PROJECT REPORT

Forest – Fire Detection

CS106

Basics of Machine Learning

Bachelor of Technology in

Computer Science & Engineering

DELHI TECHNOLOGICAL UNIVERSITY

SUBMITTED TO:	SUBMITTED BY:
Nipun Bansal Assistant Professor	Harshit Aggarwal (23/CS/167)
	Harshit Kushwaha (23/CS/169)
	Hitesh Mehta (23/CS/180)

INDEX

CANDIDATE'S DECLARATION

Harshit Aggarwal (23/CS/167), Harshit Kushwaha(23/CS/169), and Hitesh Mehta (23/CS/180) students of B.Tech. (CSE), hereby declare that our project report titled "Forest Fire Detection" which is submitted by us to the Department of Computer Science, Delhi Technological University, Delhi, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma, Fellowship or other similar title or recognition.

Place: DTU, Delhi

CERTIFICATE

This is to certify that this project report titled “Forest Fire Detection” is the bonafide work of Harshit Aggarwal (23/CS/167), Harshit Kushwaha(23/CS/169), and Hitesh Mehta (23/CS/180) submitted to the Department of Computer Science & Engineering, DTU, as part of project work carried out by the above students under my supervision. To the best of my knowledge, this work has not been submitted earlier in part or full for any Degree to this University or elsewhere.

Place: DTU, Delhi

ACKNOWLEDGEMENT

We would like to take this opportunity to express our heartfelt and warm gratitude to Nipun Bansal, Assistant Professor, our mentor who guided us in the making of this project, and the various faculty members of the Department of Computer Science & Engineering and administrative staff of Delhi Technological University who provided us with a renowned establishment to successfully gain education and create this project to its maximal potential. We would also like to express our gratitude to the university for providing the laboratories, infrastructure, test facilities and environment which allowed us to work without any obstructions. We would also like to appreciate the support provided by our seniors and peer group who aided us with all the knowledge they had regarding various topics.

A BRIEF OVERVIEW OF THE PROJECT

This project has been made by Harshit Aggarwal (23/CS/167), Harshit Kushwaha(23/CS/169), and Hitesh Mehta (23/CS/180) for completion of Lab Project of the course CS106, under the successful and grateful guidance of Nipun Bansal, Assistant Professor. Sure, here's an introduction you can use for a Python project on digit recognition using machine learning:

Project Title: Forest Fire Detection

Introduction:

Forest fires are major disasters harming both flora and fauna. They emit harmful gases, affecting human health. Wireless sensor nodes can detect fires, aiding in damage control. Machine learning enhances detection accuracy. This report compares ML techniques for predicting forest fires using sensor data.

Objective:

Earlier, fire detection relied on watching towers or satellite images, both slow and inefficient. Satellite images took time to reach authorities, and watching towers required constant monitoring, which was impractical for vast forest areas. These methods often resulted in delayed detection, allowing fires to spread extensively before intervention.

Approach:

The methodology begins with the acquisition of a dataset sourced from the UCI Machine Learning repository, encompassing index-based dataset derived from both image and video data. The image dataset comprises test and train data, each divided into three classes: default, smoke, and fire. Specifically, the test_default set contains data derived indices from 84 images, test_fire has 57 images, and test_smoke includes 30 images. On the other hand, the train_default set comprises 161 images, train_fire has 274 images, and train_smoke includes 258 images.

