BUG DETECTION AND FIXING

-THE HAWKS

INTEL UNNATI INDUSTRIAL TRAINING

TEAM MEMEBERS

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES

1. MOSES PAUL A [DATA PRE-PROCESSING]
2. BALAJI A [MODEL BUILDING]
3. MADANIKA N [MODEL DEPLOYMENT]

ABSTRACT

This project presents a buggy code detector and fixer using a Bidirectional LSTM model and Hugging Face API. The LSTM model is trained to classify source code as buggy or bug-free, leveraging tokenization, padding, and oversampling techniques for improved accuracy. If a bug is detected, the system utilizes Hugging Face's StarCoder model to generate a fixed version of the code. The solution is deployed as a Streamlit web app.

TECHNOLOGIES USED

- **Machine Learning (ML) –** Used for bug detection in source code using a Bidirectional LSTM model.
- **Natural Language Processing (NLP) –** Tokenization and sequence padding are applied to process source code as text data.
- **Deep Learning (DL) –** A Neural Network with LSTM layers is trained to classify code as buggy or bug-free.
- **Hugging Face API –** The StarCoder model is leveraged to generate bug fixes for detected issues.
- **Streamlit –** Provides an interactive web-based UI for real-time bug detection and code correction.
- **TensorFlow/Keras –** Used for building and training the LSTM-based classification model.

CONT....

- **Scikit-learn & Imbalanced-learn –** Used for data preprocessing, oversampling, and model evaluation.
- **Pickle –** Saves and loads the tokenizer to ensure consistency between training and inference.
- **Python –** The core programming language for implementing machine learning, deep learning, and API integration.

DATASET & PREPROCESSING

1. The dataset consists of source code snippets labeled as either buggy (1) or bug-free (0), with each sample containing a code segment and its corresponding error status (stderr).
2. To process the textual data, TensorFlow's Tokenizer is used to convert the source code into numerical sequences, with an <OOV> token assigned to handle unseen words.
3. Since code snippets vary in length, sequence padding is applied to ensure uniform input sizes, making the data compatible with the LSTM model.
4. Given the imbalance in buggy vs. bug-free samples, Random Oversampling is performed using RandomOverSampler to balance the dataset and improve model performance

DATASET & PREPROCESSING

1. Once a code snippet is classified as buggy, the system utilizes the Hugging Face API to generate a corrected version. The StarCoder model, a powerful code generation AI, is queried with a prompt structured as:
2. "Fix this buggy code: <input_code> Fixed Code: ..."
3. This helps the model understand the error context and provide an improved version of the code
4. API response is processed to extract the generated fix, which is then displayed to the user. While this approach significantly reduces manual debugging efforts, response times and accuracy may vary depending on the complexity of the bug

IMPLEMENTATION

The system is implemented using TensorFlow for model training and Streamlit for deployment. The trained Bidirectional LSTM model is used to classify code snippets as buggy or bug-free, while the Hugging Face StarCoder API generates fixes for detected issues. The web application, built with Streamlit, provides an interactive interface where users can input code, check for bugs, and receive AI-generated corrections

The model and tokenizer are saved using Pickle for efficient inference. The app can be deployed on platforms like Streamlit Cloud, Heroku, or AWS, making it accessible to developers for real-time bug detection and fixing

