Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Fall Semester (2023 - 2024)
PHY1901: Introduction to Innovative Projects

Project Report
on

AN EFFICIENT AND OPTIMAL ML ALGORITHM FOR REAL-TIME FOREST FIRE PREDICTION

Submitted by
Vijay Adithya R P - 20MIS0164
Sangaraju Lakshmi Lahari - 20MIS0169
Sabarinath R - 20MIS0194
Ashwath B - 20MIS0212
Kalikivayi Abhiram - 20MIS0220
Torai Abhiram Goud - 20MIS0231
Mamilla Sai Bhargav - 20MIS0241
Inukurthi Suneel Kumar - 20MIS0246

Submitted to:
Prof .Gopal

Abstract :

Forest fires pose a significant threat to both the environment and human safety, necessitating the development of accurate and efficient predictive models to mitigate their impact.Fuel, oxygen, and a heat source are the three elements necessary for a forest fire to ignite. Fuel is anything that can catch fire around a fire, including trees, grass, bushes, and even buildings. The more fuel there is, the more intense the fire becomes. Air provides the oxygen needed for a fire to ignite.

To mitigate these effects, forest fire prediction and prevention ,real-time machine learning (ML) algorithm is proposed to analyse temperature and weather data.The proposed algorithm combines data preprocessing, Feature Selection, ML model training and evaluation

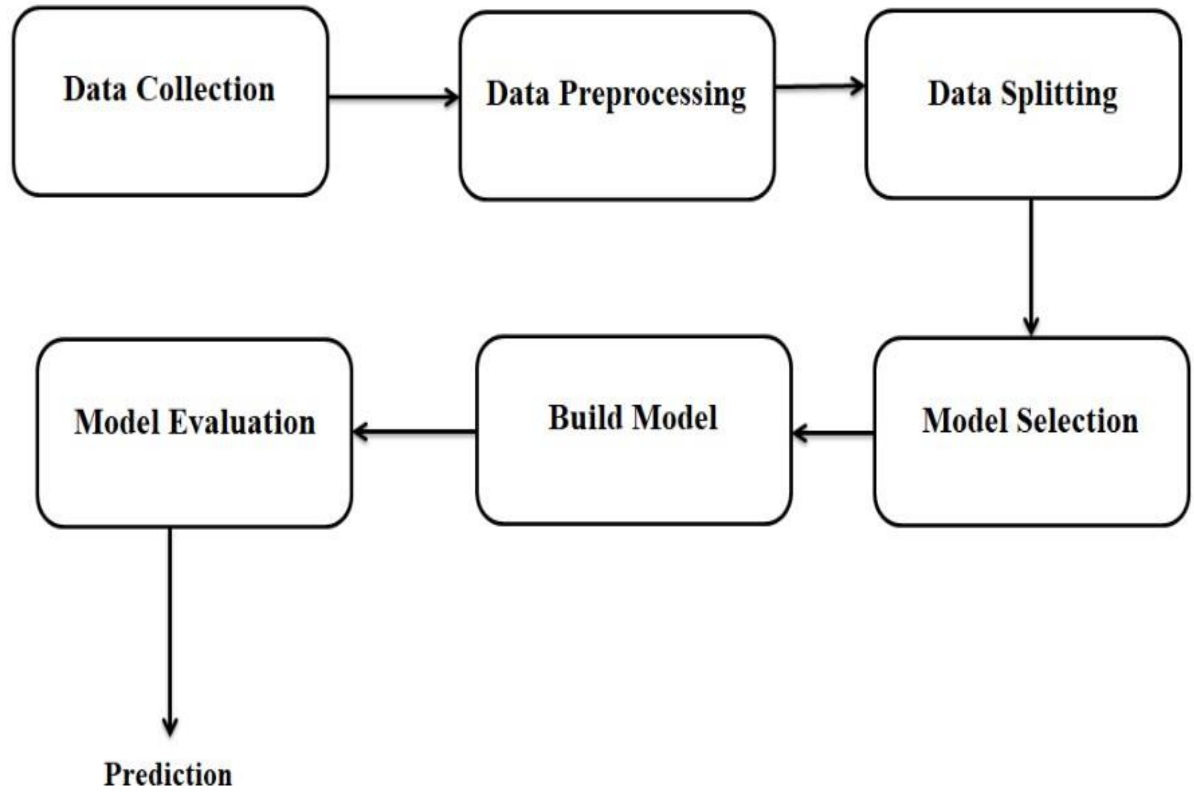
Introduction:

We are presenting an innovative machine learning (ML) algorithm designed for real-time forest fire prediction. The proposed algorithm leverages a combination of data preprocessing, Creating the Machine Learning model, Creating the web application to deploy our model. We will be analysing the temperature, weather and predict the chances of forest fires.

