
FIFA 21 EDA PROJECT
MADE BY :
MOHAB SHERIF MOHAMED
STEVEN TAMER SOLIMAN
MOHEB NASR
ABDULLAH KHALED

FIFA 21 PLAYERS OVERVIEW

1. What It Is :

The “FIFA 21 complete player dataset” is a Kaggle-hosted collection of player-level data, largely scraped from SoFIFA, that includes comprehensive attributes for players featured in FIFA 21. It typically contains:

- Player metadata: Nationality, age, club/team (though transfers around release time may cause minor inaccuracies), date of birth, wages, value, etc.
- Skill attributes: Metrics such as attacking, defending, mentality, goalkeeping, pace, passing, and more — often numbering over 100 attributes .
- Ideal format: Generally delivered as CSV files (e.g., “fifa21_male2.csv”) of modest size (~1–2 MB), featuring approximately 9–18 k player records, depending on dataset version

2. How It’s Used:

Multiple projects have leveraged this dataset for:

- Exploratory Data Analysis (EDA): Cleaning data, calculating summary statistics, and visualizing relationships (e.g., value vs. wage, distributions of ratings) .
- Dashboards & Visual Tools: Interactive visuals, player radars, and filtering tools built using Python or Power BI .
- Community Analysis: Users share insights like highest potentials, undervalued players, and nationality patterns via forums (e.g., Reddit) .graph text

DATAANALYZER – DATA LOADING & CLEANING

1.Purpose

To efficiently load a football dataset from a CSV file and clean it for analysis by:
Standardizing column names
Removing duplicates & missing values
Converting player market values from string formats (€, M, K) into numeric values

---

2. Step-by-Step Process

A. Loading the Data :

Read CSV using pandas.read_csv().
Lowercase all column names for consistency.
Preview the first 3 rows and list available columns.
Error handling: Displays a message if the file path is incorrect.

B. Cleaning the Data :

Remove duplicates – Ensures unique player entries.
Check required column value_eur – Raises an error if missing.
Drop rows with missing key fields:
age,value_eur,overall
Convert monetary values:
Removes the "€" sign.
Converts:
"M" → Millions (x * 1,000,000)
"K" → Thousands (x * 1,000)
Plain numbers → Float.
Ensure numeric type – Casts value_eur to float.

---

3. Key Features

✅ Handles different currency formats
✅ Removes irrelevant or incomplete rows
✅ Provides clear console feedback
✅ Prepares dataset for further analysis & visualization

DATAANALYZER – FEATURE ENGINEERING & STATISTICS

1.Purpose

Enhance the dataset with new calculated fields and generate summary statistics for quick insights.

---

2. Step-by-Step Process

A. . Adding New Columns (add_columns) :

agegroup – Categorizes players by age:

Bins: [15–20], [20–25], [25–30], [30–35], [35–45]

Labels:

Teen → 15–20,Young → 20–25

Prime → 25–30,Veteran → 30–35

Senior → 35–45

Benefit: Easier analysis of players by career stage.

valueperrating – Calculates market value per rating point:

valueperrating,value_eur,overall

valueperrating=

overall/

value_eur

Benefit: Shows cost efficiency of a player.

valueperrating=

overall/

value_eur

Benefit: Shows cost efficiency of a player.

B. Statistical Summary (show_statistics) :

Descriptive Statistics

Provides count, mean, min, max, and quartiles for:

age

overall

value_eur

valueperrating

Correlation Matrix

Shows relationships between variables.

Example: How strongly overall relates to value_eur.

3. Key Features

- ✅ Creates categorical and numerical insights
- ✅ Helps identify player value efficiency
- ✅ Quick data distribution and relationship checks
- ✅ Useful for exploratory data analysis (EDA)

