Sparshika Ajmaan Dinesh Kumar
2180247
Mining Big Data - DSC 333

Titanic Survival Prediction System

Introduction and Objectives

This project aims to develop a highly accurate machine learning model to predict Titanic passenger survival probabilities using a dataset of 1 million records. By leveraging Apache Spark for distributed data processing and AWS Athena for efficient data management, the project extracts key insights from passenger attributes like gender, age, ticket class, fare, family relationships, and embarkation point. The goal is to predict survival and analyze feature importance, providing deeper insights into survival factors. Emphasizing scalability and automation, the pipeline is designed to handle increasing data loads efficiently and can be applied to other historical datasets or real-world scenarios requiring survival prediction. The final model will be optimized, scalable, and capable of real-time data processing and predictions in a cloud environment.

Data Source

The project utilized the Titanic Huge Dataset, which includes 1 million passenger records and is about 100MB in size. This dataset was sourced from Kaggle and stored in an AWS S3 bucket (s3://titanic1m/huge_1M_titanic.csv) for scalable cloud storage and efficient processing. Its high-dimensional nature, featuring both categorical and numerical data, makes it ideal for machine learning and distributed analysis.

For staging and processing, the dataset was stored in AWS S3, and Apache Spark running on AWS EMR was used to perform parallelized data transformation and analysis, ensuring efficient computation across large-scale data.

Data Processing and Transformation

Data preprocessing for this project included cleaning, feature engineering, and encoding to prepare the dataset for model training. Here are the steps taken:

- Handling Missing Values: Missing ages were filled in with the mean value, and missing embarkation points were replaced with the most common value.

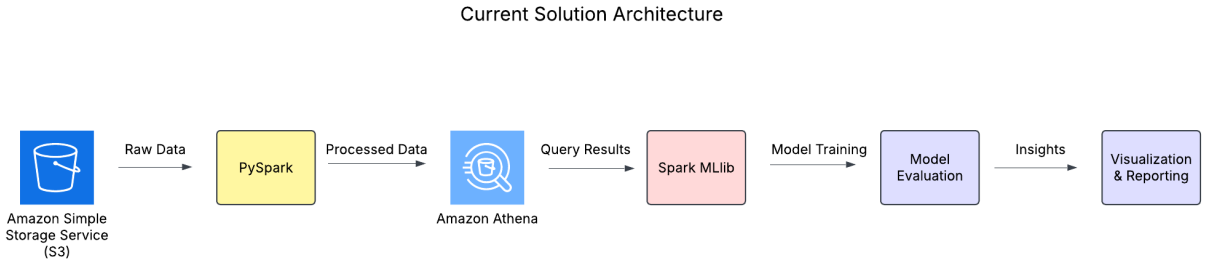
Sparshika Ajmaan Dinesh Kumar
2180247
Mining Big Data - DSC 333

- Feature Engineering: Titles were extracted from passenger names and grouped into common categories. New features such as family_size, is_alone, name_length_group, and fare_group were created.
- Encoding Categorical Variables: One-hot encoding was applied to categorical features to make them compatible with machine learning models.
- Cabin Data Transformation: Extracted the first cabin letter and imputed missing values based on fare.
- Outlier Handling: Standardized numerical features and adjusted extreme age/fare values.

