Michigan Tech

NAME OF THE STUDENTS: PETER MVUMA

COURSE: INTRODUCTION TO BIG DATA ANALYTICS

COURSE CODE: SAT 5165

ASSIGNMENT: SMALL PROJECT STATISTICAL MACHINE LEARNING FOR BIG DATA ANALYSIS WITH SPARK

DUE DATE: 11/07/2025

Distributed Decision Tree Model for Diabetes Prediction Using BRFSS 2015 Health Indicators

1. Problem Statement

In this small project,

- Apply a machine learning algorithm on a dataset with more than 5000 records
- Use at least 2 Virtual Machines(VMs).
- Evaluate the model's performance using accuracy, AUC, precision, and recall metrics • Compare the computational speed between 1 VM and 2 VMs

2. Project Overview

This project applies a distributed machine learning approach using Apache Spark to predict diabetes risk from health indicators collected through the Behavioural Risk Factor Surveillance System (BRFSS 2015) dataset. The goal was to evaluate the performance and scalability of a Decision Tree model when executed on a Spark cluster configured across two Fedora-based virtual machines. The analysis demonstrates how distributed computing can improve processing speed and manage large datasets efficiently while maintaining interpretability for healthcare applications.

I chose to implement a decision tree model in the project, while my other colleagues chose to implement logistic regression and random forest on the same dataset. This project uses a Decision Tree Classifier. I chose Decision Trees as they are non-parametric models that handle non-linear relationships effectively and provide intuitive interpretability through feature importance ranking.

For details and additional details, find the GitHub link.

3. Python Packages and ML Model Used in the project

- pyspark.sql – for distributed data loading and transformation
- pyspark.ml – for constructing the machine learning pipeline
- pyspark.ml.feature – for vector assembly of feature columns
- pyspark.ml.classification – for implementing the Decision Tree model
- pyspark.ml.evaluation – for computing performance metrics
- time, sys – for measuring runtime and controlling execution flow
- matplotlib, sklearn.metrics – for ROC curve plotting

4. Dataset Description

The dataset we chose contains health-related measurements for more than 250,000 people with a target variable showing whether each person has diabetes or not. Source dataset

The dataset used was derived from the Behavioral Risk Factor Surveillance System (BRFSS 2015), accessible from Kaggle. It includes 253,680 survey records with 21 health-related features such as High Blood Pressure, BMI, General Health, and Age. The target variable, Diabetes_binary, indicates whether a respondent had been diagnosed with diabetes (1) or not (0).
A graphical view of the dataset shows highly imbalanced data, as shown in Figure 01 below.

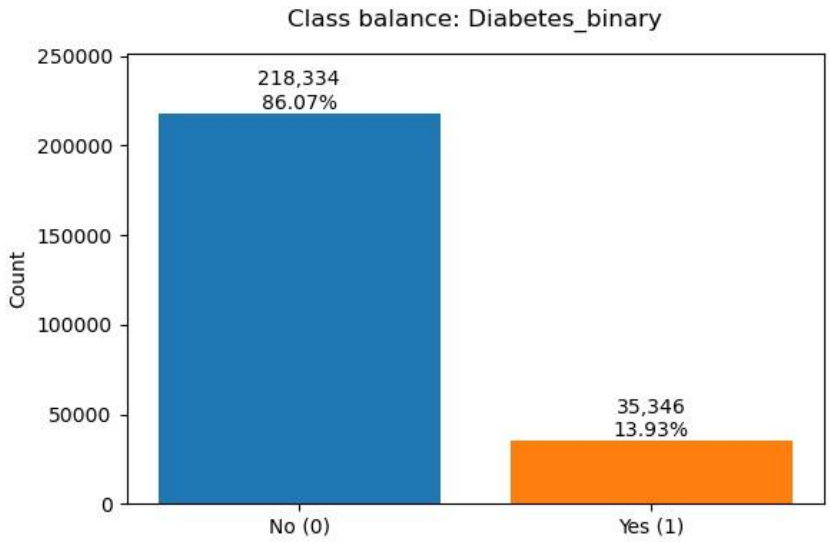


Figure 01: Confirming that the dataset is highly imbalanced, with about 86% non-diabetic cases and 14% diabetic cases. (Source Group work)

