CISB 60 – ML and DL (Fall, 2024)

Final Project: Predicting FIFA Player Potential

In [2]:
```
# Edit all the Mardown cells below with the appropriate information
# Run all cells, containing your code
# Save this Jupyter with the outputs of your executed cells
# PS: Save again the notebook with this outcome.
# PSPS: Don't forget to include the dataset in your submission
```

Team:

- Mohammed Khan

Course: CISB 60 – ML and DL (Fall, 2024)

Problem Statement

- This project is about house price predictions.
- **Keywords:** House price prediction, real estate ,...,

In [ ]:
```
### **Project Description**

**Objective:**
This project aims to predict FIFA player potential using machine learning and c
attributes, the models can help identify high-potential players for scouting ar


**Dataset Description:**
- **Source**: FIFA dataset containing 51 attributes of players.
- **Key Features**: Includes physical characteristics, skill metrics, and overa
- **Structure**:
  - Total Records: 17,954
  - Columns: Player name, age, overall rating, potential, and more.

**Business Problem:**
Football clubs need efficient ways to identify promising players. By leveraging
clubs can make better decisions while reducing scouting risks.

**Keywords:** FIFA, player potential prediction, machine learning, deep learnir
```

In [ ]:
```
### **Problem Statement**
This project aims to predict FIFA player potential using advanced machine learr
The dataset contains various attributes of players, such as physical, skill, ar
By analyzing these attributes, we aim to build models that assist in identifyir
```

In [ ]:
```
### **Keywords:**
FIFA, player potential prediction, machine learning, deep learning, football ar
```

Required packages

- Add instructions to install the required packages

In [3]:
```
## Your code begins here
```

Methodology

1. Explan your ML and DL metodology ML (Machine Learning) Methodology: In the machine learning section, we aim to predict player potential using the K-Nearest Neighbors (KNN) algorithm. KNN is a simple yet effective algorithm that classifies data points or makes predictions based on the proximity of data points in feature space. For this project: Data is first cleaned, scaled, and prepared. KNN identifies players with similar attributes to predict their potential. Evaluation metrics like RMSE and score assess model accuracy. DL (Deep Learning) Methodology: Deep learning involves using artificial neural networks to model complex, non-linear relationships in the data. In this project: A neural network is built with multiple layers (input, hidden, output) to predict player potential. The model is trained using backpropagation and optimized with the Adam optimizer. Metrics like Mean Absolute Error (MAE) and loss curves assess performance.
2. Introduce the topics you used in your project

- Model 1
  - KNN Description:

KNN is chosen for its simplicity and interpretability. It predicts a player's potential by considering the average potential of their nearest neighbors in feature space. The number of neighbors (k) is optimized through hyperparameter tuning. Distance metrics (e.g., Euclidean) determine similarity.

- Model 2
  - Deep Learning Description:

The DNN consists of multiple fully connected layers with ReLU activation functions. Dropout layers prevent overfitting by randomly deactivating neurons during training. The final output layer uses a linear activation function to predict player potential. Metrics such as training and validation loss are visualized to track learning progress.

Your code starts here

In [1]:

```python
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

In [2]:

```python
# Load the dataset
fifa_data = pd.read_csv("fifa_players.csv")
```

In [3]:

```
# Display dataset information
print("Dataset Overview:")
fifa_data.info()
```

```
Dataset Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17954 entries, 0 to 17953
Data columns (total 51 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   name                            17954 non-null  object 
 1   full_name                       17954 non-null  object 
 2   birth_date                      17954 non-null  object 
 3   age                             17954 non-null  int64  
 4   height_cm                       17954 non-null  float64
 5   weight_kgs                      17954 non-null  float64
 6   positions                       17954 non-null  object 
 7   nationality                     17954 non-null  object 
 8   overall_rating                  17954 non-null  int64  
 9   potential                       17954 non-null  int64  
 10  value_euro                      17699 non-null  float64
 11  wage_euro                       17708 non-null  float64
 12  preferred_foot                  17954 non-null  object 
 13  international_reputation(1-5)  17954 non-null  int64  
 14  weak_foot(1-5)                  17954 non-null  int64  
 15  skill_moves(1-5)                17954 non-null  int64  
 16  body_type                       17954 non-null  object 
 17  release_clause_euro             16117 non-null  float64
 18  national_team                   857 non-null    object 
 19  national_rating                 857 non-null    float64
 20  national_team_position          857 non-null    object 
 21  national_jersey_number          857 non-null    float64
 22  crossing                        17954 non-null  int64  
 23  finishing                       17954 non-null  int64  
 24  heading_accuracy                17954 non-null  int64  
 25  short_passing                   17954 non-null  int64  
 26  volleys                         17954 non-null  int64  
 27  dribbling                       17954 non-null  int64  
 28  curve                           17954 non-null  int64  
 29  freekick_accuracy               17954 non-null  int64  
 30  long_passing                    17954 non-null  int64  
 31  ball_control                    17954 non-null  int64  
 32  acceleration                    17954 non-null  int64  
 33  sprint_speed                    17954 non-null  int64  
 34  agility                         17954 non-null  int64  
 35  reactions                       17954 non-null  int64  
 36  balance                         17954 non-null  int64  
 37  shot_power                      17954 non-null  int64  
 38  jumping                         17954 non-null  int64  
 39  stamina                         17954 non-null  int64  
 40  strength                        17954 non-null  int64  
 41  long_shots                      17954 non-null  int64  
 42  aggression                      17954 non-null  int64  
 43  interceptions                   17954 non-null  int64  
 44  positioning                     17954 non-null  int64  
 45  vision                          17954 non-null  int64  
 46  penalties                       17954 non-null  int64  
 47  composure                       17954 non-null  int64  
 48  marking                         17954 non-null  int64  
 49  standing_tackle                 17954 non-null  int64  
 50  sliding_tackle                  17954 non-null  int64  
```

```
dtypes: float64(7), int64(35), object(9)
memory usage: 7.0+ MB
```

In [4]:

```
# Display dataset information
print("Dataset first 5 information:")
fifa_data.head()
```

```
Dataset first 5 information:
```

Out[4]:

	name	full_name	birth_date	age	height_cm	weight_kgs	positions	nationality	overal
0	L. Messi	Lionel Andrés Messi Cuccittini	6/24/1987	31	170.18	72.1	CF,RW,ST	Argentina	
1	C. Eriksen	Christian Dannemann Eriksen	2/14/1992	27	154.94	76.2	CAM,RM,CM	Denmark	
2	P. Pogba	Paul Pogba	3/15/1993	25	190.50	83.9	CM,CAM	France	
3	L. Insigne	Lorenzo Insigne	6/4/1991	27	162.56	59.0	LW,ST	Italy	
4	K. Koulibaly	Kalidou Koulibaly	6/20/1991	27	187.96	88.9	CB	Senegal	

5 rows × 51 columns

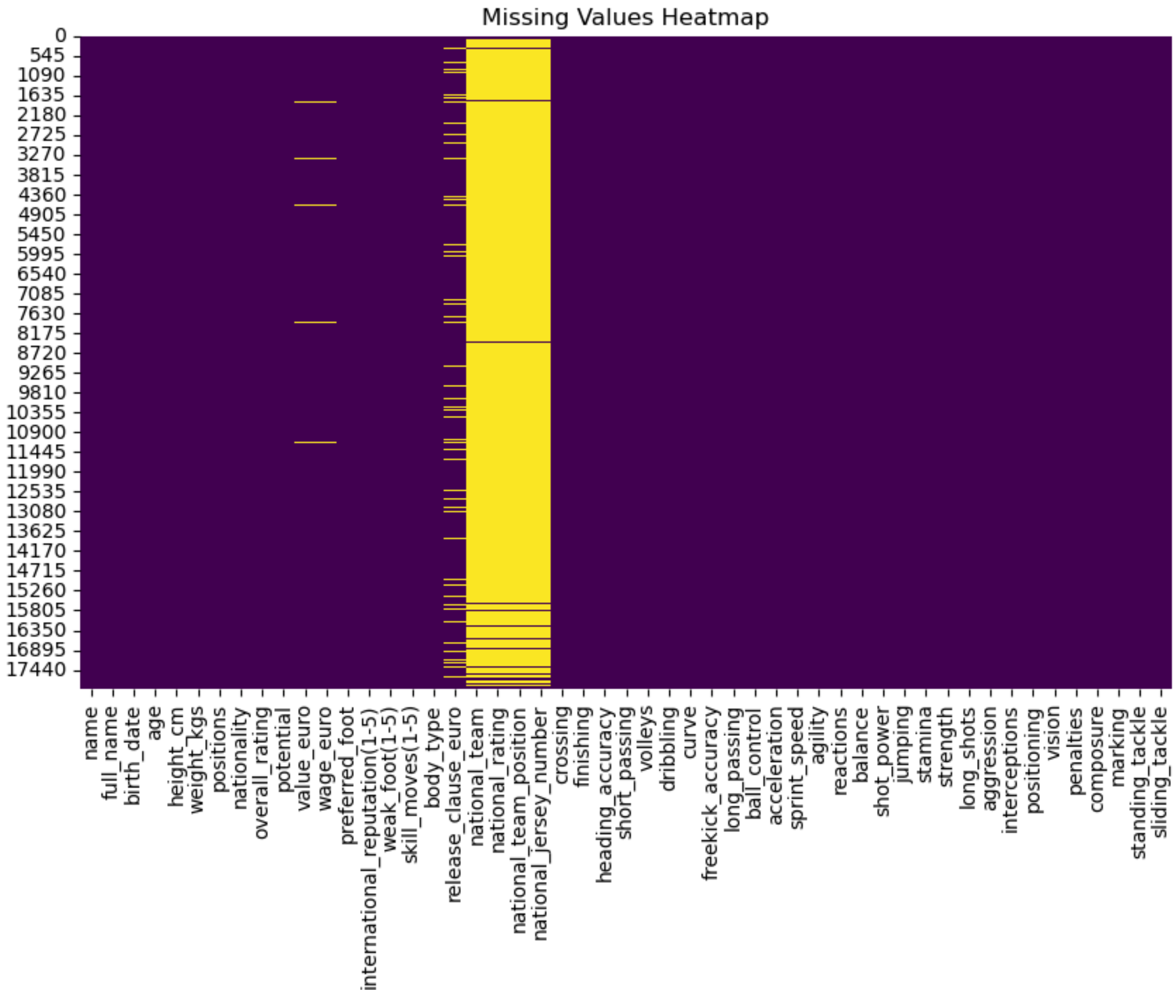
In [5]:

