02-train

June 10, 2024

[ ]:
```
import os, pickle, mlflow, logging
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

Configure logging

[ ]:
```
logging.basicConfig(filename='logs/training.log', level=logging.INFO)
```

Command to run for mlflow mlflow ui –backend-store-uri sqlite:///mlflow.db

Define functions

[ ]:
```
def load_pickle(fileName: str):
    """
    Load data from a pickle file.
    Args:
    fileName (str): Path to the pickle file.
    Returns:
    object: Data loaded from the pickle file.
    """
    try:
        with open(fileName, 'rb') as f:
            return pickle.load(f)
    except FileNotFoundError:
        logging.error(f"Error: File '{fileName}' not found.")
        return None
```

[ ]:
```
def train_(Data_path: str = 'DEST_PATH', max_depth: int = 10, random_state: int␣
 ↪= 0):
    """
    Train a random forest regressor model.
    Args:
    Data_path (str): Path to the directory containing data files.
    max_depth (int): Maximum depth of the trees in the random forest.
    random_state (int): Seed used by the random number generator.
    """
    mlflow.sklearn.autolog()

    # Load training and validation data
    X_train, y_train = load_pickle(os.path.join(Data_path, 'train.pkl'))
    X_val, y_val = load_pickle(os.path.join(Data_path, 'val.pkl'))

    # Convert target variables to numpy arrays
    y_train = y_train.to_numpy()
    y_val = y_val.to_numpy()

    # Start MLflow run
    with mlflow.start_run():
        logging.info("Training random forest regressor model...")
        # Initialize and train random forest regressor model
        rf = RandomForestRegressor(max_depth=max_depth,␣
 ↪random_state=random_state)
        rf.fit(X_train, y_train)
        y_pred = rf.predict(X_val)

        # Calculate root mean square error
        rmse = mean_squared_error(y_val, y_pred, squared=False)
        logging.info(f'Root Mean Square Error = {rmse}')
```

Entry point of the script

```
[ ]: if __name__ == '__main__':
         # Set the path to the data directory
         CURRENT_DIRECTORY = os.getcwd()
         DEST_PATH = os.path.join(CURRENT_DIRECTORY, 'DEST_PATH')

         # Train the model using the data in DEST_PATH
         train_(DEST_PATH)
```