5. Data and Preprocessing

- **Cleaning & schema** - Spark infers types; all columns are doubles.
- **Feature Engineering -** Selected numeric columns as features and combined them into a single feature vector using Vector Assembler.
- **Handling Class Imbalance -** Computed inverse-frequency class weights and added a weight column to increase model sensitivity to minority cases, specifically diabetic respondents.
- **Train-Test Split -** Used stratified sampling to allocate 80% of the data for training and 20% for testing, maintaining class proportions.
- **Model Configuration -** Set Decision Tree parameters to maxDepth=8, maxBins=128, minInstancesPerNode=20, and seed=42. Included the weight column to address class imbalance and built a pipeline for feature assembly and classification.
- **Training and Prediction -** Trained the model on distributed partitions across two nodes. Generated predictions for the test set and extracted probabilities using vector_to_array.

6. **Spark Environment and Configuration** - Hadoop Cluster verification and monitoring – **Figure 03.**

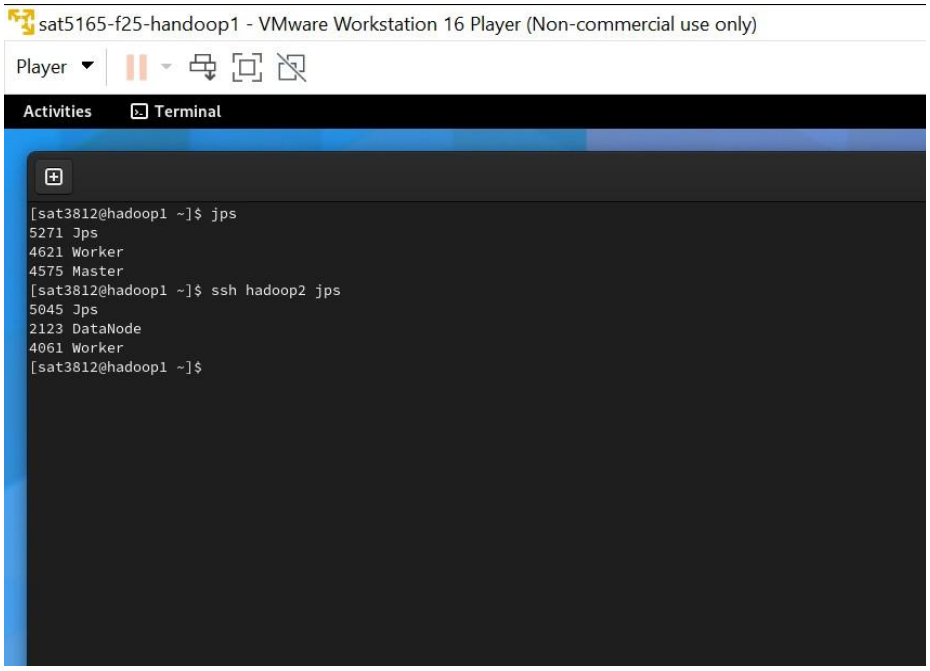


Figure 03 above is a cluster verification and monitoring stage. After configuring Hadoop and Spark, I verified that cluster services were running on both nodes using the 'jps' and 'ssh' commands. The jps command confirmed the expected Hadoop and Spark daemons by listing active Java processes. On the master node (hadoop1), the output included the Spark Master and Worker processes. On the worker node (hadoop2), the Spark Worker and Hadoop DataNode processes were shown. I used ssh hadoop2 jps to check the remote node from the master terminal without manual login. These steps confirmed proper inter-node communication and a correctly configured distributed environment for parallel data processing.

Figure 04: Showing the Spark Master web UI at spark://hadoop1:8080 reporting two active worker nodes with available CPU and memory ready for the work to be deployed.

