HINDUSTAN
INSTITUTE OF TECHNOLOGY & SCIENCE
(DEEMED TO BE UNIVERSITY)

TEAM NO: 6

AI MODEL ANALYSER

TEAM MEMBER:

SWEATHA B – 22143020

KARTHIKEYAN M – 22143002

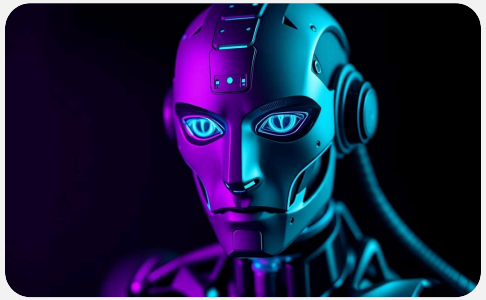
VISHNU V- 22143028

ML MODEL ANALYZER

Artificial Intelligence (AI) is widely used in various industries, but AI models often consume significant computational resources, leading to high energy consumption. To address this challenge, our project AI Model Analyzer evaluates and optimizes machine learning models based on their performance metrics and energy efficiency.

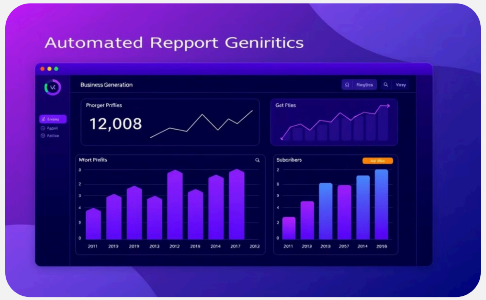
- Automate model evaluation by analyzing ML models from Python scripts.
- Assess energy consumption alongside traditional performance metrics like accuracy and execution time.
- Suggest alternative models that improve accuracy while reducing energy usage.
- Generate AI-powered reports summarizing model efficiency, data visualization and sustainability.

Innovation & Creativity



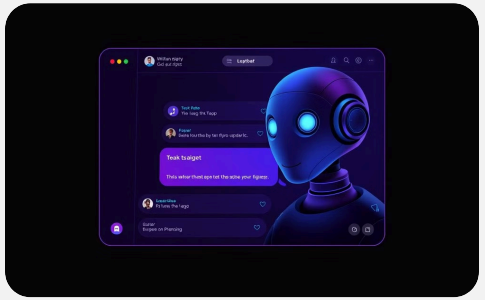
AI-Driven and Rule-Based model

Suggests optimized ML models and hyperparameters, identifies missing elements like optimizers and loss functions, and recommends best practices for better performance.



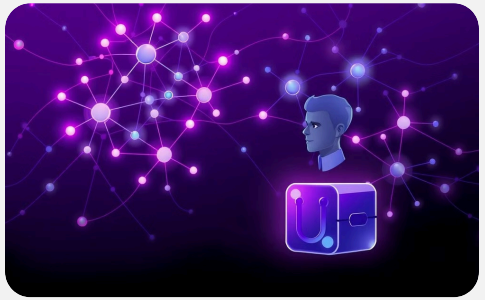
Automated Report Generation

Extracts key metrics (accuracy, precision, recall, etc.) and generates structured PDF reports with findings.



Interactive AI Chatbot

Answers ML-related queries and guides users in improving their models.



Visualization

visualizes the models accuray,F1 score, Energy consumption, confusion matrix.

Functionality & Feasibility

Upload ML Model File

Upload .py ML model for analysis.

Feature Extraction

Analyze model attributes (type, accuracy, loss functions, time).

AI-Based Recommendations

Suggest better models, tuning, and detect missing elements.

Download Report

Download report summarizing findings and recommendations.

