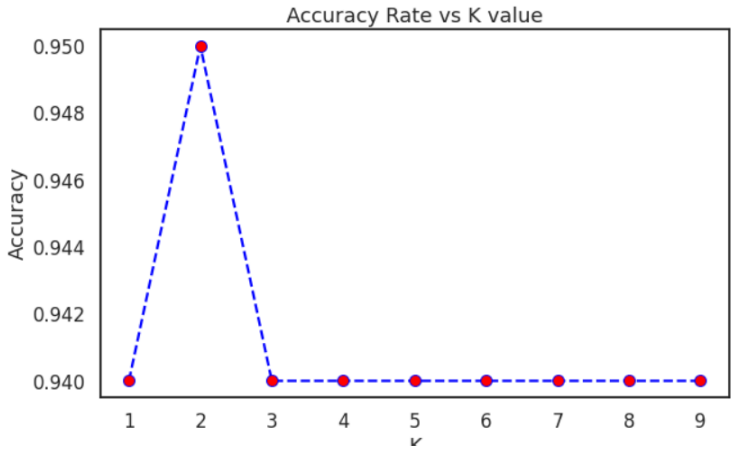
	precision	recall	f1-score	support
0	0.90	1.00	0.95	1583
2	1.00	0.81	0.89	863
accuracy			0.93	2446
macro avg	0.95	0.90	0.92	2446
weighted avg	0.94	0.93	0.93	2446

Accuracy score: 0.93

The final accuracy of Logistic Regression Model is 93%.

K Nearest Neighbours:

To select the K value, we first iterate between a number of K values, and find the accuracy rate for each K value and plot it.



This graph depicts that K=2 gives us the maximum accuracy, after which the accuracy is consistent. Hence, we use K=2 to fit our model.

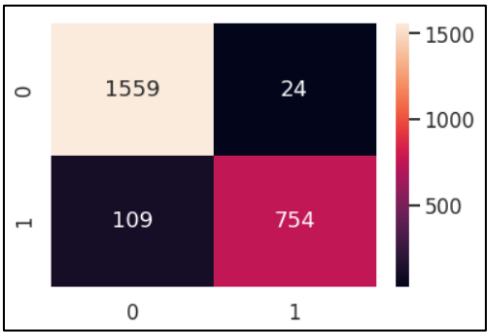
Now we calculate the accuracy.

	precision	recall	f1-score	support
0	0.93	0.98	0.96	1583
2	0.97	0.87	0.92	863
accuracy			0.95	2446
macro avg	0.95	0.93	0.94	2446
weighted avg	0.95	0.95	0.94	2446

Accuracy score: 0.95

Final accuracy for KNN is 95%

Taking a look at the confusion matrix:



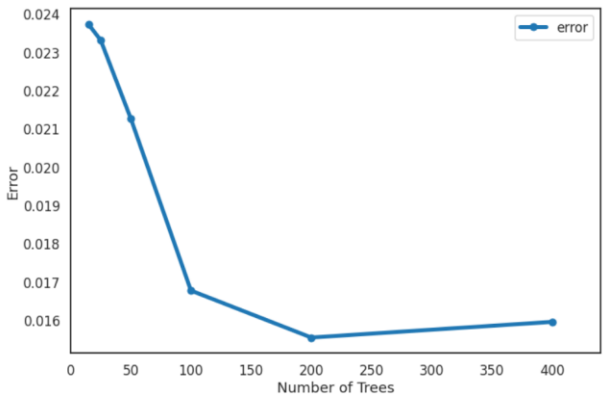
The number of type 0s predicted incorrectly are fairly higher than that of Logistic Regression that has a recall of 1.0 for type 0.

XG Boost:

I thought of choosing a Boosting algorithm over Bagging, looking at the performance of the first two algorithms, there was a need to decrease the bias.

Since the dataset has around 12,000 rows, overfitting isn't a problem. Hence a Bagging algorithm wouldn't have been much of a help.

For XG Boost, to find the best estimator, like we did for KNN, we will first iterate over a list of number of trees for our boosting model and plot it on a graph with the error.



As shown in the graph above, for *n_estimators = 200*, lowest error is shown.

To make the best use of all the hyperparameters, we cannot possibly iterate over the combinations of all possible parameters, hence we make the use of the library **GridSearchCV** provided by sklearn, to find the best estimator.

```
GV_GBC.best_estimator_
```

```
GradientBoostingClassifier(max_features=4, n_estimators=200, random_state=42)
```

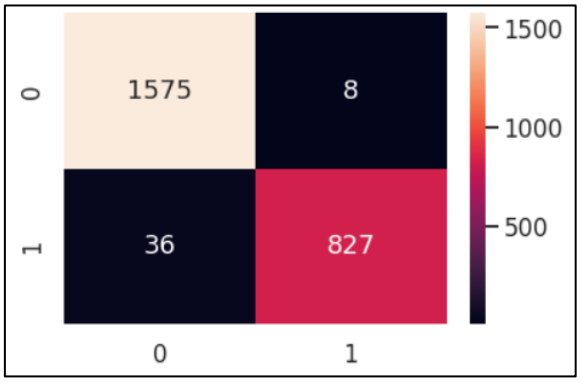
We get max_features= 4 and n_estimators=200 (As calculated by us in the above

Therefore, we use these parameters to get the accuracy of our model.

	precision	recall	f1-score	support
0	0.99	0.98	0.99	1611
2	0.96	0.99	0.97	835
accuracy			0.98	2446
macro avg	0.98	0.98	0.98	2446
weighted avg	0.98	0.98	0.98	2446

The classification report for XG Boost shows us the best results so far, with an accuracy of 98% and a higher precision and recall.

Confusion matrix for XG Boost:



Less than less than 2% or readings predicted incorrectly.

Conclusion:

It is apparent that the performance of **XG Boost** is far ahead from that of Logistic Regression and KNN. It will be the best suited algorithm with an accuracy of 98% for our prediction on the MODIS Forest Fire dataset.

Analysis:

The problem with the first two algorithms- Logistic Regression and KNN was underfitting (High Bias). XG Boost helps decrease the bias by using multiple weak models and correcting the errors of the same, thus creating an accurate model.