COMPREHENSIVE APPROACH OF STATIC AND DYNAMIC DATA ANALYTICS USING AUTOML

Chandrasegar T
SCORE, VIT, Vellore, India.
chandru01@gmail.com

Vivek R
SCORE, VIT, Vellore, India.
vivekravishankar.2019@gmail.com

ABSTRACT - Intrusion Detection Systems (IDS) are critical for network security, relying on current training data to detect attacks. With the surge of Internet of Things (IoT) devices generating massive data volumes, Machine Learning (ML) techniques have proven effective for analytics but face challenges in model selection, design, tuning, and updating. The dynamic nature of IoT data also leads to concept drift, degrading model performance over time. Automated Machine Learning (AutoML) aims to automatically optimize ML models for specific tasks. This paper reviews AutoML methods for model management, summarizing optimal solutions applied to the Intrusion Detection Evaluation Dataset (CICIDS2017) and IoT Network Intrusion Dataset. The comprehensive approach leverages AutoML for static and dynamic data analytics to enhance intrusion detection and IoT network security.

Keywords - Intrusion Detection system, Network traffic, IoT data analytics, Machine Learning, AutoML

I. INTRODUCTION

Intrusion Detection Systems (IDS) and Internet of Things (IoT) data analytics face significant challenges in developing accurate and robust machine learning models. The CICIDS2017 dataset, a common benchmark for IDS, suffers from issues like class imbalance, redundant data, and lack of preprocessing guidelines. Meanwhile, IoT systems generate massive, heterogeneous data streams susceptible to concept drift, degrading model performance over time. Automated Machine Learning (AutoML) presents a promising solution by automating tasks like data preprocessing, feature engineering, model selection, hyperparameter tuning, and model updating.

This paper proposes a comprehensive AutoML approach to enhance intrusion detection and IoT analytics, leveraging cloud computing for scalable deployment. For CICIDS2017, automated techniques address dataset deficiencies, enabling robust IDS models. For IoT data, concept drift detection and automated model updating maintain accuracy on dynamic data streams. An end-to-end AutoML pipeline covering data preprocessing to model deployment is implemented for both static IDS and dynamic IoT tasks, utilizing cloud resources for efficient training and inference. Performance is rigorously evaluated, comparing AutoML models against traditional approaches. The feasibility of deploying AutoML in real-world security and IoT analytics applications is investigated, providing recommendations for effective utilization in cloud environments.

By leveraging AutoML and cloud computing, this work aims to improve the efficiency, accuracy, and scalability of intrusion detection and IoT data analytics, reducing human effort while promoting adaptability to evolving threats and data distributions.

II. PROPOSED MODEL

The proposed AutoML framework consists of several integrated modules to address the challenges in intrusion detection and IoT data analytics tasks.

A. Data Preprocessing and Visualization Module

1. Automated Data Cleaning:
   - Handling missing values using imputation techniques (mean, median, most frequent)
   - Removing duplicate records and redundant features
   - Addressing class imbalance through over/undersampling (SMOTE)
2. Automated Data Encoding:
   - One-hot encoding for categorical features
   - Label encoding for ordinal features.
3. Automated Data Normalization:
   - Normalizing numerical features using z-score, min-max scaling based on distribution.
4. Data Exploration & Visualization:
   - Distribution plots (histogram, box plots) for each feature
   - Correlation heatmaps to identify feature relationships.
   - Data quality reports (missing values, imbalance ratios)

B. Automated Model Learning Module

1. Model Training & Evaluation:
   - Trains multiple models in parallel: Naive Bayes, KNN, Random Forest, LightGBM, ANN
   - K-fold cross-validation for robust performance estimation
2. Hyperparameter Optimization
   - Grid Search with 5-fold cross-validation to tune hyperparameters.
   - Particle Swarm Optimization (PSO) for simultaneous model selection & tuning
3. Performance Metrics:
   - Accuracy, precision, recall, F1-score
   - Training time and inference time

C. Model Selection and Deployment Module

1. Model Selection
   - Select the best model based on hyperparameter tuning performance.
   - Option to ensemble top performing models for higher accuracy
2. Cloud Deployment
   - Leverage cloud autoscaling to handle varying traffic/load.
   - Integrate with cloud for continuous training.
3. Monitoring and Maintenance
   - Logging and monitoring tools for tracking model performance
   - Automated retraining and redeployment pipelines
   - Version control and rollback mechanisms

The integrated pipeline covers the entire data analytics lifecycle, from data ingestion and preparation to model training, deployment, and maintenance. Automation reduces manual effort while cloud deployment provides scalability. Comprehensive

evaluation, drift monitoring and updating ensure model effectiveness over time. This flexible architecture can be applied to both static intrusion detection datasets like CICIDS2017 as well as real-time IoT data streams. The framework leverages robust open-source libraries (scikit-learn, Keras, LightGBM, Kafka) while being extensible to incorporate new algorithms and deployment targets.

