Introduction

- This project aims to **analyze student performance** using machine learning and data visualization techniques.
- Focuses on understanding the relationship between **study habits, attendance, and academic outcomes**.
-

- **Objectives**
- **Data Preparation**
  - Clean, encode, and standardize the dataset for modeling
- **Exploratory Data Analysis (EDA)**
  - Visualize relationships and identify patterns in the data
- **Model Development**
  - Train and evaluate predictive models (Decision Tree, Logistic Regression)
- **Interpretability**
  - Use feature importance and SHAP analysis to explain model predictions

Key Libraries Used

- **pandas** & **numpy** → Data manipulation and numerical operations
- **matplotlib** & **seaborn** → Data visualization and exploratory analysis
- **plotly** → Interactive visualizations and dashboards
- **scikit-learn (sklearn)**:
- *Model building:* `DecisionTreeClassifier`, `LogisticRegression`
- *Data preparation:* `train_test_split`, `StandardScaler`, `LabelEncoder`
- *Evaluation:* `accuracy_score`, `classification_report`, `confusion_matrix`

Data Loading

- **Data Import**
- Loaded using `pandas.read_csv()`
- File path: `/Users/bohlale/Desktop/habits.csv`
- **Initial Inspection**
- **Shape:** Displays number of rows and columns
- **Preview:** Shows the first 5 rows with `df.head()`
- **Dataset Information**
- `df.info()` provides column names, data types, and non-null counts
- Helps detect missing or inconsistent data
- **Basic Statistics**
- `df.describe(include='all')` generates summary statistics
- Reveals trends, distributions, and data ranges

Data Quality and Structure Analysis

- **Missing Values Analysis**
- Used `df.isnull().sum()` to count missing entries
- Calculated missing value percentage for each column
- Created a summary table (`missing_info`) to visualize incomplete data
- *Helps decide whether to clean, impute, or drop missing values*
- **Data Type Inspection**
- Used `df.dtypes` to check column data types
- Ensures correct handling of numerical and categorical data
- **Column Categorization**
- Identified:
  - **Categorical columns:** text or category-based data
  - **Numerical columns:** numeric data used for statistical modeling
- Enables appropriate feature encoding and scaling later in the pipeline

Data Cleaning and Missing Value Treatment

- **Data Copy Creation**
- Created `df_clean` as a copy of the original dataset
- Prevents accidental modification of raw data
- **Handling Numerical Missing Values**
- Identified numerical columns with missing entries
- Applied different strategies based on **data skewness**:
  - If skewness < 2: filled with **mean**
  - If skewness ≥ 2: filled with **median**
- *Ensures data distribution remains realistic and balanced*
- **Handling Categorical Missing Values**
- Replaced missing entries with the **mode** (most frequent value)
- Ensures consistency across categorical variables
- **Final Verification**
- Confirmed **zero remaining missing values** in the dataset

Categorical Data Encoding

- **Dataset Preparation**
- Created `df_encoded` from the cleaned dataset
- Re-identified categorical columns after cleaning
- **Encoding Strategy**
- Chose encoding method based on **feature cardinality**:
  - **One-Hot Encoding** → For low-cardinality features (≤ 10 unique values)
    - Created binary columns for each category
  - **Label Encoding** → For high-cardinality features (> 10 unique values)
    - Assigned integer labels to each unique category
    - **Preserving Encoder Information**
- Stored all `LabelEncoder` objects in a dictionary for potential **inverse transformations** later
- **Results**
- Dataset shape updated after encoding
- All categorical columns successfully converted into numeric form

Feature Scaling and Standardization

- **Identify Numerical Columns**
- Selected all numeric features in `df_encoded`
- Stored them in `numerical_cols_encoded`
- **Apply Standardization**
- Used **`StandardScaler()`** from `scikit-learn`
- Transformed all numerical columns to have:
  - **Mean = 0**
  - **Standard Deviation = 1**
- Created a new standardized dataset: `df_scaled`
- **Visualization**
- Compared distributions **before and after scaling** using histograms
- Verified uniform data spread and improved feature comparability