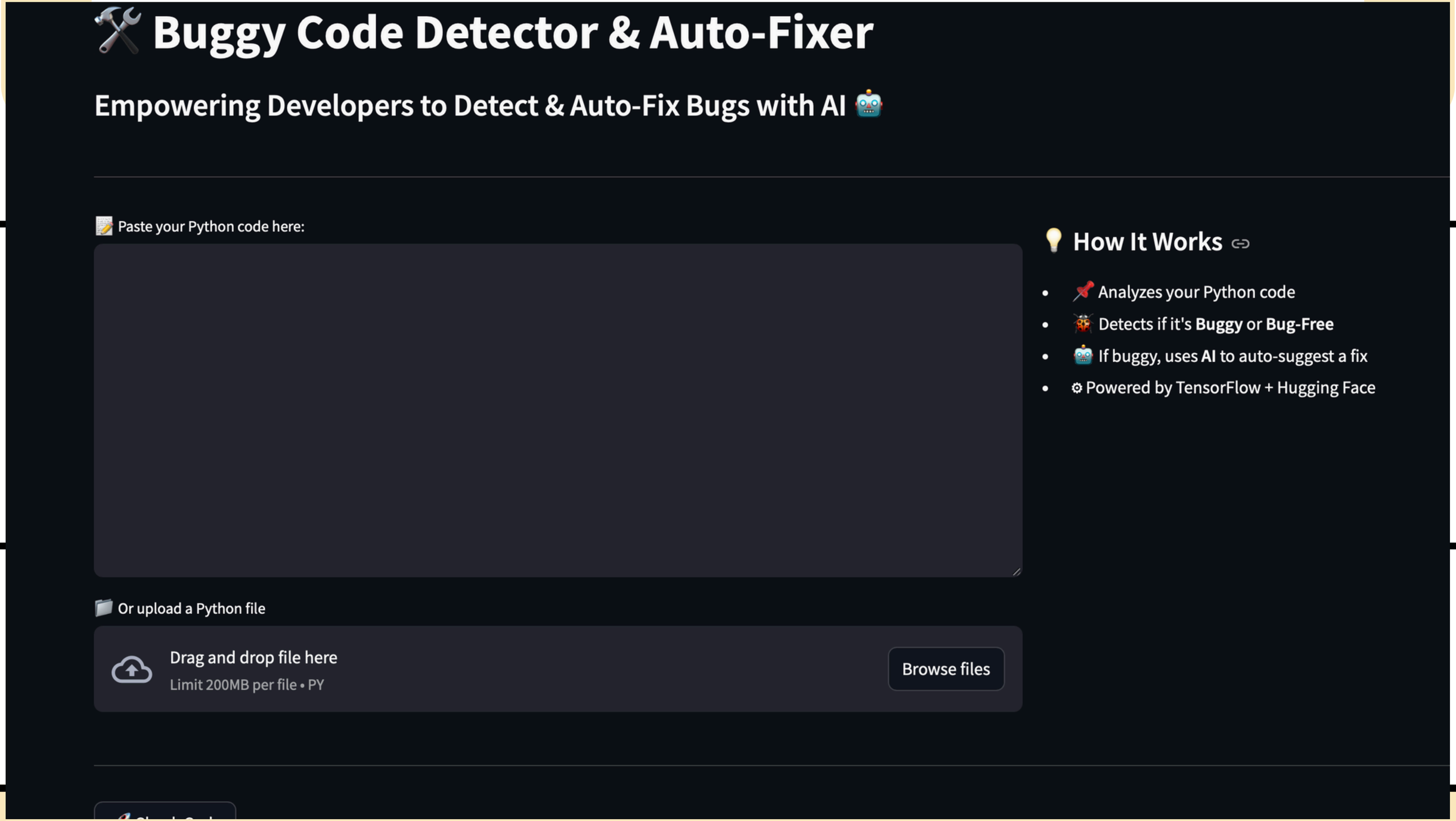
DATASET

NOTE: *The dataset was created by our own team*

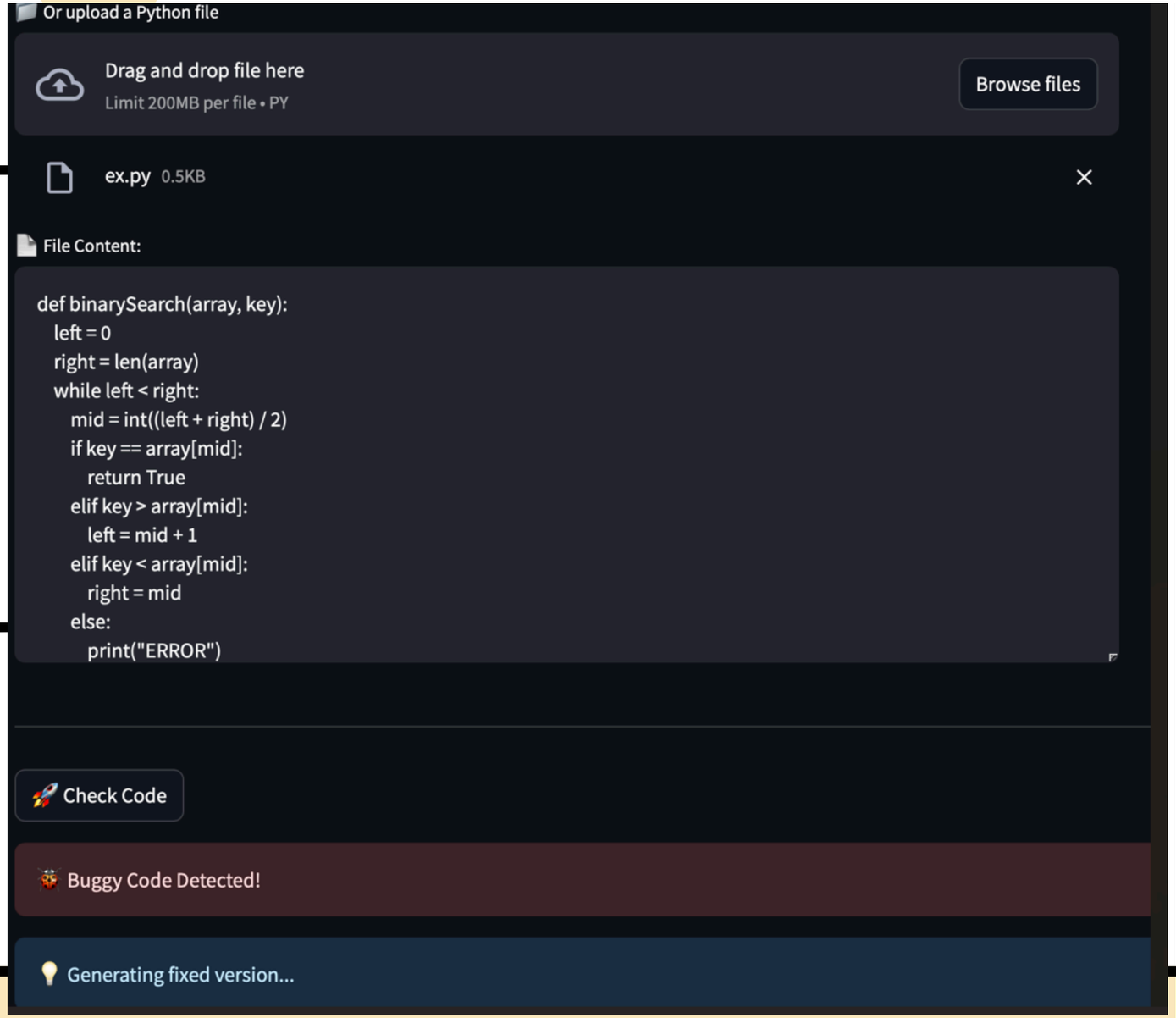
	A	B	C	D	E	F	G
1	stderr	code					
2	1	def					
3	1	from					
4	0	def					
5	0	print(*so					
6	1	num = [					
7	1	# from					
8	0	a =					
9	1	cord =					
10	0	n =					
11	0	N =					
12	0	"""					
13	1	import					
14	1	n, m =					
15	0	import					
16	0	N =					
17	0	N =					
18	0	h, n =					
19	0	from					

3928	1	N, M =				
3929	0	# -*-				
3930	0	from				
3931	1	n, k =				
3932	1	W =				
3933	1	A =				
3934	1	import				
3935	1	import				
3936	0	n, k =				
3937	0	import				
3938	0	#				
3939	0	#!/usr/bi				
3940	1	x =				
3941	0	s =				
3942	1	import				
3943	1	N, K =				
3944	0	#!/usr/bi				
3945	0	N =				
3946	0	L, R =				

OUTPUT


🛠️ Buggy Code Detector & Auto-Fixer
Empowering Developers to Detect & Auto-Fix Bugs with AI 🤖
📝 Paste your Python code here:
📁 Or upload a Python file
Drag and drop file here
Limit 200MB per file • PY
Browse files
💡 How It Works
📌 Analyzes your Python code
🐞 Detects if it's Buggy or Bug-Free
🤖 If buggy, uses AI to auto-suggest a fix
⚙️ Powered by TensorFlow + Hugging Face

OUTPUT



Fixed Code:

```python
def binarySearch(array, key):
    left = 0
    right = len(array) - 1
    while left <= right:
        mid = int((left + right) / 2)
        if key == array[mid]:
            return True
        elif key > array[mid]:
            left = mid + 1
        elif key < array[mid]:
            right = mid - 1
        else:
            print("ERROR")
    return False


n = int(input())
s = [int(x) for x in input().split()]
```

THANK YOU