The key features of our algorithm include data-driven model training, utilising historical fire data, weather conditions, and topographical information. We employ state-of-the-art ML algorithms, such as deep neural networks and ensemble methods, to enhance prediction accuracy. Furthermore, the algorithm is optimised for real-time processing, ensuring timely warnings to relevant authorities and communities.

To evaluate the performance of our algorithm, we conducted extensive experiments using diverse datasets from various geographical regions. Our results demonstrate superior prediction accuracy and reduced false alarms compared to existing methods. Additionally, the algorithm's efficiency enables it to process data in real-time, making it a valuable tool for early forest fire detection and prevention.

Methodology Proposed:



1) DATA COLLECTION:

- Gathering historical fire incident data,which includes the location,date,past forest fires information.
- Collect meteorological data such as temperature,humidity,wind speed, Moisture etc..,
- We took a dataset named Algerian forest fires dataset contains all these attributes.

2) DATA PREPROCESSING:

- Clean and preprocess the collected data, addressing missing values, outliers, and inconsistencies.
- Perform feature engineering to extract relevant information, create new features, and transform data as needed.
- Normalise or scale numerical features to ensure consistency and improve the performance of machine learning models.

3) DATA SPLITTING:

- The data processed will be divided into training and testing sets.
- The training set is the portion of the dataset used to train the machine learning model. (75%)
- The testing set, also known as the validation set, is used to assess the performance of the trained model.(25%)

4) MODEL SELECTION:

- Choosing the suitable machine learning model from different machine learning models like decision trees,random forest,XGBoost, Logistic Regression etc..,
- For our project we took the XGBoost model as it gave the highest accuracy.

5) BUILD MODEL

- We built the forest fire prediction model.
- XGBoost is the chosen model due to its high accuracy in capturing complex data patterns.
- This model's robustness and precision are vital for timely forest fire warnings and effective mitigation.

6) MODEL EVALUATION

- We evaluate the XGBoost model's performance using metrics like accuracy, precision, and recall.
- The model's generalizability is assessed through techniques like k-fold cross-validation.
- We compare the model's results to existing methods, emphasising its capacity to reduce false alarms and improve prediction accuracy.

DATASET DESCRIPTION:

The dataset includes 244 instances that group a data of region Algeria,namely the Bejaia region located in the northeast of Algeria.It includes 13 attributes namely Day, Month ,Temperature, RH, Ws, Rain, FFMC, DMC, DC, ISI, BUI, FWI, Classes.Based on the first 12 attributes we decide the classes attribute and then classify them as "not fire" and "fire" according to the data provided for each attribute in the dataset.

RESULTS AND EXPECTED OUTCOMES:

1) DATA COLLECTION:

