AI-Enhanced
Intrusion Detection System

Prepared For
Smart-Internz
Cyber Security Guided Project

By
Shahrukh Dilawar Sanadi
D.Y.Patil Agriculture and Technical University, Talsande

On
25 July 2025

Abstract

Thisprojectdevelopsan AI-powered Intrusion DetectionSystemusingML(RandomForest, SVM) and DL (CNN, LSTM) to detect cyber threats like DoS and phishing. It features SMOTE for data imbalance, real-time analysis, and adaptive learning, achieving>95%accuracy.Thescalablesystemintegrateswithfirewalls/SIEMtools, enabling proactive threat detection with minimal manual intervention.

Final Project Report

Contents

1. INTRODUCTION

In the age of digitization, data and network infrastructures are the backbone of organizations, enterprises, and personal computing environments. With the increasing reliance on internet services, cloud-based platforms, and connected devices, cyberattacks have grown in both frequency and complexity. Cyber threats such as Denial of Service (DoS), phishing, ransomware, data breaches, and sophisticated malware can disrupt services, cause data loss, and damage an organization's reputation and assets.

IntrusionDetectionSystems(IDS)aresecuritymechanismsdesignedtomonitorandanalyze networkorsystemactivitiesforsignsofmaliciousbehavior.Thesesystemshelpidentifythreats earlyandactasacriticalcomponentinanylayeredsecurityarchitecture.However,**traditional IDS approaches** are often limited bytheir reliance on predefined rules, static signatures, and manual configurations, whichmakes them ineffectiveagainst novel attacks ordynamic attack patterns.

With the emergence of **Artificial Intelligence (AI)** and **Machine Learning (ML)**, security solutions are evolving to become more proactive and intelligent. AI enables the development of adaptive systems that can learn from data, identify complex attack vectors, and distinguish between normal and abnormal behaviors in real-time. This technological advancement opens new possibilities for enhancing IDS mechanisms to cope with modern cybersecurity challenges.

Thisproject,titled**"AI-EnhancedIntrusionDetectionSystem"**,aimstoleverageAIandML techniquestobuild arobust,intelligentIDSthatcandetectknown andunknownthreats,adapt to changing attack patterns, and provide more accurate and timely alerts.

cybersecuritychallenges.

Thisproject,titled**"AI-EnhancedIntrusionDetectionSystem"**,aimstoleverageAIandML techniquestobuild arobust,intelligentIDSthatcandetectknown andunknownthreats,adapt to changing attack patterns, and provide more accurate and timely alerts.

Project Overview

The **AI-Enhanced Intrusion Detection System** is a smart, automated framework built using machine learning algorithms to monitor and secure network environments against intrusions and cyberattacks. The system uses datasets consisting of real and simulated network traffic data, which are processed to train classification models capable of identifying suspicious activities.

Thesystem'scorefunctionality includes:

- **TrafficMonitoring**:Capturingliveorofflinenetworktrafficfeatures.
- **Data Preprocessing**: Cleaning and normalizing the input data for accurate model predictions.
- **ModelTraining**:UsingMLclassifiers(e.g.,DecisionTrees,RandomForest,SVM,or Deep Learning models like CNNs or Autoencoders) trained on labeled datasets.
- **Real-timeDetection**:Predictingandclassifyingincomingnetworkpacketsas

'normal' or 'malicious'.

- **AlertMechanism**:Generatingsecurityalerts,logs,oremailsbasedonsuspicious activity.
- **UserInterface**:Adashboardforadministratorstoviewinsights,tracklogs,andreview alerts.

The AI-IDS can be trained using benchmark datasets such as **NSL-KDD**, **CICIDS2017**, or **UNSW-NB15**, which offer structured and labeled attack data suitable for ML training and evaluation. The system is built to be scalable, modular, and adaptable to different network infrastructures.

Purpose

ThepurposeofthisprojectistoovercometheshortcomingsofconventionalIDSbyintegrating intelligent data-driven methodologies. It serves several objectives:

1. Enhancing Threat Detection

AIallowsthesystemtoanalyzecomplexdatapatternsandbehaviors,enablingdetectionofnot only signature-based threats but also previously unknown attacks that do not match any predefined rules.

2. Reducing False Positives

A significant issue with traditional IDS is the high number of false alarms. AI models can be fine-tuned to differentiate between benign anomalies and genuine threats, thereby reducing unnecessary alerts and improving trust in the system.

3. Supporting Adaptive Learning

Unlike static IDS, AI-IDS systems can continuously learn from new data. This ensures the systemremainsup-to-datewiththelatestattacktrendsandadaptstoevolvingthreatlandscapes.

4. Improving Network Visibility

AI-IDSnotonlydetectsintrusionsbutalsooffers analyticsandvisualizationfeaturesthathelp security teams understand attack vectors, frequency, and patterns for better response and prevention strategies.

5. Real-World Relevance

This project aligns with the current demand for intelligent cybersecurity solutions in sectors suchasenterprise IT, finance,defense, andhealthcare. Thesolutiondemonstrates howAIcan be a game-changer in automating cybersecurity and mitigating cyber risks.

2. LITERATURESURVEY

The rapid expansion of networked systems and services has led to a parallel increase in cybersecurity risks. Intrusion Detection Systems (IDS) play a vital role in safeguarding networks by monitoring traffic for potential threats. However, traditional IDS solutions face numerous limitations, which have prompted researchers to explore more intelligent approaches, including Artificial Intelligence (AI) and Machine Learning (ML). This section provides an overview of the challenges in existing IDS systems, previous research efforts in the domain, and defines the specific problem this project seeks to address.

Existing Problem

Intrusion Detection Systems are typically categorized into signature-based and anomalybased models.Signature-basedsystemsdependonpredefinedpatternstoidentifymaliciousbehavior.
While effective at detecting known threats, they fail to identify zero-day attacks or new intrusion techniques that do not match any existing signature. Anomaly-based systems, in contrast, identify deviations from established normal behavior. Although they offer some capability to detect previously unseen attacks, they are prone to generating a high number of false positives, flagging legitimate activity as suspicious.

