

# Week 5 Cloud and API Deployment

**Name:** Bindu Musham

**Batch code:** LISUM20

**Submission date:** 5<sup>th</sup> May 2023

**Submission to:** Data Glacier

## Data:

Dataset is taken from Kaggle <https://www.kaggle.com/code/ashydv/sales-prediction-simple-linear-regression/input>

The dataset has sales data based on the money spent on 3 different modes of marketing, TV, Radio, and Newspaper.

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12.0
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9

## Problem Statement:

Build a model to predict sales on the money spent on these modes of advertising predictor variables.

## Build a model:

- Create dependent (y) and independent variables (X)

```
X = advertising.drop(['Sales'],axis=1)
y = advertising['Sales']
```

ii) The train test split with 70% train set and 30% test set.

```
#train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, test_size = 0.3, random_state = 100)
```

iii) Perform linear regression and fit the train set to predict sales based on advertising costs.

```
# Model building
from sklearn import linear_model
#Linear regression
reg=linear_model.LinearRegression()

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

iv) Saved the model using pickle.

```
# Make pickle file of our model
import pickle
pickle.dump(reg, open("model.pkl", "wb"))
```

## Creating a Flask web application:

1. Create a folder named **Week 4 deployment on flask** for this project.

app.py

templates/index.html

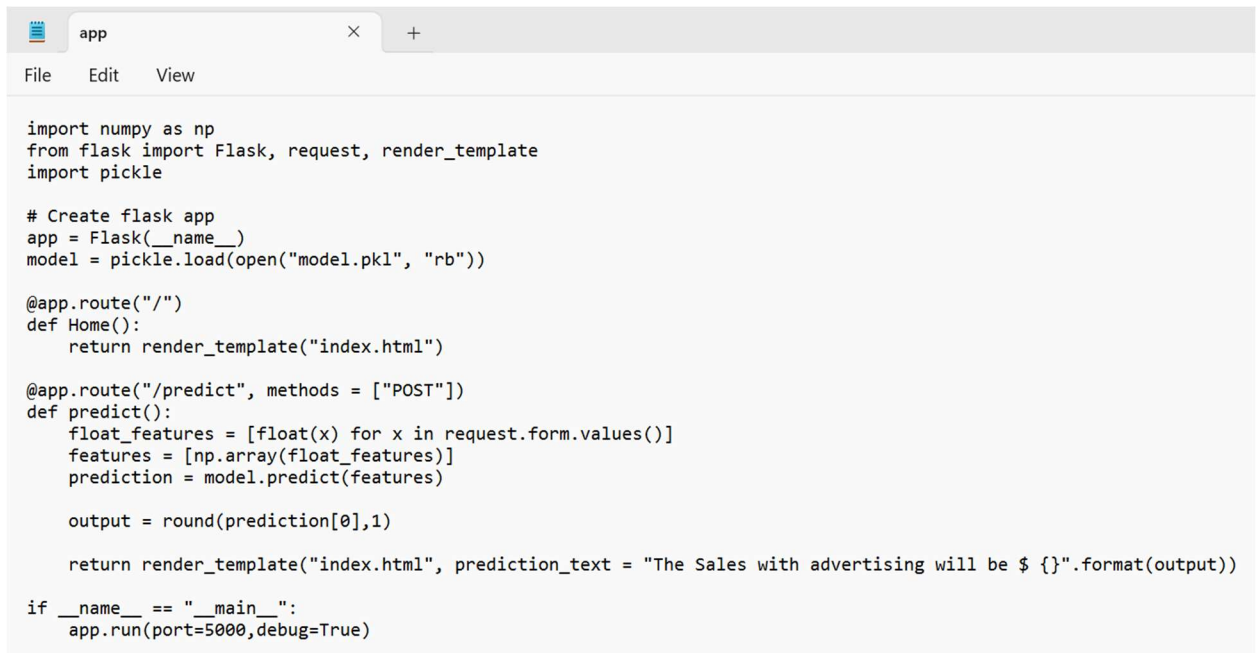
static/style.css

advertising.csv

Flask\_sales\_prediction.ipynb

2. When we ran above python jupyter notebook automatically the model.pkl file is created using pickle.

3. Create a main app.py file which has the main code with model file.