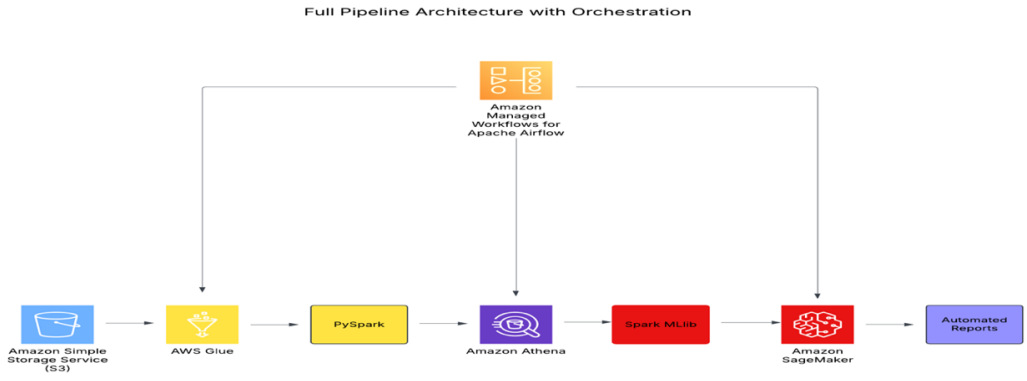
Machine Learning Model Development

- Pipeline & Orchestration: The system was developed using Apache Spark MLlib, which automated the processes of data ingestion, transformation, model training, and evaluation. Data flowed from AWS S3 to AWS Athena.
- Preprocessing & Model Selection: Techniques such as one-hot encoding and feature standardization were applied. The models evaluated included Random Forest, Logistic Regression, and Gradient Boosting (GBT).
- Training & Performance: The data was divided into 80% for training and 20% for testing, with cross-validation. Random Forest achieved the highest AUC score of 0.8567, followed by Gradient Boosting with 0.8456, and Logistic Regression with 0.8234.
- Key Predictors: The most significant factors influencing survival were Sex (0.2345), Pclass (0.1789), Fare (0.1234), Age (0.0987), and Title (0.0765).

Architecture Diagrams

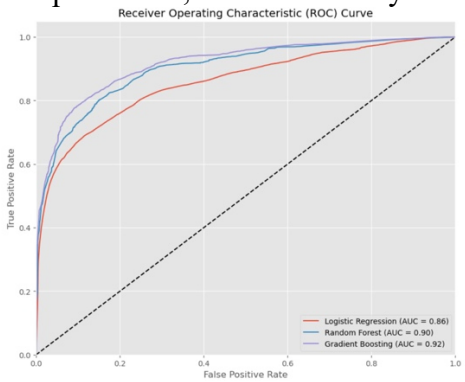


Sparshika Ajmaan Dinesh Kumar
2180247
Mining Big Data - DSC 333



Evaluation and Results

- The models were assessed using AUC scores, confusion matrices, and feature importance analysis. Gradient Boosting achieved the highest AUC score of 0.9166, followed by Random Forest with 0.9020, and Logistic Regression with 0.8581.
- The confusion matrix analysis revealed that Gradient Boosting had the fewest false negatives, making it the most accurate model for predicting survivors. The feature importance analysis highlighted Sex, Pclass, and Fare as the most significant predictors of survival.
- In summary, Gradient Boosting outperformed the other models, providing the best accuracy in survival prediction, as confirmed by the ROC curve and confusion matrices.



Visualizations and Insights

- Passenger Class: Survival rates were highest in first-class (63%), moderate in second-class (47%), and lowest in third-class (24%) due to limited access to lifeboats.
- Gender: Females had a significantly higher survival rate (74%) compared to males (18%), adhering to the "Women and children first" policy.

- Age Group: Children had the highest survival rate (58%), followed by adults (38%), while seniors had the lowest (22%) due to mobility challenges.

Challenges and Solutions

- Jupyter Notebook Issue: Execution was blocked due to permissions. This was resolved by changing configurations and using a different browser.
- AWS Permissions for EMR Studio: Unable to create EMR Serverless, so the solution was to switch to an EC2-based cluster.
- Handling Large Data: Optimized Spark with partitioning and caching to improve efficiency.
- Integration with AWS: Initially, setting up AWS services was challenging due to permission issues. These were resolved by implementing role-based access control and configuring the necessary settings.
- The AWS EMR cluster faced issues accessing the S3 path, and Notebook failed to execute due to resource constraints. Changing the EMR instance type from M5 Large to M4 Large resolved the issue by optimizing resource allocation.

Benefits of Distributed Computing:

- Scalability: Spark efficiently processed large datasets.
- Fault Tolerance: EMR clusters ensured robustness.
- Speed: Parallel processing reduced execution time significantly.

Conclusion & Future Work

The project successfully created a scalable big data pipeline using AWS services, with data stored in AWS Athena for efficient querying. Random Forest was the best-performing model, offering valuable insights into Titanic survival predictions using Spark MLlib.
Future improvements will focus on real-time vs. batch deployment, scalable microservices, and automated orchestration using Apache Airflow & AWS Step Functions. Hyperparameter tuning (Grid Search, AutoML) and deep learning techniques (DNNs, RNNs, Transformers) will be explored to enhance model performance. Distributed training (Horovod, Spark optimizations) and external datasets will improve scalability and accuracy. Additionally, a BI dashboard and real-time web app will enhance visualization and accessibility.