

# **INTERNSHIP REPORT**

## **ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**DLithe Consultancy Services Pvt. Ltd.**

## **Internship Report**

**Intern Name: Tejas M Naik**

**Reg. no: NNM23CS513**

**Period: June 04 - Aug 04**

**Job Assignment: AI-ML Internship project**

**Organization: DLithe Consultancy Services Pvt. Ltd.**

**Supervisor's Name: Ms. Archana, Ms. Sushma**

**Observations:**

**Submitted to**

**Signature of Training Supervisor**

**Signature of Co-ordinator**

**Date:**

**Date:**

**Letter of Transmittal**

To,

Program Co-ordinator  
DLithe Consultancy services  
Bengaluru

Dear Sir/madam,

We are writing to submit our report on the **Artificial Intelligence and Machine Learning Internship** that we recently completed. The training program was an invaluable learning experience, and we are grateful for the opportunity to participate.

The training program covered various aspects of **Artificial Intelligence and Machine Learning**, including foundational concepts, algorithms, programming languages such as Python, and real-world applications. We gained a comprehensive understanding of the role of **Artificial Intelligence and Machine Learning** in modern technology and industry, and also gained hands-on experience with popular AI/ML tools, frameworks, and platforms. The training highlighted the potential of AI and ML to revolutionize various domains, including healthcare, finance, marketing, and automation.

This report includes a detailed overview of the training program, outlining the topics covered, learning objectives, and the outcomes achieved.

We believe that the knowledge and skills we acquired during this internship will be valuable to our organization. The ability to work with AI/ML technologies will be increasingly important for innovation and growth.

We hope this report provides useful insights into the benefits of on-the-job training and the transformative potential of **Artificial Intelligence and Machine Learning**.

Sincerely,

Name: Tejas M Naik,

Reg. no: NNM23CS513

## Table of Contents

<b>1. Introduction</b>	<b>5</b>
<b>2. Project Overview</b>	<b>6</b>
<b>3. Problem Statement</b>	<b>7</b>
<b>4. Appendix</b>	<b>8</b>
○ GitHub	
○ Mind Map	
○ Architecture Diagram	
○ Flow Chart	
○ Use Case Diagram	
<b>5. Project / Use Case Implementation</b>	<b>12</b>
<b>6. Tools and Technologies Used</b>	<b>13</b>
<b>7. Reference Images</b>	<b>15</b>
<b>8. Training Experience</b>	<b>18</b>
<b>9. Observations</b>	<b>19</b>
<b>10. Key Learnings</b>	<b>21</b>
<b>11. AI/ML Applications in the Real World</b>	<b>22</b>
<b>12. Conclusion</b>	<b>23</b>

## Introduction

Artificial Intelligence and Machine Learning represent the cutting edge of modern technology, enabling computers to learn complex patterns and make intelligent decisions without explicit programming. In the financial sector, these technologies are revolutionizing fraud detection, risk assessment, and automated decision-making processes.

### What is Credit Card Fraud Detection?

Credit Card Fraud Detection is the process of identifying suspicious or fraudulent transactions in real-time using machine learning algorithms. It involves analyzing transaction patterns, user behavior, and risk indicators to automatically classify transactions as legitimate or fraudulent.

The Building Blocks:

- *Supervised Learning*: Training models on labeled fraud/legitimate transaction datasets
- *Classification Algorithms*: Random Forest and Decision Tree for pattern recognition and prediction
- *Data Preprocessing*: Feature scaling, encoding, and handling class imbalance using SMOTE
- *Model Evaluation*: Using precision, recall, F1-scores, and confusion matrices to measure performance

### What Can You Build?

- Real-time fraud detection systems for banks and payment processors
- Risk assessment platforms for e-commerce transactions
- Automated transaction monitoring with confidence scoring
- Interactive web applications for fraud analysis and investigation

## Overview

The AI/ML Internship Training program focused on developing a comprehensive **Credit Card Fraud Detection System** using advanced machine learning techniques. The program was designed to provide hands-on experience with data preprocessing, model training, performance evaluation, and web application deployment using Flask framework.

The training consisted of theoretical foundations and extensive practical implementation sessions. We learned fundamental ML algorithms, data preprocessing pipelines, class imbalance handling techniques, model evaluation methodologies, and web-based deployment strategies. The practical sessions involved building a complete end-to-end fraud detection system from raw data processing to production-ready web application.

