**EcoVision User Journey Guide**

PoC/Working Demo

|  |  |
| --- | --- |
| **Contributors: EcoVision Team** | **Praveen Kumar Madhava Rao** |
| **Design & Presentation Leader** | **Vidyashree Bharathwaj** |
| **Design Architect** | **Kavitha SureshKumar** |
| **Design Architect** | **Sasikala Palanisamy** |
| **Project Manager** | **Iwona Kwiek** |
| **Developer Lead** | **Kavya Singh** |
| **Review & Advancement Lead** | **Pedro Soares** |
| **Visual Innovator** | **Chandan Bhat** |
| **Visual Innovator** | **BhavyaShree Mucherla** |
| **Visual Innovation Lead** | **Usha Sham** |
| **Mentor** | **Antony Pulicken** |

|  |  |
| --- | --- |
| **Version:** | **1.0** |
| **Date:** | **23/03/2024** |

**Table of Contents**

[**Preface** 3](#_Toc162984142)

[**AI Logical Framework:** 3](#_Toc162984143)

[**Model Deployment Workflow:** 3](#_Toc162984144)

[**Model Training:** 4](#_Toc162984145)

[**Real-time Integration and Deployment:** 5](#_Toc162984146)

[**Integration with EcoVision Platform** 5](#_Toc162984147)

[**Monitoring and Performance Optimization** 5](#_Toc162984148)

[**User Interaction and Experience** 5](#_Toc162984149)

[**Evaluation and Validation:** 5](#_Toc162984150)

[**Production Deployment:** 6](#_Toc162984151)

[**Conclusion:** 6](#_Toc162984152)

# **Preface**

In this guide, we present a comprehensive framework for implementing AI in EcoVision, focusing on training a Random Forest model to optimize pipeline execution for energy efficiency. Through step-by-step instructions and code examples, we aim to provide a clear path for integrating AI into EcoVision's DevSecOps pipeline. **Introduction:** This guide presents the EcoVision AI Logical Framework, aimed at improving energy efficiency within DevSecOps pipelines. By leveraging Random Forest models, organizations can strategically optimize pipeline execution, leading to informed decision-making, reduced energy consumption, and enhanced sustainability efforts.

**AI Logical Framework:**

* **Overview of DevSecOps Pipeline Energy Efficiency:**Efficiency in DevSecOps pipelines is paramount, with a growing emphasis on energy optimization. AI-driven solutions play apivotal role in enhancing energy efficiency, ensuring sustainable practices while maintaining operational excellence.
* **Introduction to Random Forest Algorithm:**Random Forest, a powerful ensemble learning technique, stands out for its ability to handle classification tasks with high accuracy and flexibility. Its application in recommending pipeline stages for skipping aligns seamlessly with the goals of optimizing energy consumption in DevSecOps workflows.
* **Framework for Integration:** The integration framework entails meticulous steps, encompassing data preprocessing, model training, real-time prediction, and continuous improvement. By adhering to this structured approach, EcoVision can seamlessly incorporate the Random Forest model, ensuring accurate recommendations and sustained energy efficiency gains.
* **Expected Benefits:** Implementing the EcoVision AI Logical Framework promises a multitude of benefits, including heightened energy efficiency, diminished carbon footprint, and enhanced resource utilization within DevSecOps pipelines. These outcomes underscore the transformative potential of AI-driven solutions in fostering sustainable practices across the software development lifecycle.

**Model Deployment Workflow:**

* **Data Preprocessing:** In this stage, raw data collected from DevSecOps pipeline executions undergoes cleaning and transformation to ensure consistency and quality. This involves handling missing values, removing outliers, and standardizing data formats.
* **Feature Engineering:** Feature engineering is the process of selecting and creating relevant features from the raw data to improve model performance. This includes identifying key variables that influence energy consumption in pipeline stages and creating new features to capture complex relationships.
* **Model Training:** The model training phase involves feeding preprocessed data into the Random Forest algorithm to build a predictive model. During training, the algorithm learns from historical data patterns to make accurate predictions about which pipeline stages to skip for energy efficiency.
* **Real-time Prediction:** Once the Random Forest model is trained, it can be integrated into EcoVision to provide real-time recommendations for skipping pipeline stages based on code changes. This enables developers to optimize energy consumption dynamically as they work on code modifications.
* **Continuous Improvement:** The process doesn't end with model deployment. Continuous monitoring and retraining are essential to ensure the model remains effective over time. By periodically updating the model with new data and refining its parameters, we can adapt to evolving DevSecOps workflows and maintain optimal energy efficiency.

