The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 23, No. 4, pp.79-84, Aug. 31, 2023. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2023.23.4.79
JIIBC 2023-4-13

Apache Spark를 활용한 실시간 주가 예측

Real-Time Stock Price Prediction using Apache Spark

신동진[*], 황승연[**], 김정준[***]

Dong-Jin Shin[*], Seung-Yeon Hwang[**], Jeong-Joon Kim[***]

요 약 최근 분산 및 병렬 처리 기술 중 빠른 처리 속도를 제공하는 Apache Spark는 실시간 기능 및 머신러닝 기능을 제공하고 있다. 이러한 기능에 대한 공식 문서 가이드가 제공되고 있지만, 기능들을 융합하여 실시간으로 특정 값을 예측하는 방안은 제공되고 있지 않다. 따라서 본 논문에서는 이러한 기능들을 융합하여 실시간으로 데이터의 값을 예측할 수 있는 연구를 진행했다. 전체적인 구성은 Python 프로그래밍 언어에서 제공하는 주가 데이터를 다운로드하여 수집한다. 그리고 머신러닝 기능을 통해 회귀분석의 모델을 생성하고, 실시간 스트리밍 기능을 머신러닝 기능과 융합하여 실시간으로 주가 데이터 중 조정종가를 예측한다.

Abstract Apache Spark, which provides the fastest processing speed among recent distributed and parallel processing technologies, provides real-time functions and machine learning functions. Although official documentation guides for these functions are provided, a method for fusion of functions to predict a specific value in real time is not provided. Therefore, in this paper, we conducted a study to predict the value of data in real time by fusion of these functions. The overall configuration is collected by downloading stock price data provided by the Python programming language. And it creates a model of regression analysis through the machine learning function, and predicts the adjusted closing price among the stock price data in real time by fusing the real-time streaming function with the machine learning function.

Key Words : Regression Analysis, Spark, Spark Structured Streaming, Stock Prediction

Ⅰ. 서 론

최근 인공지능, 사물인터넷, 빅데이터 등 다양한 분야의 기술들이 발전하고 있다. 특히 분산 및 병렬 환경을 구성하고, 처리 속도를 향상하기 위한 기술들이 주목을 받고 있다[1]. 그중 초기에는 각 디스크를 분산 및 병렬 환경으로 구성하여 데이터를 저장하고, 디스크 기반 처리를 활용한 Map-Reduce 기술이 많이 사용되었다. 하지만, 디스크의 처리 속도의 한계에 따라 오버헤드가 발생하고, 이를 해결하기 위해 Apache Spark(이하 Spark)가 개발되었다[2]. Spark에서 데이터 저장은 HDFS, NoSQL 등과 같은 분산 파일 시스템을 이용할 수 있고, 메모리에 데이터를 업로드하여 좀 더 빠르게 처리할 수 있는 기술이다. 또한, 머신러닝, 스트리밍, Graph 등 다양한 기능들을 제공하여 최근 많이 활용되고 있다[3].

본 연구의 실시간 주가 분석에서 사용한 Spark의 장

[*]준회원, 안양대학교 컴퓨터공학과
[**]준회원, 안양대학교 컴퓨터공학과
[***]정회원, 안양대학교 소프트웨어학과 (교신저자)
접수일자 2023년 4월 25일, 수정완료 2023년 7월 2일
게재확정일자 2023년 8월 4일

Received: 25 April, 2023 / Revised: 2 July, 2023 /
Accepted: 4 August, 2023
[***]Corresponding Author: jjkim@anyang.ac.kr
Dept. Software at AnYang University, Korea.

점은 다음과 같다. 첫 번째, Spark Structured Streaming을 활용하면, 사용자가 원하는 시간 값에 따라 실시간으로 데이터를 수집할 수 있다. 두 번째, 인-메모리 기법을 통해 데이터를 처리 및 분석하기에 하드디스크에서 발생하는 Input/Output 병목 현상이 나타나지 않아 좀 더 빠르게 분석할 수 있다. 세 번째, Spark Machine Learning에서 제공하는 다양한 라이브러리를 통해 분석이 쉬우며, Python 언어와 융합을 할 수 있는 PySpark 패키지를 활용하면, 처리 및 분석에 필요한 프로그래밍 언어 장벽도 높지 않아 효율적으로 사용할 수 있다. 하지만, Spark는 자체적인 파일 시스템이 존재하지 않아 HDFS, NoSQL과 같은 외부 파일 시스템을 연결 및 설정하여야 하는 단점이 존재한다.