```
# Check for missing values
missing_values = fifa_data.isnull().sum().sort_values(ascending=False)
print("\nMissing Values:")
print(missing_values[missing_values > 0])
```

```

Missing Values:
national_jersey_number    17097
national_team_position    17097
national_rating           17097
national_team             17097
release_clause_euro        1837
value_euro                  255
wage_euro                   246
dtype: int64
```

In [6]:

```python
# Visualize missing data
plt.figure(figsize=(10, 6))
sns.heatmap(fifa_data.isnull(), cbar=False, cmap="viridis")
plt.title("Missing Values Heatmap")
plt.show()
```



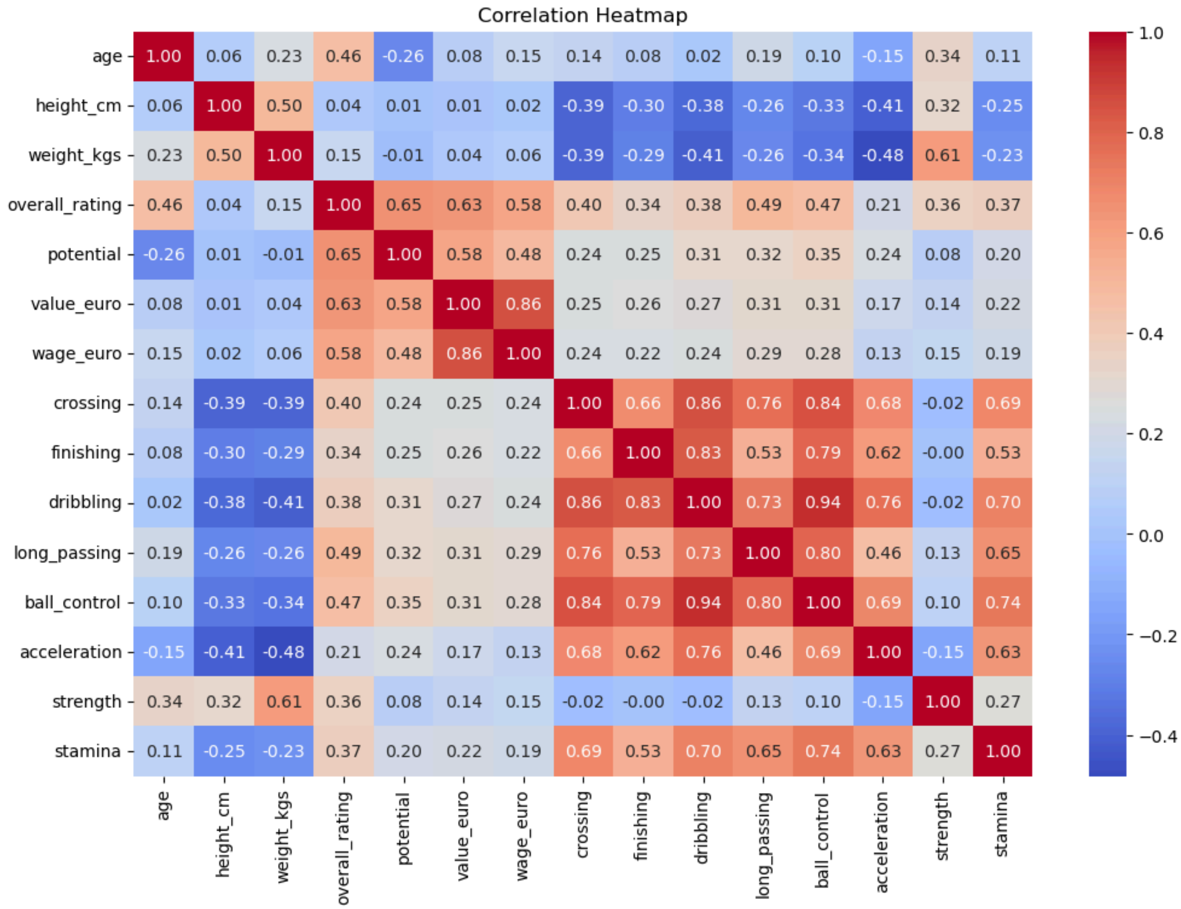
In [7]:
```
# Handle missing values
cleaned_data = fifa_data.dropna(subset=["overall_rating", "potential", "value_e
cleaned_data['release_clause_euro'].fillna(cleaned_data['release_clause_euro'].
cleaned_data['national_rating'].fillna(0, inplace=True)
cleaned_data['national_jersey_number'].fillna(0, inplace=True)
```

