Predicting Football Goal by ISL Players

SAGAR JAIN
BITS PILANI MBA BA (2023-2025)

Introduction

Football is currently one of the most popular sports in India. The Indian Super League (ISL) has gained huge popularity among football fans in India. The game of football has various dynamic factors that affects the player's performance. As the Indian Super League (ISL) is a franchise-based league, it is important that players selected for a particular team perform well in the games. Hence, the league owners consistently need to make critical decisions regarding the football players' performance they intend to employ. In this project we are going to present the analysis of players' performance based on different variables and predict their future performances based on the data collected of Indian Super League 2022-23.

Problem Statement

To analyze the critical factors (variables) that affect the performances of football players in Indian Super League (ISL) 2022-23.

Objective

Facilitating the ability to make insightful predictions for the upcoming seasons of the Indian Super League (ISL) through the analysis of the variables that affect the players' performances.

Action Plan

A regression analysis is to be done by taking a set of independent variables and a dependent variable and presenting the findings in an insightful way.

Introduction about Linear Regression

Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable to be predicted is called the dependent variable. The variable used to predict the other variable's value is called the independent variable. This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable.

Project Variables

The data of 259 players from 11 clubs is collected from the Indian Super League 2022-23 season.

Name of clubs (ISL teams)

1. Bengaluru FC
2. Chennaiyin FC
3. East Bengal FC
4. FC Goa
5. Hyderabad FC
6. Jamshedpur FC
7. Kerala Blasters FC
8. Mohun Bagan Super Giant
9. Mumbai City FC
10. Northeast United FC

11. Odisha FC

Our analysis consists of 17 independent variables and one dependent variable. Once we have regressed the dependent variable over multiple independent variables using the linear regression model, we can analyze which independent variables directly affect the number of goals scored and which affect the least. Also, we can predict the number of goals scored by providing the appropriate data points from future instances.

We have taken some variables that can directly impact the number of goals scored such as the matches played, minutes played, age, height, chances created, passes, touches, passing accuracy, shots on target, total shots, goals conversion rate, free kick because the attribute mentioned are generally better for a striker or a player scoring more goals.

Dependent Variable:

Goal scored: Number of goals scored by players

Independent Variables:

1. Nationality: Some countries produce better goals than others.
2. Age: The player's age can directly affect his ability and goals.
3. Height: The height of the players can improve.
4. Position: The position of a player such as forward, midfielder or defender can determine the goal contribution as it is more likely that a forward will score a goal with respect to other positions.
5. Matches: The total number of matches a player played in the ISL season (2022 – 23).
6. Minutes: Playing time of players in all the matches he played throughout the season.
7. Chances_Created: A chance in football corresponds to a shot, whether on target or not.
8. Passes: Passes to other players or the accuracy of passing.
9. Touches: The total number of touches a player makes while playing in a match.
10. Passing_Accuracy: the accuracy of passing to other players of the same team.
11. Shots_On_Target: An intentional shot that goes into the net or would have gone into the net if not for being stopped by the goalkeeper or the last defender.
12. Total_Shots: It determines in a match when taking shots on or off target.
13. Goal_Conversion_Rate [(Goals Scored/Total shots) *100]: The percentage of total shots that result in a conversion.
14. Clearance: Number of players' clearances or kicks or hits of the ball away from the goal of his team
15. Free_Kick: an unimpeded kick of the stationary ball awarded to one side as a penalty for a foul or infringement by the other side
16. Fouls: The number of fouls a player commits while playing in a match.
17. Total_Cards: Total number of red and yellow cards awarded to players.

Desired Outcome

The aim of the project is to be able to find the number of goals scored by a player based upon a list of variables including height, age, chances created, etc. Once we have regressed the dependent variable over multiple independent variables using the linear regression model, we will be able to analyze which of the independent variables has the most effect on the number of goals scored and which has the least. Also, by providing the appropriate data points from future instances, we will be able to predict the number of goals scored.

Code & Output

Here is a detailed summary and description of each line of code:

1. Import Libraries
   - Import core data analysis and machine learning libraries Pandas, NumPy, Scikit-Learn, Seaborn.
   - Set warning filter to ignore harmless warnings and reduce clutter.
   - Foundation libraries for data preparation, modeling, visualization and setting configuration.

