**Big Data Pipeline Project – Student Performance Factors**

Christina Vosnak

Data Science, Bellevue University

DSC 650

Professor Ismaily

June 1st, 2025

**Introduction**

For this project, I will be analyzing a dataset focused on student exam performance by designing and implementing a comprehensive data pipeline utilizing NiFi, HDFS, Hive, and Spark. The objective is to demonstrate the seamless integration of these big data technologies to ingest, store, transform, and analyze educational data on a scale.

The pipeline begins with Apache NiFi, which will be responsible for ingesting the raw dataset and transferring it into the Hadoop Distributed File System (HDFS). HDFS serves as the primary storage layer, ensuring the data is reliably and efficiently stored in a distributed manner. From there, I will use Apache Hive to create an external table on top of the data in HDFS. Hive will enable SQL-like querying capabilities, making it easier to interact with the data and explore meaningful patterns. Finally, I will utilize Apache Spark to perform deeper data analysis and transformation, allowing me to derive actionable insights related to academic factors that may influence student exam scores.

**Dataset**

The dataset I have chosen for this project is sourced from Kaggle at the following link: <https://www.kaggle.com/datasets/lainguyn123/student-performance-factors>. It contains relevant information from 6,000 students, their exam scores, and study habits. There are 20 unique features to analyze within the data, allowing for an adequate amount of data to simulate a big data architecture to process information in a modified pipeline.

**Pipeline Overview**

Drawing from examples provided by the professor, I constructed my data pipeline using the following four core components: Apache NiFi, HDFS, Hive, and Apache Spark. These components were selected based on their coverage in previous coursework and their demonstrated interoperability in distributed data environments. From the Microsoft Teams channel, I followed the professors instructions for the simple approach and ensured each step of the pipeline successfully worked.

To begin, I utilized the Example\_Project.json template as a reference to build a custom NiFi data flow tailored to my dataset. This NiFi processor was configured to ingest the dataset directly from my GitHub repository, which was created specifically for this course. Once retrieved, the dataset was routed and written into a newly created HDFS directory named /projectdata. I also renamed the file ‘mydata.csv.’ This ingestion process marked the first successful handoff in the pipeline and ensured that the data was accessible to downstream components.

After confirming the data's successful placement in HDFS, I closed NiFi and transitioned to working within the Hadoop ecosystem via the command-line interface. Using the terminal, I initiated Hive and constructed a table that pointed to the data stored in the /projectdata directory. The schema design and table creation were modeled after the assignment covered in week 3 of the course, where we first explored the Hive system. This marks the third successful component in creating the data pipeline for this project.

Following the Hive setup, I launched a Spark session with Hive support enabled to begin data analysis. I wrote PySpark queries to extract and examine key patterns from the dataset, building upon techniques learned during Week 4 of the course. These queries allowed me to filter, group, and analyze student performance metrics based on several academic and socio-demographic features. For the examples in this project, I calculated the average hours studied by the students. I also counted the number of female and male students in the dataset. Finally, I used these two queries to understand if there was a difference in average exam score by gender. While this just scratches the surface of building a pipeline to gather valuable information about a dataset, querying the data in Spark marks the completion of the fourth component of the data pipeline for this project.

Additional technical details, command executions, and visual outputs of each component's role in the pipeline are provided in the Screenshots section of this report.

**Issues Encountered**

The only major issue I encountered was with NiFi. Originally following the professor’s pre-designed flow, NiFi was unable to connect to PutHDFS. In order to get the flow to work, I needed to change the Hadoop configuration resources section from ubuntu to my username. That allowed the flow to work successfully to ingest the data into HDFS. From there, building the table and querying it was rather simple thanks to the guides provided by the professor.

**Screenshots**

1. Using Nifi to Download data from github and put it in the HDFS /projectdata directory

For the first part of the project, I followed the example project json template to use Nifi and upload the data to HDFS. I first opened Nifi containers and imported the example project.json file to Nifi as a process group. I then updated the flow. In the InvokeHTTP, I linked my raw data that I uploaded to github. In PutHDFS, I set the directory to ‘/projectdata’, a folder I made in HDFS exclusively for this project and changed ‘ubuntu’ to my username to ensure the flow worked.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Checking data is loaded into HDFS and change file name

Then in HDFS, I created a folder titled /projectdata to run my NiFi flow data into. I changed the file name to mydata.csv to ensure easy access to the file. I then closed Nifi to work exclusively in Hadoop containers for the remainder of the project.

A black screen with text on it

AI-generated content may be incorrect.

A black screen with text

AI-generated content may be incorrect.

A black screen with text and green lights

AI-generated content may be incorrect.

1. Building a Table in Hive

Next, I opened a hive session to create a table based on mydata.csv that is loaded into HDFS. I followed the example from week 3. To show the table was successfully created, I displayed the first 10 rows.