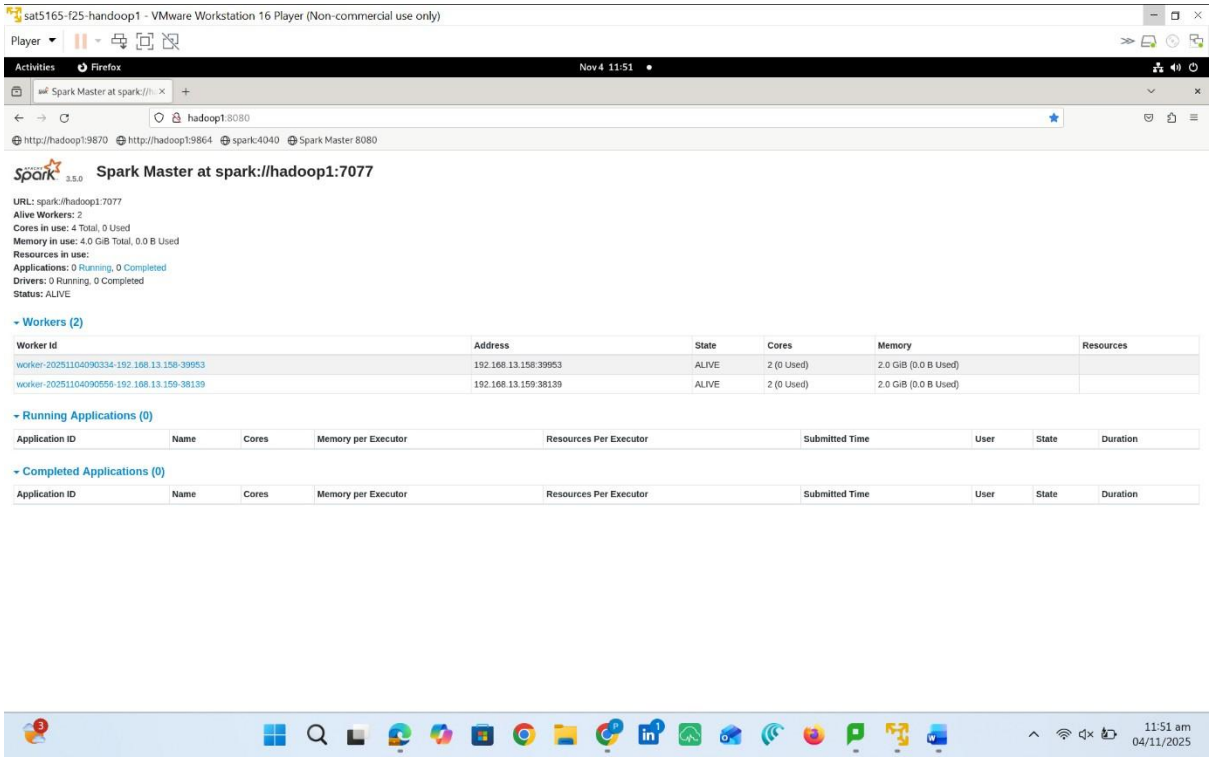
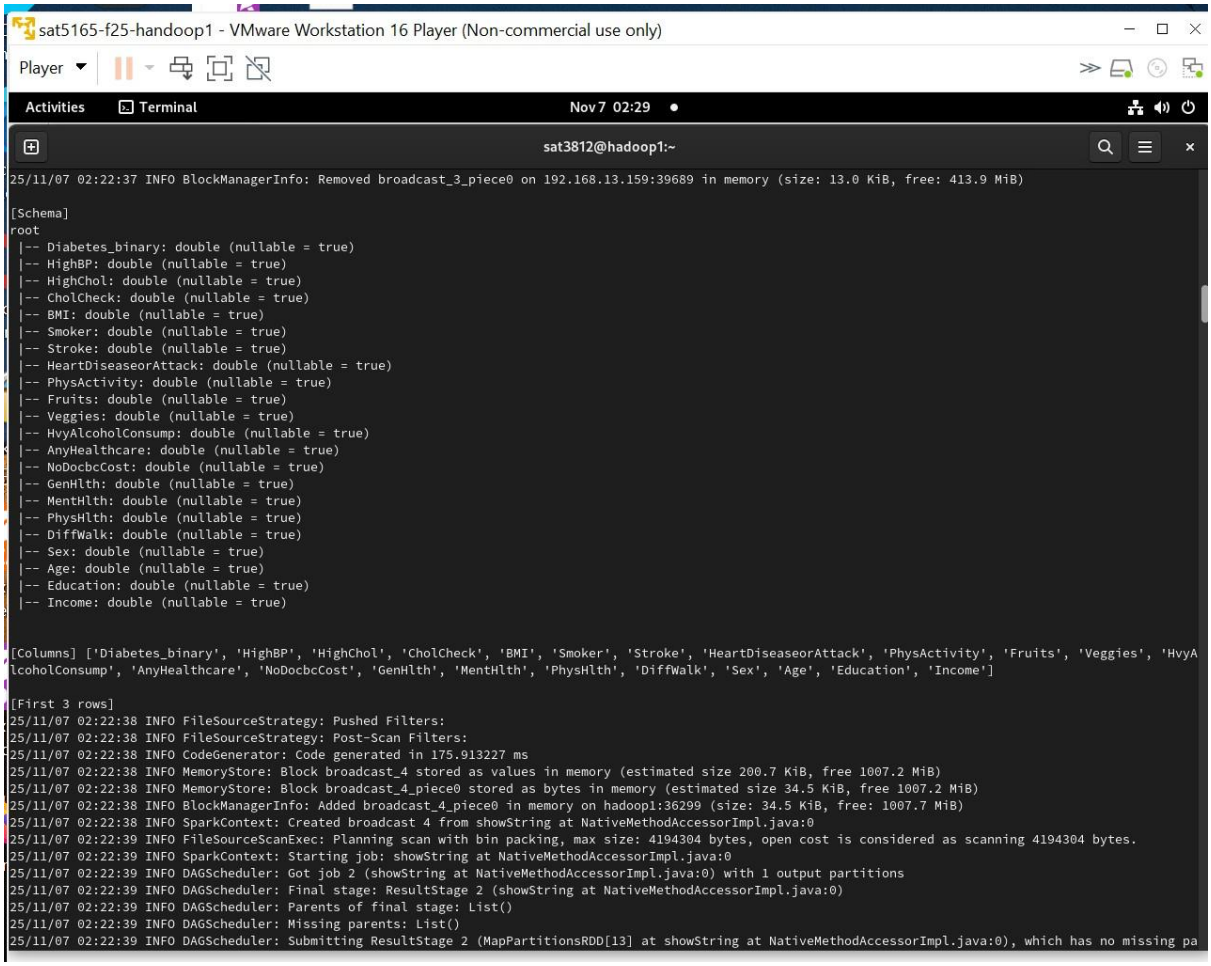


