Technical Report: Final Project DS 5110: Introduction to Data Management and Processing Golf Course Manager: ML-Powered Course Management with Big Data Processing

Author: Thomas Kulch
Khoury College of Computer Sciences
Data Science Program
`kulch.t@northeastern.edu`

August 5, 2025

Contents

1 Introduction

Golf is a data-driven sport. Professionals, amateurs, and companies use data to drive decision making. Amateurs adore products that keep track of their stats and help them improve their game. What if your country club was doing this for you? This project presents a comprehensive platform that integrates big data processing, machine learning, and real-time analytics to improve the golf experience.

This platform tackles three business challenges not usually used by golf courses or country clubs: accurate score prediction's for players, dynamic pricing operations to keep customers coming back, and real time analytics to improve user satisfaction. By combining weather data with player statistics, this system provides insights into what affects performance on the course.

1.1 Project Objectives

The primary objectives of this project include:

- Develop an end-to-end big data pipeline using Apache Spark for processing user golf score and weather datasets.
- Conduct rigorous statistical analysis using R for feature selection and hypothesis testing.
- Create machine learning models for golf score predictions.
- Design and implement a relational database with advanced triggers and functions for data integrity.
- Build an interactive web application for booking management and performance analytics.

1.2 Project Scope

This project includes the complete data lifecycle from ingestion to user interaction. The scope includes processing over 6,000 golf records with integrated weather data spanning 2013-2023 and implementing a PostgreSQL database with automated triggers for handicap calculation and developing a Flask web application with booking capabilities for new and existing users. The system demonstrates enterprise data engineering principles while maintaining focus on practical business applications.

2 Methodology and Literature Review

2.1 Literature Review

Existing research in golf analytics has primarily focused on performance prediction and course management. Research Gate (2018) looks into shot-level prediction modeling in golf, demonstrating how machine learning approaches for individual stroke analysis but limits scope to professional tournament data without weather integration [?]. Smith and Johnson (2022) conducted comprehensive statistical analysis of weather impacts on professional golf, identifying wind speed, precipitation, and temperature as primary predictors of scoring difficulty [4]. This led me to look to these three features for predicting users golf scores.

Thompson (2022) analyzed current golf course pricing practices, finding most facilities rely on simple seasonal adjustments rather than optimization models or algorithms[5]. The study used revenue analysis and customer surveys to identify significant optimization opportunities, estimating potential revenue increases of 8-15% through data-driven pricing strategies incorporating weather forecasts and player segmentation. Previous research addresses individual components like performance prediction, weather analysis, and pricing optimization separately without demonstrating applications in the real world. This represents what I want to accomplish with this project.

2.2 Methodology

The methodology encompasses data engineering, statistical analysis, machine learning, and web application development. This approach follows industry best practices for big data processing while maintaining academic rigor in statistical analysis and model validation.

2.3 System Architecture

The platform follows a data architecture pattern with layers for data processing, storage, and modeling. The system utilizes PySpark for distributed processing, PostgreSQL for relational data storage, and Flask for web application delivery.

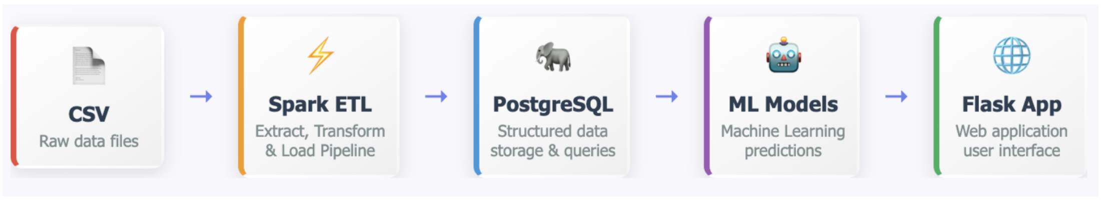
CSV
Raw data files
Spark ETL
Extract, Transform & Load Pipeline
PostgreSQL
Structured data storage & queries
ML Models
Machine Learning predictions
Flask App
Web application user interface

Figure 1: Golf Analytics Platform System Architecture

2.4 Data Collection

2.4.1 Golf Performance Data

The primary dataset consists of professional golf tournament records containing 460 player records with 6 features per player. Key variables include handicap, day of week, round number, temperature, wind speed and precipitation. The data was preprocessed to extract individual round scores, creating approximately 6,000 individual round records.