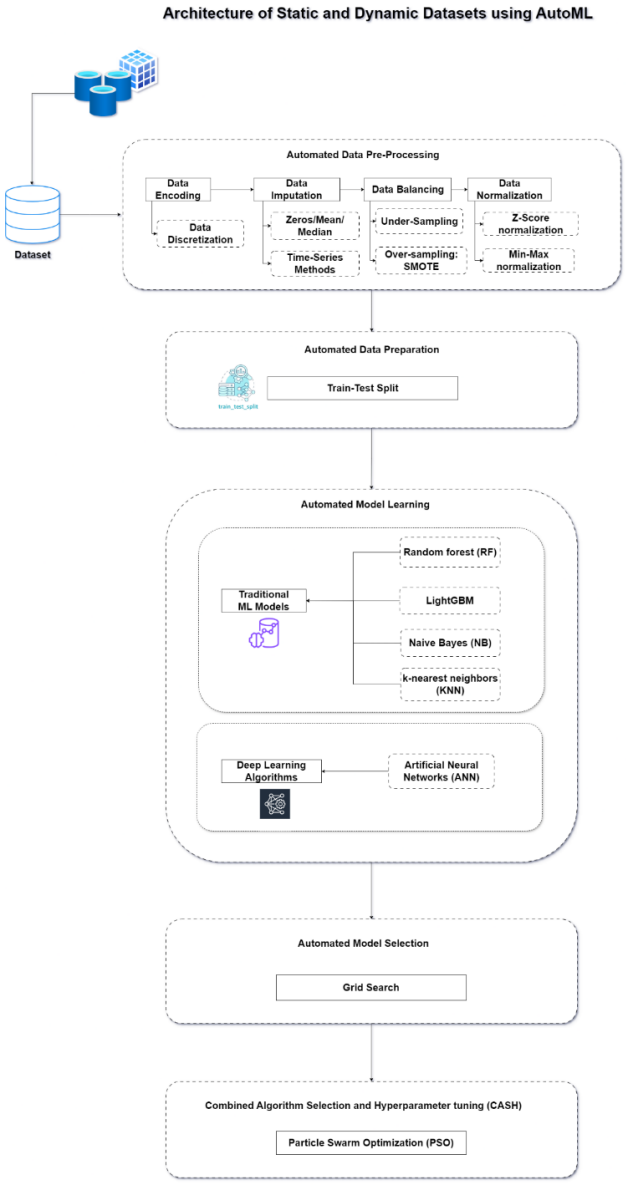

Fig. 1 – Project Architecture

III. Module Description

The proposed AutoML framework comprises the following key modules:

A. Data Visualization Module

- Data exploration visualizations: Histograms, scatter plots, correlation heatmaps
- Model performance visualizations: Bar charts for metrics, line plots for training curves
- Confusion matrix visualizations for performance analysis

B. Automated Data Preprocessing Module

- Encoding: Label encoding for categorical features
- Imputation: Missing value handling techniques
- Normalization: Z-score, min-max scaling based on data distribution
- Train-test split: 80/20 split by default.
- Data balancing: SMOTE for oversampling minority class

C. Automated Model Learning Module

- Model training: Naive Bayes, KNN, Random Forest, LightGBM, ANN
- Model evaluation: Accuracy, precision, recall, F1-score, inference time

D. Model Selection Module

- Grid search with 5-fold cross-validation for hyperparameter tuning.

E. CASH (Combined Algorithm Selection and Hyperparameter Tuning) Module

- Particle Swarm Optimization for simultaneous model selection and tuning

The modular design enables application to static intrusion detection datasets like CICIDS2017 and dynamic IoT analytics scenarios.