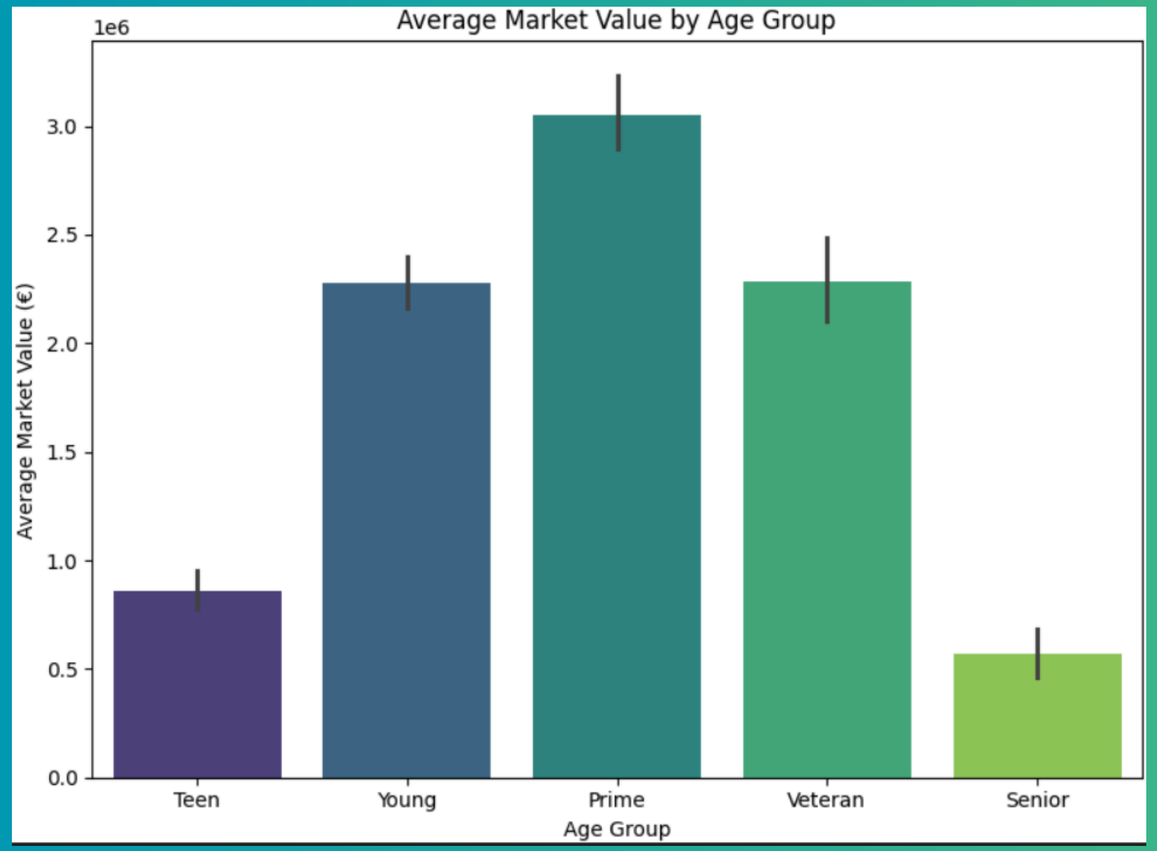
```
ss DataAnalyzer:
def __init__(self, filepath):
    self.filepath = filepath
    self.df = None

def load_data(self):
    try:
        self.df = pd.read_csv(self.filepath)
        self.df.columns = [col.lower() for col in self.df.columns]
        print(" Data loaded successfully.")
        print(" Columns:", self.df.columns.tolist())
        print(self.df.head(3))
    except FileNotFoundError:
        print(" File not found. Please check your filepath.")


def clean_data(self):
    self.df.drop_duplicates(inplace=True)

    if 'value_eur' not in self.df.columns:
        raise ValueError(" 'value_eur' column not found in dataset.")

    self.df.dropna(subset=['age', 'value_eur', 'overall'], inplace=True)

    def convert_value(val):
        try:
            val = val.replace('€', '')
            if 'M' in val:
                return float(val.replace('M', '')) * 1000000
            elif 'K' in val:
                return float(val.replace('K', '')) * 1000
            else:
```

```
        except:
            return np.nan

    if self.df['value_eur'].dtype == 'object':
        self.df['value_eur'] = self.df['value_eur'].apply(convert_value)
    self.df['value_eur'] = self.df['value_eur'].astype(float)
    print(" Data cleaned and transformed.")
    print(" Data shape after cleaning:", self.df.shape)
    print(" Missing values:\n", self.df.isna().sum())

def add_columns(self):
    self.df['agegroup'] = pd.cut(self.df['age'], bins=[15, 20, 25, 30, 35, 45],
    labels=['Teen', 'Young', 'Prime', 'Veteran', 'Senior'])
    self.df['valueperrating'] = self.df['value_eur'] / self.df['overall']
    print(" New columns added.")
    print(self.df.head(5))

def show_statistics(self):
    print(" Descriptive Statistics:")
    print(self.df[['age', 'overall', 'value_eur', 'valueperrating']].describe())
    print("\n Correlation Matrix:")
    print(self.df[['age', 'overall', 'value_eur', 'valueperrating']].corr())
```

