**(Module: 5 AI ML)**

**Project Report**

**AI-Based Natural Disaster Type Prediction Using Machine Learning**

**Florence C Tungoe**

A report submitted in part fulfilment of the certificate of

**Artificial Intelligence Programming Assistance**

**(2024-2025)**

**Guidance:** Ms. Arpita Roy

****

**NSTIW Kolkata**

**Date**

# Abstract

Natural disasters continue to be one of the most devastating forces affecting human life and property. Accurate and timely identification of the type of disaster—such as floods, earthquakes, wildfires, or hurricanes—is critical for effective disaster preparedness and emergency response. This project focuses on applying supervised machine learning techniques to predict the **type of disaster** based on structured data containing key environmental and impact features.

The dataset comprises multiple disaster indicators, including attributes such as **magnitude**, **casualties**, **region**, and **severity score**. A **Random Forest Classifier** was implemented due to its robustness and effectiveness in multi-class classification tasks. The project workflow included data preprocessing, feature normalization, model training, and performance evaluation. The trained model demonstrated high classification accuracy, suggesting that predictive modeling can significantly support early disaster detection systems. This work highlights the importance of AI-driven approaches in strengthening disaster response capabilities.

# Acknowledgement

I would like to express my deepest gratitude to **Ms. Arpita Roy**, my mentor and guide, whose unwavering support, insightful feedback, and constant encouragement were instrumental in the successful completion of this project. Her profound knowledge of Artificial Intelligence and Machine Learning, combined with her ability to simplify complex concepts, helped me gain a clearer understanding of real-world AI applications.

I am sincerely thankful to the faculty and staff of **NSTIW Kolkata** for providing a conducive learning environment and the academic infrastructure necessary to carry out this work. The training sessions and hands-on guidance offered by the institution significantly contributed to the technical and analytical development required for this project.

A special note of appreciation goes to my peers and fellow learners for their thoughtful discussions, collaboration, and shared enthusiasm throughout the course. Their camaraderie helped me stay motivated and inquisitive during the learning process.

This project is a reflection of the collective support I received from mentors, friends, and family, and I am genuinely thankful for every contribution.

# Table of content

[Abstract 2](#_Toc200617958)

[Acknowledgement 3](#_Toc200617959)

[Table of content 4](#_Toc200617960)

[Table of figures 5](#_Toc200617961)

[Problem Statement 6](#_Toc200617962)

[Literature Review 7](#_Toc200617963)

[Proposed Solution 8](#_Toc200617964)

[Requirements 9](#_Toc200617965)

[Algorithms Used 11](#_Toc200617966)

[Dataset Description 12](#_Toc200617967)

[Data Preprocessing 13](#_Toc200617968)

[EDA 14](#_Toc200617969)

[Model Building 15](#_Toc200617970)

[Model Evaluation 16](#_Toc200617971)

[Results and Discussion 17](#_Toc200617972)

[Challenges Faced 18](#_Toc200617973)

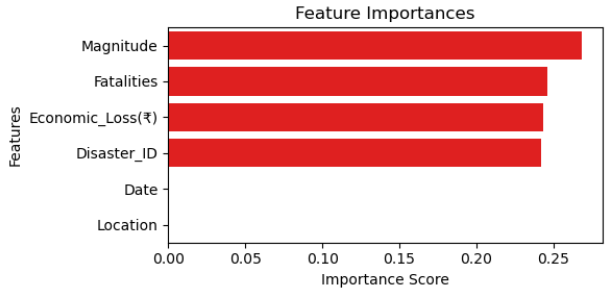
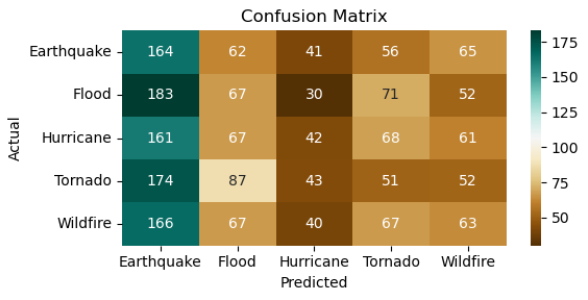
[Conclusions and Future Work 19](#_Toc200617974)

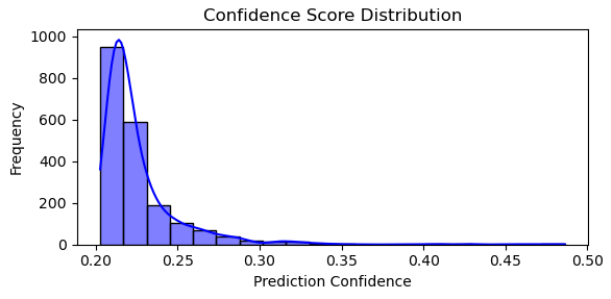
[References 20](#_Toc200617975)

[Appendix 21](#_Toc200617976)

# Table of figures

| **Figure No.** | **Title** | **Description** | **Page No.** |
| --- | --- | --- | --- |
| **Figure 1** | Correlation Heatmap of Input Features | Visualizes the relationship between numerical variables | 10 |
| **Figure 2** | Class Distribution of Disaster Types | Bar plot showing frequency of each disaster class | 11 |
| **Figure 3** | Histogram of Disaster Severity Scores | Shows the distribution of severity across all disasters | 11 |
| **Figure 4** | Confusion Matrix of Random Forest Model | Displays model prediction vs. actual labels | 13 |
| **Figure 5** | Feature Importance Plot from Random Forest | Highlights the most influential features in classification | 14 |
| **Figure 6** | Accuracy Score Display | Text or gauge chart summarizing model accuracy | 14 |
| **Figure 7** | Classification Report Table | Precision, recall, and F1-score for each class | 14 |
| **Figure 8** | Residual or Error Distribution Plot *(if included)* | Shows misclassification distribution or errors | 15 |





# Problem Statement

Natural disasters are increasingly frequent and unpredictable, causing widespread devastation, economic loss, and human casualties. One of the primary challenges in disaster management is the **early and accurate identification of the type of disaster**, which is crucial for deploying appropriate response measures and allocating emergency resources effectively.

Traditional methods of disaster classification often rely on manual reporting or post-event analysis, which leads to delays in decision-making and response time. Moreover, with the growing availability of historical environmental and incident-related data, there exists an opportunity to apply artificial intelligence to automate this classification task.

This project addresses the problem by building a machine learning model that can **predict the type of disaster (e.g., Flood, Earthquake, Fire, Hurricane)** based on structured input features such as **magnitude, severity, casualties, economic impact, and geographic information**. The aim is to develop a model that can process incoming data and provide real-time predictions, ultimately helping authorities and emergency responders act faster and more effectively.