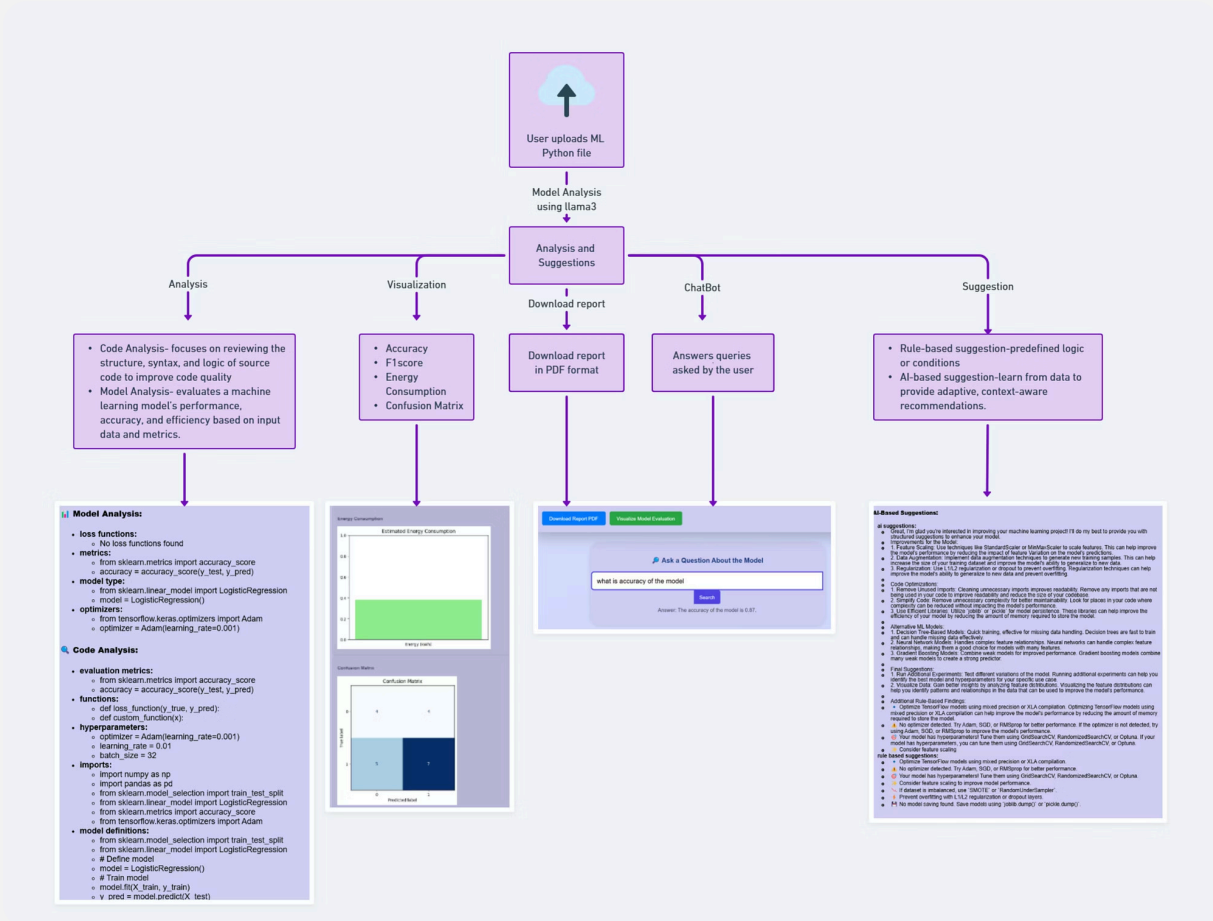
Chatbot

Answers ML-related queries and guides users in improving their models.

Visualization

visualizes the models accuray,F1 score, Energy consumption, confusion matrix.

ARCHITECTURE



Technical Stack

- **Frontend:** React.js,HTML,CSS – Provides an interactive UI
- **Backend:** Flask – API for model analysis & AI processing

Libraries:

- Scikit-learn – Extracts model details
- NumPy & Pandas – Handles numerical and tabular data
- Matplotlib & Seaborn – Visualization tools for insights
- PyPDF – Generates reports from extracted insights

NLP-Based AI Assistant (LLaMA 2)

- Analyzes model architecture
- Suggests improvements **based on best practices**
- Provides detailed **explanations for beginners**
- Performance Optimization
- Uses **Grid Search & Random Search** for hyperparameter tuning
- **Parallel Processing** speeds up analysis

Stakeholders

ML Engineers & Data Scientists

Optimize models without manual tuning, leveraging our intuitive platform for streamlined efficiency.

