1/12/2023

Konstantinos Gerovasilis, Chariton Kypraios

[επωνυμία εταιρείας]

Τεχνητή Νοημοσύνη

Εργασία: Πρόβλεψη μετοχών με αλγορίθμους μηχανικής μάθησης

Η συγκεκριμένη εργασία έχει ως στόχο την πρόβλεψη μετοχών με την χρήση αλγορίθμων μηχανικής μάθησης. Στην συγκεκριμένη εργασία χρησιμοποιούνται 3 αλγόριθμοι μηχανικής μάθησης:

1. Ο αλγόριθμος Γραμμικής Παλινδρόμησης (Linear Progression).
2. Ο αλγόριθμος Δέντρων Αποφάσεων (Decision Trees).
3. Ο αλγόριθμος Νευρωνικών Δικτύων (Neural Networks).

Σε αυτό το σημείο θα επεξηγηθούν οι συγκεκριμένοι αλγόριθμοι και θα αναλυθούν τα χαρακτηριστικά τους, η αρχιτεκτονική τους, η ποσότητα των κόμβων τους (οι κρυφοί κόμβοι κτλ), τι input παίρνουν, τι output δίνουν.

Στην συνέχεια θα αναλυθούν οι κώδικες για κάθε ένα αλγόριθμοι, καθώς και τα αποτελέσματα που δίνουν.

Να σημειωθεί πως όλοι οι αλγόριθμοι δουλεύουν με μία συγκεκριμένη μετοχή, αυτή της APPLE και όλα τα αποτελέσματα αφορούν αυτή την μετοχή. Συνεπώς τα αποτελέσματα είναι αποδιδόμενα με βάση την συγκεκριμένη μετοχή. Είναι πιθανό πως με κάποια άλλη μετοχή ο κάθε αλγόριθμος ίσως έχει διαφορετικά αποτελέσματα.

**Κώδικας Αλγόριθμου Γραμμικής Παλινδρόμησης**

Παρακάτω φαίνεται ο κώδικας του αλγορίθμου Γραμμικής Παλινδρόμησης (Input):

#Αλγόριθμος Γραμμικής Παλινδρόμησης

# Εισαγωγή απαραίτητων βιβλιοθηκών

import yfinance as yf

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

import numpy as np

# Λήψη δεδομένων από το χρηματιστήριο

#symbol = 'ALPHA.AT'

#data = yf.download(symbol, start='2020-01-01', end='2024-01-01')

symbol = 'AAPL'  # Παράδειγμα με μετοχή Apple

start\_date = '2022-01-01'

end\_date = '2023-01-01'

data = yf.download(symbol, start=start\_date, end=end\_date)

# Καθαρισμός και επεξεργασία δεδομένων

data.dropna(inplace=True) # Διαγραφή απουσιάζουσων τιμών

data['Return'] = data['Close'].pct\_change() # Υπολογισμός ημερήσιας απόδοσης

# Οπτικοποίηση δεδομένων

plt.figure(figsize=(10, 6))

plt.plot(data['Close'], label='Κλείσιμο')

plt.title(f'Ιστορική Τιμή Κλεισίματος {symbol}')

plt.legend()

plt.show()

# Δημιουργία μεταβλητών για το μοντέλο μηχανικής μάθησης

data['Previous\_Close'] = data['Close'].shift(1) # Τιμή κλεισίματος προηγούμενης ημέρας

data.dropna(inplace=True) # Διαγραφή απουσιάζουσων τιμών μετά τη μετατόπιση

# Διαχωρισμός σε σετ εκπαίδευσης και δοκιμής

X = data[['Previous\_Close']]

y = data['Close']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# Εκπαίδευση του μοντέλου

model = LinearRegression()

model.fit(X\_train, y\_train)

# Πρόβλεψη τιμών

predictions = model.predict(X\_test)

# Αξιολόγηση του μοντέλου

mse = mean\_squared\_error(y\_test, predictions)

mae = mean\_absolute\_error(y\_test, predictions)

print(f'MSE: {mse}')

print(f'MAE: {mae}')

# Οπτικοποίηση των προβλέψεων

# plt.figure(figsize=(10, 6))

# plt.scatter(y\_test, predictions)

# plt.xlabel('Πραγματικές Τιμές')

# plt.ylabel('Προβλέψεις')

# plt.title('Πραγματικές έναντι Προβλεπόμενων Τιμών')

# plt.show()

plt.scatter(X\_test, y\_test, color='black', label='Πραγματικές τιμές')

plt.scatter(X\_test, predictions, color='blue', label='Προβλέψεις')

plt.xlabel('Τιμή Κλεισίματος')

plt.ylabel('Επόμενη Τιμή Κλεισίματος')

plt.title('Πραγματικές τιμές vs Προβλέψεις')

plt.legend()

plt.show()

Πάνω από κάθε εντολή υπάρχουν και τα απαραίτητα σχόλια ώστε να εξηγείται σε κάθε γραμμή τι κάνει ο αντίστοιχος κώδικας.

Στην παρακάτω εικόνα φαίνεται το αποτέλεσμα του κώδικα (output):

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Αυτό που φαίνεται είναι οι προβλέψεις που έκανε ο αλγόριθμος σε σχέση με την πραγματική τιμή της μετοχής. Αυτό που είναι εμφανές είναι ότι οι προβλέψεις του αλγορίθμου είναι σχεδόν μία ευθεία γραμμή και σε αρκετά σημεία οι προβλέψεις του πέφτουν εκτός των πραγματικών τιμών.

