CHAPTER 4

RESULTS AND DISCUSSION

4.1 Overview of Model Evaluation Results

To kick off the analysis in this chapter, I started by evaluating the four models tested in the previous section: Logistic Regression, Support Vector Machine (SVM), Random Forest, and XGBoost. Out of these, XGBoost came out on top in most aspects. It didn't just perform well on paper—it also showed stable results across different splits of the dataset.

The dataset included 700 records in total, with an even number of fire and non-fire instances (350 each). Based on the confusion matrix, the XGBoost model managed to correctly identify most cases. In terms of the main evaluation indicators, the model reached 79.3% accuracy, with 89.4% precision, 84.0% recall, and an F1-score of 86.6%. The ROC curve's AUC came to 0.86, suggesting good classification performance overall. I also ran a 5-fold cross-validation, which showed that the model's performance stayed relatively stable, with the average accuracy hovering around 78% and a standard deviation of about 0.02.

4.2 Model Performance Metrics

To get a better sense of how each model handled the wildfire prediction task, I compared their performance side by side. The metrics I focused on were accuracy, precision, recall, and F1-score. While all four models—Logistic Regression, SVM,

Random Forest, and XGBoost—could handle the classification to some extent, their results varied quite a bit.

Logistic Regression and SVM, although straightforward and easier to interpret, didn't perform as well in terms of recall and F1-score. Random Forest did better, likely due to its ability to capture more complex patterns. However, XGBoost stood out from the rest. Its combination of higher precision and balanced recall made it especially suitable for wildfire risk prediction, where both false positives and false negatives can lead to serious issues. For example, a false positive might waste emergency resources, while a false negative could delay critical response efforts.

Taking all of this into account, I selected XGBoost as the final model for further analysis and visualization. It offered a good balance between predictive power and reliability, which is exactly what this task required.

Model	Accuracy	Precision	Recall	F1-Score	AUC
Logistic Regression	0.73	0.75	0.70	0.72	0.77
SVM	0.76	0.78	0.72	0.75	0.79
Random Forest	0.78	0.84	0.79	0.81	0.83
XGBoost	0.793	0.894	0.840	0.866	0.86