AI Researchers

Analyze large-scale ML systems with ease, gaining actionable insights to drive innovation and discovery.

Software Developers

Ensure models are efficient and seamlessly integrated into applications with our user-friendly tools and guidance.

Students & Beginners

Get AI-driven feedback on ML code, accelerating your learning journey with personalized recommendations.

Why It's User-Friendly?

No Complex Setup

Simple .py file upload enables quick analysis, eliminating the hassle of intricate configurations.

Real-Time AI Recommendations

Interactive chatbot assistance provides instant support, making model improvement intuitive and accessible.

Automated Report Generation

Easy-to-understand insights are delivered in structured reports, simplifying complex data for clear comprehension.

Sustainability

Bias Detection:

- Identifies **imbalanced datasets** and suggests **mitigation strategies**

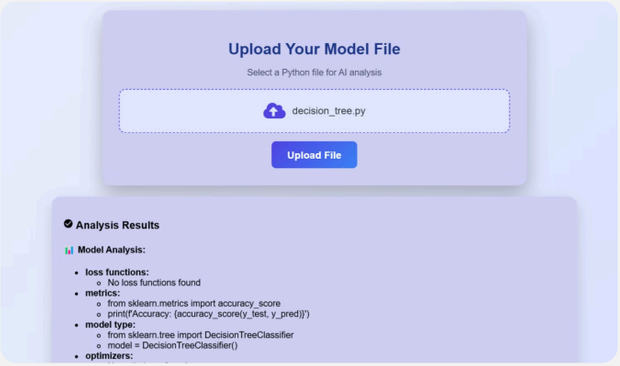
Explainability & Transparency:

- AI-generated **recommendations are explainable**
- Users understand **why a specific model is suggested**
- Security & Privacy:
- Uploaded ML models are **processed locally** (No data leakage)

Long-Term Sustainability-

- **Reduces computational waste** by suggesting efficient models
- **Encourages Green AI** by lowering energy consumption
- Can be deployed on Cloud for better accessibility

OUTPUT



AI-Based Suggestions:

- **ai suggestions:**
  - As a helpful AI assistant, I'm happy to provide you with structured suggestions for improving your machine learning project. Please find below my findings:
  - Improvements for the Model:
  - 1. Feature Scaling: Use techniques like StandardScaler or MinMaxScaler to scale features. This can help improve the model's performance by reducing the effects of feature bleeding.
  - 2. Data Augmentation: Implement data augmentation techniques to generate new training samples. This can help increase the size of your training dataset and improve the model's ability to generalize.
  - 3. Regularization: Use L1/L2 regularization or dropout to prevent overfitting. Regularization can help improve the model's ability to learn generalizable features and reduce overfitting.
  -
  - Code Optimizations:
  - 1. Remove Unused Imports: Clean up your code by removing unnecessary imports. This can improve the readability of your code and reduce the risk of unused imports.
  - 2. Simplify Code: Remove unnecessary complexity for better maintainability. This can help improve the maintainability of your code and reduce the risk of bugs.
  - 3. Use Efficient Libraries: Utilize `joblib` or `pickle` for model persistence. These libraries can help improve the performance of your model by reducing the amount of memory used for model storage.
  -
  - Alternative ML Models:
  - 1. Decision Tree-Based Models: Quick training and effective for missing data handling. Decision trees are fast to train and can handle missing data effectively.
  - 2. Neural Network Models: Handles complex feature relationships. Neural networks can handle complex feature relationships and learn non-linear patterns in the data.
  - 3. Gradient Boosting Models: Combine weak models for improved performance. Gradient boosting models can combine multiple weak models to create a strong model that performs better than any individual model.
  -
  - Final Suggestions:
  - 1. Run Additional Experiments: Test different variations of the model. This can help you identify the best combination of hyperparameters and features for your model.
  - 2. Visualize Data: Gain better insights by analyzing feature distributions. Visualizing the data can help you identify patterns and relationships that may not be immediately apparent.