### Key Technical Components:

- *Data Pipeline:* Comprehensive preprocessing using ColumnTransformer with StandardScaler and OneHotEncoder
- *Class Imbalance Handling:* SMOTE (Synthetic Minority Oversampling Technique) implementation
- *Model Training:* Random Forest and Decision Tree classifiers with hyperparameter optimization
- *Web Interface:* Flask-based application with interactive UI for real-time fraud prediction
- *Performance Visualization:* Automated confusion matrix generation and model comparison dashboard

## Problem Statement

**Objective:** Develop an intelligent credit card fraud detection system that can automatically classify financial transactions as fraudulent or legitimate with high accuracy, precision, and recall while providing real-time predictions through a user-friendly web interface.

**Business Challenge:** Credit card fraud causes billions of dollars in losses annually worldwide. Traditional rule-based detection systems are insufficient against sophisticated, evolving fraud patterns. The challenge is compounded by:

- Severe class imbalance (fraud cases typically <1% of total transactions)
- Need for real-time processing capabilities
- Requirement for high precision to minimize false positives
- Need for interpretable results for manual investigation

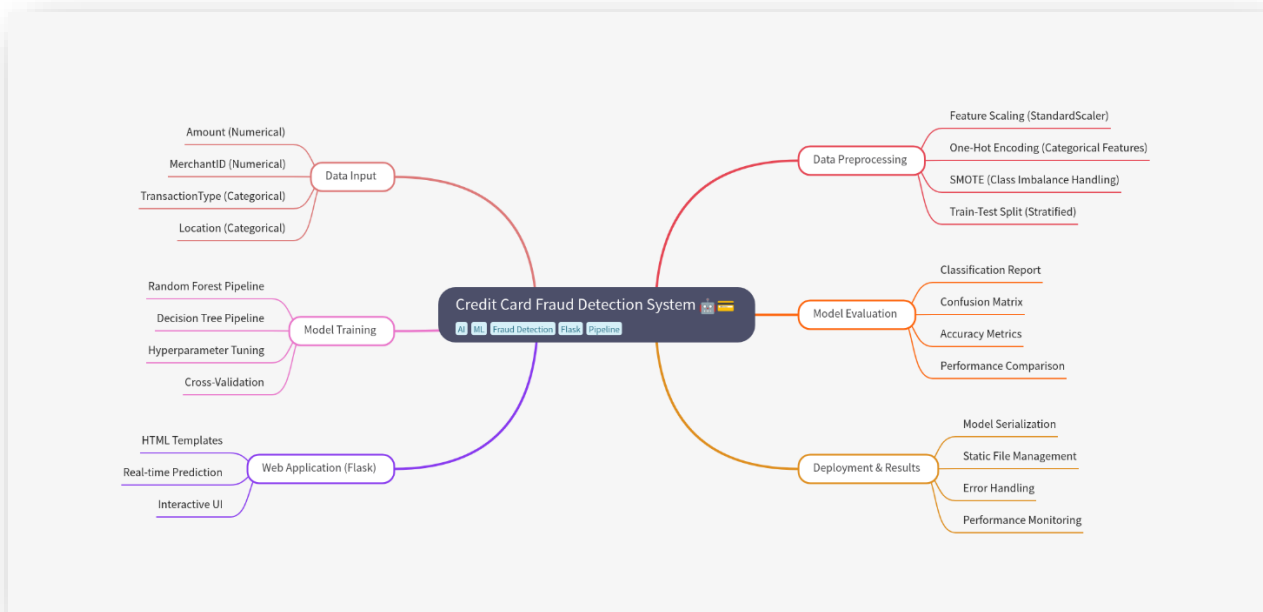
### Technical Requirements:

- Process transactions in real-time with sub-second response times
- Achieve high precision (>95%) to minimize false fraud alerts
- Maintain high recall (>90%) to catch actual fraud cases
- Handle class imbalance effectively using advanced sampling techniques
- Provide confidence scores and interpretable predictions
- Compare multiple ML algorithms for optimal performance selection
- Deploy through intuitive web interface for non-technical users

