User Testing Data Dashboard

DS Bootcamp: Advanced Database

Let’s create a dashboard for user

What to do:

- Create a new form for user data input
- Create a Java Script to control input data and generate chart
- Create a view to load model and test the data
- Modify dashboard.html
- Update urls

Add new form in forms.py

```
class CustomerPredictionForm(forms.Form):
    store_id = forms.IntegerField(label='Store ID', min_value=1)
    active = forms.IntegerField(label='Active (0 or 1)', min_value=0, max_value=1)
    total_payment = forms.FloatField(label='Total Payment', min_value=0)
    payment_count = forms.IntegerField(label='Payment Count', min_value=0)
    average_payment = forms.FloatField(label='Average Payment', min_value=0)
```

Create a new Java Script (prediction_form.js)

Create data based on user input in the form

```
document.getElementById('predictionForm').addEventListener('submit', function(e) {
  e.preventDefault();

  const data = {
    store_id: parseInt(document.getElementById('id_store_id').value),
    active: parseInt(document.getElementById('id_active').value),
    total_payment: parseFloat(document.getElementById('id_total_payment').value),
    payment_count: parseInt(document.getElementById('id_payment_count').value),
    average_payment: parseFloat(document.getElementById('id_average_payment').value)
  };
```

Create a new Java Script (prediction_form.js)

Send the data to the predict_customer view (see topic 14)

```
fetch('/predict_customer/', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'X-CSRFToken': getCookie('csrftoken')  // Optional if CSRF is enforced
  },
  body: JSON.stringify(data)
})
```

Create a new Java Script (prediction_form.js)

Prepare the response in json format to generate the chart

```
  .then(response => response.json())
  .then(result => {
    console.log("Prediction result:", result);
    document.getElementById('statusBox').innerText =
      `Prediction: ${result.prediction}, Probabilities: ${result.probability.map(p => p.toFixed(2)).join(', ')}`;
    updateChart(result.probability);
  })
  .catch(err => {
    console.error('Prediction error:', err);
    document.getElementById('statusBox').innerText = 'Error making prediction.';
  });
});
```

Create a new Java Script (prediction_form.js)

```
function getCookie(name) {
  let cookieValue = null;
  if (document.cookie && document.cookie !== '') {
    const cookies = document.cookie.split(';');
    for (let cookie of cookies) {
      cookie = cookie.trim();
      if (cookie.startsWith(name + '=')) {
        cookieValue = decodeURIComponent(cookie.substring(name.length + 1));
        break;
      }
    }
  }
  return cookieValue;
}
```

Create a new Java Script (prediction_form.js)

```
// Chart.js
let predictionChart = null;
function updateChart(probabilities) {
  const ctx = document.getElementById('predictionChart').getContext('2d');
  console.log("Canvas context:", ctx);
  const labels = probabilities.map((_, index) => `Class ${index}`);
  const data = {
    labels: labels,
    datasets: [{
      label: 'Probability',
      data: probabilities,
      backgroundColor: ['rgba(54, 162, 235, 0.7)', 'rgba(255, 99, 132, 0.7)'],
      borderColor: ['rgba(54, 162, 235, 1)', 'rgba(255, 99, 132, 1)'],
      borderWidth: 1
    }]
  };
```

Create a new Java Script (prediction_form.js)

```
const config = {
  type: 'bar',
  data: data,
  options: {
    scales: {
      y: { beginAtZero: true, max: 1 }
    }
  }
};

if (predictionChart) {
  predictionChart.data = data;
  predictionChart.update();
} else {
  predictionChart = new Chart(ctx, config);
}
}
```

Update and check again predict_customer view

```python
model_path = os.path.join(settings.BASE_DIR, 'final_customer_model.pkl')
model = joblib.load(model_path)

@csrf_exempt
def predict_customer(request):
    print(f"Request method: {request.method}")
    if request.method == "POST":
        # Parse incoming JSON data
        data = json.loads(request.body)
        print(f"Data received: {data}")

        # Prepare feature array (ensure correct feature order)
        features = np.array([
            data["store_id"],
            data["active"],
            data["total_payment"],
            data["payment_count"],
            data["average_payment"]
        ]).reshape(1, -1)
```

Update and check again predict_customer view

```
    # Make prediction and calculate probability
    prediction = model.predict(features)[0]
    probability = model.predict_proba(features)[0].tolist()

    # Return prediction and probability as JSON response
    return JsonResponse({
        "prediction": int(prediction),
        "probability": probability
    })
```

Update and check again customer_prediction_view

Make sure that now this view also include form control from the dashboard

```
def customer_prediction_view(request):
    form = CustomerPredictionForm()
    return render(request, 'dvdrental_prediction/dashboard.html', {'form': form})
```

Modify the template (dashboard.html)

Add form to the dashboard

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Customer Rental Prediction</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <h2>Customer Rental Prediction</h2>

  <!-- Prediction Form -->
  <form id="predictionForm">
  {{ form.as_p }}

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

Modify the template (dashboard.html)

Change the js source from predict.js (in topic 14) to prediction_form.js

```
  <!-- Prediction Result Display -->
  <div id="statusBox">Waiting for predictions...</div>

  <!-- Chart Display -->
  <canvas id="predictionChart" width="600" height="300"></canvas>

  <script>
    const predictUrl = "{% url 'predict_customer' %}";
  </script>
  <script src="{% static 'dvdrental_prediction/js/prediction_form.js' %}"></script>

</body>
</html>
```

Check again the urls

Make sure the path is correct

```
path('predict_customer/', views.predict_customer, name='predict_customer'),
path('', views.customer_prediction_view, name='dashboard'),
```

Run the server and Test

Customer Rental Prediction

Store ID: 1

Active (0 or 1): 1

Total Payment: 2000

Payment Count: 10

Average Payment: 100

Predict

Prediction: 1, Probabilities: 0.05, 0.95