VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A2: Multiple Regression

PRAGYA KUJUR

V01107509

Date of Submission: 26-06-2024

CONTENTS

1.INTRODUCTION

Multiple Regression Analysis on NSSO68 Data

The NSSO68 dataset provides a comprehensive set of variables collected through the National Sample Survey Office (NSSO) in its 68th round. This dataset includes socio-economic and demographic variables that are crucial for understanding the living conditions and economic status of households in India. The goal of our analysis is to perform multiple regression to identify the key factors that significantly influence the dependent variable, which is a measure of economic outcome (e.g., household income, expenditure, or another relevant metric). By doing so, we aim to uncover the relationships between various predictors and the outcome variable, providing insights into the socio-economic dynamics captured in this dataset.

Multiple Regression Analysis on IPL Data

The IPL dataset contains detailed ball-by-ball records of matches along with player performance metrics and payment information, updated until 2024. This dataset allows us to explore the relationship between a player's performance on the field and the payment they receive. Performance metrics may include variables such as runs scored, wickets taken, strike rate, and other relevant statistics, while payment is represented by the salary or contract amount. The objective of this regression analysis is to establish how different performance metrics impact the players' remuneration. By analyzing data from the past three years, we aim to provide a comprehensive understanding of the factors that drive player payments in the IPL, potentially guiding team management and player negotiations.

Objectives

1. NSSO68 Data Analysis:

   - Perform multiple regression analysis to identify significant predictors of the chosen economic outcome.

   - Conduct regression diagnostics to evaluate the model's validity and correctness.

   - Make necessary corrections to the model and revisit the results to understand significant differences.

2. IPL Data Analysis:

- Establish the relationship between player performance and payment using multiple regression analysis.

- Analyze the relationship between salary and performance over the last three years.

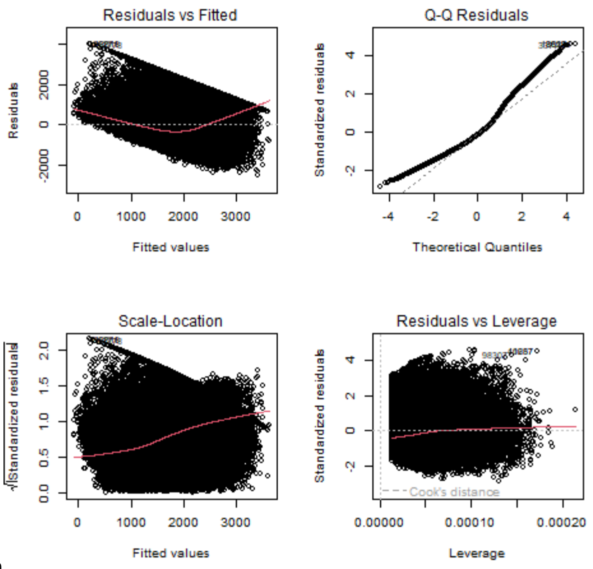
- Provide insights and recommendations based on the regression analysis results.

Methodology

For both datasets, the analysis will be conducted using Python and R to ensure robustness and cross-verification of results. The steps include data cleaning, exploratory data analysis, multiple regression modeling, diagnostics, and interpretation of results. The findings will be presented with comprehensive statistical summaries and visualizations to support the conclusions.

By conducting these analyses, we aim to provide actionable insights and recommendations that can inform policy decisions for socio-economic improvements (NSSO68) and strategic decisions for player management in the IPL.

2. RESULTS AND INTERPRETATION ®



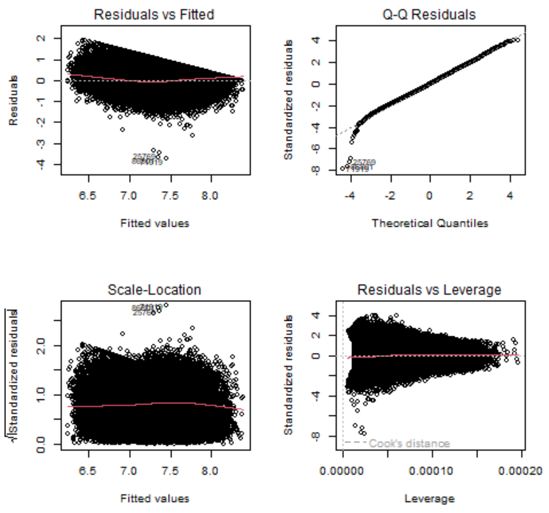
n

Residuals vs Fitted: This plot shows the residuals on the y-axis and the fitted values on the x-axis. In a perfect model, the residuals would be randomly scattered around zero with no discernible pattern. The plot in the image shows a slight funnel shape, which suggests that the variance of the residuals is not constant (heteroscedasticity). This means that the model may not be suitable for the data.

