Data Glacier

Deployment on Cloud:

California House Pricing Dataset

Name: Zohra Bouchamaoui

Batch Code: LISUM21

Submission date: 03/06/2023

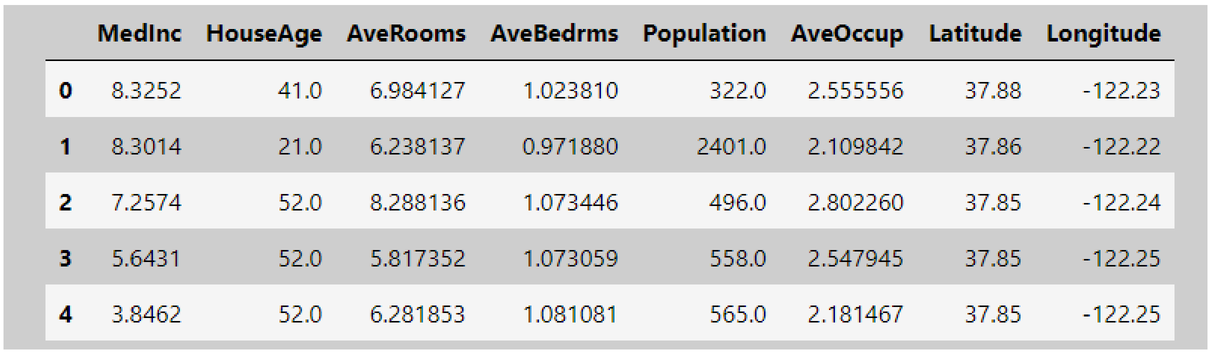
Submitted to: Data Glacier

1. Introduction:

The California housing dataset is a popular dataset widely used in machine learning and housing market analysis. It contains information about houses in various districts of California, with features such as the number of rooms, population, median income, and median house price. In this task, our goal is to build a regression model that can predict the median house price based on the given features. Accurate predictions of house prices are crucial for home buyers, sellers, and real estate agents, as it helps in making informed decisions and understanding the market trends. In addition to predicting the house prices using a regression model, we will be presenting in this document the process of deploying the app using Flask and then on the web using Heroku and AWS EC2.

2. Data Overview:

The California housing dataset (Fig. 1) consists of a set of 8 numeric, predictive features and a target variable. The target variable is the median house price for California districts (expressed in hundreds of thousands of dollars ($100,000). The input features include the median house age, average number of rooms per household, average number of bedrooms per household, average number of household members, population, median income, latitude, and longitude. Each row in the dataset represents a district in California. The size of the dataset is of 20,640 instances and there are no missing attribute values.

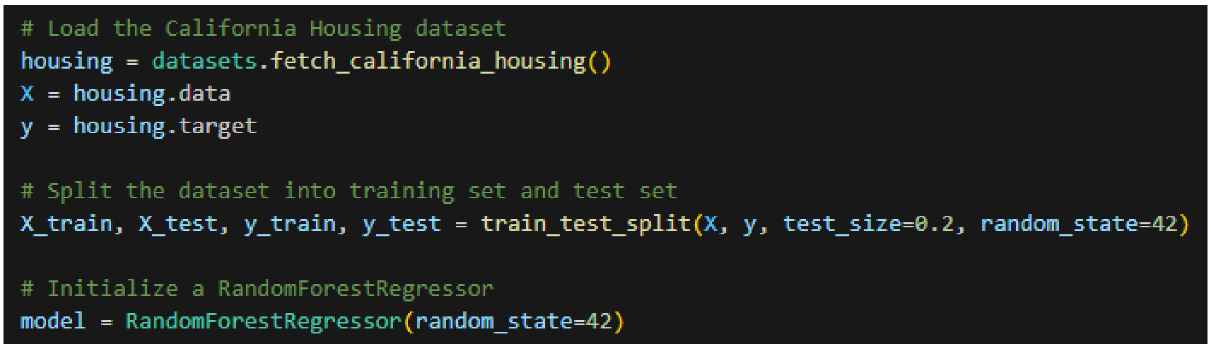


	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

Fig. 1: California housing data head overview

3. Building a Model:

In this project, we used the RandomForestRegressor from the scikit-learn library to build our regression model for predicting house prices. The RandomForestRegressor is an ensemble model that combines multiple decision trees to make accurate predictions. A random_state parameter is set to 42 for reproducibility. (Fig. 2)

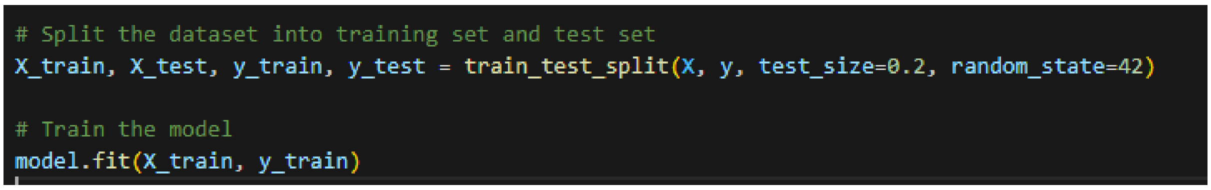


```
# Load the California Housing dataset
housing = datasets.fetch_california_housing()
X = housing.data
y = housing.target

# Split the dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize a RandomForestRegressor
model = RandomForestRegressor(random_state=42)
```

Fig 2: Model Building

4. Training Procedure:

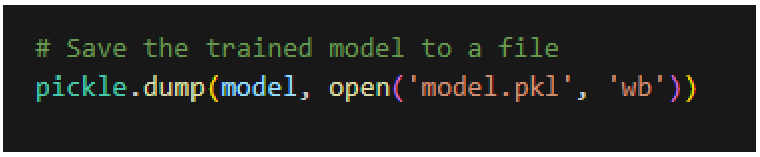
To train our model, we split the dataset into a training set and a testing set using the train_test_split function from scikit-learn (Fig. 3). We use 80% of the data for training and 20% for testing.



```
# Split the dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
model.fit(X_train, y_train)
```