VISUALIZATION (MATPLOTLIB & SEABORN)

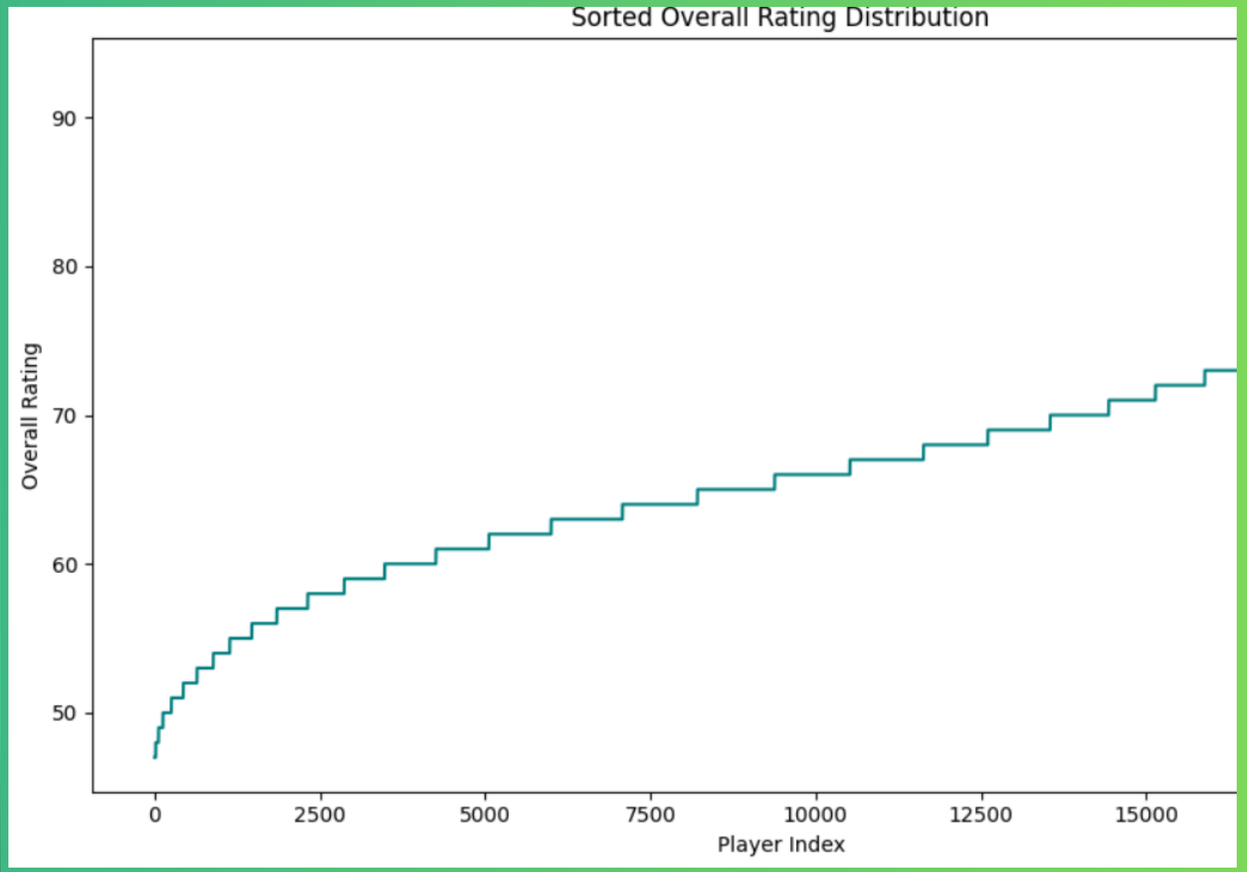
Average Market Value by Age Group

Prime-aged players between 25 and 30 years old tend to have the highest average market value. Younger players generally have lower values, although there are some notable exceptions.