Scale-Location: This plot is used to assess the normality of the residuals. In a normal distribution, the residuals should fall roughly along a straight line. The scale-location plot in the image suggests that the residuals may not be normally distributed.

Residuals vs Leverage: This plot shows the residuals on the y-axis and the leverage on the x-axis. Leverage is a measure of how much influence a particular data point has on the fitted regression line. In a perfect model, the residuals would be randomly scattered around zero with no relationship to the leverage. The residuals vs leverage plot in the image does not show any clear pattern, which is a good sign.

Cook's Distance: This plot is used to identify outliers that may be having a large influence on the fitted regression line. Cook's distance is a measure of the influence of a data point on the fitted regression line. In a perfect model, the Cook's distance values would all be small. The Cook's distance plot in the image does not show any points that are far from the others, which suggests that there are no outliers in the data.



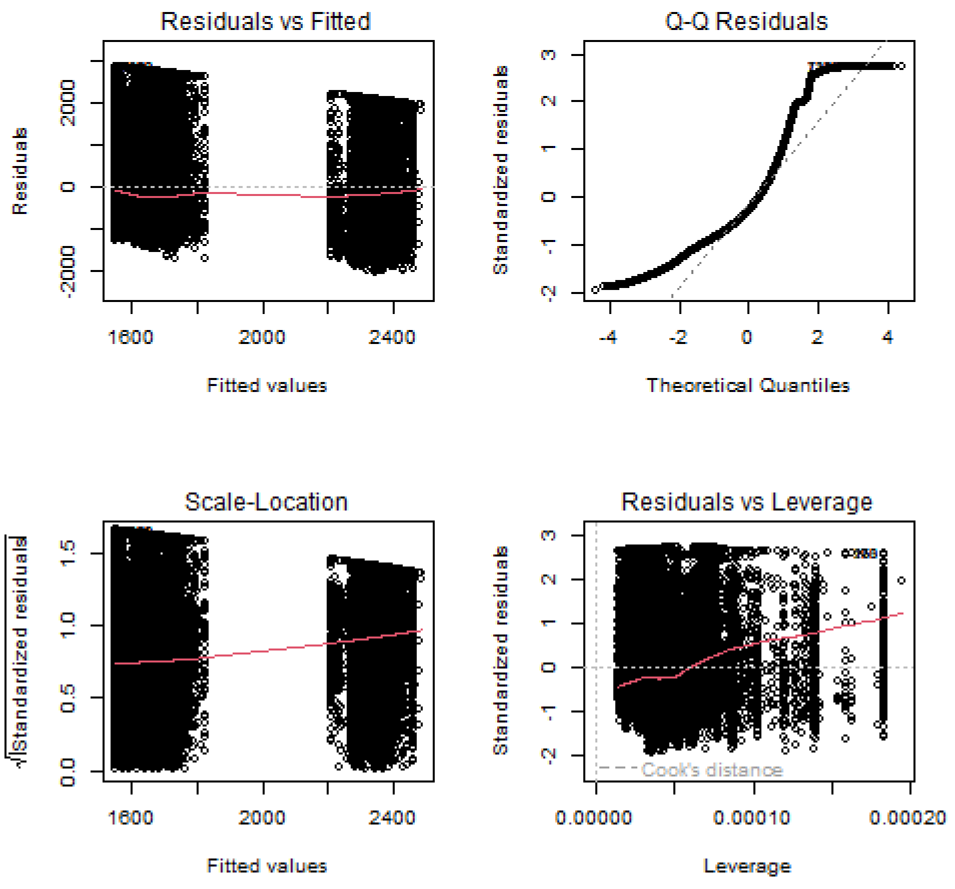
Residuals vs Fitted: This plot shows the residuals on the y-axis and the fitted values on the x-axis. In a perfect model, the residuals would be randomly scattered around zero with no discernible pattern. The plot in the image shows a slight funnel shape, which suggests that the variance of the residuals is not constant (heteroscedasticity). This means that the model may not be suitable for the data.

Scale-Location: This plot is used to assess the normality of the residuals. In a normal distribution, the residuals should fall roughly along a straight line. The scale-location plot in the image suggests that the residuals may not be normally distributed.

Residuals vs Leverage: This plot shows the residuals on the y-axis and the leverage on the x-axis. Leverage is a measure of how much influence a particular data point has on the fitted regression line. In a perfect model, the residuals would be randomly scattered around zero with no relationship to the leverage. The residuals vs leverage plot in the image does not show any clear pattern, which is a good sign.