본 논문에서는 Spark에서 제공하는 기능 중 머신러닝과 스트리밍을 융합한 실시간 머신러닝 기반 회귀분석 방안을 제안한다. 논문은 1장 서론을 시작으로 2장에서는 관련된 기술과 이론들을 살펴본다. 그리고 3장에서는 구현된 시스템의 핵심 소스코드를 소개하고, 모델을 평가하며, 마지막으로 4장에서 결론으로 마무리된다.

II. 관련 기술 및 이론

1. Spark

Spark는 Map-Reduce의 디스크 기반 처리 한계를 개선하기 위해 메모리에 데이터를 업로드하고, 처리할 수 있는 기술이다. 데이터의 저장소는 HDFS, NoSQL 등 다양한 분산 파일 시스템을 지원한다. 클러스터 기반 분산 및 병렬 처리 환경을 지원하고 있으며, 처리할 때 사용되는 리소스 매니저를 선택할 수 있다.

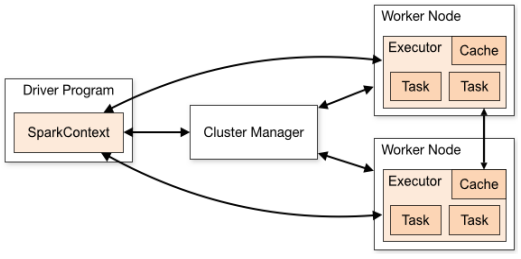


그림 1. Spark 기본 병렬 처리 구조의 모습
Fig. 1. Spark basic parallel processing structure

그림 1은 Spark의 병렬 처리 구조의 모습을 보여준다. 대표적으로는 Hadoop의 Yarn, Spark의 내부 매니저, Kubernetes를 지원하고 있으며, Apache의 Mesos도 지원은 하지만, 현재 3.3.0 기준 개발 지원이 중단된 상태이다[4]. 그림 1에서 좌측에 위치한 Driver Program이 병렬 처리 구조의 Master Node 역할을 수행하며, 데이터를 처리할 때 SparkContext라는 객체를 생성한다. 생성한 객체는 Cluster Manager를 통해 데이터를 병렬 처리하는 Worker Node로 전달되고, Worker Node에서 병렬로 데이터를 처리한다. 처리된 결과는 Master Node로 반환되어 사용자가 결과를 확인할 수 있다. 그림의 중간에 Cluster Manager가 앞서 설명한 Yarn, Kubernetes와 같은 리소스 매니저로 역할을 수행한다[5].

Spark에서는 RDD(Resilient Distributed Dataset), Dataframe, Dataset와 같은 자료구조를 지원하고 있다. 초기에는 RDD를 이용하여 Worker Node에 데이터를 공유하고, 처리하는 방법을 사용했지만, 최근에는 SQL로 데이터를 처리하기 위해 테이블 및 스키마 구조의 형태를 가지는 Dataframe과 컴파일 시 오류를 확인할 수 있는 Dataset을 이용한다[6].

2. Spark Structured Streaming

Spark에서 실시간으로 데이터를 처리할 수 있는 스트리밍 기능은 Spark Structured Streaming으로 기술 이름이 정의된다. 초기에는 RDD 형태로 데이터를 실시간 처리하기 위해 Dstream Streaming이 많이 사용되었지만, 최근에는 Dataframe 형태로 데이터를 실시간 처리할 수 있는 Structured Streaming이 많이 사용되고 있다. 그림 2는 Spark Structured Streaming의 기본 처리 과정을 보여준다.