```
import numpy as np
from flask import Flask, request, render_template
import pickle

# Create flask app
app = Flask(__name__)
model = pickle.load(open("model.pkl", "rb"))

@app.route("/")
def Home():
    return render_template("index.html")

@app.route("/predict", methods = ["POST"])
def predict():
    float_features = [float(x) for x in request.form.values()]
    features = [np.array(float_features)]
    prediction = model.predict(features)

    output = round(prediction[0],1)

    return render_template("index.html", prediction_text = "The Sales with advertising will be $ {}".format(output))

if __name__ == "__main__":
    app.run(port=5000,debug=True)
```

Steps in app.py:

- i) Flask instance is created with a unique name **app**.
- ii) Load the **model.pkl** file.
- iii) The route path provides the home page details. Whenever anyone hits the server. It will display the **index.html** page.
- iv) In the index.html code, it takes the advertising costs for TV, Radio and Newspaper as inputs and predicts the sales value.
- v) The index.html is directly taken templates folder.

**Index.html:**

```
index
File Edit View

<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link rel="stylesheet" href="static/style.css">

</head>
<body>
  <div class="login">
    <h1>Sales Prediction based on Advertising costs</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
      <input type="text" name="TV" placeholder="TV" required="required" /><br>
      <input type="text" name="Radio" placeholder="Radio" required="required" /><br>
      <input type="text" name="Newspaper" placeholder="Newspaper" required="required" /><br>

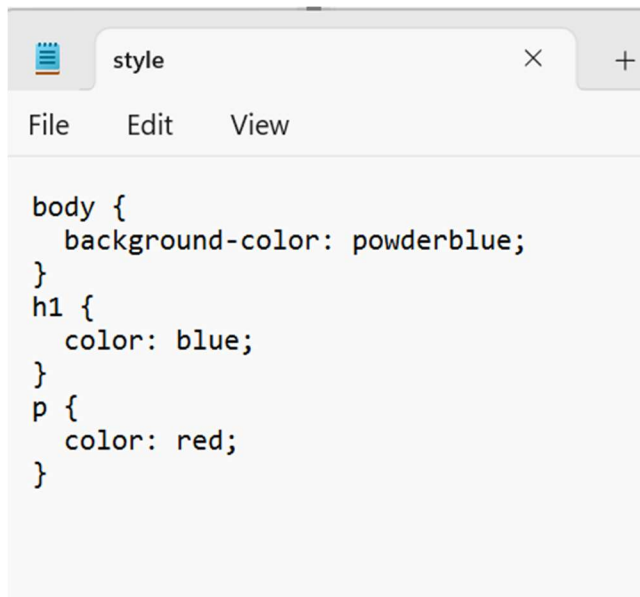
      <button type="submit" class="btn btn-primary btn-block btn-large">Predict Sales</button>
    </form>

    <br>
    <br>
    {{ prediction_text }}
  </div>

</body>
</html>
```

## Style.css:

Created a style.css for colorful homepage.

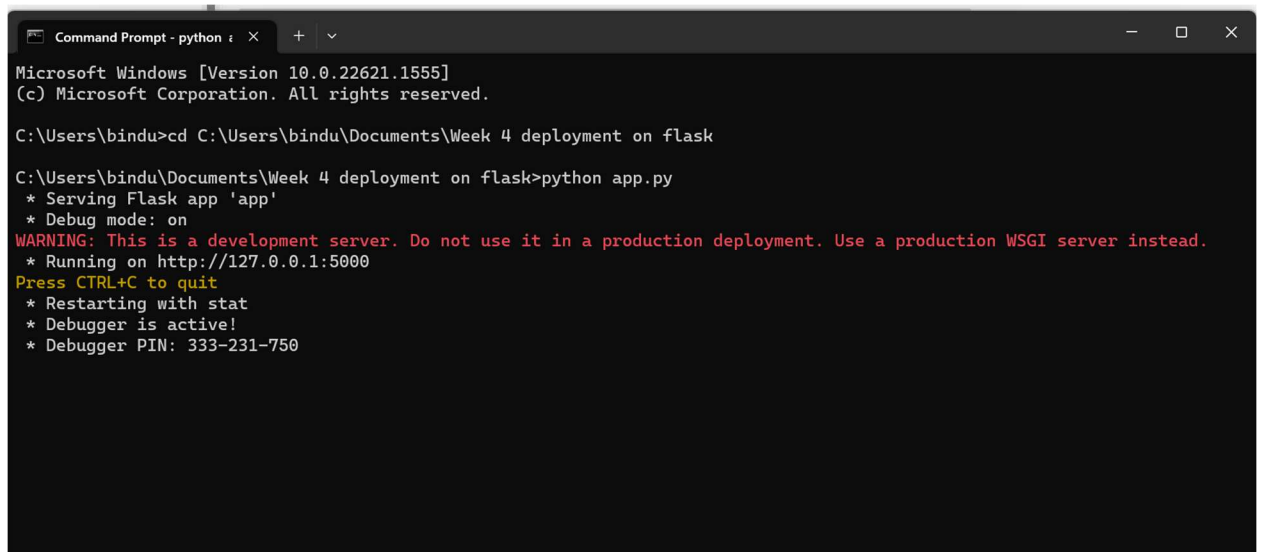


A screenshot of a web browser window. The address bar shows a file named 'style'. The browser has a menu bar with 'File', 'Edit', and 'View'. The main content area displays CSS code in a monospaced font.