Sorted Overall Rating Distribution

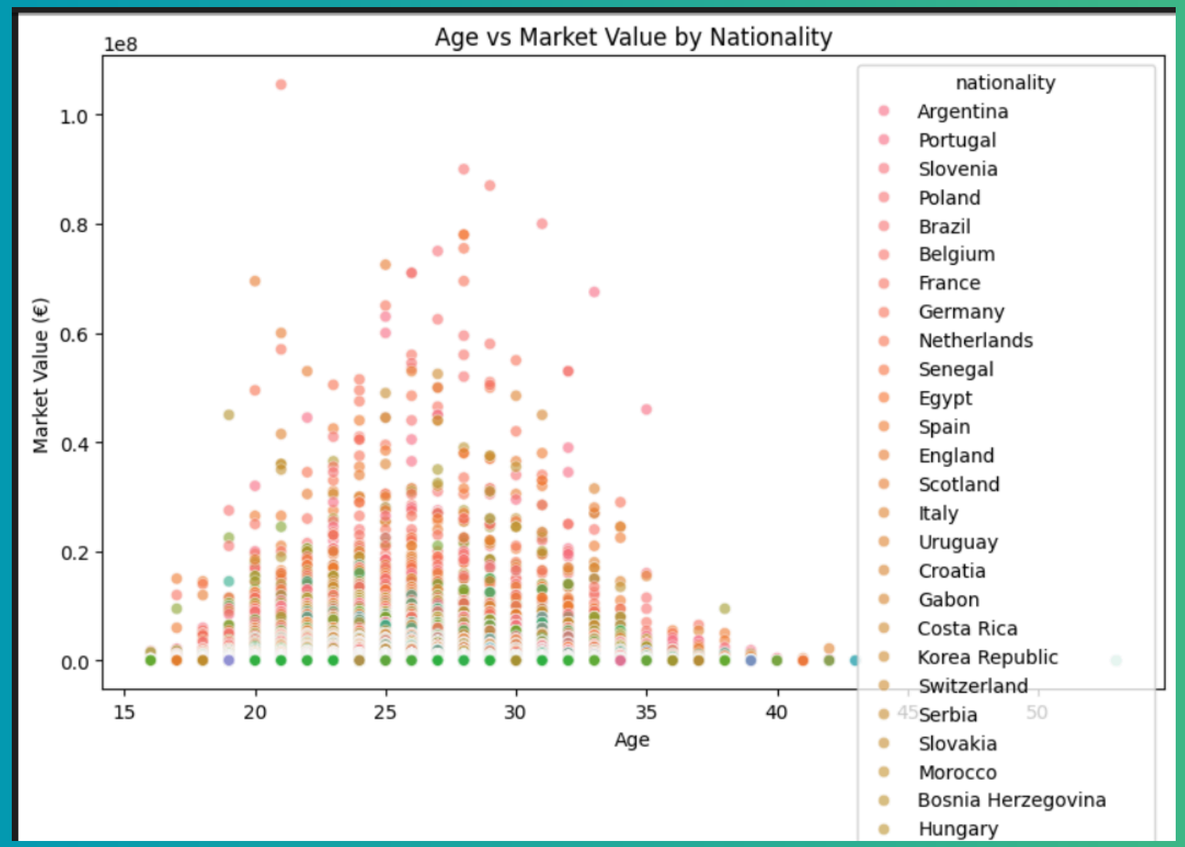
Most players are rated in the mid-70s range. Only a few elite performers exceed an overall rating of 85.