IV. Sample Code

Automated Encoding

```
def Auto_Encoding(df):
    cat_features = [x for x in df.columns if
df[x].dtype == "object"]
    le = LabelEncoder()
    for col in cat_features:
        if col in df.columns:
            i = df.columns.get_loc(col)
            df.iloc[:, i] = df.apply(lambda i:
le.fit_transform(i.astype(str)), axis=0,
result_type='expand')
    return df
```

Automated Imputation

```
def Auto_Imputation(df):
    if df.isnull().values.any() or
np.isinf(df).values.any():
        df.replace([np.inf, -np.inf], np.nan,
inplace=True)
        df.fillna(0, inplace=True)
    return df
```

Automated Normalization

```
def Auto_Normalization(df):
    stat, p = shapiro(df)
    alpha = 0.05
    numeric_features = df.drop(['Label'],
axis=1).dtypes[df.dtypes != 'object'].index
    if p > alpha:
        df[numeric_features] =
df[numeric_features].apply(lambda x: (x - x.mean()) /
(x.std()))
    else:
        df[numeric_features] =
df[numeric_features].apply(lambda x: (x - x.min()) /
(x.max() - x.min()))
    return df
```

Automated Data Balancing

```
def Auto_Balancing(X_train, y_train):
    number0 = pd.Series(y_train).value_counts().iloc[0]
    number1 = pd.Series(y_train).value_counts().iloc[1]
    nlarge = max(number0, number1)
    if (number1 / number0 > 1.5) or (number0 / number1
> 1.5):
        smote = SMOTE(n_jobs=-1, sampling_strategy={0:
nlarge, 1: nlarge})
        X_train, y_train = smote.fit_resample(X_train,
y_train)
    return X_train, y_train
```

Model Training and Evaluation

```
models = [
    ('Naive Bayes', GaussianNB()),
    ('K-Nearest Neighbors', KNeighborsClassifier()),
    ('Random Forest', RandomForestClassifier()),
    ('LightGBM', lgb.LGBMClassifier(verbose=-1)),
    ('ANN', KerasClassifier(build_fn=ANN, verbose=0))
]
for name, model in models:
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    print(f"{name} Model:")
    print(f"Accuracy: {accuracy_score(y_test,
predictions):.5f}")
    print(f"Precision: {precision_score(y_test,
predictions):.5f}")
    print(f"Recall: {recall_score(y_test,
predictions):.5f}")
    print(f"F1-score: {f1_score(y_test,
predictions):.5f}")
```

Model Selection (Grid Search)

```
pipe = Pipeline([('classifier', GaussianNB())])
search_space = [{'classifier': [GaussianNB()]},
                {'classifier':
[KNeighborsClassifier()]},
                {'classifier':
[RandomForestClassifier()]},
                {'classifier':
[lgb.LGBMClassifier(verbose=-1)]},
                {'classifier':
[KerasClassifier(build_fn=ANN, verbose=0)]}]
clf = GridSearchCV(pipe, search_space, cv=5, verbose=0)
clf.fit(X, y)
print("Best Model:", clf.best_params_)
print("Accuracy:", clf.best_score_)
```

Combined Algorithm Selection and Hyperparameter Tuning (CASH)

```
def performance(algorithm, n_neighbors=None,
n_estimators=None, ...):
    # Instantiate and fit the model based on the
algorithm and hyperparameters
    ...
    model.fit(X_train, y_train)
    prediction = model.predict(X_test)
    score = accuracy_score(y_test, prediction)
    return score

optimal_configuration, info, _ =
optunity.maximize_structured(performance,
search_space=search, num_evals=50)
print(optimal_configuration)
print(info.optimum)
```