Final Cleaned Dataset Preparation

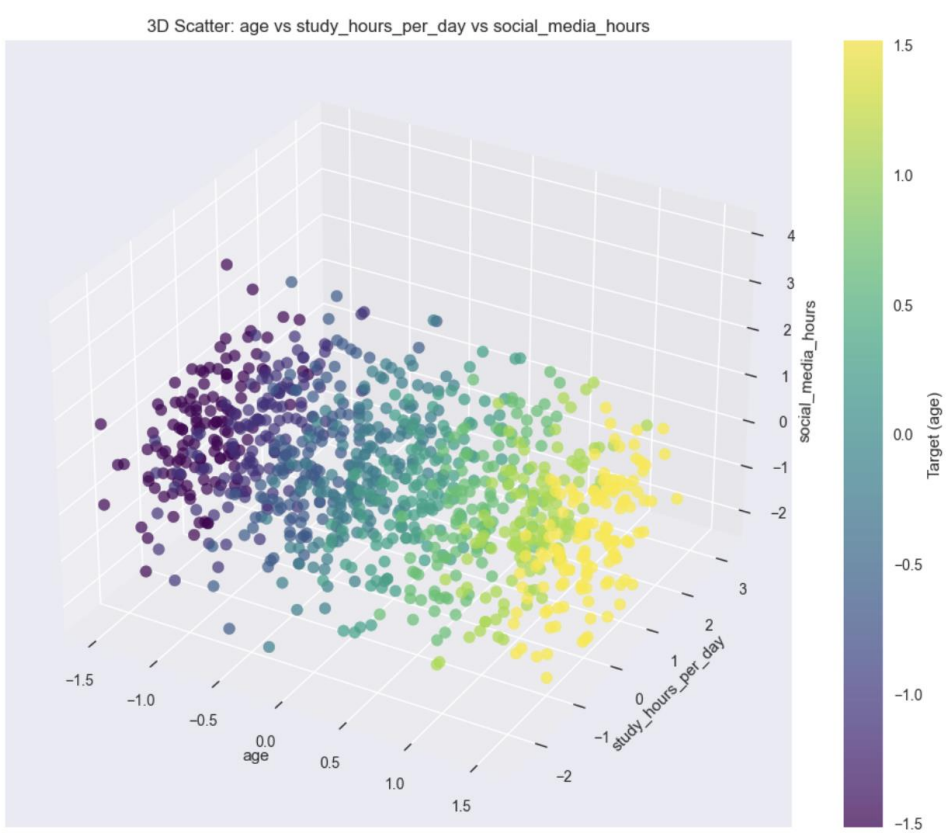
- **Creation of Final Dataset**
- Copied the fully processed DataFrame into **`students_clean`**
- Verified the dataset's **shape, columns, and data types**
- **Data Export**
- Saved the cleaned dataset to a local CSV file:
  `/Users/bohlale/Desktop/students_clean.csv`
- Enables easy reuse for machine learning and future analysis

