05_model_evaluation

February 12, 2024

[1]:
```
import os
```

[2]:
```
%pwd
```

[2]:
```
'D:\\Desktop\\Deep Learning\\Lab 2\\MNSIT-MLPClassifer\\Research'
```

[3]:
```
os.chdir("../")
```

[4]:
```
%pwd
```

[4]:
```
'D:\\Desktop\\Deep Learning\\Lab 2\\MNSIT-MLPClassifer'
```

[5]:
```
import logging
import joblib
from dataclasses import dataclass
from pathlib import Path
import pandas as pd
from sklearn.metrics import accuracy_score, precision_recall_fscore_support
from sklearn.preprocessing import StandardScaler

# Configure logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -␣
 ↪%(message)s')

@dataclass(frozen=True)
class DataTransformationConfig:
    root_dir: Path
    X_test_file: Path
    y_test_file: Path
    scaler_file: Path
    mlp_mnist_model_file: Path

class ConfigurationManager:
    def __init__(self):
        self.root_dir = Path(os.getcwd())
        self.X_test_file = self.root_dir / "dataset/Modeltraining/X_test.csv"
        self.y_test_file = self.root_dir / "dataset/Modeltraining/y_test.csv"
        self.scaler_file = self.root_dir / "Model/scaler.pkl"
```

```python
        self.mlp_mnist_model_file = self.root_dir / "Model/mlp_mnist_model.pkl"

    def get_data_transformation_config(self) -> DataTransformationConfig:
        return DataTransformationConfig(
            root_dir=self.root_dir,
            X_test_file=self.X_test_file,
            y_test_file=self.y_test_file,
            scaler_file=self.scaler_file,
            mlp_mnist_model_file=self.mlp_mnist_model_file
        )

class ModelEvaluation:
    def __init__(self, config: DataTransformationConfig):
        self.config = config
        # Corrected attribute references
        self.scaler = joblib.load(config.scaler_file)
        self.model = joblib.load(config.mlp_mnist_model_file)

    def load_test_data(self):
        # Load the test data
        X_test = pd.read_csv(self.config.X_test_file)
        y_test = pd.read_csv(self.config.y_test_file)

        logging.info("X_test and y_test loaded")

        # Scale the test data
        X_test_scaled = self.scaler.transform(X_test)
        return X_test_scaled, y_test

    def generate_prediction_from_test(self):
        X_test_scaled, y_test = self.load_test_data()
        # Predict on the test set
        y_pred = self.model.predict(X_test_scaled)

        # Calculate and log evaluation metrics, handling undefined metrics
        accuracy = accuracy_score(y_test, y_pred)
        precision, recall, fscore, _ = precision_recall_fscore_support(y_test,␣
 ↪y_pred, average='macro', zero_division=0)

        # Log the metrics
        logging.info(f"Accuracy: {accuracy}")
        logging.info(f"Precision: {precision}")
        logging.info(f"Recall: {recall}")
        logging.info(f"F1 Score: {fscore}")

def main():
    try:
        config_manager = ConfigurationManager()
        data_transformation_config = config_manager.
↪get_data_transformation_config()
        model_evaluation = ModelEvaluation(config=data_transformation_config)

        # Generate predictions and evaluate
        model_evaluation.generate_prediction_from_test()

    except Exception as e:
        logging.error(f"Error occurred: {e}")
        raise e

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

```
2024-02-07 00:03:37,136 - INFO - X_test and y_test loaded
D:\Desktop\Deep Learning\Lab 2\MNSIT-MLPClassifer\venv\lib\site-
packages\sklearn\base.py:486: UserWarning: X has feature names, but
StandardScaler was fitted without feature names
  warnings.warn(
D:\Desktop\Deep Learning\Lab 2\MNSIT-MLPClassifer\venv\lib\site-
packages\sklearn\base.py:493: UserWarning: X does not have valid feature names,
but MLPClassifier was fitted with feature names
  warnings.warn(
2024-02-07 00:03:37,672 - INFO - Accuracy: 0.11177992502244047
2024-02-07 00:03:37,674 - INFO - Precision: 0.34756743469491197
2024-02-07 00:03:37,676 - INFO - Recall: 0.11274071613956986
2024-02-07 00:03:37,678 - INFO - F1 Score: 0.0412874236340312 55
```

```
[ ]:
```