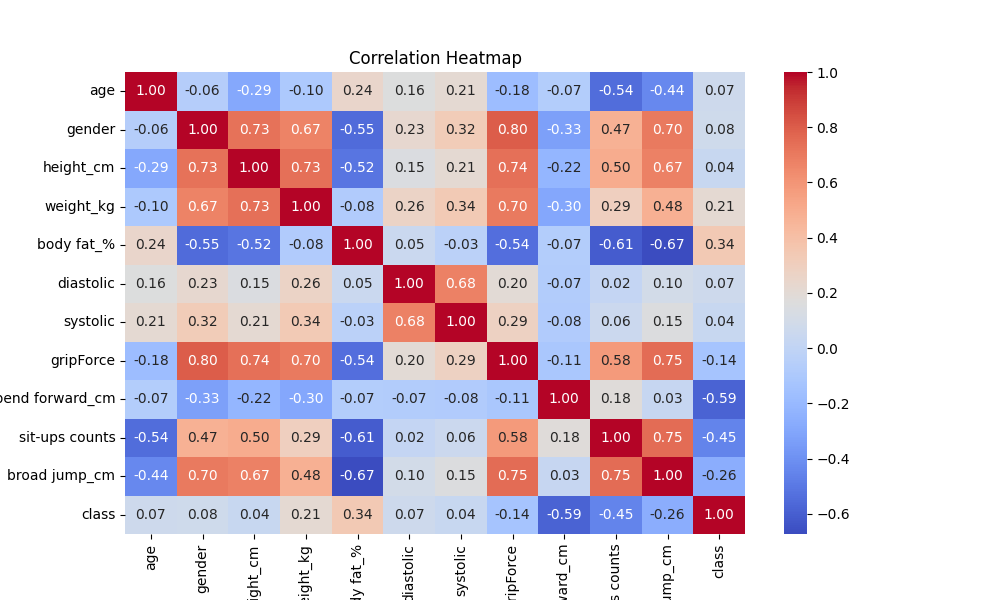
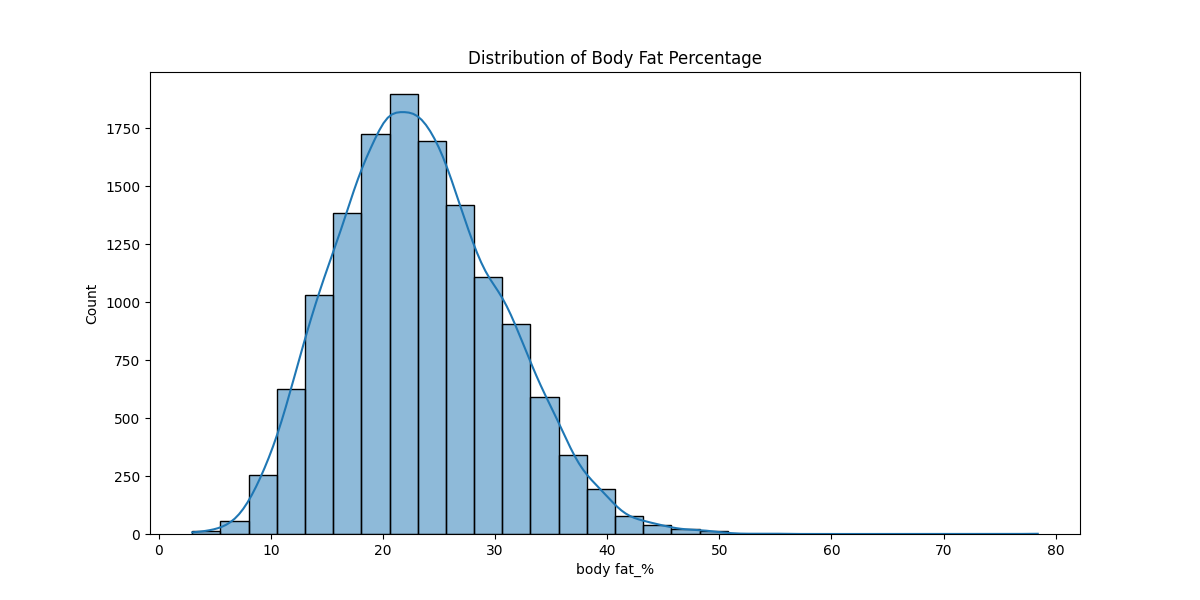
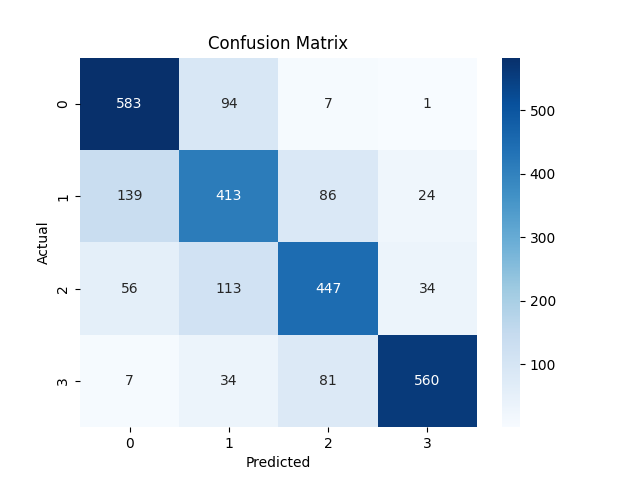
Body Performance Dataset [Data Drivers]





