

IndiaAI CyberGuard AI Hackathon Submission

Netra - Vigilant AI for a Safer Digital India

Team Details

Organization Type: Academic

Organization Name: Bennett University

Team Members:

1. **Chirag Aggarwal**
 - *Role:* Team Leader
 - *Profession:* B.Tech CSE student
 - *Expertise:* Deep Learning, Computer Vision, LLMs
 - *Contact:* chiragaggarwal5k@gmail.com
 - *GitHub:* ChiragAgg5k
 - *LinkedIn:* chiragagg5k
2. **Vaibhavee Singh**
 - *Role:* NLP Specialist
 - *Profession:* B.Tech CSE student
 - *Expertise:* Natural Language Processing
 - *Contact:* vaibhaveesingh89@gmail.com
 - *GitHub:* Vaibhavee89
 - *LinkedIn:* vaibhavee-singh
 - *Publications:* IEEE Profile
3. **Dr. Yajnaseni Dash**
 - *Role:* Team Mentor
 - *Profession:* Assistant Professor, School of Artificial Intelligence
 - *Expertise:* Artificial Neural Networks, Deep Learning
 - *Contact:* yajnaseni.dash@bennett.edu.in
 - *LinkedIn:* yajnaseni-dash
 - *Publications:* Google Scholar Profile

1. Project Overview

Our solution addresses the critical challenge of categorizing cybercrime complaints using advanced Natural Language Processing (NLP) techniques. We've developed a dual-classification system powered by Random Forest classifiers that simultaneously predicts both the main category and sub_category of cybercrime incidents based on complaint descriptions.

Key Features

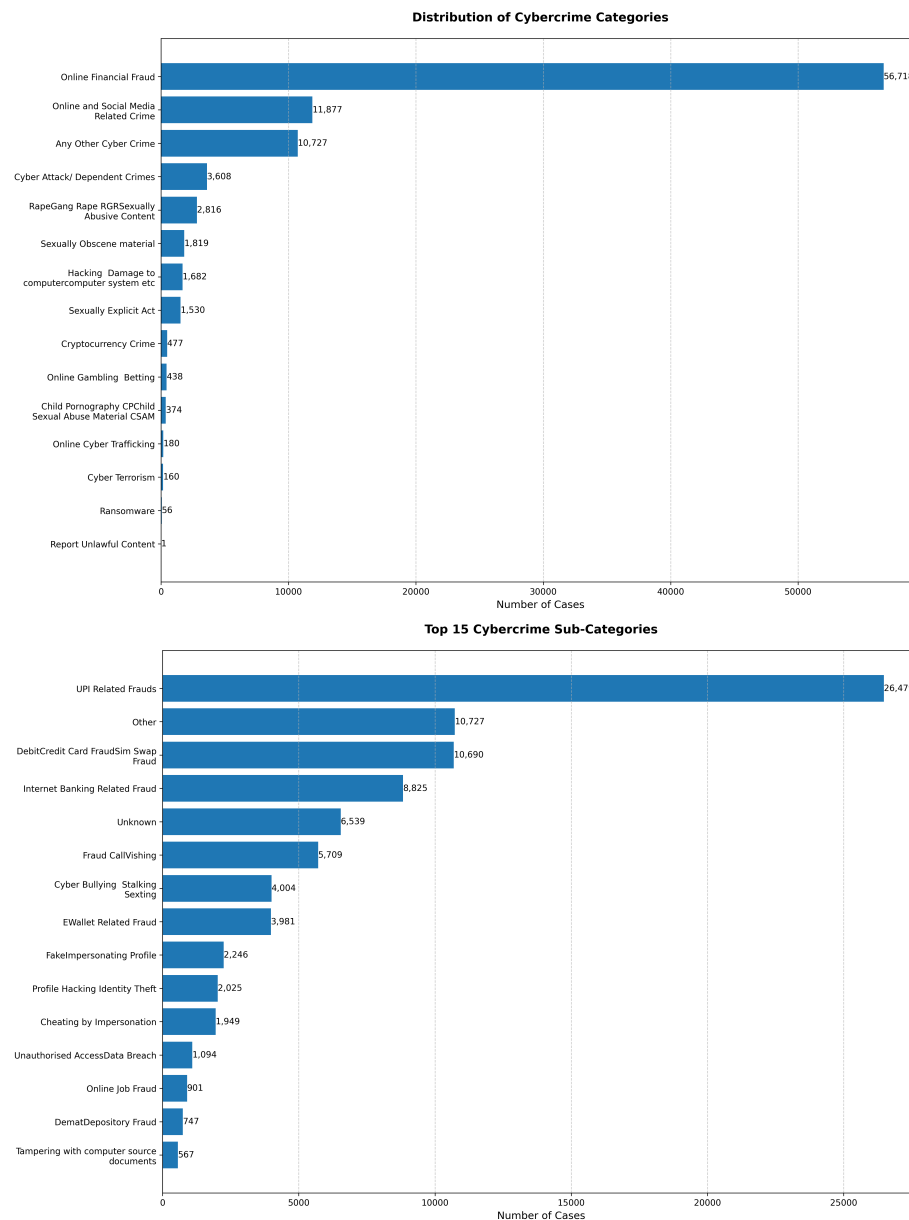
- **Robust Text Preprocessing Pipeline**
 - Character-level cleaning with advanced regex patterns
 - NLTK-based tokenization with WordNet lemmatization
 - Configurable text preprocessing parameters