1. **Top of Form**
2. **Bottom of Form**

**Model Training:**

* Initialize Random Forest: Create an instance of the Random Forest classifier from a machine learning library such as scikit-learn in Python.
* Train the Model: Fit the Random Forest model to the training data. During this process, the algorithm will build multiple decision trees based on random subsets of features and data samples.
* Hyperparameter Tuning: Tune the hyperparameters of the Random Forest model to optimize its performance. Hyperparameters include the number of trees in the forest, maximum depth of each tree, and minimum number of samples required to split a node.
* Cross-Validation: Perform cross-validation to assess the model's performance and generalization ability. This involves splitting the training data into multiple folds and evaluating the model on each fold.

EcoVision using Python with the scikit-learn library

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

import pandas as pd

# Load your dataset containing historical pipeline data

# Assuming you have a CSV file named 'pipeline\_data.csv'

pipeline\_data = pd.read\_csv('pipeline\_data.csv')

# Assuming your dataset contains features (X) and labels (y)

# Split the data into features (X) and labels (y)

X = pipeline\_data.drop('label\_column', axis=1) # Drop the label column from features

y = pipeline\_data['label\_column'] # Extract the label column

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize the Random Forest classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the Random Forest classifier

rf\_classifier.fit(X\_train, y\_train)

# Make predictions on the testing set

y\_pred = rf\_classifier.predict(X\_test)

# Calculate accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

In this code:

* Replace 'pipeline\_data.csv' with the path to your dataset containing historical pipeline data. Ensure that your dataset contains features (X) and labels (y), and replace 'label\_column' with the name of your label column.
* Adjust the parameters of the RandomForestClassifier according to your requirements. Here, n\_estimators=100 specifies the number of trees in the forest.
* The train\_test\_split function splits the dataset into training and testing sets. Adjust the test\_size parameter to control the proportion of the dataset used for testing.
* The fit method trains the Random Forest classifier on the training data.
* Finally, the predict method is used to make predictions on the testing set, and accuracy is calculated using the accuracy\_score function.accuracy\_score function.

**Real-time Integration and Deployment:**

* In the Real-time Deployment phase, the focus shifts towards integrating the trained Random Forest model seamlessly into the EcoVision platform to provide timely recommendations for optimizing energy consumption in DevSecOps pipelines. This phase involves configuring the system to leverage the predictive capabilities of the model and ensure its smooth operation within the existing EcoVision environment.

## **Integration with EcoVision Platform**

* Integrating the trained Random Forest model with the EcoVision platform requires careful coordination between data engineers, software developers, and system administrators. The following steps outline the process:
* Model Packaging: The trained model is packaged into a deployable format compatible with the EcoVision platform, such as a Docker container or a serverless function.
* API Development: An API endpoint is created to facilitate communication between EcoVision components and the deployed model. This API endpoint exposes the model's predictive capabilities, allowing EcoVision to query it for energy efficiency recommendations.
* Authentication and Authorization: Security measures, including authentication and authorization mechanisms, are implemented to ensure that only authorized users or components can access the deployed model.
* Scalability Considerations: The deployment architecture is designed to scale dynamically based on fluctuating demand. This may involve deploying the model across multiple instances or utilizing auto-scaling capabilities offered by cloud providers.

