Machine Learning Model Deployment Assignment

Objective

The primary goal of this assignment is to provide you with hands-on experience in deploying a machine learning model to a cloud environment. By completing this project, you will have successfully deployed a functional, scalable, and accessible machine learning model via a REST API. This project will challenge you to think about the entire machine learning lifecycle, from data to production.

Project Overview

You are tasked with selecting a compelling machine learning problem, developing a predictive model, and deploying it on a major cloud platform such as **Amazon Web Services (AWS)**, **Google Cloud Platform (GCP)**, or **Microsoft Azure**. Your deployed model must be able to receive input data through an API endpoint and return real-time predictions. The focus of this assignment is on the **deployment and operationalization** of the model, not just its accuracy.

Assignment Steps

Phase 1: Model Development & Preparation

1. **Problem & Data Selection**:
    - Choose a dataset and a corresponding machine learning problem that interests you. You can find datasets on platforms like Kaggle, UCI Machine Learning Repository, or data.gov.
    - Perform thorough exploratory data analysis (EDA) to understand the nuances of your data.
2. **Data Preprocessing**:
    - Develop a robust preprocessing pipeline. This should handle tasks such as cleaning missing values, feature scaling, and encoding categorical variables. Your preprocessing steps should be saved and applied consistently to new data at inference time.
3. **Model Training & Selection**:
    - Train at least two different types of machine learning models using a framework like Scikit-Learn, TensorFlow, or PyTorch.
    - Evaluate your models using appropriate metrics (e.g., accuracy, precision, recall, F1-score, MAE, R^2).
    - Select your best-performing model and save the trained model artifact (e.g., using pickle or joblib) along with your preprocessing pipeline.

Phase 2: Deployment & API Creation

1. **Cloud Environment Setup**:
    - Create an account on your chosen cloud platform (AWS, GCP, or Azure). Many platforms offer free-tier services that should be sufficient for this assignment.
    - Set up a new project or resource group for this assignment.
2. **Containerization with Docker**:
    - Write a Dockerfile to containerize your machine learning application. This container should include your trained model, preprocessing pipeline, and the necessary application code to serve predictions.
    - Your application should be built using a web framework like **FastAPI** or **Flask**.
    - Test your Docker container locally to ensure it runs correctly.
3. **Cloud Deployment**:
    - Deploy your containerized application to a cloud service. Some recommended services include:
        - **AWS**: Elastic Beanstalk, Amazon SageMaker, or Elastic Kubernetes Service (EKS) for a more advanced option.
        - **GCP**: Cloud Run, Google AI Platform, or Google Kubernetes Engine (GKE).
        - **Azure**: Azure App Service, Azure Machine Learning, or Azure Kubernetes Service (AKS).
    - Configure the deployment to be scalable and resilient.
4. **API Endpoint Configuration**:
    - Expose a secure API endpoint that can receive POST requests with input data in JSON format and return model predictions.
    - Implement basic API key authentication to secure your endpoint.

Final Report & Submission:

- Prepare a comprehensive report detailing your project.
- Create a README.md file in your code repository that provides clear instructions on how to set up and use your project.

Deliverables

1. **Project Report (PDF)**: A detailed report **WITH SCREENSHOTS** that includes:
    - An **Executive Summary** of your project.
    - A description of the **Machine Learning Problem** and the dataset used.
    - An overview of your **Model Development Process**, including preprocessing, training, and evaluation.
    - A detailed **Deployment Architecture Diagram** and an explanation of the cloud services you used.
    - A section on **Challenges Faced** and how you overcame them.

- A **Conclusion** summarizing your work and suggesting future improvements.
- **Note: your report MUST include screenshots of your working deployment for full marks.**

2. **Code Repository**: A link to a private GitHub or GitLab repository containing:
   - All **Jupyter notebooks** or Python scripts used for data analysis and model training.
   - The final trained **model artifact** and preprocessing pipeline.
   - Your **application code** (e.g., main.py for FastAPI/Flask).
   - The Dockerfile and any other configuration files (e.g., requirements.txt).
   - A comprehensive README.md file.

3. **Live API Endpoint**: A URL to your deployed and functional API endpoint, along with any necessary API keys for testing.

Restrictions

- You are **not permitted** to use GUI-based model interaction tools like Streamlit or Gradio for the core API. The model must be accessible via a direct API call.
- Your model must be **pre-trained and loaded** at inference time. You are not allowed to train your model (model.fit()) for each API request. The model should be unpickled (or loaded) when the application starts or on the first request.
  - **Update (06/15/2025):** You may train your model upon server start, rather than pulling it from a registry. Marks will be deducted for solutions that train during inference time, however.
- While you can reuse a model from a previous project, the primary focus and effort should be on the deployment architecture and process.

Marking Scheme

Component	Marks	Description
Model Development & Quality	20	The quality of the data preprocessing, model selection, and overall model performance.
Containerization & API	30	The correctness of the

		Dockerfile, the functionality of the FastAPI/Flask application, and the proper implementation of the API endpoint.
Cloud Deployment	30	The successful deployment to a cloud platform, the choice of appropriate services, and the stability of the deployed application.
Project Report & Code	20	The clarity and completeness of the final report, the