

Week 5: Cloud and API deployment

Name: Cloud and API deployment

Report date: 10/10/2024

Internship Batch: LISUM34

Version: 1.0

Data intake by: Mahima Sadananda

Data intake reviewer: Data Glacier

Data storage location: https://github.com/MahimaSadananda/DataGlacier_Internship/tree/main

Tabular data details: titanic

Total number of observations	891
Total number of files	1
Total number of features	6
Base format of the file	.CSV
Size of the data	15.1 KB

1. Model Setup

```
# Importing the Libraries
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
import pickle

[43] # Reading the data
titanic = pd.read_csv('/content/titanic.csv')

# handling missing values
titanic['Age'].fillna(titanic['Age'].mean(), inplace=True)

Show hidden output

[45] # Splitting X and y
X = titanic.drop('Survived', axis=1)
y = titanic['Survived']

[47] # logistic regressor
logistic_regressor = LogisticRegression()

# Fitting the model
logistic_regressor.fit(X, y)

LogisticRegression
LogisticRegression()

[48] # Saving model to disk
pickle.dump(logistic_regressor, open('titanic_model.pkl', 'wb'))

[49] # Loading the model back to compare results
model = pickle.load(open('titanic_model.pkl', 'rb'))
print(model.predict(X[:5]))

[0 1 0 1 0]
```

2. App.py

```
1 import numpy as np
2 from flask import Flask, request, render_template
3 import pickle
4
5 app = Flask(__name__)
6 model = pickle.load(open('model.pkl', 'rb'))
7
8 @app.route('/')
9 def home():
10     return render_template('index.html')
11
12 @app.route('/predict', methods=['POST'])
13 def predict():
14     '''
15     For rendering results on HTML GUI
16     '''
17     int_features = [int(x) for x in request.form.values()]
18     final_features = [np.array(int_features)]
19     prediction = model.predict(final_features)
20
21     if prediction[0] == 0:
22         output = "Did Not Survive :("
23     else:
24         output = "Survived :)"
25
26     return render_template('index.html', prediction_text='Passenger - {}'.format(output))
27
28 if __name__ == "__main__":
29     app.run(debug=True)
```

3. HTML

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
  <div class="login">
    <h1>Titanic Survival Prediction</h1>

    | <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict') }}" method="post">
      <input type="text" name="Pclass" placeholder="Passenger Class" required="required" />
      <input type="text" name="Age" placeholder="Passenger Age" required="required" />
      <input type="text" name="SibSp" placeholder="No. of Siblings" required="required" />
      <input type="text" name="Parch" placeholder="Parents per Children" required="required" />
      <input type="text" name="Fare" placeholder="Ride Fare" required="required" />

