

DATA GLACIER INTERNSHIP

WEEK 4: DEPLOYMENT ON FLASK

Name: Nrusimha Saraswati Sai Teja Jampani

Batch: LISUM27

Submission Date:

Submitted To: Data Glacier

Abstract:

We have developed an ML application to predict the price of used cars across the US based on the location, year and number of miles run by the car. The model is then deployed with the use of Flask and along with appropriate HTML and CSS scripts.

Dataset:

The dataset we have utilized is 'used car listing' available on Kaggle. The data is preprocessed by removing unwanted columns and the final dataset contains 5 predictors and a target column as given below.

Column	Description
Year	The year in which the car is manufactured
Mileage	The number of miles run by the car
City	The city to which the car belongs
State	The State to which the car belongs
Make	The Manufacturer of the car
Price	The Price of the car

The data intake report of the dataset is given below.

Tabular data details: Used_Cars

Total number of observations	852122
Total number of files	1
Total number of features	5
Base format of the file	csv
Size of the data	33 MB

Model:

We will be fitting a linear regression model to predict the price of the used car by using the other fields i.e., Year, Mileage, City, State and Make as predictor variables. Categorical variables City, State and Make are converted to numeric using label encoder from scikit learn.

```
le = LabelEncoder()
city_label = le.fit_transform(used_cars['City'])
state_label = le.fit_transform(used_cars['State'])
make_label = le.fit_transform(used_cars['Make'])

used_cars.drop("City", axis=1, inplace=True)
used_cars.drop("State", axis=1, inplace=True)
used_cars.drop("Make", axis=1, inplace=True)

used_cars['City'] = city_label
used_cars['State'] = state_label
used_cars['Make'] = make_label

X = used_cars.drop(columns = ['Price'])
y = used_cars['Price']
model = LinearRegression().fit(X, y)
```

We will then save our model as well as the label encoded data as a pickle file to be used in our flask program.

```
model = pickle.dump( model, open( "model.p", 'wb' ) )
```

```
city_label_encoder = pickle.dump( city_label, open( "city_label.p", 'wb' ) )
```

```
state_label_encoder=pickle.dump( state_label, open( "state_label.p", 'wb' ) )
```

```
make_label_encoder=pickle.dump( make_label, open( "make_label.p", 'wb' ) )
```

Deployment:

In our HTML script we will be using 5 text boxes for inputting the predictor values i.e., Year, Mileage, State, City and Maker. We also use a submit button to output the prediction. A text field will be dynamically filled from the flask program to output the prediction value.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Input Form</title>
  <link rel="stylesheet" href="{{ url_for('static', filename = 'css/styles.css')}}">
</head>
<body>
  <div class = 'login'>
    <div class="header">
      
      <h1>Used Car Price Predictor</h1>
    </div>
    <form action="/predict" method="post">
      <label for="Year">Year:</label>
      <input type="text" id="Year" name="Year" required><br>

      <label for="Milesrun">Milesrun:</label>
      <input type="text" id="Milesrun" name="Milesrun" required><br>

      <label for="City">City:</label>
      <input type="text" id="City" name="City" required><br>

      <label for="State">State:</label>
      <input type="text" id="State" name="State" required><br>

      <label for="Maker">Maker:</label>
      <input type="text" id="Maker" name="Maker" required><br>

      <button type="submit">Predict Price</button>
    </form>
    <br>
    <br>
    <div id = 'pred_text'>{{ pred_text }}</div>
  </div>
</body>
</html>
```

We have also used a CSS script to further format our deployment front end application. I have added an image and enhanced the aesthetics. The

script is as shown below. HTML script is placed in templates folder and CSS script is placed in static folder respectively.

```
body {
    margin: 0;
    padding: 0;
    font-family: 'Arial', sans-serif;
    background-color: #f4f4f4;
}

.login {
    max-width: 400px;
    margin: 50px auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.header {
    text-align: center;
    margin-bottom: 20px;
}

.car-photo {
    max-width: 100%;
    height: auto;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

form {
    text-align: center;
}

label {
    display: block;
    margin-bottom: 8px;
}

input {
    width: calc(100% - 16px);
    padding: 8px;
    margin-bottom: 16px;
    box-sizing: border-box;
    border: 1px solid #ddd;
    border-radius: 4px;
}

button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}

#pred_text {
    margin-top: 20px;
    text-align: center;
    font-weight: bold;
    color: #333;
}
```

In the flask application, we have first rendered the input html script to the server. Then we loaded the necessary pickle files and using the label

encoder parsed the form data to convert the categorical variables to numerical before passing to the model. The predictor data is then passed to the model to make a price prediction. This output is passed back to the front-end application to show the predicted used car price.

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

app = Flask(__name__)


# Load pickle files for the model as well as the labelencoder objects for City, State and maker
model = pickle.load(open('model.p', 'rb'))
city_label = pickle.load(open('city_label.p', 'rb'))
state_label = pickle.load(open('state_label.p', 'rb'))
make_label = pickle.load(open('make_label.p', 'rb'))

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

@app.route('/predict', methods=['POST'])
def predict():
    input_data = [x for x in request.form.values()]
    city = city_label[input_data[2]]
    state = state_label[f'{input_data[3]}']
    maker = make_label[input_data[4]]
    input_data[2] = city
    input_data[3] = state
    input_data[4] = maker
    input_data = [int(x) for x in input_data]
    input_data = [np.array(input_data)]
    prediction = model.predict(input_data)
    prediction = round(prediction[0], 2)
    return render_template('input.html', pred_text = f'The Car should be priced: {prediction} $')

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

Results:



Used Car Price Predictor


Year:

Milesrun:

City:

State:

Maker:



Used Car Price Predictor


Year:

Milesrun:

City:

State:

Maker:



Used Car Price Predictor

Year:

Milesrun:

City:

State:

Maker:

The Car should be priced: 25054.29 \$