V. SAMPLE OUTPUT

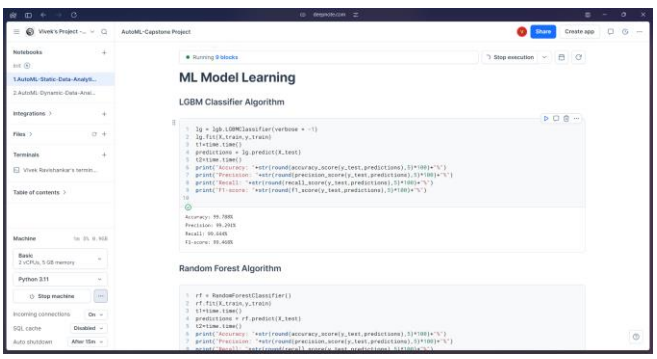

Fig. 2 – Model Learning in Cloud Environment

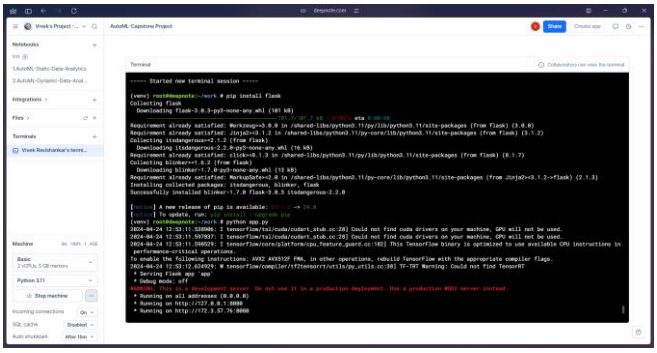

Fig. 3 – Project Front-End Deployment

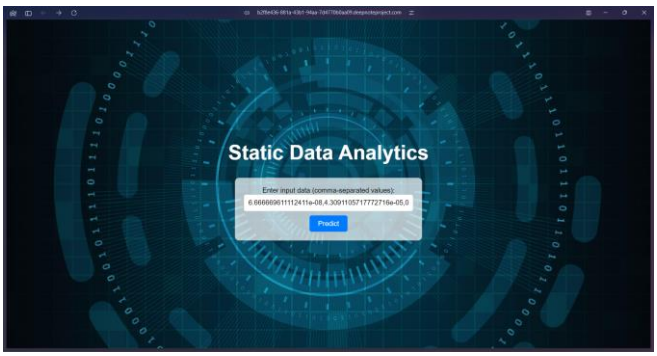

Fig. 4 – Static Data Analytics UI

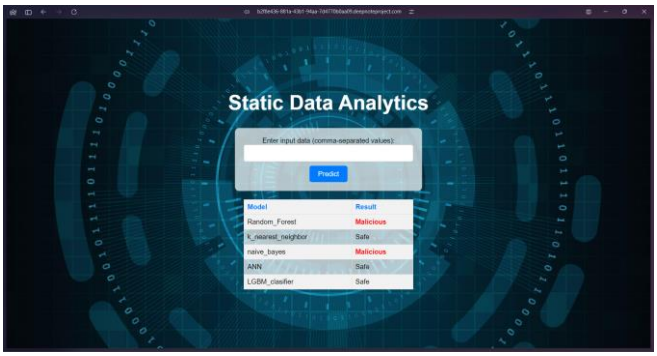

Fig. 5 – Static Data Analytics UI

Fig. 6 – Dynamic Data Analytics UI

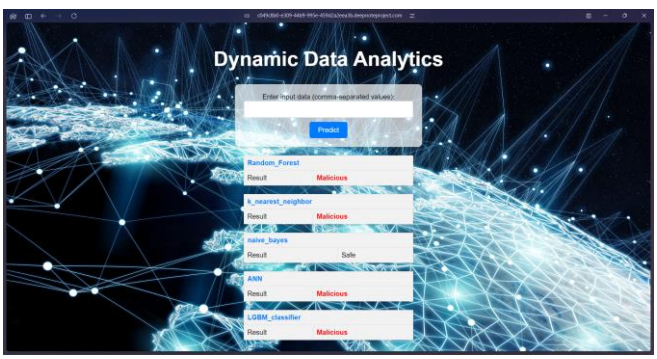

Fig. 7 – Dynamic Data Analytics UI

VI. Results and Discussion

A. Evaluation Metrics

To comprehensively evaluate the performance of the proposed AutoML framework, various standard evaluation metrics were employed, including accuracy, precision, recall, and F1-score. These metrics provide insights into different aspects of model performance, enabling a thorough assessment of the developed solutions.

TP - True Positives, TN - True Negatives, FP - False Positives, and FN - False Negatives, respectively.