```
body {  
  background-color: powderblue;  
}  
h1 {  
  color: blue;  
}  
p {  
  color: red;  
}
```

### Launch web application:

i) Run app.py as below.



A screenshot of a Windows Command Prompt window. The title bar reads 'Command Prompt - python'. The window shows the execution of a Flask application. The output includes a warning about using a development server and the URL to access the application.

```
Microsoft Windows [Version 10.0.22621.1555]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\bindu>cd C:\Users\bindu\Documents\Week 4 deployment on flask  
  
C:\Users\bindu\Documents\Week 4 deployment on flask>python app.py  
* Serving Flask app 'app'  
* Debug mode: on  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 333-231-750
```

ii) Open the URL <http://127.0.0.1:5000> in browser.

← → ↻ ⓘ 127.0.0.1:5000

Hue - Welcome Ho... YARN Databricks Commu... AWS Skill Builder Foundations: Data,... ChatGPT

## Sales Prediction based on Advertising costs

TV
Radio
Newspaper
Predict Sales

iii) Give the input advertising costs for TV, Radio, and Newspaper.

← → ↻ ⓘ 127.0.0.1:5000

Hue - Welcome Ho... YARN Databricks Commu... AWS Skill Builder Foundations: Data,... C

## Sales Prediction based on Advertising costs

150
120
90
Predict Sales

iv) Click on predict sales to get sales based on above costs.

← → ↻ ⓘ 127.0.0.1:5000/predict

Hue - Welcome Ho... YARN Databricks Commu... AWS Skill Builder Foundations: Data,... ChatGPT

## Sales Prediction based on Advertising costs

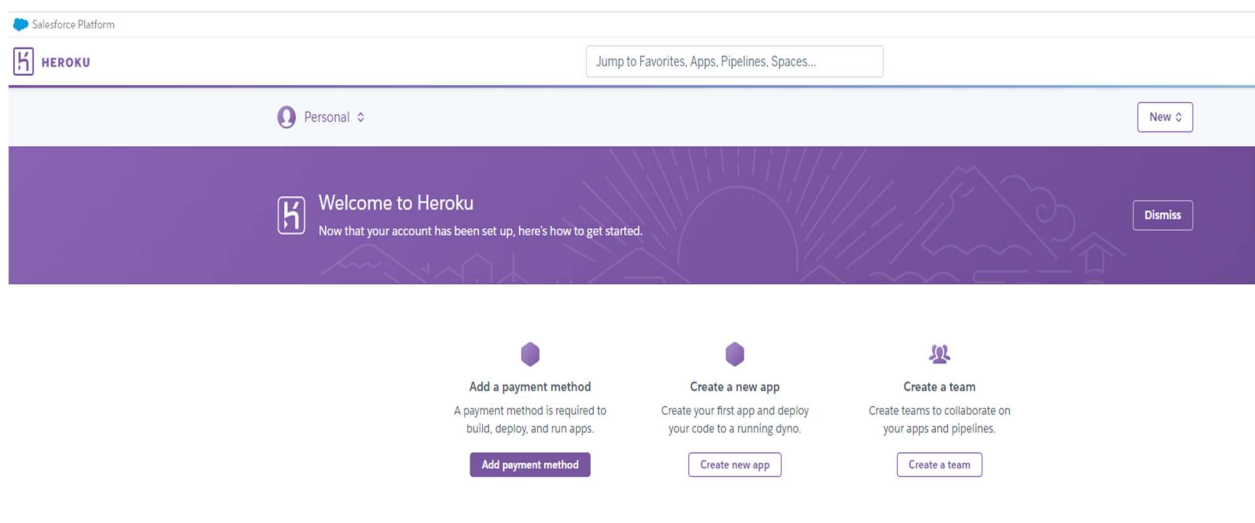
TV
Radio
Newspaper
Predict Sales

The Sales with advertising will be \$ 26.2

### Model Deployment on Heroku:

1. We can deploy our ML model by linking our GitHub repository to Heroku.