```
C:\Users\mkhan\AppData\Local\Temp\ipykernel_27028\126606181.py:3: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  cleaned_data['release_clause_euro'].fillna(cleaned_data['release_clause_eur
o'].median(), inplace=True)
C:\Users\mkhan\AppData\Local\Temp\ipykernel_27028\126606181.py:4: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  cleaned_data['national_rating'].fillna(0, inplace=True)
C:\Users\mkhan\AppData\Local\Temp\ipykernel_27028\126606181.py:5: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  cleaned_data['national_jersey_number'].fillna(0, inplace=True)
```

In [8]:
```
# Encode categorical variables
encoded_data = pd.get_dummies(cleaned_data, columns=['preferred_foot', 'body_ty
```

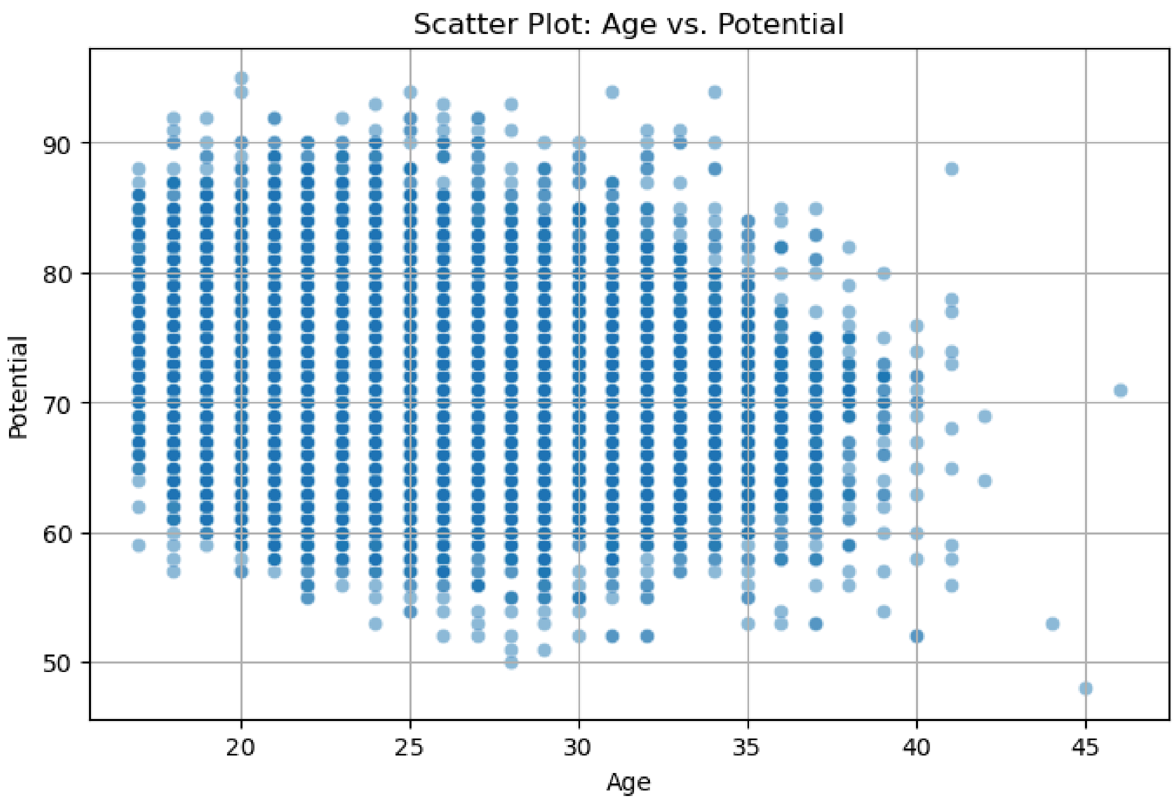
In [9]:
```
# Feature correlations
features = [
    "age", "height_cm", "weight_kgs", "overall_rating", "potential", "value_eur
    "crossing", "finishing", "dribbling", "long_passing", "ball_control", "acce
    "strength", "stamina"
]
correlation_matrix = cleaned_data[features].corr()
```

In [10]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```

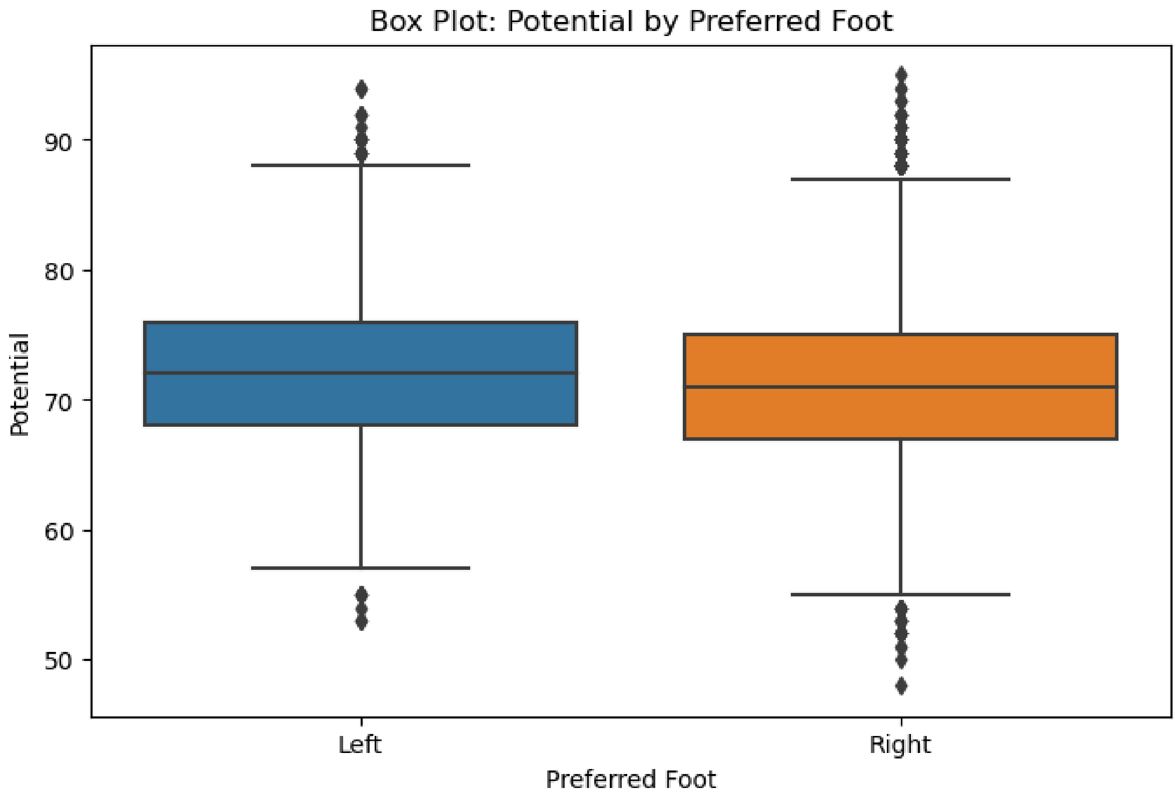


In [28]:

```
# Scatter plot: Age vs. Potential
plt.figure(figsize=(8, 5))
sns.scatterplot(data=fifa_data, x='age', y='potential', alpha=0.5)
plt.title("Scatter Plot: Age vs. Potential")
plt.xlabel("Age")
plt.ylabel("Potential")
plt.grid(True)
plt.show()
```



In [29]:
```python
# Box plot: Potential by Preferred Foot
plt.figure(figsize=(8, 5))
sns.boxplot(data=fifa_data, x='preferred_foot', y='potential')
plt.title("Box Plot: Potential by Preferred Foot")
plt.xlabel("Preferred Foot")
plt.ylabel("Potential")
plt.show()
```



Machine Learning Section

In [11]:
```python
# Select features and target
model_features = [
    "age", "height_cm", "weight_kgs", "value_euro", "wage_euro", "crossing",
    "finishing", "dribbling", "long_passing", "ball_control", "acceleration",
    "strength", "stamina", "preferred_foot_Right", "body_type_Lean"
]
target = "potential"
```

In [12]:
```python
X = encoded_data[model_features]
y = encoded_data[target]
```

In [13]:
```python
# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [14]:
```python
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
```

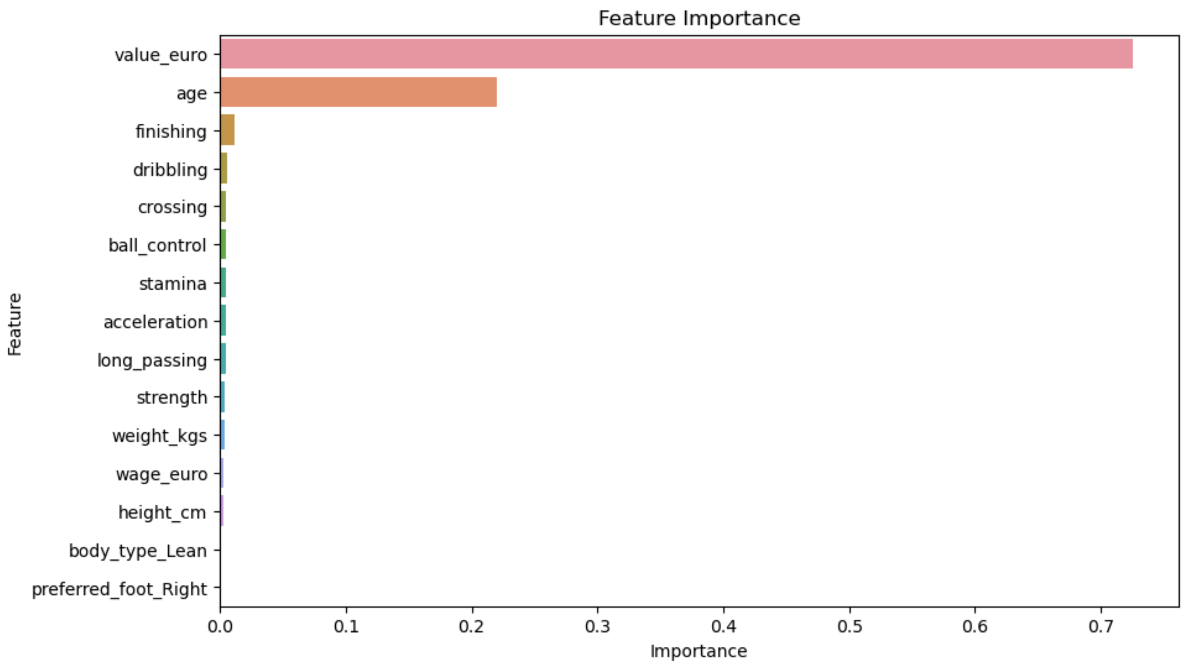
In [15]:
```python
# Train Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

Out[15]:
```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