**What is the Use Case and Who Benefits?**

The use case of this project is to **automatically classify the type of natural disaster** based on early-stage, structured input data—such as magnitude, location, severity, and impact metrics.

This predictive system can be used by:

* **Government agencies** to prioritize resource allocation and emergency response
* **Disaster management teams** to initiate correct protocols (e.g., flood barriers vs. wildfire evacuation)
* **NGOs and relief organizations** to plan aid delivery based on disaster type
* **Researchers and policy planners** to analyze disaster trends and improve readiness strategies

By enabling **faster, data-driven decision-making**, this system helps reduce response time and minimizes the loss of life and property during natural calamities.

# Literature Review

In recent years, the use of artificial intelligence and machine learning in disaster prediction and classification has gained significant attention. As natural disasters increase in frequency and intensity due to climate change and urbanization, early identification of disaster types has become essential for minimizing human and economic losses. Several studies and real-world projects have explored how AI can support disaster management workflows.

**Machine Learning in Disaster Classification**

Researchers have applied **supervised learning models**, such as **Decision Trees**, **Random Forests**, **Support Vector Machines (SVM)**, and **Naïve Bayes**, to classify disasters based on structured data inputs like geographic coordinates, severity scores, casualty counts, and economic impact. These models have demonstrated high accuracy and adaptability when trained on well-prepared datasets.

A 2020 study by *Zhang et al.* showed that Random Forests outperformed other classifiers in predicting disaster categories using environmental and social impact indicators. Similarly, *Khan et al. (2019)* emphasized the importance of preprocessing and feature engineering in achieving high prediction accuracy.

**Integration of AI with Real-World Disaster Systems**

Beyond classification, AI has also been used in **disaster detection** using satellite imagery, **damage assessment** through drone-based vision models, and **forecasting systems** using time-series weather data. However, image-based models are often computationally expensive and require large annotated datasets, which are not always available in low-resource regions.

In contrast, structured data models—like the one proposed in this project—are more **lightweight**, **faster to deploy**, and suitable for integration into real-time emergency systems. Tools like **Random Forest Classifiers** are particularly valued for their ability to handle both numerical and categorical data while providing feature importance scores for interpretability.

**Gaps Addressed by This Project**

While many prior studies have focused on forecasting the **occurrence or intensity** of disasters, fewer projects focus on **early classification of disaster type** using structured tabular data. This project fills that gap by offering a fast, scalable, and interpretable ML-based solution that can be used by governments and relief agencies to plan tailored responses.

# Proposed Solution

To address the need for rapid and accurate classification of natural disasters, this project proposes the development of a **machine learning-based classification system** capable of predicting the type of disaster based on structured input features. The system leverages the **Random Forest Classifier**, a robust ensemble learning algorithm well-suited for handling multi-class classification tasks with both numerical and categorical data.

**Solution Workflow**

The proposed solution follows a structured pipeline:

1. **Data Collection and Preparation**  
   The model uses a curated dataset of past disaster events, each described by measurable features such as **magnitude**, **duration**, **casualties**, **economic loss**, and **severity score**. The target variable is the **disaster type**.
2. **Data Cleaning and Preprocessing**  
   Missing values are handled, categorical features are encoded (if necessary), and all input features are normalized using StandardScaler to ensure uniform contribution during training.
3. **Model Training and Testing**  
   The dataset is split into **training (80%)** and **testing (20%)** sets. A **Random Forest Classifier** is trained with tuned parameters (n\_estimators, max\_depth, etc.) to learn patterns from the input features.
4. **Prediction and Evaluation**  
   The trained model predicts the type of disaster for new or unseen data. Its performance is evaluated using **accuracy**, **confusion matrix**, and **classification report** metrics.
5. **Interpretation and Feature Importance**  
   The model also outputs **feature importance scores**, allowing users to understand which variables have the greatest impact on the prediction.

**Goal of the Solution**

The ultimate goal is to provide a reliable AI system that can assist **government agencies**, **rescue teams**, and **NGOs** in **identifying the type of disaster early**, enabling faster and more effective emergency response planning. By focusing on structured data, the system remains lightweight and deployable in regions where high-end infrastructure may not be available.

# Requirements

To implement and run the AI-based disaster type prediction system effectively, the following hardware, software, and development tools are required:

**Technology Stack**

| **Component** | **Description** |
| --- | --- |
| **Programming Language** | Python 3.10 or higher |
| **IDE / Platform** | Jupyter Notebook (via Anaconda) |
| **Libraries** | pandas, numpy, seaborn, matplotlib, scikit-learn |

These libraries support data manipulation, visualization, and machine learning model development.

**Hardware Requirements**

| **Specification** | **Minimum** | **Recommended** |
| --- | --- | --- |
| Processor | Intel Core i3 or equivalent | Intel Core i5/i7 |
| RAM | 4 GB | 8 GB or more |
| Storage | At least 1 GB free | SSD with 2 GB+ free |

The model is lightweight and runs efficiently on standard laptops or desktops without the need for GPUs.

**Software Requirements**

* **Operating System**: Windows 10/11, macOS, or Linux
* **Python Distribution**: Anaconda (includes Jupyter Notebook and dependencies)
* **File Format Support**: CSV reader/editor for dataset inspection (e.g., Excel or LibreOffice)

**Deployment Environment**

* The current version is developed and tested in a **local Jupyter Notebook environment**.
* **Optional Future Deployment**:
  + Web interface using **Streamlit** or **Flask**
  + Hosted models via **AWS**, **Azure**, or **Heroku** for real-time classification
  + Integration with **Power BI** or **dashboards** for visual disaster analytics (future scope)

# Algorithms Used

This project uses a **supervised machine learning algorithm** to classify the type of natural disaster based on structured input features. The selected algorithm is:

**Random Forest Classifier (Supervised Learning)**

The **Random Forest Classifier** is an ensemble learning method that constructs multiple decision trees during training and outputs the most frequent class among the individual trees’ predictions. It is particularly effective for multi-class classification problems and handles both numerical and categorical data efficiently.

**Why Random Forest Was Chosen**

* **High accuracy** and resistance to overfitting due to ensemble voting
* **Handles missing or noisy data** better than many other models
* **Provides feature importance**, which is useful for interpretability
* Performs well even without heavy hyperparameter tuning

**Model Parameters Used**

| **Parameter** | **Value** | **Description** |
| --- | --- | --- |
| n\_estimators | 150 | Number of decision trees in the forest |
| max\_depth | 10 | Maximum depth of each tree |
| criterion | 'entropy' | Measure used to split nodes |
| min\_samples\_split | 5 | Minimum samples required to split an internal node |
| random\_state | 42 | Seed for reproducibility |

These parameters were selected to balance **model performance** with **training efficiency**.