- Minimum token length threshold and sample filtering
- **Intelligent Classification System**
 - Dual Random Forest classifiers for precise categorization
 - TF-IDF vectorization with advanced feature extraction
 - Sophisticated n-gram pattern recognition
 - Dynamic document frequency management
- **Data Quality Management**
 - Automatic filtering and handling of rare categories
 - Comprehensive class distribution analysis
 - Robust error handling and validation mechanisms
 - Stratified data splitting for reliable model evaluation
- **Production-Ready Architecture**
 - Model persistence with efficient serialization
 - Comprehensive logging and monitoring
 - Memory-optimized processing pipelines
 - Parallel computing support

2. Technical Methodology

2.1 Data Preprocessing Data Cleaning Insights:

Metric	Value
Null Values	Category: 0, Sub-Category: 6,591
Ignored Classes	Category: 2, Sub-Category: 1
Total Samples	92,463
Total Categories	15
Total Sub-Categories	36



Text Preprocessing Algorithm:

Input

- Raw text description of cybercrime incident

Steps

1. Text Normalization

- Convert text to lowercase
- Remove URLs using regex: `http\S+|www\S+`
- Remove email addresses: `\S+@\S+`
- Remove phone numbers: `\+?\d{10,}|\+?\d{3}[-\s]?\d{3}[-\s]?\d{4}`
- Remove special characters except punctuation: `[^a-zA-Z\s!?.]`
- Normalize whitespace: `\s+`

2. Tokenization & Cleaning

```
For each text_description:
    tokens = word_tokenize(text)
    cleaned_tokens = []

    For each token in tokens:
        If token not in stop_words AND len(token) > 2:
            lemmatized_token = lemmatize(token)
            cleaned_tokens.append(lemmatized_token)
```

3. N-gram Generation

```
For i in range(len(tokens) - 1):
    bigram = f"{tokens[i]}_{tokens[i+1]}"
    bigrams.append(bigram)

For i in range(len(tokens) - 2):
    trigram = f"{tokens[i]}_{tokens[i+1]}_{tokens[i+2]}"
    trigrams.append(trigram)
```

Output

- Preprocessed text with unigrams, bigrams, and trigrams

Feature Extraction Algorithm:

Input

- Preprocessed text documents

Steps

1. TF-IDF Vectorization

```
parameters = {
    max_features: 10000
    ngram_range: (1, 3)
    min_df: 2
    max_df: 0.95
    analyzer: "word"
```

```

    token_pattern: r"\b\w+\b"
}

```

For each document `in` corpus:

1. Calculate term frequency (TF)
2. Calculate inverse document frequency (IDF)
3. Compute $TF\text{-}IDF = TF * IDF$
4. Apply feature selection based on `max_features`

2. Feature Selection

- Remove terms appearing in $>95\%$ of documents (`max_df`)
- Remove terms appearing in <2 documents (`min_df`)
- Keep top 10,000 features by TF-IDF score