## Appendix

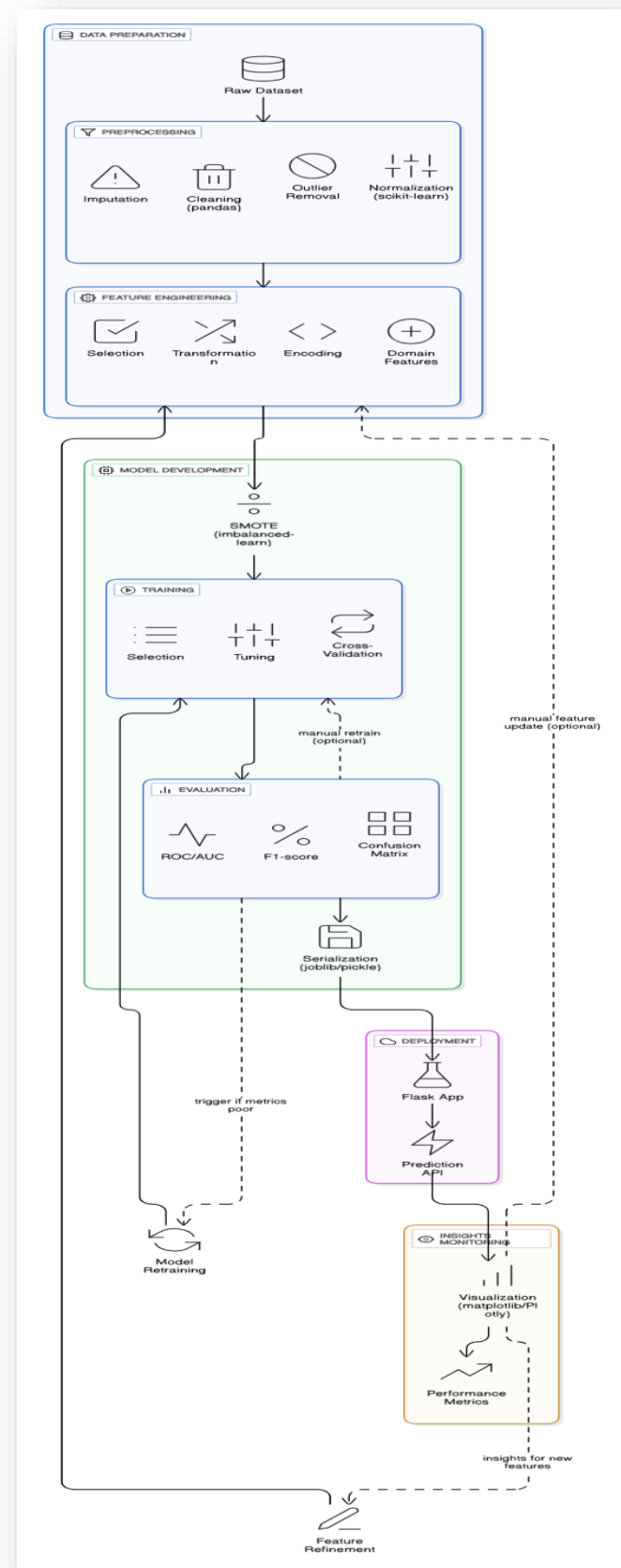
GitHub: <https://github.com/naiktejas/Credit-Card-Fraud-Detection.git>

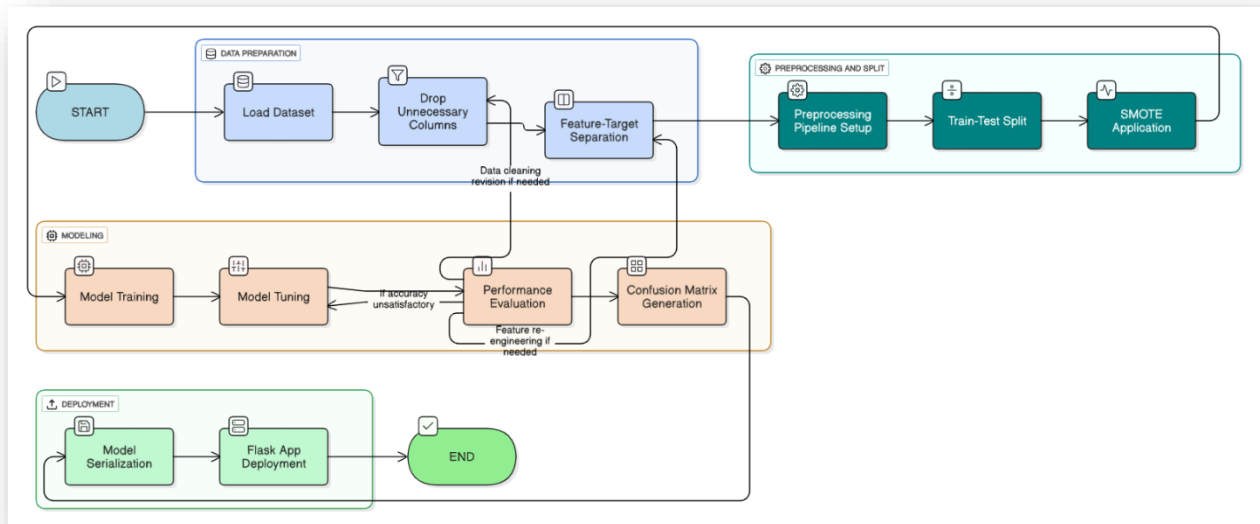
Mind Map:



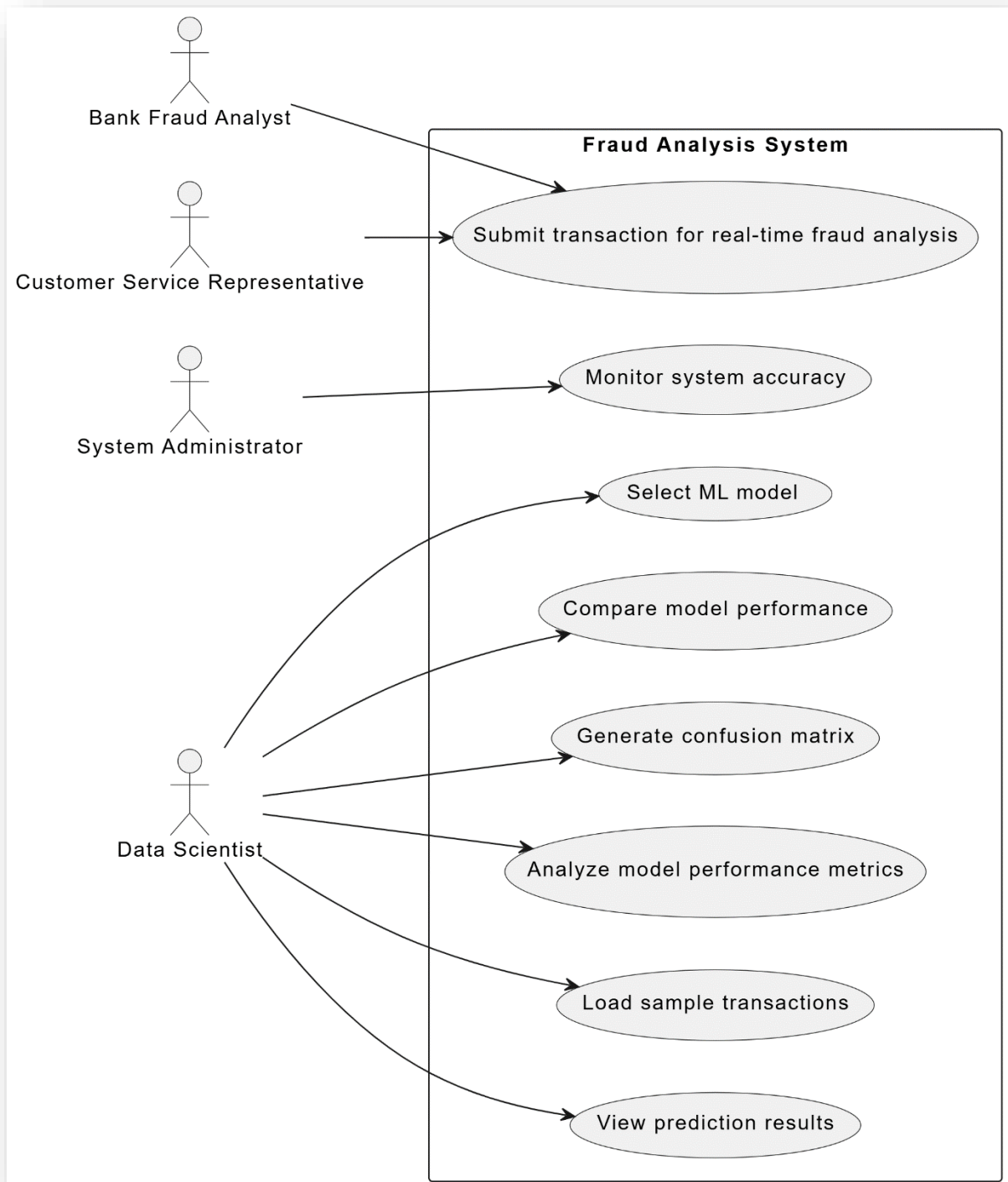


### Architecture Diagram:



**Flow Chart:**

Use case Diagram:



## Project / Use Case implementation

### Use Case: Real-time Credit Card Fraud Detection Web Application

**Target Audience:** This system caters to financial institutions, payment processors, fraud analysts, and customer service representatives who need to quickly assess transaction legitimacy.

**Goal:**

- *Accurate Detection:* Effectively identify fraudulent transactions using state-of-the-art ML algorithms
- *User-Friendly Interface:* Provide intuitive web interface for non-technical users
- *Real-time Processing:* Enable immediate fraud assessment for time-critical decisions
- *Comprehensive Analysis:* Offer detailed performance metrics and model comparison capabilities

**Key Features Implemented:**

**Interactive Transaction Analysis:**

- Real-time fraud prediction with confidence scoring
- Support for both Random Forest and Decision Tree models
- Dynamic form validation and error handling
- Sample data loading for quick testing (fraud/legitimate examples)

**Advanced ML Pipeline:**

- Automated data preprocessing with ColumnTransformer
- SMOTE implementation for handling class imbalance
- Stratified train-test splitting for reliable evaluation
- Cross-model performance comparison

**Performance Monitoring:**

- Real-time accuracy, precision, recall, and F1-score calculation
- Automated confusion matrix generation and visualization
- Model performance comparison dashboard
- Static file management for confusion matrix images

**Technical Implementation Details:**

- *Backend:* Flask web framework with robust error handling
- *ML Pipeline:* scikit-learn with imblearn for SMOTE integration
- *Data Processing:* pandas for data manipulation and preprocessing
- *Visualization:* matplotlib for confusion matrix generation
- *Model Persistence:* joblib for efficient model serialization
- *Frontend:* Responsive HTML with modern CSS styling and interactive elements

## Tools and Technologies Used

During the project, we utilized a comprehensive technology stack including:

### Machine Learning & Data Science:

**scikit-learn:** Core machine learning library for algorithm implementation

- RandomForestClassifier for ensemble learning
- DecisionTreeClassifier for interpretable predictions
- StandardScaler for numerical feature normalization
- OneHotEncoder for categorical feature encoding
- ColumnTransformer for integrated preprocessing pipeline
- train\_test\_split for data partitioning
- classification\_report, accuracy\_score, confusion\_matrix for evaluation

**imblearn:** Specialized library for handling imbalanced datasets

- SMOTE (Synthetic Minority Oversampling Technique) for class balancing
- Pipeline integration for seamless preprocessing

**pandas:** Data manipulation and analysis

- DataFrame operations for data processing
- CSV file reading and data exploration
- Feature engineering and data cleaning

**NumPy:** Numerical computing foundation for efficient array operations

### Web Development Framework:

**Flask:** Lightweight Python web framework

- Route handling for different endpoints
- Template rendering with Jinja2
- Form data processing and validation
- Static file serving for images and CSS
- Error handling and debugging capabilities

### Data Visualization:

**Matplotlib:** Plotting library for confusion matrix visualization

- ConfusionMatrixDisplay for professional matrix plots
- Figure saving for web integration
- Customizable plot styling and formatting

### Development Environment:

**Visual Studio Code:** Primary IDE for development with Python extensions

**Python 3.x:** Core programming language

**Git:** Version control for project management

### **Model Persistence & Deployment:**

**joblib:** Efficient model serialization and loading

- Model persistence for production deployment
- Test data saving for consistent evaluation
- Feature list preservation for preprocessing

**HTML/CSS:** Frontend user interface development

- Responsive design with modern styling
- Interactive form elements and user feedback
- Professional dark theme with gold accents