VISUALIZATION (MATPLOTLIB & SEABORN)

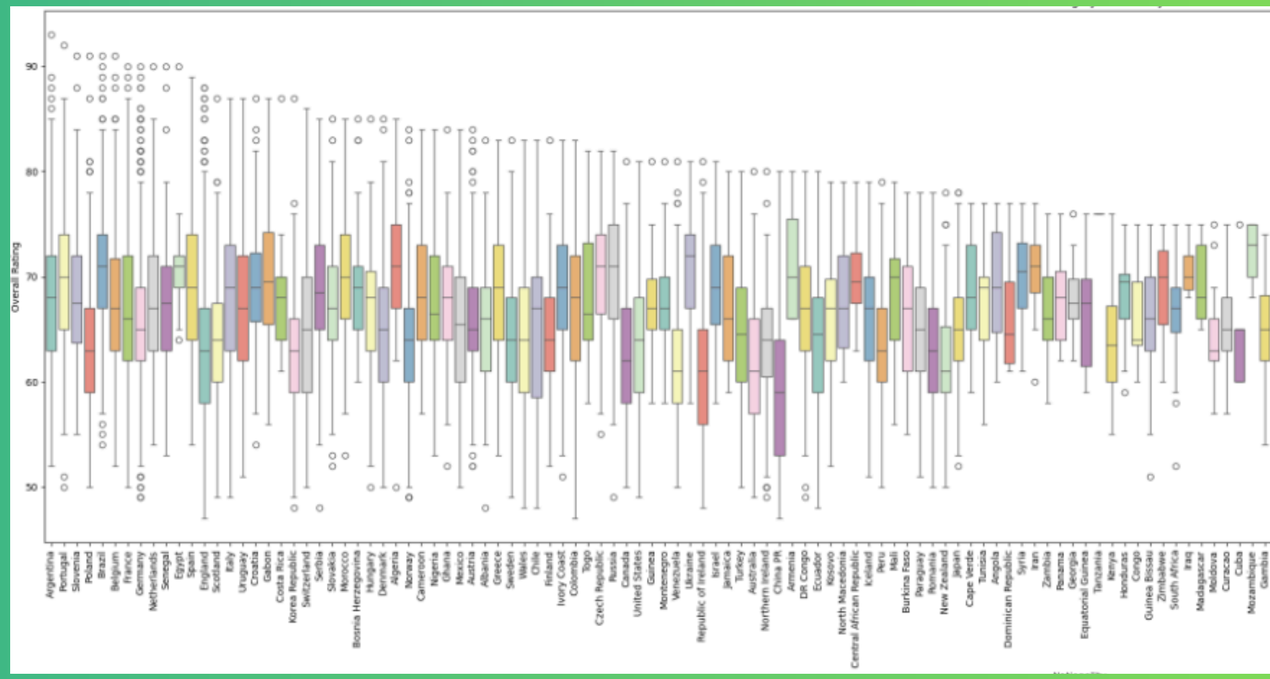
Age vs Market Value by Category

Market value trends vary significantly depending on position, club, or nationality. Some younger players in certain roles achieve high valuations early in their careers.



Sorted Overall Rating Distribution

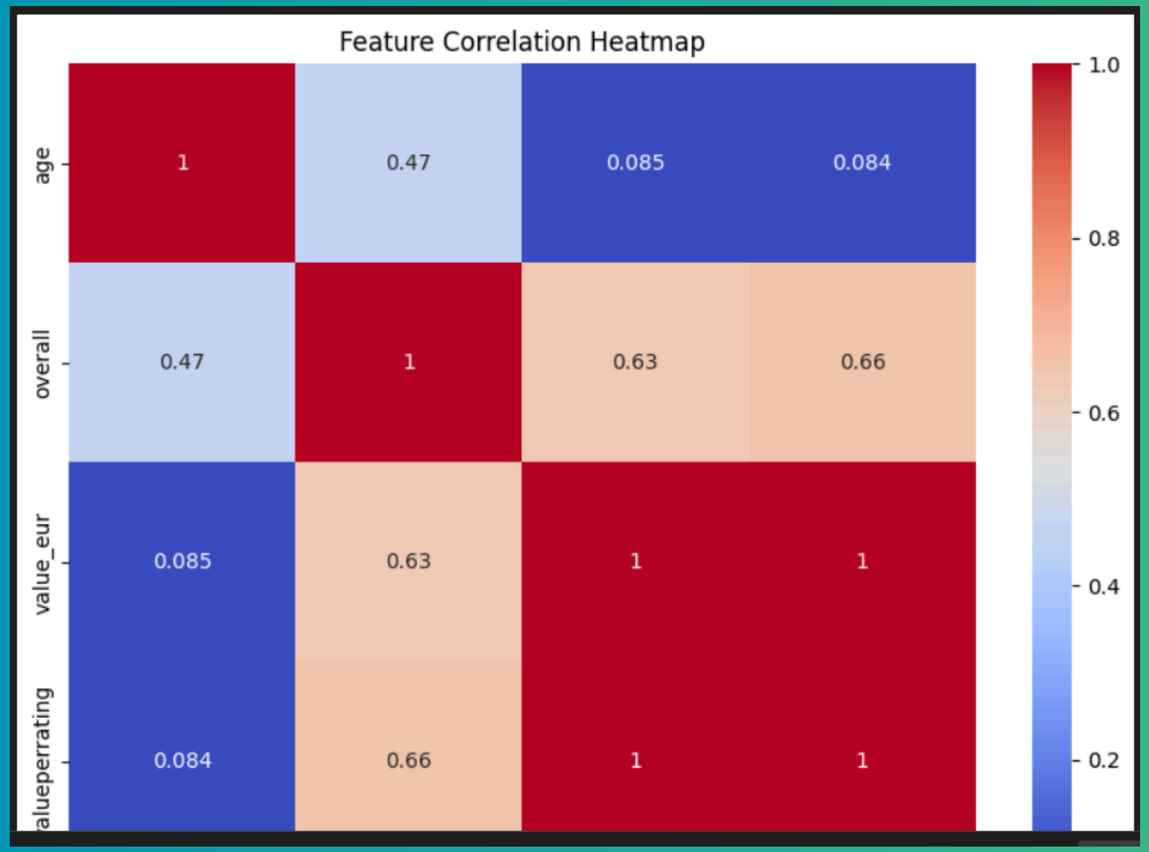
Most players are rated in the mid-70s range. Only a few elite performers exceed an overall rating of 85.



