

HealthCare_ChatBot Documentation

1- Importing Libraries

Data Manipulation and Visualization:

- **numpy:** A fundamental library for numerical computations in Python, providing arrays and mathematical functions.
- **pandas:** A powerful tool for data analysis and manipulation, offering data structures like DataFrames and Series.
- **matplotlib.pyplot:** A plotting library for creating various types of visualizations, including line plots, scatter plots, histograms, and more.
- **seaborn:** A high-level data visualization library built on top of matplotlib, providing attractive and informative plots.
- **missingno:** A library for visualizing missing data patterns in DataFrames.
- **wordcloud:** A library for generating word clouds, which are visual representations of text data where the size of words reflects their frequency.
- **json:** A library for working with JSON (JavaScript Object Notation) data, which is a common format for data exchange.

Text Processing:

- **nltk:** The Natural Language Toolkit, a suite of libraries for natural language processing tasks, including tokenization, stemming, and lemmatization.
- **wordnet:** A lexical database of English words, concepts, and their relationships.
- **stopwords:** A collection of commonly used words that are often removed from text data before analysis.
- **WordNetLemmatizer:** A tool for reducing words to their base form (lemma).

Machine Learning:

- **sklearn.model_selection:** A module for splitting data into training and testing sets, performing cross-validation, and grid search for hyperparameter tuning.
- **sklearn.feature_extraction.text:** A module for converting text data into numerical representations, such as TF-IDF vectors.
- **sklearn.ensemble.RandomForestClassifier:** A machine learning algorithm that combines multiple decision trees for classification.
- **sklearn.svm.LinearSVC:** A support vector machine algorithm for classification.
- **sklearn.metrics.accuracy_score:** A metric for evaluating the accuracy of a classification model.

General Purpose:

- **random:** A library for generating random numbers and performing random sampling.
- **warnings:** A module for handling warnings that may be raised during code execution.

2- Loading Dataset

Loading the **JSON file** that contains a list of intents, each with a **tag, patterns, and responses**. The tag identifies the intent, the patterns are phrases that trigger the intent, and the responses are possible responses to the intent.