Fig. 3: Model training

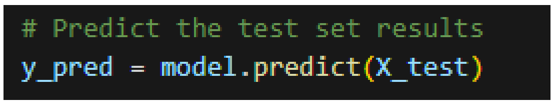
After training the model, we can save it to a file (Fig. 4) using the pickle module for future use or deployment:



```
# Save the trained model to a file
pickle.dump(model, open('model.pkl', 'wb'))
```

Fig. 4: Save the trained model

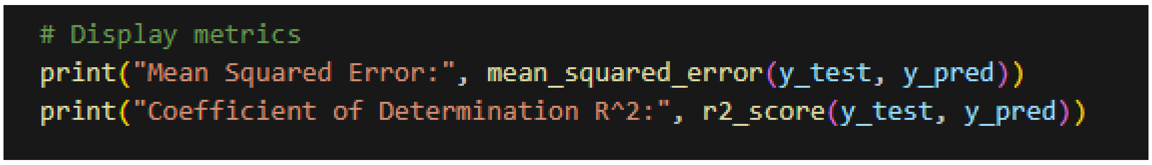
Next, the trained model is used to make predictions on the test set (Fig. 5):



```
# Predict the test set results
y_pred = model.predict(X_test)
```

Fig. 5: Trained model predictions

Finally, the performance of our model is evaluated by calculating the mean squared error (MSE) and the coefficient of determination (R^2) using the mean_squared_error and r2_score function from scikit-learn (Fig. 6):



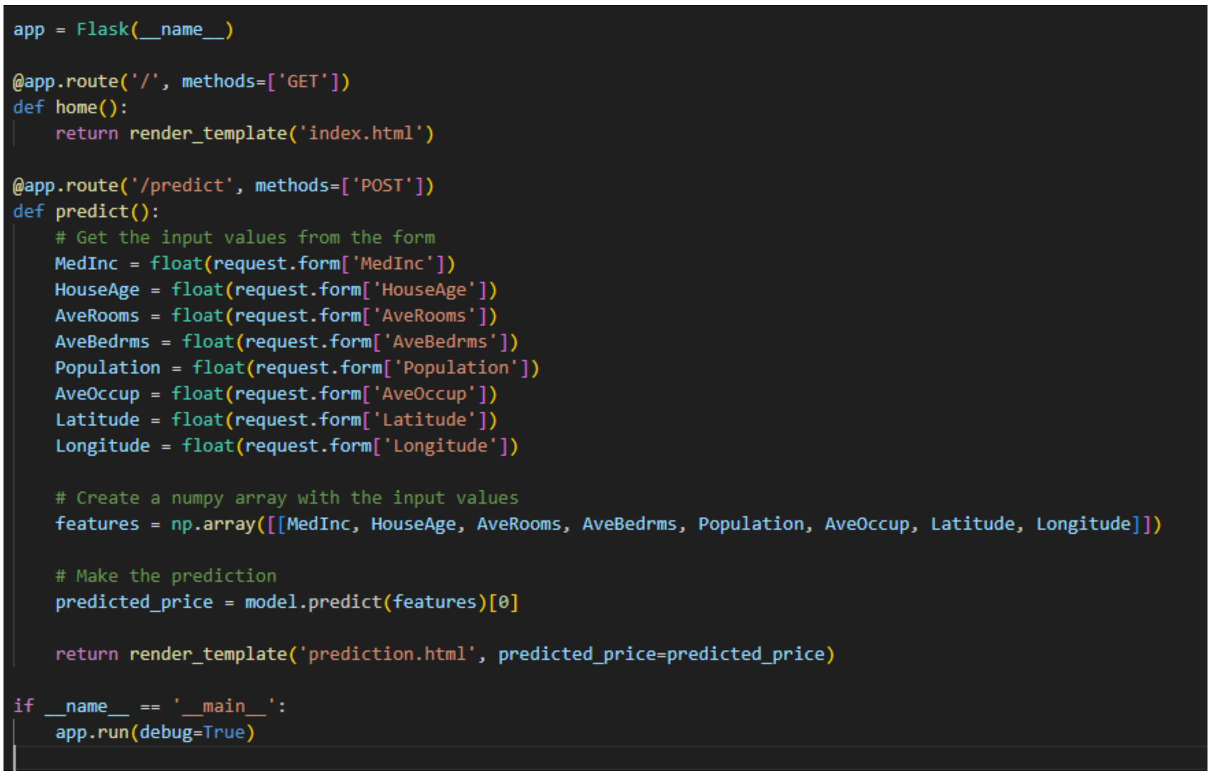
```python
# Display metrics
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("Coefficient of Determination R^2:", r2_score(y_test, y_pred))
```

Fig. 6: Performance metrics

The mean squared error measures the average squared difference between the predicted house prices and the actual prices using the test set. Therefore, a lower MSE value indicates a better performance. On the other hand, the coefficient of determination (R^2) represents the proportion of variance in the target variable that is explained by the model. It ranges from 0 to 1, with 1 indicating a perfect fit. This means that the higher the R^2 value is, the better the fit of the model to the data is.

5. Flask Deployment:

To deploy the California house price prediction model using Flask, an API, which takes input from the user and returns the predicted house price is created. The Flask web framework is used to handle the HTTP requests and render the HTML templates.