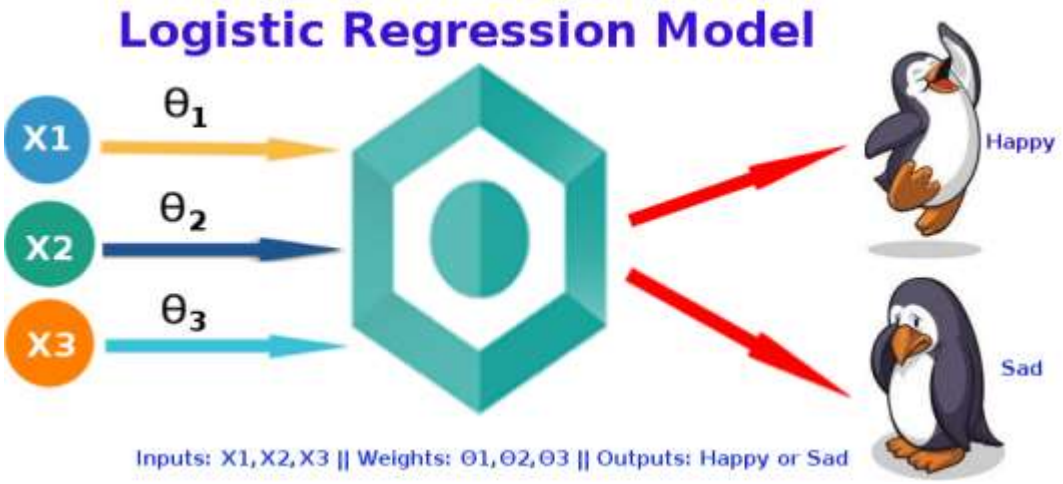
Additionally, the video dataset is also segmented into test and train data. The test_video set consists of 3 videos, while the train_video set encompasses 12 videos, incorporating various scenarios such as fire with smoke, only fire, only smoke, and no fire videos.

The process flow involves several steps:

1. Loading Dataset: The dataset is imported into the system for analysis and odel training.
2. Customizing Dataset: Relevant adjustments are made to the dataset to suit the requirements of the analysis and model training process.
3. Factor Identification: Factors or attributes most responsible for the spread of fire are identified through data analysis.

Following dataset preparation, the methodology employs various machine learning algorithms:

1) Logistic Regression: The algorithm is trained using the dataset, and metrics such as accuracy, precision, and recall are determined. The model's performance is evaluated by testing it with custom input.



2) SVM: Support Vector Machine model is fitted to the dataset, trained, and evaluated for accuracy, precision, and recall. Predictions are made using the test input data.

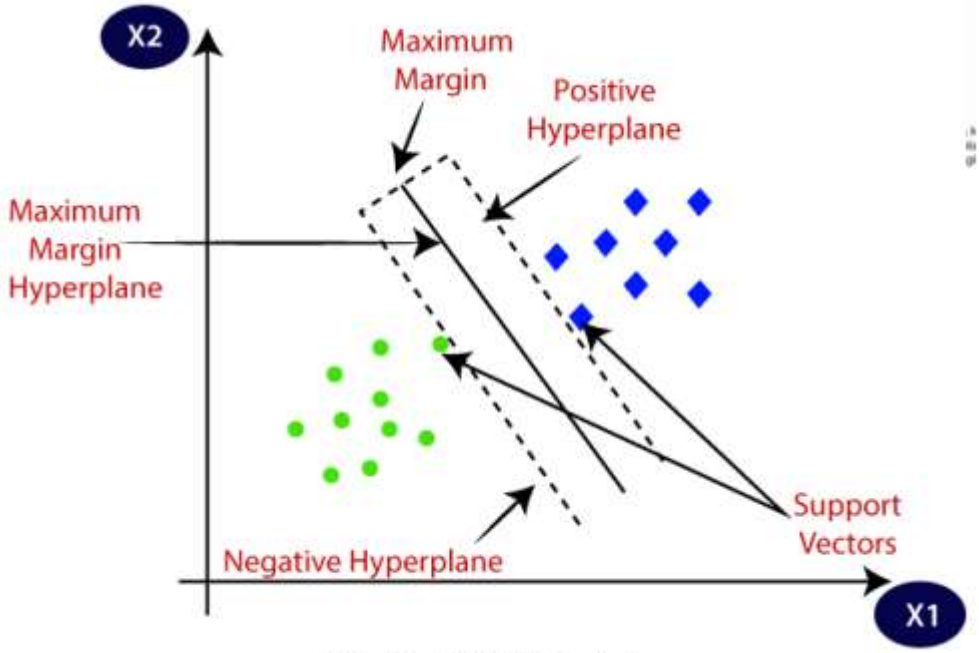
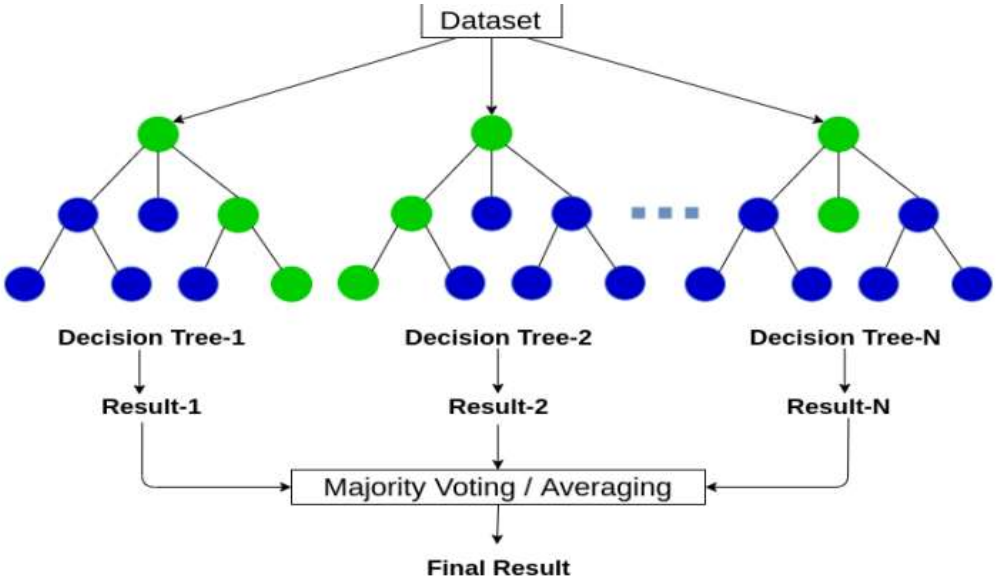


Fig 3: SVM Model

3) Random Forest: The Random Forest algorithm is applied to the dataset, trained, and predictions are made. Accuracy, precision, and recall metrics are calculated, and the model's performance is evaluated using custom input data.



Each algorithm's workflow is visually represented in corresponding figures, illustrating the steps involved in the model training and evaluation process.

Tools and Libraries:

- Python: Programming language for implementing the project.

- NumPy, Pandas: For data manipulation and preprocessing.

- Scikit-learn: For machine learning models and evaluation.

- Matplotlib, Seaborn: For data visualization.

Expected Outcome:

For this purpose, “Forest-fires.csv” dataset from UCI machine learning repository was taken and machine learning algorithms were applied to find accuracy of detection. The dataset “Forest-fires.csv” contains 517 instances and 13 attributes.

PYTHON CODE

```
# -- coding: utf-8 --
"""Forest_fire_prediction.ipynb

Automatically generated by Colab.

Original file is located at
https://colab.research.google.com/drive/1M9xmWo3Khy2nHf8ud4XE7AniRW8KCpgn

*[Fetching and preprocessing the data](https://)*
"""
```

1. Importing Python Libraries

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

from sklearn import metrics
from sklearn.metrics import classification_report,confusion_matrix

import warnings
warnings.filterwarnings(action="ignore")
```

2. Data Preprocessing

```
# %matplotlib inline
 pd.set_option("display.max_rows", 1000)
 pd.set_option("display.max_columns", 1000)


 fires = pd.read_csv("forestfires.csv")  #fetching the dataset


 #changing days into numeric quantity because machine learning model deals with numbers
 fires.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7), inplace=True)

 #changing month into numeric quantity
 fires.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12),
 inplace=True)

 #given area of land burnt, but we have to predict if there is fire or not so changing values of area to 0 and 1
 only
 #here 0 represet there is not fire and 1 represent fire, changing all values of area which are greater than 0 to 1
 fires['area'].values[fires['area'].values > 0] = 1

 #renaming the area attribute to output for clear understanding
 fires = fires.rename(columns={'area': 'output'})


 from sklearn.preprocessing import StandardScaler
 #standardization of data
 #removing the mean and scaling it to unit variance
 #score=(x-mean)/std
 scaler = StandardScaler()
```