Cook's Distance: This plot is used to identify outliers that may be having a large influence on the fitted regression line. Cook's distance is a measure of the influence of a data point on the fitted regression line. In a perfect model, the Cook's distance values would all be small. The Cook's distance plot in the image does not show any points that are far from the others, which suggests that there are no outliers in the data.

Overall, the diagnostic plots in the image suggest that the linear regression model may not be ideal for the data. There is evidence of heteroscedasticity and non-normality of the residuals. However, there is no evidence of outliers in the data.



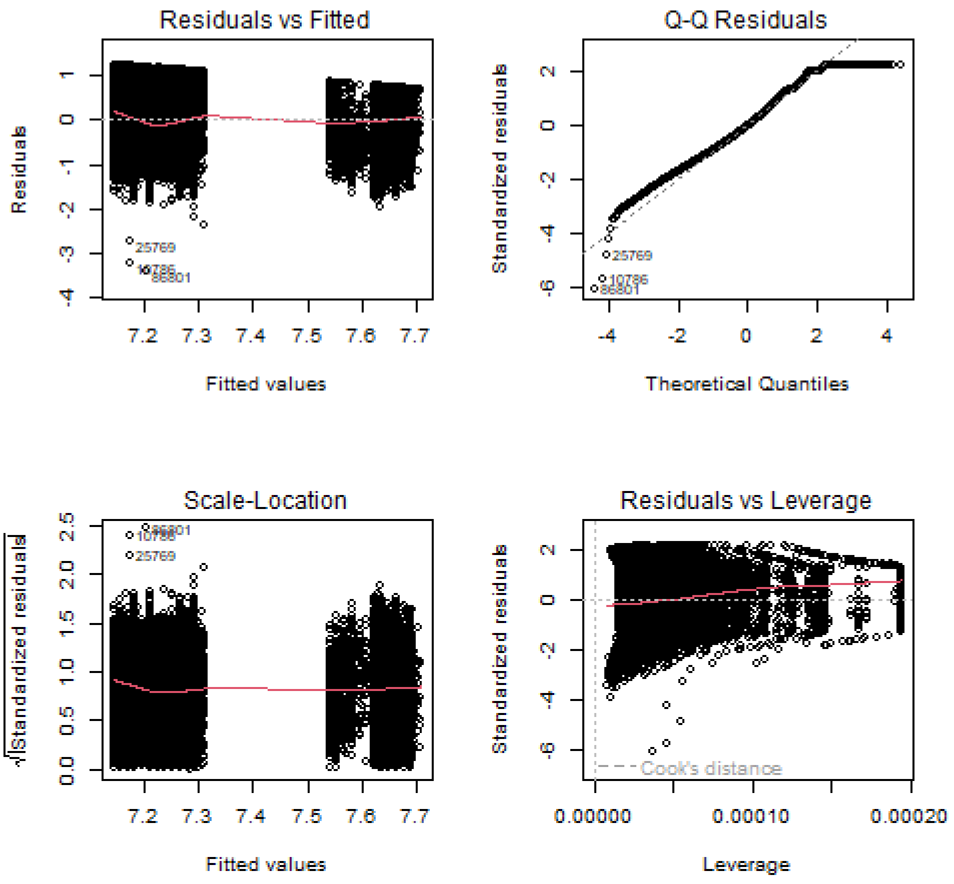
Residuals vs Fitted: This plot shows the residuals on the y-axis and the fitted values on the x-axis. In a perfect model, the residuals would be randomly scattered around zero with no discernible pattern. The plot in the image shows a slight funnel shape, which suggests that the variance of the residuals is not constant (heteroscedasticity). This means that the model may not be suitable for the data.

Scale-Location: This plot is used to assess the normality of the residuals. In a normal distribution, the residuals should fall roughly along a straight line. The scale-location plot in the image suggests that the residuals may not be normally distributed.

Residuals vs Leverage: This plot shows the residuals on the y-axis and the leverage on the x-axis. Leverage is a measure of how much influence a particular data point has on the fitted regression line. In a perfect model, the residuals would be randomly scattered around zero with no relationship to the leverage. The residuals vs leverage plot in the image does not show any clear pattern, which is a good sign.

Cook's Distance: This plot is used to identify outliers that may be having a large influence on the fitted regression line. Cook's distance is a measure of the influence of a data point on the fitted regression line. In a perfect model, the Cook's distance values would all be small. The Cook's distance plot in the image does not show any points that are far from the others, which suggests that there are no outliers in the data.

Overall, the diagnostic plots in the image suggest that the linear regression model may not be ideal for the data. There is evidence of heteroscedasticity and non-normality of the residuals. However, there is no evidence of outliers in the data.