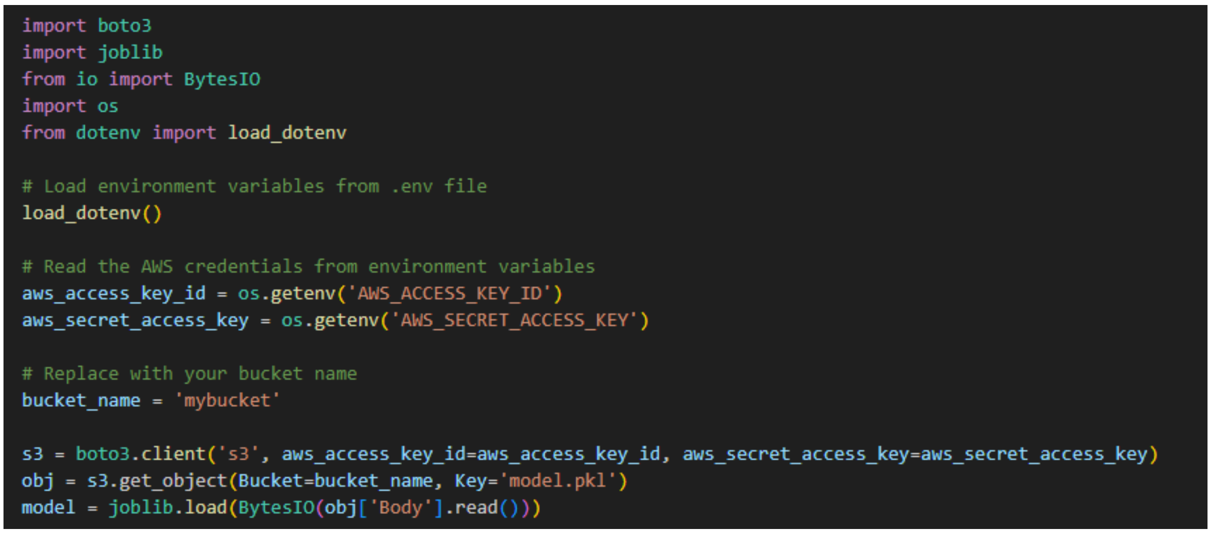
```python
app = Flask(__name__)

@app.route('/', methods=['GET'])
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    # Get the input values from the form
    MedInc = float(request.form['MedInc'])
    HouseAge = float(request.form['HouseAge'])
    AveRooms = float(request.form['AveRooms'])
    AveBedrms = float(request.form['AveBedrms'])
    Population = float(request.form['Population'])
    AveOccup = float(request.form['AveOccup'])
    Latitude = float(request.form['Latitude'])
    Longitude = float(request.form['Longitude'])

    # Create a numpy array with the input values
    features = np.array([[MedInc, HouseAge, AveRooms, AveBedrms, Population, AveOccup, Latitude, Longitude]])

    # Make the prediction
    predicted_price = model.predict(features)[0]

    return render_template('prediction.html', predicted_price=predicted_price)

if __name__ == '__main__':
    app.run(debug=True)
```

Fig. 7: Flask Deployment

In the code above, we create a Flask application and define two routes. The '/' route which renders he HTML 'index.html' template which will display a form for the users to enter input values. The '/predict' route handles the form submission, retrieves the input values, makes a prediction using the loaded model, and renders the 'prediction.html' template which displays the predicted house price. (Fig. 7)

6. Heroku Deployment:

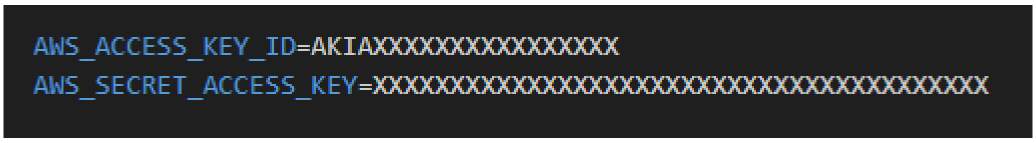
Since the saved model, model.pkl, is 125,645 kb (approx. 123 MB) it means that we will have some trouble uploading it to GitHub (which has a maximum file size limit of 100 MB for individual files). To be able to deploy the saved model to Heroku (Fig. 8), we used a cloud storage approach. By uploading our model.pkl to a cloud storage service like Amazon S3 and modifying our app.py file to download the model file from cloud storage when it starts up. This way, the model file is not included in the Git repository and our app can still access it when it runs.



```python
import boto3
import joblib
from io import BytesIO
import os
from dotenv import load_dotenv

# Load environment variables from .env file
load_dotenv()

# Read the AWS credentials from environment variables
aws_access_key_id = os.getenv('AWS_ACCESS_KEY_ID')
aws_secret_access_key = os.getenv('AWS_SECRET_ACCESS_KEY')

# Replace with your bucket name
bucket_name = 'mybucket'

s3 = boto3.client('s3', aws_access_key_id=aws_access_key_id, aws_secret_access_key=aws_secret_access_key)
obj = s3.get_object(Bucket=bucket_name, Key='model.pkl')
model = joblib.load(BytesIO(obj['Body'].read()))
```