## Reference Images

```

29 | ('num', StandardScaler(), numerical_features),
30 | ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
31 | ])
32 |
33 | # Pipelines for both models
34 | pipelines = {
35 |     'rf': ImbPipeline([
36 |         ('preprocessor', preprocessor),
37 |         ('smote', SMOTE(random_state=42)),
38 |         ('classifier', RandomForestClassifier(random_state=42))
39 |     ]),
40 |     'dt': ImbPipeline([
41 |         ('preprocessor', preprocessor),
42 |         ('smote', SMOTE(random_state=42)),
43 |         ('classifier', DecisionTreeClassifier(random_state=42))
44 |     ])
45 | }
46 |
47 | # Split data
48 | X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2, random_state=42)
49 |
50 | # Ensure static folder exists for saving plots
51 | os.makedirs("static", exist_ok=True)
52 |
53 | # Train and evaluate models
54 | for name, pipeline in pipelines.items():
55 |     print(f"\n🔄 Training {name.upper()}...")
56 |     pipeline.fit(X_train, y_train)
57 |     preds = pipeline.predict(X_test)
58 |
59 |     # Print classification report
60 |     print(f"\n📊 Classification Report for {name.upper()}:\n")
61 |     print(classification_report(y_test, preds))
62 |
63 |     # Print accuracy
64 |     accuracy = accuracy_score(y_test, preds)
65 |     print(f"✅ Accuracy for {name.upper()}: {accuracy:.4f}")

```


Sample of training the model

```

6 | app = Flask(__name__)
7 |
8 | # Load models and test data
9 | rf_model = joblib.load("rf_model.pkl")
10 | dt_model = joblib.load("dt_model.pkl")
11 | features = ['Amount', 'MerchantID', 'TransactionType', 'Location']
12 | X_test, y_test = joblib.load("test_data.pkl")
13 |
14 | # Available options
15 | transaction_types = ['purchase', 'refund']
16 | locations = list(X_test['Location'].unique())
17 |
18 | @app.route("/")
19 | def home():
20 |     return render_template("index.html",
21 |         features=features,
22 |         transaction_types=transaction_types,
23 |         locations=locations,
24 |         preset={})
25 |
26 | @app.route("/sample/<label>")
27 | def sample(label):
28 |     label_int = 1 if label == "fraud" else 0
29 |     sample_row = X_test[y_test == label_int].iloc[0]
30 |     sample_dict = sample_row.to_dict()
31 |
32 |     return render_template("index.html",
33 |         features=features,
34 |         transaction_types=transaction_types,
35 |         locations=locations,
36 |         preset=sample_dict)
37 |
38 | @app.route("/predict", methods=["POST"])
39 | def predict():
40 |     try:
41 |         model_choice = request.form["model"]
42 |         model = rf_model if model_choice == "RandomForest" else dt_model

```

Sample of flask app

 **Credit Card Fraud Detection**

● Load Legit Sample

● Load Fraud Sample

Select Model:

Random Forest

Amount:

2801.06

Merchant ID:

918


Transaction Type:


purchase


Location:

New York


Predict

Result:  LEGITIMATE

Prediction:  LEGITIMATE  
Confidence: 99.00%

 **Model Evaluation Metrics**

- **Accuracy:** 99.84%
- **Precision (Fraud):** 0.9967
- **Recall (Fraud):** 1.0000
- **F1-Score (Fraud):** 0.9984

 **Confusion Matrix**

RF Confusion Matrix

Legit	19735	65
Fraud	0	19800
	Legit	Fraud


Predicted label

017500

Output of a legit sample

16



 **Credit Card Fraud Detection**

● Load Legit Sample

● Load Fraud Sample

**Select Model:**

Random Forest

**Amount:**

2996.57

**Merchant ID:**

659

**Transaction Type:**


purchase

**Location:**


Houston

Predict

Prediction: ⚠️ **FRAUDULENT**  
Confidence: 100.00%

 **Model Evaluation Metrics**

- **Accuracy:** 99.84%
- **Precision (Fraud):** 0.9967
- **Recall (Fraud):** 1.0000
- **F1-Score (Fraud):** 0.9984

 **Confusion Matrix**

RF Confusion Matrix

Legit	19735	65
Fraud	0	19800
	Legit	Fraud

Predicted label

0 2500 5000 7500 10000 12500 15000 17500

Output of a fraudulent sample

## Training Experience

**Hands-on Learning:** Our training program provided extensive hands-on experience with the complete machine learning pipeline, from data preprocessing to model deployment. We worked with real-world fraud detection scenarios, developing practical skills in feature engineering, class imbalance handling, model evaluation, and web application development using Flask framework.

