Machine Learning Project Documentation

Mariam Kili Bechir

Deployment

1. Overview

Chronic disease prediction system is deployed using streamlit python library. This deployed application allows us to take the required inputs from user and then predict three different types of chronic diseases: heart disease, diabetes disease and Parkinson disease. The types of diseases that needs to be predicted should be selected by the user. After selecting a disease, the application will then make a prediction according to the inputs given and according also to the model selected. The models deployed play the role of the prediction behind the system. If the prediction is one, the output on the application will print a message saying: “The person has a highly chance to be affected by the disease”. If the prediction is zero, the application will then print “The person has a low chance to be affected by the disease.

The final deployment ensures accessibility for users to benefit from accurate chronic disease predictions, aligning with the project's broader goals. The outcomes underscore the effectiveness of machine learning in enhancing healthcare decision-making.

2. Model Serialization

In this project, ‘Joblib’ is used for models’ serialization. Joblib is a python package used to save and load machine learning models. Joblib is especially useful for machine learning models because it allows us to save the state of your computation and resume our work later or on a different machine. Since data is stored as byte strings rather than objects, it may be stored quickly and easily in a smaller amount of space than traditional pickling. Joblib module in Python is especially used to execute tasks parallelly using Pipelines rather than executing them sequentially one after another. Joblib module lets the user use the full potential of their devices by utilizing all the cores present in their device to make the process as fast as possible.

```
import joblib
filename = 'diabetes_model.joblib'
joblib.dump(model2, open(filename, 'wb'))
```

```
# loading the saved model\n",
loaded_model = joblib.load(open('diabetes_model.joblib', 'rb'))
```

Figure 1- Serialization of GradientBoosting model for diabetes disease

In this code, 'diabetes_model.joblib' is the name of the file from which the model will be loaded, and ‘loaded_model’ is the loaded model. Once the model is loaded, we can can use it to make predictions.

```
import joblib
filename = 'heart_model.joblib'
joblib.dump(model1, open(filename, 'wb'))
```

```
# loading the saved model
loaded_model = joblib.load(open('heart_model.joblib', 'rb'))
```

Figure 2-Serialization of SVM model on heart disease

In this code, 'heart_model.joblib' is the name of the file from which the model trained on SVM will be loaded, and ‘loaded_model’ is the loaded model. Once the model is loaded, we can can use it to make predictions.

```
import joblib
filename = 'parkinson_model.joblib'
joblib.dump(model2, open(filename, 'wb'))
```

```
# loading the saved model\n",
loaded_model = joblib.load(open('parkinson_model.joblib', 'rb'))
```

Figure 3- Serialization of Random Forest on Parkinson disease

In this code, 'parkinson_model.joblib' is the name of the file from which the model trained on Random Forest Classiifer will be loaded, and ‘loaded_model’ is the loaded model. Once the model is loaded, we can can use it to make predictions.

3. Model Serving

Streamlit allows us to create web application that can take inputs and return model predictions. In this streamlit application structure, first we loaded the serialized models. The serialized models are the joblib files. These files have stored the trained model's states. As users interact with our Streamlit app, the loaded model is used to make predictions based on the input data.

In our case, on-premises deployment is considered, the requirements and steps for setting up the Streamlit app and serving the model on local servers are:

1. Environment Setup:

A Python Virtual Environment is created for this application using ‘conda’ in order to manage dependencies. The environment name created is called ‘predapp’.

2. Install Dependencies: We installed necessary dependencies such as Streamlit, joblib, and scikit-learn. All neceassaries dependencies are stored in a files called “requirements.txt”.

```
streamlit==1.15.0
joblib==1.0.1
streamlit_option_menu
scikit-learn==0.24.2
```

These are the necessaries dependencies with the versions used to build the streamlit application. We install them using *"pip install -r requirements.txt"*.

3. Streamlit App Creation:

- **App Structure:** We create a file named "multiple_disease_pred.py" for our Streamlit app and we organize it with the joblib files(serialized modesl)
- **Streamlit App Code:** We then used python wrote the necessary script to load the serialized model and create User Interface components for user inputs.