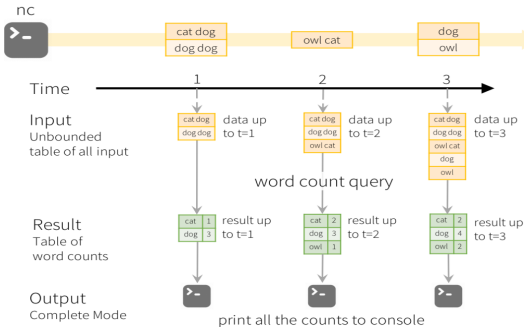


그림 2. Spark Structured Streaming의 기본 처리 과정
Fig. 2. Basic processing of Spark Structured Streaming

그림 2의 nc는 netcat 서버를 의미하여, 실시간으로 데이터를 입력하는 부분이다. nc에서 데이터를 입력하

면, Dataframe 형태로 데이터가 구성되고, 입력된 데이터를 집계(Aggregation) 연산을 수행하여 실시간으로 처리 결과를 확인할 수 있다. 그림의 예시는 입력한 문자열을 띄어쓰기를 기준으로 단어를 분리하고, 분리된 단어의 개수를 집계하는 워드 카운팅 연산이 사용되었다. 특히, Output에 사용된 "Complete" 모드는 새로운 데이터가 입력되더라도 이전에 입력한 데이터를 다시 불러와서 함께 집계 연산을 수행한다. 하지만, "Append" 모드는 새로운 데이터가 입력되면, 이전 데이터는 무시하고, 입력된 데이터만 처리하여 연산을 수행한다[7].

3. Spark Machine Learning

Spark에서 다양한 머신러닝 라이브러리를 지원하는 기능은 MLlib(Machne Learning Library)로 기술 이름이 정의된다. 지원하는 라이브러리는 회귀분석, 의사 결정 트리, 랜덤 포레스트 등 다양하게 있으며, 최근 TensorFlow와 같은 딥러닝 프레임워크도 사용이 가능하다. 본 논문에서는 회귀분석 중 시계열 형태의 데이터가 구성되어 있을 때 특정 값의 상관관계를 확인하고, 예측할 수 있는 선형회귀(LinearRegression) 라이브러리를 사용했다. 종속 변수 Y는 예측값, 독립 변수 X가 한 개 이상일 경우 다변량 변수가 되고, 이를 수식(1)과 같이 표현할 수 있기 때문에 특정 Y 값을 예측하는 예측 모델링을 수행할 수 있다[8].

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_p X_{\pi} + \epsilon_i \quad (1)$$

Ⅲ. 실시간 처리 및 예측 시스템 소개

본 논문에서 실시간으로 주가를 예측하기 위해 개발된 시스템의 전체 구조는 그림 3과 같다.

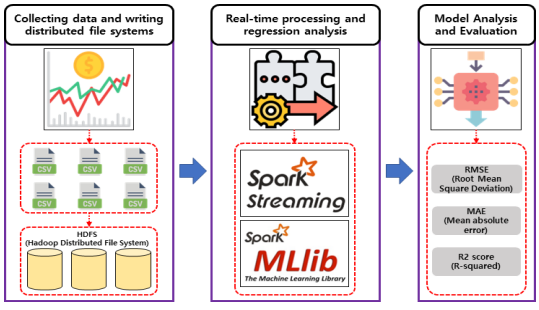


그림 3. 실시간 주가 예측 시스템 구조
Fig. 3. Real-time stock price predicting system structure

첫 번째, Python 라이브러리에서 제공하는 yfinance를 활용하여 데이터를 수집하고, CSV 포맷 형태로 저장한다. 저장된 데이터를 8:2 비율로 학습 및 테스트용으로 분할하고, HDFS에 저장한다. 두 번째, 학습 데이터를 입력값으로 설정하고, MLlib 라이브러리를 활용하여 선형회귀 모델을 생성한다. 그리고 생성된 모델을 실시간으로 불러와 테스트 데이터를 입력값으로 설정하고, Spark Structured Streaming 라이브러리를 활용하여 실시간으로 주가를 예측한다. 마지막으로, 처리가 완료되는 결과들을 CSV 포맷 형태로 다시 HDFS에 저장하여 RMSE, MAE, R2 Score와 같은 모델의 성능을 평가한다.