i) In the home page of Heroku click on Create a new app button.



ii) Enter application details and create application.

Create New App

App name

mi-model-application

mi-model-application is available

Choose a region

United States

Add to pipeline...

Create appCancel

iii) Link GitHub repository to Heroku.

HEROKU

Jump to Favorites, Apps, Pipelines, Spaces...

Personal > mi-model-application

Open appMore

OverviewResourcesDeployMetricsActivityAccessSettings

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

Connect to GitHub

Container Registry

Use Heroku CLI

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

View your code diffs on GitHub

Connect your app to a GitHub repository to see commit diffs in the activity log.

Deploy changes with GitHub

Connecting to a repository will allow you to deploy a branch to your app.

Automatic deploys from GitHub

Select a branch to deploy automatically whenever it is pushed to.

Create review apps in pipelines

Pipelines connected to GitHub can enable review apps, and create apps for new pull requests. [Learn more](#)

Connect to GitHub





Heroku Git  
Use Heroku CLI



GitHub  
Connected



Container Registry  
Use Heroku CLI

Connected to [bmusham/DataGlacier\\_Flask](#) by [bmusham](#)

Disconnect...

Releases in the [activity feed](#) link to GitHub to view commit diffs



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.

Enable Automatic Deploys

iv) Deploy the branch on Heroku website.

 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.

**Enable Automatic Deploys**

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

 main