4. Run the Streamlit App Locally:

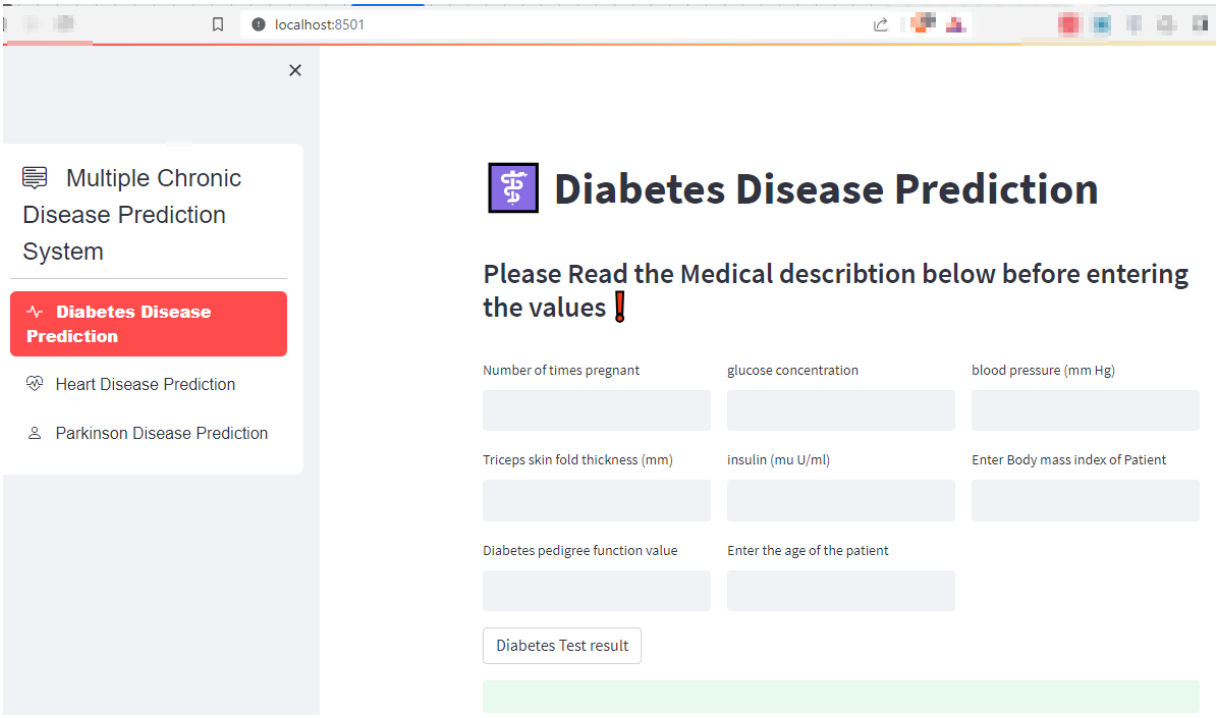
- **Run the App:** Execute the following command in the terminal to run the Streamlit app locally: '*streamlit run multiple_disease_pred.py*'

```
(C:\Users\Mariam\predapp) C:\Users\Mariam\Desktop\UndpTechLeader\Multipledisease\datasets_Notebooks\new_data\Capstoneproject_Chro
nic_Disease>streamlit run multiple_disease_pred.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.113:8501
```

5. Access the App : We just now need to open a web browser and go to http://localhost:8501 to interact with the Streamlit app.



4. API Integration

We used streamlit to deploy instantly the chronic disease prediction application. Streamlit is an open-source python tool for rapidly creating interactive, data-centric web applications. It provides a simple and intuitive API for designing the user interface of our application. We wrote our Streamlit application scripts in Python, and Streamlit converted these scripts into standalone web applications that can be shared and interacted with.

After building our user interface using some API provided by streamlit, we then deployed our apps for free using Streamlit Cloud. While deploying the application, first we need to host our codes, and our models on a github repository. Streamlit works together with this github repository by analyzing the version of every dependency used while building the application. After analyzing our application successfully, we just connect our github repository, the main branch in which all codes are stored, and the Main file path (where streamlit code is written) with the streamlit cloud deploy button to deploy our application. The deployement takes a little bit time because the application will be prepared on the cloud and make sure that all dependencies are compatible each other's.