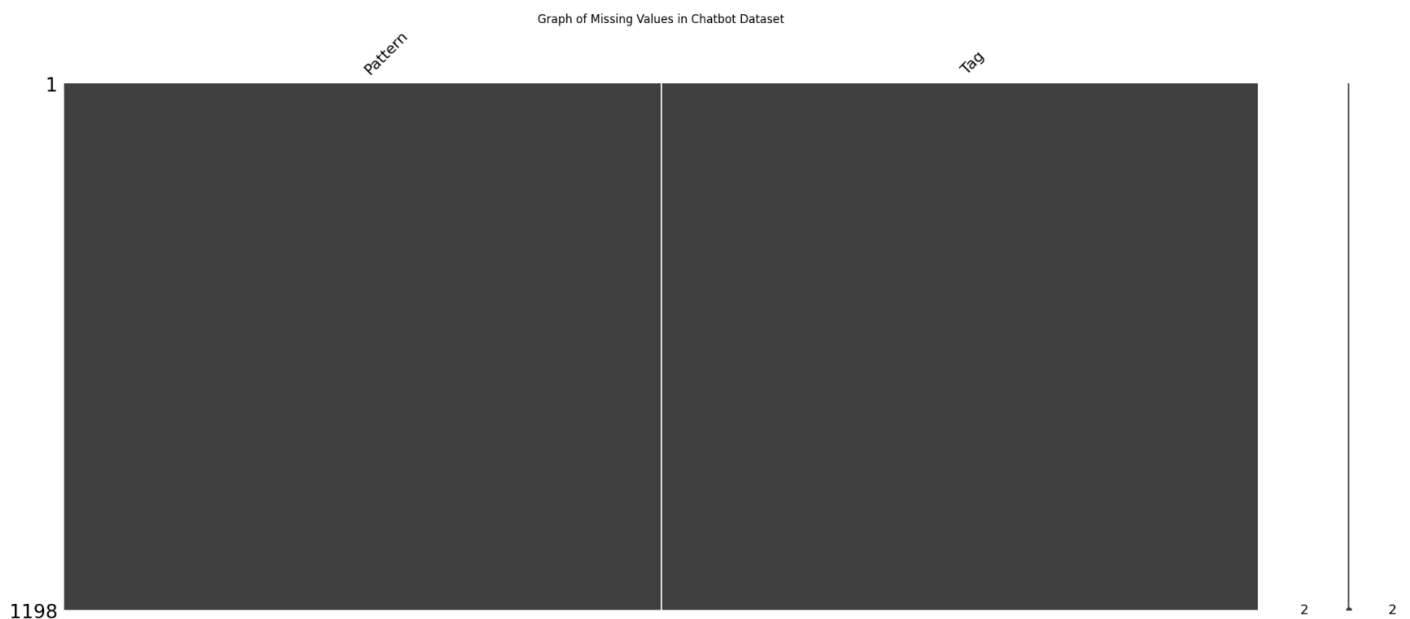
2- Creating DataFrame

- The combined code reads the **JSON** file containing intent data.
- Extracts the **patterns** and **tags** from each intent.
- Creates a pandas DataFrame to store the extracted information.
- Returns the final DataFrame, which contains a list of patterns and their corresponding tags.
- The DataFrame shape is (1198, 2).

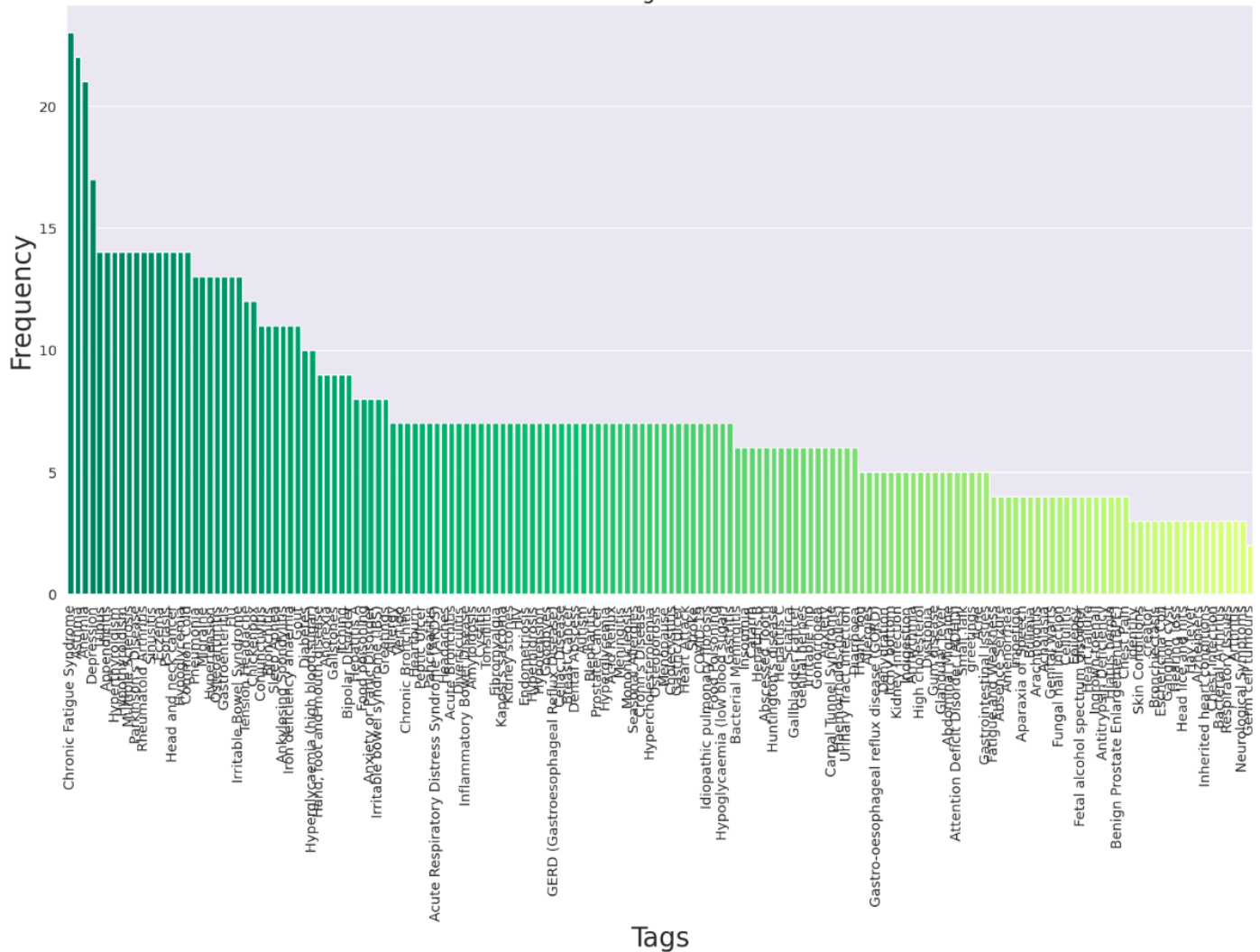
	Pattern	Tag
0	Hello!	Greetings
1	Hi there!	Greetings
2	Good morning!	Greetings
3	Good afternoon!	Greetings
4	Good evening!	Greetings
...
1193	I'm feeling pressure in my lower abdomen, and ...	Respiratory Issues
1194	I've been wheezing, and it feels like I can't ...	Respiratory Issues
1195	One side of my face feels numb, and I've been ...	Neurological Symptoms
1196	My vision is blurry, and my arm feels weak all...	Neurological Symptoms
1197	I feel numbness and tingling in my arms, and I...	Neurological Symptoms

