# A novel trading system for the stock market using Deep Q-Network action and instance selection

Myeongseok Park, Jaeyun Kim, David Enke

Soonchunhyang University, South Korea

https://www.sciencedirect.com/science/article/abs/pii/S0957417424019109

IIE4122 Midterm Project | 25.04.21 Mon

TEAM 13 | Kim Wonjun(2022147002), Kim Hyeonjin(2021125085), Jang Seohyun(2021190002)

# Introduction

Consequently, trading systems designed to generate high stock market returns are developed via several supervised learning methods

✓ However, methods based on them make it difficult to adapt to the real-time nature of the stock market (can be noisy and fail to consider the nonlinear and complex nature of stock prices)

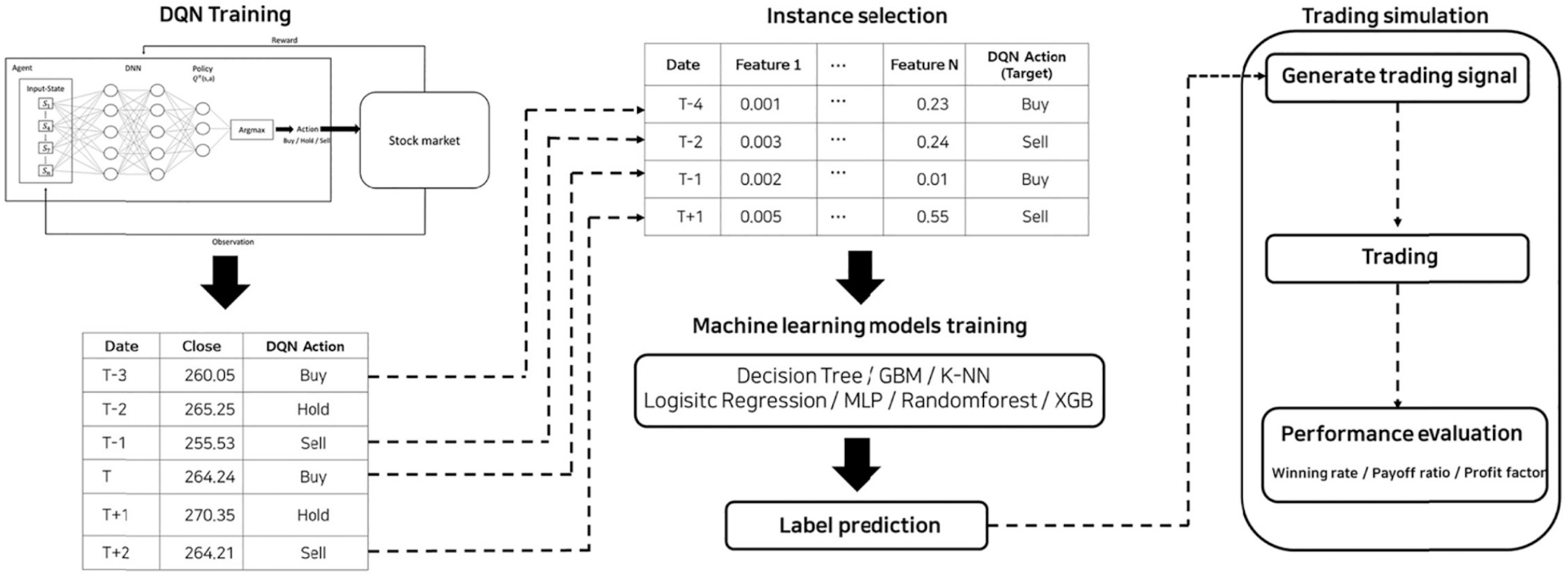
**Contribution**

1. By applying reinforcement learning to stock trading, flexible strategies can be developed to adapt to changing market conditions
2. Existing literatures use price up/down labeling as labels, while this paper uses buy/sell signals trained by DQN

This study proposes a Deep Q-Network (DQN) Action Instance Selection Trading System (DAIS) to improve the limitations of both supervised learning and reinforcement learning trading systems