      | <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

    <br>
    <br>
    {{ prediction_text }}

  </div>
  
</body>
</html>
```

4. CSS

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);

.btn {
  display: inline-block;
  *display: inline;
  *zoom: 1;
  padding: 4px 10px 4px;
  margin-bottom: 0;
  font-size: 13px;
  line-height: 18px;
  color: #333333;
  text-align: center;
  text-shadow: 0 1px 1px rgba(255, 255, 255, 0.75);
  vertical-align: middle;
  background-color: #f5f5f5;
  background-image: -moz-linear-gradient(top, #ffffff, #e6e6e6);
  background-image: -ms-linear-gradient(top, #ffffff, #e6e6e6);
  background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#ffffff), to(#e6e6e6));
  background-image: -webkit-linear-gradient(top, #ffffff, #e6e6e6);
  background-image: -o-linear-gradient(top, #ffffff, #e6e6e6);
  background-image: linear-gradient(to bottom, #ffffff, #e6e6e6); /* Updated for gradient direction */
  background-repeat: repeat-x;
  filter: progid:dximagetransform.microsoft.gradient(startColorstr=#ffffff, endColorstr=#e6e6e6, GradientType=0);
  border-color: #e6e6e6 #e6e6e6 #e6e6e6;
  border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);
  border: 1px solid #e6e6e6;
  -webkit-border-radius: 4px;
  -moz-border-radius: 4px;
  border-radius: 4px;
  -webkit-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);
  -moz-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);
  box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);
  cursor: pointer;
  *margin-left: .3em;
}

.btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] {
  background-color: #e6e6e6;
}

.btn-large {
  padding: 9px 14px;
  font-size: 15px;
  line-height: normal;
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
}

.btn:hover {
}

.btn-primary {
  background-color: #007bff; /* Changed to a blue color */
  background-image: -moz-linear-gradient(top, #3399ff, #007bff); /* Updated for blue gradient */
  background-image: -ms-linear-gradient(top, #3399ff, #007bff);
  background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#3399ff), to(#007bff));
  background-image: -webkit-linear-gradient(top, #3399ff, #007bff);