**Mentorship:** We were fortunate to have mentors Ms. Archana and Ms. Sushma, experienced AI/ML professionals who provided invaluable guidance on algorithm selection, hyperparameter tuning, evaluation metrics interpretation, and deployment best practices. Their expertise was crucial in understanding advanced concepts like SMOTE, ensemble methods, and production-ready ML systems.

**Collaboration:** Working with cross-functional teams enhanced our communication skills essential for ML projects. We learned to explain complex technical concepts to non-technical stakeholders, collaborate on model improvement strategies, and integrate feedback from domain experts in fraud detection.

**Exposure to Industry Trends:** Through workshops, seminars, and hands-on projects, we stayed current with latest developments in fraud detection, ensemble learning methods, class imbalance techniques, and ML deployment practices. This broader perspective prepared us for real-world challenges in financial AI applications.

**Use of Industry-standard Tools and Technologies:** During training, we gained experience with production-grade tools including scikit-learn, Flask, imblearn, pandas, and modern development practices. This allowed us to develop practical skills directly applicable to industry requirements.

**Iterative Process:** We learned that developing effective fraud detection systems requires extensive experimentation, hyperparameter tuning, and continuous model evaluation. The iterative nature of ML development became clear through hands-on implementation and performance optimization.

## Observations

During our comprehensive training on AI/ML Fraud Detection, we observed several critical aspects:

**Importance of Data Quality and Preprocessing:** The success of fraud detection models heavily depends on comprehensive data preprocessing and feature engineering. Our implementation using ColumnTransformer with StandardScaler and OneHotEncoder demonstrated the critical impact of proper data preparation on model performance.

**Class Imbalance Challenges:** Fraud detection suffers from severe class imbalance, with legitimate transactions vastly outnumbering fraudulent ones. Our implementation of SMOTE (Synthetic Minority Oversampling Technique) proved essential for achieving balanced model performance across both classes.

**Algorithm Selection and Performance Trade-offs:** Through direct comparison of Random Forest and Decision Tree algorithms, we observed distinct performance characteristics:

- *Random Forest:* Higher accuracy and robustness due to ensemble approach, better handling of overfitting
- *Decision Tree:* More interpretable results crucial for regulatory compliance and fraud investigation

**Feature Engineering Impact:** Transaction amount, merchant patterns, location data, and transaction types emerged as strong fraud indicators. Our categorical encoding and numerical scaling preprocessing pipeline significantly improved model discrimination capability.

**Evaluation Metrics Significance:** Traditional accuracy metrics can be misleading in imbalanced datasets. Our focus on precision, recall, and F1-scores provided more meaningful performance assessment for fraud detection applications.

**Real-time Deployment Considerations:** Flask web application deployment demonstrated the importance of model serialization, efficient prediction pipelines, and user-friendly interfaces for practical fraud detection systems.

### Ethical Considerations in Fraud Detection:

- *Bias Prevention:* ML models can perpetuate biases in historical data, potentially discriminating against certain demographic groups or geographic regions
- *Privacy Protection:* Transaction data requires careful handling to maintain customer privacy while enabling effective fraud detection
- *False Positive Impact:* Incorrect fraud flags can severely disrupt legitimate customer transactions, requiring careful precision-recall optimization

**Visualization and Interpretability:** Confusion matrix generation and performance dashboards proved crucial for model validation, stakeholder communication, and continuous monitoring of system performance.

**Scalability and Production Readiness:** Our implementation highlighted the importance of efficient model persistence, error handling, and scalable web architecture for production fraud detection systems.

## Key Learnings

During the comprehensive training program, we acquired a diverse range of skills and deep understanding of AI/ML concepts:

### Core Machine Learning Concepts:

**Supervised Learning Mastery:** Gained comprehensive understanding of classification algorithms, training methodologies, and evaluation techniques specifically applied to fraud detection scenarios.

**Ensemble Methods:** Learned how Random Forest combines multiple decision trees to create robust, high-performance predictions while reducing overfitting risks.

**Tree-based Algorithms:** Mastered Decision Tree implementation, understanding decision boundaries, feature importance, and interpretability benefits for regulatory compliance.

**Class Imbalance Handling:** Developed expertise in SMOTE technique for synthetic sample generation, understanding its impact on model performance and generalization.