1. 데이터 수집 및 저장

모델 생성 및 주가 예측에 사용된 데이터는 Python 프로그래밍 언어에서 제공하는 yfinance 라이브러리를 통해 "apple" 기업의 데이터를 다운로드하고, 8:2 비율로 데이터를 분할했다. 그림 4는 데이터 수집 및 데이터 분할에 사용된 소스코드를 나타낸다.

```
df = pdr.get_data_yahoo("aapl", start='1980-12-12')
file_name = "apple_stock.csv"
file_path = "/home/hadoop/struct_streaming_files/"
df.to_csv(file_path+file_name, header=True)

put = Popen(["hadoop", "fs", "-put", "-f", file_name, "/"],
        stdin=PIPE, bufsize=-1)
put.communicate()

for i in range(1, 22): # 2021 ~ 2022년도 테스트 데이터로 분할
  start_ran = "{}-{}-{}".format("2021", str(month_index).zfill(2),
"01")
  end_ran = "{}-{}-{}".format("2021", str(month_index).zfill(2),
"29")
  df = pd.read_csv("apple_stock.csv", encoding="utf-8")
  random_df = df[df['Date'].between(start_ran, end_ran)]
  month_index = month_index + 1

random_df.to_csv("./refined_csv_files_3/streaming_test_{}.csv"
.
  format(str(i).zfill(2)), encoding="utf-8", index=False)
```

그림 4. 데이터 수집 및 분할에 사용된 소스코드
Fig. 4. Source code used for data collection and splition

yfinance 라이브러리에 사용되는 pandas_datareader와 HDFS에 데이터를 저장할 때 쉘 터미널에서 실행되는 명령어를 입력받기 위한 subprocess 라이브러리를 임포트한다. 그리고 1980년도 12월 12일부터 2022년 9월 30일까지 데이터를 다운로드받고, to_csv 함수를 통해 데이터를 로컬 파일 시스템에 저장한다. 저장 후 subprocess의 Popen 함수를 통해 HDFS에 데이터를 분산 저장한다. 학습 데이터는 엑셀 프로그램을 통해

1980년도부터 2020년도까지만 저장하고, 나머지는 삭제했으며, 테스트용 데이터는 학습 데이터에서 삭제한 부분인 2021년도부터 2022년도 9월까지를 한 달을 기준으로 나누어 하나의 파일씩 구성했다.

2. 선형회귀 모델 생성

선형회귀 모델은 PySpark의 LinearRegression 라이브러리를 사용했으며, VectorAssembler 라이브러리를 통해 모델 생성에 필요한 독립 변수를 지정했다. 그림 5는 선형회귀 모델 생성에 사용된 소스코드를 나타낸다.

```
original = spark.read.csv('hdfs://hadoop-name:9000/
            apple_stock_train.csv',
header=True,inferSchema=True)

data = original_data.drop("Date")

feature_columns = data.columns[:-1]
assembler = VectorAssembler(inputCols = feature_columns,
outputCol = 'features')
data_2 = assembler.transform(data)

algo = LinearRegression(featuresCol="features", labelCol="Adj
Close")
model = algo.fit(data_2)

model.write().overwrite().save("hdfs://hadoop-name:9000/reg
_model")
```

그림 5. 선형회귀 모델 생성에 사용된 소스코드
Fig. 5. Source code used to create a linear regression model

HDFS에 저장된 apple_stock_train.csv 파일을 Spark의 read.csv 함수를 통해 읽어오고, “Date” 컬럼은 삭제한다. 삭제된 Dataframe의 마지막 컬럼인 “Adj Close”는 선형회귀 모델의 예측할 종속 변수로 설정하기 때문에 제외하고, 나머지 컬럼들을 다변량 독립 변수로 설정하기 위해 VectorAssembler 함수를 사용했다. 그리고 LinearRegression 함수를 통해 선형회귀 모델을 생성하고, HDFS의 “/reg_model” 경로에 모델을 저장한다. 그림 6은 생성된 선형회귀 모델이 저장된 HDFS의 “/reg_model” 디렉토리의 파일 목록들을 나타낸다.

```
hadoop@hadoop-name:~/struct_streaming_files$ hadoop fs -ls /
Found 7 items
-rw-r--r--   3 hadoop supergroup  268435456 2022-06-20 22:36 /DummyData
-rw-r--r--   3 hadoop supergroup     139689 2022-09-28 15:56 /apple_stock.csv
drwxr-xr-x   - hadoop supergroup          0 2022-06-20 22:38 /ectest
drwxr-xr-x   - hadoop supergroup          0 2022-09-01 22:02 /flume
drwxr-xr-x   - hadoop supergroup          0 2022-10-05 16:30 /reg_model
drwxr-xr-x   - hadoop supergroup          0 2022-09-30 16:29 /streaming
drwxr-xr-x   - hadoop supergroup          0 2022-06-20 22:35 /test
hadoop@hadoop-name:~/struct_streaming_files$ hadoop fs -ls /reg_model
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2022-10-05 16:30 /reg_model/data
drwxr-xr-x   - hadoop supergroup          0 2022-10-05 16:30 /reg_model/metadata
```