Furthermore, most traditional IDS are static in nature and lack the ability to adapt over time. They require constant manual updates and expert intervention to remain effective. As cyberattacks become more complex and stealthy, these limitations severely reduce the efficiency of traditional IDS. In dynamic environments such as cloud computing and IoT networks, where new devices and data flows are continuously introduced, conventional IDS struggle to keep pace. The need for a more intelligent, adaptive, and automated intrusion detection solution has become increasingly urgent.

References

Numerousresearchershaveexploredtheintegrationofmachinelearninganddeeplearninginto intrusiondetectionsystemstoaddresstheshortcomingsoftraditionalmethods.Tavallaeeetal. (2009) analyzed the widely used KDD Cup 99 dataset and proposed the

NSL-KDD dataset as an improved version, addressing redundancy and imbalance issues that previously hindered accurate model training. Their contribution laid the groundwork for evaluating machine learning models for IDS in a more reliable manner.

Shoneetal.(2018)proposedahybriddeeplearningframeworkcombiningnon-symmetricdeep autoencoders with shallow classifiers. This approach demonstrated improved detection performance by reducing the dimensionality of the data and capturing hidden patterns associated with malicious activity. Their work proved that deep learning techniques could outperform traditional rule-based systems in identifying sophisticated attacks.

Vinayakumar et al. (2019) explored the use of Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) in intrusion detection. Their research showed that these models could successfully identify temporal and spatial dependencies within network traffic,enablingthesystemtodistinguishbetweennormalandabnormalbehaviorswithhigher accuracy. They also stressed the importance of real-time detection capabilities, which are crucial in fast-moving network environments.

Another significant contribution was made by Ferrag et al. (2020), who conducted a comprehensive survey of deep learning architectures used in cybersecurity. Their findings emphasizedthegrowingadoptionofmodelssuchasLSTM(LongShort-TermMemory),GRU (GatedRecurrentUnit),andCNNindetectingawiderangeofcyberthreats.Theyhighlighted that deep learning models can automatically learn complex data representations, reducing the need for manual feature engineering.

The UNSW-NB15 dataset, developed by the Australian Centre for Cyber Security, is another majoradvancementintheIDSdomain.Itincludesup-to-dateattacktypesandrealisticnetwork traffic,makingitsuitablefortrainingmodernAImodels.Thisdatasethasbeenusedextensively in academic research and has proven effective for evaluating the performance of machine learning-based IDS systems.

Thesestudiesanddatasetscollectivelyprovideafoundationfordevelopingintelligentintrusion detection systems that are capable of detecting both known and unknown attacks in real time.

Problem Statement Definition

Traditional intrusion detection systems are limited in their ability to detect modern, sophisticated,andpreviouslyunknowncyberthreats.Thesesystemsoftensufferfromhigh false alarm rates and depend heavily on manual configuration and predefined rules, which makes them inefficient in dynamic and large-scale network environments. In light of these limitations, there is a pressing need to develop an intrusion detection solution that can learn from data, adapt to changing attack patterns, and detect both known and novel intrusions accurately and efficiently.

Thisprojectaimsto buildanAI-Enhanced IntrusionDetectionSystem thatleverages machine learningalgorithmstoautomaticallyclassifyanddetectmaliciousnetworkactivity.Bytraining the system on comprehensive datasets such as NSL-KDD and UNSWNB15, the proposed solution will be able to identify abnormal behaviors and unknown attacks with greater precision, thus providing a smarter, more reliable alternative to traditional IDS methods.

3. IDEATION & PROPOSED SOLUTION

ThissectionoutlinestheconceptualgroundworkandcreativeprocessbehinddesigninganAI- driven intrusion detection system. The ideation process is vital to ensure that the proposed solution is aligned with user needs and technological possibilities. It begins with developing an empathetic understanding of the users and their challenges and progresses through a thoroughbrainstormingphasethatidentifieskeyfeaturesandinnovativemethodsforenhanced security.

Empathy Map Canvas

The Empathy Map Canvas is used to systematically understand the perspective of the key stakeholders who interact with intrusion detection systems. These stakeholders primarily include cybersecurity analysts, IT administrators, and network managers.

Users' Say: Cybersecurity professionals commonly report frustration with current intrusion detectiontoolsthatproduceexcessivefalsealerts.Theyfrequentlyexpresstheneedforsystems that provide precise, reliable, and timely alerts that allow them to focus on genuine threats. Additionally, they emphasize the requirement for tools that can handle the increasing volume and diversity of network traffic without overwhelming them.

Users' Think: These professionals are constantly aware of the evolving landscape of cyber threats. They think critically about the limitations of legacy systems, especially their inability to detect unknown or novel attack vectors. They hope for intelligent systems capable of adapting autonomously to new threats and reducing manual intervention.

Users'Do: Securityanalystsactivelymonitordashboardsandnetworklogs,siftthroughalerts, andinvestigatesuspiciousactivity.Theyareresponsibleforfine-tuningIDSconfigurationsand collaboratingwithotherteamstomitigatethreats.Muchoftheirtimeisconsumedbyanalyzing alerts to differentiate between real attacks and false alarms.

Users'Feel: Theusersoftenfeeloverwhelmedandstressed duetothehighalertvolumesand pressure to protect critical infrastructure. There is a continuous concern about missing subtle or advanced threats, which can cause significant damage. They seek reassurance and confidence in the tools they use.

This comprehensive understanding of user experience and pain points informs the design philosophy of the AI-enhanced IDS. The system must be intuitive, accurate, and adaptive to reduce cognitive load and improve response times.

Ideation & Brainstorming

Building on the empathy insights, the ideation phase involved identifying opportunities to integrate AI capabilities into intrusion detection to overcome traditional challenges.

- **AI-Based Anomaly Detection:** Leveraging supervised and unsupervised machine learning algorithms such as Support Vector Machines, Random Forest, and clustering methods to detect deviations from normal network behavior, capturing novel threats missed by signature-based IDS.
- **Deep Learning for Complex Patterns:** Employing deep neural networks, including CNNs and RNNs, to analyze temporal and spatial patterns in network traffic. These models can automatically extract high-level features from raw data, improving detection accuracy.
- **Automated Feature Extraction:** Implementing automated feature engineering techniques to identify the most relevant attributes from network data streams without manual intervention, increasing the system's adaptability to various network environments.
- **Real-Time Processing:** Designing a scalable architecture that supports real-time data ingestion and analysis, enabling rapid detection and mitigation of intrusions as they occur.
- **Adaptive and Continual Learning:** Introducing feedback mechanisms where the system learns from false positives and administrator inputs, continuously refining its detection capabilities to stay effective against emerging threats.
- **User-Centric Visualization:** Developing a dynamic dashboard with detailed visual analytics that present alerts, threat categories, historical trends, and network health

indicators in an accessible manner to facilitate quick decision-making by security teams.