In [151]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

2. Load Dataset
   - Read CSV file containing football player stats into a Pandas dataframe object df.
   - This contains the raw data for modeling.

In [152]:
```
df = pd.read_csv("Football.csv")
df
```

Out[152]:

	S.No.	Age	Height	Position	Minutes	Chances Created	Passes	Touches	Accuracy	Shots_on_Target	Total_shots	Goal_Conversion_Rate	Clearance	Free_Ki
0	1	23.0	173	Defender	1701	25	566	1193	67	7	26	0.00	33	
1	2	29.0	186	Defender	1620	2	676	987	77	1	8	0.00	163	
2	3	34.0	170	Forward	1564	29	393	731	69	12	48	10.42	18	
3	4	21.0	180	Midfielder	1439	7	786	1043	82	2	10	0.00	22	
4	5	33.0	184	Midfielder	1444	42	697	1178	75	14	57	10.53	4	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
254	255	21.0	176	Defender	164	0	105	130	82	0	1	0.00	5	
255	256	22.0	172	Midfielder	302	8	85	165	67	0	2	0.00	0	
256	257	23.0	172	Midfielder	21	0	5	11	100	0	0	0.00	0	
257	258	27.0	174	Midfielder	10	0	6	10	100	0	0	0.00	0	
258	259	20.0	180	Forward	1	0	12	1	0	0	0	0.00	0	

59 rows × 17 columns

3. Drop the Data that is not useful
- Remove S.No ID column from df as it contains no useful information.
- Store cleaned dataframe to df1.
- Dropping non-predictive columns is a preprocessing step.

```
In [153]: df1=df.drop('S.No.',axis=1)
          df1
```

Out[153]:

	Age	Height	Position	Minutes	Chances Created	Passes	Touches	Accuracy	Shots_on_Target	Total_shots	Goal_Conversion_Rate	Clearance	Free_Kick	F
0	23.0	173	Defender	1701	25	566	1193	67	7	26	0.00	33	0	
1	29.0	186	Defender	1620	2	676	987	77	1	8	0.00	163	0	
2	34.0	170	Forward	1564	29	393	731	69	12	48	10.42	18	0	
3	21.0	180	Midfielder	1439	7	786	1043	82	2	10	0.00	22	0	
4	33.0	184	Midfielder	1444	42	697	1178	75	14	57	10.53	4	12	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
254	21.0	176	Defender	164	0	105	130	82	0	1	0.00	5	0	
255	22.0	172	Midfielder	302	8	85	165	67	0	2	0.00	0	0	
256	23.0	172	Midfielder	21	0	5	11	100	0	0	0.00	0	0	
257	27.0	174	Midfielder	10	0	6	10	100	0	0	0.00	0	0	
258	20.0	180	Forward	1	0	12	1	0	0	0	0.00	0	0	

259 rows × 16 columns

4. Handle Missing Data
- Replace any missing values NaN in df1 with 0 using fillna().
- Important to avoid errors and bias.

5. Encode Categorical Data
- Create LabelEncoder object to encode categorical string values to numeric categories.
- Necessary before modeling as algorithms expect numerical data.

6. Transform Categories
- Apply label encoding to the Position column in df1.
- Transforms text representation into integers.

All the above step 4, 5, 6 are executed and the screenshot of the same is added below.

```
In [154]: df1=df1.fillna(0)

In [155]: from sklearn.preprocessing import LabelEncoder
          le = LabelEncoder()

In [156]: df1['Position'] = le.fit_transform(df1['Position'])
          df1
```