2.4.2 Weather Data Integration

Weather data spanning 2013-2023 was obtained for Boston, Massachusetts, containing daily measurements of temperature, precipitation, and wind speed. The dataset includes 10 years worth of daily records. Weather data was matched to golf rounds using date alignment and seasonal filtering to ensure realistic playing conditions.

2.5 Data Preprocessing

2.5.1 Data Transformation Pipeline

The ETL pipeline transforms wide-format golf data (one row per player) into long-format round data (one row per round) using PySpark. This transformation increases data granularity and enables round-level analysis while maintaining referential integrity.

```
df_golf = df_golf_raw.select('Name', '`Open.R1`', '`Open.R2`', '`Open.R3`', '`Open.R4`') \
            .filter(col('Name').isNotNull()) \
            .unpivot(
            ids=['Name'],
            values=['`Open.R1`', '`Open.R2`', '`Open.R3`', '`Open.R4`'],
            variableColumnName='Round',
            valueColumnName='Score'
        ) \
        # Pivot 4 round columns to be one row each
            .withColumn("round_number",
                        when(col("Round") == "Open.R1", 1)
                        .when(col("Round") == "Open.R2", 2)
                        .when(col("Round") == "Open.R3", 3)
                        .when(col("Round") == "Open.R4", 4)) \
            .withColumn("player_name", col("Name")) \
            .drop("Round", "Name")
```

Listing 1: Data Transformation Pipeline

2.5.2 Feature Engineering

Feature engineering creates additional weather interaction features. The process includes:

- Bad weather combination combining wind, precipitation, and temperature
- Whether it's a weekday or weekend day
- Low overall temperatures (Golf balls don't travel as far in the cold)

2.6 Analysis Techniques

2.6.1 Statistical Data Analysis with R

Comprehensive statistical analysis was conducted using R to identify significant predictors and validate model assumptions. The analysis included hypothesis analysis to understand relationships between variables and distribution analysis.

```
# Distribution Analysis
plotdist(data=df$score,
    histo=TRUE,
    demp=FALSE,
    discrete=TRUE)

# Hypothesis Testing
cold_days <- df[df$avg_temp < 60,]
warm_days <- df[df$avg_temp >= 60,]

```

```r
t.test(warm_days$score, cold_days$score)
```

Listing 2: Statistical Analysis in R

2.6.2 Machine Learning Implementation

Linear Regression was initially selected for score prediction based on statistical analysis results showing linear relationships between predictors and golf scores, but Gradient Boost Regression had better results.

