**Fake News Detection with Spark: Big Data Project Report**

**1. Introduction**

The spread of misinformation online poses a significant challenge in modern society. Fake news influences public opinion, affects political outcomes, and erodes trust in institutions. Tackling this issue at scale requires processing high volumes of real-time, diverse data—making it a quintessential **Big Data problem**. In this project, we implemented a pipeline to classify text-based news content as real or fake using Apache Spark in the Databricks Community Edition. Our goal was to simulate an environment that mirrors the demands of real-world streaming fake news detection.

**2. Dataset**

We used two labeled datasets: Fake.csv containing fabricated news articles and Real.csv containing factual news. Both datasets included a text field with the news content. Each dataset was pre-labeled: 0 for fake, 1 for real. The combined dataset included 23,481 fake samples and 21,417 real samples, offering a well-balanced base for supervised learning.

**3. Methodology**

**Preprocessing:**

* Lowercasing and removal of non-alphabetic characters
* Tokenization of text
* Stopword removal
* Feature vectorization using **HashingTF + IDF**

**Model Training:**

We experimented with two classifiers:

* **Random Forest** (simplified to avoid overfitting): 10 trees, max depth 4
* **Naive Bayes**: Multinomial mode

The dataset was split 80/20 into training and test sets. Evaluation used **Area Under ROC Curve (AUC)** and **accuracy**.

**Evaluation Results:**

|  |  |  |
| --- | --- | --- |
| **Model** | **AUC** | **Accuracy** |
| Random Forest | 0.8611 | 0.7338 |
| Naive Bayes | 0.5428 | 0.9826\* |

\*Note: The high accuracy of Naive Bayes is misleading due to biased predictions. Random Forest was selected as the final model for its superior generalization capacity.

**4. Streaming Architecture**

We simulated a real-time stream of assertive messages using Spark Structured Streaming. A static CSV file (stream1.csv) was periodically ingested to emulate incoming assertions. The streaming pipeline:

1. Reads new messages
2. Applies the trained Random Forest model
3. Outputs predictions with a timestamp

All processes were built and tested in Databricks using PySpark.

**5. Data Persistence**

Predictions from streaming simulations were saved in **append mode** to a persistent location in the Databricks FileStore as CSVs. Each record included:

* Text
* Prediction (0 or 1)
* Timestamp

We registered this dataset as a temporary Spark SQL table (stream\_results) to enable aggregation queries and analysis. Example SQL:

SELECT prediction, COUNT(\*) FROM stream\_results GROUP BY prediction;

This allows near real-time monitoring of the classification output.

**6. Results**

Upon applying the final model to the streaming dataset, we observed that all test assertions were predicted as fake, indicating residual bias. This led to further consideration of improving feature representation, such as replacing HashingTF with CountVectorizer or exploring Logistic Regression.

Despite this, the pipeline remains functional and fully operational under Big Data constraints: distributed processing, scalable ingestion, persistent storage, and queryable outputs.

**7. Final Thoughts**

This project demonstrates how Apache Spark and Databricks can be used to build an end-to-end fake news detection system at small scale, while preparing for real-world, large-scale deployment.

In production, improvements would include:

* Integration with streaming APIs (e.g., Twitter, RSS)
* More advanced models (e.g., BERT, Logistic Regression with regularization)
* Real-time dashboards with prediction rates and anomaly alerts

Our architecture is designed with these evolutions in mind, offering a solid foundation for future enhancements in combatting fake news using Big Data tools.