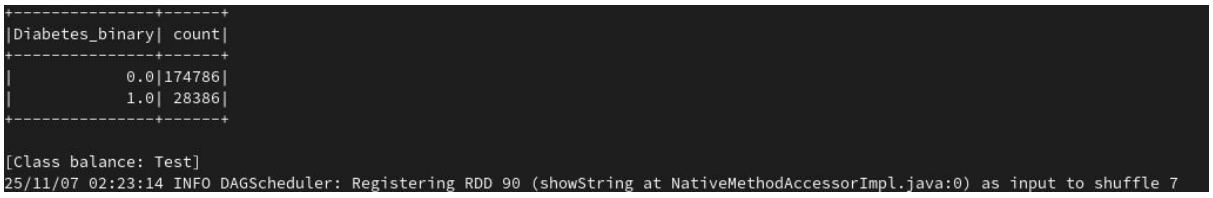
Figure 04 above shows that the Apache Spark cluster was configured in a master–worker setup with two active nodes, each offering 2 CPU cores and 2 GB of memory. This configuration supports parallel data processing and distributed model training, enhancing speed and scalability for large datasets. The Spark Web UI verified that both workers were operational, providing a stable environment for machine learning tasks, including preprocessing and classification.

7. **Running of the Spark job and initiation of the distributed computing work, Figure 05.**



In Figure 05 above, the Spark job successfully validated the dataset by displaying its schema and column structure, confirming accurate data loading and type inference. Subsequently, it initiated the first stages of distributed computation, executing the initial transformations and actions across cluster nodes to prepare the data for model training and analysis

8. **Spark running and showing the class distribution within the test subset is shown in Figure 06.**