# Methodology

2) The learning data (buy/sell signal + all technical indicators) and machine learning algorithm were constructed using instance selection




| Date | Close | DQN Action |
|---|---|---|
| T-3 | 260.05 | Buy |
| T-2 | 265.25 | Hold |
| T-1 | 255.53 | Sell |
| T | 264.24 | Buy |
| T+1 | 270.35 | Hold |
| T+2 | 264.21 | Sell |

| Date | Feature 1 | ··· | Feature N | DQN Action (Target) |
|---|---|---|---|---|
| T-4 | 0.001 | ··· | 0.23 | Buy |
| T-2 | 0.003 | ··· | 0.24 | Sell |
| T-1 | 0.002 | ··· | 0.01 | Buy |
| T+1 | 0.005 | ··· | 0.55 | Sell |

1) Gather information about the stock market and extract actions

3) Use a machine learning algorithm to predict and generate trading signals

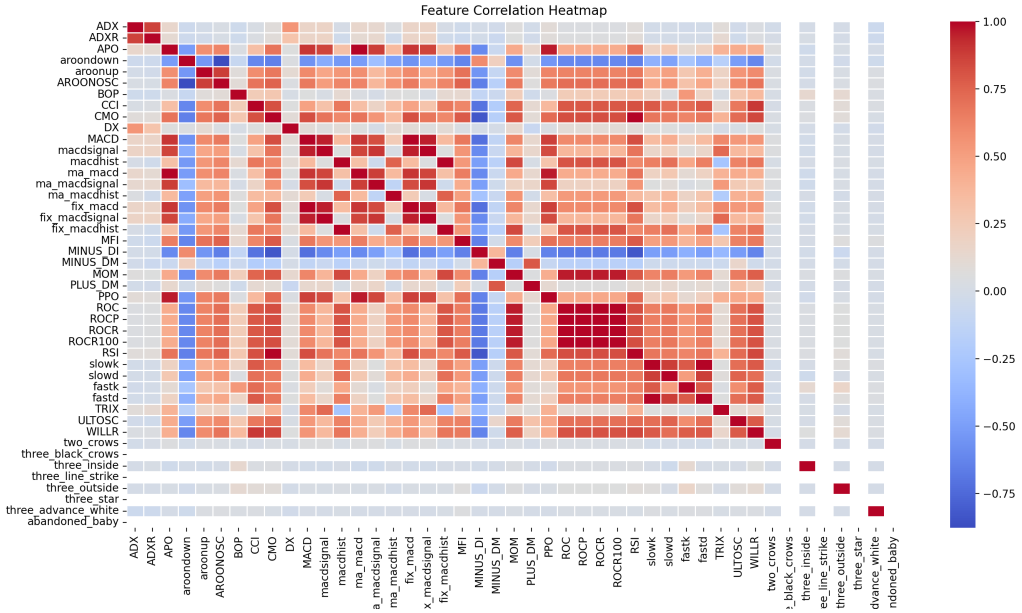
4) Conduct a simulation using the generated trading signal and evaluate its performance

# Contribution

✓ Limitation of Paper:

> This study had some limitations. First, we arbitrarily selected the technical indicators to construct the reinforcement learning environment. The results may have been different if we had used more technical indicators or significant technical indicators. Second, we used the DQN