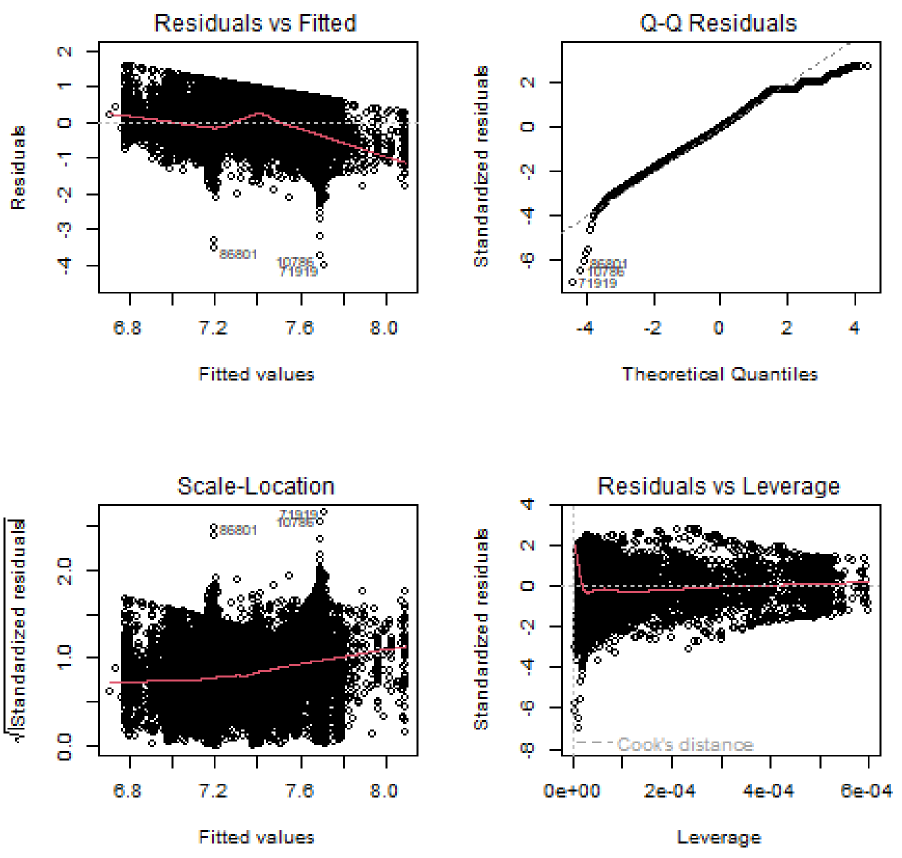
Residuals vs Fitted: This plot shows the residuals on the y-axis and the fitted values on the x-axis. In a perfect model, the residuals would be randomly scattered around zero with no discernible pattern. The plot in the image shows a slight funnel shape, which suggests that the variance of the residuals is not constant (heteroscedasticity). This means that the model may not be suitable for the data.

Scale-Location: This plot is used to assess the normality of the residuals. In a normal distribution, the residuals should fall roughly along a straight line. The scale-location plot in the image suggests that the residuals may not be normally distributed.

Residuals vs Leverage: This plot shows the residuals on the y-axis and the leverage on the x-axis. Leverage is a measure of how much influence a particular data point has on the fitted regression line. In a perfect model, the residuals would be randomly scattered around zero with no relationship to the leverage. The residuals vs leverage plot in the image does not show any clear pattern, which is a good sign.

Cook's Distance: This plot is used to identify outliers that may be having a large influence on the fitted regression line. Cook's distance is a measure of the influence of a data point on the fitted regression line. In a perfect model, the Cook's distance values would all be small. The Cook's distance plot in the image does not show any points that are far from the others, which suggests that there are no outliers in the data.

Overall, the diagnostic plots in the image suggest that the linear regression model may not be ideal for the data. There is evidence of heteroscedasticity and non-normality of the residuals. However, there is no evidence of outliers in the data.



Residuals vs Fitted: This plot shows the residuals on the y-axis and the fitted values on the x-axis. In a perfect model, the residuals would be randomly scattered around zero with no discernible pattern. The plot in the image shows a slight funnel shape, which suggests that the variance of the residuals is not constant (heteroscedasticity). This means that the model may not be suitable for the data.

Scale-Location: This plot is used to assess the normality of the residuals. In a normal distribution, the residuals should fall roughly along a straight line. The scale-location plot in the image suggests that the residuals may not be normally distributed.