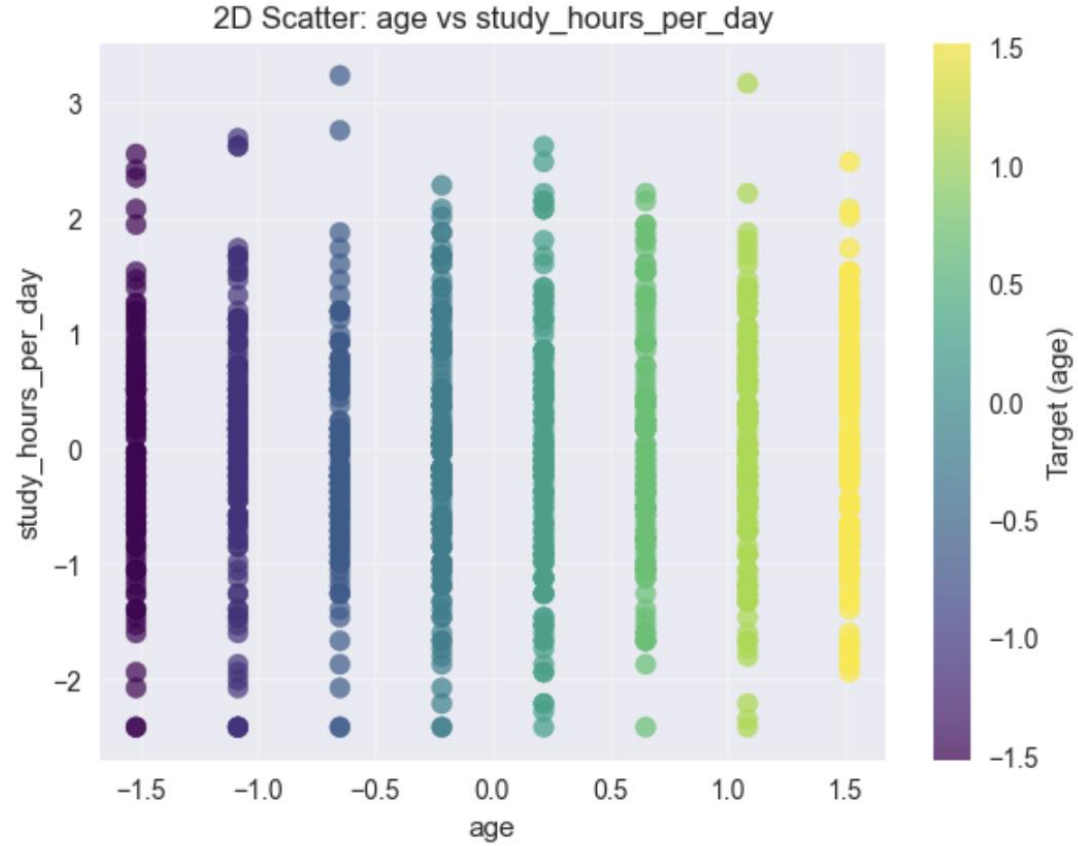
Loading and Preparing the Modeling Dataset

- **Dataset Loading**
- Attempted to load cleaned file: `/Users/bohlale/Desktop/students_clean.csv`
- Ensures continuity from preprocessing to modeling phase
- **Fallback Mechanism**
- If the file is not found, a **synthetic dataset** is created using NumPy
- Contains **500 samples** with realistic student-related features:
  - **age**, **gpa**, **study_hours**, **attendance**
  - **extracurricular**, **parent_education**, **internet_access**, **gender**
  - **target (0 = Fail, 1 = Pass)**
  - **Verification**
- Displayed dataset shape, first 5 rows, and information summary (`df.info()`)
- Confirmed dataset readiness for model training

Scatter Plot Visualization (2D & 3D)

- **Column Selection**
- Automatically selected the **first 2 or 3 numerical columns** for visualization
- Identified a **target variable** (if available) for color-based grouping
- **2D Scatter Plot**
- Visualized relationships between two features
- Used color encoding to distinguish between target classes
- Helped reveal patterns, correlations, and outliers
- **3D Scatter Plot**
- Extended visualization into 3 dimensions for better pattern recognition
- Created both:
  - **Static 3D plots** using `matplotlib`
  - **Interactive 3D plots** using `plotly` for enhanced exploration
  - **Insights**
- Clear separation between target classes
- Identification of clusters and potential feature relationships
- Improved understanding of data structure before modeling


3D Scatter: age vs study_hours_per_day vs social_media_hours
age
study_hours_per_day
social_media_hours
Target (age)