그림 6. 생성된 선형회귀 모델 저장 확인
Fig. 6. Verifying and storing generated linear regression models

3. 실시간 주가 예측

실시간 주가 예측은 PySpark의 Spark Structured Streaming 라이브러리를 사용했으며, Dataframe 형태의 데이터를 실시간으로 입력받으면, 입력받은 데이터를 배치처리하여 처리 결과를 즉시 반환한다. 그림 7은 실시간 주가 예측에 사용된 소스코드를 나타낸다.

```
original_schema = StructType([
    StructField('Date', DateType(), True),
    StructField('High', DoubleType(), True),
    StructField('Low', DoubleType(), True),
    StructField('Open', DoubleType(), True),
    StructField('Close', DoubleType(), True),
    StructField('Volume', DoubleType(), True),
    StructField('Adj Close', DoubleType(), True)
])

original = spark.readStream.
    format("csv").schema(original_schema)
    .option("header", True).option("maxFilesPerTrigger", 1) \
    .load("hdfs://hadoop-name:9000/streaming/")
original.isStreaming

df = original.select("Date", "High", "Low", "Open", "Close",
                     "Volume", "Adj Close")
data = df.drop("Date")

feature_columns = data.columns[:-1]
assembler = VectorAssembler(inputCols = feature_columns,
                            outputCol = 'features')
data_2 = assembler.transform(data)

loaded_model = LinearRegressionModel.load
               ("hdfs://hadoop-name:9000/reg_model")
predictions = loaded_model.transform(data_2)

result_df = predictions.select("Adj Close", "Prediction")

result_df.writeStream \
    .trigger(processingTime='10 seconds') \
    .outputMode("append") \
    .format("csv") \
    .option("path", "hdfs://hadoop-name:9000/result_df") \
    .option("checkpointLocation",
            "hdfs://hadoop-name:9000/checkpoints") \
    .start() \
    .awaitTermination()
```

그림 7. 실시간 주가 예측에 사용된 소스코드
Fig. 7. Source code used for real-time stock price predicting

앞선 2절에서는 read.csv 함수로 데이터를 읽어왔지만, 실시간으로 데이터를 읽어올 때는 Dataframe의 스키마 구조(컬럼명, 컬럼데이터 형, Null 값 유무)를 미리 지정 하여야 하므로 StructType() 함수를 통해 스키마를 미리 지정했다. 그리고 readStream 함수를 통해 실시간으로 CSV 데이터를 Dataframe 형태로 읽어온다. 읽어올 파일은 3장의 1절에서 생성한 테스트용 데이터이며, 읽어 올 때 “maxFilesPerTrigger” 옵션을 1로 지정하여 하나의 파일 읽어 올 수 있게 구성했다.

읽어온 테스트용 데이터는 3장의 2절에서 구성한 처리 과정을 동일하게 거치고, HDFS에 저장되어있는 선형 회귀 모델을 불러온다. 불러온 모델에 테스트용 데이터를 실시간 입력값으로 설정하면, “Adj Close” 값을 예측하여 Dataframe 형태로 저장하고, writeStream 함수를 통해 HDFS의 “/result_df”에 재저장한다. 예측에 필요한 처리 시간을 무한히 주면, 실시간 처리가 원활하지만, 현재 구성되어있는 전체 Worker Node의 메모리에 영향을 주기 때문에 10초에 한 번씩 처리하는 트리거 옵션을 추가로 구성했다.