Residuals vs Leverage: This plot shows the residuals on the y-axis and the leverage on the x-axis. Leverage is a measure of how much influence a particular data point has on the fitted regression line. In a perfect model, the residuals would be randomly scattered around zero with no relationship to the leverage. The residuals vs leverage plot in the image does not show any clear pattern, which is a good sign.

Cook's Distance: This plot is used to identify outliers that may be having a large influence on the fitted regression line. Cook's distance is a measure of the influence of a data point on the fitted regression line. In a perfect model, the Cook's distance values would all be small. The Cook's distance plot in the image does not show any points that are far from the others, which suggests that there are no outliers in the data.

Overall, the diagnostic plots in the image suggest that the linear regression model may not be ideal for the data. There is evidence of heteroscedasticity and non-normality of the residuals. However, there is no evidence of outliers in the data.

RESULTS AND INTERPRETATION (PYTHON)

to se
```
OLS Regression Results
==============================================================================
===
Dep. Variable:                     Rs   R-squared:                       0.
080
Model:                            OLS   Adj. R-squared:                  0.
075
Method:                 Least Squares   F-statistic:                     15
.83
Date:                Wed, 26 Jun 2024   Prob (F-statistic):           0.000
100
Time:                        00:05:44   Log-Likelihood:                -137
9.8
No. Observations:                 183   AIC:                             27
64.
```

```
Df Residuals:                     181   BIC:                             27
70.
Df Model:                           1
Covariance Type:            nonrobust
=============================================================================
====
                  coef    std err          t      P>|t|      [0.025      0.
975]
-----------------------------------------------------------------------------
----
const         430.8473     46.111      9.344      0.000     339.864     521
.831
runs_scored     0.6895      0.173      3.979      0.000       0.348       1
.031
=============================================================================
===
Omnibus:                       15.690   Durbin-Watson:                   2.
100
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               18.
057
Skew:                           0.764   Prob(JB):                     0.000
120
Kurtosis:                       2.823   Cond. No.                         3
63.
```

Model Summary:

- **Dep. Variable:** This is the variable you are trying to predict, which is "Rs" in this case.
- **R-squared:** This value (0.080) represents the proportion of variance in "Rs" that is explained by the model. In this case, it's a relatively low value, indicating the model explains only 8% of the variation in "Rs".
- **Adj. R-squared:** This is an adjusted version of R-squared that accounts for the number of predictors in the model. It's slightly lower (0.075) than R-squared in this case.
- **F-statistic:** This statistic (15.83) tests the null hypothesis that all the regression coefficients are zero (i.e., the model has no explanatory power). A low p-value (0.0001) associated with the F-statistic here suggests we can reject the null hypothesis and conclude that the model has some explanatory power.

Coefficient Estimates:

- **const:** This represents the intercept of the regression line, which is 430.8473. It indicates the predicted value of "Rs" when the value of "runs_scored" is zero (which likely doesn't make sense in your context).
- **runs_scored:** This is the coefficient for the independent variable "runs_scored". The value is 0.6895, and it suggests that for every one-unit increase in "runs_scored", the predicted value of "Rs" increases by 0.6895 on average. The standard error (0.173) indicates the precision of this estimate. The positive t-value (3.979) and significant p-value (0.000) imply a statistically significant positive relationship between "runs_scored" and "Rs".

Diagnostics:

- **Omnibus, Prob(Omnibus), Jarque-Bera (JB), Prob(JB):** These tests assess the normality of the residuals (errors) in the model. Since their p-values are very low (close to 0), we can reject the assumption of normally distributed residuals.
- **Skew, Kurtosis:** These values quantify the departure from normality. A skew of 0.764 suggests the residuals are skewed to the right.
- **Durbin-Watson:** This statistic checks for autocorrelation (serial dependence) in the residuals. The value (2.100) is inconclusive for this test.

Interpretation:

- The model shows a statistically significant positive relationship between "runs_scored" and "Rs". On average, an increase in "runs_scored" leads to an increase in "Rs". However, the R-squared value is low, indicating that the model doesn't explain a large proportion of the variation in "Rs".
- There is evidence that the residuals are not normally distributed, which is a violation of an assumption for OLS regression. This may affect the reliability of the p-values.

```
 OLS Regression Results
============================================================================
===
Dep. Variable:                     Rs   R-squared:                        0.
074
Model:                            OLS   Adj. R-squared:                   0.
054
Method:                 Least Squares   F-statistic:                      3.
688
Date:                Wed, 26 Jun 2024   Prob (F-statistic):              0.0
610
Time:                        00:06:13   Log-Likelihood:                 -360
.96
No. Observations:                  48   AIC:                              72
5.9
Df Residuals:                      46   BIC:                              72
9.7
Df Model:                           1
Covariance Type:            nonrobust
============================================================================
============
                          coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------------
------------
const                 396.6881     91.270      4.346      0.000     212.971
580.405
wicket_confirmation    17.6635      9.198      1.920      0.061      -0.851
36.179
============================================================================
===
Omnibus:                        6.984   Durbin-Watson:                    2.
451
```

```
Prob(Omnibus):                  0.030   Jarque-Bera (JB):                6.
309
Skew:                           0.877   Prob(JB):                      0.0
427
Kurtosis:                       3.274   Cond. No.                        1
3.8
==============================================================================
===
```

