Week4: Deployment on Flask & Week5: Cloud and API Development

NAME: Siddharth Dudugu

Batch: LISUM 27

Data Science Intern at Data Glacier

Data:

The data (sample) were collected in S‹o Paulo, Brazil, in a university area, where there are some parties with groups of students from 18 to 28 years of age (average). The dataset used for this activity has 7 attributes, being a Target, with a period of one year. You have to predict the quantity of beer consumption based on the features that contain climate conditions of a given day.

Dataset Description:

The dataset contains 7 features

- Data: date of the record
- Temperatura Media (C): Average temperature of the day in Celsius
- Temperatura Minima (C): Minimum temperature of the day in Celsius
- Temperatura Maxima (C): Maximum temperature of the day in Celsius
- Precipitacao (mm): Percipitation in mm
- Final de Semana: If the day is weekend or not
- Consumo de cerveja (litros): Beer consumption in litres

How the data looks :

```
import pandas as pd
df1 = pd.read_csv('Consumo_cerveja.csv', parse_dates=['Data'])

df1.head()
```

	Data	Temperatura Media (C)	Temperatura Minima (C)	Temperatura Maxima (C)	Precipitacao (mm)	Final de Semana	Consumo de cerveja (litros)
0	2015-01-01	27,3	23,9	32,5	0	0.0	25.461
1	2015-01-02	27,02	24,5	33,5	0	0.0	28.972
2	2015-01-03	24,82	22,4	29,9	0	1.0	30.814
3	2015-01-04	23,98	21,5	28,6	1,2	1.0	29.799
4	2015-01-05	23,82	21	28,3	0	0.0	28.900

Data Pre-processing:

In [2]:
```
df1.shape
```

Replace ',' with '.' in columns 'Temperatura Media (C)', 'Temperatura Minima (C)', 'Temperatura Maxima (C)', and 'Precipitacao (mm)'

In [3]:
```
df1['Temperatura Media (C)']=df1['Temperatura Media (C)'].str.replace(',','.')
df1['Temperatura Minima (C)'] = df1['Temperatura Minima (C)'].str.replace(',','.')
df1['Temperatura Maxima (C)'] = df1['Temperatura Maxima (C)'].str.replace(',','.')
df1['Precipitacao (mm)'] = df1['Precipitacao (mm)'].str.replace(',','.')
```

In [4]:
```
df1.head()
```

Create new feature 'Month' from the dates, consisting of the month of the year Create new feature 'Day' from the dates, consisting of the day of the week Set values from 'Data' column as indexes

In [5]:
```
df1['Month']=df1.Data.dt.month
df1['day']=df1.Data.dt.dayofweek
df1.iloc[335:341]
```

In [6]:
```
df1.set_index('Data',inplace=True)
```

Only drop those instances where all values are null Also, check the duplicate value

In [7]:
```
print(df1.isnull().sum())
print(df1.shape)
```

In [8]:
```
print(df1.isnull().all(axis=1).sum()) # calculate the number of rows which have null values in all columns
```

We can see that the 576 instances have all null values in all columns. So we can easily drop those instances

.0]:
```
df1.dropna(how='all',inplace=True)
```

.1]:
```
df1.shape
```

.2]:
```
print(df1.isnull().sum())
```

.3]:
```
if df1.duplicated().any():
 print('True: duplicate instances')
else:
 print('False: No duplicate instances')
```

Check the data-types of the features Convert them to appropriate data types

.4]:
```
df1.info()
```

The columns with dtype object will be needed to be converted to appropriate dtype

.5]:
```
df1['Temperatura Media (C)']=df1[['Temperatura Media (C)']].astype(float)
df1['Temperatura Minima (C)'] = df1['Temperatura Minima (C)'].astype(float)
df1['Temperatura Maxima (C)'] = df1['Temperatura Maxima (C)'].astype(float)
df1['Precipitacao (mm)'] = df1['Precipitacao (mm)'].astype(float)
# Final de semana is a categorical column(like yes or no) so it should be int, not float
df1['Final de Semana'] = df1['Final de Semana'].astype(int)
```

.6]:
```
df1.info()
```

Data Model and Evaluation :

Split the dataset using sklearn, with 20% for testing with random_state=7

```
from sklearn.model_selection import train_test_split
X= df1.drop(columns=['Consumo de cerveja (litros)'],axis=1)
y= df1['Consumo de cerveja (litros)']
X_train,X_test,y_train,y_test=train_test_split(X, y,test_size=0.20,
random_state = 7)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
print('The final coefficients after training is:',lr.coef_)
print('The final intercept after training is:',lr.intercept_)
```

```
from sklearn.metrics import r2_score,mean_squared_error, mean_absolute_error
y_pred = lr.predict(X_test)
print("r2 score of our model is:", r2_score(y_test,y_pred))
print("mean absolute error of our model is:", mean_absolute_error(y_test,y_pred))
print("root mean squared error of our model is:", mean_squared_error(y_test,y_pred,squared=False))
```