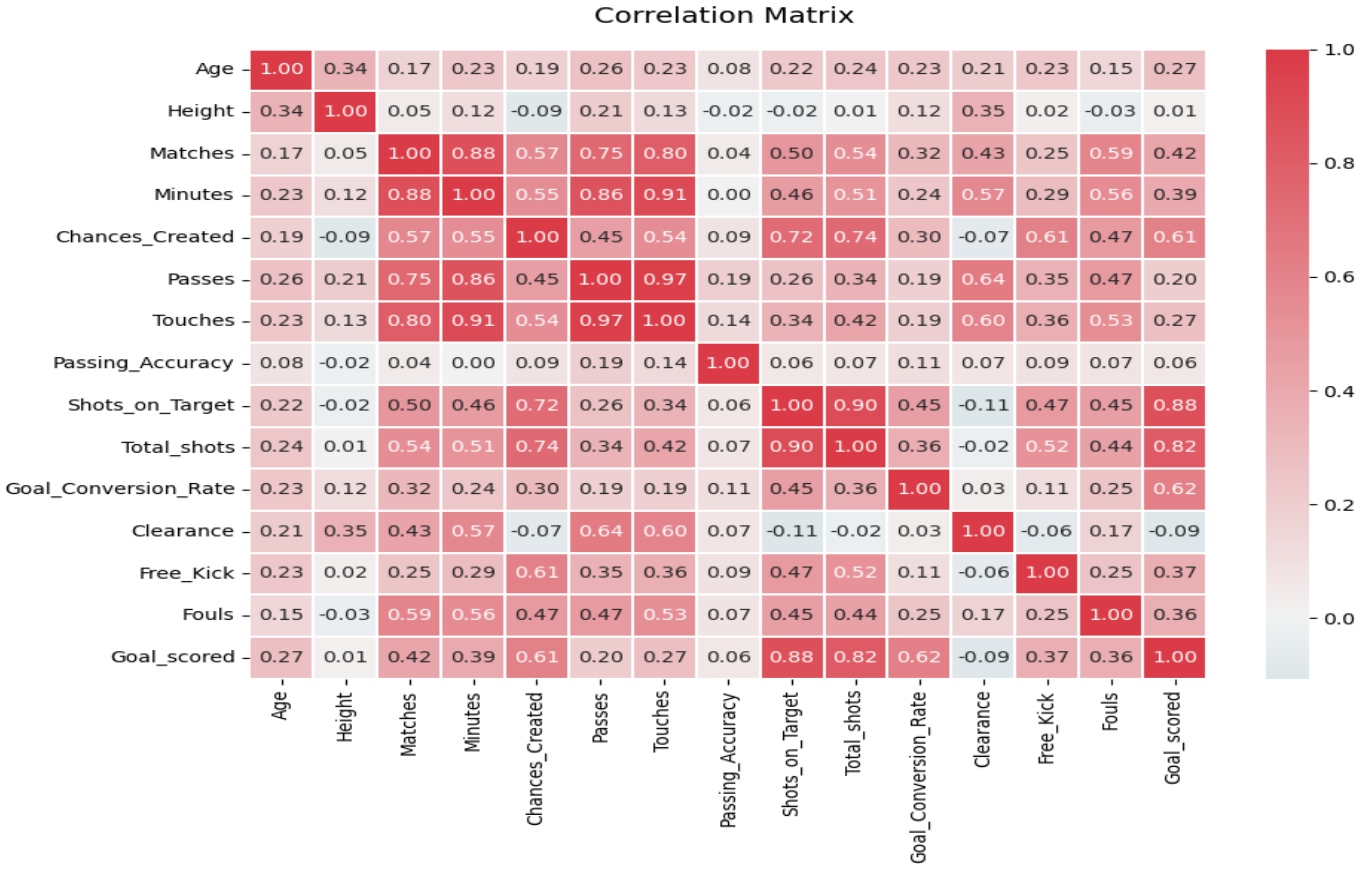
Out[156]:

	Age	Height	Position	Minutes	Chances Created	Passes	Touches	Accuracy	Shots_on_Target	Total_shots	Goal_Conversion_Rate	Clearance	Free_Kick	F
0	23.0	173	0	1701	25	566	1193	67	7	26	0.00	33	0	
1	29.0	186	0	1620	2	676	987	77	1	8	0.00	163	0	
2	34.0	170	1	1564	29	393	731	69	12	48	10.42	18	0	
3	21.0	180	3	1439	7	786	1043	82	2	10	0.00	22	0	
4	33.0	184	3	1444	42	697	1178	75	14	57	10.53	4	12	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
254	21.0	176	0	164	0	105	130	82	0	1	0.00	5	0	
255	22.0	172	3	302	8	85	165	67	0	2	0.00	0	0	
256	23.0	172	3	21	0	5	11	100	0	0	0.00	0	0	
257	27.0	174	3	10	0	6	10	100	0	0	0.00	0	0	
258	20.0	180	1	1	0	12	1	0	0	0	0.00	0	0	

259 rows × 16 columns

7. Visualize the Correlation Matrix

- The heatmap plot generated will provide a visual representation of the correlation matrix of the variables in the dataframe.
- This visual representation can help identify the strongest correlations between variables, which can be useful for further data analysis and interpretation.