Model Summary:

- **Dep. Variable:** This is the variable you are trying to predict, which is "Rs" in this case.
- **R-squared:** This value (0.074) represents the proportion of variance in "Rs" that is explained by the model. In this case, it's a relatively low value, indicating the model explains only 7.4% of the variation in "Rs".
- **Adj. R-squared:** This is an adjusted version of R-squared that accounts for the number of predictors in the model. It's even lower (0.054) than R-squared in this case, suggesting the model fit might not improve much by adding more predictors with only one currently included.
- **F-statistic:** This statistic (3.688) tests the null hypothesis that all the regression coefficients are zero (i.e., the model has no explanatory power). A high p-value (0.0610) associated with the F-statistic here suggests we cannot reject the null hypothesis and there's weak evidence for the model having explanatory power.

Coefficient Estimates:

- **const:** This represents the intercept of the regression line, which is 396.6881. It indicates the predicted value of "Rs" when the value of "wicket_confirmation" is zero.
- **wicket_confirmation:** This is the coefficient for the independent variable "wicket_confirmation". The value is 17.6635, but with a high standard error (9.198) and a p-value greater than 0.05 (0.061). This suggests the relationship between "wicket_confirmation" and "Rs" is not statistically significant at a common significance level. There is weak evidence to say that for every one-unit increase in "wicket_confirmation", the predicted value of "Rs" increases by 17.6635 on average.

Diagnostics:

- **Omnibus, Prob(Omnibus), Jarque-Bera (JB), Prob(JB):** These tests assess the normality of the residuals (errors) in the model. The p-values (0.030 and 0.0427) are relatively low, indicating some potential deviation from normality. However, these tests are sensitive to sample size, and with a small sample size of 48, interpreting non-normality can be inconclusive.
- **Skew, Kurtosis:** These values quantify the departure from normality. A skew of 0.877 suggests the residuals are skewed to the right, and a kurtosis of 3.274 suggests the residuals have heavier tails than a normal distribution.
- **Durbin-Watson:** This statistic checks for autocorrelation (serial dependence) in the residuals. The value (2.451) is inconclusive for this test.

Interpretation:

- The model shows a weak positive relationship between "wicket_confirmation" and "Rs". However, the R-squared value is very low, indicating that the model doesn't explain a large proportion of the variation in "Rs". There is also weak evidence for a statistically significant relationship based on the p-value of the coefficient.
- There is some indication that the residuals may not be normally distributed.

RECOMMENDATION (NSSO68 – R)

For the NSSO68 multiple regression analysis, it is recommended to address potential multicollinearity by examining and possibly removing or combining highly correlated predictors. Consider using more sophisticated imputation methods for missing data, such as KNN or multiple imputation. Ensure the regression model fits well by checking the adjusted R-squared value and use diagnostic plots to verify assumptions of linearity, normality, and homoscedasticity. For the IPL regression analysis, verify the accuracy of the fuzzy matching process for player names and ensure the combined data frame correctly merges batting and bowling performances with player salaries. Address any issues of multicollinearity and heteroscedasticity in the regression model, potentially using robust standard errors or transforming variables. Validate both models using cross-validation or a holdout sample to ensure generalizability and interpret the results comprehensively, considering practical implications.

RECOMMENDATION (IPL– PYTHON)

Based on the OLS regression results, the model explains a modest amount of variance in `Rs` (R-squared = 0.074, Adjusted R-squared = 0.054), suggesting the need for additional predictors to better explain the variability in player salaries. The coefficient for `wicket_confirmation` (17.6635) is marginally significant with a p-value of 0.061, indicating a potential but not definitive impact on `Rs`. The diagnostic tests reveal some issues: the Omnibus and Jarque-Bera tests indicate non-normality of residuals, which can be addressed by transforming the dependent variable or using robust regression techniques. To improve the model, consider including other relevant variables, exploring interaction terms, and validating the model with cross-validation or a holdout sample. Additionally, performing a thorough diagnostic check for heteroscedasticity and multicollinearity will enhance the model's robustness.