그림 8은 실시간으로 테스트 데이터가 입력되어 배치 처리된 결과를 저장할 때 서버 장애에 대비하여 손실되는 데이터를 임시 보관하기 위한 “/checkpoints” 디렉토리와 처리된 실제 예측 결과를 CSV 형태로 저장하는 “/result_df” 디렉토리의 모습을 보여준다. 그림 9는 실시간 주가 예측을 통해 배치 처리되어 나타나는 콘솔 화면의 결과를 보여준다. 총 21개의 테스트 데이터가 1분 간격으로 입력되기 때문에 배치 횟수는 0부터 시작하여 20까지 수행되었다. 0번, 1번째 배치 처리 결과는 그림으로 표현하고, 중간 결과를 생략한 다음 마지막 배치 처리 결과인 20번째가 출력된 모습을 보여준다.

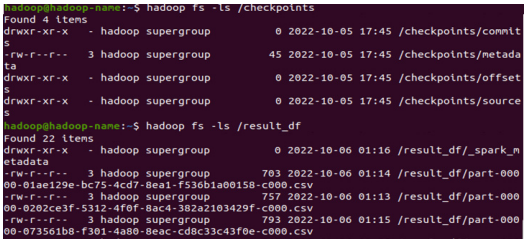


그림 8. 실시간 주가 예측을 통해 저장된 결과 파일 확인
Fig. 8. Check saved results through real-time stock price prediction

4. 모델 평가

모델 평가는 3절에서 수행한 실시간 예측을 통해 결과들이 저장되어있는 HDFS의 “/result_df” 디렉토리 내에 CSV 파일들을 모두 불러와 합병한다. 그리고 실제값이 존재하는 컬럼인 “Adj Close”과 예측하여 생성된 컬럼인 “Prediction”을 비교한다.

비교 및 평가에 사용된 성능 지표는 RMSE(Root Mean Square Deviation), MAE(Mean absolute error), R2(R-Squared) Score가 사용된다. RSME는 모델의 손실함수로 실제값과 예측값의 차이인 오차들의 제곱 편균으로 정의되고, MAE는 실제값과 예측값의 차이인 오차들의 절대값 평균으로 정의되며, R2 Score는 분산 기반 예측 성능 평가로 RSME, MAE와 다르게 상대적으로 어느 정도의 성능을 가지고 있는 모델인지에 대한 직관적인 판단이 가능하다. RMSE와 MAE는 0에 가까울수록 성능이 우수하고, R2 Score는 1에 가까울수록 성능이 우수하다. 그림 10은 PySpark의 Mllib 라이브러리 중 RegressionEvaluator 함수를 통해 3가지 지표를 평가한 결과의 모습을 보여준다.

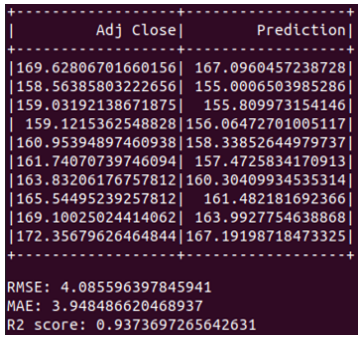


그림 10. RMSE, MAE, R2 Score 성능 평가 결과
Fig. 10. RMSE, MAE, R2 Score performance evaluation results

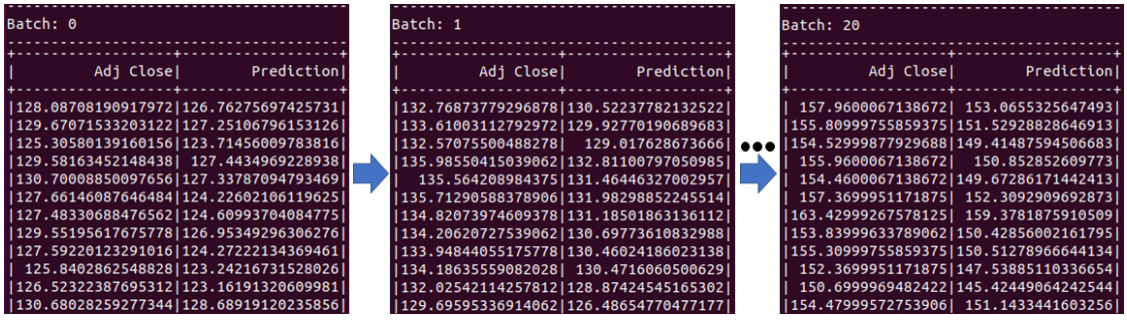


그림 9. 실시간 주가 예측을 통해 출력된 배치 처리 결과
Fig. 9. Batch processing results output through real-time stock price prediction