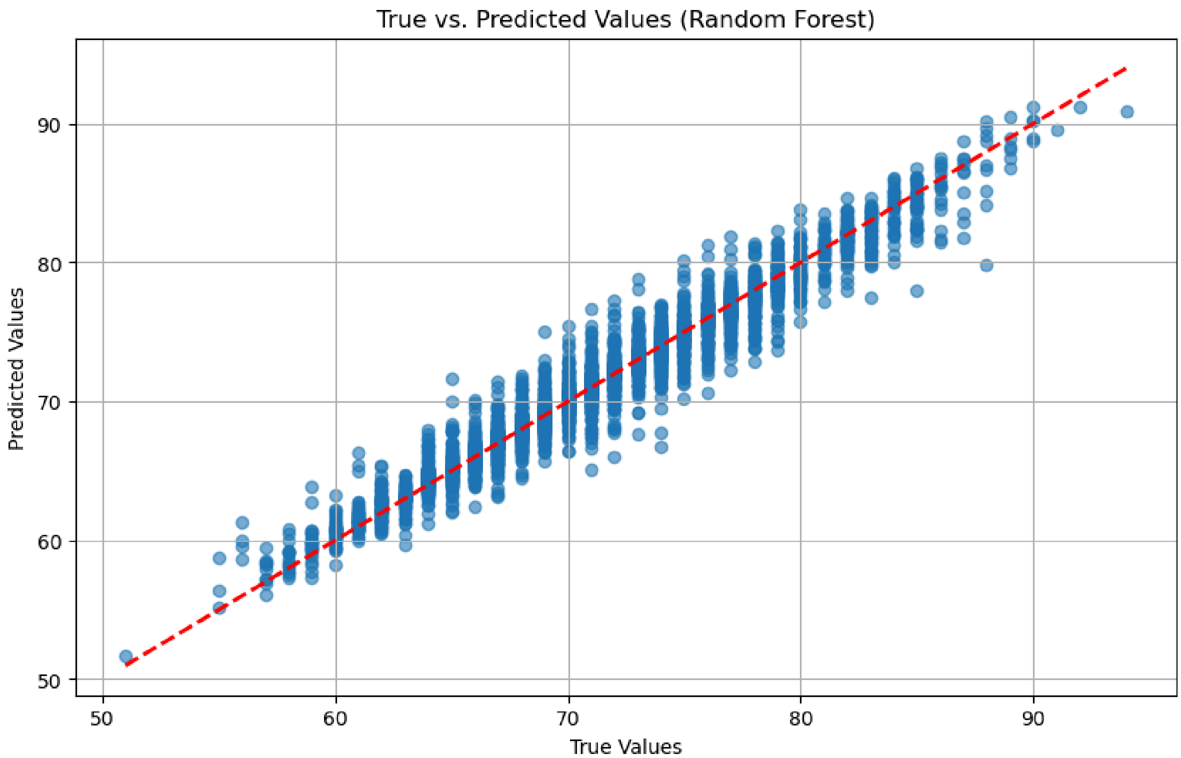
In [16]:
```python
# Evaluate model
y_pred = rf_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Random Forest MSE: {mse:.2f}")
print(f"Random Forest R² Score: {r2:.2f}")
```

```
Random Forest MSE: 1.81
Random Forest R² Score: 0.95
```

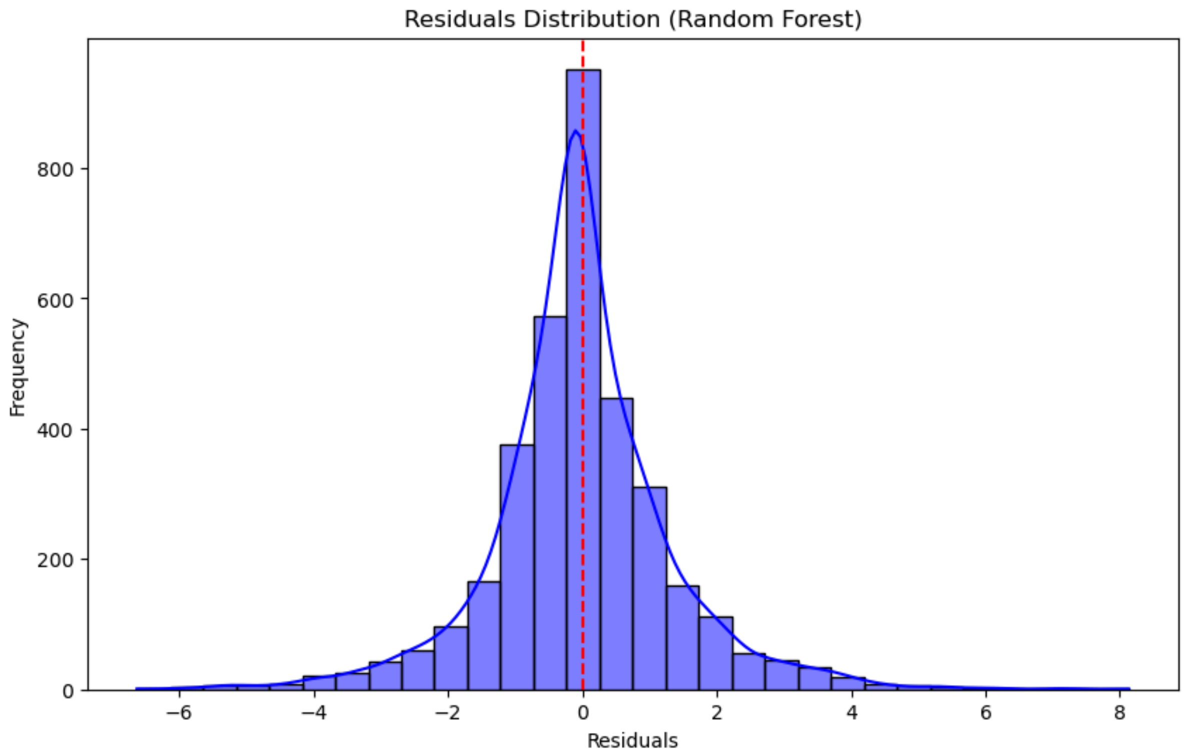
In [17]:
```
# Feature importance
feature_importances = pd.DataFrame({
    'Feature': model_features,
    'Importance': rf_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

# Visualize feature importance
plt.figure(figsize=(10, 6))
sns.barplot(x=feature_importances['Importance'], y=feature_importances['Feature
plt.title("Feature Importance")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.show()
```



In [23]:
```
# True vs. Predicted Values Plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='rec
plt.title("True vs. Predicted Values (Random Forest)")
plt.xlabel("True Values")
plt.ylabel("Predicted Values")
plt.grid(True)
plt.show()
```