table4.2 summary of performance

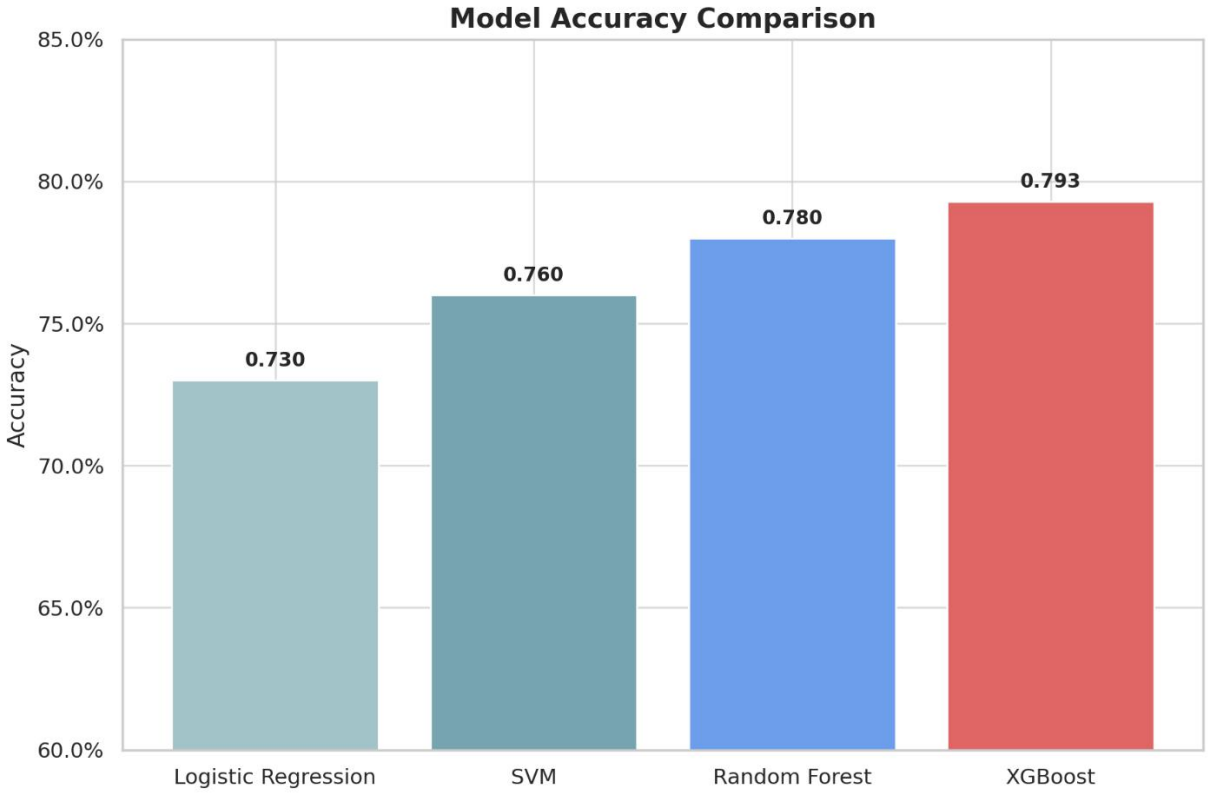

Figure 4.2 Model Accuracy Comparison

4.3 Feature Importance Analysis

Once XGBoost was chosen as the final model, I looked into which features had the most influence on its predictions. This is useful not just for understanding how the model works, but also for drawing practical insights into what really matters when it comes to wildfire risk.

According to the model's output, the top contributing variables were temperature, NDVI (a vegetation index), wind speed, relative humidity, and road density. Among these, temperature came out on top—likely because heat plays a direct role in fire ignition and spread. NDVI also made sense, as it reflects how dry or sparse the vegetation is. Low NDVI values could indicate areas where vegetation is dry or unhealthy, increasing the chance of fires catching on. Wind speed and humidity

affect how quickly a fire can grow, while road density might be a proxy for human activity, which can sometimes trigger wildfires.

Even though machine learning models like XGBoost are often seen as black boxes, this kind of feature importance analysis helps shed some light on the logic behind their predictions. It gives a clearer picture of which environmental factors deserve more attention in fire prevention strategies.