**Deploy Branch**

- v) Once we click the deploy branch button, the required libraries are installed from requirements.txt file.
- vi) The application is successfully deployed on Heroku.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

 main

**Deploy Branch**

Receive code from GitHub



Build main **b4b15811**



Release phase



Deploy to Heroku

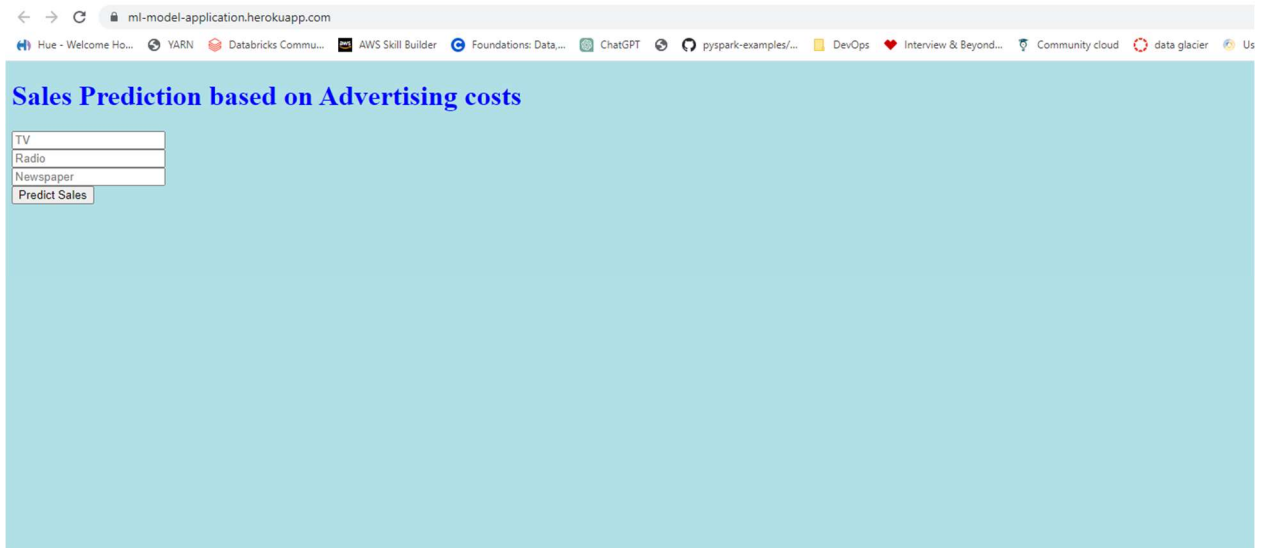


Your app was successfully deployed.

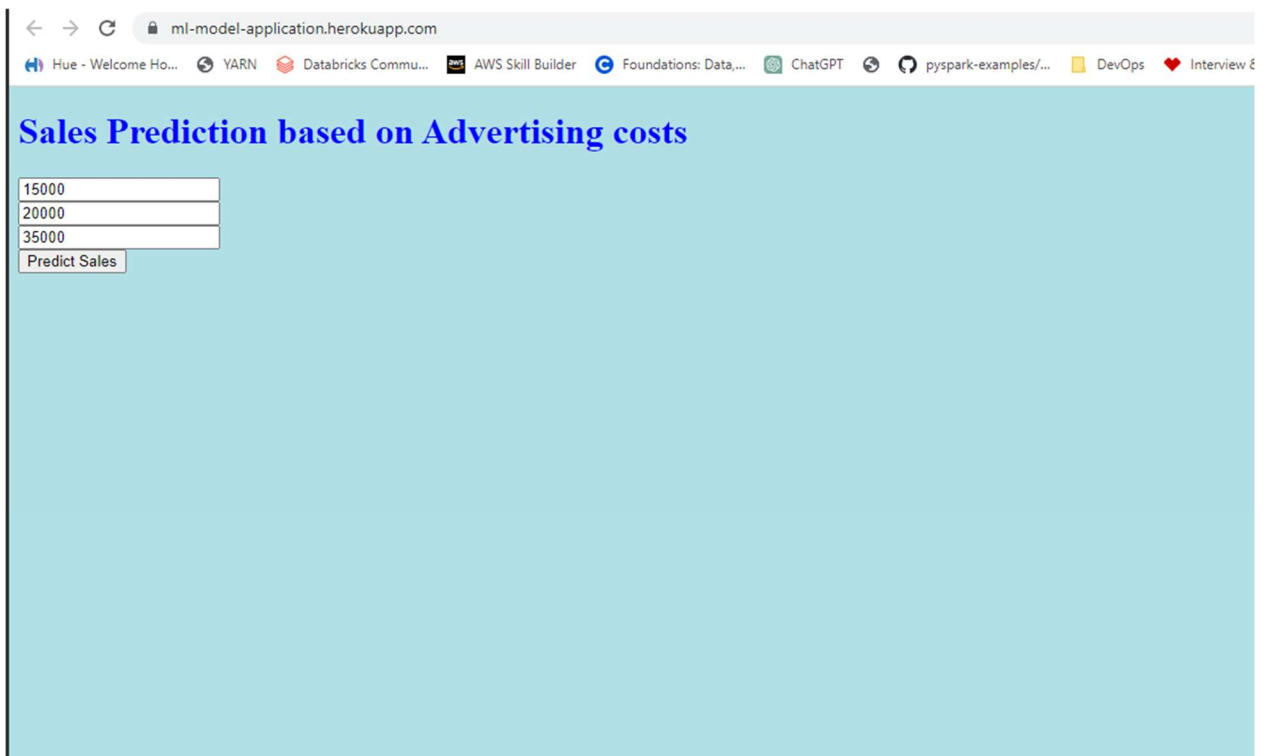
 **View**

- vii) After successful deployment click on view.

viii) The application is deployed on <https://ml-model-application.herokuapp.com/> website.



The screenshot shows a web browser window with the URL `ml-model-application.herokuapp.com`. The page title is "Sales Prediction based on Advertising costs". The form contains three input fields labeled "TV", "Radio", and "Newspaper", all of which are empty. Below these fields is a button labeled "Predict Sales". The browser's address bar and tabs are visible at the top.



This screenshot shows the same application interface as the first one, but with numerical values entered into the input fields. The "TV" field contains "15000", the "Radio" field contains "20000", and the "Newspaper" field contains "35000". The "Predict Sales" button remains visible below the fields. The browser window and tabs are also visible at the top.

← → ↻ ml-model-application.herokuapp.com/predict

Hue - Welcome Ho... YARN Databricks Commu... AWS Skill Builder Foundations: Data,... ChatGPT pyspark-examples/... DevOps Interview & Beyc

## Sales Prediction based on Advertising costs

TV
Radio
Newspaper
Predict Sales

The Sales with advertising will be \$ 3232.2

The application is working fine on cloud.