

Name:Efe KARA
Batch code:LISUM16
Submission date:23/12/2022
Submitted to:Data Glacier

1.Selected dataset: <https://www.kaggle.com/datasets/ashydv/advertising-dataset>

Description: Relationship of the impact based on platforms(TV,Radio,Newspapers) advertising on sales. Which platform is best for advertising.

2. The file named regression_model.pkl

```
import pickle  
import numpy as np  
import pandas as pd  
from sklearn.metrics import mean_squared_error  
  
df =  
pd.read_csv("/home/efe/PycharmProjects/pythonProject/flask/data/Advertising.csv",  
index_col=0)  
  
from sklearn.linear_model import LinearRegression  
  
X = df.drop('sales', axis=1)  
y = df[["sales"]]  
  
reg_model = LinearRegression()  
reg_model.fit(X, y)  
  
y_pred = reg_model.predict(X)  
rmse = np.sqrt(mean_squared_error(y, y_pred))  
print("RMSE:", rmse)  
pickle.dump(reg_model, open('regression_model.pkl', 'wb'))  
  
print("Model Saved")
```

3.The file named “app.py” run from under the directory that contains template.html open in terminal, by `python app.py` command this will show you the local API interface. Additionally of course html template running on background.

```
import numpy as np  
from flask import Flask, request, render_template  
import pickle  
  
app = Flask(__name__, template_folder="templates") # initialize the flask app.  
starts.  
  
model = pickle.load(open('regression_model.pkl', 'rb')) # recall saved templates  
  
# Defining the homepage for flask root path
```

```

@app.route('/')
def home():
    return render_template('templates.html') # defining the homepage for
templates

# After the above section, there is interaction in templates.html.
# This interaction is maintained by the following sections.

@app.route('/predict', methods=['POST'])
def predict():
    # It continues to create again after calculating by taking the values entered the
texts in the interface.
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)
    predicted_y = int(np.round(prediction, 2))

    # return the output back
    return render_template('templates.html', prediction_text='Predicted Sales:
{}'.format(predicted_y))

if __name__ == "__main__":
    app.run(port=9997)

```

This was app.py file.



ML API x +

← → ↻ ⓘ 127.0.0.1:9997/predict

Sales Prediction Model Deployment APP

Predicted Sales: 20

This is a simple prediction based on trained model.

HTML running on the background

```
<!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>Sales Prediction Model
Deployment APP</h1> <!-- section for taking inputs for model --> <form
action="{{ url_for('predict') }}"method="post"> <input type="text" name="tv"
placeholder="TV (0-10)" required="required" /> <input type="text" name="radio"
placeholder="Radio" required="required" /><input type="text" name="newspaper"
placeholder="Newspaper" required="required" /> <button type="submit" class="btn btn-
primary btn-block btn-large">Predict</button> </form> <br> <br> {{ prediction_text }}
</div></body></html>
```

This was template.html file.