```
#fitting forest fire dataset to scaler by removing the attribute output
scaler.fit(fires.drop('output',axis=1))

scaled_features = scaler.transform(fires.drop('output',axis=1))
df_feat = pd.DataFrame(scaled_features,columns=fires.columns[:-1])



from sklearn.model_selection import train_test_split
X = df_feat
y = fires['output']
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.35,random_state=200)
```

3. Applying Logistic Regression

```
#importing logistic regression
from sklearn.linear_model import LogisticRegression
logistic_model = LogisticRegression()
logistic_model.fit(X_train,y_train)

predictions = logistic_model.predict(X_test)

#finding precision,recall,accuracy
print("Precision:",metrics.precision_score(y_test, predictions))
print("Recall:",metrics.recall_score(y_test, predictions))
print("Accuracy:",metrics.accuracy_score(y_test, predictions))

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))



#prediction using logistic regression
class_label={1:'There is Fire',0:'There is no fire'}
x_new=[[1, 4, 9 ,1 ,91.5, 130.1, 807.1, 7.5, 21.3, 35, 2.2, 0]]


y_predict=logistic_model.predict(x_new)
print(class_label[y_predict[0]])
```

4. Applying SVM

```
# Support Vector Machine
from sklearn.svm import SVC

# fit a SVM model to the data

X = fires.drop('output', axis=1)
y = fires['output']

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3,random_state=101)

svc = SVC()
svc.fit(X_train, y_train)
# make predictions
prediction = svc.predict(X_test)
```

```
# summarize the fit of the model
print(metrics.classification_report(y_test, prediction))
print(metrics.confusion_matrix(y_test, prediction))

print("Accuracy:",metrics.accuracy_score(y_test, prediction))
print("Precision:",metrics.precision_score(y_test, prediction))
print("Recall:",metrics.recall_score(y_test, prediction))


#prediction using svm
classes={0:'safe',1:'On Fire'}
x_new=[[1, 4, 9 ,1 ,91.5, 130.1, 807.1, 7.5, 21.3, 35, 2.2, 0]]
y_predict=svc.predict(x_new)
print(classes[y_predict[0]])
```

5. **Applying Random Forest**

```
#import random forest
from sklearn.ensemble import RandomForestClassifier
X = fires.drop('output', axis=1)
y = fires['output']

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3,random_state=101)

# fit a Naive Bayes model to the data
random_forest = RandomForestClassifier()
random_forest.fit(X_train,y_train)
# print(random_forest)
# make predictions

predict = random_forest.predict(X_test)
# summarize the fit of the model
print(metrics.classification_report(y_test, predict))
print(metrics.confusion_matrix(y_test, predict))

print("Accuracy:",metrics.accuracy_score(y_test, predict))
print("Precision:",metrics.precision_score(y_test, predict))
print("Recall:",metrics.recall_score(y_test, predict))

#prediction using random forest
classes={0:'safe',1:'On Fire'}
x_new=[[1, 4, 9 ,1 ,91.5, 130.1, 807.1, 7.5, 21.3, 35, 2.2, 0]]
y_predict=random_forest.predict(x_new)
                    print(classes[y_predict[0]])
```

FUTURE OUTCOMES AND FINDINGS

We will be finding a method based on machine learning which will be

- Accurate in prediction
- Fault Tolerant
- Robust and then finding its space and time complexity and will try to optimise it.

CONCLUSION

Wireless sensor networks are helpful in detecting events. In the case of forest fire detection wireless network sensor nodes remove the difficulty faced in traditional methods like man standing on a tower and monitoring the environment. Now with the use of WSN we can put sensor nodes in each and every part of forest and mostly in the region where the risk is high. All the data collected by sensor nodes have to be aggregated to reach the result so it is done by using tree based and cluster-based methods.

The machine learning techniques add enhancement to the security of wireless sensor networks. With the use of machine learning techniques, the problem of faulty nodes is minimized. With the use of regression algorithm network lifetime is enhanced and with the use of decision tree algorithm network lifetime is enhanced as well as accuracy. SVM and neural network give better results.