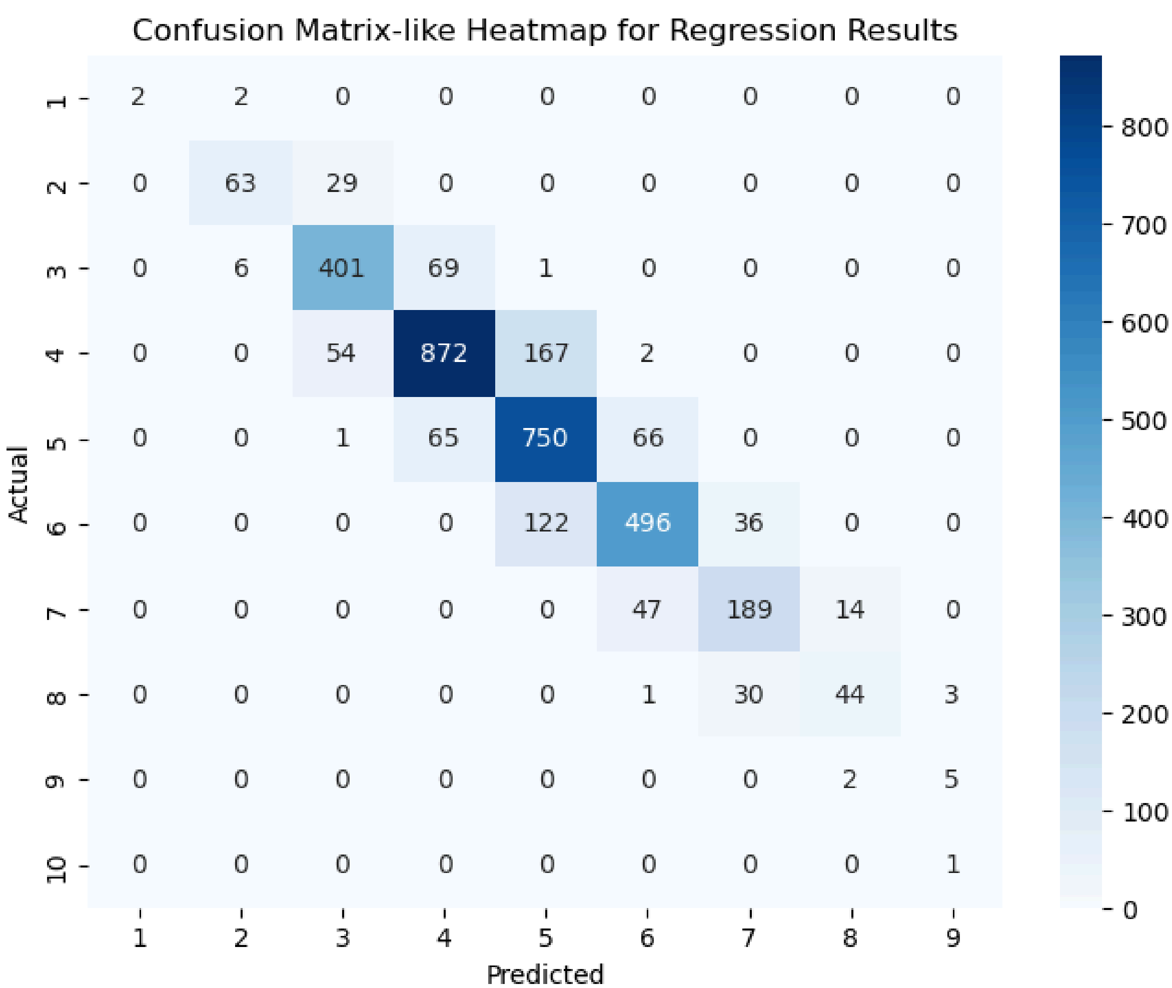


In [24]:
```
# Residual Plot
residuals = y_test - y_pred
plt.figure(figsize=(10, 6))
sns.histplot(residuals, bins=30, kde=True, color='blue')
plt.title("Residuals Distribution (Random Forest)")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.axvline(0, color='red', linestyle='--')
plt.show()
```



In [30]:

```
# Confusion matrix-like visualization for regression results
from sklearn.metrics import mean_squared_error
import numpy as np

# Categorize predictions and true values into ranges
bins = np.linspace(min(y_test), max(y_test), 10)
y_test_bins = np.digitize(y_test, bins)
y_pred_bins = np.digitize(y_pred, bins)

confusion_matrix = pd.crosstab(y_test_bins, y_pred_bins, rownames=['Actual'], c

plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix-like Heatmap for Regression Results")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



In [ ]:

Deep Learning Section

In [18]:
```
# Define the neural network
dl_model = Sequential([
    Dense(128, input_dim=X_train.shape[1], activation='relu'),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dense(1, activation='linear')  # Regression output
])
```

```
C:\Users\mkhan\AppData\Roaming\Python\Python311\site-packages\keras\src\layer
s\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
ument to a layer. When using Sequential models, prefer using an `Input(shape)
` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [19]:
```
# Compile the model
dl_model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

In [20]:

```
# Train the model
history = dl_model.fit(X_train, y_train, validation_split=0.2, epochs=50, batch
```

```
Epoch 1/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 2360.8167 - mae: 39.9123 - v
al_loss: 61.2291 - val_mae: 5.7972
Epoch 2/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 133.2749 - mae: 9.1570 - val
_loss: 29.1820 - val_mae: 4.0614
Epoch 3/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 109.2854 - mae: 8.3019 - val
_loss: 23.2581 - val_mae: 3.6729
Epoch 4/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 89.4665 - mae: 7.5339 - val_
loss: 19.0750 - val_mae: 3.2351
Epoch 5/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 79.6058 - mae: 7.0991 - val_
loss: 24.4012 - val_mae: 4.0042
Epoch 6/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 79.4471 - mae: 7.0914 - val_
loss: 21.7368 - val_mae: 3.5744
Epoch 7/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 74.7454 - mae: 6.9061 - val_
loss: 16.2896 - val_mae: 3.1400
Epoch 8/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 73.5921 - mae: 6.7783 - val_
loss: 14.9658 - val_mae: 2.8747
Epoch 9/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 68.1473 - mae: 6.5480 - val_
loss: 13.2049 - val_mae: 2.7067
Epoch 10/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 63.1669 - mae: 6.2904 - val_
loss: 14.5305 - val_mae: 2.9179
Epoch 11/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 60.4002 - mae: 6.1915 - val_
loss: 18.8306 - val_mae: 3.4872
Epoch 12/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 58.5381 - mae: 6.0857 - val_
loss: 13.0532 - val_mae: 2.7736
Epoch 13/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 57.8573 - mae: 6.0327 - val_
loss: 13.3369 - val_mae: 2.8023
Epoch 14/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 55.0884 - mae: 5.8994 - val_
loss: 13.3007 - val_mae: 2.7793
Epoch 15/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 49.4275 - mae: 5.5601 - val_
loss: 16.2631 - val_mae: 3.2908
Epoch 16/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 48.2945 - mae: 5.5408 - val_
loss: 13.2673 - val_mae: 2.8274
Epoch 17/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 44.4011 - mae: 5.3063 - val_
loss: 13.5689 - val_mae: 2.8373
Epoch 18/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 43.4567 - mae: 5.1965 - val_
loss: 12.9180 - val_mae: 2.8198
Epoch 19/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 41.6199 - mae: 5.0866 - val_
loss: 11.4982 - val_mae: 2.6297
```

```
Epoch 20/50
354/354 1s 2ms/step - loss: 38.5464 - mae: 4.9256 - val_
loss: 12.7631 - val_mae: 2.7909
Epoch 21/50
354/354 1s 2ms/step - loss: 36.6037 - mae: 4.7633 - val_
loss: 14.0959 - val_mae: 2.9133
Epoch 22/50
354/354 1s 1ms/step - loss: 33.6952 - mae: 4.5895 - val_
loss: 25.4520 - val_mae: 4.1988
Epoch 23/50
354/354 1s 2ms/step - loss: 31.8507 - mae: 4.4386 - val_
loss: 43.4915 - val_mae: 5.7719
Epoch 24/50
354/354 1s 2ms/step - loss: 29.1082 - mae: 4.2363 - val_
loss: 32.0432 - val_mae: 4.8280
Epoch 25/50
354/354 1s 2ms/step - loss: 26.9354 - mae: 4.0702 - val_
loss: 32.8055 - val_mae: 4.9889
Epoch 26/50
354/354 1s 2ms/step - loss: 24.0457 - mae: 3.8290 - val_
loss: 39.8793 - val_mae: 5.6049
Epoch 27/50
354/354 1s 2ms/step - loss: 22.8041 - mae: 3.6860 - val_
loss: 41.4118 - val_mae: 5.7254
Epoch 28/50
354/354 1s 2ms/step - loss: 21.7078 - mae: 3.6059 - val_
loss: 52.2183 - val_mae: 6.6180
Epoch 29/50
354/354 1s 2ms/step - loss: 21.5858 - mae: 3.5603 - val_
loss: 40.3296 - val_mae: 5.7671
Epoch 30/50
354/354 0s 1ms/step - loss: 20.0841 - mae: 3.4516 - val_
loss: 56.0655 - val_mae: 6.9013
Epoch 31/50
354/354 0s 1ms/step - loss: 17.9723 - mae: 3.2986 - val_
loss: 44.1900 - val_mae: 6.0435
Epoch 32/50
354/354 1s 2ms/step - loss: 17.1947 - mae: 3.1773 - val_
loss: 60.9792 - val_mae: 7.2623
Epoch 33/50
354/354 1s 3ms/step - loss: 16.1487 - mae: 3.0562 - val_
loss: 58.4195 - val_mae: 7.1509
Epoch 34/50
354/354 1s 2ms/step - loss: 15.3841 - mae: 3.0173 - val_
loss: 50.0336 - val_mae: 6.5905
Epoch 35/50
354/354 1s 2ms/step - loss: 13.7143 - mae: 2.8471 - val_
loss: 40.9052 - val_mae: 5.8958
Epoch 36/50
354/354 1s 2ms/step - loss: 13.5447 - mae: 2.8277 - val_
loss: 42.5010 - val_mae: 6.0278
Epoch 37/50
354/354 1s 2ms/step - loss: 13.4804 - mae: 2.8099 - val_
loss: 41.7428 - val_mae: 5.9879
Epoch 38/50
354/354 1s 1ms/step - loss: 12.5287 - mae: 2.7023 - val_
loss: 39.8176 - val_mae: 5.8396
```