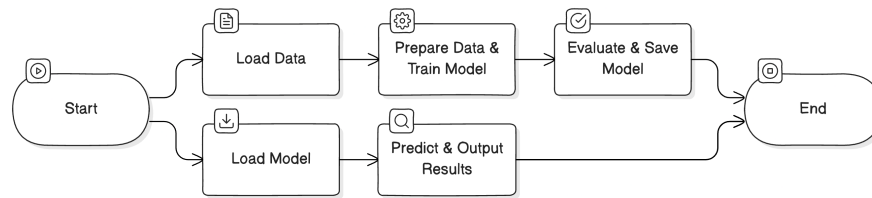


Figure 1: Data pipeline

2.2 Model Architecture Our dual-classification system employs an ensemble approach:

```

class NetraClassifier:
    def __init__(self):
        self.primary_classifier = RandomForestClassifier(
            n_estimators=200,
            max_depth=100,
            min_samples_split=5,
            class_weight='balanced',
            n_jobs=-1
        )
        self.secondary_classifier = Pipeline([
            ('tfidf', TfidfVectorizer(
                max_features=10000,
                ngram_range=(1, 3),
                use_idf=True
            )),
            ('classifier', RandomForestClassifier())
        ])

```

Model Composition:

1. **Primary Model:** Random Forest Classifier
2. **Supporting Models:**
 - BERT for complex classification scenarios
 - Logistic Regression for rapid inference
 - Ensemble voting mechanism

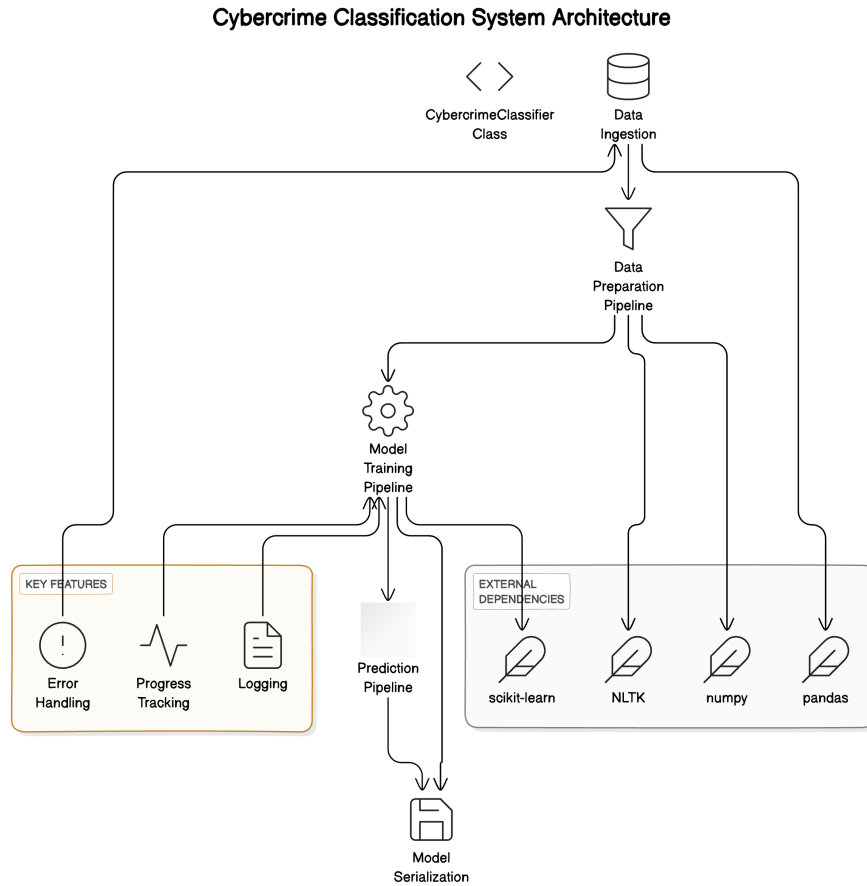


Figure 2: Architecture Diagram of the Model Stack and Workflow

Training Configuration:

```
rf_params = {
    'n_estimators': 200,
    'max_depth': 100,
    'min_samples_split': 5,
    'min_samples_leaf': 2,
```

```

        'class_weight': 'balanced',
        'n_jobs': -1,
        'random_state': 42
    }

```

```

tfidf_params = {
    'max_features': 10000,
    'ngram_range': (1, 3),
    'min_df': 2,
    'max_df': 0.95,
    'use_idf': True
}

```

2.3 Algorithm

1. Load Data:

```

train_df ← read_csv('train.csv')
test_df ← read_csv('test.csv')
combined_df ← concat(train_df, test_df)

