Spark for Big Data Preprocessing and Statistical Analysis

Stephen Oduro
Class: SAT5165 - Big Data Analytics

October 23, 2024

Abstract

This project utilizes Apache Spark and distributed computing to preprocess and perform statistical analysis on the Cardiovascular Disease Dataset, which comprises over 70,000 patient records. By leveraging Spark's scalability across two virtual machines (VMs), the project efficiently handles data preprocessing tasks such as cleaning, imputation, and encoding, followed by statistical analyses including correlation analysis and logistic regression. The objective is to extract actionable insights to predict cardiovascular risk factors effectively, demonstrating the advantages of distributed computing in managing large-scale datasets.

Contents

1 Introduction

In the era of big data, efficiently processing and analyzing large datasets is paramount for deriving meaningful insights. Apache Spark, a powerful distributed computing framework, offers scalability and speed that are essential for handling extensive data preprocessing and statistical analysis tasks. This project focuses on applying Spark to the Cardiovascular Disease Dataset, aiming to predict cardiovascular risk based on various patient factors.

2 Objectives

The primary objectives of this project are:

1. **Efficient Data Preprocessing:** Utilize Apache Spark to clean the Cardiovascular Disease Dataset by handling missing values, removing duplicates, and encoding categorical variables.

2. **Statistical Analysis:** Perform correlation analysis and logistic regression to identify significant predictors of cardiovascular disease.

3. **Distributed Computing:** Implement a distributed Spark environment using two virtual machines to enhance computational efficiency and reduce processing time.

4. **Insight Extraction:** Derive meaningful insights to aid in predicting cardiovascular risk based on patient data.

3 Implementation Strategy

The project follows a structured approach comprising the following steps:

3.1 Dataset Selection

The Cardiovascular Disease Dataset from Kaggle, containing over 70,000 records with features such as age, gender, cholesterol levels, and blood pressure, was selected for analysis. The dataset is available on GitHub: First-Large-Project Repository.

3.2 Data Preprocessing

- **Cleaning:** Address missing values using mean imputation for numerical features and mode imputation for categorical features.

- **Duplicate Removal:** Identify and eliminate duplicate records to ensure data integrity.

- **Encoding:** Apply StringIndexer and OneHotEncoder to transform categorical variables into numerical formats suitable for analysis.

- **Feature Assembly and Scaling:** Combine encoded categorical features with numerical features using VectorAssembler and apply StandardScaler for feature scaling.

3.3 Statistical Analysis Methods

- **Correlation Analysis:** Compute Pearson correlation coefficients between features to identify relationships between variables.
- **Logistic Regression:** Develop a predictive model to assess cardiovascular risk based on selected features.

3.4 Python Libraries and Tools

- `pyspark` for executing Spark operations.
- `pandas` for initial data exploration.
- `scikit-learn` for implementing regression and correlation techniques.
- `matplotlib` and `seaborn` for visualizing the results.
- `plotly` for interactive visualizations.

4 VM Setup

To facilitate distributed computing, two virtual machines (VMs) were configured as follows:

4.1 Environment Setup

- **Virtual Machines:**
  - **Hadoop 1 (Master VM):** IP Address `192.168.13.122`
  - **Hadoop 2 (Worker VM):** IP Address `192.168.13.123`
- **Access Tools:**
  - **BIG-IP Edge Client:** Used to connect to the Michigan Tech VPN for remote access.
  - **VM Remote Console (VMRC):** Utilized for remote management of VMs.
  - **vSphere Client:** Accessed to launch and manage VMs.

4.2 Apache Spark Configuration

1. **SSH Key Pair Generation:** Generated on Hadoop 1 using:

   ```
   ssh-keygen -t rsa
   ```

   Copied the public key to Hadoop 2's `authorized_keys` using:

   ```
   cat .ssh/id_rsa.pub >> ~/.ssh/authorized_keys
   ```

2. **Spark Installation and Setup:** Spark was already installed on both VMs. Transferred Spark from Hadoop 1 to Hadoop 2 using:

```
cd /opt
tar czf spark.tar.gz spark
scp spark.tar.gz root@hadoop2:/opt
```

On Hadoop 2:

```
cd /opt
tar xvzf spark.tar.gz
```

3. **Starting Spark Services:**

   - **Master VM Only:**

     ```
     /opt/spark/sbin/start-master.sh
     ```

   - **Specific Worker (Hadoop 1):**

     ```
     /opt/spark/sbin/start-worker.sh spark://hadoop1:7077
     ```

   - **All VMs (Master and Workers):**

     ```
     /opt/spark/sbin/start-all.sh
     ```

4. **Submitting Spark Jobs:**

   ```
   /opt/spark/bin/spark-submit --master spark://hadoop1:7077 /home/sat3812/Downloads
   ```

5 Code Explanation and Method

The Python script `large_project.py` was developed to perform data preprocessing and exploratory data analysis using PySpark. Below is an overview of the key components and functionalities of the script:

5.1 Importing Libraries

Essential libraries such as `pyspark`, `pandas`, `matplotlib`, `seaborn`, and `plotly` were imported to facilitate data processing and visualization. `vector_to_array` from `pyspark.ml.functions` was utilized to convert vector columns into array format for easier feature extraction.

5.2 Data Preprocessing

- **Handling Missing Values:** Numerical features were imputed using mean values, while categorical features were imputed using mode values.
- **Duplicate Removal:** Duplicate records were identified and removed to maintain data integrity.
- **Categorical Encoding:** StringIndexer and OneHotEncoder were applied to convert categorical variables (`gender`, `cholesterol`, `gluc`) into numerical formats. The encoded vectors were further transformed into individual scalar columns (`gender_encoded_0`, `gender_encoded_1`, etc.) using `vector_to_array` for accurate feature naming and correlation computation.