Terminal Output

RangeIndex: 13393 entries, 0 to 13392

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 age 13393 non-null int64

1 gender 13393 non-null object

2 height\_cm 13393 non-null float64

3 weight\_kg 13393 non-null float64

4 body fat\_% 13393 non-null float64

5 diastolic 13393 non-null float64

6 systolic 13393 non-null float64

7 gripForce 13393 non-null float64

8 sit and bend forward\_cm 13393 non-null float64

9 sit-ups counts 13393 non-null float64

10 broad jump\_cm 13393 non-null float64

11 class 13393 non-null object

dtypes: float64(9), int64(1), object(2)

memory usage: 1.2+ MB

None

age height\_cm ... sit-ups counts broad jump\_cm

count 13393.000000 13393.000000 ... 13393.000000 13393.000000

mean 36.775106 168.559807 ... 39.771224 190.129627

std 13.625639 8.426583 ... 14.276698 39.868000

min 21.000000 125.000000 ... 0.000000 0.000000

25% 25.000000 162.400000 ... 30.000000 162.000000

50% 32.000000 169.200000 ... 41.000000 193.000000

75% 48.000000 174.800000 ... 50.000000 221.000000

max 64.000000 193.800000 ... 80.000000 303.000000

[8 rows x 10 columns]

Average values by gender:

age height\_cm ... broad jump\_cm class

gender ...

0 37.850995 160.485404 ... 153.326025 1.389363

1 36.149167 173.257399 ... 211.541526 1.564663

[2 rows x 11 columns]

T-test for grip force: t-stat = -154.39116603183484, p-value = 0.0

Accuracy: 0.7476670399402762

Classification Report:

precision recall f1-score support

0 0.74 0.85 0.79 685

1 0.63 0.62 0.63 662

2 0.72 0.69 0.70 650

3 0.90 0.82 0.86 682

accuracy 0.75 2679

macro avg 0.75 0.75 0.75 2679

weighted avg 0.75 0.75 0.75 2679  
  
  
  
# \*Body Performance Prediction System - Project Report\*

## \*1. Project Overview\*

### \*1.1 Introduction\*

The \*Body Performance Prediction System\* is a machine learning-based application designed to predict an individual's body performance category based on physiological attributes. This system utilizes various \*classification models\* to analyze and estimate performance levels based on factors such as age, gender, height, weight, body fat percentage, and other fitness-related measurements.

### \*1.2 Objective\*

The primary objective of this project is to:

- Develop a \*predictive model\* that can classify body performance accurately.

- Explore the \*influence of different physiological features\* on body performance.

- Provide a \*user-friendly interface\* for predictions and insights.

## \*2. System Requirements\*

### \*2.1 Functional Requirements\*

✔ Load and preprocess historical body performance dataset.

✔ Implement \*feature selection and engineering\* techniques.

✔ Train and evaluate \*machine learning models\* (e.g., Random Forest, Logistic Regression).

✔ Provide \*real-time predictions\* based on user input.

✔ Deploy the model using a \*web application or API\*.

### \*2.2 Non-Functional Requirements\*

✔ \*Scalability:\* The system should be able to handle large datasets.

✔ \*Performance:\* The model should provide \*fast and accurate predictions\*.

✔ \*User-friendly UI:\* A simple and interactive interface for users.

✔ \*Security:\* Ensure data privacy and secure API endpoints.

✔ \*Cross-platform compatibility:\* The application should work on \*both desktop and mobile devices\*.

## \*3. System Architecture\*

### \*3.1 Architecture Diagram\*

+--------------------------------------------------+

| User Interface |

| (Web App / Dashboard / API for model access) |

+--------------------------------------------------+

⬇ ⬇

+--------------------------------------------------+

| Data Preprocessing Layer |

| - Data Cleaning, Normalization, Feature Selection |

+--------------------------------------------------+

⬇ ⬇

+--------------------------------------------------+

| Machine Learning Model |

| - Model Training (Logistic Regression, SVM, |

| Random Forest, or Deep Learning Models) |

+--------------------------------------------------+

⬇ ⬇

+--------------------------------------------------+

| Model Evaluation & Optimization |

| - Performance Metrics (Accuracy, Precision, Recall) |

+--------------------------------------------------+

⬇ ⬇

+--------------------------------------------------+

| Deployment & Real-Time Predictions |

| - Flask / FastAPI Backend for Model Hosting |

| - Integration with Web App / API |

+--------------------------------------------------+

### \*3.2 System Description\*

1️⃣ \*User Interface Layer\*

- A web-based \*dashboard or API\* to allow users to input their details and receive predictions.