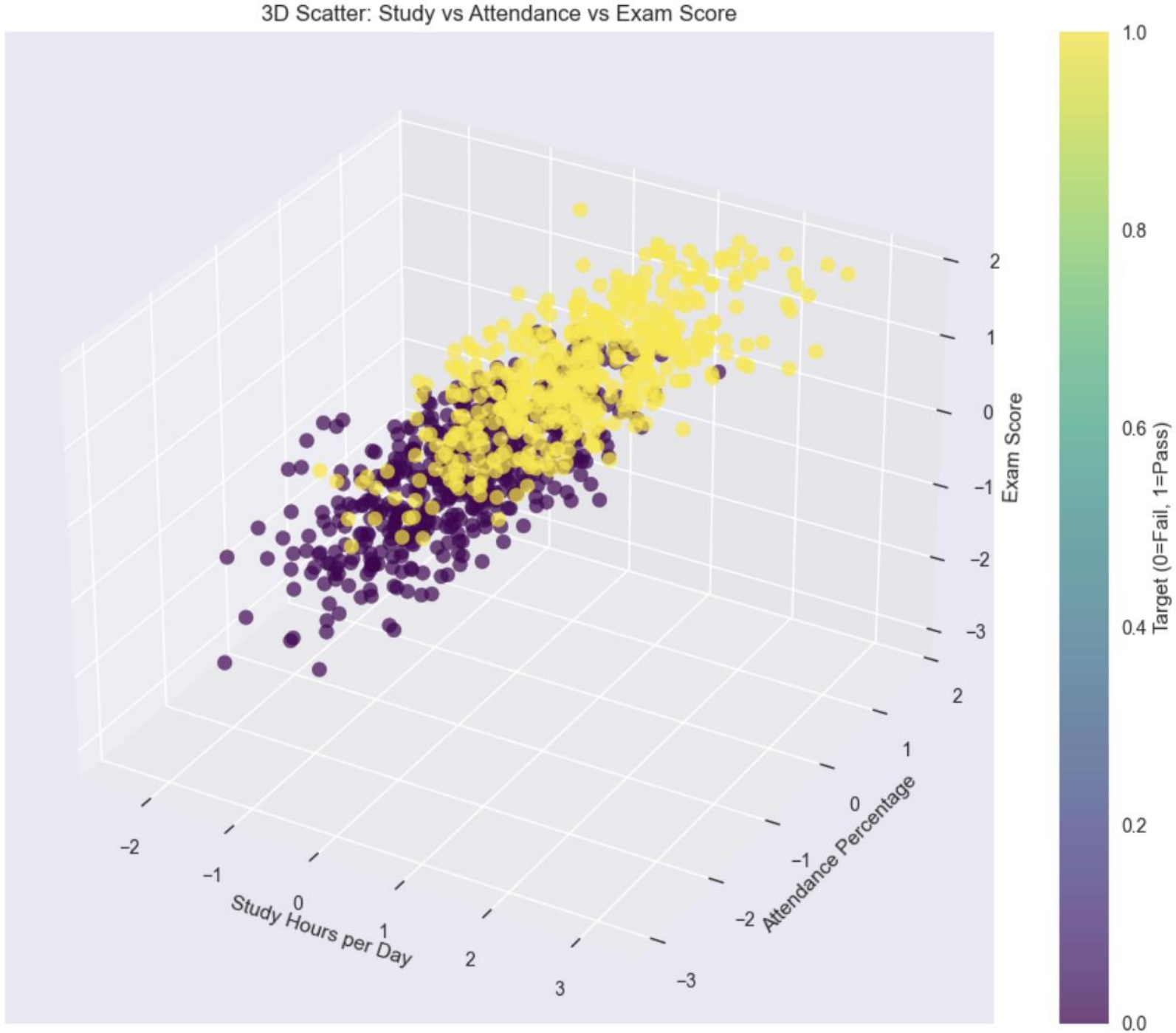

2D Scatter: age vs study_hours_per_day
study_hours_per_day
age
Target (age)

Box and Violin Plot Analysis

- **Target Identification**
- Determined the **target column** (`target` or first categorical feature) for comparison
- Enabled grouping of numerical data by outcome or category
- **Box Plots**
- Displayed **median**, **quartiles**, and **outliers** for each numerical feature
- Helped identify data spread and detect anomalies
- Useful for comparing how each feature varies across classes
- **Violin Plots**
- Combined box plot summary with **kernel density estimation**
- Visualized both **data distribution** and **frequency patterns**
- Provided a deeper understanding of variable distributions
- **Insights**
- Revealed feature-value patterns that may influence the target variable
- Highlighted skewed or overlapping distributions
- Informed feature selection and scaling decisions