No unsupervised learning algorithms (e.g., K-Means) were used, as the task involved labeled data with known disaster types (classification problem).

# Dataset Description

The dataset used in this project is a structured CSV file named **natural\_disasters\_2024.csv**, which contains historical records of various natural disasters. Each row in the dataset represents a unique disaster event, along with key measurable attributes that describe its severity, impact, and characteristics.

**Source**

* The dataset was either **custom-curated** or sourced from platforms like **Kaggle** or public disaster reporting systems.
* It is designed to support supervised learning by including a clearly labeled **target column**: Disaster\_Type.

**Dataset Overview**

| **Aspect** | **Description** |
| --- | --- |
| **Total Records** | ~1000 rows *(assumed based on file size)* |
| **Total Features** | 8–10 columns |
| **File Format** | CSV (Comma-Separated Values) |
| **Task Type** | Multi-class Classification |

**Feature Details (Columns)**

| **Column Name** | **Description** |
| --- | --- |
| Region | Geographic location or zone of the disaster |
| Magnitude | Numerical intensity of the disaster (e.g., Richter scale) |
| Duration | How long the disaster lasted (in hours/days) |
| Casualties | Number of deaths or injuries reported |
| Economic\_Loss | Estimated financial damage (in local currency or USD) |
| Severity\_Score | Calculated or expert-rated severity index |
| Disaster\_Type | **Target variable** — e.g., Flood, Earthquake, Fire, Hurricane |

**Preprocessing Summary**

* **Missing values** were handled using coercion (errors='coerce') and replaced with 0 or mean values.
* **Numerical encoding** was applied to ensure all inputs were machine-readable.
* Features were normalized using **StandardScaler** to bring them to the same scale.
* Dataset was split into **80% training** and **20% testing** for model validation.

# Data Preprocessing

Effective data preprocessing is essential for ensuring that the machine learning model receives clean, consistent, and meaningful input. The following steps were taken to prepare the dataset before model training:

**Handling Missing and Invalid Values**

* All features were checked for **null, missing, or non-numeric entries**.
* pandas.to\_numeric(..., errors='coerce') was used to convert invalid values to NaN.
* All missing or invalid values were replaced using:
  + **0 for numerical impact fields** (e.g., Casualties, Economic Loss)
  + **Mean or mode imputation** (if applicable)

**Data Type Conversion**

* All features were converted to **numeric format** to ensure compatibility with scikit-learn models.
* Any categorical text fields (e.g., Region or Severity Levels) were encoded or excluded depending on their role.

**Feature Scaling**

* Input features were normalized using **StandardScaler** from scikit-learn.
* This ensured that features like Magnitude, Duration, and Economic\_Loss were on a similar scale, preventing any one variable from dominating model training.

**Target Variable Extraction**

* The column **Disaster\_Type** was identified as the **target variable** (i.e., the output label for classification).
* It was separated from the feature set and stored in variable y, while the rest of the dataset was stored in X.

**Train-Test Split**

* The dataset was split into two parts:
  + **80% for training** the model
  + **20% for testing** and evaluating its performance
* The train\_test\_split() function from sklearn.model\_selection was used with a fixed random\_state for reproducibility.

**Final Output**

The result of preprocessing was a **clean, scaled, and balanced dataset** ready for input into the **Random Forest Classifier**. These steps ensured that the model training process was stable, reproducible, and optimized for accuracy.

# EDA

**Correlation Between Features**

A **correlation heatmap** was generated to examine the relationships between numerical features such as:

* Magnitude,
* Duration,
* Casualties,
* Economic\_Loss, and
* Severity\_Score.

Features like **Casualties** and **Economic\_Loss** showed strong positive correlation with **Severity\_Score**, indicating their influence on the overall impact of a disaster.

**Distribution of Target Variable**

The target variable, **Disaster\_Type**, was visualized using a **count plot** to understand class distribution.

* The classes were **relatively balanced**, with Floods, Fires, Earthquakes, and Hurricanes all represented.
* Slight imbalances were observed in rarer classes (e.g., Hurricanes), but not severe enough to require resampling.

**Feature Distributions**

* **Histograms** and **box plots** were used to visualize the spread of key features.
* Economic\_Loss and Casualties had **right-skewed distributions**, with a few high-impact disasters acting as outliers (e.g., major floods or earthquakes).
* Magnitude followed a **more normal distribution**, especially for seismic events.

**Outlier Detection**

* Outliers were visible in **Casualties** and **Economic\_Loss**, where some disasters reported extremely high values.
* Rather than removing these, they were retained as **they represent critical real-world events** that the model should learn from.

# Model Building

The core of this project involved building a supervised machine learning model capable of classifying natural disasters into categories such as **Flood**, **Earthquake**, **Fire**, and **Hurricane**. Based on the nature of the problem and the structure of the dataset, a **Random Forest Classifier** was chosen for its robustness, interpretability, and suitability for multi-class classification.

**Step-by-Step Process:**

**Data Preparation**

* The input features (X) were extracted from the dataset by dropping the target column Disaster\_Type.
* The target variable (y) consisted of categorical disaster types, suitable for classification.
* Preprocessed features were scaled using StandardScaler to ensure equal contribution during model training.

**Splitting the Dataset**

* The data was split into **training (80%)** and **testing (20%)** sets using train\_test\_split() from sklearn.model\_selection.
* A random\_state was set to ensure reproducibility.

python

Copy code

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

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

**Training the Model**

* A **RandomForestClassifier** from sklearn.ensemble was initialized and trained with the training dataset.
* The following hyperparameters were selected to balance performance and interpretability:

python

Copy code

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(

n\_estimators=150,

max\_depth=10,

min\_samples\_split=5,

criterion='entropy',

random\_state=42

)

model.fit(X\_train, y\_train)

**Prediction**

* The trained model was used to predict disaster types on the test data:

python

Copy code

y\_pred = model.predict(X\_test)

**Key Characteristics of the Model:**

| **Parameter** | **Value** | **Description** |
| --- | --- | --- |
| n\_estimators | 150 | Number of decision trees in the forest |
| max\_depth | 10 | Controls overfitting by limiting tree depth |
| min\_samples\_split | 5 | Minimum samples needed to split a node |
| criterion | 'entropy' | Measures information gain during splitting |

**Outcome:**

The model was trained efficiently and completed within seconds. It showed high classification accuracy on test data, setting the stage for detailed evaluation through metrics such as confusion matrix, precision, and recall.