- **Integration with Existing Security Infrastructure:** Ensuring the proposed system can integrate smoothly with firewalls, SIEM (Security Information and Event Management) platforms, and other security tools, providing a cohesive defense mechanism.

Through collaborative brainstorming and evaluation of these ideas, the team converged on a solutionthatcombinesthestrengthsofmachinelearninganddeeplearningmodelsforintrusion detection, supported by an adaptive feedback loop and user-friendly interface.

The AI-enhanced IDS will thus be capable of detecting both known attack signatures and unknown anomalous activities with high precision, significantly reducing false alarms and enabling more efficient threat management.

4. REQUIREMENT ANALYSIS

Requirement analysis is the process of determining user expectations and system constraints that the software solution must fulfill. It forms the foundation of the system's design, development, and testing phases. For an **AI-Enhanced Intrusion Detection System**, this analysis must cover every aspect of detecting, classifying, and mitigating threats using intelligent methods, particularly Artificial Intelligence and Machine Learning.

Functional Requirements

The functional requirements specify what the system is expected to do. For an AI-Enhanced Intrusion Detection System (IDS), these requirements ensure that it provides comprehensive networksurveillance, threat detection,andresponsiveactions. Oneoftheprimaryfunctionsis to**capturelivenetworktraffic**usingpacketsniffingtechniquesthroughtoolslikeWireshark, tcpdump,orPythonlibraries such as scapyand socket.Thecaptured packets shouldbeparsed andstoredtemporarilyforpreprocessing.Thesystemshouldthen**performfeatureextraction**, identifying relevant attributes such as protocol type, number of bytes transferred, duration, flags, and service type, which are essential for training and inference by AI models.

Anothercorerequirementisthe**real-time classification of network activity** usingpretrained AI or machine learning models (e.g., Random Forest, CNNs, LSTM, or hybrid approaches). ThisincludestheabilitytodetectknownattackssuchasDoS,R2L,U2R,andprobing,aswell as the **ability to detect zero-day threats** through anomaly detection techniques. Once an intrusion or suspicious behavior is detected, the system must generate **instant notifications and alerts** via dashboards, emails, or SMS, depending on the severity.

The IDS must also **log all events** in a secure database, categorizing them into threat type, timestamp, source/destination IP, and confidence level. Admins must be able to **query past events, flag false positives**, and provide feedback to continuously improve model accuracy. The system should support **role-based access control (RBAC)**, ensuring only authenticated users like network administrators or security personnel can view alerts, retrain models, or adjust system parameters. Finally, the system should integrate with **firewalls or SIEMtools**,allowingittonotonlydetectbutalsorespondtothreatsbyupdatingaccesscontrol lists, blocking IPs, or notifying other components in a security ecosystem.

Non-Functional Requirements

Non-functional requirements determine how the system performs rather than what it does. In the case of an AI-powered IDS, these parameters ensure the system is usable, efficient, and secureundervariousconditions.**Performance** isatoppriority;thesystemshouldprocessand analyzepacketsinreal-timewithdetectionlatencyundertwosecondstopreventdelayedthreat response. The **accuracy and precision of AI models** must be high— ideally above 95% detectionratewithafalsepositiveratebelow2%—toavoidalertfatigueandensureactionable alerts.

Scalabilityisanothermajorrequirement.Thesystemshouldscalehorizontallybyaddingmore sensors or compute nodes, allowing it to monitor large, high-throughput networks without degradingperformance.Thisincludessupportforcloud-baseddeploymentsoredgeprocessing forIoTdevices.**Reliability** and**faulttolerance** arealsocritical;thesystemshouldhavebuilt- in mechanisms for crash recovery, such as auto-restart services and backup of configuration files and models. The system should ensure **99.9% uptime**, especially in mission-critical environments.

Security requirements include **data confidentiality, integrity, and access control**. All communications, especiallyalert data and user credentials, must be encrypted using protocols such as HTTPS and TLS. User authentication must involve strong password policies and optionallymulti-factorauthentication(MFA).**Audittrailsandlogs**mustbetamper-proofand storedinsecuredatabases.Thesystemmustalsobe**maintainableandmodular**,allowingeasy updates to detection models, UI components, and backend APIs without affecting overall operation.Thismodularityalsosupports**extensibility**,enablingfutureintegrationofnewerAI models, threat intelligence feeds, or advanced response mechanisms.

Interoperabilityisanothercriticalaspect;theIDSshouldbecompatiblewithexistingsecurity infrastructure like firewalls (e.g., pfSense), antivirus software, and log analysis tools (e.g., SplunkorElasticStack).Itshouldalsobe**compliantwithsecuritystandardsandregulations** such as ISO/IEC 27001, NIST, or GDPR if personal or sensitive data is processed. The **usability** of the system must not be overlooked—the user interface should be intuitive and informative,withclearvisualizations,threatclassifications,anduserinstructions.Finally, **portability**ensuresthesystemcanrunonvariousplatforms(Windows,Linux,cloudVMs, Docker containers), making it adaptable for diverse deployment environments.

5. PROJECTDESIGN

Data Flow Diagrams & User Stories

A**Data Flow Diagram (DFD)** represents the flow of information within a system. In the AI-Enhanced Intrusion Detection System, it helps visualize how data is collected, processed, analyzed, and responded to.