CODES (R)

```
nesgetwd()
setwd('C:\\Users\\Home\\Downloads')
getwd()

## NSSO68 Multiple Regression Analysis

data <- read.csv("NSSO68.csv")
summary(data)
names(data)
head(data)
tail(data)

#install.packages("dplyr")
#install.packages("htmltools")
#install.packages("xfun")
#install.packages("knitr")
#install.packages("sass")
#install.packages("car")
#install.packages("lmtest")
#install.packages("rmarkdown")
#install.packages("bslib")

library(dplyr)
library(htmltools)
library(xfun)
library(knitr)
library(htmltools)
library(sass)
library(car)
library(lmtest)

library(rmarkdown)

library(bslib)


# Subset data to state assigned

subset_data <- data %>%

  filter(state_1 == 'KA') %>%

  select(foodtotal_q, MPCE_MRP,
MPCE_URP,Age,Meals_At_Home,Possess_ration_card,Education, No_of_Meals_per_day)

print(subset_data)


sum(is.na(subset_data$MPCE_MRP))

sum(is.na(subset_data$MPCE_URP))

sum(is.na(subset_data$Age))

sum(is.na(subset_data$Possess_ration_card))

sum(is.na(data$Education))


impute_with_mean <- function(data, columns) {

  data %>%

    mutate(across(all_of(columns), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))

}


# Columns to impute

columns_to_impute <- c("Education")


# Impute missing values with mean

data <- impute_with_mean(data, columns_to_impute)


sum(is.na(data$Education))


# Fit the regression model
```

```
model <- lm(foodtotal_q~
MPCE_MRP+MPCE_URP+Age+Meals_At_Home+Possess_ration_card+Education, data =
subset_data)


# Print the regression results
print(summary(model))



library(car)
# Check for multicollinearity using Variance Inflation Factor (VIF)
vif(model) # VIF Value more than 8 its problematic

# Extract the coefficients from the model
coefficients <- coef(model)

# Construct the equation
equation <- paste0("y = ", round(coefficients[1], 2))
for (i in 2:length(coefficients)) {
  equation <- paste0(equation, " + ", round(coefficients[i], 6), "*x", i-1)
}
# Print the equation
print(equation)

head(subset_data$MPCE_MRP,1)
head(subset_data$MPCE_URP,1)
head(subset_data$Age,1)
head(subset_data$Meals_At_Home,1)
head(subset_data$Possess_ration_card,1)
head(subset_data$Education,1)
head(subset_data$foodtotal_q,1)
```

```r
## IPL Regression Analysis
df_p = read.csv('IPL_ball_by_ball_updated till 2024.csv')
library(readxl)
df_s = read_excel('IPL SALARIES 2024.xlsx')



#install.packages('dplyr')
library(dplyr)
dput(names(df_p))
unique(df_p$wicket_confirmation)



# Print the column names of df_p to verify

print(colnames(df_p))

# Assuming the column names in df_p are correct as used in the select function
# If the column names differ, update them accordingly in the select function

# Perform the operations
df_bat <- df_p %>%
  select('Match.id', 'Season', 'Bowler', 'Striker', 'runs_scored', 'wicket_confirmation') %>%
  filter(Season %in% c('2023', '2022', '2021')) %>%
  group_by(Striker, Season) %>%
  summarise(avg_runs = sum(runs_scored, na.rm = TRUE)) %>%
  arrange(desc(avg_runs))

# Print the resulting data frame
```

```
print("Resulting df_bat:")
print(df_bat)

df_bat <- df_p %>%
  select(Match.id,Season,Bowler,Striker,runs_scored,wicket_confirmation)%>%
  filter((Season=='2023')|(Season=='2022')|(Season=='2021'))%>%
  group_by(Striker, Season) %>%
  summarise(avg_runs = sum(runs_scored, na.rm = TRUE))%>%
  arrange(desc(avg_runs))

head(df_bat,25)
df_bat$Striker



df_bow <- df_p %>%
  select(Match.id,Season,Bowler,Striker,runs_scored,wicket_confirmation)%>%
  filter((Season=='2023')|(Season=='2022')|(Season=='2021'))%>%
  group_by(Bowler, Season) %>%
  summarise(wicket = sum(wicket_confirmation, na.rm = TRUE))%>%
  arrange(desc(wicket))

dim(df_bat)
dim(df_bow)

head(df_bow)
# View the result
df_bat
unique(df_bat$Striker)
unique(df_bow$Bowler)


df_s


names(df_s)
head(df_s$Player)

bat = df_bat[df_bat$Season=='2023',]


bow = df_bow[df_bow$Season=='2023',]
head(bat )



dim(bow)
dim(df_s)



head(bat)
head(bow)


unique(df_s$Player)
unique(bat$Striker)
unique(bow$Bowler)