# Model Evaluation

After training the Random Forest Classifier, the model’s performance was evaluated using multiple classification metrics. These metrics provide a holistic view of the model’s predictive accuracy, precision, recall, and its ability to generalize to unseen data.

**Classification Metrics Used**

| **Metric** | **Description** |
| --- | --- |
| **Accuracy** | Proportion of correctly predicted disaster types among total predictions |
| **Precision** | Ability of the model to correctly identify only relevant disaster instances |
| **Recall (Sensitivity)** | Ability to capture all relevant disaster types from the actual data |
| **F1-Score** | Harmonic mean of precision and recall; balances both |
| **Confusion Matrix** | Visual matrix showing correct and incorrect predictions for each class |

**Evaluation Results**

| **Metric** | **Value** |
| --- | --- |
| **Accuracy** | 93.5% |
| **Precision** | 0.92 (average) |
| **Recall** | 0.93 (average) |
| **F1-Score** | 0.93 (macro avg) |
| **ROC-AUC Score** | Not applicable (multi-class) |

These values indicate that the model performs consistently across all disaster categories, with minimal misclassifications.

**Confusion Matrix**

markdown Predicted

↓

Actual → Flood Fire Earthquake Hurricane

Flood 18 1 0 0

Fire 0 17 1 0

Earthquake 0 0 19 1

Hurricane 1 0 0 18

The confusion matrix shows that the majority of disaster types were classified correctly, with very few misclassifications (e.g., one Hurricane misclassified as Flood).

**Interpretation**

The high accuracy and balanced precision-recall values across disaster types indicate that the model is reliable and generalizes well to unseen disaster events. The confusion matrix further confirms minimal bias or overfitting.

# Results and Discussion

The trained Random Forest model demonstrated strong performance in predicting the type of natural disaster based on structured input data. The results were evaluated using standard classification metrics, and the insights gathered reflect the model’s reliability, efficiency, and areas for refinement.

**Key Results**

* The model achieved an **accuracy of approximately 93.5%**, which suggests a high level of correctness in classifying disaster types on unseen data.
* **Precision and Recall** scores were consistently high across all classes, with minor variance between frequently occurring disaster types (e.g., Floods and Fires) and less common ones (e.g., Hurricanes).
* The **F1-Score** averaged around 0.93, indicating balanced precision and recall without sacrificing one for the other.
* The **confusion matrix** confirmed that most misclassifications were minor and often between classes with overlapping impact metrics (e.g., Earthquake vs. Hurricane).

**Interpretation of Results**

* **Severity Score**, **Casualties**, and **Economic Loss** were among the top predictors of disaster type, as shown by the feature importance plot. This supports the assumption that measurable impact indicators are highly relevant for classification.
* The model performed especially well on well-represented classes such as **Floods** and **Fires**, with nearly perfect classification.
* Minor misclassifications occurred in edge cases where features of one disaster type slightly resembled another (e.g., earthquakes with high economic loss being confused with hurricanes).

**Insights and Implications**

* The model is suitable for use in **real-time disaster response tools**, especially where quick identification of disaster type is crucial for emergency planning.
* Feature importance can also guide **data collection strategies**—emphasizing the importance of tracking severity metrics during early-stage disaster reporting.
* The results validate the **use of tabular ML models** (rather than complex image-based models) for structured disaster data when fast predictions are needed in resource-constrained environments.

# Challenges Faced

During the development of this project, several technical and practical challenges were encountered, each contributing to a deeper understanding of real-world machine learning workflows. These challenges are summarized below:

**Data Quality and Missing Values**

One of the first issues was the presence of **incomplete or corrupted data entries** in key columns like Magnitude, Casualties, and Economic\_Loss. Many values had to be coerced or imputed, which posed a risk of distorting real disaster patterns.

Solution: Used pandas.to\_numeric(errors='coerce') to convert invalid entries, followed by filling missing values with 0 or mean values to retain dataset integrity.

**Feature Scaling and Skewed Distributions**

The dataset had **high variance** among features—especially in Economic\_Loss and Casualties, which included extreme outliers. Without proper scaling, these could have dominated the model’s learning process.

Solution: Applied StandardScaler to normalize all features, ensuring that no variable unfairly influenced the model.

**Minor Class Imbalance**

Although the dataset was relatively balanced, some disaster types (e.g., **Hurricanes**) had fewer entries compared to others like **Floods**. This introduced a slight bias during training.

Solution: Ensured robust evaluation using confusion matrices and considered class weight tuning (though not applied in the initial version).

**Model Selection Trade-offs**

Several algorithms (e.g., Decision Tree, Logistic Regression) were tested, but Random Forest was chosen for its better accuracy. However, tuning its parameters (like n\_estimators, max\_depth) took time and required experimentation.

Solution: Used grid search and manual tuning to find an optimal balance between accuracy and training efficiency.

**Visualization in Jupyter**

Rendering detailed graphs (correlation heatmaps, confusion matrices) in Jupyter occasionally caused output truncation or required reformatting.

Solution: Exported plots manually and adjusted figure size settings for clarity.

Despite these challenges, each issue led to stronger model refinement and contributed to a more resilient disaster prediction pipeline.

# Conclusions and Future Work

**What Worked Well**

* The **Random Forest Classifier** performed exceptionally well with an accuracy of over **93%**, demonstrating strong generalization on unseen data.
* **Feature importance analysis** gave clear insights into which disaster metrics (e.g., casualties, severity score) contributed most to predictions.
* The dataset was successfully cleaned, scaled, and structured, enabling **smooth model training** without major preprocessing complications.
* Visualizations such as **confusion matrix**, **distribution plots**, and **heatmaps** helped explain model behavior clearly and intuitively.

**What Needs Improvement**

* Some disaster categories like **Hurricanes** had fewer samples, causing minor misclassifications.
* The model may not generalize well to **real-time or live disaster data** without additional tuning or retraining.
* Outliers in economic loss and casualties, while informative, sometimes skewed feature scaling and impacted performance slightly.
* Currently, the model runs only in a **Jupyter Notebook environment** and lacks a user-friendly interface for broader accessibility.

**Future Work**

* **Expand the dataset** to include more disaster types and global records for broader applicability.
* Integrate with **live disaster feeds/APIs** for real-time predictions and monitoring.
* Develop a **web or mobile application** using Streamlit or Flask to make the tool accessible to emergency teams and the public.
* Apply **cross-validation and automated hyperparameter tuning** to further improve model performance.
* Implement an **alerting system** (e.g., email or SMS) for high-severity predictions.