Fig. 8: Heroku Deployment

To securely story our AWS Access Key ID and Secret Access Key when uploading our code to GitHub, we tried avoiding hardcoding them directly in the code. Instead, we used environment variables by creating a .env file (Fig. 9) in the root directory of the project:



```
AWS_ACCESS_KEY_ID=AKIAXXXXXXXXXXXXXXXX
AWS_SECRET_ACCESS_KEY=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

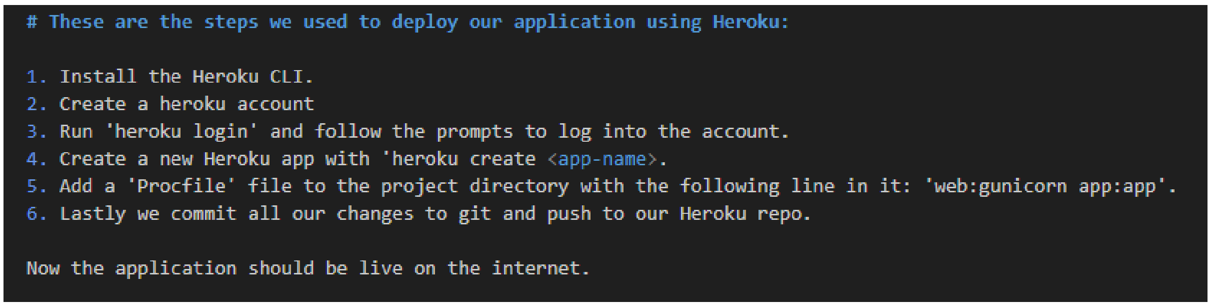
Fig. 9: AWS Access Keys

The .env file was added to a 'gitignore' file to prevent it from being tracked by Git.

Before deploying the app on Heroku, we need to create a 'requirements.txt' file that lists all the Python dependencies required for the app. This can be done using the following command:

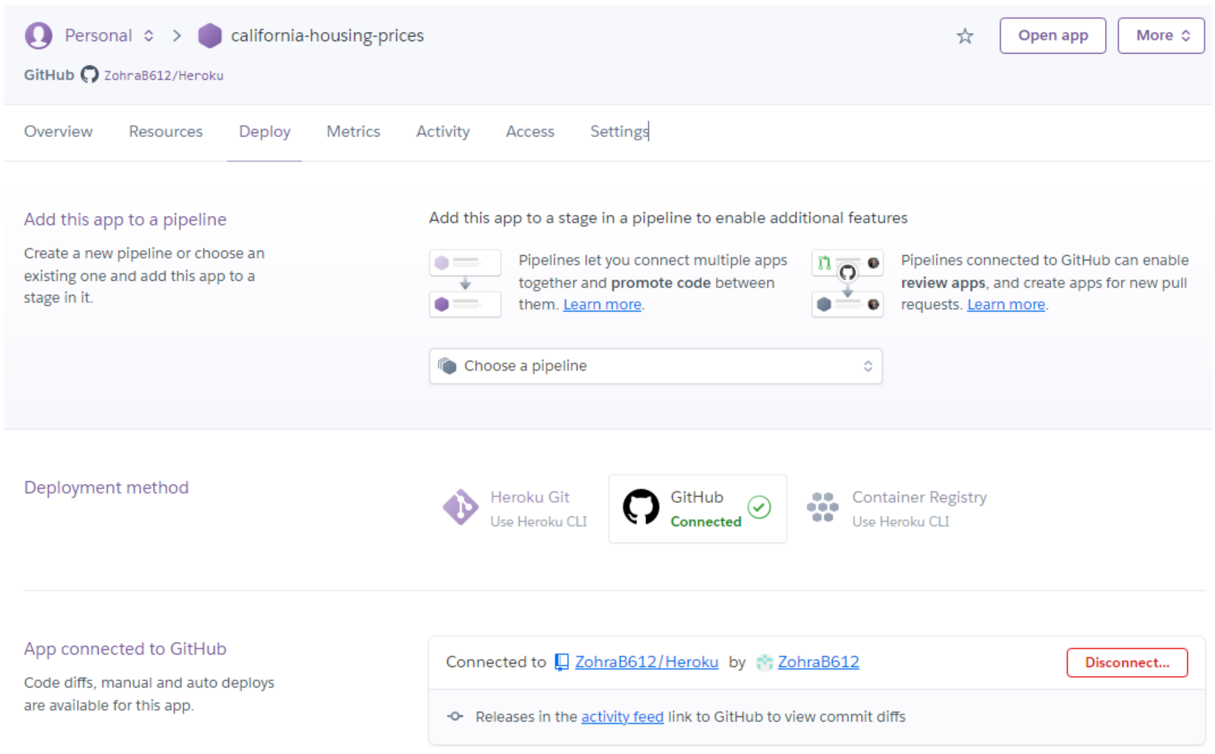
pip freeze > requirements.txt

Now, here are the steps for deploying the Flask app on Heroku (Fig. 10):



```
# These are the steps we used to deploy our application using Heroku:

1. Install the Heroku CLI.
2. Create a heroku account
3. Run 'heroku login' and follow the prompts to log into the account.
4. Create a new Heroku app with 'heroku create <app-name>.
5. Add a 'Procfile' file to the project directory with the following line in it: 'web:gunicorn app:app'.
6. Lastly we commit all our changes to git and push to our Heroku repo.