Συνεπώς φαίνεται ότι ο αλγόριθμος Γραμμικής Παλινδρόμησης δεν αποτελεί το πιο αξιόπιστο μέσο πρόβλεψης για την συγκεκριμένη μετοχή.

**Κώδικας Αλγορίθμου Δέντρων Απόφασης**

Παρακάτω φαίνεται ο κώδικας του αλγορίθμου Απόφασης Δέντρων (Input):

#Αλγόριθμος Δέντρων Απόφασης

import yfinance as yf

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

# Λήψη δεδομένων μετοχής από το yfinance

symbol = 'AAPL'  # Παράδειγμα με μετοχή Apple

start\_date = '2022-01-01'

end\_date = '2023-01-01'

data = yf.download(symbol, start=start\_date, end=end\_date)

# Επιλογή τιμής κλεισίματος ως χαρακτηριστικό

features = data[['Close']]

# Εισαγωγή της μελλοντικής τιμής κλεισίματος ως μεταβλητή πρόβλεψης

features['Next\_Close'] = features['Close'].shift(-1)

# Αφαίρεση των τελευταίων γραμμών που περιέχουν NaN λόγω του shift

features = features.dropna()

# Επιλογή των χαρακτηριστικών και της μεταβλητής πρόβλεψης

X = features[['Close']]

y = features['Next\_Close']

# Διαίρεση των δεδομένων σε σύνολα εκπαίδευσης και ελέγχου

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Δημιουργία και εκπαίδευση του μοντέλου Δέντρα Απόφασης

model = DecisionTreeRegressor()

model.fit(X\_train, y\_train)

# Προβλέψεις στα δεδομένα ελέγχου

predictions = model.predict(X\_test)

# Αξιολόγηση του μοντέλου

mse = mean\_squared\_error(y\_test, predictions)

print(f'Mean Squared Error: {mse}')

# Παρουσίαση των πραγματικών τιμών και των προβλέψεων

plt.scatter(X\_test, y\_test, color='black', label='Πραγματικές τιμές')

plt.scatter(X\_test, predictions, color='blue', label='Προβλέψεις')

plt.xlabel('Τιμή Κλεισίματος')

plt.ylabel('Επόμενη Τιμή Κλεισίματος')

plt.title('Πραγματικές τιμές vs Προβλέψεις')

plt.legend()

plt.show()

Πάνω από κάθε εντολή υπάρχουν και τα απαραίτητα σχόλια ώστε να εξηγείται σε κάθε γραμμή τι κάνει ο αντίστοιχος κώδικας.

Στην παρακάτω εικόνα φαίνεται το αποτέλεσμα του κώδικα (output):

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Παρατηρείται πως οι προβλέψεις του αλγόριθμου Δέντρων Απόφασης είναι αρκετά προσεγγιστικές και σε πολλά σημεία οι προβλέψεις ήταν ακριβής ή αρκετά κοντά στις πραγματικές τιμές.

Συνεπώς φαίνεται ότι ο αλγόριθμος Δέντρων Απόφασης θα μπορούσε να αποτελέσει ένα αξιόπιστο μέσο πρόβλεψης για την συγκεκριμένη μετοχή.

**Κώδικας αλγορίθμου Νευρωνικών Δικτύων**

#Νευρωνικά δίκτυα

import yfinance as yf

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, LSTM

from matplotlib import pyplot as plt

# Λήψη δεδομένων μετοχής από το yfinance

symbol = 'AAPL'  # Παράδειγμα με μετοχή Apple

start\_date = '2022-01-01'

end\_date = '2023-01-01'

data = yf.download(symbol, start=start\_date, end=end\_date)

# Επιλογή τιμής κλεισίματος ως χαρακτηριστικό

features = data[['Close']]

# Κλιμακοποίηση των χαρακτηριστικών στο εύρος [0, 1]

scaler = MinMaxScaler()

features\_scaled = scaler.fit\_transform(features)

# Δημιουργία συνόλων εκπαίδευσης και ελέγχου

X, y = [], []

look\_back = 10  # Παράδειγμα με παραθυροσχήματα μήκους 10

for i in range(len(features\_scaled) - look\_back):

    X.append(features\_scaled[i:i+look\_back])

    y.append(features\_scaled[i+look\_back])

X, y = np.array(X), np.array(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Δημιουργία και εκπαίδευση του μοντέλου νευρωνικού δικτύου

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1], 1)))

model.add(LSTM(units=50))

model.add(Dense(units=1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

model.fit(X\_train, y\_train, epochs=10, batch\_size=32)

# Προβλέψεις στα δεδομένα ελέγχου

predictions\_scaled = model.predict(X\_test)

predictions = scaler.inverse\_transform(predictions\_scaled)

# Αξιολόγηση του μοντέλου

mse = mean\_squared\_error(features[-len(y\_test):], predictions)

print(f'Mean Squared Error: {mse}')

# Παρουσίαση των πραγματικών τιμών και των προβλέψεων

plt.plot(features.index[-len(y\_test):], features[-len(y\_test):], label='Πραγματικές τιμές')

plt.plot(features.index[-len(y\_test):], predictions, label='Προβλέψεις')

plt.xlabel('Ημερομηνία')

plt.ylabel('Τιμή Κλεισίματος')

plt.title('Πραγματικές τιμές vs Προβλέψεις')

plt.legend()

plt.show()