  background-image: -o-linear-gradient(top, #3399ff, #007bff);
  background-image: linear-gradient(to bottom, #3399ff, #007bff); /* Updated for gradient direction */
  background-repeat: repeat-x;
  filter: progid:dximagetransform.microsoft.gradient(startColorstr=#3399ff, endColorstr=#007bff, GradientType=0);
  border: 1px solid #0056b3; /* Darker border color */
  text-shadow: 1px 1px 1px rgba(0, 0, 0, 0.4);
  box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.5);
}

.btn-primary:hover, .btn-primary:active, .btn-primary.active, .btn-primary.disabled, .btn-primary[disabled] {
  filter: none;
  background-color: #0056b3; /* Darker on hover */
}

.btn-block {
  width: 100%;
  display: block;
}

* {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  -ms-box-sizing: border-box;
  -o-box-sizing: border-box;
  box-sizing: border-box;
}
```

```
html {
  width: 100%;
  height: 100%;
  overflow: hidden;
}

body {
  width: 100%;
  height: 100%;
  font-family: 'Open Sans', sans-serif;
  color: #fff;
  font-size: 18px;
  text-align: center;
  letter-spacing: 1.2px;
  background: linear-gradient(to bottom, #1e90ff, #0056b3) !important; /* Updated background to blue gradient */
}

.login {
  position: absolute;
  top: 40%;
  left: 50%;
  margin: -150px 0 0 -150px;
  width: 400px;
  height: 400px;
}

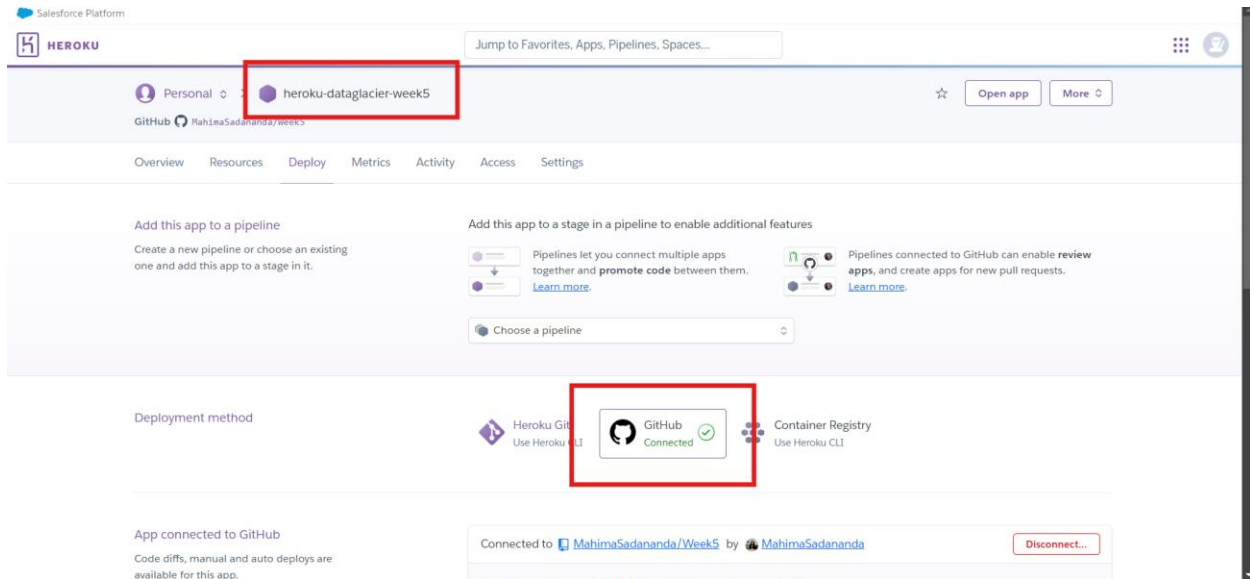
.login h1 {
  color: #fff;
  text-shadow: 0 0 10px rgba(0, 0, 0, 0.3);
  letter-spacing: 1px;
  text-align: center;
}

input {
  width: 100%;
  margin-bottom: 10px;
  background: rgba(0, 0, 0, 0.3);
  border: none;
  outline: none;
  padding: 10px;
  font-size: 13px;
  color: #fff;
  text-shadow: 1px 1px 1px rgba(0, 0, 0, 0.3);
  border: 1px solid rgba(0, 0, 0, 0.3);
  border-radius: 4px;
  box-shadow: inset 0 -5px 45px rgba(100, 100, 100, 0.2), 0 1px 1px rgba(255, 255, 255, 0.2);
  -webkit-transition: box-shadow .5s ease;
  -moz-transition: box-shadow .5s ease;
  -o-transition: box-shadow .5s ease;
  -ms-transition: box-shadow .5s ease;
  transition: box-shadow .5s ease;
}

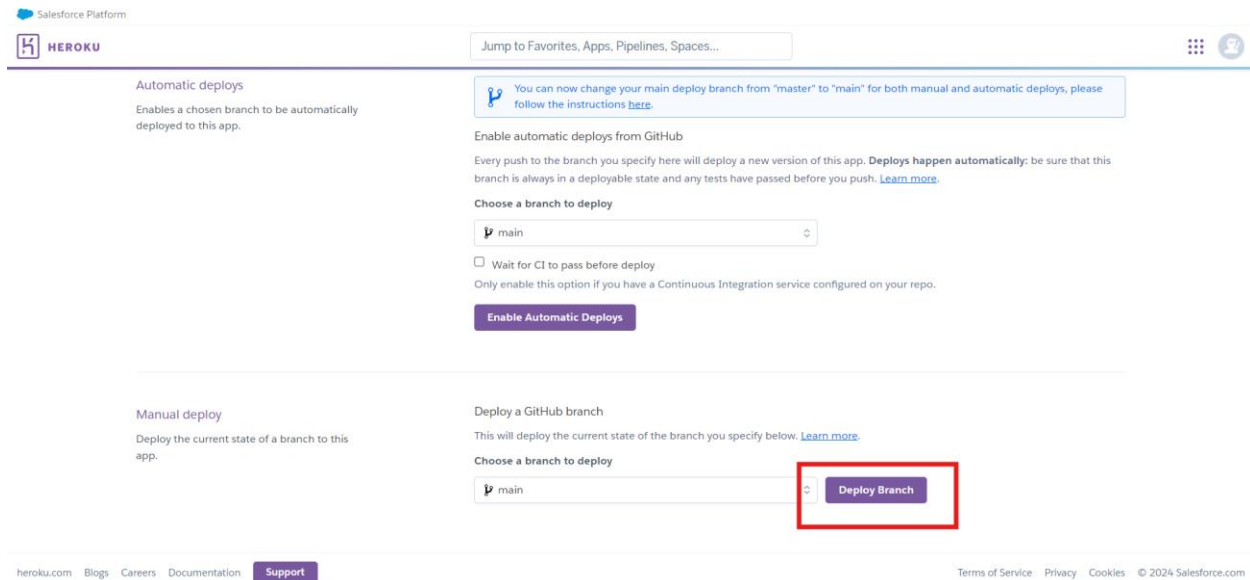
input:focus {
  box-shadow: inset 0 -5px 45px rgba(100, 100, 100, 0.4), 0 1px 1px rgba(255, 255, 255, 0.2);
}
```