VISUALIZATION (MATPLOTLIB & SEABORN)

Feature Correlation Heatmap

Market value is strongly correlated with overall rating. Age shows a moderate positive correlation with market value. The value-per-rating metric can help identify players who might be undervalued.



```
def create_visuals(self):
    print("Generating visualizations with insights...")

    possible_cats = ['position', 'club', 'nationality', 'team_position', 'player_positions']
    category_col = next((col for col in possible_cats if col in self.df.columns), None)



    plt.figure(figsize=(8, 6))
    sns.barplot(x='agegroup', y='value_eur', data=self.df, estimator=np.mean, palette='viridis')
    plt.title('Average Market Value by Age Group')
    plt.xlabel('Age Group')
    plt.ylabel('Average Market Value (€)')
    plt.tight_layout()
    plt.show()
    print(" Insight: Prime-aged players (25–30) tend to have the highest average market value.")



    plt.figure(figsize=(10, 6))
    self.df['overall'].sort_values().reset_index(drop=True).plot(kind='line', color='teal')
    plt.title('Sorted Overall Rating Distribution')
    plt.xlabel('Player Index')
    plt.ylabel('Overall Rating')
    plt.tight_layout()
    plt.show()
    print(" Insight: Most players cluster around the mid-70s rating, with fewer elite performers.")
```

```
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='age', y='value_eur', hue=category_col, data=self.df, alpha=0.6)
    plt.title(f'Age vs Market Value by {category_col.capitalize()}')
    plt.xlabel('Age')
    plt.ylabel('Market Value (€)')
    plt.tight_layout()
    plt.show()
    print(f" Insight: Market value trends vary across {category_col}.")


    plt.figure(figsize=(30, 10))
    sns.boxplot(x=category_col, y='overall', data=self.df, palette='Set3')
    plt.xticks(rotation=90)
    plt.title(f'Overall Rating by {category_col.capitalize()}')
    plt.xlabel(category_col.capitalize())
    plt.ylabel('Overall Rating')
    plt.tight_layout()
    plt.show()
    print(f" Insight: Rating distributions differ across {category_col}.")


    plt.figure(figsize=(8, 6))
    sns.heatmap(self.df[['age', 'overall', 'value_eur', 'valueperrating']].corr(), annot=True, cmap='coolwarm')
    plt.title('Feature Correlation Heatmap')
    plt.tight_layout()
    plt.show()
    print(" Insight: Market value is strongly correlated with overall rating and moderately with age.")
    print(" Visualizations complete.")

__name__ == "__main__":
 analyzer = DataAnalyzer("players 21.csv")
```

THANK YOU