jupyter IIP IMPL Last Checkpoint: 14 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
import collections
from collections import Counter
from sklearn import preprocessing
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict
```

In [2]:
```
data = pd.read_csv('Algerian_forest_fires_dataset_UPDATE(1).csv')
data
```

Out[2]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
0	1	6	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	'not fire '
1	2	6	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	'not fire '
2	3	6	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	'not fire '
3	4	6	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	'not fire '
4	5	6	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	'not fire '
...	...	...	...	...	...	...	...	...	...	...	...	...	...
239	25	8	35	60	15	0.0	88.9	43.9	181.3	8.2	54.7	20.3	'fire '
240	27	8	33	82	21	0.0	84.9	47.0	200.2	4.4	59.3	13.2	'fire '
241	20	9	28	84	18	0.0	83.8	13.5	49.3	4.5	16.0	6.3	'fire '
242	26	8	31	78	18	0.0	85.8	45.6	190.6	4.7	57.1	13.7	'fire '
243	28	8	34	64	16	0.0	89.4	50.2	210.4	7.3	62.9	19.9	'fire '

244 rows × 13 columns

jupyter IIP IMPL Last Checkpoint: 15 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [3]:
```
# checking the dataype of the parameters(Columns)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          244 non-null    int64
 1   month        244 non-null    int64
 2   Temperature  244 non-null    int64
 3   RH           244 non-null    int64
 4   Ws           244 non-null    int64
 5   Rain         244 non-null    float64
 6   FFMC         244 non-null    float64
 7   DMC          244 non-null    float64
 8   DC           244 non-null    float64
 9   ISI          244 non-null    float64
 10  BUI          244 non-null    float64
 11  FWI          244 non-null    float64
 12  Classes      244 non-null    object
dtypes: float64(7), int64(5), object(1)
memory usage: 24.9+ KB
```

In [4]:
```
# Descriptive Analysis
data.describe()
```

Out[4]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI
count	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000	244.000000
mean	15.754098	7.500000	32.172131	61.938525	15.504098	0.760656	77.887705	14.673361	49.288115	4.759836	16.673361	7.049180
std	8.825059	1.112961	3.633843	14.884200	2.810178	1.999406	14.337571	12.368039	47.619662	4.154628	14.201648	7.428366
min	1.000000	6.000000	22.000000	21.000000	6.000000	0.000000	28.600000	0.700000	6.900000	0.000000	1.100000	0.000000
25%	8.000000	7.000000	30.000000	52.000000	14.000000	0.000000	72.075000	5.800000	13.275000	1.400000	6.000000	0.700000

jupyter IIP IMPL Last Checkpoint: 17 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [5]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
data['Classes'] = le.fit_transform(data['Classes'])
```

In [6]:
```
mean = data.mean()
mean
```

Out[6]:
```
day            15.754098
month           7.500000
Temperature    32.172131
RH             61.938525
Ws             15.504098
Rain            0.760656
FFMC           77.887705
DMC            14.673361
DC             49.288115
ISI             4.759836
BUI            16.673361
FWI             7.049180
Classes         0.434426
dtype: float64
```

In [7]:
```
# Checking the null values in dataset
data.isnull().any()
```

Out[7]:
```
day            False
month          False
Temperature    False
RH             False
Ws             False
Rain           False
FFMC           False
DMC            False
DC             False
ISI            False
BUI            False
```

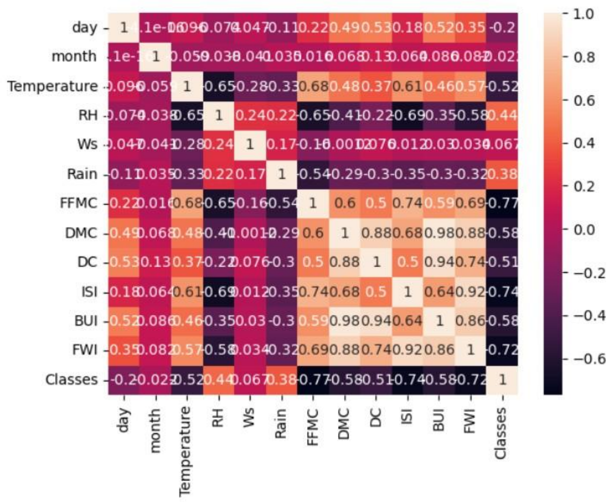
2) VISUALIZATION:

jupyter IIP IMPL Last Checkpoint: 19 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [9]:
```
#Heat Map
hm=data.corr()
sns.heatmap(hm,annot=True)
```

Out[9]: <Axes: >



3) DATA PREPROCESSING:

jupyter IIP IMPL Last Checkpoint: 20 minutes ago (autosaved)

In [10]:
```
#extracting numerical columns values
x_independent = data.iloc[:,:-1]
y_dependent=data.iloc[:,12:13]
x_independent
```