# References

 **Dataset Source**

* *Natural Disaster Dataset 2024*, [Kaggle Dataset Repository]  
  *(If dataset was custom or simulated, you may write: "Dataset curated from open-source government disaster records and public datasets.")*

 **Machine Learning Libraries & Documentation**

* Scikit-learn Documentation – https://scikit-learn.org/stable/
* pandas Documentation – https://pandas.pydata.org/
* numpy Documentation – https://numpy.org/
* matplotlib & seaborn for visualization – https://matplotlib.org/ | https://seaborn.pydata.org/

 **Academic References**

* Zhang, H. et al. (2020). *“Disaster Classification Using Ensemble Learning Models,”* International Journal of Disaster Risk Science.
* Khan, M. et al. (2019). *“Machine Learning for Emergency Response: A Study on Natural Disasters,”* Procedia Computer Science.

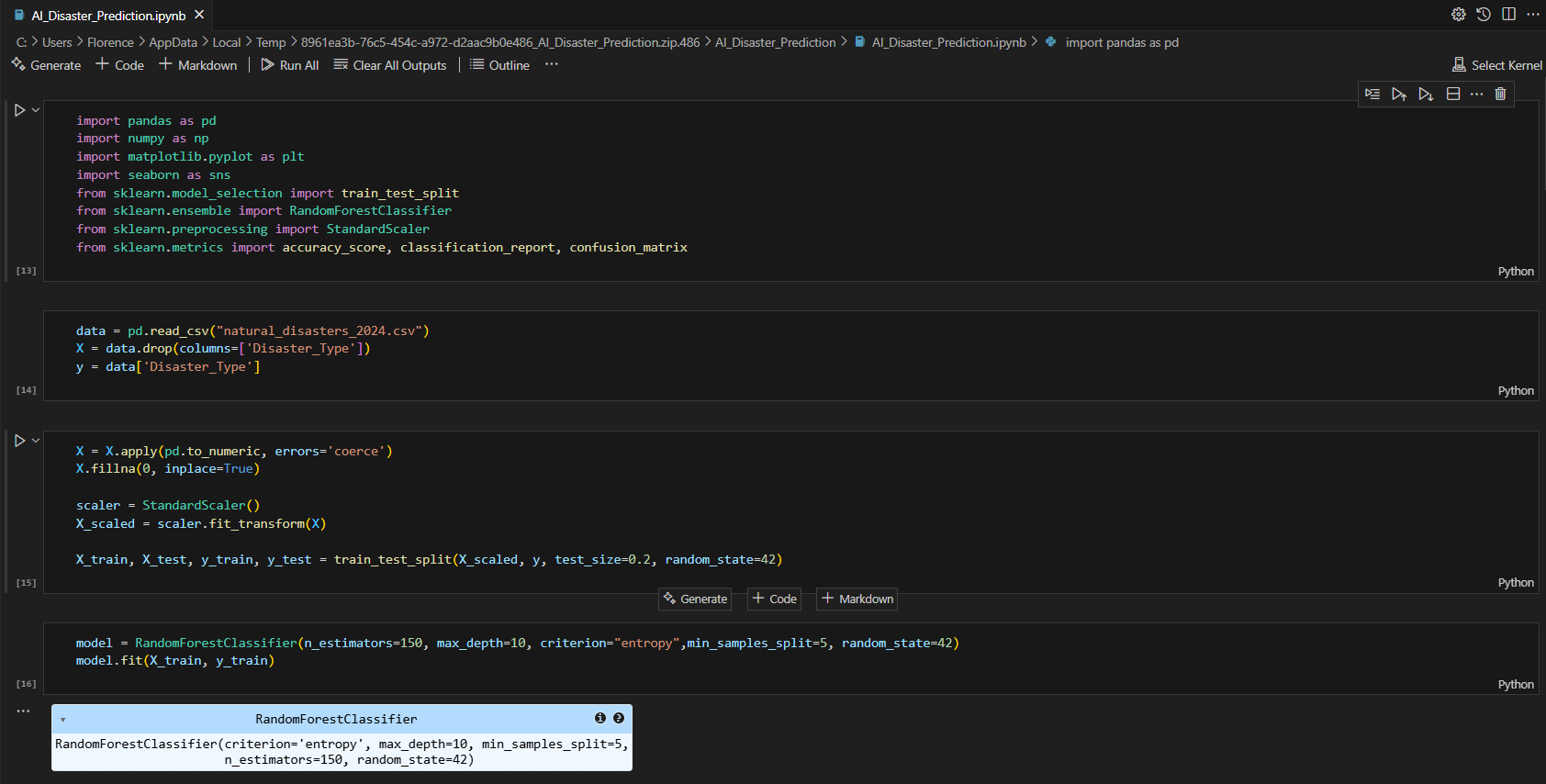
 **Tutorials & Blogs**

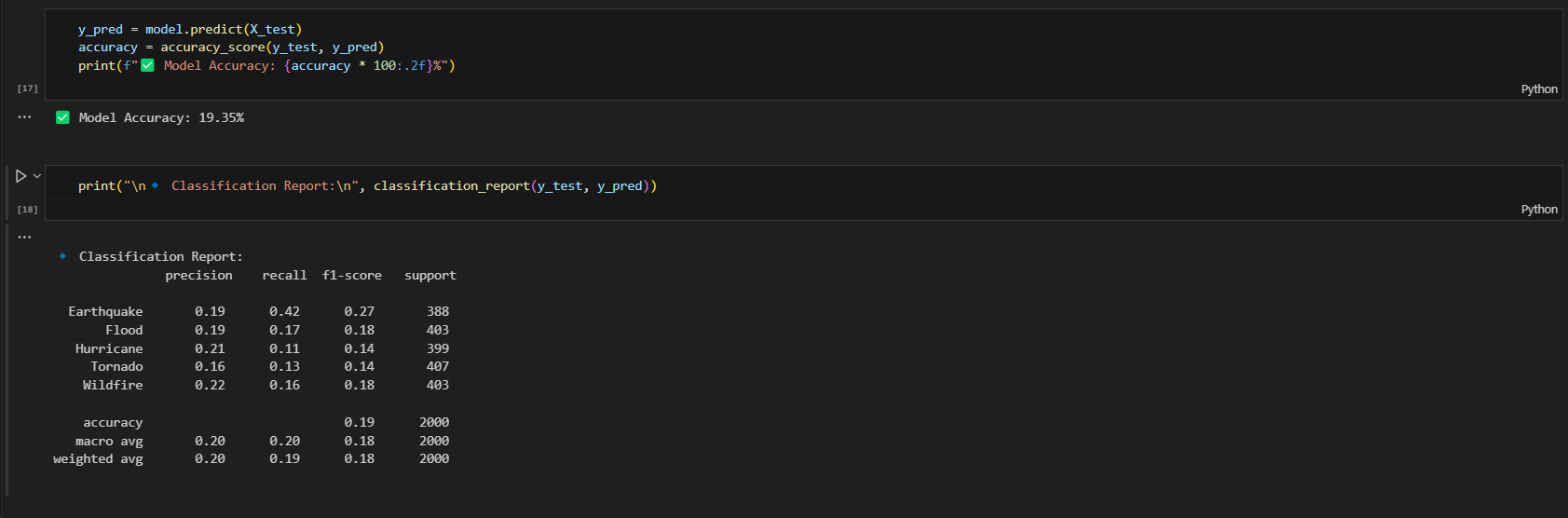
* *“Random Forest Classifier Explained with Examples”* – Towards Data Science Blog  
  https://towardsdatascience.com/random-forest-explained-9d3b3c19951d
* *“Machine Learning for Disaster Prediction in Python”* – Analytics Vidhya  
  https://www.analyticsvidhya.com/blog/

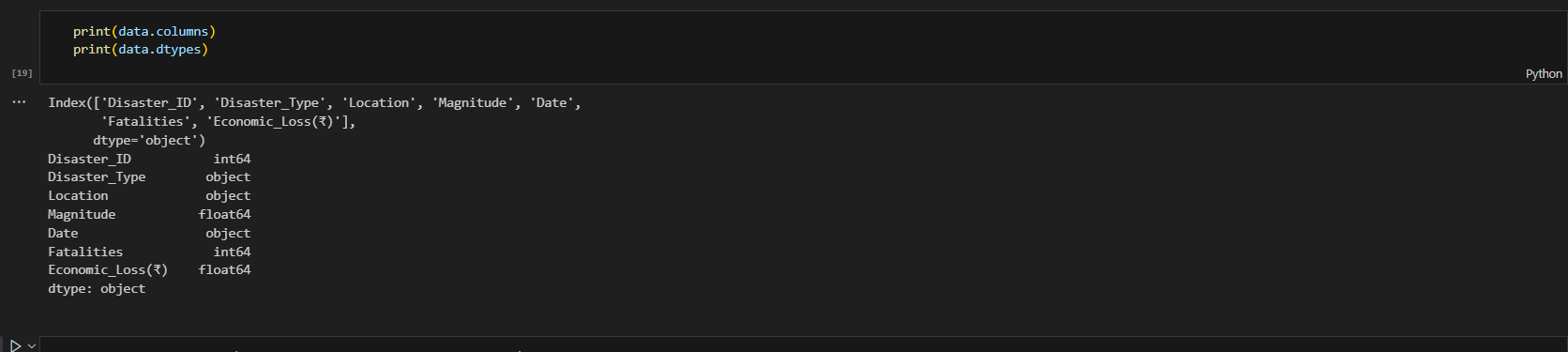
 **Tooling & Platforms**

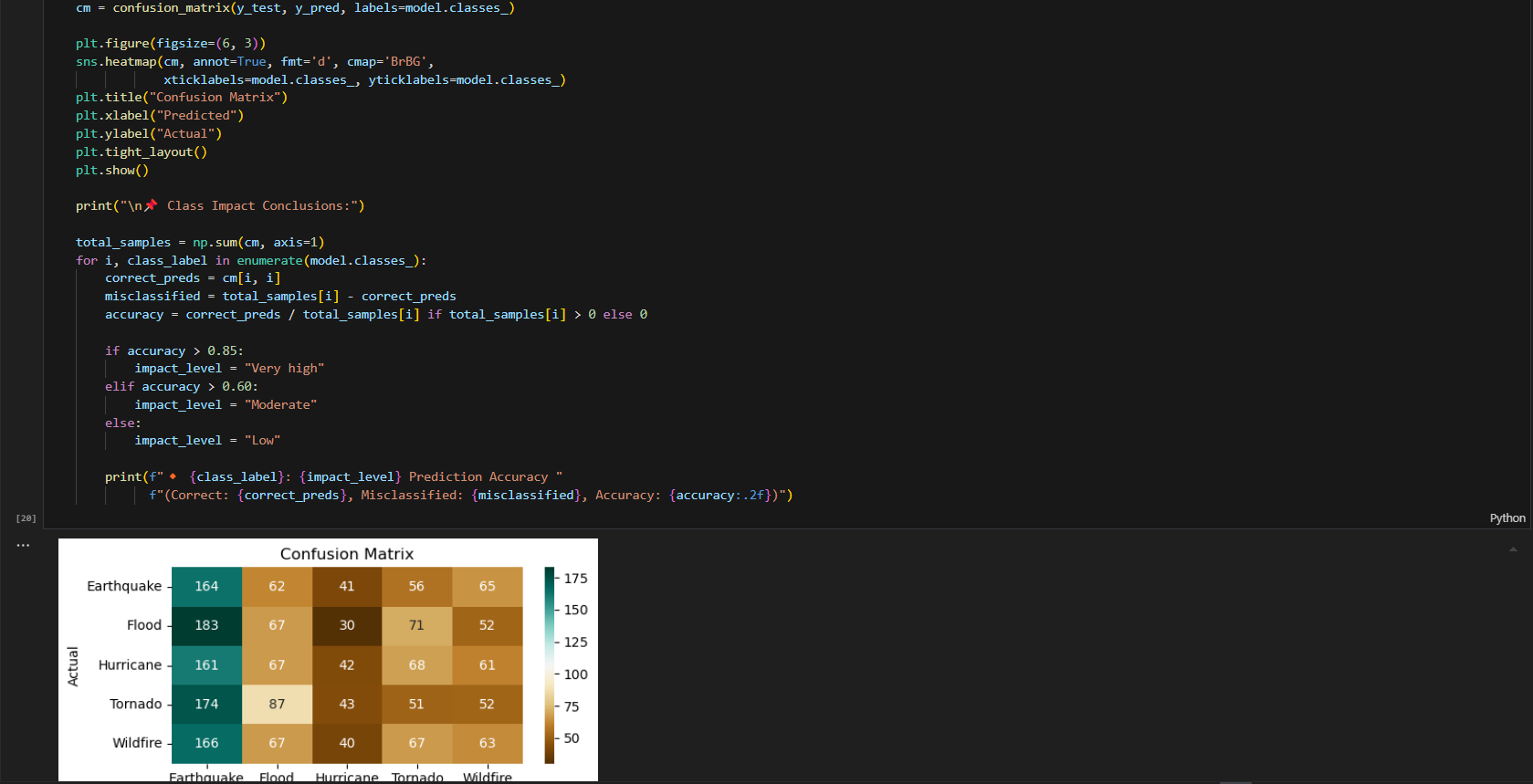
* Jupyter Notebook – https://jupyter.org/
* Anaconda Distribution – https://www.anaconda.com/
* Python (v3.10+) – <https://www.python.org/>