## **Monitoring and Performance Optimization**

* Once deployed, continuous monitoring and performance optimization are crucial to ensure the reliability and effectiveness of the deployed model. Key considerations include:
* Performance Metrics: Metrics such as response time, throughput, and error rates are monitored to assess the model's performance in real-world scenarios.
* Alerting and Logging: Automated alerting mechanisms are implemented to notify administrators of any anomalies or performance degradation. Detailed logging is maintained to facilitate troubleshooting and analysis.
* Load Testing: The deployed model undergoes rigorous load testing to simulate peak usage scenarios and identify potential bottlenecks or scalability issues.
* Feedback Loop Integration: Feedback mechanisms are established to gather user feedback and telemetry data from EcoVision components. This feedback is used to iteratively refine the model and improve its predictive accuracy over time.

## **User Interaction and Experience**

* In the Real-time Deployment phase, user interaction with the deployed model is streamlined to ensure a seamless experience. This involves:
* User Interface Integration: The model's recommendations are presented within the EcoVision user interface, providing users with actionable insights to optimize their DevSecOps pipelines.
* Visualization and Reporting: Visualizations and reports are generated to summarize the model's recommendations and highlight areas for energy efficiency improvement. This enables users to track their progress and make informed decisions.
* Feedback Mechanisms: Mechanisms for providing feedback on the model's recommendations are implemented to foster collaboration and continuous improvement. Users can provide input on the relevance and effectiveness of the recommendations, which informs future model iterations.

**Evaluation and Validation:**

* In the final stages of our EcoVision AI Logical Framework implementation, thorough evaluation and validation are paramount to ensuring the reliability and effectiveness of the deployed model. This phase encompasses a series of essential steps aimed at rigorously testing and validating the model's performance against predefined criteria.
* Performance Metrics Selection: Our first task is to carefully select appropriate performance metrics that accurately reflect the model's effectiveness in optimizing energy efficiency within DevSecOps pipelines. Metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC) will serve as key indicators of the model's performance.
* Test Data Preparation: We meticulously prepare a separate test dataset, distinct from the training data, to evaluate the model's performance in real-world scenarios. This dataset is designed to cover a diverse range of input data, ensuring comprehensive testing across various use cases and scenarios.
* Model Evaluation: With the test dataset in hand, we apply the trained EcoVision AI model to make predictions and measure its performance against the selected metrics. By comparing the model's predictions to the ground truth labels, we gain valuable insights into its accuracy, reliability, and overall performance.
* Cross-Validation: To validate the robustness and generalization ability of the model, we employ cross-validation techniques such as k-fold cross-validation. This involves splitting the dataset into multiple subsets, training the model on different combinations of these subsets, and evaluating its performance across various folds to ensure consistent results.
* Error Analysis: A critical aspect of the evaluation process is conducting a detailed analysis of prediction errors to identify common patterns or trends. This analysis helps us pinpoint areas where the model may be struggling and provides valuable insights for further refinement and improvement.
* Validation with Stakeholders: Finally, we validate the model's performance with input and feedback from relevant stakeholders, including developers, domain experts, and end-users. Their perspectives and insights play a crucial role in assessing the model's practical utility and guiding any necessary adjustments or refinements.

By systematically evaluating and validating the EcoVision AI model, we ensure its reliability, effectiveness, and alignment with our objectives of optimizing energy efficiency and enhancing resource utilization within DevSecOps pipelines. This rigorous approach to evaluation and validation underscores our commitment to delivering a robust and dependable solution that meets the needs of our stakeholders and drives tangible benefits across our organization.

**Production Deployment:**In the Production Deployment phase, the validated EcoVision AI model is integrated into the production environment, ensuring seamless operation and scalability. Rigorous testing and monitoring protocols are implemented to maintain reliability and optimize performance in real-world scenarios.

**Conclusion:**

The integration of EcoVision into Kyndryl's DevSecOps pipeline marks a significant milestone in the journey towards sustainable software engineering practices. By leveraging AI-driven solutions, Kyndryl reaffirms its commitment to environmental responsibility while enhancing development efficiency. This integration not only minimizes resource consumption but also reduces the carbon footprint associated with software development processes, demonstrating Kyndryl's dedication to innovation and environmental stewardship.