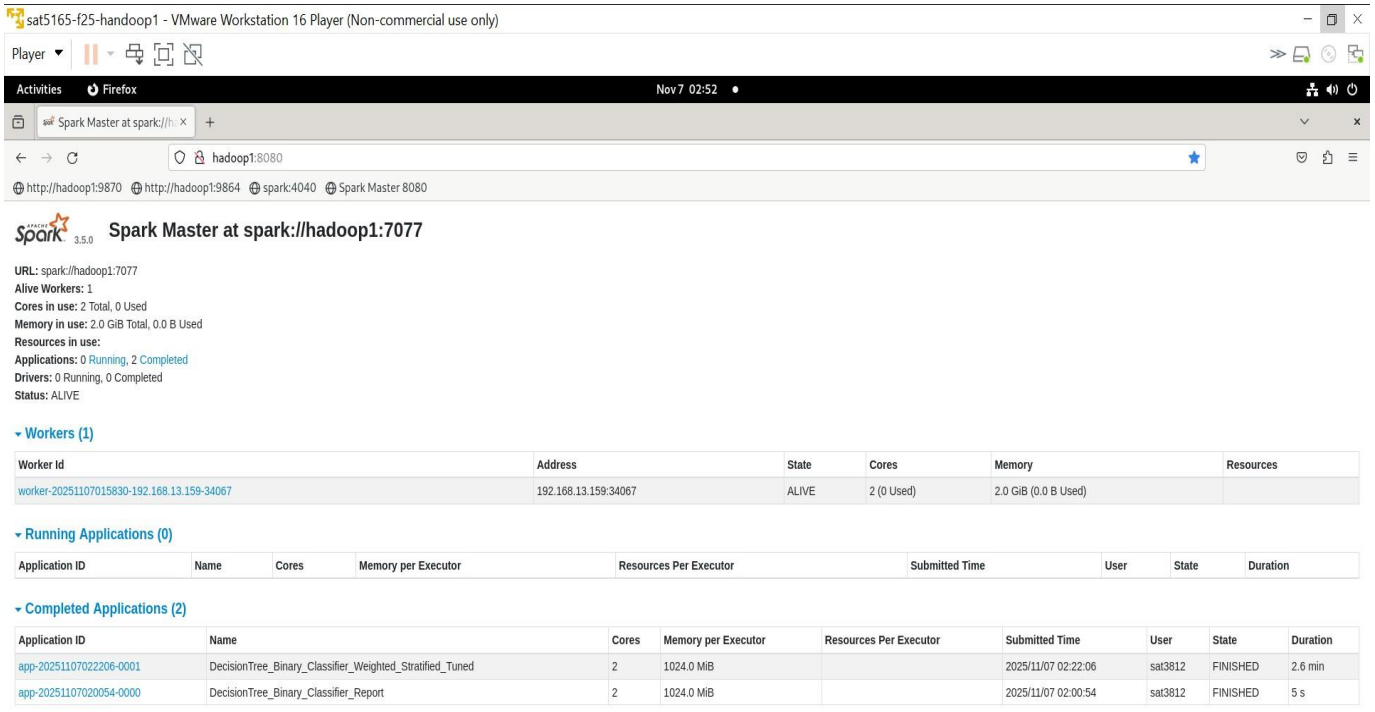


In Figure 06 above, the class distribution within the test subset comprised 174,786 non-diabetic and 28,386 diabetic instances. This indicates that the stratified sampling approach effectively maintained the original class proportions from the full dataset, thereby ensuring representativeness and reliability in subsequent model evaluation.

9. **The feature importance results from the Decision Tree model indicated – Figure 07.**

My Decision Tree model successfully identified the most influential predictors of diabetes, as shown in Figure 07 above. The top contributing features included HighBP (0.4612)**,** General Health (0.2877)**,** and BMI (0.1035)**,** indicating that blood pressure, overall health perception, and body mass index are key determinants of diabetes risk in this dataset. The model achieved a tree depth of 8 with 279 nodes, effectively capturing nonlinear relationships between features during distributed computation. These predictors within this model align with established clinical knowledge.

10. Performance in Computational Speed



11. Performance in Computational Speed and observations

VM Configuration	Hadoop1 only Master local)	Two VMs (distributed)
Actual time taken	~18minutes	~2.6minutes
Observations	Model trained successfully, but slower on large CSV reads	faster due to parallel execution and task splitting
Runtime comparison	Two-VM setup is 6.9 × faster, showing an 85.6 % reduction in computation time	

In the table above, the distributed Spark cluster (two VMs) reduced model training and prediction time from 18 minutes to 2.6 minutes, a 6.9-fold improvement (about 86% faster). This demonstrates Spark's efficiency in parallelizing tasks and distributing data across worker nodes.