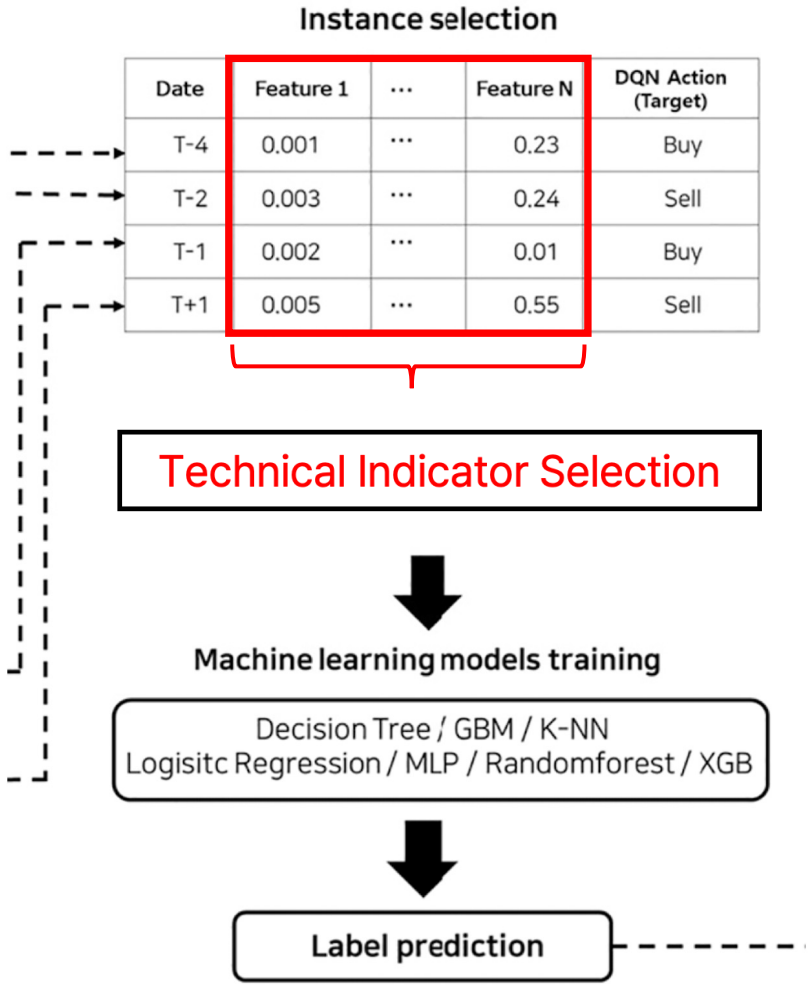
✓ Correlations among technical indicators




∴ Technical Indicator Selection is needed.

→ Feature selection is done using 1) Randomforest
2) mRMR

N → Select Top 20 features



# Improvement

## Feature Selection code

```python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from mrmr import mrmr_classif

feature_cols = [
    'ADX','ADXR','APO','aroondown','aroonup','AROONOSC','BOP','CCI','CMO','DX',
    'MACD','macdsignal','macdhist','ma_macd','ma_macdsignal','ma_macdhist',
    'fix_macd','fix_macdsignal','fix_macdhist','MFI','MINUS_DI','MINUS_DM',
    'MOM','PLUS_DM','PPO','ROC','ROCP','ROCR','ROCR100','RSI','slowk','slowd',
    'fastk','fastd','TRIX','ULTOSC','WILLR','two_crows','three_black_crows',
    'three_inside','three_line_strike','three_outside','three_star',
    'three_advance_white','abandoned_baby'
]

train_combined = []
for i in train_data:
    if 'label' in i.columns:
        tmp = i[feature_cols + ['label']].dropna()
        if not tmp.empty:
            train_combined.append(tmp)

data_all = pd.concat(train_combined, ignore_index=True)

X = data_all[feature_cols]
y = data_all['label']
y_bin = (y == 1).astype(int)  # Buy = 1, Sell = 0

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_df = pd.DataFrame(X_scaled, columns=X.columns)
```

```python
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_scaled, y_bin)
importances = rf.feature_importances_
selected_rf = X.columns[np.argsort(importances)[-20:]]

selected_mrmr = mrmr_classif(X=X_df, y=y_bin, K=20)

print("RandomForest Top 20 Features:", selected_rf.tolist())
print("MRMR Top 20 Features:", selected_mrmr)

import seaborn as sns
import matplotlib.pyplot as plt

corr_matrix = X_df.corr()
plt.figure(figsize=(15, 12))
sns.heatmap(corr_matrix, cmap='coolwarm', annot=False, fmt='.2f', linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.tight_layout()
plt.show()
```

# Improvement

Selected Top 20 Features

**RandomForest Top 20 Features:** ['MOM', 'CCI', 'PPO', 'MINUS_DI', 'MACD', 'TRIX', 'fix_macd', 'ma_macdsignal', 'DX', 'MFI', 'ULTOSC', 'macdhist', 'ADXR', 'fix_macdhist', 'ma_macdhist', 'ADX', 'macdsignal', 'fix_macdsignal', 'MINUS_DM', 'PLUS_DM']

**MRMR Top 20 Features:** ['PLUS_DM', 'two_crows', 'ADX', 'MINUS_DI', 'MINUS_DM', 'ma_macdsignal', 'DX', 'ADXR', 'fix_macdsignal', 'ma_macdhist', 'macdsignal', 'BOP', 'fix_macd', 'MACD', 'APO', 'ma_macd', 'TRIX', 'macdhist', 'fix_macdhist', 'PPO']

