README.md

Medical Cost Prediction

Predicting medical costs of individuals based on different features using several ML (Regression) algorithms. The application was deployed on AWS EC2 through AWS ECR (Dockerized Container).

Dataset

The Medical Cost Prediction consists of around 1300 records and six independent variables along with `charges` target variable:

1. `age` : Age of the individual
2. `children` : Number of children the individual has
3. `bmi` : Body Mass Index of the individual - where bmi <18.5 falls under underweight range, 18.5 - 24.9 falls under normal range, 25.0 - 29.9 falls under overweight range, and >30.0 falls under obese range
4. `sex` : Sex of the individual - Male or Female
5. `smoking` : Whether the individual is a smoker or not
6. `region` : What region the individual belongs to - Northeast, Northwest, Southeast, Southwest

Getting Started (Cloning)

Clone the repo using

```
git clone https://github.com/NvkAnirudh/Medical_Cost_Prediction.git
```

Installing

Install the required packages from `requirements.txt` after commenting out `-e .` which runs `setup.py` automatically.

```
pip install -r requirements.txt
```

Usage

1. Once cloned, run the `data_ingestion.py` script to load, transform, and train different ML algorithms (Regression) on loaded data. This script creates all the required artifacts (train, test, and validation data, model, and preprocessor pickle files).

Model.pkl will have the best model with the best parameters from different models used.

```
python src/components/data_ingestion.py
```

2. After the training, run the test script `test.py` in the Tests folder to get the r2 score from the best model on test data.

```
python Tests/test.py
```

3. Run the `application.py` which is a Flask application to get the required UI locally.

```
python application.py
```

And, that's it, the application should run perfectly locally, and you can test the UI out and play with it.

Deployment using AWS ECR and AWS EC2

1. Create a docker image and check if it is working locally.
2. Add workflows to your GitHub repo. The `.yaml` script can be found in `Actions` section of the repo. Choose the appropriate deployment method to get the script, `Amazon ECS` in this case.

3. Create an IAM User - with `AmazonEC2ContainerRegistryFullAccess` and `AmazonEC2FullAccess` policies attached - and create access keys

4. Create a new ECR repository

5. Create an EC2 instance with enough disk space

6. Once created, connect to the instance and install the necessary packages for docker:

   ```
   sudo apt-get update -y

   sudo apt-get upgrade

   curl -fsSL https://get.docker.com -o get-docker.sh

   sudo sh get-docker.sh

   sudo usermod -aG docker ubuntu

   newgrp docker
   ```

7. Initiate Runner setup (Github Repo -> Settings -> Actions (on the LHS) -> Under Actions, click Runners

8. Download and Configure the runner on the EC2 instance using the commands mentioned when you create a new runner (in the path mentioned above).

9. Create 5 secret keys in GitHub Actions:

   - `AWS_ACCESS_KEY_ID` : Access key received when you create access keys with your IAM User

   - `AWS_SECRET_ACCESS_KEY` : Secret Access key received when you create access keys with your IAM User

   - `AWS_REGION` : Region of your EC2 instance

   - `AWS_ECR_LOGIN_URI` : ECR repository's login URI

   - `ECR_REPOSITORY_NAME` : Name of the ECR repository created

Authors

- Anirudh Nuti - *Initial Work* - NVK Anirudh

License

Acknowledgements

Thanks to Krish Naik for his informational videos