- **Feature Assembly and Scaling:** Encoded categorical features were combined with numerical features using VectorAssembler. StandardScaler was applied to standardize the feature set.

5.3 Statistical Analysis

- **Correlation Analysis:** Pearson correlation coefficients were computed between each feature and the target variable (`cardio`) to identify significant relationships.
- **Logistic Regression:** A Logistic Regression model was trained to predict cardiovascular risk based on the processed features. Hyperparameter tuning was performed using CrossValidator with a parameter grid for regularization parameters (`regParam`) and Elastic Net mixing parameter (`elasticNetParam`).

5.4 Visualization

- **Box Plots and Histograms:** Visualizations were generated to understand the distribution and outliers in numerical features.
- **Bar Charts:** Frequency distributions of categorical features were visualized.
- **Correlation Matrix Heatmap:** A heatmap was created to visualize the correlation between different features.
- **Feature Importance Plot:** The coefficients from the Logistic Regression model were plotted to assess feature importance.
- **Confusion Matrix:** A confusion matrix was generated to evaluate the performance of the Logistic Regression model.

6 Performance of the Program

6.1 1. Execution with One VM (Hadoop 1)

- **Configuration:**
  - **Master VM:** Hadoop 1 (`192.168.13.122`)
  - **Worker VM:** Hadoop 1 only
- **Execution:**

```
/opt/spark/bin/spark-submit --master spark://hadoop1:7077 /home/sat3812/Downloads
```

- **Observations:**
  - **Processing Time:** Approximately 3.0 minutes
  - **Consistency:** Repeated runs yielded similar processing times (2.9 to 3.0 minutes), indicating consistent performance.

6.2 2. Enhanced Execution with Two VMs (Hadoop 1 and Hadoop 2)

- **Configuration:**
  - **Master VM:** Hadoop 1 (`192.168.13.122`)
  - **Worker VM:** Hadoop 2 (`192.168.13.123`)
- **Execution:**

```
/opt/spark/bin/spark-submit --master spark://hadoop1:7077 /home/sat3812/Downloads
```

- **Observations:**
  - **Processing Time:** Approximately 30 seconds
  - **Performance Improvement:** The processing time reduced by approximately 90%, showcasing the efficiency gains from distributed computing across two VMs.

6.3 Summary of Performance Gains

- **Single VM Processing:** 3.0 minutes
- **Dual VM Processing:** 30 seconds
- **Performance Boost:** 6x faster processing with two VMs

7 Challenges and Resolutions

7.1 1. Feature Encoding Mismatch

- **Issue:** The script attempted to access individual encoded feature columns (e.g., `gender_encoded_0`) that did not exist in the DataFrame, leading to an `IllegalArgumentExceptio`
- **Resolution:**
  - **Conversion to Array:** Utilized `vector_to_array` to convert encoded vector columns into arrays.
  - **Extraction of Individual Elements:** Extracted each element of the array into separate scalar columns, ensuring that `feature_names` accurately reflected the DataFrame's structure.
  - **Verification:** Implemented checks to ensure that the number of features matched the number of coefficients in the Logistic Regression model, preventing `ValueError` due to mismatched array lengths.

7.2 2. Correlation Matrix Alignment

- **Issue:** The target variable `cardio` was not present in the correlation matrix, causing errors when attempting to access its correlations.
- **Resolution:**
  - **Feature-Target Correlation Computation:** Developed a dedicated function to compute Pearson correlations between each feature and the target variable.
  - **Top Feature Selection:** Identified the top 5 features with the highest absolute correlations with `cardio` for focused analysis.

7.3 3. Spark Warnings

- **Issue:** Encountered warnings related to the native Hadoop library and large plan string representations.
- **Resolution:**
  - **Log Level Adjustment:** Set Spark's log level to `ERROR` to suppress non-critical warnings.
  - **Logger Configuration:** Specifically suppressed warnings from `org.apache.spark` and `org.apache.hadoop.util.NativeCodeLoader` to reduce console clutter without affecting essential logs.

8 Conclusion

The project successfully demonstrated the efficacy of Apache Spark in handling large-scale data preprocessing and statistical analysis tasks. By configuring a distributed environment across two virtual machines, significant performance improvements were achieved, reducing processing time from approximately 3.0 minutes to 30 seconds. The implementation addressed critical challenges related to feature encoding and data alignment, ensuring accurate and efficient analysis. The insights derived from correlation analysis and logistic regression models provide valuable predictive capabilities for assessing cardiovascular risk factors.

9 Future Work

For future enhancements, the following aspects could be explored:

1. **Model Optimization:** Explore advanced machine learning algorithms and hyperparameter tuning to improve prediction accuracy.
2. **Scalability Testing:** Extend the distributed setup to include more VMs and assess the scalability and performance gains.
3. **Real-Time Data Processing:** Implement real-time data ingestion and processing pipelines using Spark Streaming for dynamic analysis.

4. **Deployment:** Develop a user-friendly interface or dashboard to visualize predictions and insights in real-time for practical applications.

10 References

1. Apache Spark Documentation
2. Cardiovascular Disease Dataset on Kaggle
3. PySpark API Documentation
4. GitHub Repository: First-Large-Project

11 Appendix

11.1 GitHub Repository

All code and visualizations generated during this project are available in the GitHub repository at First-Large-Project. The `images` folder within the repository contains all the generated plots and charts used in this report.