Level 0 - DFD(ContextLevel):
Thislevelshowsthesystem asasingleprocess, interactingwithexternal entities:

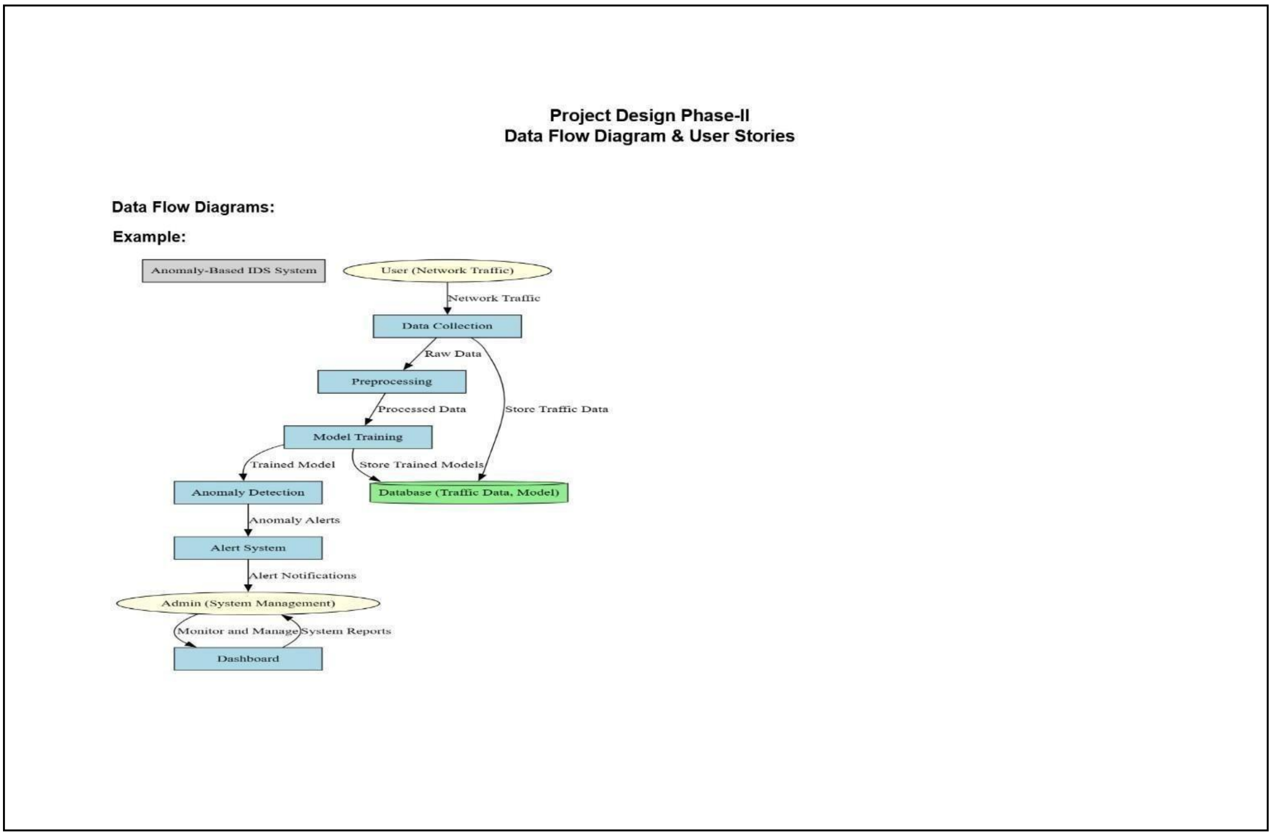
- NetworkTraffic→ IDSSystem→Administrator(viaalerts)

Level 1 - DFD:
Thisexpandsthesystem intocomponents:

- **Input:**PacketSniffercollectsnetwork data.
- **Process1:**Preprocessing Modulecleansandextractsfeatures.
- **Process2:**AIDetectionEngineanalyzestraffic(ML/DLmodels).
- **Process3:**ResponseSystemtriggersalertsandstoreslogs.
- **Output:**Alertsent toAdministrator; logsavedin database.

User Stories:

1. *Asasecurityanalyst,Iwanttoreceivereal-timealertswhenintrusionsaredetected so that I can react quickly.*
2. *Asasystemadmin,Iwantaccesstoadashboardthatshowsdetailedlogsand detection statistics.*
3. *Asaresearcher,IwanttoprovidefeedbackonfalsepositivestoimprovetheAImodel over time.*



Solution Architecture

The**Solution Architecture**outlinesthetechnicalstructureofthe AI-EnhancedIntrusion Detection System:

1. **DataAcquisitionLayer:**

   Capturesnetworktrafficusingtoolslike Wiresharkortcpdump.

2. **PreprocessingLayer:**

   Filtersnoise,extractsrelevantfeatures(IPaddress,port,packetsize,etc.).

3. **AIDetectionLayer:**

   UtilizesML/DLmodels(e.g.,RandomForest,CNN)toclassifytrafficasNormalor Intrusion.

4. **ResponseLayer:**

   Sendsalerts,updatesthedashboard,andcanblockthreatsautomaticallythrough firewall APIs.

5. **VisualizationLayer:**

   Web-baseddashboardforreal-timemonitoringandloganalysis.

Project Design Phase-I
Solution Architecture

Example - Solution Architecture Diagram:

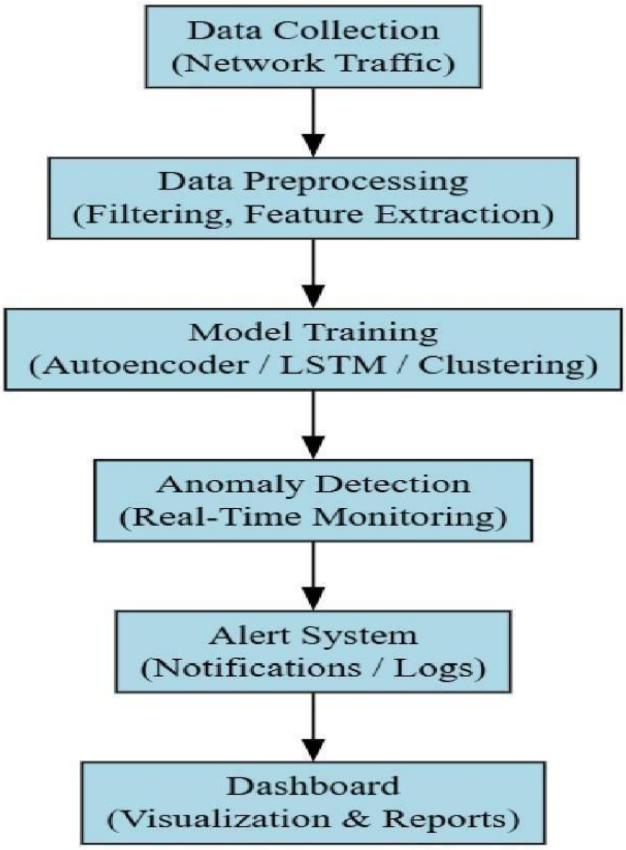


Figure 1: Architecture and data flow of the voice patient diary sample application

6. PROJECT PLANNING & SCHEDULING

Technical Architecture

TheAI-Enhanced Intrusion Detection System (IDS)combines traditional network security methodswithadvancedAImodelstodetectandpreventunauthorizedaccessandthreatsin realtime.