Out[10]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI
0	1	6	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5
1	2	6	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4
2	3	6	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1
3	4	6	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0
4	5	6	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5
...	...	...	...	...	...	...	...	...	...	...	...	...
239	25	8	35	60	15	0.0	88.9	43.9	181.3	8.2	54.7	20.3
240	27	8	33	82	21	0.0	84.9	47.0	200.2	4.4	59.3	13.2
241	20	9	28	84	18	0.0	83.8	13.5	49.3	4.5	16.0	6.3
242	26	8	31	78	18	0.0	85.8	45.6	190.6	4.7	57.1	13.7
243	28	8	34	64	16	0.0	89.4	50.2	210.4	7.3	62.9	19.9

244 rows × 12 columns

In [11]:
```
y_dependent
```

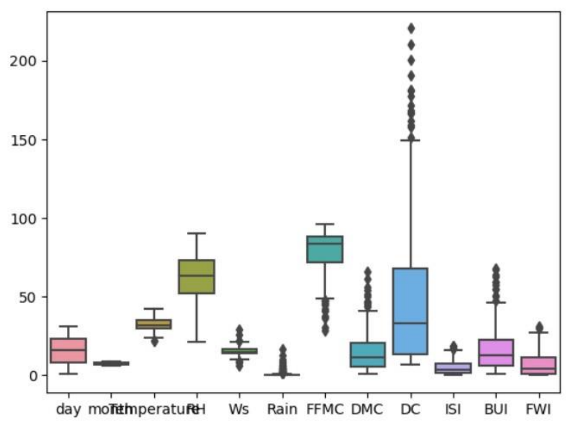
Out[11]:

	Classes
0	1
1	1
2	1

jupyter IIP IMPL Last Checkpoint: 22 minutes ago (autosaved)

In [13]:
```
sns.boxplot(x_independent)
```

Out[13]: <Axes: >



jupyter IIP IMPL Last Checkpoint: 22 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Markdown

In [14]:
```
# #Calculating quartiles for x_independent
quantile = x_independent.quantile(q=[0.25,0.75])
quantile
```

Out[14]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI
0.25	8.0	7.0	30.0	52.00	14.0	0.0	72.075	5.80	13.275	1.4	6.000	0.700
0.75	23.0	8.0	35.0	73.25	17.0	0.5	88.300	20.75	68.150	7.3	22.525	11.375

In [15]:
```
# #IQR
IQR = quantile.iloc[1] - quantile.iloc[0]
IQR
```

Out[15]:
```
day            15.000
month           1.000
Temperature     5.000
RH             21.250
Ws              3.000
Rain            0.500
FFMC           16.225
DMC            14.950
DC             54.875
ISI             5.900
BUI            16.525
FWI            10.675
dtype: float64
```

In [16]:
```
#calculating upper extreme
upper_extreme = quantile.iloc[1] + (1.5*IQR)
upper_extreme
```

Out[16]:
```
day             45.5000
month            9.5000
Temperature     42.5000
```

jupyter IIP IMPL Last Checkpoint: 23 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Markdown

In [18]:
```
#removing outliers from the extracted numeric columns

removed_outliers = x_independent[(x_independent >=lower_extreme)&(x_independent <=upper_extreme)]
removed_outliers.to_csv('file1.csv')
removed_outliers
```

Out[18]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI
0	1	6	29.0	57	18.0	0.0	65.7	3.4	7.6	1.3	3.4	0.5
1	2	6	29.0	61	13.0	NaN	64.4	4.1	7.6	1.0	3.9	0.4
2	3	6	26.0	82	NaN	NaN	NaN	2.5	7.1	0.3	2.7	0.1
3	4	6	25.0	89	13.0	NaN	NaN	1.3	6.9	0.0	1.7	0.0
4	5	6	27.0	77	16.0	0.0	64.8	3.0	14.2	1.2	3.9	0.5
...	...	...	...	...	...	...	...	...	...	...	...	...
239	25	8	35.0	60	15.0	0.0	88.9	NaN	NaN	8.2	NaN	20.3
240	27	8	33.0	82	21.0	0.0	84.9	NaN	NaN	4.4	NaN	13.2
241	20	9	28.0	84	18.0	0.0	83.8	13.5	49.3	4.5	16.0	6.3
242	26	8	31.0	78	18.0	0.0	85.8	NaN	NaN	4.7	NaN	13.7
243	28	8	34.0	64	16.0	0.0	89.4	NaN	NaN	7.3	NaN	19.9

244 rows × 12 columns

In [19]:
```
#Finding null values after removing outliers
removed_outliers.isnull().any()
```