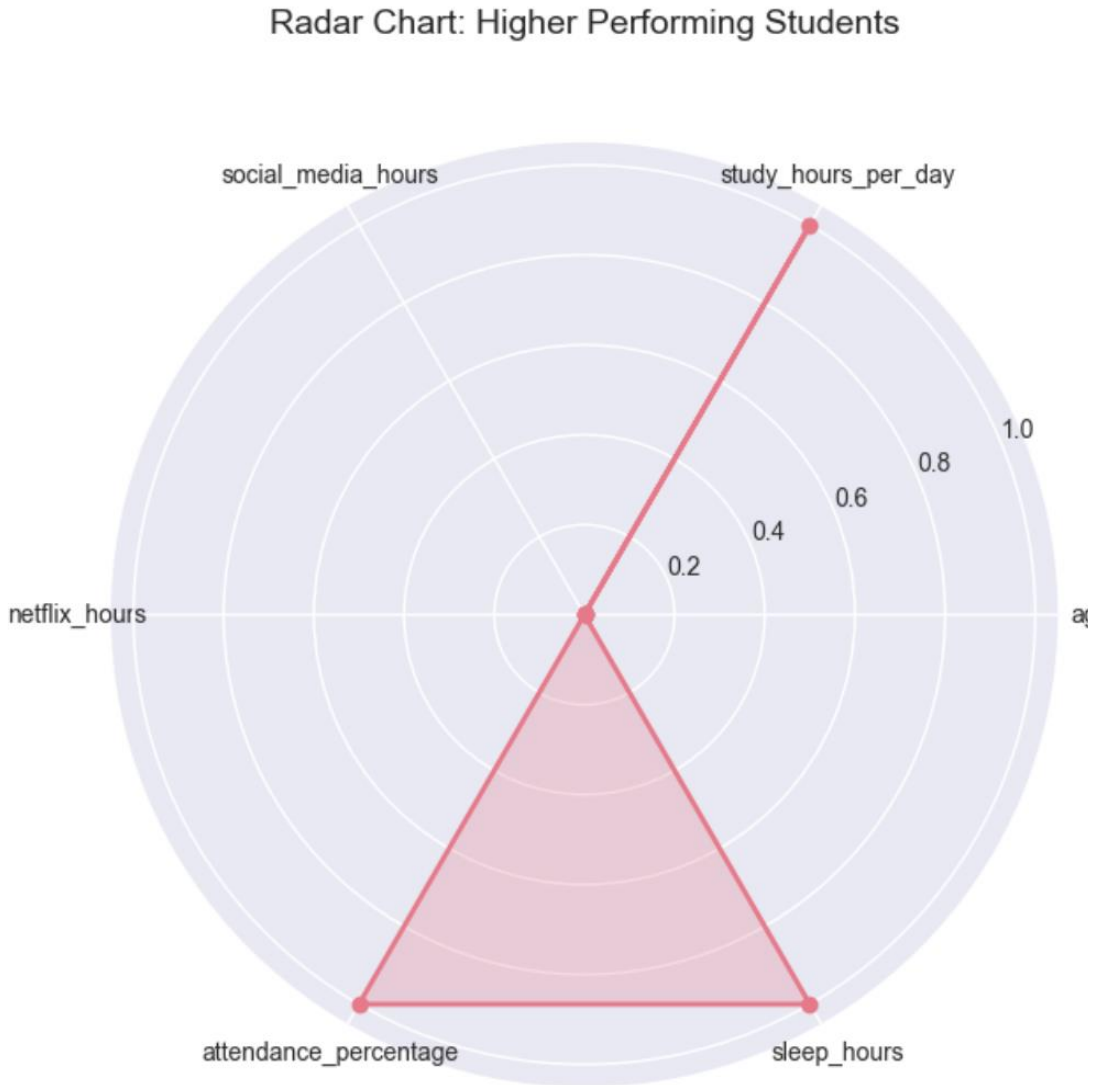
Scatter Plot Analysis with Key Student Metrics