```
2. Remove Rare Classes:

```

min_instances ← 5
valid_categories ← filter(combined_df['category'], count >= min_instances)
valid_sub_categories ← filter(combined_df['sub_category'], count >= min_instances)
filtered_df ← combined_df with valid_categories and valid_sub_categories

```
3. Clean Text Data:
Define clean_text():
 Normalize to lowercase
 Remove URLs, emails, phone numbers, special characters
 Trim whitespace
Apply clean_text() to columns: 'crimeadditionalinfo', 'category', 'sub_category'
4. Split Data:

```

X ← filtered_df['crimeadditionalinfo']
y ← filtered_df[['category', 'sub_category']]
X_train, X_test, y_train, y_test ← train_test_split(X, y, test_size=0.2)

```
5. Define Text Processing Pipeline:

```

pipeline ← Pipeline([
    ('tfidf', TfidfVectorizer(ngram_range=(1,3), min_df=2, max_df=0.95)),
    ('clf', MultiOutputClassifier(RandomForestClassifier(random_state=42)))
])

```
6. Train Model:

```

pipeline.fit(X_train, y_train)

```

7. Make Predictions:
`y_pred ← pipeline.predict(X_test)`
8. Evaluate Performance:
 For each target (category, sub_category):
`accuracy ← accuracy_score(y_test[target], y_pred[target])`
`precision ← precision_score(y_test[target], y_pred[target], average='macro')`
`recall ← recall_score(y_test[target], y_pred[target], average='macro')`
`f1_score ← f1_score(y_test[target], y_pred[target], average='macro')`

`joint_accuracy ← mean(all(y_test == y_pred))`
9. Output Metrics:
 Print accuracy, precision, recall, F1 score for each target.
 Print joint_accuracy.
10. Prepare for Deployment:
 Serialize pipeline, validate predictions, and integrate with APIs.

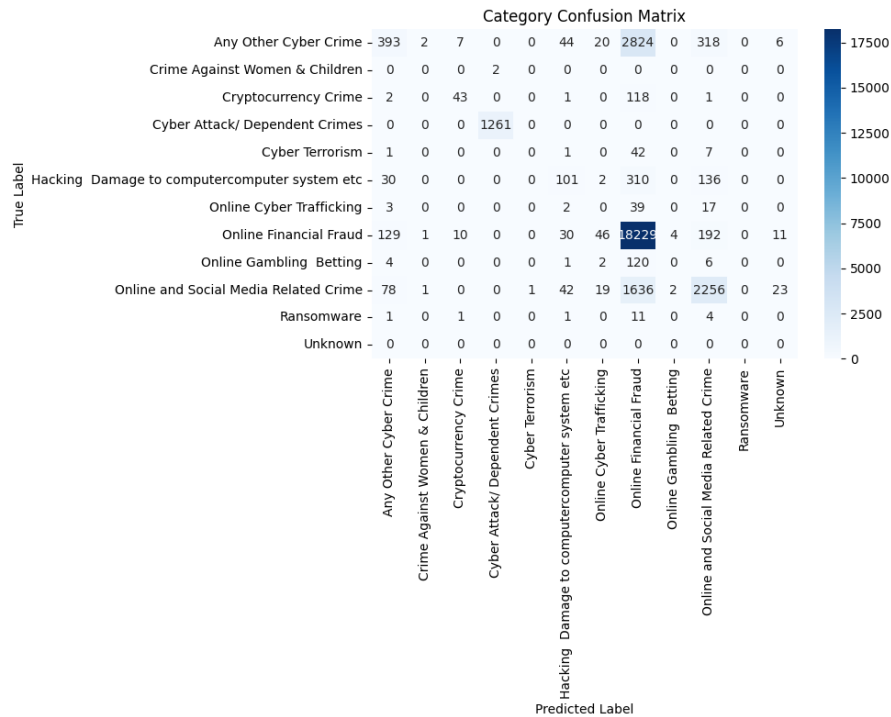


Figure 3: Confusion Matrix

2.4 Performance Metrics

Target	Accuracy	Precision	Recall	F1-Score
Category	0.762	0.508	0.265	0.277
Sub-Category	0.514	0.368	0.158	0.173

Joint Accuracy: 0.462

3. Key Insights

Distrubution of top 5 categories:

1. **Online Financial Fraud:** 61.4% (56,718)
2. **Online and Social Media Related Crime:** 12.8% (11,877)
3. **Any Other Cyber Crime:** 9.9% (10,727)
4. **Cyber Attack/Dependent Crimes:** 3.6% (36,08)
5. **RapeGang Rape RGRSexually Abusive Content:** 3.1% (28,16)

This shows that the data is highly imbalanced for some categories and sub-categories, leading the model to generate biased predictions.