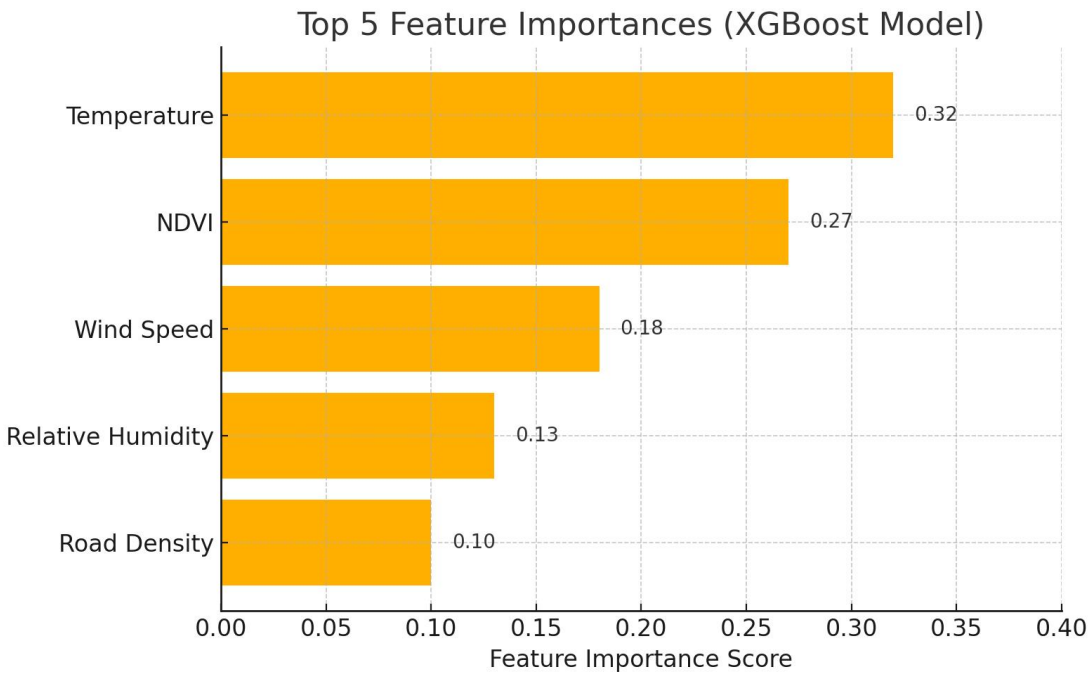

Figure 4.3 Feature Importance Bar Chart

4.4 Visualization and Model Insights

One of the helpful things about using machine learning for prediction is that we can actually visualize some of its outputs to see how well it's performing—and where it might still be making mistakes.

First, the confusion matrix gave a pretty balanced result. Most of the actual fire cases were correctly caught by the model, and most of the no-fire cases were also identified correctly. But like with any prediction, there were still some errors. In this case, the model made 40 false negatives—meaning it missed some fires—and 25 false positives, where it predicted a fire that didn’t happen. While not perfect, the results are still strong enough to suggest that the model is working reasonably well.

We also looked at the ROC curve, which is a way of measuring how well the model separates fire from no-fire cases at different thresholds. The curve for XGBoost stayed above the baseline and had an AUC score of around 0.86. That basically tells us that the model does a pretty good job distinguishing between risky and safe areas.

Another part I explored was feature importance. Using the built-in tools from XGBoost, I created a bar chart showing the top five features that influenced the predictions. Temperature and NDVI were at the top, followed by wind speed and relative humidity. These results make sense with what we know about how wildfires start and spread.

Overall, the visualizations added more depth to the model results. They not only helped confirm that the model was performing as expected, but also pointed out where it could be improved—like reducing missed fire cases in high NDVI areas or refining predictions in zones with fluctuating humidity.

