**MLOps Assignment 1**

**Task M3 Summary**

**Description of the Work Completed**

This project involved training a machine learning model, performing hyperparameter tuning using Optuna, and packaging the model for deployment using Flask and Docker.

The main components of the project were:

|  |  |
| --- | --- |
| Hyperparameter Tuning with Optuna: | * A Random Forest model was chosen for the task of predicting California housing prices. * Optuna was used to perform hyperparameter tuning on the model, optimizing parameters such as `n\_estimators`, `max\_depth`, and `min\_samples\_split` to minimize the negative mean squared error through cross-validation. * The tuning process involved running 100 trials to find the best set of parameters, which were then used to train the final model. |
| Model Packaging with Flask and Docker: | * The best-performing model and the fitted StandardScaler were serialized using `joblib` for easy loading and use in deployment. * A Flask application (`m3\_flask.py`) was created to serve the model, exposing a `/predict` endpoint to receive feature data via POST requests and return predictions. * A Dockerfile was crafted to containerize the application, ensuring the environment is consistent and portable. The Docker image includes all necessary dependencies and exposes the Flask application on port 5000. |
| Testing the Deployment: | * A PowerShell script (`testflask.txt`) was written to test the Flask API by sending a sample request and checking the prediction output. |

**Justification for the Choices Made**

|  |  |
| --- | --- |
| Optuna for Hyperparameter Tuning: | * **Efficiency:** Optuna is highly efficient for hyperparameter optimization, utilizing techniques like Tree-structured Parzen Estimator (TPE) to efficiently search the parameter space. * **Flexibility:** It supports a wide range of search spaces and is easy to integrate with Python code, making it ideal for this experimentation task. |
| Random Forest Model: | * **Robustness:** Random Forest is known for its robustness and ability to handle complex datasets without overfitting, making it a suitable choice for regression tasks like predicting housing prices. * **Ease of Tuning:** The model's performance can be significantly improved through careful tuning of its parameters, which aligns well with the project's objectives. |
| Flask for Model Serving: | * **Simplicity:** Flask provides a simple yet powerful framework for building web applications, making it easy to create a RESTful API to serve the model. * **Scalability:** Flask applications can be easily scaled and integrated with other services, which is advantageous for deploying machine learning models in production. |
| Docker for Containerization: | * **Portability:** Docker ensures the application can run in any environment without compatibility issues, making it easy to deploy the model across different platforms. * **Reproducibility:** By containerizing the application, we ensure that the model's performance is consistent regardless of the underlying infrastructure. |