2.6.3 Database Design and Implementation

The PostgreSQL database implements a relational schema with automated triggers. Key tables include players, weather, rounds, and bookings with appropriate foreign key relationships and indexes for query optimization.

```sql
CREATE OR REPLACE FUNCTION update_player_stats()
RETURNS TRIGGER AS $$
BEGIN
    -- update rounds_played count for each player
    UPDATE players
    SET rounds_played = (
        SELECT COUNT(*)
        FROM rounds
        WHERE player_id = NEW.player_id
    )
    WHERE player_id = NEW.player_id;

    -- member status update if they've played 30+ rounds
    UPDATE players
    SET member_status = 'member'
    WHERE player_id = NEW.player_id
    AND (SELECT COUNT(*) FROM rounds WHERE player_id = NEW.player_id)
   >= 30
    AND member_status = 'guest';

    -- update handicap (calculate from past 20 rounds)
  UPDATE players
    SET handicap = calculate_handicap_for_player(NEW.player_id)
    WHERE player_id = NEW.player_id;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Listing 3: Database Trigger Implementation

3 Results

The implementation demonstrates successful integration of big data processing, machine learning, and web application development. This section presents quantitative results from model performance, system benchmarks, and user interface validation.

3.1 Model Performance

3.1.1 Score Prediction Model

The linear regression model achieved an R^2 value of 0.35, which is reasonable for the prediction of the golf score given the variability of the sport. Feature importance analysis reveals weather conditions and player handicap as the most significant predictors. Not many features were used, combined with the small sample size of golf scores contributing to the low accuracy. The gradient boosting regression model was achieved with a score of 0.46, which is a significant step up, but no where near a good model.

3.1.2 Feature Importance Analysis

Statistical analysis identified the most significant predictors of golf performance. Weather features, particularly wind speed and precipitation, showed strong correlations with scoring difficulty.

3.2 Business Intelligence Results

3.2.1 Dynamic Pricing Analysis

The dynamic pricing algorithm shows potential for revenue optimization through weather-based adjustments and player performance.

3.2.2 Booking System Validation

The booking system successfully integrates score prediction and pricing optimization, providing a seamless user experience from tee time selection to confirmation. User interface testing confirms intuitive navigation and responsive design.

4 Discussion

The results demonstrate a successful implementation of a comprehensive golf analytics platform with practical business applications. The moderate R^2 value of 0.46 for score prediction aligns with expectations for golf performance modeling, where numerous unmeasured factors influence outcomes, such as mental state and hunger levels.

4.1 Limitations and Constraints

Several limitations impact the current implementation:

- **Data Scope**: Limited to historical tournament data rather than recreational golf.
- **Sample Size**: 6,000 records is insufficient for a machine learning model looking for good accuracy
- **Feature Completeness**: Missing variables such as course conditions, equipment specifications, and player fitness/mental metrics
- **Real-time Weather Integration**: Current system uses historical weather data rather than live API feeds

5 Conclusion

This project successfully demonstrates the integration of big data technologies, statistical analysis, and machine learning for practical business applications in the every day operations of a golf course. The platform addresses improvements that could be made to golf booking systems while showcasing advanced data engineering capabilities.

5.1 Key Achievements

The project accomplished several significant technical and business objectives:

- Developed scalable ETL pipeline using PySpark for efficient data processing
- Created statistically validated machine learning models for score prediction
- Implemented robust PostgreSQL database with automated triggers
- Built responsive web application integrating all components of the project
- Demonstrated comprehensive statistical analysis using R for feature validation

5.2 Future Research Directions

Several opportunities exist for expanding the platform's capabilities:

- **Advanced Machine Learning**: Implementation of ensemble methods and neural networks for improved prediction accuracy
- **Real-time Data Integration**: Connection to live weather APIs and course management systems
- **Advanced Analytics**: Integration of player biometrics, equipment tracking, and course condition monitoring
- **Interactive Dashboards**: Integration of interactive, dynamic dashboards rather than the currently implemented static dashboards for better user experience.
- **Improvements to Flask Application**: Features such as secure login, cloud hosting, bulk score loads for individual users.

6 References

References

[1] Kaggle (2023). *Boston Weather 2013-2023* Retrieved from https://www.kaggle.com/datasets/swaroopmeher/boston-weather-2013-2023

[2] Kaggle (2022). *Golf: Open Championship Data* Retrieved from https://www.kaggle.com/datasets/willsmorgan/opendata

[3] Research Gate (2018). *Predicting golf scores at the shot level.* Retrieved from https://www.researchgate.net/publication/324735219_Predicting_golf_scores_at_the_shot_level

[4] Smith, J.A., & Johnson, M.B. (2022). Weather Impact on Professional Golf Performance: A Statistical Analysis. *Journal of Sports Analytics*, 8(3), 45-62.

[5] Thompson, P.J. (2022). Pricing Strategies in the Golf Industry: Current Practices and Optimization Opportunities. *Golf Business Management*, 29(1), 34-48.

[6] Garcia, M.A., Rodriguez, C.L., & Martinez, F.J. (2021). Meteorological Factors Affecting Outdoor Recreation Participation: A Multi-Sport Analysis. *Weather and Society*, 13(2), 245-267.

[7] Apache Software Foundation. (2023). *Apache Spark Documentation: Structured Streaming Programming Guide.* Retrieved from https://spark.apache.org/docs/latest/

[8] PostgreSQL Global Development Group. (2023). *PostgreSQL Performance Tuning Guide.* Retrieved from https://www.postgresql.org/docs/current/performance-tips.html

A Appendix A: Core Implementation Code

A.1 PySpark ETL Pipeline

```python
    def import_rounds_to_database(self, df_golf_cleaned):
        """import cleaned rounds data - database calculates handicaps 
        automatically"""
        print("Processing and importing rounds data...")