Out[19]:
```
day            False
month          False
Temperature     True
RH             False
```

jupyter IIP IMPL Last Checkpoint: 24 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Markdown

In [20]:
```
#Replacing null values
removed_outliers['Temperature'].fillna(removed_outliers['Temperature'].mean(),inplace=True)
removed_outliers['Ws'].fillna(removed_outliers['Ws'].mean(),inplace=True)
removed_outliers['Rain'].fillna(removed_outliers['Rain'].mean(),inplace=True)
removed_outliers['FFMC'].fillna(removed_outliers['FFMC'].mean(),inplace=True)
removed_outliers['DMC'].fillna(removed_outliers['DMC'].mean(),inplace=True)
removed_outliers['ISI'].fillna(removed_outliers['ISI'].mean(),inplace=True)
removed_outliers['BUI'].fillna(removed_outliers['BUI'].mean(),inplace=True)
removed_outliers['FWI'].fillna(removed_outliers['FWI'].mean(),inplace=True)
removed_outliers['DC'].fillna(removed_outliers['DC'].mean(),inplace=True)
```

In [21]:
```
#Checking whether null values are removed
removed_outliers.isnull().sum()
```

Out[21]:
```
day            0
month          0
Temperature    0
RH             0
Ws             0
Rain           0
FFMC           0
DMC            0
DC             0
ISI            0
BUI            0
FWI            0
dtype: int64
```

In [22]:
```
removed_outliers
```

Out[22]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI
0	1	6	29.0	57	18.000000	0.000000	65.700000	3.40000	7.600000	1.3	3.400000	0.5

Lipynb#

4) SPLITTING THE DATA:

jupyter IIP IMPL Last Checkpoint: 27 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Markdown

Split the data into dependent and independent variables

In [30]:
```
x=X #independent values
y=y_dependent
```

In [31]:
```
x
```

Out[31]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI
0	0.000000	0.000000	0.277778	0.521739	0.727273	0.00000	0.360759	0.066832	0.004919	0.08125	0.050661	0.018587
1	0.033333	0.000000	0.277778	0.579710	0.272727	0.12799	0.333333	0.084158	0.004919	0.06250	0.061674	0.014870
2	0.066667	0.000000	0.111111	0.884058	0.497689	0.12799	0.671562	0.044554	0.001405	0.01875	0.035242	0.003717
3	0.100000	0.000000	0.055556	0.985507	0.272727	0.12799	0.671562	0.014851	0.000000	0.00000	0.013216	0.000000
4	0.133333	0.000000	0.166667	0.811594	0.545455	0.00000	0.341772	0.056931	0.051300	0.07500	0.061674	0.018587
...	...	...	...	...	...	...	...	...	...	...	...	...
239	0.800000	0.666667	0.611111	0.565217	0.454545	0.00000	0.850211	0.299302	0.238855	0.51250	0.295904	0.754647
240	0.866667	0.666667	0.500000	0.884058	1.000000	0.00000	0.765823	0.299302	0.238855	0.27500	0.295904	0.490706
241	0.633333	1.000000	0.222222	0.913043	0.727273	0.00000	0.742616	0.316832	0.297962	0.28125	0.328194	0.234201
242	0.833333	0.666667	0.388889	0.826087	0.727273	0.00000	0.784810	0.299302	0.238855	0.29375	0.295904	0.509294
243	0.900000	0.666667	0.555556	0.623188	0.545455	0.00000	0.860759	0.299302	0.238855	0.45625	0.295904	0.739777

244 rows × 12 columns

In [32]:
```
y
```

Out[32]:

	Classes

5) BUILD THE MODEL:

jupyter IIP IMPL Last Checkpoint: 28 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

