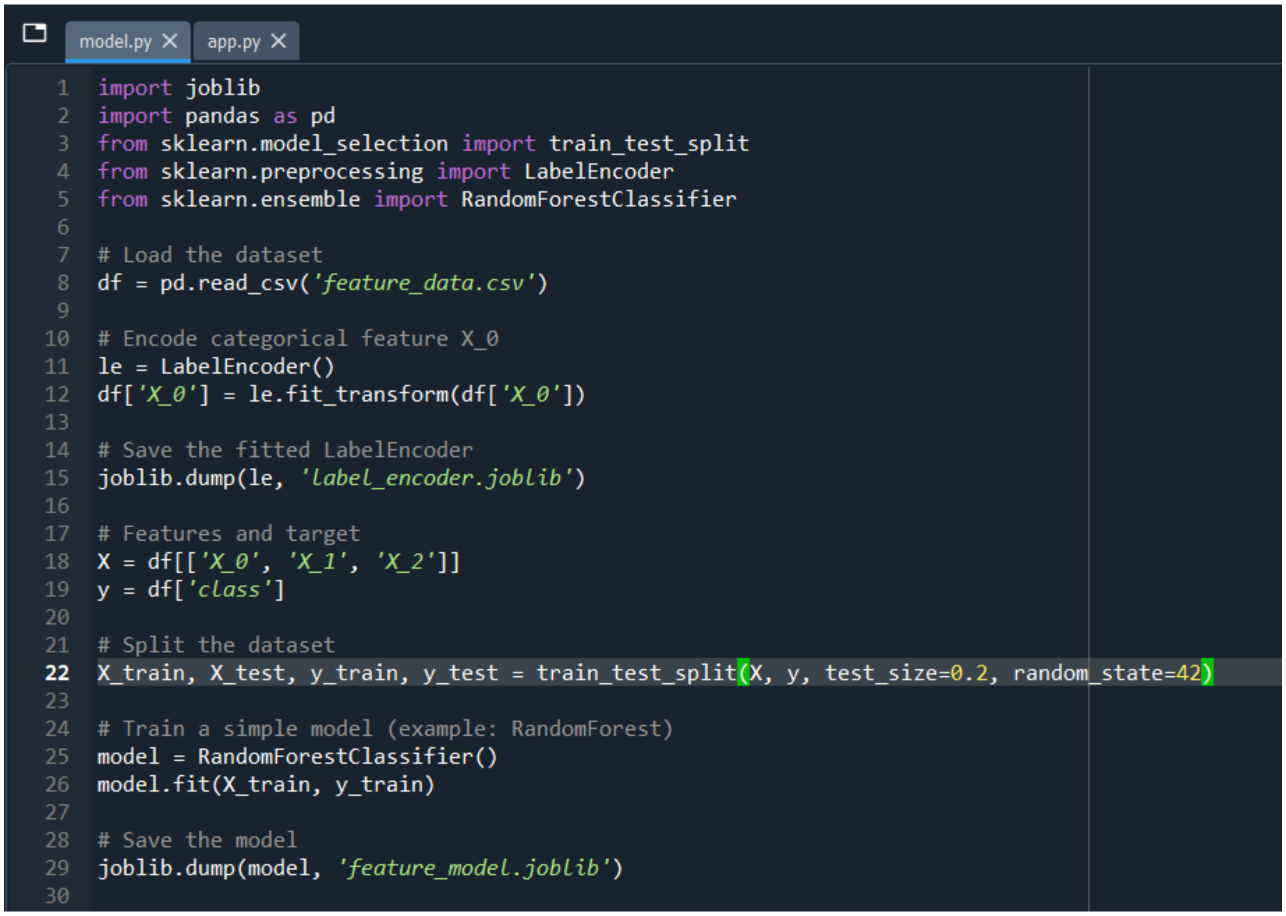
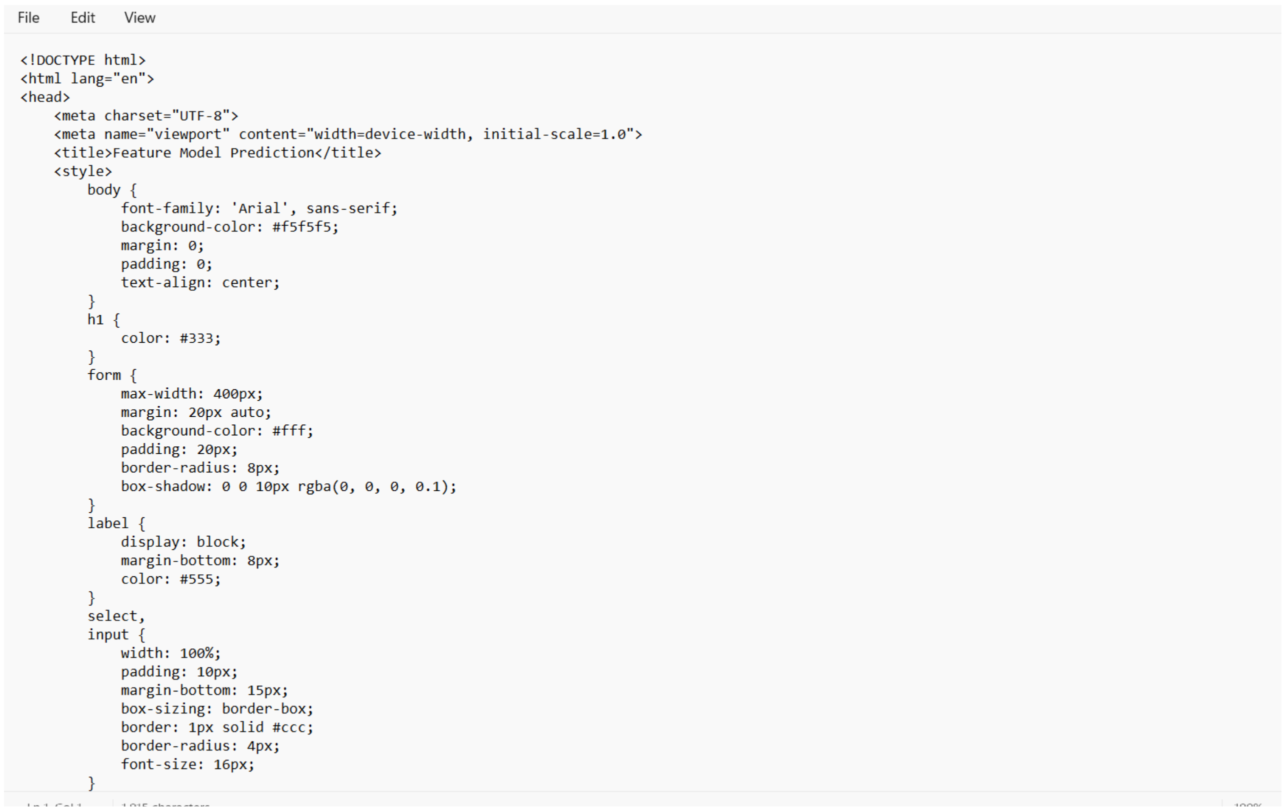
Name: Ayush Jha
Internship Batch: LISUM30
Submitted by: Ayush Jha
Submitted to: Data Glacier