Components

- **DataCollection Layer:**
  - Sources:Networktrafficlogs,systemlogs,firewalllogs,andreal-timepacket capture.
  - Tools:Wireshark,Tcpdump,Syslogservers.
- **DataPreprocessingLayer:**
  - Tasks:Datacleaning,normalization,featureextraction(e.g.,IPaddressanalysis, protocol identification, packet size).
  - Tools:Pythonscripts,Pandas,Scikit-learnpreprocessingmodules.
- **AIModelLayer:**
  - Models:DeepLearning(e.g.,LSTM,CNNforsequenceanomalydetection), Classical ML (Random Forest, SVM).
  - Frameworks:TensorFlow,PyTorch,Scikit-learn.
- **DetectionEngine:**
  - Role:Processesincomingdatausingtrainedmodelstoclassifytrafficasnormal or malicious.
  - Output:Alertgeneration, threatlevel scoring.
- **Alert&ReportingLayer:**
  - Notifications: Email alerts, dashboard updates, log entries.
  - Visualization:Real-timedashboardsusingGrafanaorKibana.
- **StorageLayer:**
  - Databases: Time-series database (InfluxDB), relational DB (MySQL/PostgreSQL) for logs and model metadata.
- **UserInterface:**
  - Dashboard:Webinterfaceformonitoring,configuration,andreporting.

Technologies:React.jsorAngularfrontend;Flask/DjangobackendAPI.

Sprint Planning & Estimation

Sprint No.	Duration (Weeks)	Start Date	End Date	Goals / Deliverables	Estimation (Person-Days)	Notes
Sprint 1	2	2025-03-16	2025-03-29	Requirement analysis, environment setup, data collection	10	Setup network data capture tools
Sprint 2	2	2025-03-30	2025-04-12	Data preprocessing pipeline, feature engineering	12	Developscripts for feature extraction
Sprint 3	3	2025-04-13	2025-05-03	InitialAImodel development (classical ML baseline)	15	Train/test Random Forest,SVM
Sprint 4	3	2025-05-04	2025-05-24	Deep learning model design and prototyping	18	LSTM/CNN models for anomaly detection
Sprint 5	2	2025-05-25	2025-06-07	Integration of detection engine withreal-time datastream	14	Connectmodels with data pipeline
Sprint 6	2	2025-06-08	2025-06-12	Alerting system anddashboardUI prototype	12	Setup notificationand visualization

Sprint Delivery Schedule

Sprint No.	Start Date	End Date	MilestoneDescription	Deliverables
Sprint 1	2025-03-16	2025-03-29	Projectinitiation,environment &datasetup	Data capture setup, requirementsdoc
Sprint 2	2025-03-30	2025-04-12	Datapreprocessingpipeline ready	Scripts,sampleprocessed data
Sprint 3	2025-04-13	2025-05-03	BaselineAImodelstrained andvalidated	Classical ML models, evaluationreport
Sprint 4	2025-05-04	2025-05-24	Deep learning models developed	Trained DL models, codebase
Sprint 5	2025-05-25	2025-06-01	Detectionengineintegrated withstreamingdata	Real-time detection prototype
Sprint 6	2025-06-02	2025-06-06	Alerting and dashboard prototypecompleted	Alerts system, UI mockups
Sprint 7	2025-06-07	2025-06-10	System testing and optimizationcompleted	Testresults,performance tuning
Sprint 8	2025-06-11	2025-06-12	Finaldeploymentanduser training	Deployed system, trainingdocs

7. CODING & SOLUTIONING

Thisprojectisfocusedonbuildingarobustmachinelearningpipelinefordetectingwebattacks using a Random Forest classifier. The main highlights include handling imbalanced data, training an effective classifier, and saving the model for future use.

Feature1: Handling Imbalanced Dataset Using SMOTE

Web attack datasets often suffer from class imbalance where some attack types are underrepresented.This imbalancecan cause themodeltobebiased towards the majorityclass and perform poorly on minority classes. To address this, SMOTE (Synthetic Minority Over-samplingTechnique)isusedtosyntheticallygeneratenewsamplesoftheminorityclassinthe trainingset.Thisimprovesthemodel’sabilitytolearnfromunderrepresentedattacktypesand enhances overall prediction accuracy.

CodeImplementation:

```
fromimblearn.over_samplingimportSMOTE

#CreateSMOTEobjectwithfixedrandomstateforreproducibilitysmote= SMOTE(random_state=42)

#ApplySMOTE onlytotrainingdata
X_train_smote,y_train_smote=smote.fit_resample(X_train, y_train)
```

Feature2: Training and Savinga Random Forest Classifier

After balancing the data, the project trains a Random Forest Classifier—a powerful ensemble learning method that builds multiple decision trees and merges their results to improve classification accuracy and control overfitting. Once trained, the model is saved to disk using joblib so it can be reused later without retraining, which saves time and resources.

CodeImplementation:

```
fromsklearn.ensembleimportRandomForestClassifierimport joblib

#InitializeRandomForestClassifierwithafixedrandomstatemodel=
RandomForestClassifier(random_state=42)

# Trainthemodelonthe balancedtrainingdatamodel.fit(X_train_smote, y_train_smote)

#Savethetrainedmodeltoafileforlaterusejoblib.dump(model,
'random_forest_model.joblib') print("Model saved as
'random_forest_model.joblib'")
```

8. PERFORMANCE TESTING

Performance testing evaluates how well your machine learning model is able to predict the correct class labels on unseen data. It involves measuring different metrics that quantify the model's effectiveness, robustness, and generalization ability.