Run Markdown

In [75]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

Train-Test Split

In [34]:
```
from sklearn.model_selection import train_test_split
```

In [35]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25, random_state=0)
```

Build the Model

In [36]:
```
x
```

Out[36]:

	day	month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI
0	0.000000	0.000000	0.277778	0.521739	0.727273	0.00000	0.360759	0.066832	0.004919	0.08125	0.050661	0.018587
1	0.033333	0.000000	0.277778	0.579710	0.272727	0.12799	0.333333	0.084158	0.004919	0.06250	0.061674	0.014870
2	0.066667	0.000000	0.111111	0.884058	0.497689	0.12799	0.671562	0.044554	0.001405	0.01875	0.035242	0.003717
3	0.100000	0.000000	0.055556	0.985507	0.272727	0.12799	0.671562	0.014851	0.000000	0.00000	0.013216	0.000000
4	0.133333	0.000000	0.166667	0.811594	0.545455	0.00000	0.341772	0.056931	0.051300	0.07500	0.061674	0.018587
...	...	...	...	...	...	...	...	...	...	...	...	...
239	0.800000	0.666667	0.611111	0.565217	0.454545	0.00000	0.850211	0.299302	0.238855	0.51250	0.295904	0.754647
240	0.866667	0.666667	0.500000	0.884058	1.000000	0.00000	0.765823	0.299302	0.238855	0.27500	0.295904	0.490706
241	0.633333	1.000000	0.222222	0.913043	0.727273	0.00000	0.742616	0.316832	0.297962	0.28125	0.328194	0.234201

6) MODEL EVALUATION:

XGBOOST:

jupyter IIP IMPL Last Checkpoint: 32 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

Run Markdown

XGBOOST

In [37]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\srihari inukurthi\anaconda3\lib\site-packages (2.0.1)
Requirement already satisfied: numpy in c:\users\srihari inukurthi\anaconda3\lib\site-packages (from xgboost) (1.23.5)
Requirement already satisfied: scipy in c:\users\srihari inukurthi\anaconda3\lib\site-packages (from xgboost) (1.10.0)
Note: you may need to restart the kernel to use updated packages.
```

In [38]:
```
from xgboost import XGBClassifier
import xgboost as xgb
```

In [39]:
```
xg= xgb.XGBClassifier(n_estimators=100)
```

In [40]:
```
xg.fit(x_train,y_train)
```

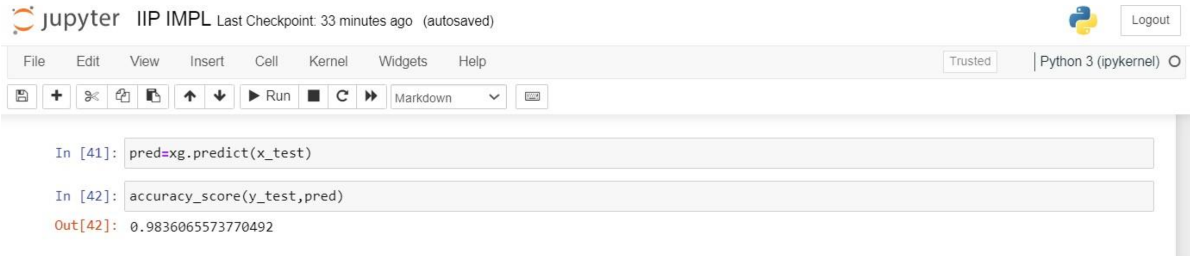
Out[40]:

XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
```

In [41]:
```
pred=xg.predict(x_test)
```

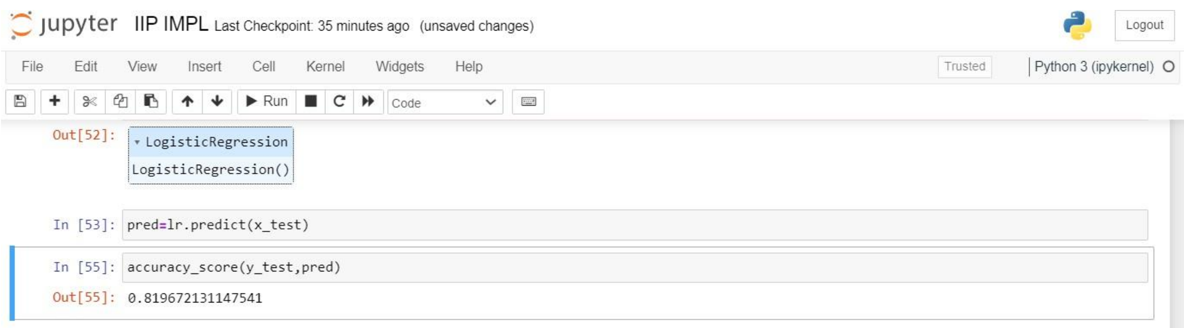
ACCURACY :



COMPARISON ANALYSIS :

RANDOM FOREST:

LOGISTIC REGRESSION :



DECISION TREE:

jupyter IIP IMPL Last Checkpoint: 36 minutes ago (autosaved)