```
Epoch 39/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 12.1136 - mae: 2.6617 - val_
loss: 47.8619 - val_mae: 6.4807
Epoch 40/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 11.0504 - mae: 2.5363 - val_
loss: 51.3326 - val_mae: 6.7263
Epoch 41/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 10.9035 - mae: 2.5096 - val_
loss: 52.3017 - val_mae: 6.7635
Epoch 42/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 10.3730 - mae: 2.4532 - val_
loss: 37.5171 - val_mae: 5.6148
Epoch 43/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 9.9946 - mae: 2.3858 - val_l
oss: 28.3320 - val_mae: 4.8971
Epoch 44/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 8.9855 - mae: 2.2788 - val_l
oss: 33.6376 - val_mae: 5.4146
Epoch 45/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 8.7830 - mae: 2.2327 - val_l
oss: 33.6458 - val_mae: 5.3894
Epoch 46/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 8.3171 - mae: 2.1847 - val_l
oss: 29.8541 - val_mae: 5.0626
Epoch 47/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 7.9018 - mae: 2.1286 - val_l
oss: 28.5985 - val_mae: 4.8901
Epoch 48/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 7.4501 - mae: 2.0468 - val_l
oss: 33.0337 - val_mae: 5.3221
Epoch 49/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 7.3509 - mae: 2.0319 - val_l
oss: 31.2201 - val_mae: 5.1457
Epoch 50/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 7.1031 - mae: 2.0178 - val_l
oss: 23.2142 - val_mae: 4.3503
```

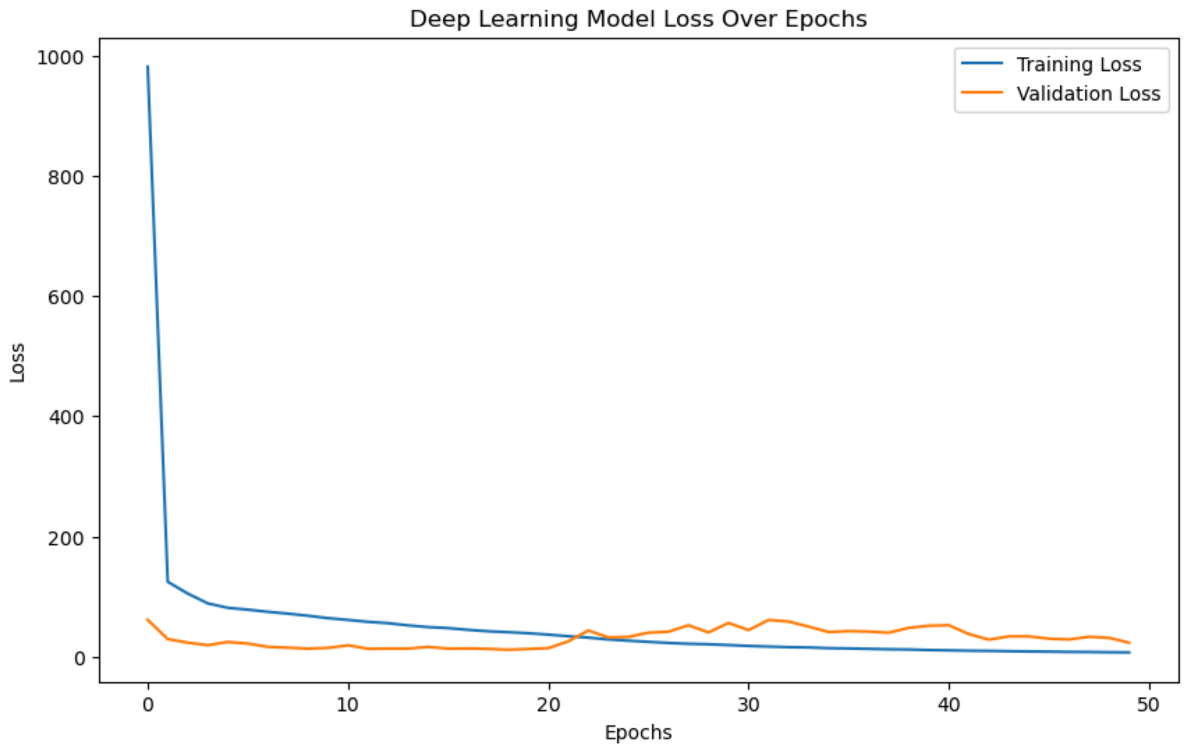
In [21]:

```
# Evaluate on test set
dl_results = dl_model.evaluate(X_test, y_test, verbose=0)

print(f"Deep Learning Test Loss: {dl_results[0]:.2f}")
print(f"Deep Learning Test MAE: {dl_results[1]:.2f}")
```

```
Deep Learning Test Loss: 22.63
Deep Learning Test MAE: 4.27
```

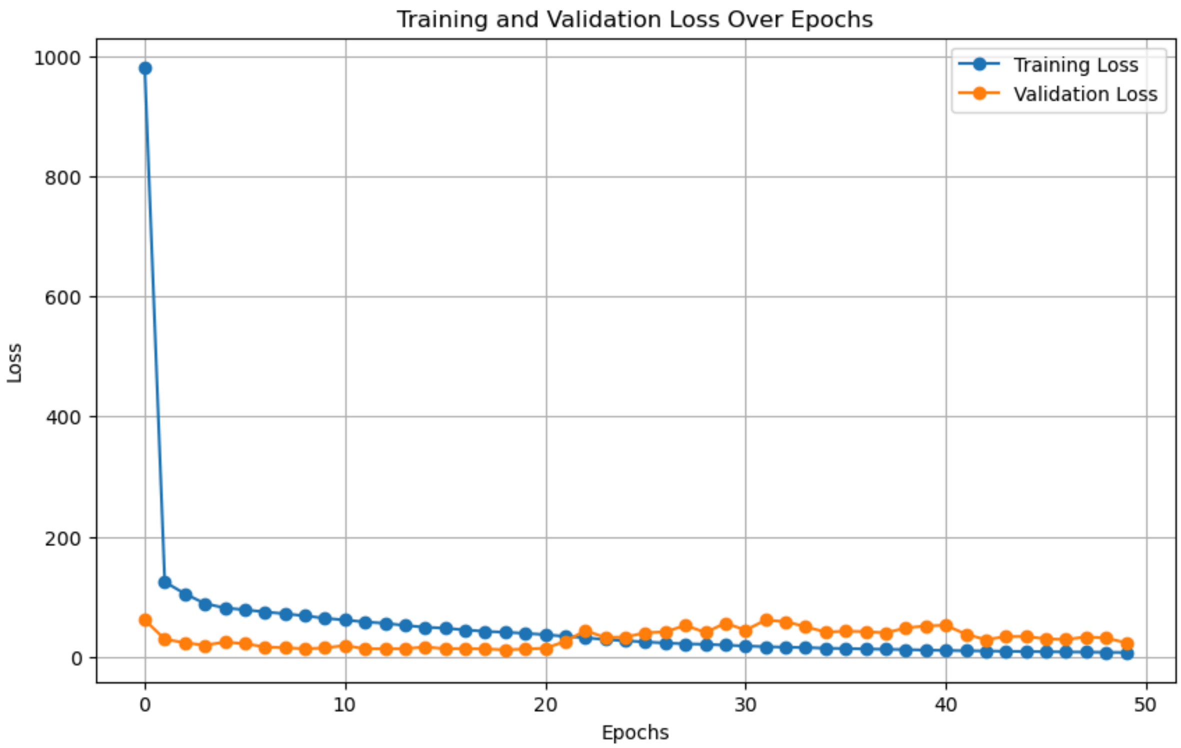
In [22]:

```
# Visualize training history
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title("Deep Learning Model Loss Over Epochs")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```