4- Data Exploration and Visualization

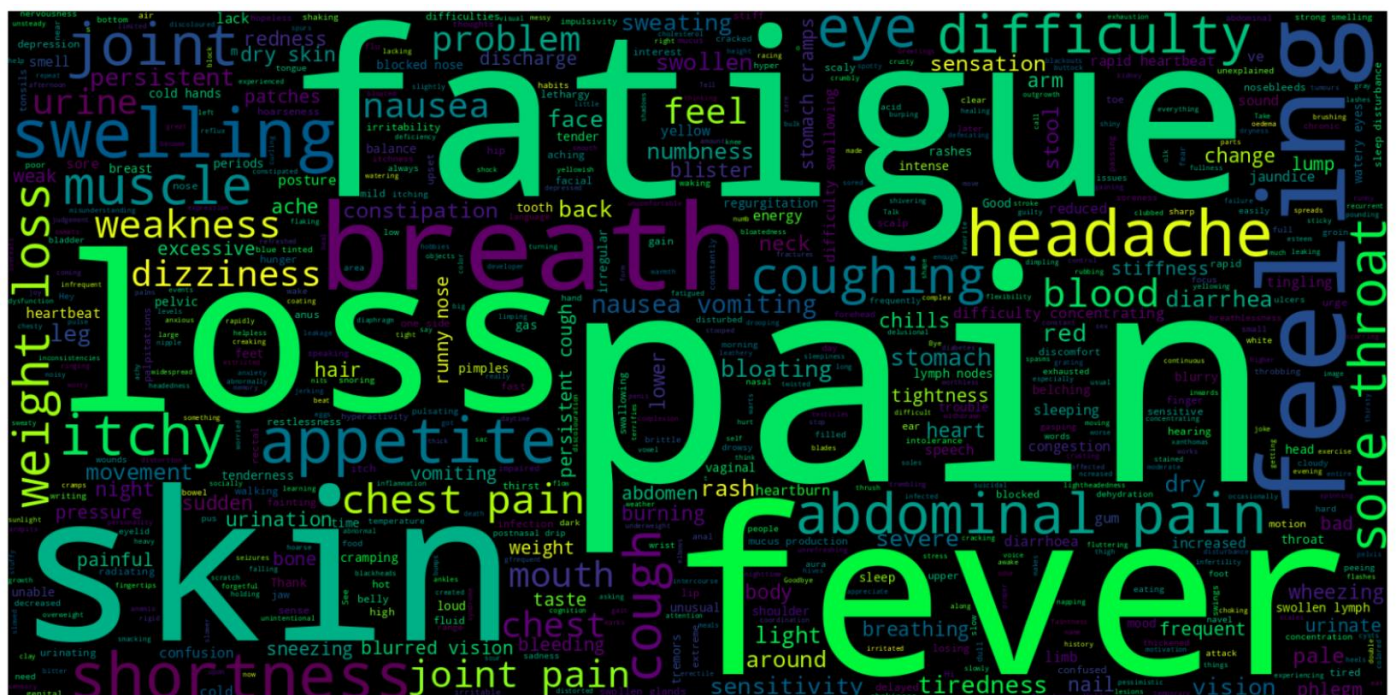
- Creating the **missing value** visualization using **missingno** library.