Now the application should be live on the internet.
```

Fig. 10: Steps to deploy app on Heroku



Personal > california-housing-prices

GitHub ZohraB612/Heroku

Open app | More

Overview | Resources | Deploy | Metrics | Activity | Access | Settings

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Choose a pipeline

Deployment method

Heroku Git — Use Heroku CLI

GitHub — Connected

Container Registry — Use Heroku CLI

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

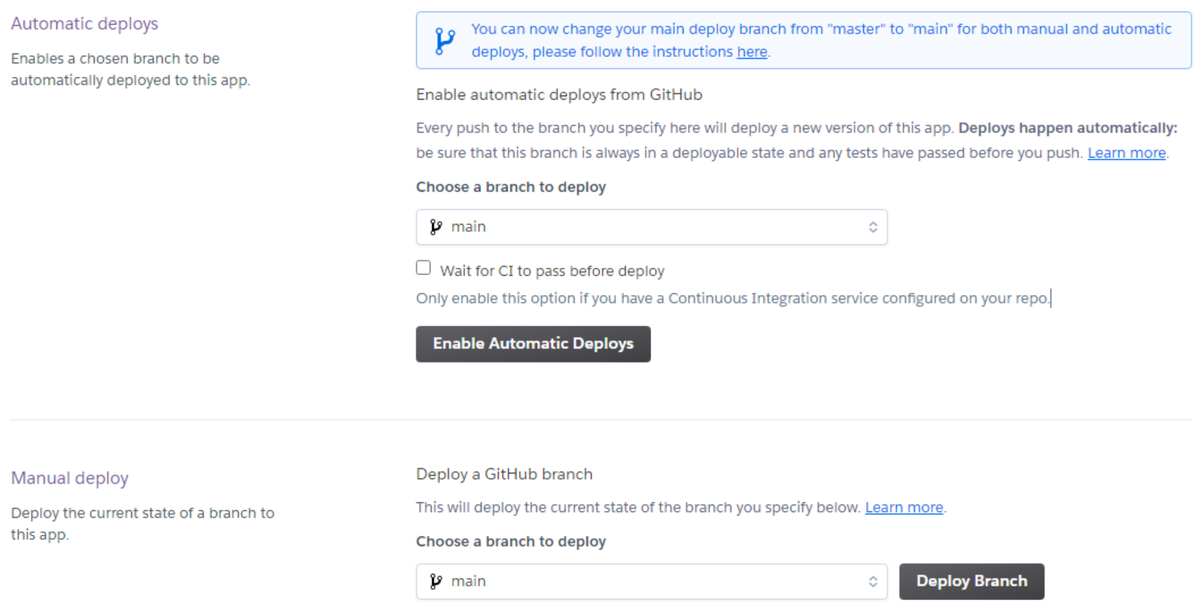
Connected to ZohraB612/Heroku by ZohraB612

Disconnect...

Releases in the activity feed link to GitHub to view commit diffs

Fig. 11: Heroku

Once the files have been pushed to GitHub, and our Heroku account is connected to our Github account (Fig. 11), we choose the repo where