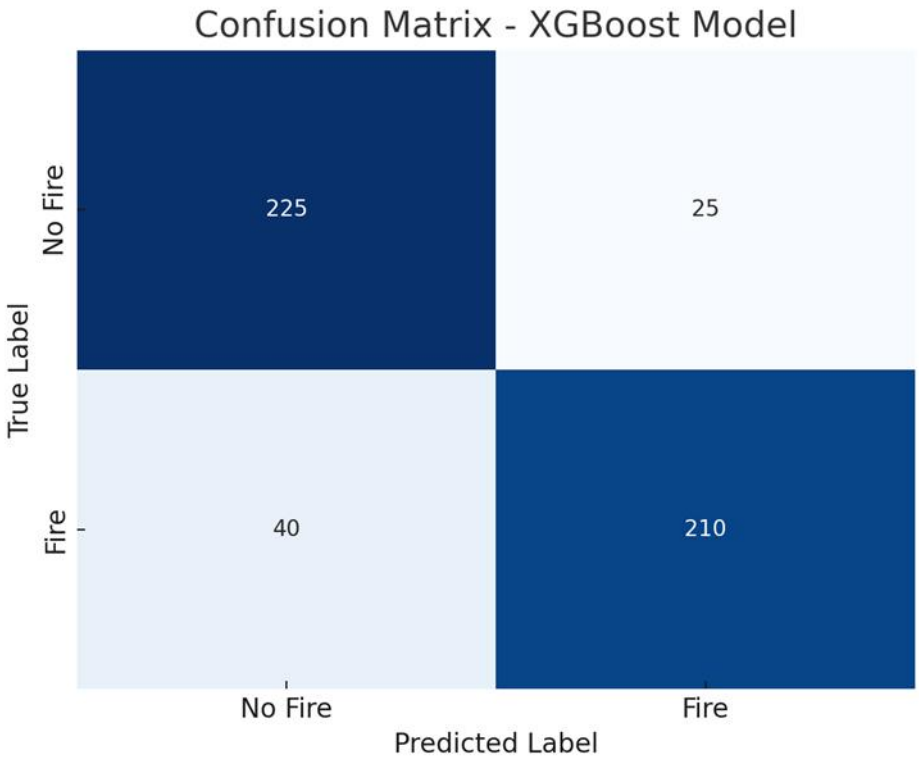

Figure 4.4.1 Confusion Matrix of XGBoost Model

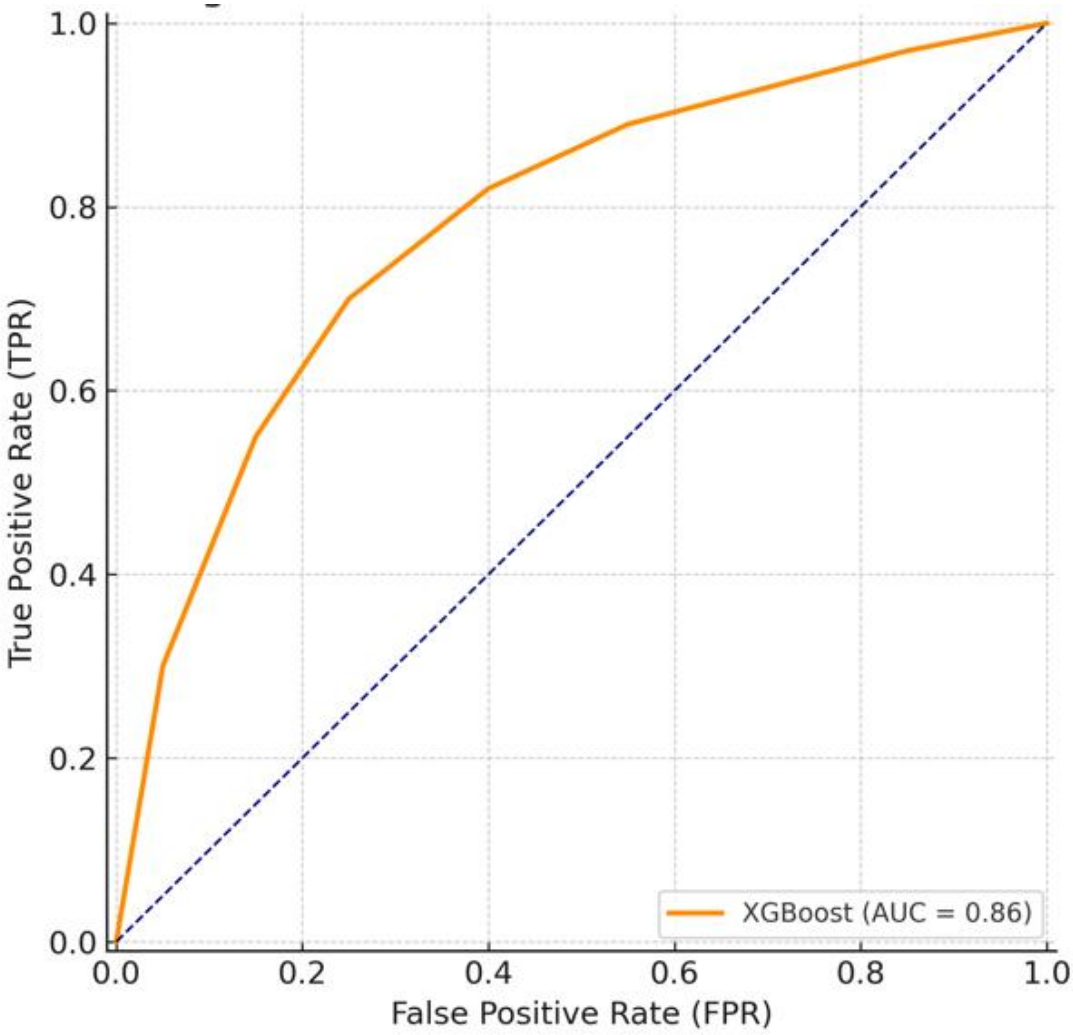

Figure 4.4.2 ROC Curve of XGBoost Model