dagfile_template.py

```python
'''
Explanation of Workflow:
    The DAG is named ml_pipeline and starts execution from January 1, 2024.
    The DAG is scheduled to run weekly (@weekly).
    Three tasks are defined:
        Preprocessing Data (preprocess_task)
        Training Model (train_task)
        Evaluating Model (evaluate_task)
    The tasks are executed in sequence:
        First, preprocessing is done.
        After preprocessing, training starts.
        Once training is complete, evaluation takes place.
'''

# Import necessary modules from Airflow
from airflow import DAG  # DAG (Directed Acyclic Graph) is the core concept in Airflow for workflow management
from airflow.operators.python import PythonOperator  # Import PythonOperator to execute Python functions as Airflow tasks
from datetime import datetime  # Import datetime module to define the start date of the DAG

# Define our first task: Preprocessing the data
def preprocess_data():
    print("Preprocessing data...")  # This function prints a message indicating that data preprocessing is happening

# Define our second task: Training the machine learning model
def train_model():
    print("Training model...")  # This function prints a message indicating that the model is being trained

# Define our third task: Evaluating the model performance
def evaluate_model():
    print("Evaluate Models...")  # This function prints a message indicating that the model evaluation is happening

# Define the DAG (Directed Acyclic Graph) for our machine learning pipeline
with DAG(
    'ml_pipeline',  # Name of the DAG
    start_date=datetime(2024, 1, 1),  # Set the start date for the DAG execution
    schedule_interval='@weekly'  # Schedule the DAG to run weekly
) as dag:

    # Define the first task: Preprocessing data
    preprocess = PythonOperator(
        task_id="preprocess_task",  # Unique identifier for this task
        python_callable=preprocess_data  # Function to execute when this task runs
    )

    # Define the second task: Training the model
    train = PythonOperator(
        task_id="train_task",  # Unique identifier for this task
        python_callable=train_model  # Function to execute when this task runs
    )

    # Define the third task: Evaluating the model
    evaluate = PythonOperator(
        task_id="evaluate_task",  # Unique identifier for this task
        python_callable=evaluate_model  # Function to execute when this task runs
    )

    # Set task dependencies (execution order)
    preprocess >> train >> evaluate  # Preprocessing must complete before training, and training must complete before evaluation

```