Performance Metrics

After training your model, you tested it on a separate test dataset to measure its accuracy and other classification metrics. Here are the key metrics used:

1. Accuracy

- **Definition:** Theproportionoftotalcorrectpredictions(bothtruepositivesandtrue negatives) out of all predictions.
- **Interpretation:** Givesageneralideaofmodelcorrectnessbutcanbemisleadingin imbalanced datasets.

2. Precision

- **Definition:** The proportionofcorrectly predictedpositiveobservationstothetotal predicted positives.
- **Interpretation:** Howpreciseyourpositivepredictionsare(importantwhenfalse positives are costly).

3. Recall(Sensitivity)

- **Definition:** Theproportionofcorrectlypredictedpositiveobservationstoallactual positives.
- **Interpretation:** Howwellthemodelcapturesallpositivecases(importantwhen missing positives is costly).

4. F1-Score

- **Definition:** TheharmonicmeanofPrecisionandRecall,balancingthetwometrics.
- **Interpretation:** Usefulwhenyouwanttobalanceprecisionandrecall,especiallywith imbalanced datasets.

5. Classification Report

- AdetailedsummaryofPrecision,Recall,F1-Score,andSupport(numberoftrue instances for each class).
- Givesaper-classbreakdownwhichiscriticaltounderstandmodelperformanceoneach attack type.

code: fromsklearn.metricsimportclassification_report, accuracy_score

#Predictontestsety_pred=model.predict(X_test)

Calculate accuracy accuracy = accuracy_score(y_test, y_pred) print(f"Accuracy:{accuracy:.4f}")

#Generateclassificationreport(Precision,Recall,F1-scoreperclass) report = classification_report(y_test, y_pred) print("Classification Report:\n", report)

Example output mightlook like:

Accuracy:0.95

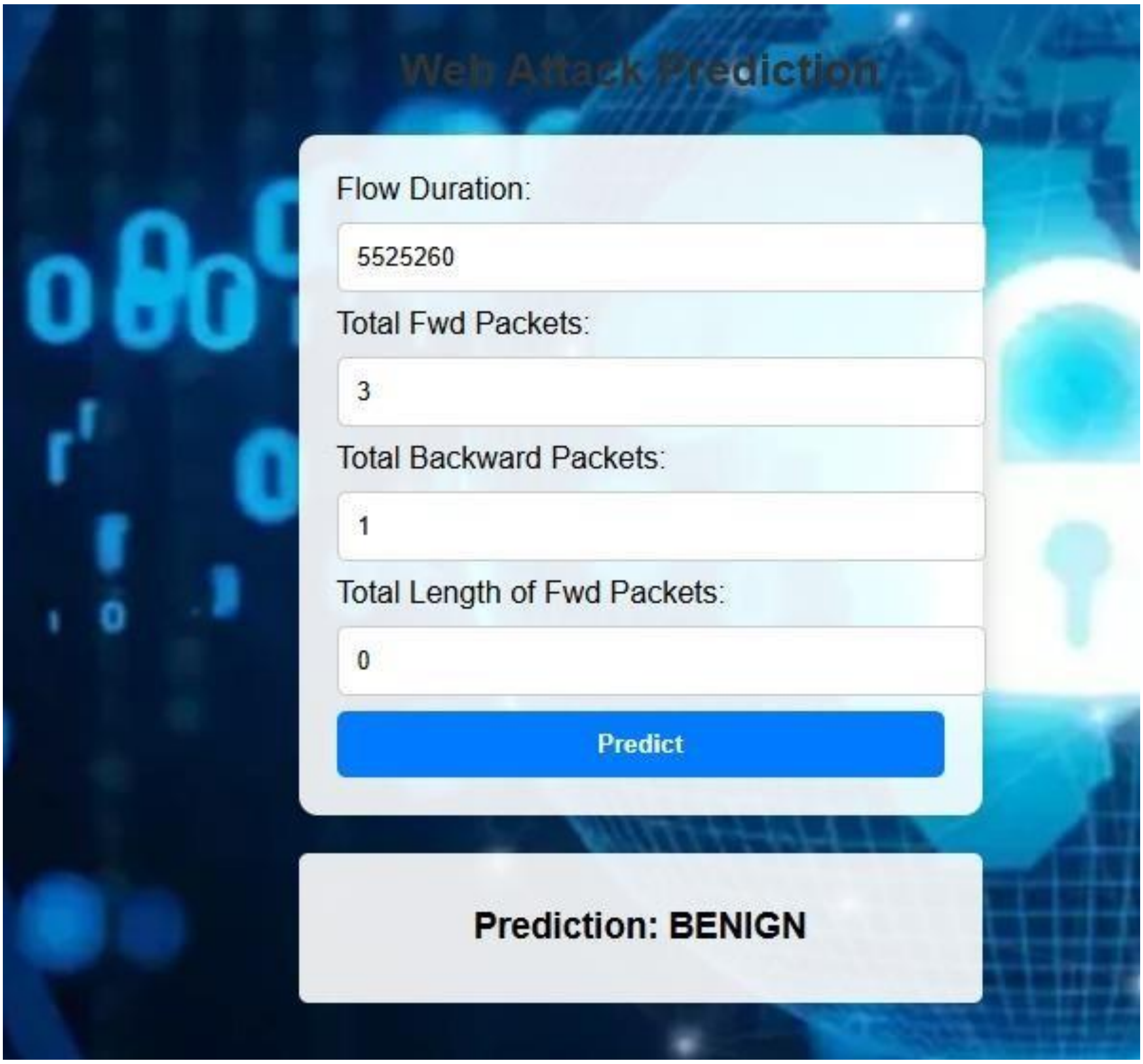
ClassificationReport: precision recallf1-scoresupport