Ⅳ. 결 론

본 논문에서는 분산 및 병렬 처리 기술 중 Spark를 기반으로 스트리밍과 머신러닝 기능을 통해 실시간으로 주가를 예측하는 연구를 진행했다. 모델의 성능 평가 결과 RMSE와 MAE 지표는 낮은 결과를 기록했지만, R2 Score 지표는 0.93이라는 높은 수치를 기록했다. 회귀 분석은 독립 변수 X가 예측하고자 하는 종속 변수 Y에 영향을 주기 때문에 데이터의 연관성이 있는 데이터일수록 결과가 좋다. 본 연구에서 사용된 데이터 중 독립 변수는 주가 시장에서 기록되는 시작가, 하향가, 고가 등 거래량과 같은 관련성이 적은 데이터로 구성되었기 때문에 모델의 성능은 좋지 못하다. 따라서 향후 종속 변수와 독립 변수가 연관성이 많은 데이터를 수집하고, 모델의 성능을 향상시킬 수 있는 연구를 진행하고자 한다.

References

[1] Dong-Jin Shin, Ji-Hun Park, Ju-Ho Kim, Kwang-Jin Kwak, Jeong-Min Park, Jeong-Joon Kim, "Big Data-based Sensor Data Processing and Analysis for IoT Environment", The Journal of the Institute of Internet, Broadcasting and Communication (IIBC), Vol. 19, No. 1, pp. 117-126, Feb 2019.
DOI: https://doi.org/10.7236/JIIBC.2019.19.1.117

[2] Salman Salloum, Ruslan Dautov, Xiaojun Chen, Patrick Xiaogang Peng, Joshua Zhexue Huang, "Big data analytics on Apache Spark", International Journal of Data Science and Analytics, Vol. 1 No. 3, pp. 145-164. Oct 2016.
DOI: https://doi.org/10.1007/s41060-016-0027-9

[3] Nikitha Johnsirani Venkatesan, ChoonSung Nam, Dong Ryeol Shin "Deep Learning Frameworks on Apache Spark: a review. IETE Technical Review, Vol. 36, No. 2, pp. 164-177. May 2018.
DOI: https://doi.org/10.1080/02564602.2018.1440975

[4] Apache Spark Cluster Mode Overview Guide, "https://spark.apache.org/docs/latest/cluster-overview.html"

[5] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, Ion Stoica, "Spark: Cluster computing with working sets", In 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10), 2010.

[6] Apache Spark DataFrames and Datasets Guide, "https://spark.apache.org/docs/latest/sql-programming-guide.html"

[7] Apache Spark Streaming Programming Guide, "https://spark.apache.org/docs/latest/streaming-programming-guide.html"

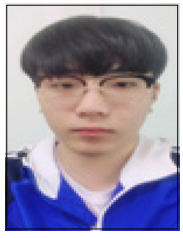
[8] Apache Spark Machine Learning Library Guide, "https://spark.apache.org/docs/latest/ml-guide.html"

저 자 소 개

신 동 진(준회원)

• Dong-Jin Shin received BS in department of computer science and MS in department of smart manufacturing engineering at the Korea Polytechnic University in 2018 and 2020. He is currently studying PhD in department of computer science at AnYang University. His research interests include Big Data, Internet of Things(IoT), Artificial Intelligence (AI), Distributed File Systems.

황 승 연(준회원)



• Seung-Yeon Hwang is received his BS in department of computer science at Korea Polytechnic University in 2019. He is currently studying combined MS/PhD in department of computer science at Anyang University. His research interests include Big Data, Data Analysis, Machine Learning and Deep Learning.

김 정 준(정회원)

• Jeong-Joon Kim received BS and MS in computer science at Konkuk University in 2003 and 2005, respectively. In 2010, he received PhD in at Konkuk University. He is currently a professor in software major ICT Convergence Engineering at AnYang University. His research interests include Database Systems, Big Data, Semantic Web, Geographic Information Systems (GIS) and Ubiquitous Sensor Network (USN), etc.

※ 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2022R1F1A1062953).