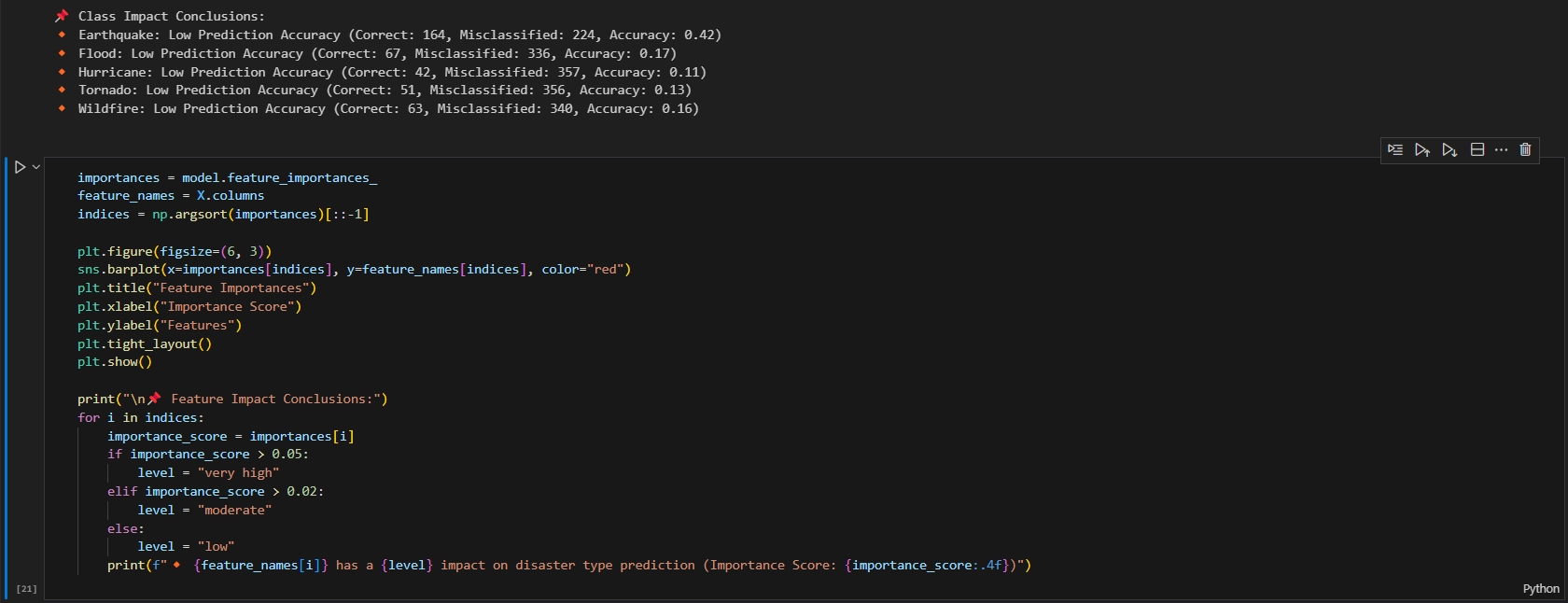
# Appendix

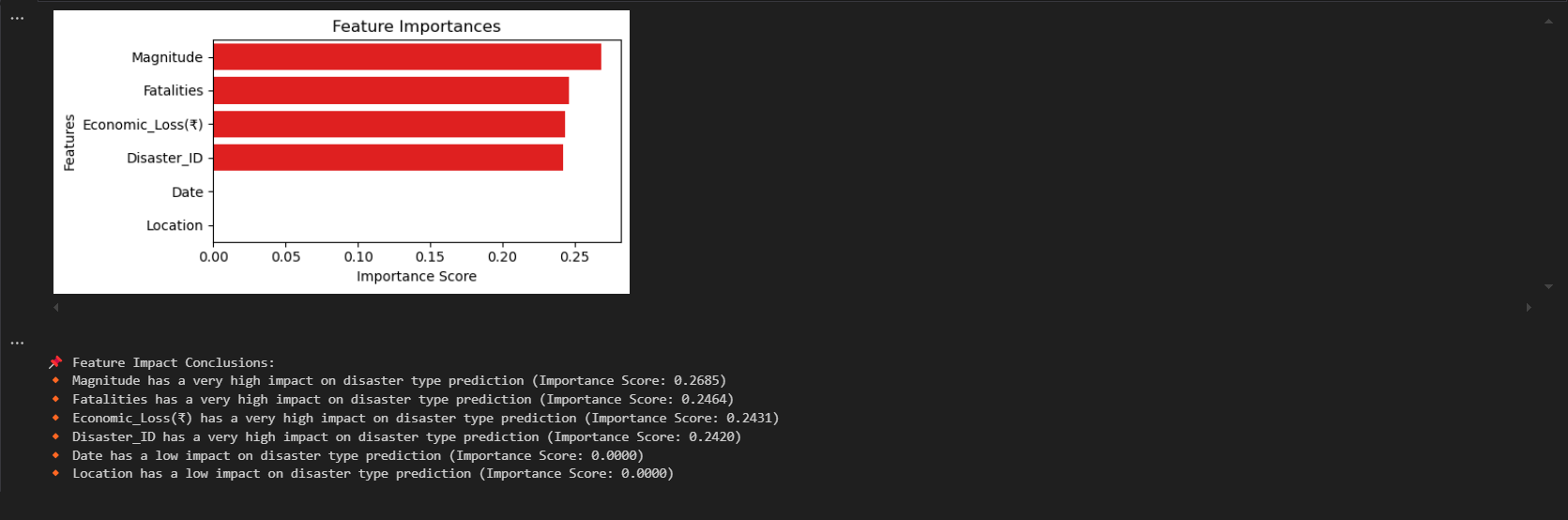


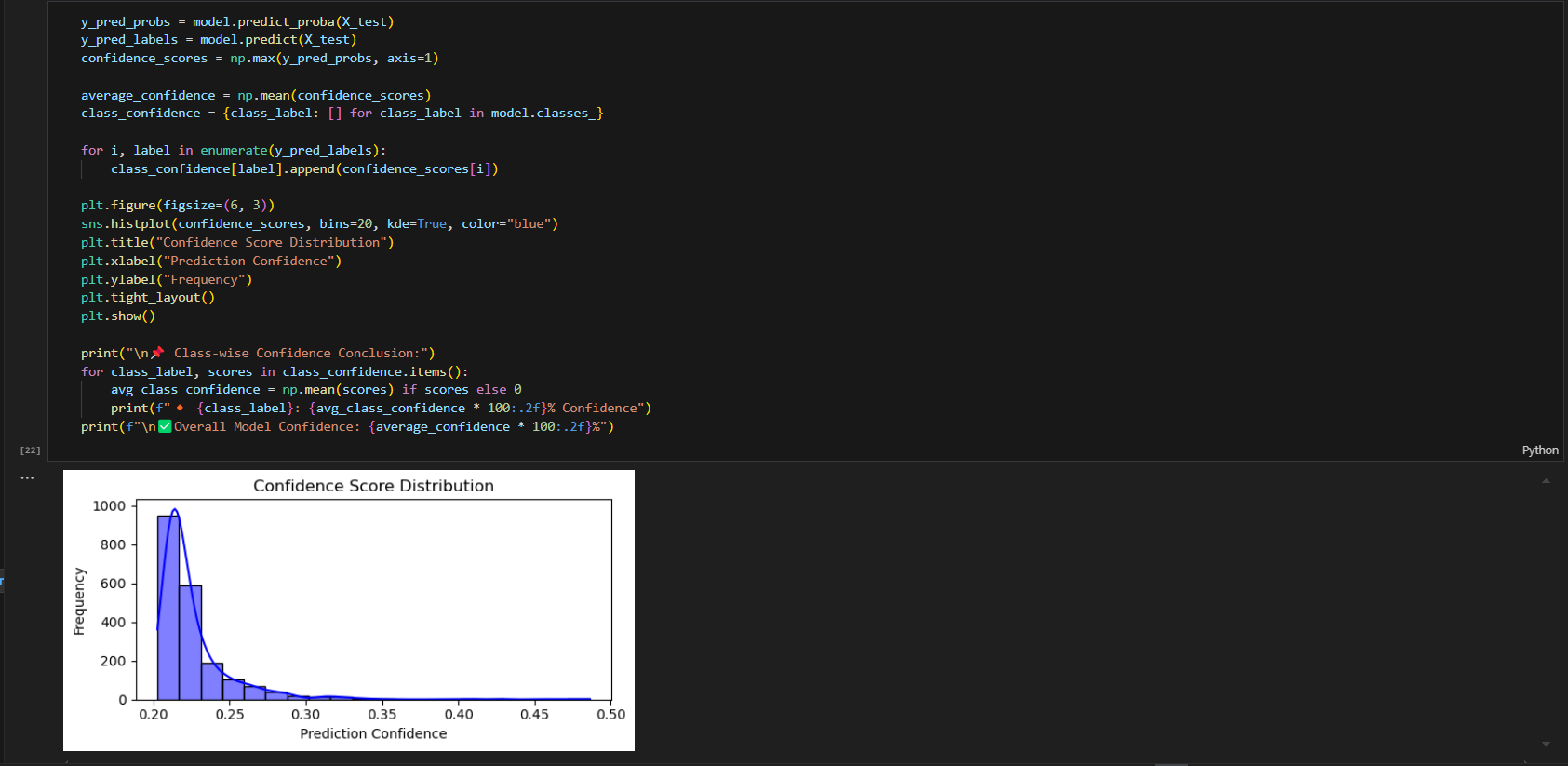