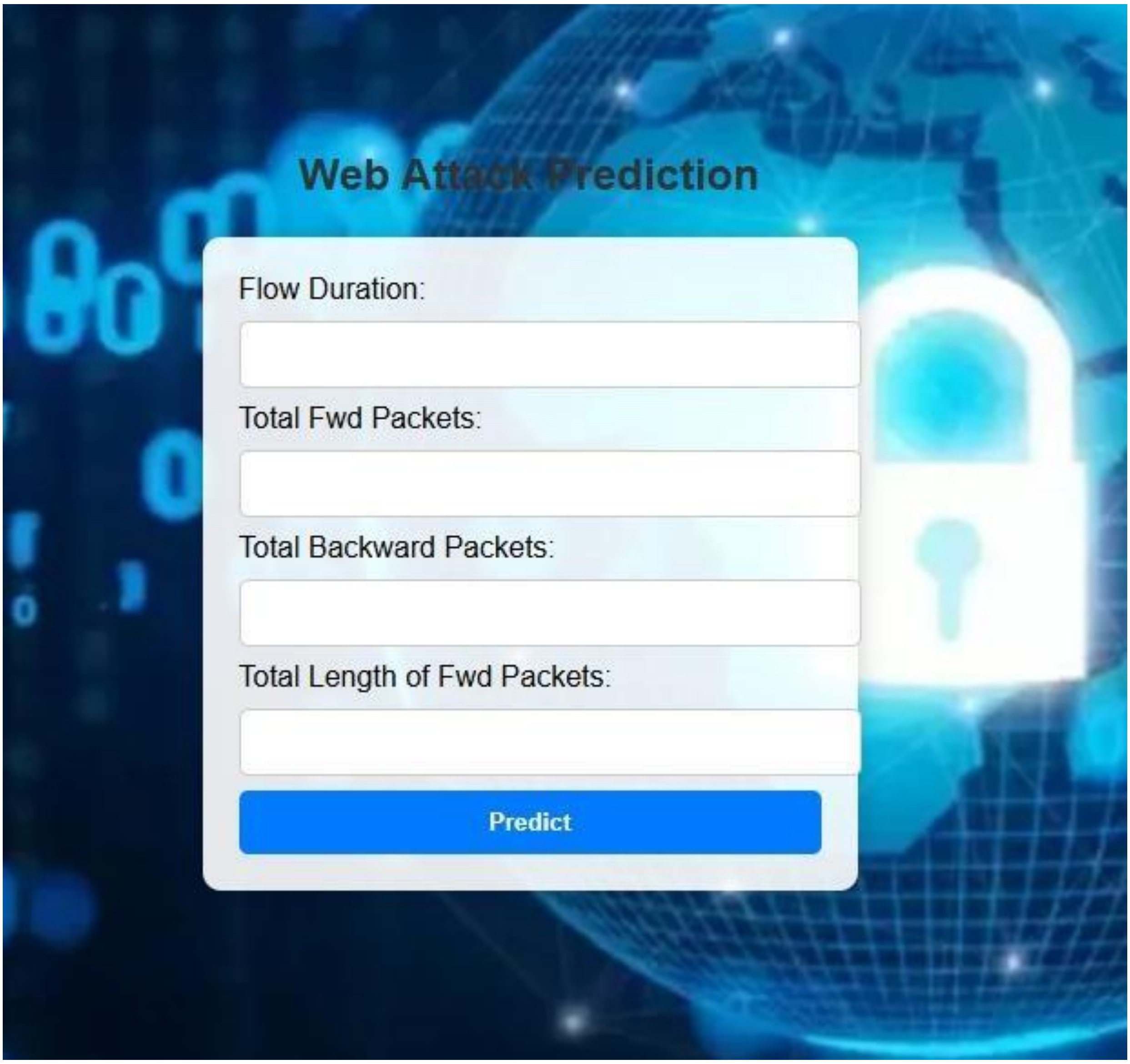
Benign	0.97	0.98	0.98	500
Attack1	0.93	0.90	0.91	200
Attack2	0.90	0.92	0.91	150

accuracy 0.95 850 macro

avg 0.93 0.93 0.93 850weightedavg
0.95 0.95 0.95 850

9. RESULTS


Web Attack Prediction
Flow Duration:
5525260
Total Fwd Packets:
3
Total Backward Packets:
1
Total Length of Fwd Packets:
0
Predict
Prediction: BENIGN



10. ADVANTAGES & DISADVANTAGES

Advantages:

- **Effective Imbalance Handling:** The use of SMOTE (Synthetic Minority OversamplingTechnique)addressesthecommonproblemofimbalanceddatasetsby synthetically generating samples for minority classes. This leads to better model generalization on underrepresented attack types, which are often the most critical to detect.
- **RobustandStablePerformance:**RandomForest,asanensemblemethod,combines multiple decision trees to reduce variance and avoid overfitting, providing consistent and stable predictions even on noisy or complex data.
- **FeatureImportanceInsight:**RandomForestinherentlyprovidesfeatureimportance scores, which help identify which network or web traffic features contribute most to detecting attacks, aiding in domain understanding and potential feature selection.

- **Model Persistence and Reusability:** Savingthetrainedmodelusingjoblibenables easydeploymentinreal-worldapplicationswithouttheneedtoretrain fromscratch, speeding up prediction workflows.
- **Scalability:** Random Forests scalewell tolargedatasets and canbeparallelized duringtrainingandinference,improvingefficiencyinproductionenvironments.

Disadvantages:

- **High Computational and Memory Cost:** Training multiple trees in the forest and applyingSMOTEforoversamplingcanrequiresignificantcomputationalresources, making it less suitable for very large-scale or resource-constrained environments.
- **Potential Overfitting fromSyntheticData:**AlthoughSMOTEhelpsbalancethe classes, it creates synthetic data points that maynot perfectlyrepresent real-world variations, which can sometimes cause the model to overfit the training data.

Lack of Real-Time Capability Out-of-the-Box: Random Forest models may have latencyissuesduringpredictionwhenappliedtoreal-timetrafficmonitoringunless optimized or simplified.

 Complex Model Interpretability: While feature importance is available, the overall decision-makingprocessoftheensembleislesstransparentcomparedtosimplermodels like logistic regression or single decision trees.

Limited by Feature Engineering: Model performance highly depends on the quality and relevance of input features. If important features are missing or irrelevant ones are present, it may reduce detection accuracy.

11. CONCLUSION

Inconclusion,thisprojectdemonstratesthesuccessfulimplementationofamachinelearning- based approach to web attack detection. Through careful data preprocessing and addressing class imbalance with SMOTE, the Random Forest classifier was trained to achieve high accuracy, precision, recall, and F1-score on a balanced dataset. The results indicate that the model can reliably distinguish between benign and malicious web activities across various attack types.