12. Model Performance evaluation metrics as shown in the terminal

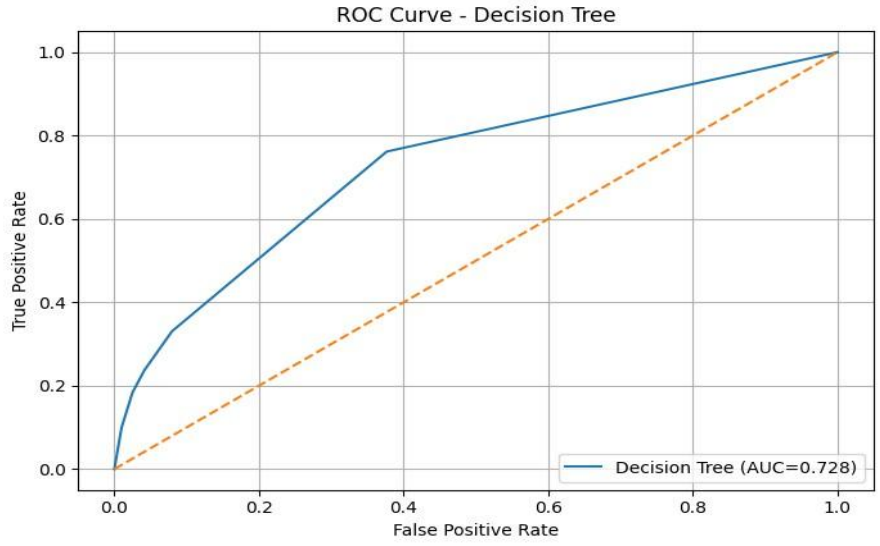
```
25/11/07 02:24:36 INFO DAGScheduler: Job 89 finished: collectAsMap at MulticlassMetrics.scala:61, took 1.192741 s

[Model Performance Metrics]
Threshold (tuned for F1): 0.65
Accuracy:                 0.7814
Precision:                0.3455
Recall/Sensitivity:       0.6552
Specificity:              0.8016
Weighted Precision:       0.8543
Weighted Recall:          0.7814
F1 Score:                 0.8068
AUC (ROC):                0.6924
AUPR (PR Curve):          0.3302
```

13. Model Performance metrics and interpretations

Evaluation Metrics	Value	Interpretation
Accuracy	**0.78**	Indicates that 78% of the test instances were correctly classified.
AUC (ROC)	**0.69**	Demonstrates good separation between diabetic and non-diabetic cases.
Weighted Precision (PPV)	**0.85**	85% of predicted diabetic cases were true positives.
Weighted Recall	**0.78**	The model detected 78% of actual diabetic cases.
Specificity	**0.80**	80% of non-diabetic cases were correctly identified as negative.
F1 Score	**0.80**	Reflects balanced precision–recall performance suitable for moderately imbalanced data.

14. The Decision Tree model ROC Curve



Overall take on metrics is that these metrics indicate moderate predictive capability, with balanced sensitivity and specificity. ROC curves confirmed meaningful separation between diabetic and non-diabetic classes.

15. Lessons learn and limitations noted

The distributed implementation improved computational efficiency by reducing runtime and allowing greater memory allocation. However, the AUC value of approximately 0.69 indicates that, while the Decision Tree identifies key relationships, its generalizability is limited.

Potential enhancements include applying ensemble methods such as Gradient Boosted Trees, implementing cross-validation for hyperparameter tuning, and conducting feature selection or interaction analysis. These complex features were not integrated because the virtual machines crashed and froze most of the time I attempted to add them to the model.

Despite these limitations, the framework demonstrates that scalable, interpretable machine learning methods can support population-level risk prediction in healthcare.

16. Conclusion

The project successfully implemented a distributed Decision Tree classifier using PySpark on a two-node cluster. The setup achieved both computational efficiency and interpretability. The approach shows promise for scaling public health data analytics and integrating predictive modeling within clinical decision support systems.