### Advanced Programming Skills:

- *Python Proficiency:* Developed strong programming skills in pandas for data manipulation, scikit-learn for ML implementation, and Flask for web development.
- *Pipeline Development:* Learned to create integrated ML pipelines using ColumnTransformer, combining preprocessing steps with model training for production-ready systems.
- *Model Deployment:* Gained hands-on experience in model serialization using joblib, Flask web application development, and real-time prediction systems.

### Data Science Methodology:

- *Feature Engineering:* Understanding which transaction characteristics indicate fraud risk and how to properly encode and scale different data types.
- *Model Evaluation:* Mastered precision, recall, F1-score, and confusion matrix analysis for imbalanced datasets, learning when each metric is most appropriate.
- *Performance Optimization:* Learned techniques for hyperparameter tuning, cross-validation, and model comparison for optimal fraud detection performance.

### Industry Best Practices:

- *Version Control:* Implemented Git for tracking model iterations, code changes, and collaborative development practices.
- *Documentation:* Developed skills in creating comprehensive technical documentation for model maintenance and knowledge transfer.
- *Testing and Validation:* Learned robust testing procedures for ML pipelines, including edge case handling and error management.

### Business Understanding:

- *Domain Knowledge:* Gained insight into financial fraud patterns, regulatory requirements, and business impact of false positives/negatives in fraud detection.
- *Stakeholder Communication:* Developed ability to explain complex ML concepts to non-technical users and translate business requirements into technical solutions.

## Real-World Applications of AI/ML

### ➤ Financial Services

- *Fraud Detection*: Real-time identification of suspicious transactions (e.g., Visa, Mastercard).
- *Credit Risk & Scoring*: Predict loan defaults, automate credit decisions.
- *Algorithmic Trading*: Market prediction, portfolio optimization, high-frequency trading.
- *AML Compliance*: Detects patterns linked to money laundering activities.

### ➤ E-commerce & Payments

- *Payment Fraud Prevention*: Protects users from fraud and identity theft (e.g., PayPal, Stripe).
- *Account Security*: Detects suspicious login behavior and prevents unauthorized access.
- *Dynamic Pricing*: Adjusts prices in real time based on demand and competition.

### ➤ Healthcare & Insurance

- *Insurance Fraud Detection*: Identifies fake claims and unusual billing patterns.
- *Medical Diagnosis*: Supports doctors with image analysis and disease prediction.
- *Clinical Trials*: Optimizes patient selection and predicts outcomes.

### ➤ Technology & Cybersecurity

- *Threat Detection*: Identifies malware and prevents cyberattacks.
- *Spam & Phishing Filters*: Blocks malicious emails using ML.
- *Content Moderation*: Detects and removes harmful or policy-violating content.

### ➤ Transportation & Logistics

- *Autonomous Vehicles*: Enables real-time decision making and object detection.
- *Supply Chain Optimization*: Improves delivery routes, inventory, and demand forecasting.

## Conclusion

The **Credit Card Fraud Detection System** project provided an invaluable opportunity to bridge theoretical AI/ML knowledge with practical, real-world implementation. We successfully built a robust fraud detection pipeline that showcases the real-world value of machine learning in financial security.

### Key Achievements

- *ML Development:* Built and deployed Random Forest and Decision Tree models with complete preprocessing, SMOTE balancing, and evaluation metrics.
- *Web App Integration:* Developed a Flask-based interface for real-time predictions, sample data input, and model comparison.
- *Optimization:* Applied advanced techniques to handle class imbalance and automate performance evaluation.
- *Deployment:* Demonstrated model serialization, error handling, and scalable architecture.

### Skills Gained

- End-to-end ML pipeline design with ColumnTransformer, SMOTE, and joblib.
- In-depth understanding of ensemble and interpretable ML models.
- Practical Flask integration for deploying ML models as web services.

### Business Impact

This project demonstrated how AI/ML enhances financial security, reduces manual effort, speeds up fraud detection, and improves customer trust through automation.

### Industry Readiness

We are now well-prepared for careers in data science, ML engineering, and fintech, with hands-on experience directly applicable to real-world challenges in fraud prevention.

### Future Scope

- Incorporate advanced models like **XGBoost** or **neural networks**.
- Use **Kafka** for real-time data streaming.
- Enhance interpretability with **LIME/SHAP**.
- Deploy on **cloud platforms** for scalable solutions.

This project laid a solid foundation in applied machine learning and positioned us to contribute meaningfully to AI-driven innovations in financial security.