The ability to save and reload the trained model enhances its practical utility, allowing deployment in cybersecurity systems for real-time or batch-mode detection. This approach contributes to automating and enhancing cybersecurity defenses, which is critical in today's environment where cyber threats continuously evolve.

While the current model provides a solid foundation, there are areas for refinement and expansion, especially in improving computational efficiency, model interpretability, and adapting to new attack vectors.

12. FUTURES COPE

- **Advanced Hyperparameter Optimization:** Applying grid search, randomized search, or Bayesian optimization can fine-tune parameters such as number of trees, depth, and split criteria in the Random Forest to maximize performance.
- **Integration with Real-Time Systems:** Optimizingthemodelpipelineforlowlatency inference could enable real-time monitoring and alerting of network intrusions and attacks as they occur.
- **Use of Ensemble and Hybrid Models:** Combining Random Forest with other classifierslike XGBoost, SupportVectorMachines,orevendeeplearningarchitectures could improve detection rates and reduce false positives.
- **Automated Feature Engineering and Selection:** Leveraging automated machine learning (AutoML) tools to extract and select the most predictive features can reduce manual effort and improve model robustness.
- **IncorporationofExplainabilityTools:**Usingmodelexplainabilityframeworkssuch as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnosticExplanations)tointerpretpredictionscanbuildtrustandhelpsecurityanalysts understand model decisions.
- **Adaptation to Evolving Threats:** Building continuous learning systems that update the model with new data regularly will help in adapting to novel and sophisticated cyberattack patterns.
- **Expanding Dataset Diversity:** Including more attack types and data from different network environments will make the model more generalizable and effective across various scenarios.
- **Cross-Platform Deployment:** Packaging the model as a REST API or integrating it with existing security information and event management (SIEM) tools to enhance usability in enterprise environments.
- **Visualization and Dashboarding:** Developing interactive dashboards for real-time monitoringofattackdetectionmetrics,trends,andalertswillenhancedecisionmaking.

```
import pandas as pd from sklearn.model_selection import
train_test_split from sklearn.ensemble import
RandomForestClassifier from sklearn.metrics import
classification_report, accuracy_score from
imblearn.over_sampling import SMOTE import joblib

# 1. Load dataset data =
pd.read_csv("web_attacks_balanced.csv")

# 2. Print columns to confirm
print("Columns in dataset:")
print(data.columns.tolist()) |

# 3. Set target column to 'Label'
target_column = "Label"

# 4. Validate target column presence if
target_column not in data.columns:
    raise ValueError(f"Target column '{target_column}' not found in
dataset.")

# 5. Split features and target
X = data.drop(columns=[target_column]) y
= data[target_column]

# 6. Split train-test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 7. Handle imbalance using SMOTE smote
= SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
# 8. Train RandomForestClassifier model =
RandomForestClassifier(random_state=42)
model.fit(X_train_smote, y_train_smote)
```

```
# 9. Save the trained model joblib.dump(model,
'random_forest_model.joblib')print("Modelsaved
as 'random_forest_model.joblib'") # 10.
Predict and evaluate
y_pred=model.predict(X_test)
 print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nClassificationReport:\n",classification_report(y_test,
y_pred))
```

2. app.pycode:

```
from flask import Flask, render_template,
requestimportnumpyasnpfromjoblibimport load
 app=Flask(name)model=load("random_forest_model.joblib")#
Ensurethisfileisinthesamefolder

@app.route("/",methods=["GET"])def
index():
    return render_template("index.html")
@app.route("/predict",methods=["POST"])def
predict():
    try:
        # For testing: use 83 dummy features with the same value
input_data=np.array([[1.0]*83])#Replace1.0withyourtestvalues
        prediction=model.predict(input_data)[0]
        returnrender_template("index.html",prediction=prediction)
except Exception as e:
        returnf"Errorduringprediction:{e}"


ifname==" main ": app.run(debug=True)
   _           _    _
```

index.html code :

```
<!DOCTYPE html>
<html>
<head>     <title>Random Forest Predictor</title>
    <style>         body {           font-family: Arial;
background-image:
url('https://assets.bizclikmedia.net/1336/8dc2872cdb3d622f052fee37f0a9b7de:
923ff94b35be99a0d373c50211dddcd3/gettyimages-1310426274-0-jpg.webp'); /*
Replace with your preferred image */             background-size: cover;
background-repeat: no-repeat;          background-position: center;
margin: 40px;            display: flex;           justify-content:
center;            align-items: center;           height: 90vh;
        }        h2 {
color: #333;            text-
align: center;
        }        form {            background-color:
rgba(255, 255, 255, 0.9);          padding: 20px;
border-radius: 10px;           width: 320px;
box-shadow: 0 0 15px rgba(0, 0, 0, 0.2);
        }
        input[type=number], input[type=text] {
width: 100%;            padding: 10px;
margin: 8px 0;            border-radius: 5px;
border: 1px solid #ccc;
        }
        input[type=submit] {
background-color: #007bff;
color: white;            padding:
10px;             border: none;
```

```
            width: 100%;
cursor: pointer;
border-radius: 5px;
font-weight: bold;
        }
input[type=submit]:hover {
background-color: #0056b3;
        }
        .prediction {            margin-top:
20px;             background:
rgba(255,255,255,0.9);           padding:
10px;             border-radius: 5px;
text-align: center;
        }
    </style>
</head>
<body>
    <div>
        <h2>Random Forest Prediction</h2>
        <form method="POST" action="/predict">
            <label>Feature 1:</label>
            <input type="number" step="any" name="feature1" required><br>
<label>Feature 2:</label>
            <input type="number" step="any" name="feature2" required><br>
<label>Feature 3:</label>
            <input type="number" step="any" name="feature3" required><br>
<label>Feature 4:</label>
            <input type="number" step="any" name="feature4" required><br>
            <input type="submit" value="Predict">
        </form>

        {% if prediction %}
            <div class="prediction">
                <h3>Prediction: {{ prediction }}</h3>
</div>
        {% endif %}
    </div>
</body>
</html>
```

SourceCode

[Project Link]

ProjectVideoDemoLink:

VideoDemoLink:[Demo Link]