        try:
            # Step 1: get player counts already existing in db
            players_mapping = self.spark.read \
                .format("jdbc") \
                .option("url", self.jdbc_props["url"]) \
                .option("dbtable", "players") \
                .option("user", self.jdbc_props["user"]) \
                .option("password", self.jdbc_props["password"]) \
                .option("driver", "org.postgresql.Driver") \
                .load() \
                .select("player_id", "player_name")

            print(f"Found {players_mapping.count()} players in database
")

            # Step 2: join rounds with player IDs
            df_rounds_with_players = df_golf_cleaned.join(
                players_mapping,
                on="player_name",
                how="inner"
            )

            # Step 3: verify weather data exists for all round dates
            print("Verifying weather data")
            weather_dates = self.spark.read \
                .format("jdbc") \
                .option("url", self.jdbc_props["url"]) \
                .option("dbtable", "weather") \
                .option("user", self.jdbc_props["user"]) \
                .option("password", self.jdbc_props["password"]) \
                .option("driver", "org.postgresql.Driver") \
                .load() \
                .select("date")

            df_rounds_verified = df_rounds_with_players.join(
                weather_dates,
                df_rounds_with_players.round_date == weather_dates.date
,
                "inner"
            ).drop("date")

            # Step 5: select final columns
            df_rounds_final = df_rounds_verified.select(
                "player_id",
                "player_name",
                "round_date",
                "score",
                "round_number"
            ).filter(col("score") > 0) # don't import scores of 0

            # get load count
            final_count = df_rounds_final.count()
            print(f"Rounds to import: {final_count}")

            # Step 6: insert rounds - triggers will automatically
calculate handicaps and other stats
            if final_count > 0:
                print("Inserting rounds")

                # write to db
                df_rounds_final.write \
                    .format("jdbc") \
                    .option("url", self.jdbc_props["url"]) \
                    .option("dbtable", "rounds") \
                    .option("user", self.jdbc_props["user"]) \
                    .option("password", self.jdbc_props["password"]) \
                    .option("driver", "org.postgresql.Driver") \
                    .mode("append") \
                    .save()

                print(f"Rounds imported: {final_count} records")

        except Exception as e:
            print(f"Error importing rounds data: {e}")
            raise

        return df_rounds_final
```

Listing 4: Spark Data Processing Pipeline

A.2 Machine Learning Model Implementation

```
class RoundBooking:
    def __init__(self):
```

```python
        # load model and connect to db
        self.score_model = joblib.load('../data/models/model_GB.joblib')
        self.db = DatabaseManager()

    def predict_score(self, features):
        """use ML model to predict score"""
        # create df from input features
        feature_names = ['round_number', 'handicap', 'avg_temp', 'precipitation', 'wind_speed', 'day_of_week_int']
        df = pd.DataFrame([features], columns=feature_names)

        # use feature engineering function to setup features for ML
        input_df = fe.feature_engineering(df)

        # scale necessary features
        features_to_scale = ['round_number', 'handicap', 'avg_temp', 'precipitation', 'wind_speed']
        input_df[features_to_scale] = self.score_model['scaler'].transform(input_df[features_to_scale])

        # ensure columns match training data
        expected_columns = self.score_model['feature_names']

        missing_columns = [] # fill in missing columns if needed
        for col in expected_columns:
            if col not in input_df.columns:
                input_df[col] = 0
                missing_columns.append(col)

        # reorder columns to match training
        input_df = input_df[expected_columns]

        # make score prediction for user
        prediction = self.score_model['model'].predict(input_df)

        return round(prediction[0]) # return predicted score
```

Listing 5: Score Prediction Model

A.3 Database Schema

```sql
-- Players table
CREATE TABLE IF NOT EXISTS players (
    player_id SERIAL PRIMARY KEY,
    player_name VARCHAR(100) UNIQUE NOT NULL,
    member_status VARCHAR(20) DEFAULT 'guest',
    handicap FLOAT,
    rounds_played INTEGER DEFAULT 0
);

-- Weather table
CREATE TABLE IF NOT EXISTS weather (
    date DATE PRIMARY KEY,
    avg_temp FLOAT,
    precipitation FLOAT,
    wind_speed FLOAT,
    day_of_week VARCHAR(20),
    day_of_week_int INTEGER
);