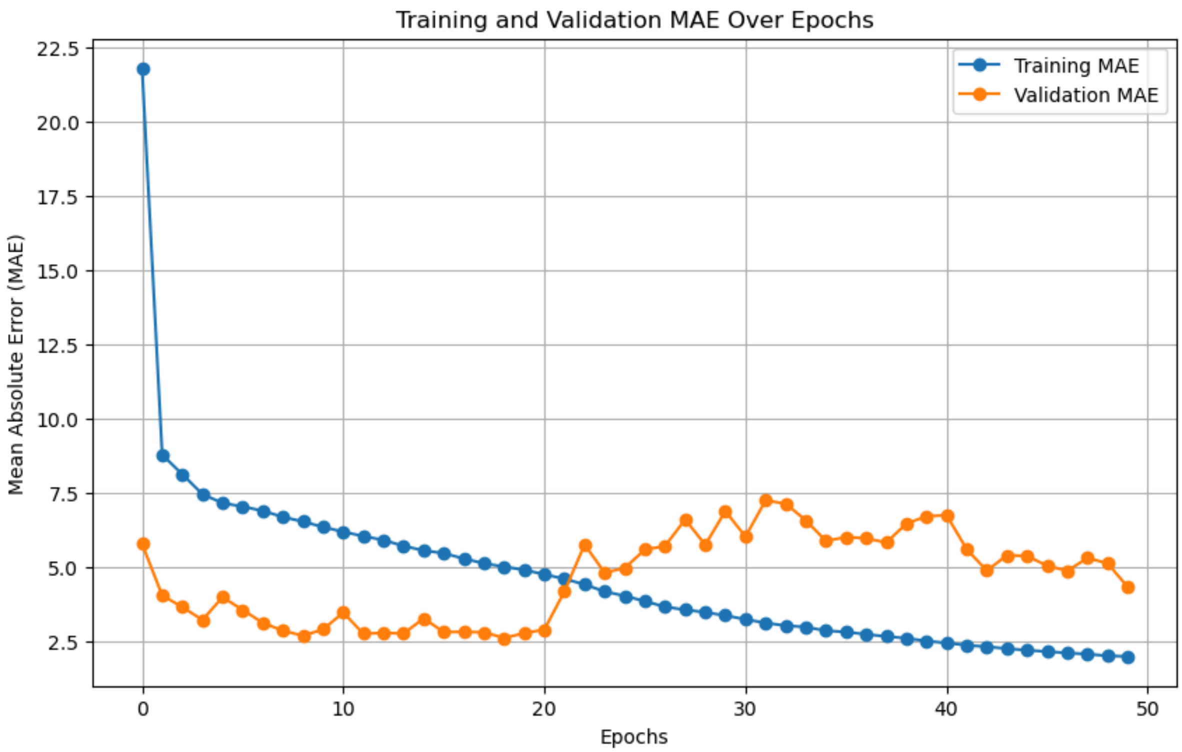


In [25]:

```
# Plot Training and Validation Loss
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Training Loss', marker='o')
plt.plot(history.history['val_loss'], label='Validation Loss', marker='o')
plt.title("Training and Validation Loss Over Epochs")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)
plt.show()
```

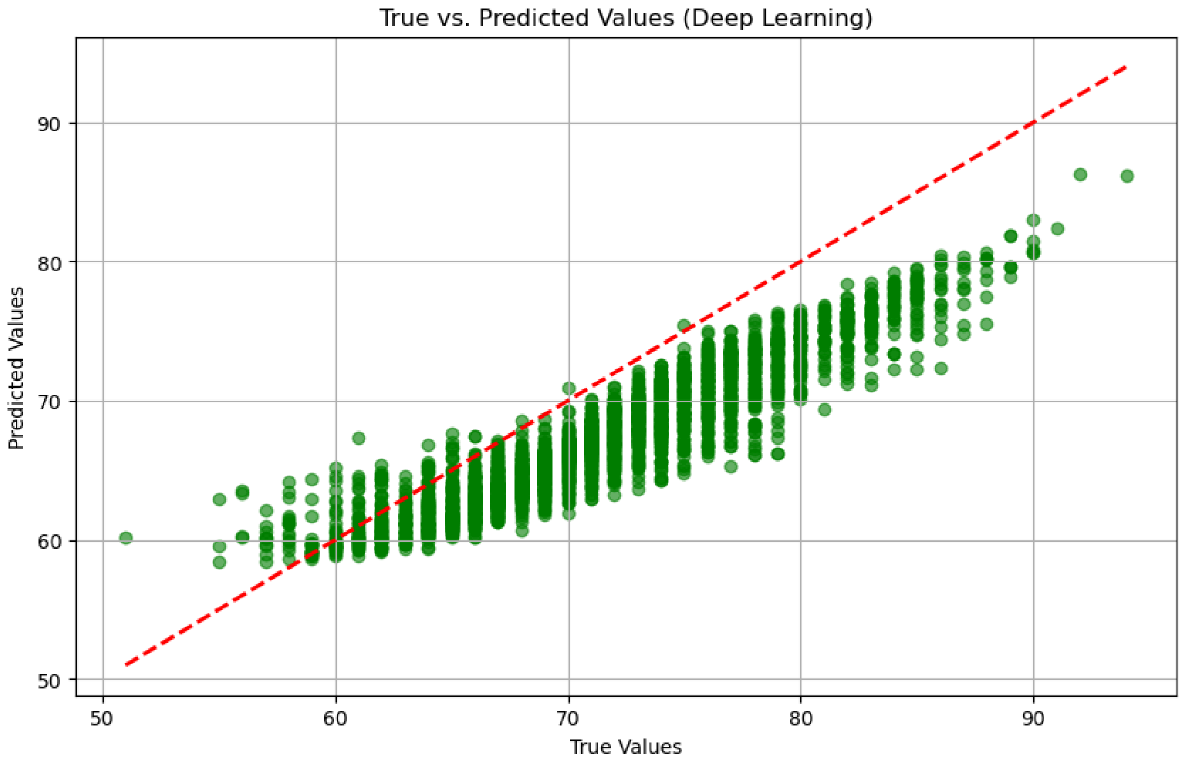


In [26]:
```
# Plot Training and Validation MAE
plt.figure(figsize=(10, 6))
plt.plot(history.history['mae'], label='Training MAE', marker='o')
plt.plot(history.history['val_mae'], label='Validation MAE', marker='o')
plt.title("Training and Validation MAE Over Epochs")
plt.xlabel("Epochs")
plt.ylabel("Mean Absolute Error (MAE)")
plt.legend()
plt.grid(True)
plt.show()
```



```
In [27]: # True vs. Predicted Values Scatter Plot
y_dl_pred = dl_model.predict(X_test).flatten()

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_dl_pred, alpha=0.6, color='green')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='rec
plt.title("True vs. Predicted Values (Deep Learning)")
plt.xlabel("True Values")
plt.ylabel("Predicted Values")
plt.grid(True)
plt.show()
```

```
111/111 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```



In [31]:
```
from tensorflow.keras.callbacks import TensorBoard
import datetime

# TensorBoard setup
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)

# Model training with TensorBoard callback
history = dl_model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=50,
    batch_size=32,
    verbose=1,
    callbacks=[tensorboard_callback]
)
```

```
Epoch 1/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 7.0558 - mae: 1.9912 - val_l
oss: 21.8121 - val_mae: 4.2511
Epoch 2/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 6.8658 - mae: 1.9418 - val_l
oss: 19.9683 - val_mae: 4.0161
Epoch 3/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 6.4287 - mae: 1.8858 - val_l
oss: 24.0675 - val_mae: 4.4614
Epoch 4/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 6.2698 - mae: 1.8626 - val_l
oss: 18.1380 - val_mae: 3.8127
Epoch 5/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.9702 - mae: 1.8067 - val_l
oss: 16.5177 - val_mae: 3.6317
Epoch 6/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.7988 - mae: 1.7949 - val_l
oss: 17.0902 - val_mae: 3.7087
Epoch 7/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.9902 - mae: 1.8382 - val_l
oss: 17.6852 - val_mae: 3.8046
Epoch 8/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.6135 - mae: 1.7671 - val_l
oss: 17.5584 - val_mae: 3.7491
Epoch 9/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.5161 - mae: 1.7332 - val_l
oss: 17.3618 - val_mae: 3.7516
Epoch 10/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.5013 - mae: 1.7441 - val_l
oss: 15.7107 - val_mae: 3.4837
Epoch 11/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.1521 - mae: 1.6864 - val_l
oss: 17.4275 - val_mae: 3.7205
Epoch 12/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.2380 - mae: 1.6906 - val_l
oss: 14.4171 - val_mae: 3.3757
Epoch 13/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.0566 - mae: 1.6641 - val_l
oss: 14.5546 - val_mae: 3.4122
Epoch 14/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 5.0519 - mae: 1.6630 - val_l
oss: 17.3743 - val_mae: 3.7308
Epoch 15/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.8877 - mae: 1.6400 - val_l
oss: 15.8087 - val_mae: 3.5656
Epoch 16/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.8021 - mae: 1.6180 - val_l
oss: 19.2609 - val_mae: 3.9886
Epoch 17/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.7583 - mae: 1.6109 - val_l
oss: 17.3000 - val_mae: 3.7671
Epoch 18/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.7592 - mae: 1.6062 - val_l
oss: 13.7713 - val_mae: 3.2916
Epoch 19/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.6130 - mae: 1.5963 - val_l
oss: 14.1414 - val_mae: 3.3210
```