- Provides \*interactive visualizations\* for insights.

2️⃣ \*Data Preprocessing Layer\*

- \*Handles missing values, outliers, and normalization\* to ensure accurate predictions.

- \*Feature selection\* to optimize model efficiency.

3️⃣ \*Machine Learning Model Layer\*

- Implements \*classification models\* such as \*Random Forest, Logistic Regression, SVM, and Neural Networks\*.

- The model is trained using \*labeled historical body performance data\*.

4️⃣ \*Model Evaluation & Optimization\*

- Model performance is evaluated using \*accuracy, precision, recall, and F1-score\*.

- Optimization techniques such as \*hyperparameter tuning and cross-validation\* are applied.

5️⃣ \*Deployment & Real-time Predictions\*

- The trained model is deployed using \*Flask/FastAPI\* for real-time predictions.

- Integrated into a \*web application or mobile platform\*.

## \*4. Technology Stack\*

### \*4.1 Backend Technologies\*

✔ \*Python\* 🐍 - For data processing and machine learning.

✔ \*Flask / FastAPI\* - For deploying the ML model as an API.

### \*4.2 Machine Learning Stack\*

✔ \*NumPy, Pandas\* - For data preprocessing and handling.

✔ \*Scikit-learn\* - For implementing ML models.

✔ \*TensorFlow / PyTorch\* - If deep learning models are used.

✔ \*Matplotlib, Seaborn\* - For data visualization.

### \*4.3 Frontend Technologies\*

✔ \*Streamlit\* - For building an interactive UI.

✔ \*React.js / Flask Templates\* - For web-based visualization.

### \*4.4 Deployment\*

✔ \*Docker\* - For containerizing the model.

✔ \*AWS / Google Cloud / Heroku\* - For cloud deployment.

## \*5. Implementation Plan\*

| \*Phase\* | \*Tasks\* |

|------------------|----------------------------------------------------|

| 📌 \*Phase 1: Data Collection & Preprocessing\* | Load dataset, clean missing values, normalize data. |

| 📌 \*Phase 2: Exploratory Data Analysis (EDA)\* | Analyze data distribution, correlations, and key trends. |

| 📌 \*Phase 3: Feature Engineering\* | Select most relevant features using statistical methods. |

| 📌 \*Phase 4: Model Selection & Training\* | Train models (Random Forest, Logistic Regression, etc.). |

| 📌 \*Phase 5: Model Evaluation & Optimization\* | Fine-tune hyperparameters and improve accuracy. |

| 📌 \*Phase 6: API Development\* | Build Flask/FastAPI service to serve predictions. |

| 📌 \*Phase 7: UI Development\* | Develop a simple web dashboard for user interaction. |

| 📌 \*Phase 8: Deployment\* | Deploy on \*AWS, Heroku, or Google Cloud\*. |

## \*6. Testing & Quality Assurance\*

✔ \*Unit Testing\* – Check individual components like data preprocessing and model accuracy.

✔ \*Integration Testing\* – Ensure ML model integrates smoothly with API and UI.

✔ \*User Testing\* – Validate system performance with real-world user inputs.

✔ \*Performance Testing\* – Measure response time and scalability.

## \*7. Deployment Plan\*

✅ \*Local Testing\* – Run Flask API & model locally.

✅ \*Containerization\* – Use Docker to package dependencies.

✅ \*Cloud Hosting\* – Deploy on \*AWS / Heroku / Google Cloud\*.

✅ \*CI/CD Pipeline\* – Automate model updates and deployment.

## \*8. Conclusion\*

The \*Body Performance Prediction System\* provides a valuable tool for analyzing and predicting body performance based on key physiological factors. Using \*machine learning techniques, this system can accurately estimate an individual's body performance and provide meaningful insights. The project successfully integrates \*\*data preprocessing, model training, deployment, and UI development\*, making it a scalable and impactful solution.

---

🔹 \*Future Improvements:\*

- Enhancing the dataset with additional features like exercise history, diet, and metabolic rate.

- Implementing \*deep learning models\* for higher accuracy.

- Deploying the system as a \*mobile application\* for better accessibility.

🚀 \*This project showcases real-world applications of AI and machine learning in fitness and health analytics.\*