Save the Model :

In [26]:
```
import pickle
```

In [27]:
```
with open('beer_pred_model.pkl', 'wb') as f:
    pickle.dump(lr, f)
```

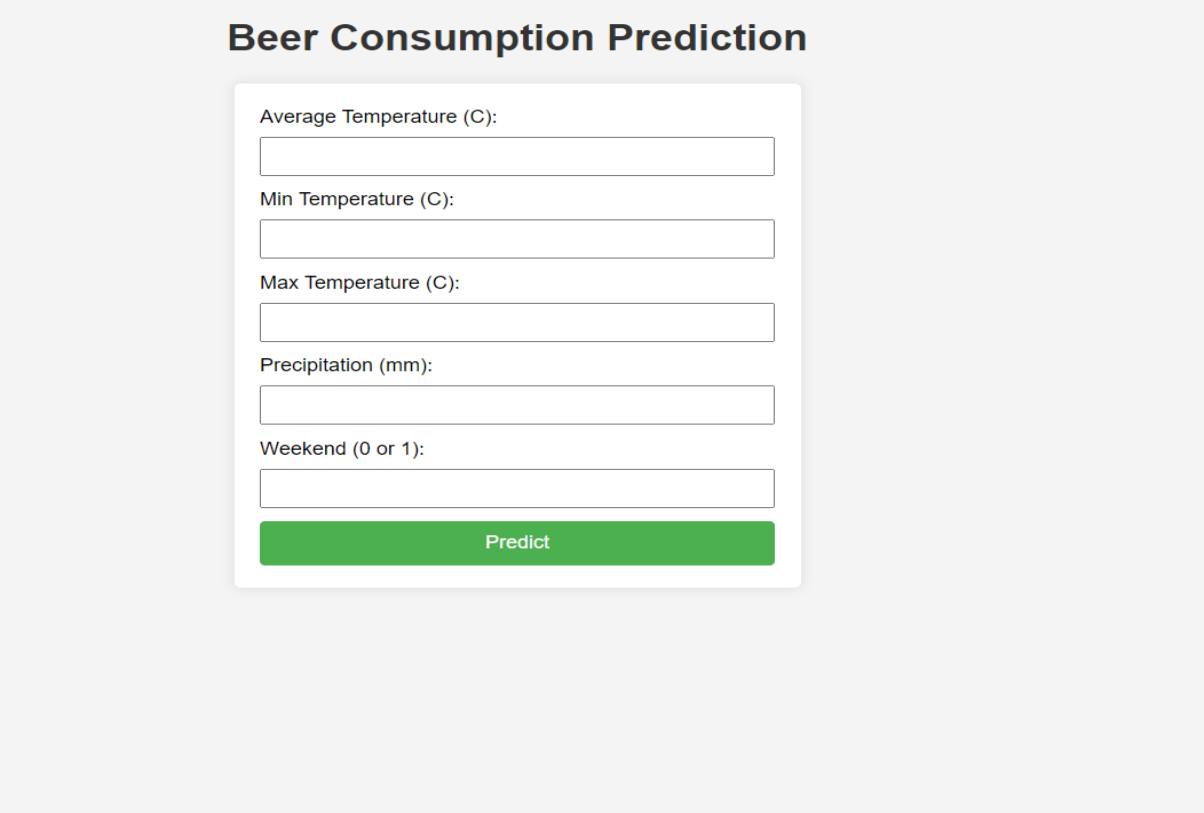
App.py:

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

app = Flask(__name__, template_folder='templates')

# Load the model
with open('beer_pred_model.pkl', 'rb') as f:
    model = pickle.load(f)

@app.route('/')
def home():
    return render_template('index.html')

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

    predicted_beer_consumption = prediction[0]

    return render_template('index.html', prediction_text=f"Predicted beer consumption: {predicted_beer_consumption:.2f} liters")

if __name__ == "__main__":
    app.run(debug=True, use_reloader=False)
```

HTML template – index.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Beer Consumption Prediction</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 20px;
        }
        h1 {
            text-align: center;
            color: #333;
        }
        form {
            max-width: 400px;
            margin: 0 auto;
            background-color: #fff;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        label {
            display: block;
            margin-bottom: 8px;
        }
        input {
            width: 100%;
            padding: 8px;
            margin-bottom: 12px;
            box-sizing: border-box;
        }
          button {
              background-color: #4caf50;
              color: white;
              padding: 10px 15px;
              border: none;
              border-radius: 4px;
              cursor: pointer;
              width: 100%;
              font-size: 16px;
          }
          button:hover {
              background-color: #45a049;
          }
    </style>
</head>
<body>
    <h1>Beer Consumption Prediction</h1>
    <form method="post" action="/predict">
        <label for="TemperatureAvg">Average Temperature (C):</label>
        <input type="text" name="TemperatureAvg" required><br>

        <label for="TemperatureMin">Min Temperature (C):</label>
        <input type="text" name="TemperatureMin" required><br>

        <label for="TemperatureMax">Max Temperature (C):</label>
        <input type="text" name="TemperatureMax" required><br>

        <label for="Precipitation">Precipitation (mm):</label>
        <input type="text" name="Precipitation" required><br>

        <label for="Weekend">Weekend (0 or 1):</label>
        <input type="text" name="Weekend" required><br>

        <button type="submit">Predict</button>
    </form>
</body>
</html>
```