Model Creation:



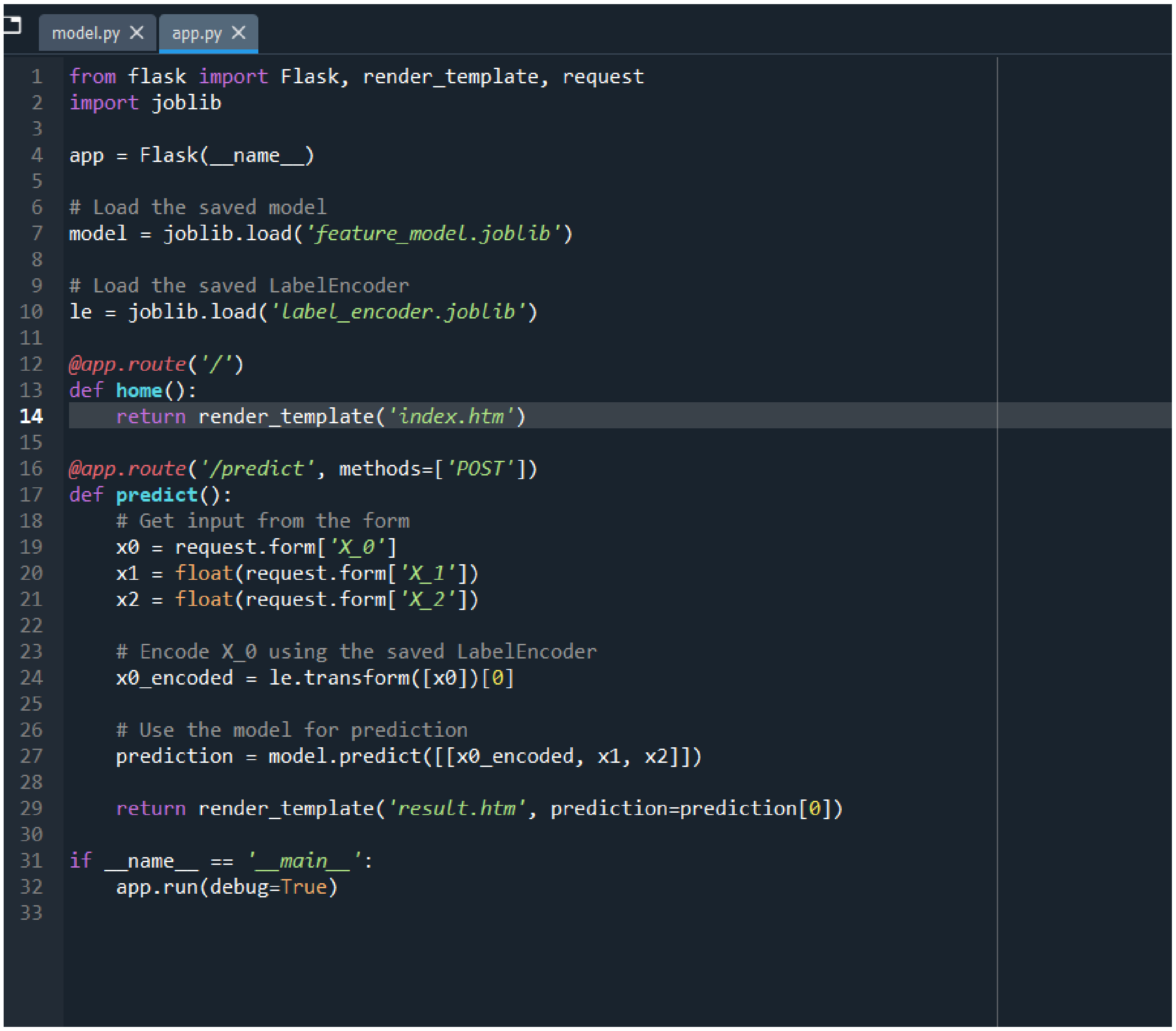
```python
import joblib
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier

# Load the dataset
df = pd.read_csv('feature_data.csv')

# Encode categorical feature X_0
le = LabelEncoder()
df['X_0'] = le.fit_transform(df['X_0'])

# Save the fitted LabelEncoder
joblib.dump(le, 'label_encoder.joblib')

# Features and target
X = df[['X_0', 'X_1', 'X_2']]
y = df['class']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a simple model (example: RandomForest)
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Save the model
joblib.dump(model, 'feature_model.joblib')

```

HTML Creation:



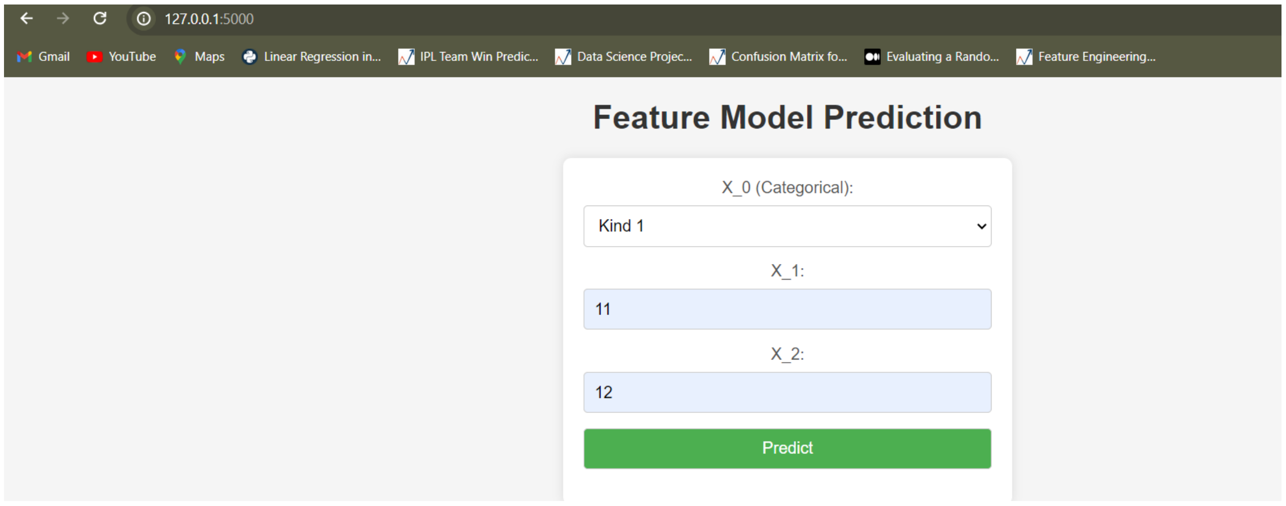
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Feature Model Prediction</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f5f5f5;
            margin: 0;
            padding: 0;
            text-align: center;
        }
        h1 {
            color: #333;
        }
        form {
            max-width: 400px;
            margin: 20px auto;
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        label {
            display: block;
            margin-bottom: 8px;
            color: #555;
        }
        select,
        input {
            width: 100%;
            padding: 10px;
            margin-bottom: 15px;
            box-sizing: border-box;
            border: 1px solid #ccc;
            border-radius: 4px;
            font-size: 16px;
        }
```

App creation



```
from flask import Flask, render_template, request
import joblib

app = Flask(__name__)

# Load the saved model
model = joblib.load('feature_model.joblib')

# Load the saved LabelEncoder
le = joblib.load('label_encoder.joblib')

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

@app.route('/predict', methods=['POST'])
def predict():
    # Get input from the form
    x0 = request.form['X_0']
    x1 = float(request.form['X_1'])
    x2 = float(request.form['X_2'])

    # Encode X_0 using the saved LabelEncoder
    x0_encoded = le.transform([x0])[0]

    # Use the model for prediction
    prediction = model.predict([[x0_encoded, x1, x2]])

    return render_template('result.htm', prediction=prediction[0])

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

Result:

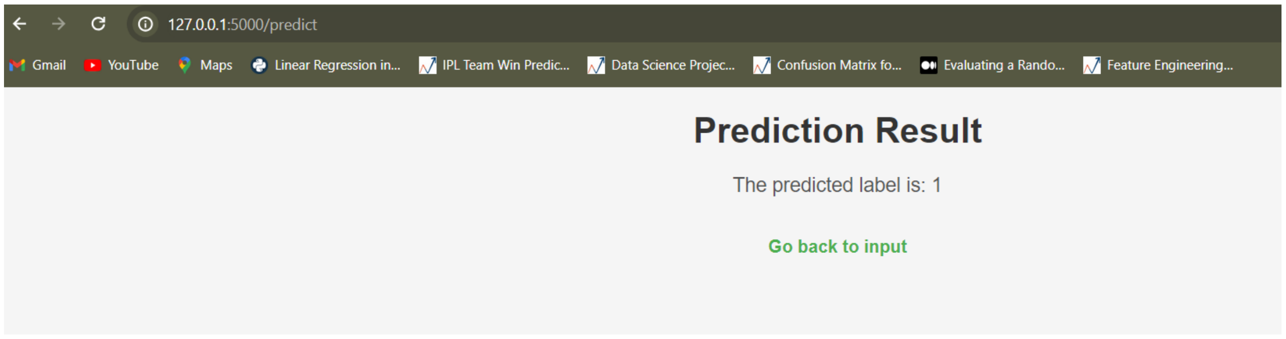


Feature Model Prediction

X_0 (Categorical): Kind 1

X_1: 11

X_2: 12



Prediction Result

The predicted label is: 1

Go back to input