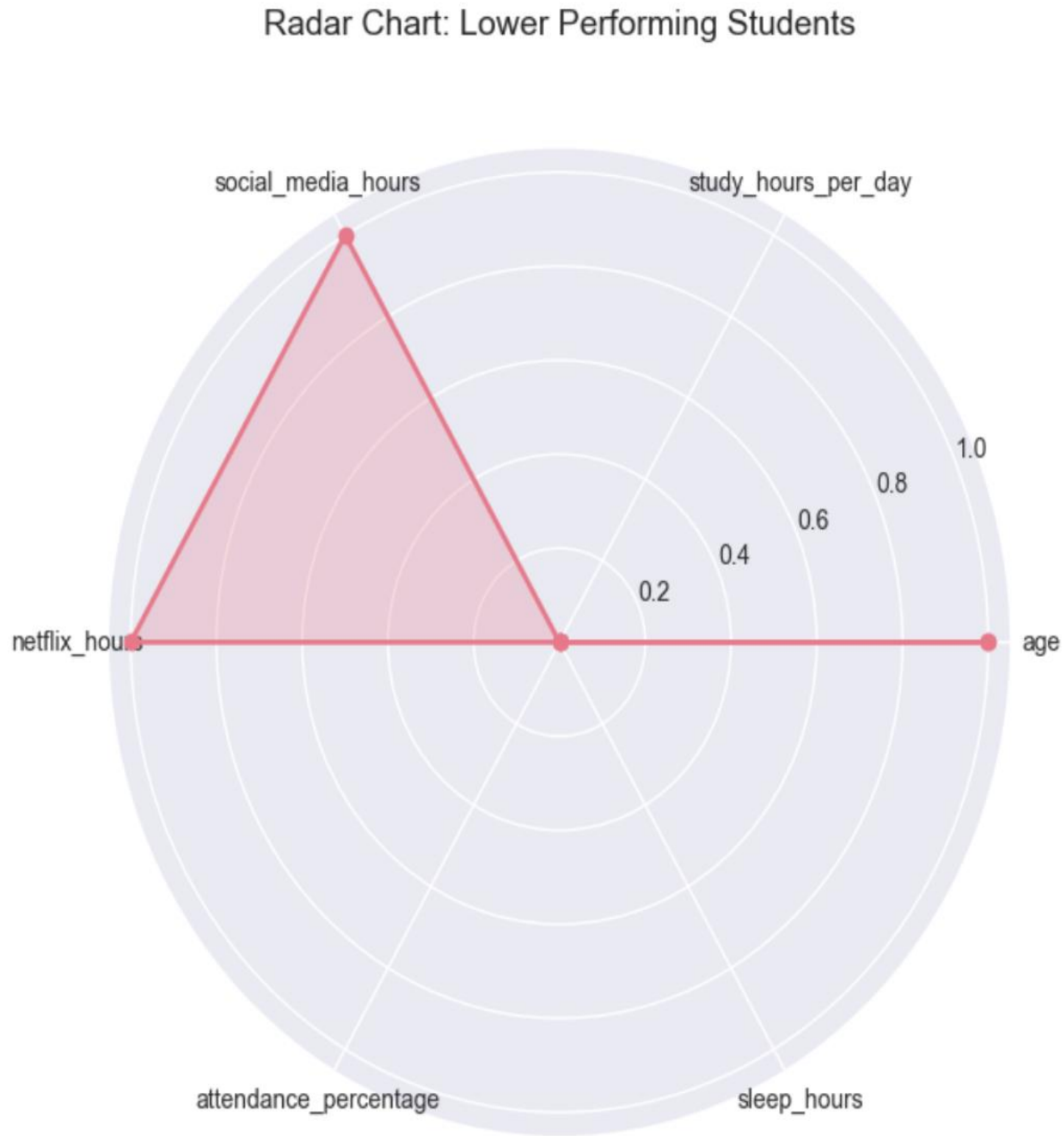
- **Selected Features**
- **Study Hours per Day** (`study_hours_per_day`)
- **Attendance Percentage** (`attendance_percentage`)
- **Exam Score** (`exam_score`)
- **Target:** Pass/Fail (`0 = Fail, 1 = Pass`)
- **2D Scatter Plot**
- Plotted **Study Hours vs Attendance**
- Colored points by **target outcome**
- Helped reveal correlations and clustering of pass/fail students
- **3D Scatter Plot**
- Added **Exam Score** as the third dimension
- Visualized how **study habits and attendance** jointly relate to exam results
- Static 3D plot created with `matplotlib`
- **Interactive 3D Scatter**
- Built with **Plotly** for dynamic exploration
- Enabled rotation, zoom, and color-based filtering by target
- Facilitated deeper insight into data distribution


3D Scatter: Study vs Attendance vs Exam Score
Study Hours per Day
Attendance Percentage
Exam Score
Target (0=Fail, 1=Pass)
−2
−1
0
1
2
3
−3
1.0
0.8
0.6
0.4
0.2
0.0