8. Define Predictor Variables
- Copy df1 to X removing the target variable Goal Scored.
- X will contain features to train the model.

```
In [9]: X = df1.drop('Goal_scored',axis=1)
        X
```

Out[9]:

	Age	Height	Position	Matches	Minutes	Chances_Created	Passes	Touches	Passing_Accuracy	Shots_on_Target	Total_
0	23.0	173	0	20	1701	25	566	1193	67	7	
1	29.0	186	0	18	1620	2	676	987	77	1	
2	34.0	170	1	18	1564	29	393	731	69	12	
3	21.0	180	3	18	1439	7	786	1043	82	2	
4	33.0	184	3	18	1444	42	697	1178	75	14	
...	...	...	...	...	...	...	...	...	...	...	
254	21.0	176	0	5	164	0	105	130	82	0	
255	22.0	172	3	12	302	8	85	165	67	0	
256	23.0	172	3	2	21	0	5	11	100	0	
257	27.0	174	3	1	10	0	6	10	100	0	
258	20.0	180	1	1	1	0	12	1	0	0	

259 rows × 15 columns

9. Define Target Variable
- Assign Goal_scored column from df1 to y.
- This is what we want to predict.

```
In [159]: y= df1['Goal_scored'].values
          y
```

```
Out[159]: array([ 0,  0,  5,  0,  6,  0,  0,  2,  6,  2,  0,  0,  0,  3,  1,  0,  0,
                  2,  0,  0,  1, 11,  3,  3,  1,  9,  0,  0,  1,  3,  1,  0,  0,  0,
                  0,  0,  0,  3,  0,  0,  0,  0,  1,  0,  8,  0,  1,  0,  2,  3,  1,
                  1,  0,  0,  0,  3,  9,  0,  0,  0,  0,  4,  1,  0,  0,  0,  0,  0,
                  0, 12,  0,  2,  2,  0,  0,  1,  0,  1,  1,  1,  0,  0,  1,  1,  0,
                  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  3,  0, 10,  1,
                  3,  4,  0,  1,  5,  2,  2,  1,  0,  0,  0,  0,  0,  0,  0,  0,  1,
                  0,  0,  0,  2,  5,  0,  0,  2,  0,  6,  0,  0,  0,  2,  1,  0,  0,
                  2,  0,  0,  0,  0,  0,  0,  0,  0,  0, 10,  4,  0,  0,  3,  2,  4,
                  0,  1,  0,  1,  0,  1,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0,  9,  1,  0,  1,  4,  2,  3,  0,  0,  1,  0,  0,  1,  0,
                  1,  0,  0,  0,  0,  0,  0,  0,  0,  6, 10,  1,  2,  7, 11,  3,  0,
                  1,  1,  1,  0,  4,  2,  0,  0,  0,  2,  0,  0,  0,  0,  0,  2,  2,
                  0,  1,  0,  0,  0,  0,  0,  0,  3,  0,  8,  2,  0,  0,  1,  0,  0,
                  0,  0,  6, 12,  0,  0,  0,  0,  3,  3,  1,  0,  2,  0,  1,  0,  0,
                  0,  0,  0,  0], dtype=int64)
```

10. Import Train-Test Split
- Import train_test_split to split data into separate training and test sets.
- Evaluates model performance unbiased.