A black background with a black screen

AI-generated content may be incorrect.

1. Executing search queries using PySpark

Finally, I exited hive to start a Spark session to query the data. I first checked that Hive is supported by Spark and then conducted 3 queries on the data. The first found the average hours studied by the students. The second found the count of female and male students, and the third found the average exam score by gender.

A screen shot of a computer

AI-generated content may be incorrect.

A computer screen shot of a black screen

AI-generated content may be incorrect.

**Code**

|  |  |
| --- | --- |
| Open terminal and navigate to directory | cd dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase |
| Start Docker Containers | docker-compose up -d |
| Go back to enter NiFi directory | cd .. cd nifi |
| Start NiFi | /bin/bash nifi-\*/bin/nifi.sh start |
| Access NiFi User Interface | Localhost:8443/nifi/ |
| Login Using Credentials from Week 1 | Username and Password |
| Drag Process Group to NiFi flow | Import Example\_Project.json |
| InvokeHTTP | Change raw file to raw file on my GitHub |
| PutHDFS | Change directory to /projectdata |
| Hadoop Configuration Resources | Change ‘ubuntu’ to ‘cvosnak’ |
| Navigate to HDFS in terminal | cd.. cd hadoop-hive-spark-hbase |
| Enter master container | docker-compose exec master bash |
| List current folders | hdfs dfs -ls / |
| Create /projectdata folder | hdfs dfs -mkdir /projectdata |
| Check folder has been created | hdfs dfs -ls / |
| Return to NiFi User Interface | Run once InvokeHTTP and Start PutHDFS to load data to HDFS |
| Check data loaded into HDFS | hdfs dfs -ls /projectdata |
| Change name of file to ‘mydata.csv’ | hdfs dfs -mv /projectdata/53f3b3c7-94e9-4e60-956f-02036c524b71 /projectdata/mydata.csv |
| Check file is changed to ‘mydata.csv’ | hdfs dfs -ls /projectdata |
| Stop NiFi | Exit; cd..; cd nifi; /bin/bash nifi-\*/bin/nifi.sh stop |
| Return to master container and start a hive session | cd .. cd hadoop-hive-spark-hbase; docker-compose exec master bash; hive |
| Create table | CREATE TABLE students(  Hours\_Studied INT,  Attendance INT,  Parental\_Involvement STRING,  Access\_to\_Resources STRING,  Extracurricular\_Activities STRING,  Sleep\_Hours INT,  Previous\_Scores INT,  Motivation\_Level STRING,  Internet\_Access STRING,  Tutoring\_Sessions INT,  Family\_Income STRING,  Teacher\_Quality STRING,  School\_Type STRING,  Peer\_Influence STRING,  Physical\_Activity INT,  Learning\_Disabilities STRING,  Parental\_Education\_Level STRING,  Distance\_from\_Home STRING,  Gender STRING,  Exam\_Score INT)  ROW FORMAT DELIMITED  FIELDS TERMINATED BY ','  STORED AS TEXTFILE  tblproperties("skip.header.line.count"="1"); |
| Load Data from HDFS | LOAD DATA INPATH '/projectdata/mydata.csv' INTO TABLE students; |
| Run query to view data (only showing 10 rows for simplicity) | SELECT \* FROM students LIMIT 10; |
| Exit Hive | exit; |
| Start a Spark Session | $SPARK\_HOME/bin/spark-shell --master yarn --driver-memory 2g --executor-memory 1g --executor-cores 1 |
| Check that Hive is supported by Spark | spark.conf.get("spark.sql.catalogImplementation") |
| Query data from Hive (only showing 10 rows for simplicity) | val df = spark.sql("SELECT \* FROM students LIMIT 10") |
| Show the dataframe | df.show() |
| Calculate Average Hours Studied | spark.sql("SELECT AVG(hours\_studied) AS avg\_hours FROM students").show() |
| Count Number of Female and Male Students | spark.sql("SELECT gender, COUNT(\*) AS count FROM students GROUP BY gender").show() |
| Find Average Exam score by gender | spark.sql("SELECT gender, AVG(exam\_score) AS avg\_score FROM students GROUP BY gender").show() |
| Exit scala | :quit |
| Exit master container | exit |
| Close Containers | docker-compose down |

**Conclusion**

Overall, just from some simple queries about the data, we found valuable information by building this big data pipeline. In the spark queries, we found that students spent on average 20 hours studying overall. We also found that the gender of the students was split 2793 female to 3814 males. However, when looking at average exam scores between the two genders, both males and females averaged at 67%. As stated previously, this is just a small glimpse into the valuable insights that can be gathered by creating a data pipeline. By using the NiFi, HDFS, Hive and Spark components, we can successfully process data, create tables and run queries as data scientists in the real world.