```
Epoch 20/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.5580 - mae: 1.5847 - val_l
oss: 15.9713 - val_mae: 3.5325
Epoch 21/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.4756 - mae: 1.5552 - val_l
oss: 15.5634 - val_mae: 3.4834
Epoch 22/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.4192 - mae: 1.5609 - val_l
oss: 13.6270 - val_mae: 3.2891
Epoch 23/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.3514 - mae: 1.5314 - val_l
oss: 16.1044 - val_mae: 3.6045
Epoch 24/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.6010 - mae: 1.5634 - val_l
oss: 14.0938 - val_mae: 3.3067
Epoch 25/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.3295 - mae: 1.5378 - val_l
oss: 11.5513 - val_mae: 2.9376
Epoch 26/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.2732 - mae: 1.5205 - val_l
oss: 11.7098 - val_mae: 3.0031
Epoch 27/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.1749 - mae: 1.4920 - val_l
oss: 13.4141 - val_mae: 3.2194
Epoch 28/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.2798 - mae: 1.5127 - val_l
oss: 13.0060 - val_mae: 3.2156
Epoch 29/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.2907 - mae: 1.5295 - val_l
oss: 13.7055 - val_mae: 3.2363
Epoch 30/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.1720 - mae: 1.4965 - val_l
oss: 12.5416 - val_mae: 3.1174
Epoch 31/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.0325 - mae: 1.4821 - val_l
oss: 10.5649 - val_mae: 2.9054
Epoch 32/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.9882 - mae: 1.4671 - val_l
oss: 11.7065 - val_mae: 2.9864
Epoch 33/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.0279 - mae: 1.4866 - val_l
oss: 12.2868 - val_mae: 3.1479
Epoch 34/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.0007 - mae: 1.4737 - val_l
oss: 12.2819 - val_mae: 3.1129
Epoch 35/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.1805 - mae: 1.5045 - val_l
oss: 10.3223 - val_mae: 2.8056
Epoch 36/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.9037 - mae: 1.4542 - val_l
oss: 13.2681 - val_mae: 3.1773
Epoch 37/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.9669 - mae: 1.4705 - val_l
oss: 14.6843 - val_mae: 3.4455
Epoch 38/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.0510 - mae: 1.4617 - val_l
oss: 10.0642 - val_mae: 2.7988
```

```
Epoch 39/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.7931 - mae: 1.4254 - val_l
oss: 11.2842 - val_mae: 2.9759
Epoch 40/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.7446 - mae: 1.4186 - val_l
oss: 14.4052 - val_mae: 3.3422
Epoch 41/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.9305 - mae: 1.4456 - val_l
oss: 12.7288 - val_mae: 3.1599
Epoch 42/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.5872 - mae: 1.3996 - val_l
oss: 12.1998 - val_mae: 3.0624
Epoch 43/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 4.1423 - mae: 1.4687 - val_l
oss: 13.0107 - val_mae: 3.1933
Epoch 44/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.9138 - mae: 1.4234 - val_l
oss: 10.9625 - val_mae: 2.8861
Epoch 45/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.7473 - mae: 1.4266 - val_l
oss: 13.0028 - val_mae: 3.1772
Epoch 46/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.9098 - mae: 1.4357 - val_l
oss: 10.2345 - val_mae: 2.7965
Epoch 47/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.6212 - mae: 1.3955 - val_l
oss: 10.8252 - val_mae: 2.8630
Epoch 48/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.8664 - mae: 1.4526 - val_l
oss: 10.8232 - val_mae: 2.8617
Epoch 49/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.8166 - mae: 1.4182 - val_l
oss: 12.5546 - val_mae: 3.1572
Epoch 50/50
354/354 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 3.7474 - mae: 1.4122 - val_l
oss: 12.2438 - val_mae: 3.0675
```

In [ ]:
```
from sklearn.model_selection import GridSearchCV

# Define parameter grid
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5, 10]
}

# Grid search
grid_search = GridSearchCV(estimator=RandomForestRegressor(random_state=42), pa
grid_search.fit(X_train, y_train)

# Best parameters
print("Best Parameters:", grid_search.best_params_)
print("Best R² Score:", grid_search.best_score_)
```

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
[CV] END .max_depth=10, min_samples_split=2, n_estimators=50; total time=
3.2s
[CV] END .max_depth=10, min_samples_split=2, n_estimators=50; total time=
3.2s
[CV] END .max_depth=10, min_samples_split=2, n_estimators=50; total time=
3.2s
[CV] END max_depth=10, min_samples_split=2, n_estimators=100; total time=
6.9s
[CV] END max_depth=10, min_samples_split=2, n_estimators=100; total time=
7.3s
[CV] END max_depth=10, min_samples_split=2, n_estimators=100; total time=
7.6s
[CV] END max_depth=10, min_samples_split=2, n_estimators=150; total time=  1
1.1s
[CV] END max_depth=10, min_samples_split=2, n_estimators=150; total time=  1
1.0s
[CV] END max_depth=10, min_samples_split=2, n_estimators=150; total time=  1
1.6s
[CV] END .max_depth=10, min_samples_split=5, n_estimators=50; total time=
3.7s
[CV] END .max_depth=10, min_samples_split=5, n_estimators=50; total time=
3.7s
[CV] END .max_depth=10, min_samples_split=5, n_estimators=50; total time=
3.7s
[CV] END max_depth=10, min_samples_split=5, n_estimators=100; total time=
7.8s
[CV] END max_depth=10, min_samples_split=5, n_estimators=100; total time=
7.8s
[CV] END max_depth=10, min_samples_split=5, n_estimators=100; total time=
6.6s
[CV] END max_depth=10, min_samples_split=5, n_estimators=150; total time=  1
1.3s
[CV] END max_depth=10, min_samples_split=5, n_estimators=150; total time=  1
0.4s
[CV] END max_depth=10, min_samples_split=5, n_estimators=150; total time=  1
0.0s
[CV] END max_depth=10, min_samples_split=10, n_estimators=50; total time=
3.1s
[CV] END max_depth=10, min_samples_split=10, n_estimators=50; total time=
3.4s
[CV] END max_depth=10, min_samples_split=10, n_estimators=50; total time=
3.0s
[CV] END max_depth=10, min_samples_split=10, n_estimators=100; total time=
6.2s
[CV] END max_depth=10, min_samples_split=10, n_estimators=100; total time=
7.5s
[CV] END max_depth=10, min_samples_split=10, n_estimators=100; total time=
6.7s
[CV] END max_depth=10, min_samples_split=10, n_estimators=150; total time=  1
0.2s
[CV] END max_depth=10, min_samples_split=10, n_estimators=150; total time=  1
0.3s
[CV] END max_depth=10, min_samples_split=10, n_estimators=150; total time=
9.7s
[CV] END .max_depth=20, min_samples_split=2, n_estimators=50; total time=
6.2s
```

```
[CV] END .max_depth=20, min_samples_split=2, n_estimators=50; total time=
6.1s
[CV] END .max_depth=20, min_samples_split=2, n_estimators=50; total time=
6.0s
[CV] END max_depth=20, min_samples_split=2, n_estimators=100; total time=  1
2.5s
[CV] END max_depth=20, min_samples_split=2, n_estimators=100; total time=  1
2.8s
[CV] END max_depth=20, min_samples_split=2, n_estimators=100; total time=  1
2.5s
[CV] END max_depth=20, min_samples_split=2, n_estimators=150; total time=  1
8.6s
[CV] END max_depth=20, min_samples_split=2, n_estimators=150; total time=  2
0.9s
```

Conclusions

In [ ]:
```
EDA revealed strong correlations between potential and attributes like value_eu
Machine Learning: Random Forest achieved score of ~0.85, indicating good predic
Deep Learning: The neural network effectively modeled player potential with a n
```

References

- Academic (if any)
- Online (if any)

In [ ]:
```
```

Credits

- If you use and/or adapt your code from existing projects, you must provide links and acknowldge the authors.

> *This code is based on .... (if any)*

In [ ]:
```
```

In [ ]:
```
# End of Project
```