HTML visuals:



Model Deployment using Heroku:

5. Model deployment using Heroku

We're ready to start our Heroku deployment now that our model has been trained, the machine learning pipeline has been set up, and the application has been tested locally. There are a few ways to upload the application source code onto Heroku. The easiest way is to link a GitHub repository to your Heroku account.

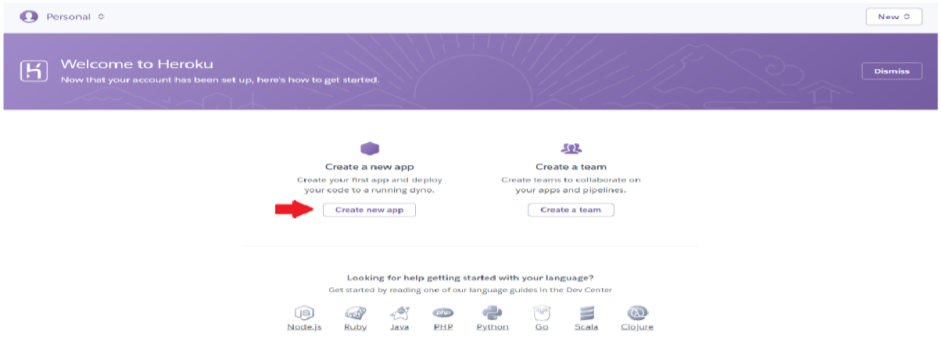
Requirement.txt

It is a text file containing the python packages required to execute the application.

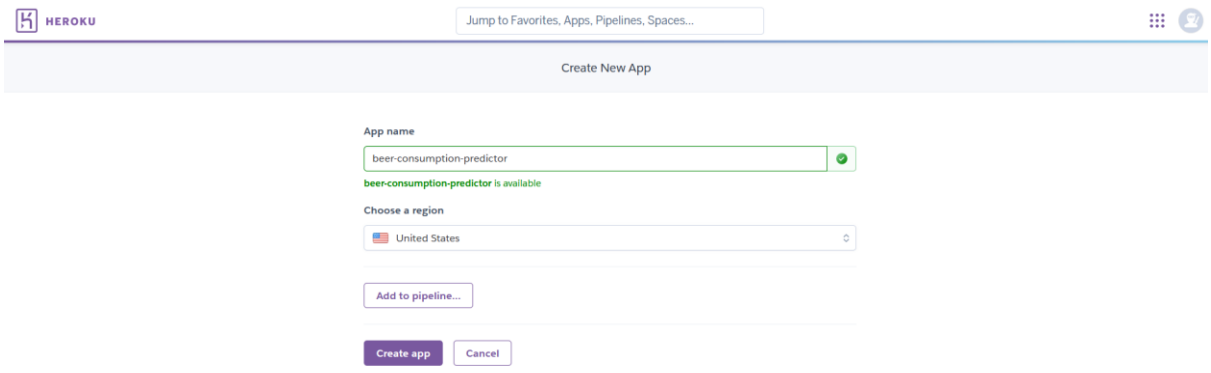
5.1 Steps for Model Deployment Using Heroku

Once we uploaded files to the GitHub repository, we are now ready to start deployment on Heroku. Follow the steps below:

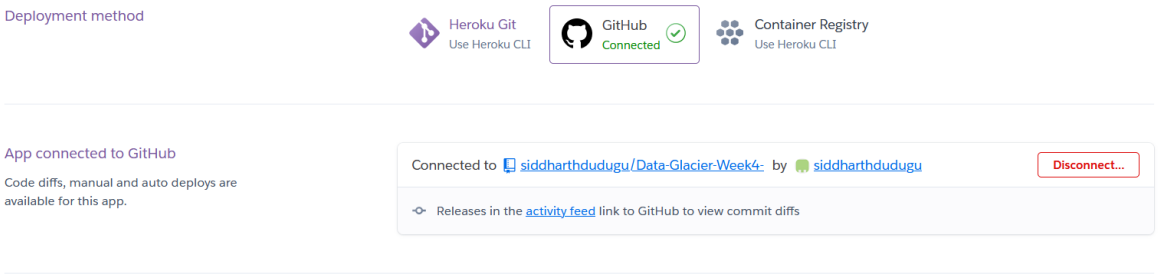
1. After sign up on **heroku.com** then click on **Create new app.**



Naming the APP:



Connecting to GITHUB:



Deployment: