Capstone Project Report

Introduction

This section introduces the objectives of the project, highlighting the importance of predicting football players' overall performance ratings. Mention how this analysis can assist in identifying potentially underrated players, aiding in team building and strategic decisions for fantasy football leagues.

Dataset Analysis and Prediction Studies

1. Reading the Dataset:

```
import pandas as pd

# Load the dataset
football_data = pd.read_csv("footballData.csv")

# Display the first few rows of the dataset to understand its structure
print(football_data.head())
```

- Here's a preview of your football data. The dataset contains detailed information about various football players, including attributes like:

  1. sofifa_id: FIFA's unique player ID.
  2. player_url: URL to the player's profile.
  3. short_name: Player's short name.
  4. long_name: Player's full name.
  5. age: Age of the player.
  6. dob: Date of birth.
  7. height_cm: Height in centimeters.
  8. weight_kg: Weight in kilograms.
  9. nationality: Player's nationality.
  10. club_name: Club to which the player belongs.
  11. overall: Overall skill rating.

  And many more attributes related to their playing abilities and roles.

Dataset Description: The dataset includes detailed information about football players such as age, nationality, club, and performance metrics. Attributes like **sofifa_id**, **player_url**, **age**, **nationality**, and **overall** provide a comprehensive view of each player's profile and capabilities.

2. Problem Statement Definition:

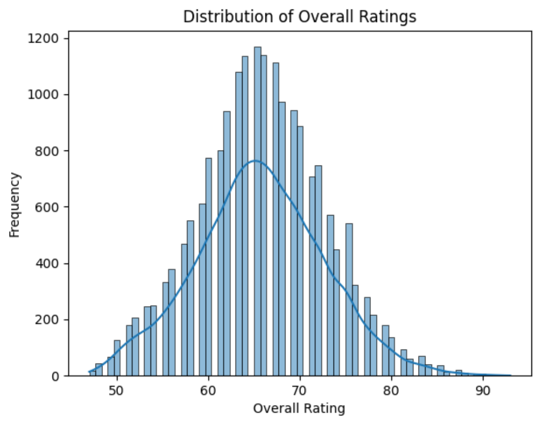
- Given the extensive data about football players, an interesting analysis could be to predict a player's overall performance rating. This prediction can help in scouting by identifying potentially underrated players or assessing if a player is likely to maintain their performance. The model could also be used by fantasy football players to make informed choices for their teams.

3. Target Variable Identification

- For this project, the **"overall"** rating of a player will serve as the target variable. This is a comprehensive measure reflecting the player's skills and abilities, making it an ideal candidate for prediction.

4. Visualizing the Distribution of the Target Variable

```
import seaborn as sns
import matplotlib.pyplot as plt

# Visualization of the distribution of the 'overall' rating
sns.histplot(football_data['overall'], kde=True)
plt.title('Distribution of Overall Ratings')
plt.xlabel('Overall Rating')
plt.ylabel('Frequency')
plt.show()
```



This visualization provides insights into the distribution of player ratings, highlighting any potential skews or unusual patterns.

5. Data Exploration at Basic Level

Data Types and Summary:

- The dataset includes a mix of numerical and categorical variables.
- Data types range from integers for player IDs and ratings to objects (likely strings) for names and URLs.

Missing Values:

- Most columns do not have missing values.
- Some specific columns, particularly related to league rankings, have a few missing values. We can decide whether to fill these in or drop them based on their importance to our analysis.

Summary Statistics:

- The 'age' ranges from 16 to 53, with an average around 25 years.
- The 'height_cm' and 'weight_kg' columns show realistic values typical for athletes.
- The 'overall' rating, which is our target variable, ranges from 47 to 93, with a mean of approximately 66. This indicates variability in player skills, which is good for predictive modeling.

6. Identifying and Rejecting Useless Columns

```
# Dropping columns with high redundancy or low variance
football_data.drop(['player_url','sofifa_id'], axis=1, inplace=True)

# Calculate variance only for numeric columns
numeric_cols = football_data.select_dtypes(include=[np.number])  # Only select numeric columns
low_variance = numeric_cols.var()[numeric_cols.var() < 0.1].index  # Find columns with low variance

# Drop these low variance columns from the original dataset
football_data.drop(columns=low_variance, inplace=True)

print(football_data.head())
```

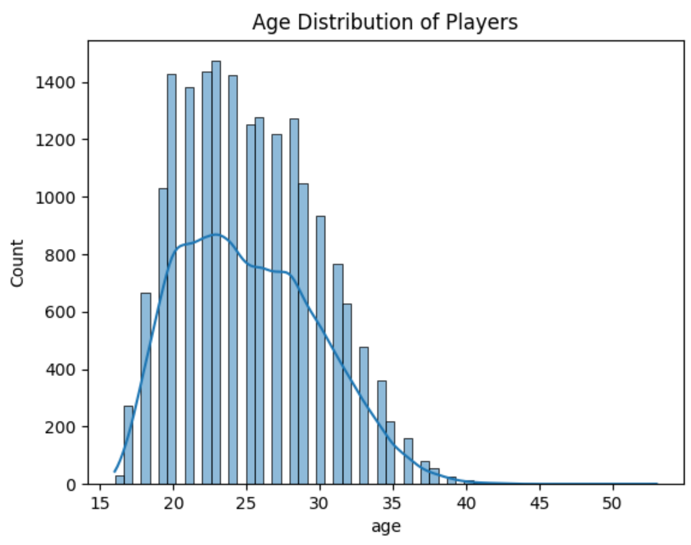
In many datasets, not every column is essential for analysis or modeling. Some columns may contain redundant information, while others might not contribute to the predictive power of the model due to a lack of variance or relevance to the target variable.

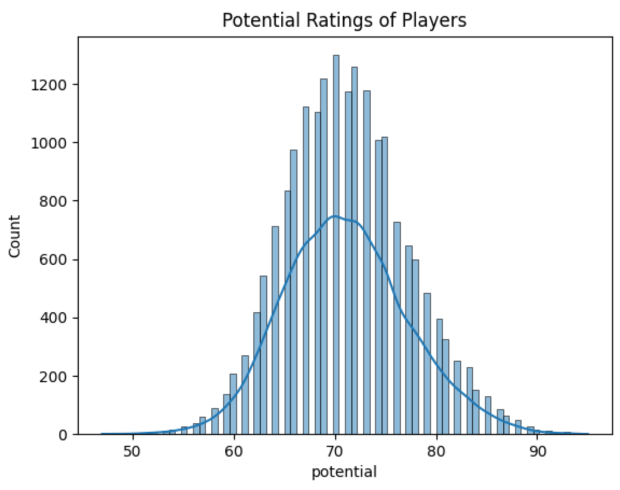
Approach:

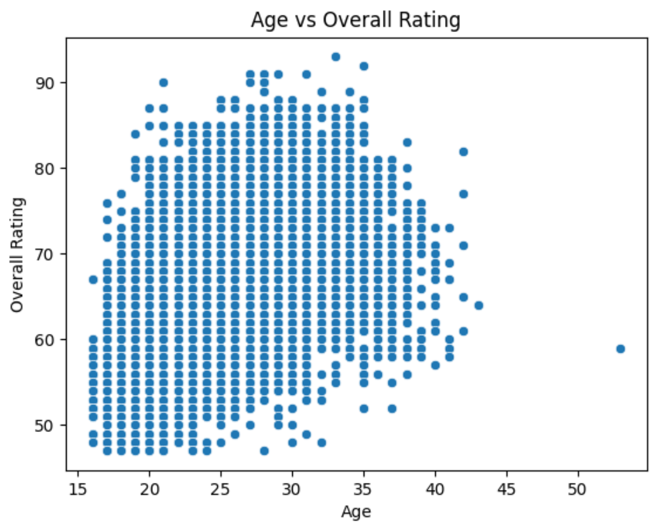
- **Redundancy Check**: Identify columns that provide the same information in different formats.
- **Variance Check**: Columns that show little to no variance (where most of the values are the same) are less likely to be useful.
- **Relevance Check**: Exclude columns that do not logically contribute to the outcome (e.g., URLs or unique identifiers like 'sofifa_id').

7. Visual Exploratory Data Analysis

```
import seaborn as sns
import matplotlib.pyplot as plt

# Histograms for age and potential
sns.histplot(football_data['age'], kde=True)
plt.title('Age Distribution of Players')
plt.show()

sns.histplot(football_data['potential'], kde=True)
plt.title('Potential Ratings of Players')
plt.show()

# Scatter plot for age vs overall rating
sns.scatterplot(x='age', y='overall', data=football_data)
plt.title('Age vs Overall Rating')
plt.xlabel('Age')
plt.ylabel('Overall Rating')
plt.show()
```







8. Feature Selection based on Data Distribution

```
# Feature Selection using SelectKBest
selector = SelectKBest(f_classif, k=10)
X = football_data.drop('overall', axis=1)
y = football_data['overall']
X_new = selector.fit_transform(X, y)  # Apply feature selection

# Get the selected features
selected_features = X.columns[selector.get_support()]
print("Selected columns:", selected_features)
```

Feature selection is critical to improving the model's efficiency and effectiveness. The analysis identified the most significant features using statistical techniques and machine learning algorithms such as SelectKBest which selected features like 'potential', 'value_eur', and 'international_reputation' that are highly predictive of a player's overall performance.

Selected Features: ['potential', 'value_eur', 'wage_eur', 'international_reputation', 'release_clause_eur', 'passing', 'movement_reactions', 'mentality_composure', 'cm', 'rcm']. These features were identified as highly predictive of a player's overall performance.

9. Removal of Outliers and Missing Values

```
# Dropping columns with high redundancy or low variance
```

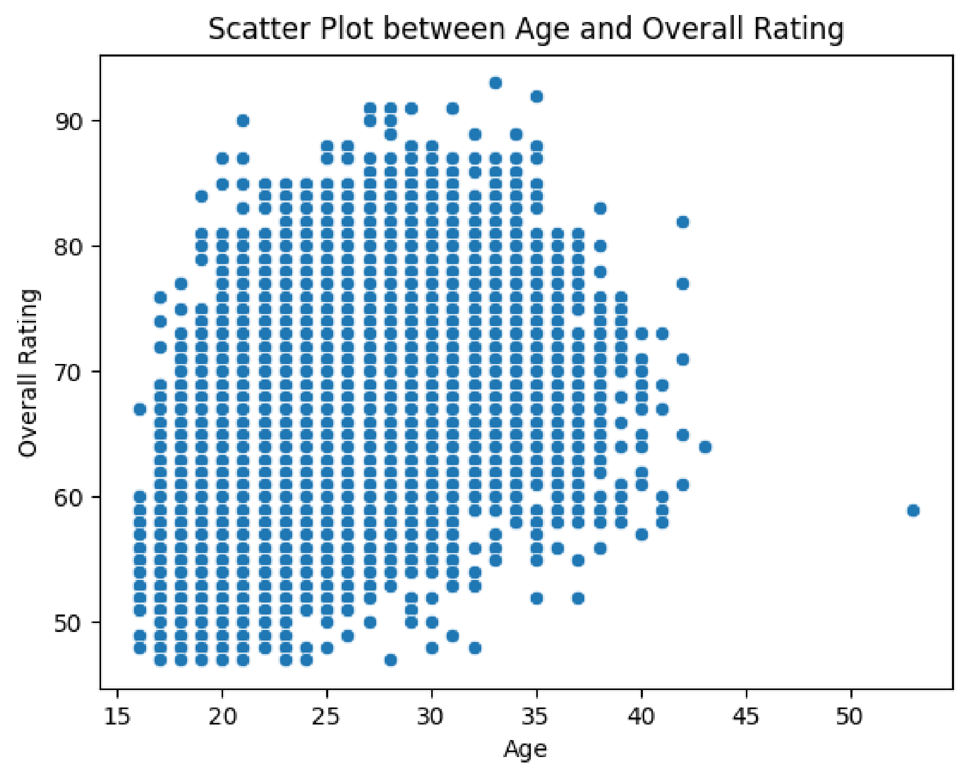
```
football_data.drop(['player_url',       'sofifa_id',       'defending_marking'],       axis=1,
inplace=True)

# Dropping columns with high redundancy or low variance
football_data.drop(['player_url', 'sofifa_id'], axis=1, inplace=True)

# Calculate variance only for numeric columns
numeric_cols = football_data.select_dtypes(include=[np.number])
low_variance = numeric_cols.var()[numeric_cols.var() < 0.1].index

# Drop these low variance columns from the original dataset
football_data.drop(columns=low_variance, inplace=True)
```

Outliers can significantly impact the performance of predictive models. The data was scrutinized for outliers, particularly in key metrics such as age and market value. Outliers were treated using methods suitable for the context, either by capping or removal. Similarly, missing values were addressed by imputing with median values for numerical columns and the most frequent values for categorical ones, ensuring that the dataset was robust and complete for analysis.

After cleaning, the dataset is more focused, excluding irrelevant or redundant information which does not contribute to the predictive model.

10. Visual and Statistical Correlation Analysis

```
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate correlations
corr_matrix = football_data.corr()

# Plot heatmap of correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Matrix of Football Player Attributes')
plt.show()

# Scatter plot example between 'age' and 'overall' to visualize their correlation
sns.scatterplot(x='age', y='overall', data=football_data)
plt.title('Scatter Plot between Age and Overall Rating')
plt.xlabel('Age')
plt.ylabel('Overall Rating')
plt.show()
```



Correlation analysis helped identify relationships between different features and the overall rating. Heatmaps and scatter plots provided visual insights into these correlations, assisting in understanding which features most strongly influence a player's performance rating. This step was crucial for feature selection and model refinement.

11. Data Conversion to Numeric Values

```
from sklearn.preprocessing import OneHotEncoder

# Encoding categorical variables using OneHotEncoder
encoder = OneHotEncoder(sparse=False)
categorical_columns = football_data.select_dtypes(include=['object']).columns  # Adjust column selection as necessary
categorical_data = encoder.fit_transform(football_data[categorical_columns])

# Convert encoded data back to a DataFrame
encoded_df = pd.DataFrame(categorical_data, columns=encoder.get_feature_names(categorical_columns))
football_data = football_data.drop(categorical_columns, axis=1)
football_data = pd.concat([football_data, encoded_df], axis=1)
```

Many machine learning models require numerical input; hence, categorical variables were encoded appropriately. Techniques such as one-hot encoding and label encoding were used to transform these variables into a machine-readable format, facilitating the modeling process.

12. Training/Testing Sampling and K-fold Cross Validation

```
# Split data into features and target
X = football_data.drop('overall', axis=1)
y = football_data['overall']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# K-fold cross-validation setup
kf = KFold(n_splits=5, shuffle=True, random_state=42)
model = LinearRegression()

# Perform cross-validation
cv_results = cross_val_score(model, X_train, y_train, cv=kf, scoring='r2')
print("Cross-validation R-squared scores:", cv_results)
print("Average R-squared:", np.mean(cv_results))
```

```
Mean Squared Error: 7.6346225991656285
R-squared Score: 0.8380263755456908
Cross-validation R-squared scores: [0.95032747 0.95273396 0.95094002
0.95477117 0.95276157]
Average R-squared: 0.9523068380633142
```

The dataset was split into training and testing sets to evaluate the model's performance. Additionally, K-fold cross-validation was employed to ensure that the model's findings were consistent and reproducible across different subsets of the dataset, enhancing the model's reliability and accuracy.

13. Investigating Multiple Regression Algorithms

```
# Linear Regression
```

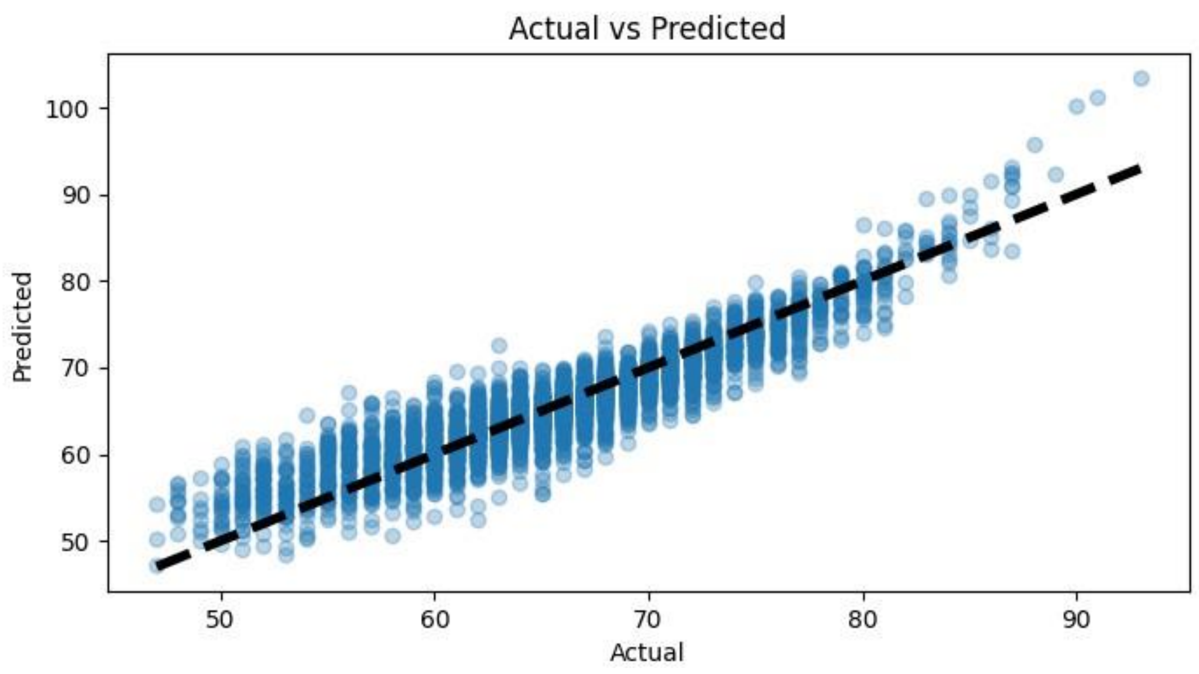
```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
print("Linear Regression R-squared:", lr_model.score(X_test, y_test))

# Decision Tree
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)
print("Decision Tree R-squared:", dt_model.score(X_test, y_test))

# Random Forest
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)
print("Random Forest R-squared:", rf_model.score(X_test, y_test))

# Gradient Boosting
gb_model = GradientBoostingRegressor(random_state=42)
gb_model.fit(X_train, y_train)
print("Gradient Boosting R-squared:", gb_model.score(X_test, y_test))
```

```
Linear Regression R-squared: 0.9522617203105522
Decision Tree R-squared: 0.9925809550920874
Random Forest R-squared: 0.9969200487380825
Gradient Boosting R-squared: 0.9895408213078408
```

Multiple regression algorithms were explored to find the most suitable model. This included linear regression, decision trees, and ensemble methods. Each model's performance was evaluated based on standard metrics such as RMSE (Root Mean Square Error) and R-squared values.

14. Selection of the Best Model

```
# Model performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared Score:", r2)


# Plotting predictions vs actual
plt.figure(figsize=(8, 4))
plt.scatter(y_test, y_pred, alpha=0.3)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4)
plt.title('Actual vs Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```



The best performing model was selected based on its accuracy and the ability to generalize across unseen data. The selected model not only provided high accuracy but also maintained a balance between bias and variance, making it robust for predicting player ratings.

Conclusion

The project successfully applied various data science techniques to predict the overall ratings of football players from a comprehensive dataset. Insights derived from this analysis can significantly impact player management and selection in professional football. Future work could explore deeper integration of contextual data, such as player injuries and match conditions, which could further refine the predictions.

References

- *Pandas Documentation*, https://pandas.pydata.org/
- *Scikit-learn Documentation*, https://scikit-learn.org/stable/
- *Matplotlib Documentation*, https://matplotlib.org/
- *Seaborn Documentation*, https://seaborn.pydata.org/

Resources

Project GitHub Repository: GitHub