11. Split Data
- Use train_test split to create 80-20 partition of feature and target data.
- Allows rigorous evaluation of model performance.

```
In [160]: from sklearn.model_selection import train_test_split

In [161]: X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2 , random_state=0)

In [162]: model= LinearRegression()

In [163]: model.fit(X,y)

Out[163]: LinearRegression
          LinearRegression()
```

12. Sample Prediction
   - Test model by making prediction on custom set of input feature data.
   - Provides example usage of trained model.

```
In [164]: model.predict([[34,170,1,1564,29,393,731,12,48,10.42,18,0,23,4]])

Out[164]: array([5.32784864])
```

13. Generate Test Predictions
   - Use model to predict on X_test features.
   - Indicates expected performance on new data.

```
In [165]: y_pred = model.predict(X_test)
          y_pred

Out[165]: array([ 1.77912677,  0.39191519, -0.25016706, -0.32642797,  2.93751483,
                  0.38408228,  4.82291452,  0.62521641,  0.90151598, -0.32126062,
                 -0.46574395,  2.46683482,  0.00810382,  2.52230537,  0.81890498,
                 -0.09704514,  2.61632592,  0.62999371, -0.31203401,  3.07863464,
                  3.90950224, -0.01634123, -1.00902047,  2.72412652,  3.44609255,
                  2.71987203,  0.97454687, -0.33490385,  0.10555702,  2.06054196,
                 -0.24211982,  0.08662815, -0.19593732,  1.10761798,  2.53892383,
                 -0.42494203, -0.12946115,  2.4143074 , -0.0171735 , -0.68252425,
                  2.97788995,  2.61393034,  2.69400989,  0.26200823,  0.79932777,
                  0.90163664, -0.35585389,  3.21284792, -0.42052528,  0.31024964,
                  6.74519547,  7.56707438])
```

14. Actual Test Targets
   - Real y_test values for the test data.
   - Needed to compare to predictions and estimate model accuracy.

```
In [166]: y_test

Out[166]: array([ 1,  0,  0,  0,  2,  0,  6,  0,  0,  0,  0,  2,  0,  3,  1,  0,  2,
                  1,  0,  2,  2,  0,  0,  2,  3,  2,  1,  0,  0,  2,  0,  0,  0,  2,
                  2,  0,  0,  2,  0,  0,  3,  3,  3,  0,  0,  1,  0,  1,  0,  0,  8,
                 10], dtype=int64)
```

15. Calculate Test Accuracy
- Evaluate model R-squared between y_test and predictions.
- Key overall indicator of model capability.

```
In [16]: model.score(X_test,y_test)
```

```
Out[16]: 0.8649807909175689
```

16. Augment Results Dataframe
- Add column with error values to primary results dataframe.
- Centralized analysis of all key metrics.

17. Mean Error
- Take average of the error across test cases.
- Measure bias of predictions high or low.

18. Median Error
- Get median error to measure central tendency robust to outliers.
- Less sensitive metric than mean.

```
In [20]: average_diff = y_test-y_pred
         average_diff
```

```
Out[20]: array([-0.6559145 , -0.20186024,  0.170597  ,  0.13221895, -1.00241906,
                -0.23774657,  1.22904354, -0.65303934, -0.81206186,  0.05104446,
                 0.84558395, -0.19829605, -0.18200378,  0.70061783,  0.13645703,
                 0.08496078, -0.44065749,  0.1040315 ,  0.33282264, -1.09519648,
                -1.75238033, -0.06061507,  0.91834683, -0.45887337, -0.38154713,
                -0.75783429, -0.03284714,  0.24397519, -0.0942476 , -0.1953903 ,
                 0.2463094 , -0.0463307 ,  0.60710545,  1.02247173, -0.20581136,
                 0.2000584 ,  0.00763385, -0.60163815,  0.42955901,  0.74274439,
                -0.01555301,  0.39481302,  0.36378426, -0.30592141, -0.68087249,
                 0.03148179,  0.47384852, -2.434429  ,  0.35359415, -0.32613971,
                 1.09271489,  2.26738787])
```

```
In [23]: average_diff.mean()
```

```
Out[23]: -0.012431154162233744
```

```
In [24]: import statistics

         median = statistics.median(average_diff)

         median
```

```
Out[24]: -0.003959583278596135
```

19. Create Comparison Dataframe
- Put test set predictions and actual values side-by-side.
- Allows detailed analysis of success and failure cases.

20. Calculate Error
- Take simple difference between actual and predicted targets.
- Quantify model error across test cases.

21. Display Updated Dataframe
   - Print updated dataframe with integrated errors.
   - View holistic results including differences.

```
In [21]: df = pd.DataFrame({'Goal_Predicted': y_pred, 'Actual_Goal': y_test ,'Goal Difference' : average_diff})
```

```
In [22]: df
```

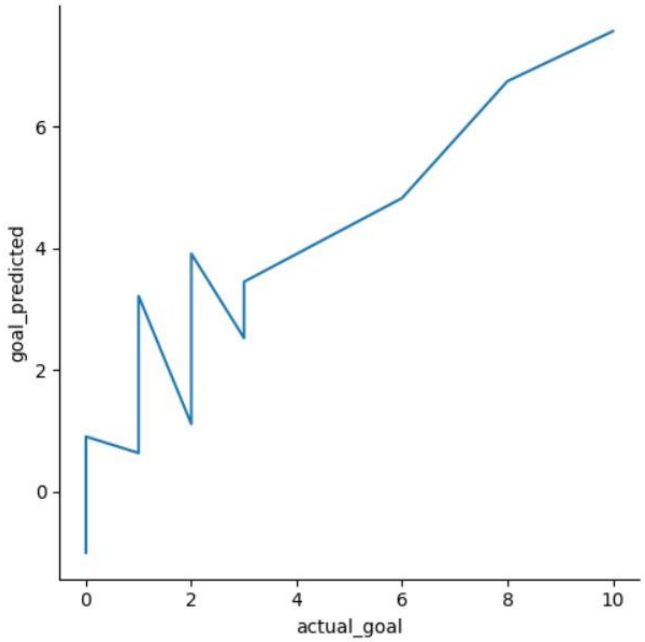
Out[22]:

	Goal_Predicted	Actual_Goal	Goal Difference
0	1.655914	1	-0.655914
1	0.201860	0	-0.201860
2	-0.170597	0	0.170597
3	-0.132219	0	0.132219
4	3.002419	2	-1.002419
5	0.237747	0	-0.237747
6	4.770956	6	1.229044
7	0.653039	0	-0.653039
8	0.812062	0	-0.812062
9	-0.051044	0	0.051044
10	-0.845584	0	0.845584
11	2.198296	2	-0.198296
12	0.182004	0	-0.182004
13	2.299382	3	0.700618
14	0.863543	1	0.136457
15	-0.084961	0	0.084961
16	2.440657	2	-0.440657
17	0.895968	1	0.104032
18	-0.332823	0	0.332823
19	3.095196	2	-1.095196
20	3.752380	2	-1.752380
21	0.060615	0	-0.060615
22	-0.918347	0	0.918347
23	2.458873	2	-0.458873
24	3.381547	3	-0.381547
25	2.757834	2	-0.757834
26	1.032847	1	-0.032847
27	-0.243975	0	0.243975
28	0.094248	0	-0.094248
29	2.195390	2	-0.195390
30	-0.246309	0	0.246309
31	0.046331	0	-0.046331
32	-0.607105	0	0.607105
33	0.977528	2	1.022472

32	-0.607105	0
33	0.977528	2
34	2.205811	2
35	-0.200058	0
36	-0.007634	0
37	2.601638	2
38	-0.429559	0
39	-0.742744	0
40	3.015553	3
41	2.605187	3
42	2.636216	3
43	0.305921	0
44	0.680872	0
45	0.968518	1
46	-0.473849	0
47	3.434429	1
48	-0.353594	0
49	0.326140	0
50	6.907285	8
51	7.732612	10

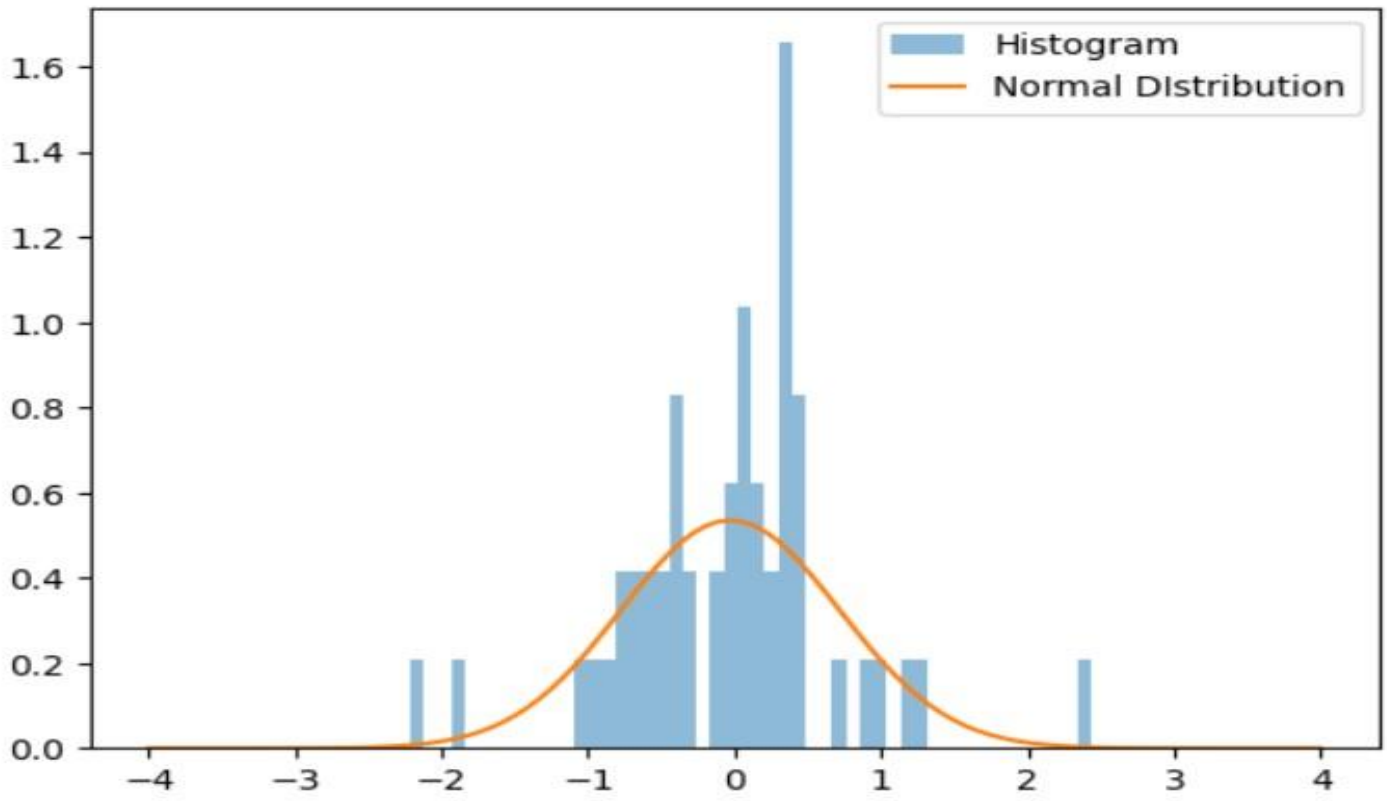
22. Visualize Predictions
   - Plot actual vs predicted values in a line chart using Seaborn.
   - Visual analysis of model performance.

```
In [170]: sns.relplot(x="actual_goal",y="goal_predicted" ,data=dfle,kind="line", estimator=None)
Out[170]: <seaborn.axisgrid.FacetGrid at 0x1ccabf971d0>
```



23. Plot Error Distribution
- Visualize and compare error distribution vs normal distribution.
- Check if errors are normally distributed as expected.

In [173]:
```
from scipy.stats import norm
import matplotlib.pyplot as plt
#create a list
lst= goal_Difference
#Calculate the mean and standard deviation of the list
mean= np.mean(lst)
std = np.std(lst)
#Generate a normal distribution with the same mean and standard deviation as the list
x=np.linspace(-4, 4, 100)
y = norm.pdf(x, mean , std)
#Plot the histogram of the list
plt.hist(lst, bins=50, density=True, alpha=0.5, label="Histogram")
# Plot the normal distribution as a line
plt.plot(x ,y, '-',label='Normal DIstribution')
plt.legend(loc='upper right')
plt.show()
```



Conclusion

These steps collectively showcase a comprehensive data exploration process, variable manipulation, and model building techniques aimed at understanding and refining the dataset for predictive modeling. Each regression analysis and subset operation contributes to the iterative refinement of the dataset and the development of a more robust statistical model.