Aim

To develop and evaluate reinforcement learning models for stock trading using historical market data.

Objective

1. Implement and test different reinforcement learning algorithms for stock trading.
2. Analyze the performance of A2C and PPO models in a stock trading environment.
3. Visualize the trading results to evaluate the effectiveness of the models.

Summary

This project involves applying reinforcement learning (RL) techniques to stock trading simulations. Using historical market data, we employ RL algorithms, specifically A2C and PPO, to develop trading strategies. The project utilizes the gym-anytrading environment for simulating trading scenarios and evaluates the models' performance by visualizing trading actions and rewards.

Tools and Libraries Used

- **Python Libraries**: TensorFlow 1.15.0, Gym, Gym-Anytrading, Stable Baselines, Stable Baselines3, NumPy, Pandas, Matplotlib
- **Environment**: Jupyter Lab
- **GPU**: RTX 3060

Procedure

1. **Data Loading**: Load historical market data from a CSV file using Pandas and preprocess it by setting the 'Date' column as the index.

CODE:

```
df = pd.read_csv('data/gmedata.csv')

df['Date'] = pd.to_datetime(df['Date'])

df.set_index('Date', inplace=True)
```

2. **Environment Setup**: Create a stock trading environment using the gym-anytrading library and define the observation and action spaces.

CODE:

```
env = gym.make('stocks-v0', df=df, frame_bound=(5,100), window_size=5)
```

3. **Initial Exploration**: Sample random actions in the environment to get a baseline understanding of its behavior.

CODE:

```
state = env.reset()
```

```
while True:
    action = env.action_space.sample()
    n_state, reward, done, info = env.step(action)
    if done:
        print("info", info)
        break
```

4. **Model Training with A2C**: Train the A2C model using the defined environment. After training, evaluate the model's performance in a different time frame.

CODE:

```
env_maker = lambda: gym.make('stocks-v0', df=df, frame_bound=(5,100), window_size=5)
env = DummyVecEnv([env_maker])
model = A2C('MlpLstmPolicy', env, verbose=1)
model.learn(total_timesteps=1000000)
```

5. **Model Evaluation with A2C**: Run the trained A2C model in a test environment and visualize the trading results.

CODE:

```
env = gym.make('stocks-v0', df=df, frame_bound=(90,110), window_size=5)
obs = env.reset()
while True:
    obs = obs[np.newaxis, ...]
    action, _states = model.predict(obs)
    obs, rewards, done, info = env.step(action)
    if done:
        print("info", info)
        break
```

6. **Model Training with PPO**: Similarly, train the PPO model and evaluate its performance.

CODE:

```
env = DummyVecEnv([env_maker])
model = PPO('MlpPolicy', env, verbose=1)
```

```
model.learn(total_timesteps=1000000)
```

7. **Model Evaluation with PPO**: Run the trained PPO model in a test environment and visualize the trading results.

CODE:

```
env = gym.make('stocks-v0', df=df, frame_bound=(90,110), window_size=5)

obs = env.reset()

while True:

  obs = obs[np.newaxis, ...]

  action, _states = model.predict(obs)

  obs, rewards, done, info = env.step(action)

  if done:

    print("info", info)

    break
```

8. **Visualization**: Plot the trading results for both models to analyze their performance.

CODE:

```
plt.figure(figsize=(15,6))

plt.cla()

env.render_all()

plt.show()
```

Highlights

- **Algorithm Comparison**: The project compares the performance of A2C and PPO algorithms in a stock trading environment.
- **Visualization**: Effective use of Matplotlib to visualize the trading actions and results for better interpretation of model performance.
- **Comprehensive Approach**: The project covers data preprocessing, environment setup, model training, and evaluation, providing a holistic view of RL in trading.

Conclusion

The project successfully implements and evaluates reinforcement learning models for stock trading. Both A2C and PPO algorithms were tested, with results visualized to assess their effectiveness. This approach demonstrates the potential of RL for developing automated trading strategies and provides insights into the performance of different RL algorithms in financial simulations.