→ We adopted top 20 features from **random forest**
according to the results of performance improvement

# Results

**Table 5**
Trading performance by DQN and DAIS trading system.

| Model | No. trades | Winning ratio | Payoff ratio | Profit factor | Sharpe ratio |
|---|---|---|---|---|---|
| Decision tree | 90.30 | 0.48 | 1.10 (0.30) | 1.11 (0.53) | 1.03 |
| GBM | 88.27 | 0.47 | 1.10 (0.28) | 1.10 (0.57) | 0.66 |
| knn | 78.62 | 0.49 | 1.10 (0.40) | 1.13 (0.67) | 0.92 |
| Logisitic Regression | 52.70 | 0.49 | 1.12 (0.59) | 1.21 (1.93) | 0.58 |
| MLP | 61.35 | 0.49 | 1.12 (0.53) | 1.21 (0.89) | 0.67 |
| Random forest | 76.98 | 0.49 | 1.15 (0.34) | 1.20 (0.58) | 1.09 |
| XGB | 89.04 | 0.49 | 1.04 (0.28) | 1.09 (0.52) | 0.93 |
| DQN | 84.01 | 0.46 | 0.97 (0.45) | 0.94 (0.59) | 0.38 |

Note: Values for trading performance are given as average (standard deviation).

Trading Performance of Original Paper

| methodology | year | stock_name | No.trades | Win% |
|---|---|---|---|---|
| pred_logistic | 2019~2020 | 005930_삼성전자 | 33 | 0.666667 |
| pred_decision | 2019~2020 | 005930_삼성전자 | 98 | 0.581633 |
| pred_naive (DQN) | 2019~2020 | 005930_삼성전자 | 29 | 0.758621 |
| pred_randomforest | 2019~2020 | 005930_삼성전자 | 65 | 0.461538 |
| pred_knn | 2019~2020 | 005930_삼성전자 | 51 | 0.509804 |
| pred_neural | 2019~2020 | 005930_삼성전자 | 42 | 0.52381 |
| pred_voting | 2019~2020 | 005930_삼성전자 | 45 | 0.488889 |
| pred_gbm | 2019~2020 | 005930_삼성전자 | 107 | 0.336449 |

Trading Performance of Proposed Methodology (using RF)

- ✓ Overall, the **winning ratio** has generally improved compared to the pre-improvement range (0.48~0.49).
- ✓ Several models achieved a **winning ratio above 50%**, suggesting a **higher likelihood of making profitable trades**.
- ✓ However, the **number of trades has slightly decreased**, averaging around 50–60 trades over two years (indicating relatively low trading frequency)
- ✓ Although the improved winning ratio implies better algorithmic performance, the **small number of trades (n)** limits the application of the law of large numbers, potentially introducing statistical noise.

# Code Instruction Manual

✓ Development Environment: Python 3.10.1, Torch 2.5.1 => MUST DOWNGRADE

✓ TA-Lib must to be downloaded at https://github.com/TA-Lib/ta-lib-python.git
=> Then, run !pip install ta-lib-0.6.4-src.tar.gz within VScode (Linux version)

**Windows**

For 64-bit Windows, the easiest way is to get the *executable installer*:

1. Download ta-lib-0.6.4-windows-x86_64.msi.
2. Run the Installer or run `msiexec` from the command-line.

Alternatively, if you prefer to get the libraries without installing, or would like to use the 32-bit version:

- Intel/AMD 64-bit ta-lib-0.6.4-windows-x86_64.zip
- Intel/AMD 32-bit ta-lib-0.6.4-windows-x86_32.zip

**Linux**

Download ta-lib-0.6.4-src.tar.gz and:

```
$ tar -xzf ta-lib-0.6.4-src.tar.gz
$ cd ta-lib-0.6.4/
$ ./configure --prefix=/usr
$ make
$ sudo make install
```

> If you build `TA-Lib` using `make -jX` it will fail but that's OK! Simply rerun `make -jX` followed by `[sudo] make install`.