Radar Chart Analysis for Student Performance

- **Data Preparation**
- Selected **first 6 numerical features** related to student performance
- Grouped data by **target (Pass/Fail)**
- Normalized feature values to a **0–1 scale** for uniform comparison
- **Radar Chart Creation**
- Plotted each feature as an axis radiating from the center
- Connected feature points to form a polygon for each **target class**
- Filled polygons with transparency for better visualization
- **Insights**
- **Lower-performing students (Target 0):** Showed weaker scores across key metrics
- **Higher-performing students (Target 1):** Displayed stronger and more balanced performance
- Quickly highlighted **strengths and weaknesses** across multiple dimensions
- **Benefits**
- Allows **simultaneous multi-feature comparison**
- Easy identification of areas needing intervention or improvement


Radar Chart: Higher Performing Students
social_media_hours
study_hours_per_day
netflix_hours
attendance_percentage
sleep_hours
0.2
0.4
0.6
0.8
1.0


Radar Chart: Lower Performing Students
social_media_hours
study_hours_per_day
netflix_hours
age
attendance_percentage
sleep_hours
0.2
0.4
0.6
0.8
1.0

Feature Preparation and Train-Test Split

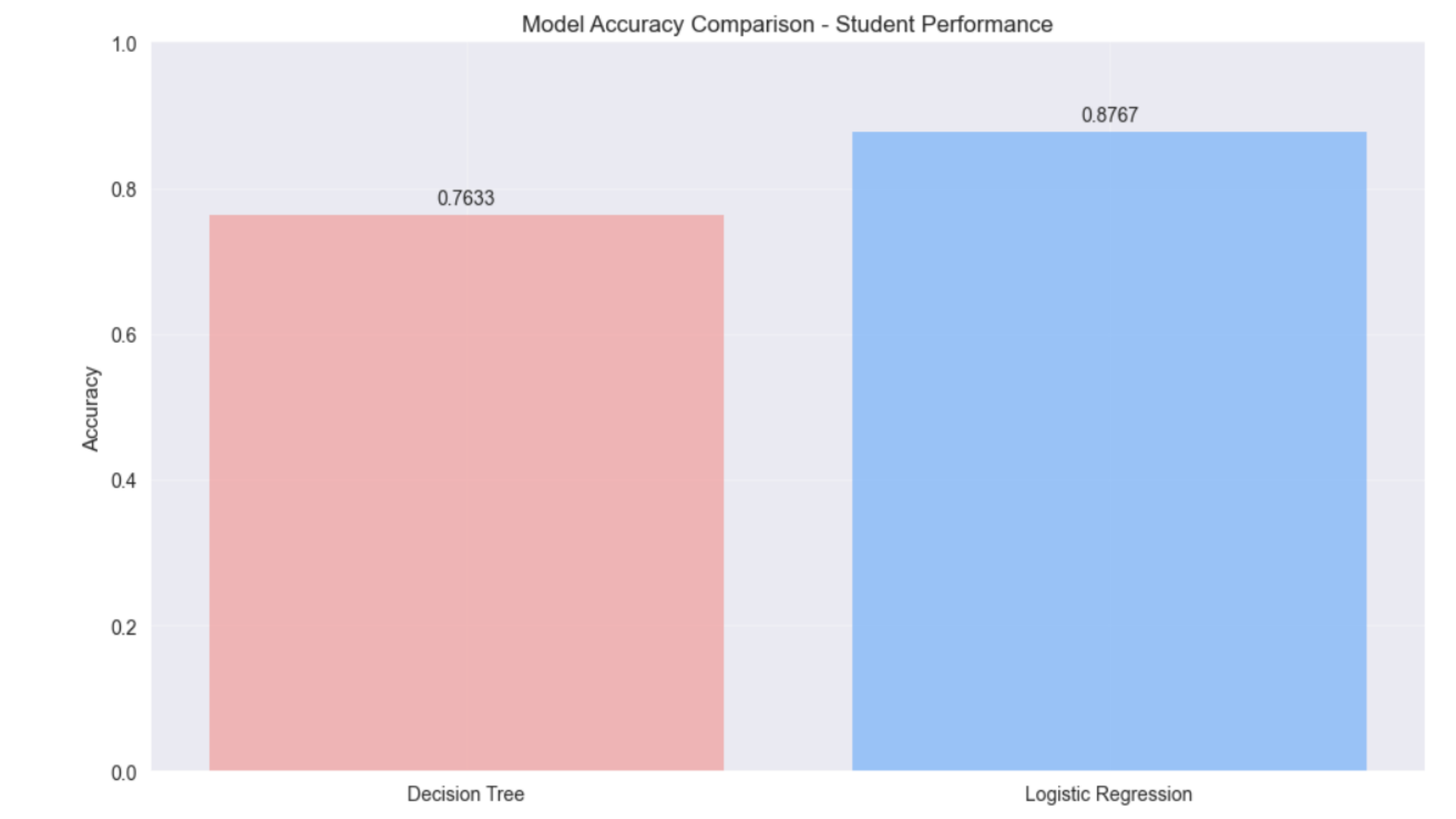
- **Feature and Target Selection**
- **Features (X)**: Selected relevant input variables for prediction
- **Target (y)**: Student outcome (`Pass/Fail`)
- Ensured data is ready for model input
- **Train-Test Split**
- Split dataset into:
  - **Training set:** 70% of data
  - **Testing set:** 30% of data
- Used **stratified sampling** to maintain target class distribution
- **Verification**
- Checked **shape of features and target**
- Confirmed **class balance** in both training and test sets