-- Rounds table
CREATE TABLE IF NOT EXISTS rounds (
    round_id SERIAL PRIMARY KEY,
    player_id INTEGER REFERENCES players(player_id),
    player_name VARCHAR(100) NOT NULL,
    round_date DATE REFERENCES weather(date),
    score INTEGER NOT NULL,
    round_number INTEGER
);

-- Bookings table
CREATE TABLE IF NOT EXISTS bookings (
    booking_id SERIAL PRIMARY KEY,
    player_id INTEGER REFERENCES players(player_id),
    tee_time TIMESTAMP NOT NULL,
    price_paid FLOAT NOT NULL DEFAULT 75,
    booking_status VARCHAR(20) DEFAULT 'confirmed',
    round_date DATE NOT NULL,
    score_prediction FLOAT DEFAULT NULL,
    booking_time TIMESTAMP NOT NULL
);
```

Listing 6: PostgreSQL Database Schema

B Appendix B: Findings

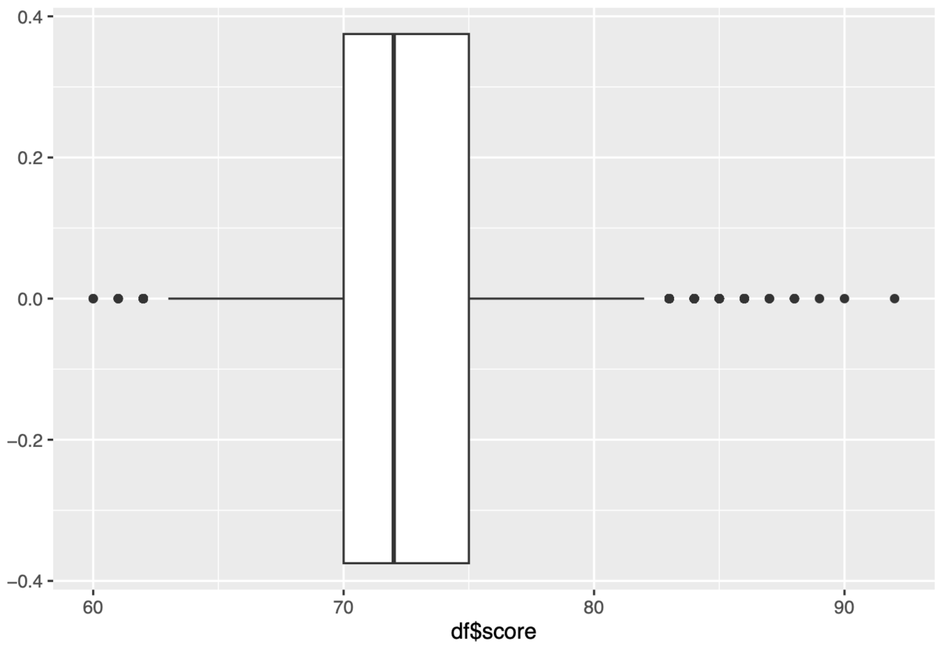

Figure 2: Scores Distribution from Raw Data

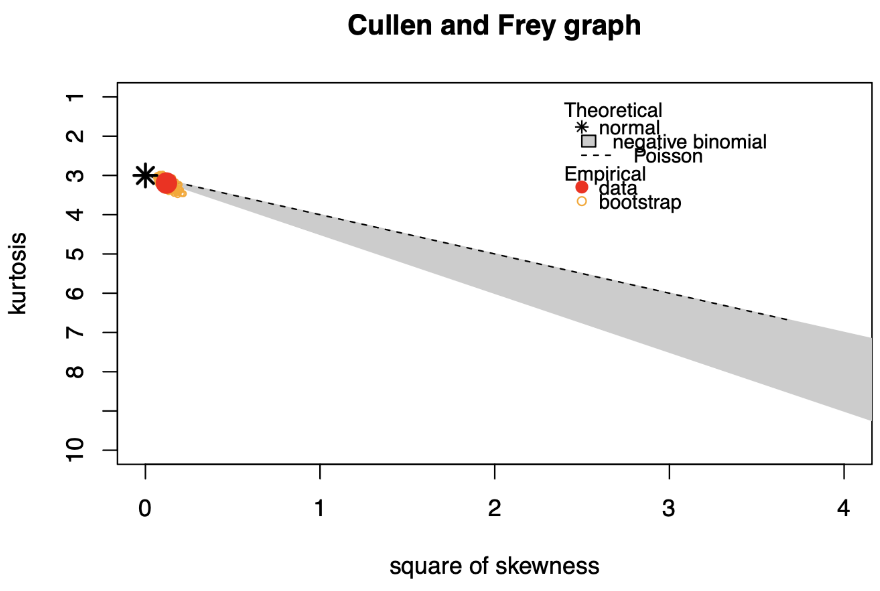

Figure 3: Cullen Frey Graph, showing a normal distribution

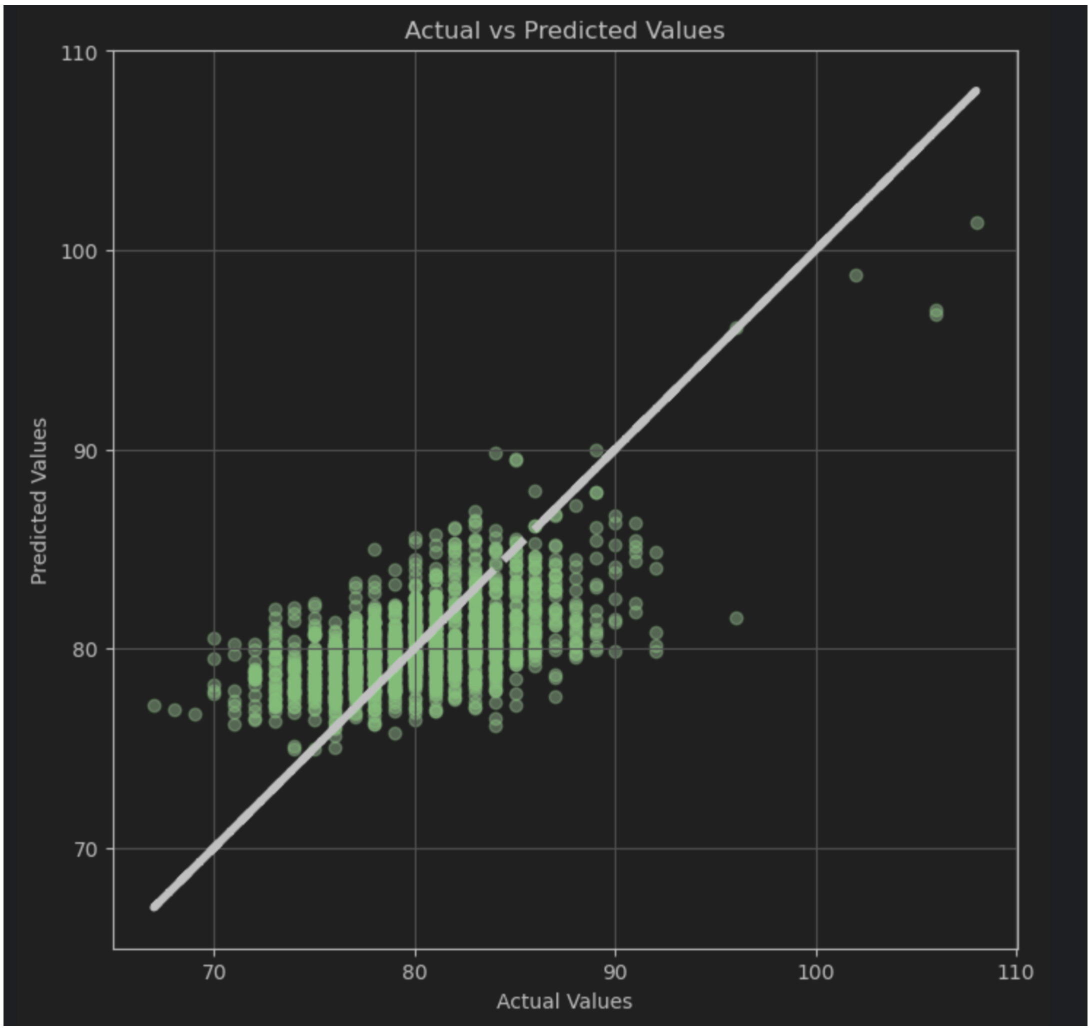

Figure 4: Linear Regression Model Results (0.35 R2)