✓ Other libraries(ex. ko_KR.UTF-8, mplfinance, tqdm) can be installed within VScode if needed (installation commands are inserted within submitted codes -> Google-colab basis)

Brief Instructions -> Detailed instructions are given

1. Run ‘main.py’
   1a. Relocate prediction results to appropriate folders
2. Run ‘main2.py’
3. Retrieve final results in .csv format

# Code Instruction Manual

Run ‘main.py’ with the following commands sequentially

0. Modify path at line[7]

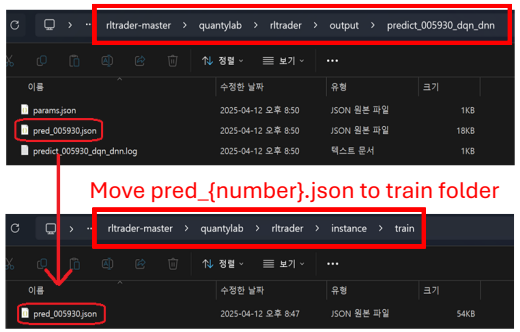
1. train data

```
python main.py --mode train --ver v5 --name 005930 --stock_code 005930 --rl_method dqn --net dnn --start_date 20130101 --end_date 20181231
```

2. predict train set

```
python main.py --mode predict --ver v5 --name 005930 --stock_code 005930 --rl_method dqn --net dnn --start_date 20130101 --end_date 20181231
```

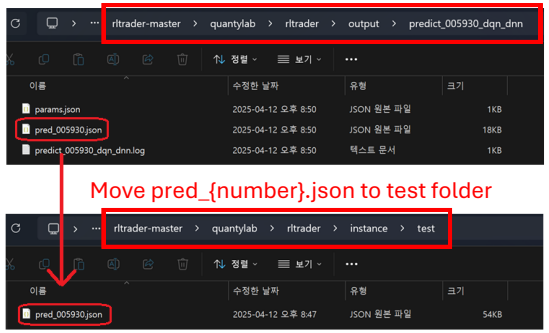
2a. Relocate train set prediction results




3. predict test set

```
python main.py --mode predict --ver v5 --name 005930 --stock_code 005930 --rl_method dqn --net dnn --start_date 20190101 --end_date 20201231
```

3a. Relocate test set prediction results




4. test data

```
python main.py --mode test --ver v5 --name 005930 --stock_code 005930 --rl_method dqn --net dnn --start_date 20190101 --end_date 20201231
```

# Code Instruction Manual

Run ‘main2.py’ => Retrieve final results in .csv format

Example of final result:

| year | stock_nam | No.trades | Win% | Average g | Average lo | Payoff ratio | Profit factc | Model |
|---|---|---|---|---|---|---|---|---|
| 2019~202( | 005930_삼 | 33 | 0.666667 | 3809.166 | 3489.998 | 1.091452 | 2.182905 | pred_logistic |
| 2019~202( | 005930_삼 | 98 | 0.581633 | 2691.063 | 2805.99 | 0.959042 | 1.333303 | pred_decision |
| 2019~202( | 005930_삼 | 29 | 0.758621 | 3203.988 | 3172.075 | 1.01006 | 3.174476 | pred_naive |
| 2019~202( | 005930_삼 | 65 | 0.461538 | 1853.863 | 2035.293 | 0.910858 | 0.780736 | pred_randomforest |
| 2019~202( | 005930_삼 | 51 | 0.509804 | 1050.141 | 1388.124 | 0.756518 | 0.786779 | pred_knn |
| 2019~202( | 005930_삼 | 42 | 0.52381 | 1439.551 | 1464.789 | 0.98277 | 1.081048 | pred_neural |
| 2019~202( | 005930_삼 | 45 | 0.488889 | 1347.668 | 1472.151 | 0.915441 | 0.87564 | pred_voting |
| 2019~202( | 005930_삼 | 107 | 0.336449 | 1171.088 | 1673.365 | 0.69984 | 0.354849 | pred_gbm |

We have attached an inference code ‘inference.ipynb’ to help with code execution

# Thank You

IIE4122 Midterm Project | 25.04.21 Mon

TEAM 13 | Kim Wonjun(2022147002), Kim Hyeonjin(2021125085), Jang Seohyun(2021190002)

[Reference] Park, M., Kim, J., & Enke, D. (2024). A novel trading system for the stock market using Deep Q-Network action and instance selection. Expert Systems with Applications, 257, 125043.