the files have been uploaded (i.e., Heroku). Next, we choose a branch to deploy (if more than one branch are available)(Fig. 12). In our case, we only have the main branch. Finally we deploy the branch and wait for the process to complete. Once the process is completed, a link is provided to be able to access the deployed web version of the app.



Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. Learn more.

Choose a branch to deploy

main

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Manual deploy

Deploy the current state of a branch to this app.

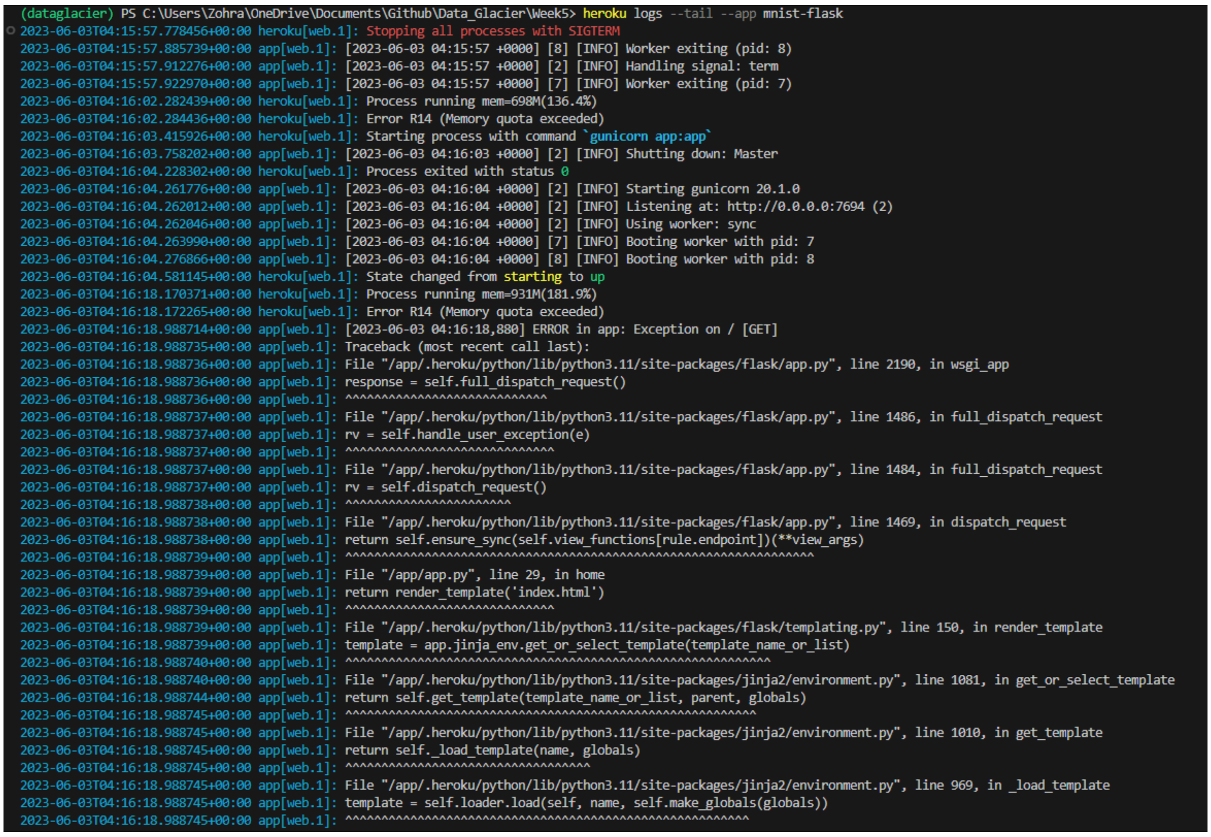
Deploy a GitHub branch

This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy

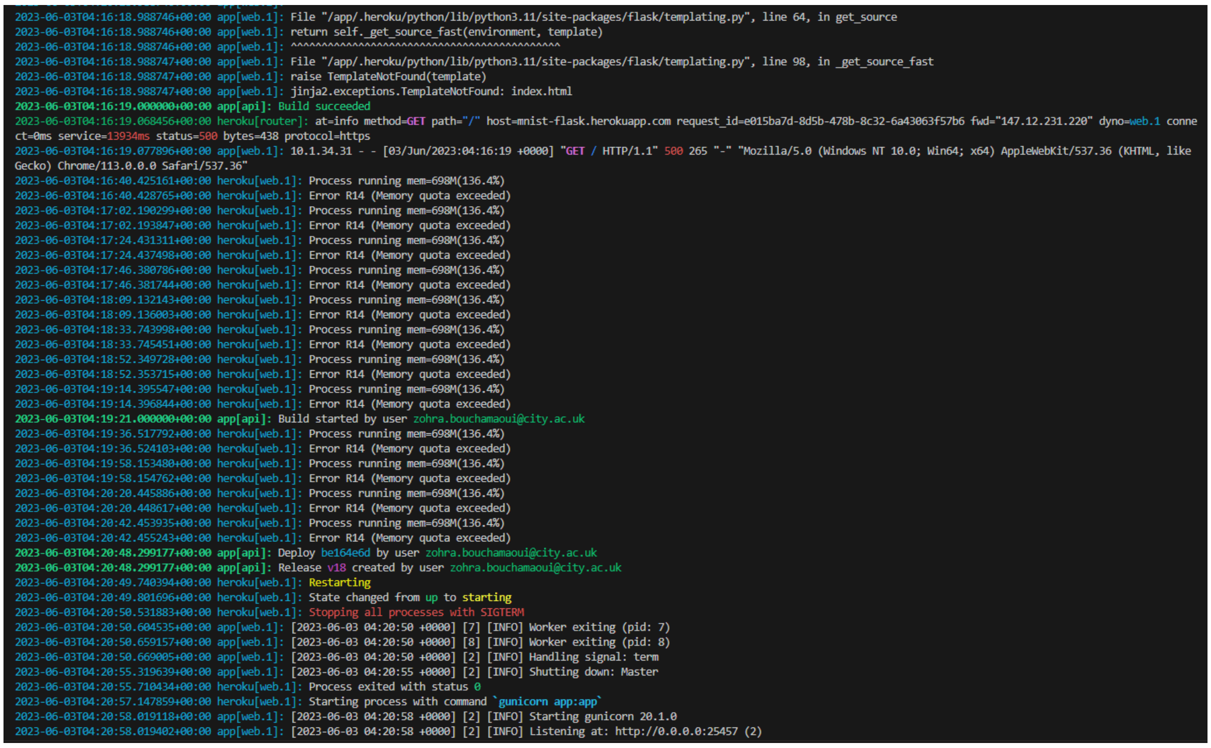
main

Fig. 12: Heroku

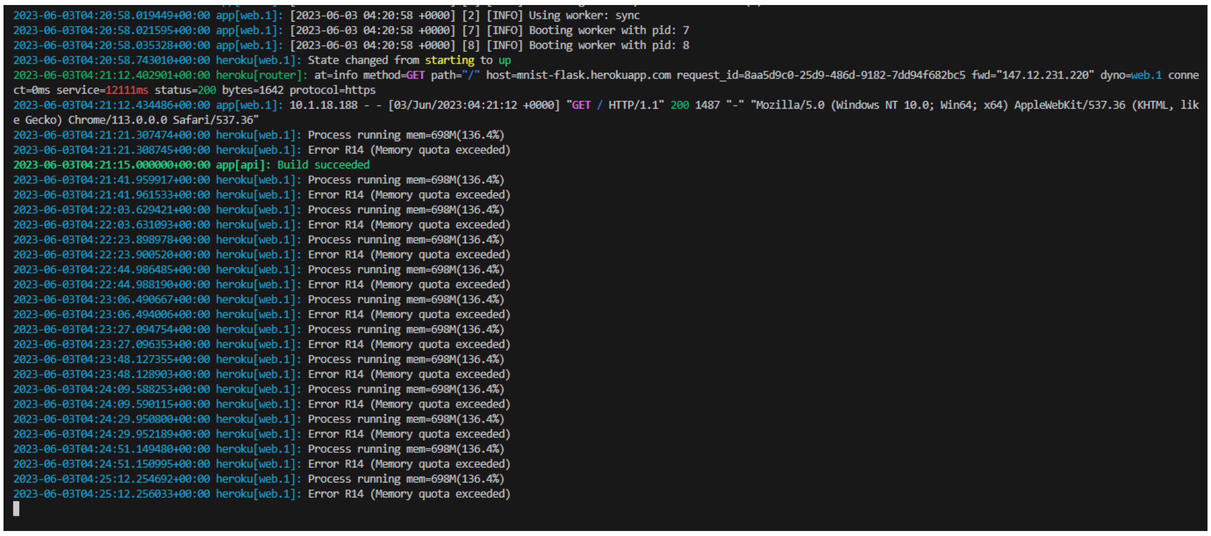


