Introduction to Spark for Machine Learning

1. What is Apache Spark?

Apache Spark is an open-source distributed computing system designed for fast and scalable data processing. It enables large-scale data processing across clusters of computers and provides APIs in Python, Java, Scala, and R.

Key Features:

- **Distributed Computing**: Processes data in parallel across multiple machines.
- **In-Memory Processing**: Significantly faster than disk-based systems.
- **Fault-Tolerance**: Automatically recovers from failures.
- **Easy Integration**: Supports various data sources (HDFS, S3, Databases, etc.).
- **Unified Analytics**: Supports SQL, streaming, graph processing, and machine learning.

Slide 2: Key Spark Concepts

1. Resilient Distributed Dataset (RDD)

- Immutable, distributed collection of objects
- Supports **fault tolerance** and **parallel processing**
- Not optimized for SQL-style operations

2. Why Use Spark for Machine Learning?

- **Handles Large Datasets**: Can process terabytes of data efficiently.
- **Scalability**: Runs on a single machine or a distributed cluster.
- **Built-in ML Library**: Includes Spark MLlib for scalable ML models.
- **Interoperability**: Can be used with Pandas, TensorFlow, PyTorch, and other ML frameworks.

2. DataFrames

- Distributed table similar to Pandas DataFrame
- Optimized for performance using Catalyst optimizer
- Allows SQL-style queries

3. User-Defined Functions (UDFs)

- Custom functions to apply transformations on DataFrames
- Used for tasks like feature engineering in ML workflows

Slide 3: Spark ML & Machine Learning Pipelines

What is Spark ML?

- A library for scalable machine learning
- Provides tools for:
    - **Feature Engineering** (VectorAssembler, StandardScaler, etc.)
    - **Model Training** (GBTClassifier, RandomForest, etc.)
    - **Hyperparameter Tuning** (CrossValidator, ParamGridBuilder)

---

3. Spark ML Overview

Spark ML Components:

- **DataFrame-based API**: Uses DataFrames for easier manipulation and optimization.
- **Feature Engineering**: Provides tools like VectorAssembler, StandardScaler, and StringIndexer.
- **Model Selection & Tuning**: Supports hyperparameter tuning using CrossValidator and ParamGrid.
- **Classification, Regression, Clustering**: Implements common ML models like Decision Trees, Random Forests, and Gradient Boosted Trees (GBT).

Workflow in Spark ML:

1. **Load Data** → Read from CSV, Parquet, or a database.
2. **Preprocess Data** → Handle missing values, feature scaling, encoding.
3. **Feature Engineering** → Convert raw features into vectors.
4. **Train Model** → Use classifiers like Logistic Regression, Decision Trees, or GBT.
5. **Evaluate Model** → Compute accuracy, AUC, precision-recall.
6. **Save & Deploy Model** → Save trained models for future use.

Spark ML Pipelines

- Chain multiple ML tasks into a structured workflow
- Similar to **scikit-learn Pipelines**

Slide 4: Implementing a Binary Classifier in Spark

1. Load & Process Data

```
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler, StandardScaler
```

---

4. Spark ML Example: Binary Classification

Code Example (Using Gradient Boosted Trees)

2. Train a Model

```
from pyspark.ml.classification import GBTClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator

# Train GBT Model
gbt = GBTClassifier(featuresCol="scaledFeatures", labelCol="label", maxIter=3,
maxDepth=3)
best_model = gbt.fit(train_df)

# Evaluate Model
predictions = best_model.transform(test_df)
evaluator = BinaryClassificationEvaluator(labelCol="label",
rawPredictionCol="rawPrediction", metricName="areaUnderROC")
roc_auc = evaluator.evaluate(predictions)
print(f"ROC-AUC: {roc_auc}")

# Save Model
best_model.write().overwrite().save("saved_gbt_model")
```

Slide 5: Performance Report & Saving Results

1. Generate Performance Report

```
report = f"""
ROC-AUC: {roc_auc:.4f}
PR-AUC: {pr_auc:.4f}
"""
```

---

5. Hands-on: Running Spark ML Code

Steps to Run the Classifier:

1. **Set up a Spark environment** (Databricks, AWS EMR, or local Spark installation).
2. **Load labeled data** from CSV files (fraud and non-fraud datasets).
3. **Preprocess data** (convert embeddings to vectors, normalize features).
4. **Train the classifier** using GBT.
5. **Evaluate performance** (ROC-AUC, precision, recall, F1-score).
6. **Save the model** for future inference.

2. Save Predictions

```
predictions.select("docid", "label", "prediction").write.csv("predictions.csv",
header=True)
```

Running in Databricks or Local Spark:

- Use `spark.read.csv("data.csv")` to load datasets.

- Run `spark-submit` for batch jobs.
- Use Databricks notebooks for interactive Spark ML development.

3. Save Model for Reuse

```
model_path = "saved_gbt_model"
best_model.write().overwrite().save(model_path)
print(f"✅ Model saved at: {model_path}")
```

Slide 6: Key Takeaways

- Spark is a powerful tool for distributed ML on large datasets
- Spark ML provides **scalable machine learning pipelines**
- **RDDs, DataFrames, and UDFs** help with efficient data handling
- Spark's built-in ML tools simplify training, evaluation, and deployment

---

Conclusion

- Apache Spark is a powerful framework for scalable machine learning.
- Spark ML simplifies model training and deployment at scale.
- The Spark GBT classifier can efficiently handle large datasets for fraud detection and other binary classification tasks.