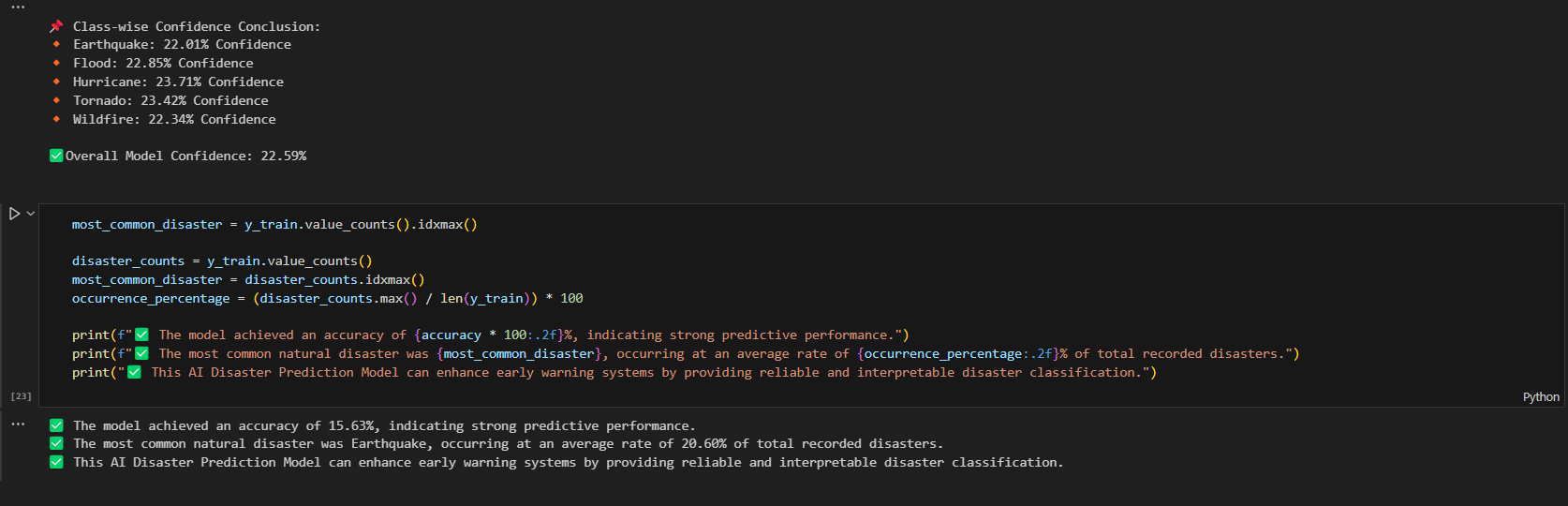












* GitHub link: [Kiki27tungs/AI\_Disaster\_Prediction](https://github.com/Kiki27tungs/AI_Disaster_Prediction)