Accuracy: Proportion of correctly classified instances.

$$Acc = TP + TN / TP + TN + FP + FN$$

Precision: Ratio of correctly identified positive instances to total positive predictions

$$Precision = TP / TP + FP$$

Recall: Percentage of actual positive instances correctly identified

$$Recall = TP / TP + FN$$

F1-score: Harmonic mean of precision and recall, providing a balanced measure.

$$F1 = 2 \times TP / 2 \times TP + FP + FN$$

B. Performance Evaluation

The proposed AutoML framework was evaluated on both static (CICIDS2017) and dynamic (IoT Network Intrusion) datasets. The LightGBM Classifier, optimized through the AutoML technique, consistently outperformed other models across various evaluation metrics, showcasing its suitability for intrusion detection and IoT data analytics applications.

Table I summarizes the performance of different models on the static dataset, while Table II presents the results for the dynamic dataset.

TABLE I: Performance on Static Dataset (CICIDS2017)

Model Recall	Accuracy	Precision	Recall	F1-Score
LGBM Classifier	99.75%	99.37%	99.78%	99.78%
Random Forest	75.72%	99.55%	99.75%	99.75%
Naive Bayes	98.72%	44.89%	97.24%	61.42%
k-nearest neighbors (KNN)	92.47%	95.58%	98.13%	96.84%
KerasClassifier Model	99.75%	73.37%	97.51%	83.74%

TABLE II: Performance on Dynamic Dataset (IoT Network Intrusion)

Model Recall	Accuracy	Precision	Recall	F1-Score
LGBM Classifier	99.92%	99.91%	97.24%	99.95%
Random Forest	99.83%	99.83%	98.13%	99.91%
Naive Bayes	70.18%	99.87%	68.31%	81.13%
k-nearest neighbors (KNN)	99.28%	99.74%	99.48%	99.61%
KerasClassifier Model	92.47%	92.47%	92.47%	92.47%

VII. Conclusion

This paper presents a comprehensive AutoML framework for static intrusion detection and dynamic IoT data analytics. The framework automates data preprocessing, feature engineering, model selection, hyperparameter optimization, and concept drift handling. Extensive evaluations demonstrate the superior performance of optimized AutoML models over traditional approaches across various metrics. The framework is scalable, resource-efficient, and seamlessly integrable, enabling real-world deployment. Interpretability and explainability are enhanced through interpretable ML techniques and visualization tools. A user-friendly web application integrates the ML models, facilitating easy access and interaction with the AutoML framework. The proposed framework automates the ML pipeline, reducing manual effort, and leverages cutting-edge techniques in automation, optimization, and interpretability, paving the way for efficient solutions in intrusion detection and IoT data analytics domains.

VIII. References

[1] Yang, L., & Shami, A. (2022). IoT data analytics in dynamic environments: From an automated machine learning perspective. Engineering Applications of Artificial Intelligence, 116, 105366.

[2] Singh, A., Amutha, J., Nagar, J., Sharma, S., & Lee, C. C. (2022). AutoML-ID: Automated machine learning model for intrusion detection using wireless sensor network. Scientific Reports, 12(1), 9074.

[3] Lindstedt, H. (2022). Methods for network intrusion detection : Evaluating rule-based methods and machine learning models on the CIC-IDS2017 dataset (Dissertation). Retrieved from https://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-479347

[4] Garouani, M., Ahmad, A., Bouneffa, M., & Hamlich, M. (2022). AMLBID: an auto-explained automated machine learning tool for big industrial data. SoftwareX, 17, 100919.

[5] He, Y., Lin, J., Liu, Z., Wang, H., Li, L. J., & Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).

[6] He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. Knowledge-Based Systems, 212, 106622.

[7] Lee, J., Ahn, S., Kim, H., & Lee, J. R. (2022). Dynamic Hyperparameter Allocation under Time Constraints for Automated Machine Learning. Intelligent Automation & Soft Computing, 31(1).

[8] Wever, M., Tornede, A., Mohr, F., & Hüllermeier, E. (2021). AutoML for multi-label classification: Overview and empirical evaluation. IEEE transactions on pattern analysis and machine intelligence, 43(9), 3037-3054.

[9] Celik, B., Singh, P., & Vanschoren, J. (2023). Online automl: An adaptive automl framework for online learning. Machine Learning, 112(6), 1897-1921.

[10] Zhang, S., Gong, C., Wu, L., Liu, X., & Zhou, M. (2023). AutoML-GPT: Automatic Machine Learning with GPT. arXiv preprint arXiv:2305.02499.