5. Heroku Deployment

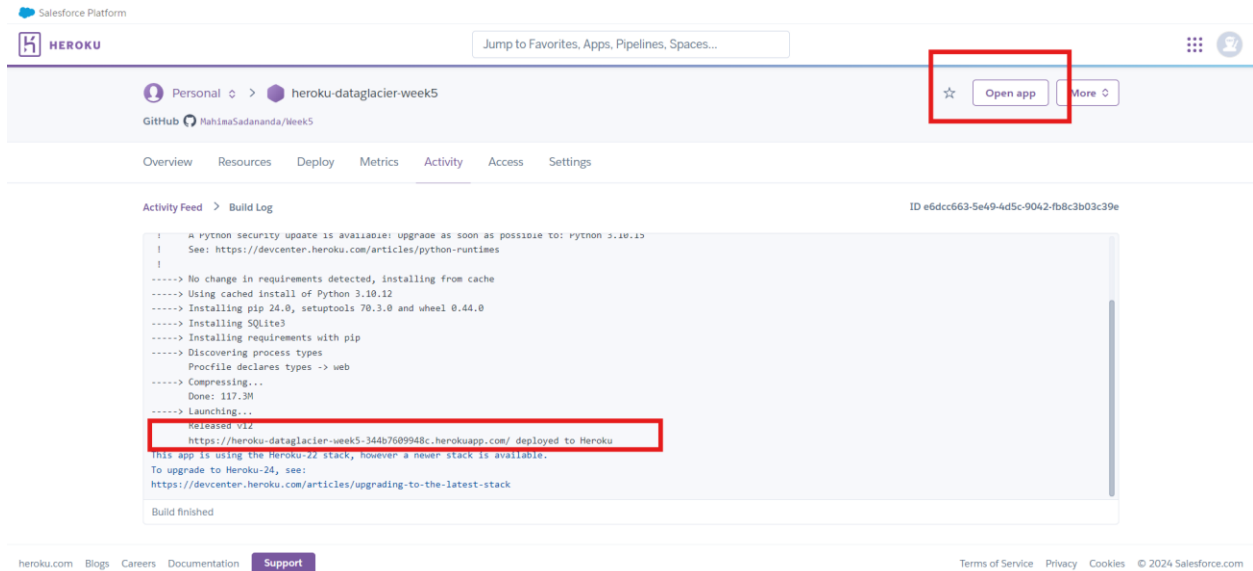
- Created an app and linked to Github Repo



