Machine Learning Systems Laboratory Examination

Duration: 1 hour
Maximum Marks: 50
Dataset: Iris Dataset
Instructions:

- All code should be written in Python
- The Iris dataset is available via `from sklearn.datasets`
- You may use common ML libraries like scikit-learn, pandas, numpy
- Document your code with appropriate comments
- Submit your code in a single Python file with clear section markers.

Question 1: Data Structure and Processing Pipeline (15 marks)

Using the Iris dataset, create a structured data processing pipeline:

```python
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

# Load the iris dataset
```

a) Create a data processing class that implements:

- Conversion of data to pandas DataFrame with proper column names
- Feature scaling using StandardScaler
- Train-test split with experiment tracking

```python
class IrisDataProcessor:
    def __init__(self):
        # Initialize your processor here
        pass

    def prepare_data(self):
        # Implement data preparation steps
        pass

    def get_feature_stats(self):
        # Implement basic statistical analysis
```

```
        pass
```

Question 2: Experiment Tracking and Model Development (20 marks)

Implement an experiment tracking system using MLflow for the Iris classification task:

a) Create an experimentation class that:

- Trains multiple models (Logistic Regressor, Random Forest)
- Tracks experiments with MLflow
- Implements cross-validation
- Records metrics (accuracy, precision, recall)

```python
class IrisExperiment:
    def __init__(self, data_processor):
        # Initialize your experiment
        pass

    def run_experiment(self):
        # Implement experiment workflow
        pass

    def log_results(self):
        # Implement MLflow logging
        pass
```

Question 3: Model Optimization and Testing (15 marks)

Implement model optimization and testing framework:

a) Create a model optimization class that:

- Implements model quantization (For Logistic regressor)
- Includes simple unit tests

```python
class IrisModelOptimizer:
    def __init__(self, experiment):
        # Initialize optimizer
        pass

    def quantize_model(self):
```

```
        # Implement model quantization
        pass

    def run_tests(self):
        # Implement unit tests
        pass
```

Question 4: Containerization and Deployment (15 marks)

a) Create a Dockerfile that:

- Uses an appropriate base image for Python ML applications
- Installs all required dependencies
- Sets up MLflow tracking
- Includes proper security considerations

b) Create a docker-compose.yml file that:

- Sets up the ML application service
- Configures MLflow tracking server
- Manages environment variables
- Sets up appropriate volume mounts for model artifacts.

Example Usage Template:

Python

```
def main():
    # Initialize processor
    processor = IrisDataProcessor()
    X_train, X_test, y_train, y_test = processor.prepare_data()

    # Run experiments
    experiment = IrisExperiment(processor)
    experiment.run_experiment()

    # Optimize and test
    optimizer = IrisModelOptimizer(experiment)
    optimizer.quantize_model()
    optimizer.run_tests()

if __name__ == "__main__":
    main()
```

Expected Output:

Your code should produce:

1. Processed dataset with scaled features
2. MLflow tracking results for multiple models
3. Quantized model with test results
4. Git commit history for model versions