**Machine Learning Model Deployment Report**

1. **Introduction**

**Project Objective**

This project entails the deployment of an advanced **Titanic Survival Prediction Model** as a RESTful API on **Azure Cloud Services**, enabling real-time inference on passenger survival likelihood. By leveraging contemporary methodologies in data science and software engineering, it integrates model training, containerization, and cloud-based hosting to furnish a scalable solution for predictions.

**Data Used**

* **Dataset Source:** The Titanic Dataset from Kaggle
* **Target Variable:** Survived (Binary classification: 0 or 1)
* **Features:**
  + **Numerical:** Age, Fare, SibSp, and Parch
  + **Categorical:** Sex, Pclass, and Embarked
* **Data Preprocessing:** Comprehensive procedures were undertaken, including the resolution of missing values, the transformation and encoding of categorical attributes, and the normalization of pertinent numerical features. The dataset was partitioned on an 80/20 basis for training and testing, respectively, to ensure robust performance estimates.

**Model Developed**

* **Algorithms:**
  + Random Forest (Baseline)
  + **XGBoost (Deployed Model)** employing the parameters n\_estimators=300, max\_depth=7, and learning\_rate=0.05
* **Performance Comparison:**

|  |  |  |
| --- | --- | --- |
| Metric | Random Forest | XGBoost |
| Accuracy | 80.4% | **83.2%** |
| Precision | 80.9% | **81.4%** |
| Recall | 68.9% | **77.0%** |
| F1 Score | 74.4% | **79.2%** |

Given XGBoost’s enhanced accuracy and recall, the final model was serialized as xgb\_titanic\_model.pkl for deployment.

1. **Deployment Overview**

**Architecture**

**Deployment Steps (Azure CLI)**

1. **Build Docker Image Locally**
   * cd /path/to/project
   * docker build -t titanic-api:v1 .
2. **Push Image to ACR**
   * az login
   * az acr login --name titanicacr
   * docker tag titanic-api:v1 titanicacr.azurecr.io/titanic-api:v1
   * docker push titanicacr.azurecr.io/titanic-api:v1
3. **Create & Configure Azure App Service** 
   * az webapp create --resource-group titanic-rg --plan titanic-app-plan \
4. **Code and Resources**

**Data Preprocessing & Model Training**

* **train\_model.py**: Demonstrates ingestion of the Titanic dataset, cleansing measures, encoding, XGBoost training, and final pickling to xgb\_titanic\_model.pkl.

**API & Deployment**

* **main.py**: A FastAPI server script that unpickles the XGBoost model, provides a /predict endpoint, and processes incoming JSON requests.
* **Dockerfile**: Specifies runtime requirements (Python 3.9+), includes relevant libraries, and sets the container entry point to launch the FastAPI server.

**Access Instructions & Credentials**

* A valid Azure subscription is necessary. Security measures should be employed for credential management, e.g., .env files or Azure Key Vault.
* **App Service URL**: Example endpoint https://titanic-api-service.azurewebsites.net/predict.
* **Repository**: Either a GitHub link or a compressed archive can be offered, containing all source code and environment files.

1. **Evaluation and Conclusion**

**Challenges Faced**

1. **Tuning Discrepancies**: Ensuring the model did not overfit after parameter adjustments.
2. **ACR Authentication**: Maintaining valid credentials for seamless Docker push/pull operations.
3. **DataFrame Warnings**: Transitioning from inplace Pandas modifications to .assign() for improved forward compatibility.

**Model Performance**

* The XGBoost model attains an accuracy of roughly **83.2%**, surpassing the baseline Random Forest.
* End-to-end latency remains minimal, with test predictions handled swiftly.
* Verified via tools such as **Postman** and **cURL** to confirm reliability under varied test cases.

**Future Work**

* **Security Enhancements**: Integrate API tokens or JWT for controlled endpoint access.
* **Scalability**: Investigate hosting on Azure Kubernetes Service (AKS) or employing serverless architectures.
* **Monitoring & Logging**: Utilize Azure Monitor or Application Insights to gain deeper insights into usage patterns and potential error modes.

1. **Deliverables**
2. **Report (This Document)**  
   Provides a thorough discussion of the architectural decisions, the underlying dataset, and the deployment routine.
3. **Codebase**  
   Incorporates train\_model.py, main.py, Dockerfile, and relevant dependencies for reproducible execution.
4. **Live API URL**  
   For instance, https://titanic-api-service.azurewebsites.net/predict, tested with representative JSON inputs for passenger attributes.