Our application is now accessible on the web: https://capstoneprojectchronicdisease-4t9vcgbfqw3botzprmyfbt.streamlit.app/

5. Security Considerations

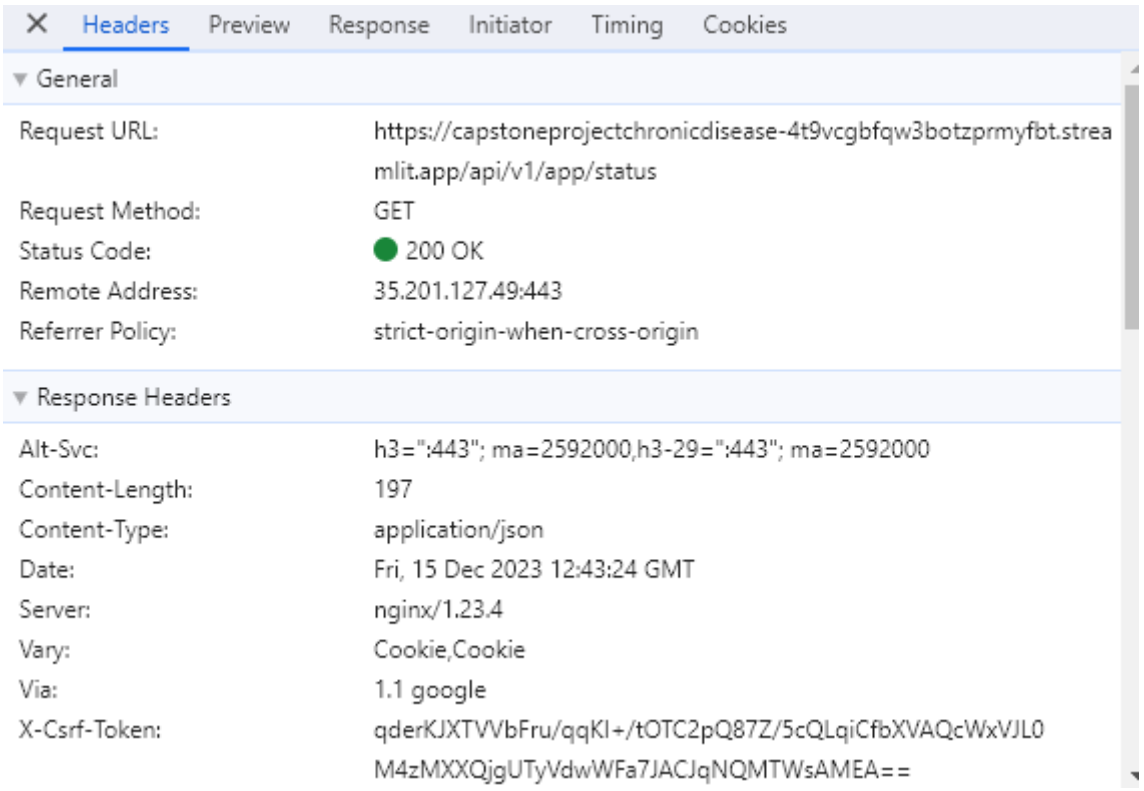
Authentication and Authorization: Streamlit itself doesn't provide built-in authentication or authorization mechanisms. But the streamlit app itself uses session state for user authentication and restrict access to sensitive data based on roles.

Encryption:

When deploying the machine learning application, especially if it involves handling sensitive data, it's crucial to ensure that data transmission is encrypted. Streamlit apps typically run over HTTPS, providing a secure connection between the user's browser and the server. This ensures that data transmitted between the user and the server is encrypted.

6. Monitoring and Logging

Monitoring our deployed model's performance and logging relevant data is crucial for ensuring its reliability and identifying potential issues. Logging in Streamlit apps can be achieved using Python's built-in **logging** module. So, for our application, we insert some logging code in order to log relevant information, warnings, errors, user access etc . Log messages is written to a file named 'app.log'.




```
{
    "isOwner": true,
    "status": 5,
    "creatorId": "2KTS445JOFLEZ34Y4O6BO7HI0P2CVDU33ME2DPU5PLUVE9SQLA",
    "platformStatus": 0,
    "viewerAuthEnabled": false,
    "streamlitVersion": "1.15.0",
    "supportHostingSubdomain": true
}
```