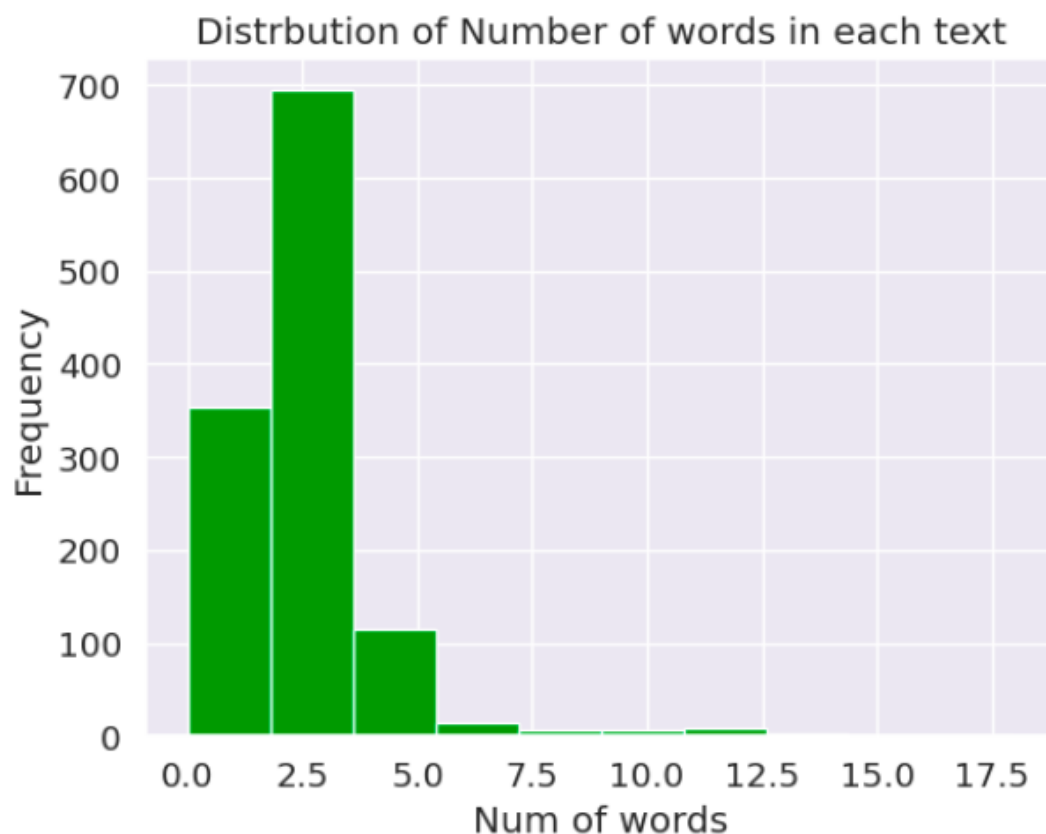
Chatbot Tags Distribution



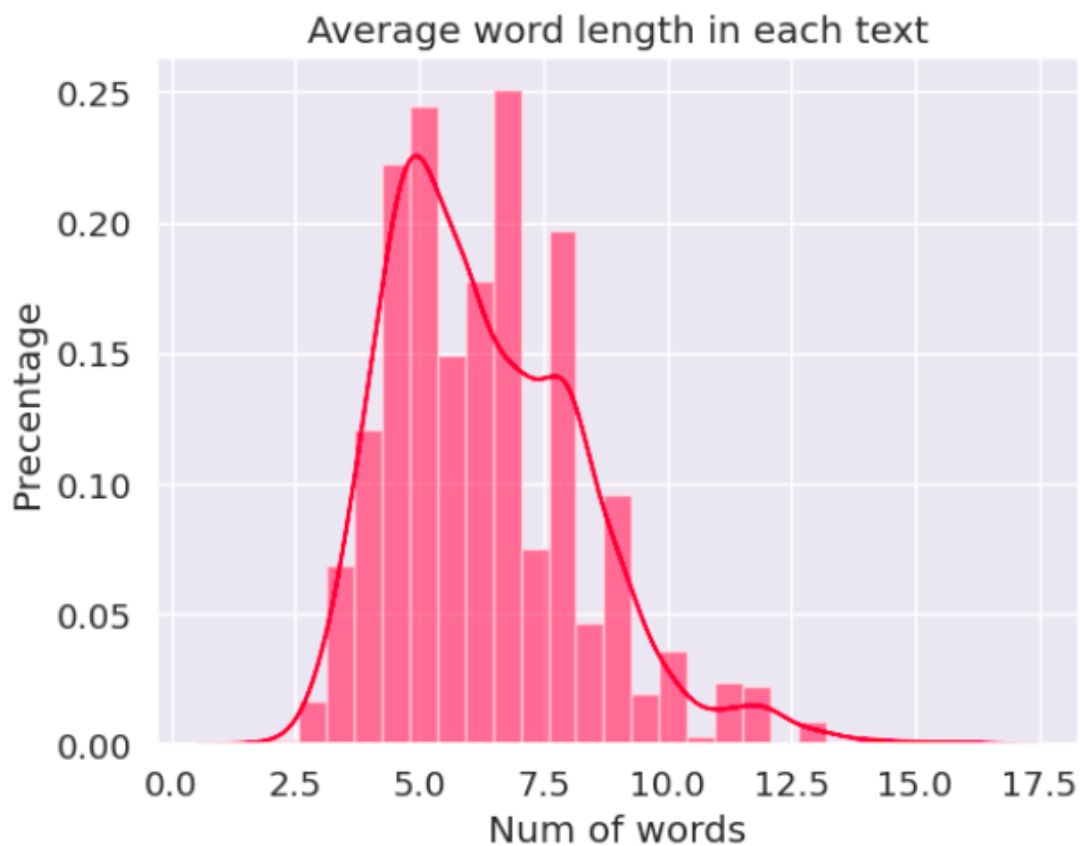
- Effectively generating a **word cloud** using the wordcloud tool that highlights the **most frequent patterns** from your DataFrame's Pattern column.



- Calculating and visualizing the **distribution of words per text** in your DataFrame's **Pattern** column.



- Calculating and visualizing the **average word length for each text** in your DataFrame's **Pattern** column.



7- Performing Synonym Replacement

This function aims to create **new sentences** by replacing some words with their **synonyms**, potentially **increasing the variety** of the data.

Example:

- For example, the **input** sentence is "the patient has a bad headache".
- The **limit** is set to 2, allowing up to 2 words to be replaced.
- The **function** replaces "bad" with "terrible" and "severe", and also replaces "patient" with "sufferer".
- The output is a **list of augmented sentences** with **different variations** of the original sentence.

8- Downloading NLTK Package

Downloads **three essential resources** from the Natural Language Toolkit (NLTK) library:

- **punkt**: This is a **sentence tokenizer** that helps **break down** text into **individual sentences**.
- **wordnet**: This is a lexical database containing **English words, their meanings**, and how they relate to each other. It's useful for tasks like synonym replacement, word sense disambiguation, etc.
- **stopwords**: This is a collection of **common words** in English (e.g., "the", "a", "an") that are often filtered out during text processing as they don't carry much meaning in the context of analysis.