```
Out[58]: DecisionTreeClassifier
         DecisionTreeClassifier(criterion='entropy', random_state=42)

In [59]: pred=df.predict(x_test)

In [60]: accuracy_score(y_test,pred)

Out[60]: 0.9508196721311475
```

KNN :

jupyter IIP IMPL Last Checkpoint: 36 minutes ago (autosaved)

```
Out[63]: KNeighborsClassifier
         KNeighborsClassifier()

In [64]: #test the model
         pred=knn.predict(x_test)

In [65]: accuracy_score(y_test,pred)

Out[65]: 0.8852459016393442
```

Ensemble Learning (Hybrid approach - RandomForest, KNN, XGBoost) :

jupyter IIP IMPL Last Checkpoint: 38 minutes ago (autosaved)

```
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)

Out[72]: VotingClassifier
         random_forest            xgboost          knn
         RandomForestClassifier   XGBClassifier    KNeighborsClassifier

In [73]: # Make predictions with the ensemble model
         ensemble_preds = voting_clf.predict(x_test)

In [74]: # Evaluate the performance of individual models and the ensemble
         print("Random Forest Accuracy:", accuracy_score(y_test, rf_preds))
         print("XGBoost Accuracy:", accuracy_score(y_test, xgb_preds))
         print("KNN Accuracy:", accuracy_score(y_test, knn_preds))
         print("Ensemble Accuracy:", accuracy_score(y_test, ensemble_preds))

         Random Forest Accuracy: 0.9672131147540983
         XGBoost Accuracy: 0.9836065573770492
         KNN Accuracy: 0.8852459016393442
         Ensemble Accuracy: 0.9672131147540983
```

Contribution:

Team Member	Reg Number	Contribution
Vijay Adithya R P	20MIS0164	Interdisciplinary Collaboration
Sangaraju Lakshmi Lahari	20MIS0169	Development of the Proposed ML algorithm and expected outcomes
Sabarinath R	20MIS0194	Writing and Documentation
Ashwath B	20MIS0212	Insights and Recommendations
Kalikivayi Abhiram	20MIS0220	Methodolgy Selection
Torai Abhiram Goud	20MIS0231	Project Management and Coordination
Mamilla Sai Bhargav	20MIS0241	Data Analysis and Interpretation
Inukurthi Suneel Kumar	20MIS0246	Development of the Proposed ML algorithm and expected outcomes

CONCLUSION :

In conclusion, forest fire prediction using ML models represents a significant advancement in safeguarding the environment, infrastructure, and human lives. By harnessing data-driven models and advanced prediction techniques, we enhance our ability to anticipate forest fire occurrences accurately. This, in turn, leads to timely warnings and swift responses, ultimately reducing the devastating impact of forest fires on ecosystems, infrastructure, and human safety. The potential to save lives, protect our environment, and minimize economic losses makes forest fire prediction using ML models an invaluable tool for forest management and disaster response agencies.The application of machine learning models for forest fire prediction represents a crucial step in mitigating the environmental and safety threats posed by these fires.High accuracy achievement by the utilization of **XGBoost ML Model**.This, in turn, leads to timely warnings and swift response, ultimately reducing the devastating impact of forest fires.

FUTURE WORKS:

Our upcoming initiatives encompass several key objectives. Firstly, we plan to deploy our meticulously trained machine learning models within a web application, thereby enabling real-time forest fire prediction. Ensuring the seamless handling of incoming data streams is a core priority to maintain timely and accurate predictions. Continuous performance monitoring will be integral to upholding the system's high accuracy and effectiveness. User feedback and updated data will serve as valuable guides for making necessary improvements. Furthermore, our vision includes the integration of sensor data, which promises to bolster our forest fire prediction capabilities and contribute to more proactive prevention measures. These collective endeavours aim to reinforce the reliability and utility of our forest fire prediction system.