```
(dataglacier) PS C:\Users\Zohra\OneDrive\Documents\Github\Data_Glacier\Week5> heroku logs --tail --app mnist-flask
2023-06-03T04:15:57.778456+00:00 heroku[web.1]: Stopping all processes with SIGTERM
2023-06-03T04:15:57.885739+00:00 app[web.1]: [2023-06-03 04:15:57 +0000] [8] [INFO] Worker exiting (pid: 8)
2023-06-03T04:15:57.912276+00:00 app[web.1]: [2023-06-03 04:15:57 +0000] [2] [INFO] Handling signal: term
2023-06-03T04:15:57.922970+00:00 app[web.1]: [2023-06-03 04:15:57 +0000] [7] [INFO] Worker exiting (pid: 7)
2023-06-03T04:16:02.282439+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:16:02.284436+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:16:03.415926+00:00 heroku[web.1]: Starting process with command `gunicorn app:app`
2023-06-03T04:16:03.758202+00:00 app[web.1]: [2023-06-03 04:16:03 +0000] [2] [INFO] Shutting down: Master
2023-06-03T04:16:04.228302+00:00 heroku[web.1]: Process exited with status 0
2023-06-03T04:16:04.261776+00:00 app[web.1]: [2023-06-03 04:16:04 +0000] [2] [INFO] Starting gunicorn 20.1.0
2023-06-03T04:16:04.262012+00:00 app[web.1]: [2023-06-03 04:16:04 +0000] [2] [INFO] Listening at: http://0.0.0.0:7694 (2)
2023-06-03T04:16:04.262046+00:00 app[web.1]: [2023-06-03 04:16:04 +0000] [2] [INFO] Using worker: sync
2023-06-03T04:16:04.263990+00:00 app[web.1]: [2023-06-03 04:16:04 +0000] [7] [INFO] Booting worker with pid: 7
2023-06-03T04:16:04.276866+00:00 app[web.1]: [2023-06-03 04:16:04 +0000] [8] [INFO] Booting worker with pid: 8
2023-06-03T04:16:04.581145+00:00 heroku[web.1]: State changed from starting to up
2023-06-03T04:16:18.170371+00:00 heroku[web.1]: Process running mem=931M(181.9%)
2023-06-03T04:16:18.172265+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:16:18.988714+00:00 app[web.1]: [2023-06-03 04:16:18,880] ERROR in app: Exception on / [GET]
2023-06-03T04:16:18.988735+00:00 app[web.1]: Traceback (most recent call last):
2023-06-03T04:16:18.988736+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.11/site-packages/flask/app.py", line 2190, in wsgi_app
2023-06-03T04:16:18.988736+00:00 app[web.1]: response = self.full_dispatch_request()
2023-06-03T04:16:18.988736+00:00 app[web.1]: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2023-06-03T04:16:18.988737+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.11/site-packages/flask/app.py", line 1486, in full_dispatch_request
2023-06-03T04:16:18.988737+00:00 app[web.1]: rv = self.handle_user_exception(e)
2023-06-03T04:16:18.988737+00:00 app[web.1]: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2023-06-03T04:16:18.988737+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.11/site-packages/flask/app.py", line 1484, in full_dispatch_request
2023-06-03T04:16:18.988737+00:00 app[web.1]: rv = self.dispatch_request()
2023-06-03T04:16:18.988738+00:00 app[web.1]: ^^^^^^^^^^^^^^^^^^^^^^^
2023-06-03T04:16:18.988738+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.11/site-packages/flask/app.py", line 1469, in dispatch_request
2023-06-03T04:16:18.988738+00:00 app[web.1]: return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
2023-06-03T04:16:18.988739+00:00 app[web.1]: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2023-06-03T04:16:18.988739+00:00 app[web.1]: File "/app/app.py", line 29, in home
2023-06-03T04:16:18.988739+00:00 app[web.1]: return render_template('index.html')
2023-06-03T04:16:18.988739+00:00 app[web.1]: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2023-06-03T04:16:18.988739+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.11/site-packages/flask/templating.py", line 150, in render_template
2023-06-03T04:16:18.988739+00:00 app[web.1]: template = app.jinja_env.get_or_select_template(template_name_or_list)
2023-06-03T04:16:18.988740+00:00 app[web.1]: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2023-06-03T04:16:18.988740+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.11/site-packages/jinja2/environment.py", line 1081, in get_or_select_template
2023-06-03T04:16:18.988744+00:00 app[web.1]: return self.get_template(template_name_or_list, parent, globals)
2023-06-03T04:16:18.988745+00:00 app[web.1]: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2023-06-03T04:16:18.988745+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.11/site-packages/jinja2/environment.py", line 1010, in get_template
2023-06-03T04:16:18.988745+00:00 app[web.1]: return self._load_template(name, globals)
2023-06-03T04:16:18.988745+00:00 app[web.1]: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2023-06-03T04:16:18.988745+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.11/site-packages/jinja2/environment.py", line 969, in _load_template
2023-06-03T04:16:18.988745+00:00 app[web.1]: template = self.loader.load(self, name, self.make_globals(globals))
2023-06-03T04:16:18.988745+00:00 app[web.1]: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Fig. 13: Deployment Process (1)