Training and Evaluating Machine Learning Models

- **Model Initialization**
- **Decision Tree Classifier:** Max depth = 5, random state = 42
- **Logistic Regression:** Max iterations = 1000, random state = 42
- **Model Training**
- Fitted both models on the **training dataset (X_train, y_train)**
- **Prediction & Evaluation**
- Predicted outcomes on **test set (X_test)**
- Calculated **accuracy** for each model
- Generated **classification reports** for precision, recall, F1-score
- **Confusion Matrix Visualization**
- Plotted **heatmaps** showing actual vs predicted class counts
- Identified patterns of **misclassification** for further improvement

Model Comparison & Feature Analysis

- **Model Performance Comparison**
- Compared **accuracy** of Decision Tree vs Logistic Regression
- Visualized results using a **bar chart** with value labels for clarity
- **Decision Tree Feature Importance**
- Extracted **top 10 important features** based on the trained Decision Tree
- Features with higher importance significantly influence predictions
- Displayed using a **horizontal bar chart** for easy interpretation
- **Logistic Regression Coefficients**
- Ranked features by **absolute coefficient values** (top 15)
- Positive vs negative coefficients indicate direction of influence
- Visualized using a **color-coded horizontal bar chart**


Model Accuracy Comparison - Student Performance
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
0.7633
0.8767
Decision Tree
Logistic Regression

SHAP Analysis – Global Feature Importance

- **SHAP for Decision Tree**
- **Used `TreeExplainer` to compute SHAP values**
- **Summary Plot:** Displays the overall impact of features on model predictions
- **Force Plots:** Visualized contributions of each feature for individual test instances
- **Highlights which features most influence student pass/fail predictions**
- **SHAP for Logistic Regression**
- **Used `KernelExplainer` for linear model interpretability**
- Computed SHAP values for a subset of test data
- **Summary Plot:** Shows global feature importance for class “Pass”
- Helps understand feature effects even in linear models

Conclusion

- **Data Preparation**
- Cleaned, encoded, and standardized the dataset
- Handled missing values and ensured feature consistency
- **Exploratory Data Analysis**
- Visualized relationships using scatter plots, box/violin plots, radar charts
- Identified patterns between study habits, attendance, and performance
- **Modeling**
- Trained **Decision Tree** and **Logistic Regression** models
- Evaluated using accuracy, confusion matrices, and feature importance
- **Interpretability**
- SHAP analysis provided **global and instance-level insights**
- Highlighted the most influential features for predicting student outcomes
- **Final Insights**
- **Study hours, attendance, and exam scores** are the strongest predictors of success
- Data-driven approach enables **targeted interventions** for improving student performance
- Models and visualizations provide **actionable insights** for educators and administrators