**Diabetes Detection Using Machine Learning**

**Team Members**

Fawad Arabzada ▪ Maryam Timorian ▪ Zabehullah Nasiri

**Deployment**

**1. Overview**

The deployment phase of the diabetes prediction model involves making the trained machine learning model available for real-world use. The model is integrated into a web application, allowing users to input relevant medical data and receive predictions on diabetes risk. The deployment pipeline includes model serialization, API integration, security measures, and monitoring.

**2. Model Serialization**

The trained model is serialized using **joblib** to ensure efficient storage and fast loading. The model is saved as a .pkl file, which preserves the learned parameters for later use.

python

CopyEdit

import joblib

# Save the trained model

joblib.dump(model, 'diabetes\_model.pkl')

# Load the model for deployment

loaded\_model = joblib.load('diabetes\_model.pkl')

**3. Model Serving**

The serialized model is deployed using **Flask**, a lightweight web framework in Python. This allows users to send their input data and receive predictions via a simple web interface. The deployment can be done on **Heroku, Render, or an AWS EC2 instance**.

**4. API Integration**

A REST API is created to serve the model, allowing users to send POST requests with medical data and receive a diabetes prediction. The API has an endpoint /predict, which takes JSON input and returns a prediction.

Example Flask API:

python

CopyEdit

from flask import Flask, request, jsonify

import joblib

import numpy as np

app = Flask(\_\_name\_\_)

# Load the model

model = joblib.load("diabetes\_model.pkl")

@app.route('/predict', methods=['POST'])

def predict():

data = request.get\_json()

input\_features = np.array(data['features']).reshape(1, -1)

prediction = model.predict(input\_features)

return jsonify({'prediction': int(prediction[0])})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**5. Security Considerations**

* **Data Encryption:** User input is secured using **HTTPS** to prevent unauthorized access.
* **Authentication & Authorization:** API endpoints require an authentication token to prevent misuse.
* **Cross-Origin Resource Sharing (CORS):** Implemented to control access to the API from specific frontend applications.

**6. Monitoring and Logging**

To ensure reliable performance, the deployed model is monitored using **logging and performance tracking tools** such as:

* **Prometheus & Grafana:** For real-time monitoring of API response times.
* **Logging (Python logging module):** Tracks API requests and model performance.
* **Alerts & Retraining:** If accuracy drops significantly, alerts notify the development team to retrain the model.