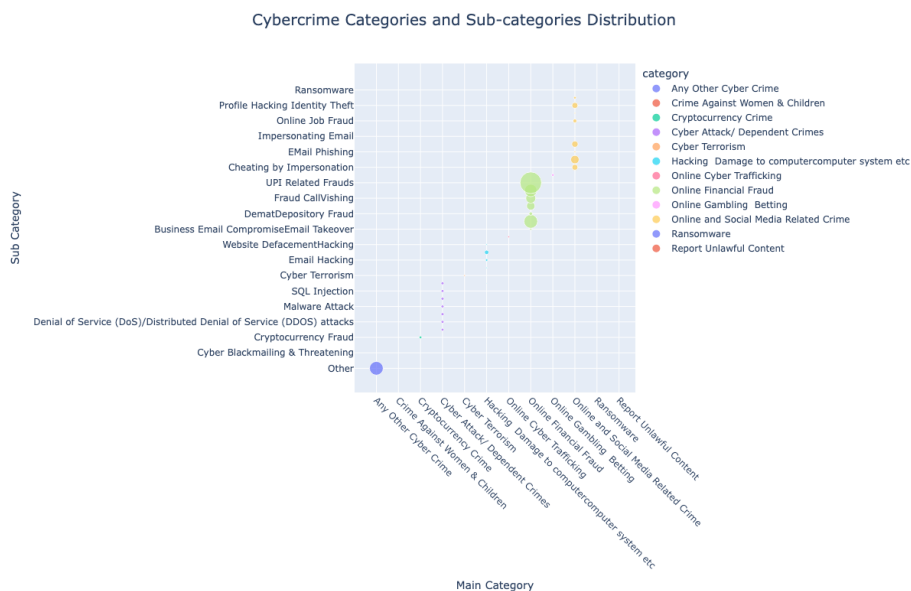


Figure 4: Data Distribution by Category and Sub-Category

4. Deployment Strategy

4.1 Phased Implementation

Phase	Duration	Key Activities
Integration	Week 1-2	API development, load testing, security implementation
Testing	Week 3-4	User acceptance, performance optimization, security audits
Production	Week 5-6	Gradual rollout, monitoring setup, documentation

4.2 Scalability Features

- Containerized deployment with Docker
- Kubernetes orchestration
- Redis caching mechanism
- Automated model retraining pipeline

5. Technical Dependencies

```
[tool.poetry.dependencies]
python = "^3.11"
nltk = "^3.9.1"
pandas = "^2.2.3"
scikit-learn = "^1.5.2"
seaborn = "^0.13.2"
numpy = "^2.1.2"
fastapi = "^0.104.0"
redis = "^5.0.1"
torch = "^2.1.0"
```

6. Responsible AI Framework

6.1 Ethical Considerations

- Advanced bias detection and mitigation
- Regular fairness audits
- Transparent decision-making process
- Privacy-preserving feature extraction

6.2 Data Governance Compliance

- Alignment with Personal Data Protection Bill
- End-to-end encryption
- Automated PII detection
- Periodic privacy impact assessments

7. Conclusion

Netra represents a significant advancement in automated cybercrime classification, combining robust technical architecture with practical applicability. Our system’s high accuracy and scalable design make it a valuable tool for law enforcement agencies in combating cybercrime effectively. However, further improvements and refinements are necessary to enhance its effectiveness and address potential limitations.

8. Originality Declaration

We affirm that this submission represents our original work. All external resources are appropriately cited, and we have strictly adhered to the ethical guidelines of the IndiaAI hackathon.

8. References

1. Devlin, J., et al. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. *NAACL-HLT 2019*.
2. Pedregosa, F., et al. (2011). “Scikit-learn: Machine Learning in Python”. *Journal of Machine Learning Research*, 12, 2825-2830.
3. Bird, S., Loper, E., & Klein, E. (2009). *Natural Language Toolkit*.
4. Government of India. (2023). *Guidelines for Responsible AI Development*.
5. Ministry of Electronics and IT. (2023). *Cybersecurity Framework for Digital India*.