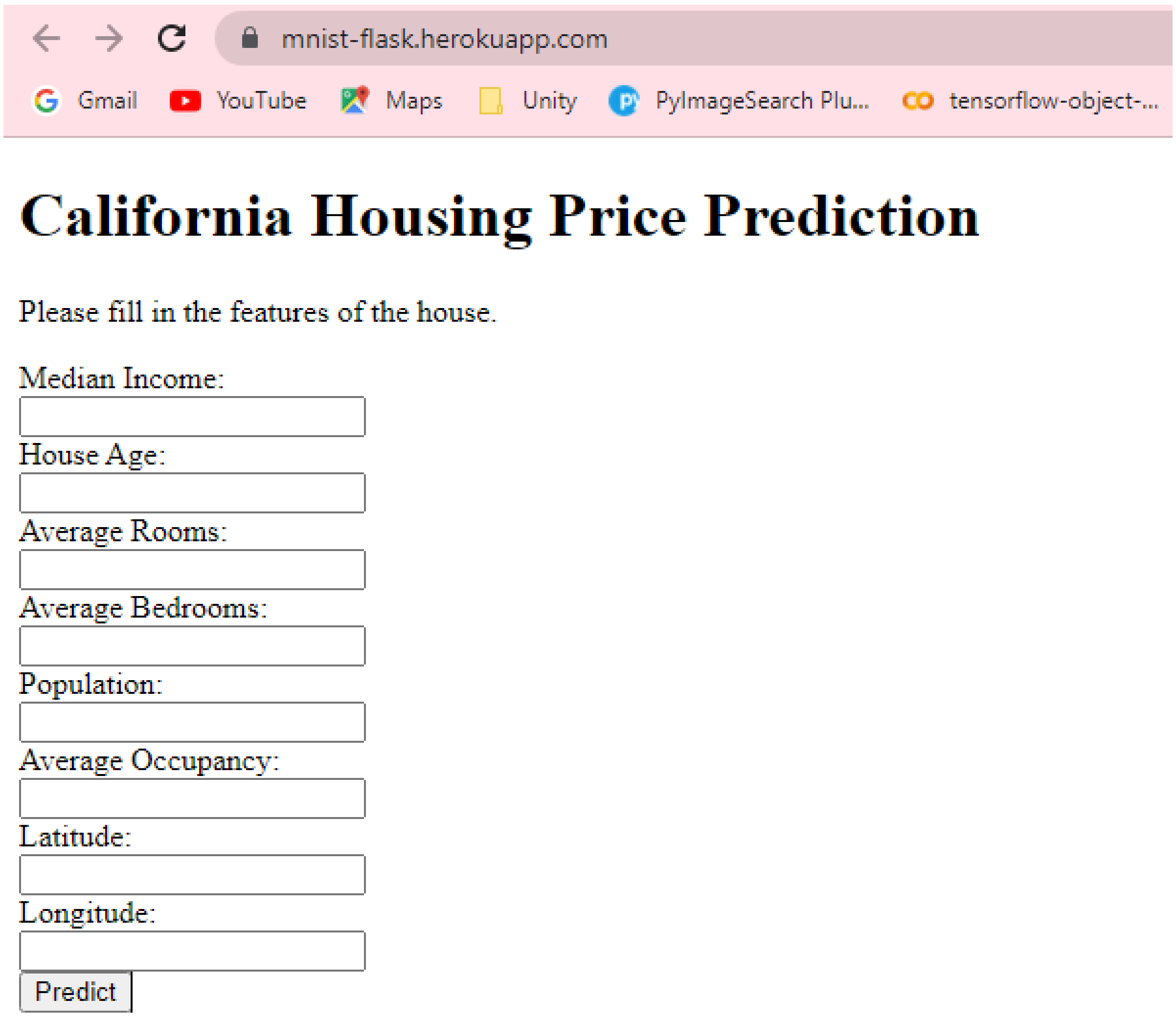


```
2023-06-03T04:16:18.988746+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.11/site-packages/flask/templating.py", line 64, in get_source
2023-06-03T04:16:18.988746+00:00 app[web.1]: return self._get_source_fast(environment, template)
2023-06-03T04:16:18.988746+00:00 app[web.1]: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2023-06-03T04:16:18.988747+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.11/site-packages/flask/templating.py", line 98, in _get_source_fast
2023-06-03T04:16:18.988747+00:00 app[web.1]: raise TemplateNotFound(template)
2023-06-03T04:16:18.988747+00:00 app[web.1]: jinja2.exceptions.TemplateNotFound: index.html
2023-06-03T04:16:19.000000+00:00 app[api]: Build succeeded
2023-06-03T04:16:19.068456+00:00 heroku[router]: at=info method=GET path="/" host=mnist-flask.herokuapp.com request_id=e015ba7d-8d5b-478b-8c32-6a43063f57b6 fwd="147.12.231.220" dyno=web.1 conne
ct=0ms service=13934ms status=500 bytes=438 protocol=https
2023-06-03T04:16:19.077896+00:00 app[web.1]: 10.1.34.31 - - [03/Jun/2023:04:16:19 +0000] "GET / HTTP/1.1" 500 265 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/113.0.0.0 Safari/537.36"
2023-06-03T04:16:40.425161+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:16:40.428765+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:17:02.190299+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:17:02.193847+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:17:24.431311+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:17:24.437498+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:17:46.380786+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:17:46.381744+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:18:09.132143+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:18:09.136003+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:18:33.743998+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:18:33.745451+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:18:52.349728+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:18:52.353715+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:19:14.395547+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:19:14.396844+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:19:21.000000+00:00 app[api]: Build started by user zohra.bouchamaoui@city.ac.uk
2023-06-03T04:19:36.517792+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:19:36.524103+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:19:58.153480+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:19:58.154762+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:20:20.445886+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:20:20.448617+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:20:42.453935+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:20:42.455243+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:20:48.299177+00:00 app[api]: Deploy be164e6d by user zohra.bouchamaoui@city.ac.uk
2023-06-03T04:20:48.299177+00:00 app[api]: Release v18 created by user zohra.bouchamaoui@city.ac.uk
2023-06-03T04:20:49.740394+00:00 heroku[web.1]: Restarting
2023-06-03T04:20:49.801696+00:00 heroku[web.1]: State changed from up to starting
2023-06-03T04:20:50.531883+00:00 heroku[web.1]: Stopping all processes with SIGTERM
2023-06-03T04:20:50.604535+00:00 app[web.1]: [2023-06-03 04:20:50 +0000] [7] [INFO] Worker exiting (pid: 7)
2023-06-03T04:20:50.659157+00:00 app[web.1]: [2023-06-03 04:20:50 +0000] [8] [INFO] Worker exiting (pid: 8)
2023-06-03T04:20:50.669005+00:00 app[web.1]: [2023-06-03 04:20:50 +0000] [2] [INFO] Handling signal: term
2023-06-03T04:20:55.319639+00:00 app[web.1]: [2023-06-03 04:20:55 +0000] [2] [INFO] Shutting down: Master
2023-06-03T04:20:55.710434+00:00 heroku[web.1]: Process exited with status 0
2023-06-03T04:20:57.147859+00:00 heroku[web.1]: Starting process with command `gunicorn app:app`
2023-06-03T04:20:58.019118+00:00 app[web.1]: [2023-06-03 04:20:58 +0000] [2] [INFO] Starting gunicorn 20.1.0
2023-06-03T04:20:58.019402+00:00 app[web.1]: [2023-06-03 04:20:58 +0000] [2] [INFO] Listening at: http://0.0.0.0:25457 (2)
```



```
2023-06-03T04:20:58.019449+00:00 app[web.1]: [2023-06-03 04:20:58 +0000] [2] [INFO] Using worker: sync
2023-06-03T04:20:58.021595+00:00 app[web.1]: [2023-06-03 04:20:58 +0000] [7] [INFO] Booting worker with pid: 7
2023-06-03T04:20:58.035328+00:00 app[web.1]: [2023-06-03 04:20:58 +0000] [8] [INFO] Booting worker with pid: 8
2023-06-03T04:20:58.743010+00:00 heroku[web.1]: State changed from starting to up
2023-06-03T04:21:12.402901+00:00 heroku[router]: at=info method=GET path="/" host=mnist-flask.herokuapp.com request_id=8aa5d9c0-25d9-486d-9182-7dd94f682bc5 fwd="147.12.231.220" dyno=web.1 conne
ct=0ms service=12111ms status=200 bytes=1642 protocol=https
2023-06-03T04:21:12.434486+00:00 app[web.1]: 10.1.18.188 - - [03/Jun/2023:04:21:12 +0000] "GET / HTTP/1.1" 200 1487 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, lik
e Gecko) Chrome/113.0.0.0 Safari/537.36"
2023-06-03T04:21:21.307474+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:21:21.308745+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:21:15.000000+00:00 app[api]: Build succeeded
2023-06-03T04:21:41.959917+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:21:41.961533+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:22:03.629421+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:22:03.631093+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:22:23.898978+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:22:23.900520+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:22:44.986485+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:22:44.988190+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:23:06.490667+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:23:06.494006+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:23:27.094754+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:23:27.096353+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:23:48.127355+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:23:48.128903+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:24:09.588253+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:24:09.590115+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:24:29.950800+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:24:29.952189+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:24:51.149480+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:24:51.150995+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
2023-06-03T04:25:12.254692+00:00 heroku[web.1]: Process running mem=698M(136.4%)
2023-06-03T04:25:12.256033+00:00 heroku[web.1]: Error R14 (Memory quota exceeded)
```

Fig. 14: Deployment Process (2)



California Housing Price Prediction

Please fill in the features of the house.

Median Income:

House Age:

Average Rooms:

Average Bedrooms:

Population:

Average Occupancy:

Latitude:

Longitude:

Fig. 15: index.html template page