9- Processing Data and Synonym Augmentation for Intent Classification

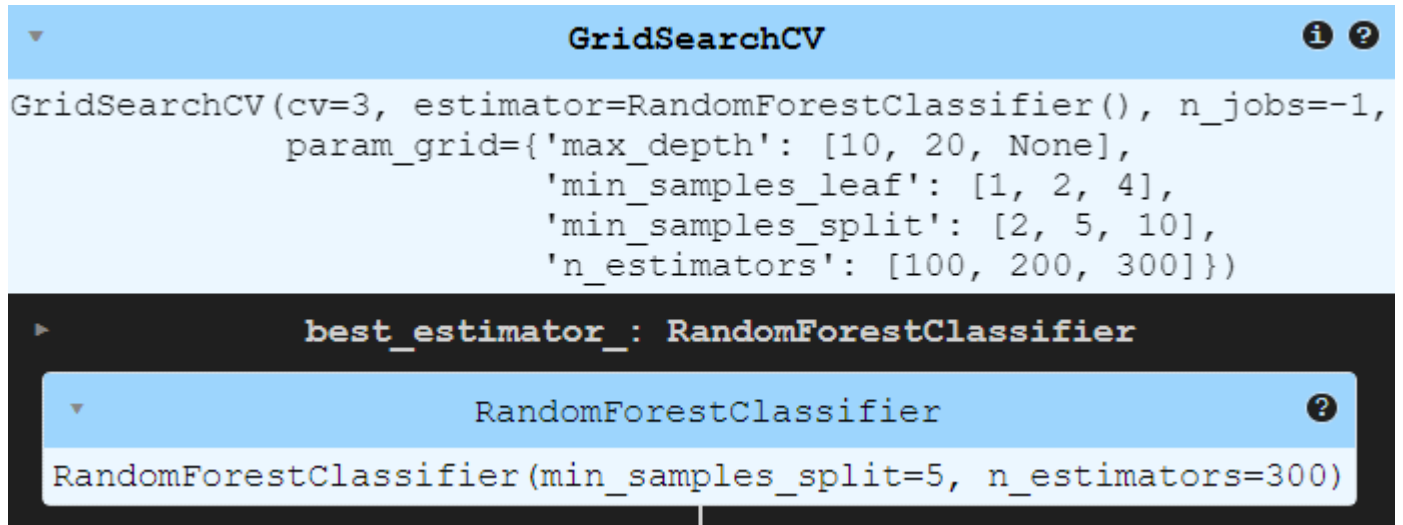
- The code **processes** each intent's patterns, **tokenizes** them, **filters out stopwords** and non-alphabetic characters, **lemmatizes** the tokens, and **appends the processed text to text_data** along with the corresponding label.
- It also **generates** augmented sentences using **the synonym_replacement function**, ensuring the limit per tag is **not exceeded**.
- The final output shows the **number of processed text examples and labels**.

10- Vectorizing Text Data

- The **TfidfVectorizer** converts text data into numerical representations by calculating the term frequency-inverse document frequency (**TF-IDF**) of each term in each document. TF-IDF weights terms based on their importance within a document and across the entire corpus.
- The **fit_transform** method combines the fitting and transformation steps into one operation, learning the vocabulary from the text_data and then transforming the data into the TF-IDF matrix.
- The **X** matrix now contains the **numerical** representation of the text data, ready to be used for machine learning tasks.
- The **y** variable holds the corresponding **labels** for each example, which will be used for supervised learning algorithms.

11- Training and Evaluating the Model

- Training a machine learning model (**Random Forest**) with **hyperparameter tuning** using **GridSearchCV**.
- Training the model to the **full** dataset and getting **74.14% Accuracy**.



```
GridSearchCV(cv=3, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'max_depth': [10, 20, None],
                          'min_samples_leaf': [1, 2, 4],
                          'min_samples_split': [2, 5, 10],
                          'n_estimators': [100, 200, 300]})
```

```
best_estimator_: RandomForestClassifier
```

```
RandomForestClassifier(min_samples_split=5, n_estimators=300)
```

12- Testing Responses

Testing the chatbot by **implementing** a basic **chatbot function** that can **interact** with the user by understanding their intent and **providing** appropriate responses **based on the trained model**.

13- Saving the Model, Vectorizer and Dataset

This will **load** the saved **model**, **vectorizer**, and **intents** data into Python **objects** for **further** use.

- The saved **model** and the saved **vectorizer** will be in the **model** directory.
 - The saved **dataset** will be in the **dataset** directory.
-

14- Model Deployment

- Using **Flask** and a **python environment**, **successfully deployed** the model to work on **local host**.
- **Here is an image of the final result after deployment:**