- Deployed the model



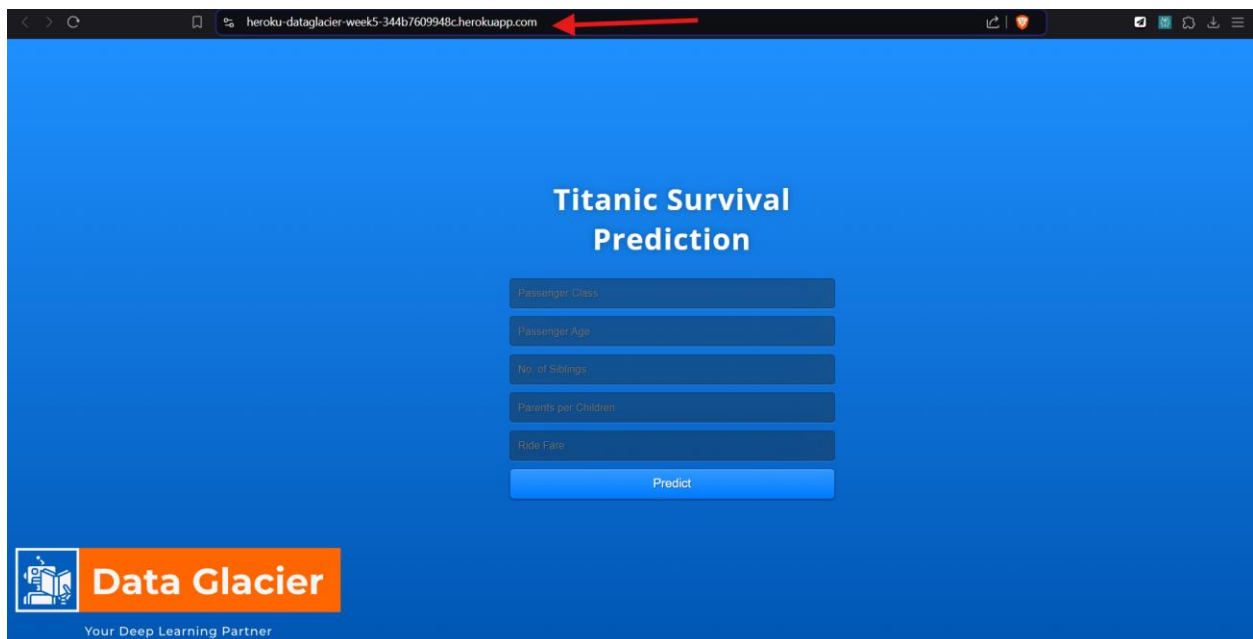
- Build Log is shown below. Click on the link or “Open App” to launch the app



The screenshot shows the Heroku dashboard for the app `heroku-dataglacier-week5`. The `Open app` button is highlighted with a red box. The build log shows the app was successfully deployed to Heroku. The build log text is as follows:

```
! A python security update is available: upgrade as soon as possible to: python 3.10.13
! See: https://devcenter.heroku.com/articles/python-runtimes
!
----> No change in requirements detected, installing from cache
----> Using cached install of Python 3.10.12
----> Installing pip 24.0, setuptools 70.3.0 and wheel 0.44.0
----> Installing SQLite3
----> Installing requirements with pip
----> Discovering process types
Procfile declares types -> web
----> Compressing...
Done: 117.3M
----> Launching...
Release v12
https://heroku-dataglacier-week5-344b7609948c.herokuapp.com/ deployed to Heroku
This app is using the Heroku-22 stack, however a newer stack is available.
To upgrade to Heroku-24, see:
https://devcenter.heroku.com/articles/upgrading-to-the-latest-stack
Build finished
```

- App is deployed on Heroku (can be accessed here <https://heroku-dataglacier-week5-344b7609948c.herokuapp.com/>)



The screenshot shows the Titanic Survival Prediction app running on Heroku. The app has a blue background and features a form with input fields for Passenger Class, Passenger Age, No. of Siblings, Parents per Child, and Ride Fare, followed by a Predict button. The Data Glacier logo is visible in the bottom left corner.

- Entered the values

The screenshot shows a web browser window with the URL `heroku-dataglacier-week5-344b7609948c.herokuapp.com`. The page has a blue background and is titled "Titanic Survival Prediction". It features a form with five input fields containing the values: 1, 12, 1, 1, and 50. Below these fields is a blue "Predict" button. In the bottom left corner, there is a logo for "Data Glacier" with the tagline "Your Deep Learning Partner".

- Clicked on "Predict"

The screenshot shows the same web application after the "Predict" button was clicked. The URL in the browser is now `heroku-dataglacier-week5-344b7609948c.herokuapp.com/predict`, indicated by a red arrow. The form fields are now labeled: "Passenger Class", "Passenger Age", "No. of Siblings", "Parents per Children", and "Ride Fare". Below these fields is a blue "Predict" button. A red box highlights the output text: "Passenger - Survived :)". The "Data Glacier" logo and tagline remain in the bottom left corner.