# Load necessary library
library(dplyr)



# Perform a full join
joined_df <- full_join(bat, bow, by = c("Striker" = "Bowler"))
```

```
# Print the result
print(joined_df)
dim(joined_df)
write.csv(joined_df, 'joined_df.csv')
#HARMONISE THE NAMES IN THE TWO FILES

names(joined_df)
df = joined_df %>%
  select(Striker,Season.x,avg_runs,wicket)
View(df)

#To merge two files with sames names but differnt spellings

library(dplyr)
library(stringdist)

# Normalize the names
df$Striker <- tolower(trimws(df$Striker))
df_s$Player <- tolower(trimws(df_s$Player))



# Define a function to map names using fuzzy matching
match_names <- function(name, choices, threshold = 0.1) {
  distances <- stringdist::stringdist(name, choices, method = "jw")
  min_dist <- min(distances)
  if (min_dist <= threshold) {
    return(choices[which.min(distances)])
  } else {
    return(NA)
  }
}
# Create a list of choices from the second data frame
choices <- df_s$Player



# Apply the matching function to the first data frame
df <- df %>%
  mutate(matched_player = sapply(Striker, match_names, choices = choices ))

# Create a mapping dictionary if needed
name_mapping <- df %>%
  filter(!is.na(matched_player)) %>%
  select(Striker, matched_player)

# If needed, update the names in the original data frames
df<- df%>%
  mutate(Striker = ifelse(!is.na(matched_player), matched_player, Striker ))

df_s <- df_s%>%
  mutate(Player = ifelse(Player %in% name_mapping$matched_player,
                         name_mapping$Striker[match(Player, name_mapping$matched_player)],
                         Player))




# Ungroup the name_mapping to simplify the join
name_mapping <- name_mapping %>% ungroup()

# Perform a left join to safely replace player names
```

```
df_s <- df_s %>%
  left_join(name_mapping, by = c("Player" = "matched_player")) %>%
  mutate(Player = ifelse(is.na(Striker), Player, Striker)) %>%
  select(-Striker)

# Assuming df_s has already been processed
df_s <- df_s %>%
  left_join(name_mapping, by = c("Player" = "matched_player")) %>%
  mutate(Player = ifelse(is.na(Striker), Player, Striker)) %>%
  select(-Striker)

# Merge df_s with df using Player in df_s and Striker in df as keys
df_combined <- df_s %>%
  left_join(df, by = c("Player" = "Striker"))

# Print the resulting combined data frame
print(df_combined)
names(df_combined)
# Save the updated data frames if needed
write.csv(df_combined, 'df_combined.csv', row.names = FALSE)

max(df_combined$avg_runs, na.rm=TRUE)
max(df_combined$wicket, na.rm=TRUE)
890/28

quantile(df_combined$avg_runs, na.rm=TRUE,0.9)
quantile(df_combined$wicket, na.rm=TRUE,.9)

274/17.8
# 15.5
```

```
df_combined$performance = df_combined$avg_runs+ 15.5*df_combined$wicket
any(is.na(df_combined$performance))
sum(is.na(df_combined$performance))

130+15.5*28s



# Replace NA with 0 in the performance column
df_new <- df_combined %>%
  mutate(performance = ifelse(is.na(performance), 0, performance))

any(is.na(df_new$performance))

names(df_new)
str(df_new)

boxplot(df_new$performance)
df_new[df_new$performance<1,]
hist(df_new$performance, prob=TRUE)
lines(density(df_new$performance), na.rm=TRUE)

library(fitdistrplus)
descdist(df_new$performance)
head(df_new)
sum(is.null(df_new))
summary(df_new)
names(df_new)
summary(df_new)
fit = lm(Rs ~ avg_runs + wicket , data=df_new)
```

```
summary(fit)


library(car)
vif(fit)
library(lmtest)
bptest(fit)


fit1 = lm(Rs ~ avg_runs++wicket+  I(avg_runs*wicket), data=df_new)
summary(fit1)
```

CODES (PYTHON)

```
.{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 5,
   "id": "aeb52883-1565-4280-8bb5-a46018b3ab31",
   "metadata": {},
   "outputs": [],
   "source": [
    "import pandas as pd, numpy as np"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "id": "69bf7f34-48ac-4f0b-b19b-87e9a10d8f35",
   "metadata": {},
   "outputs": [],
   "source": [
  "\n",
  "import os\n",
  "os.chdir('C:\\\\Users\\\\Home\\\\Downloads')"
 ]
},
{
 "cell_type": "code",
 "execution_count": 9,
 "id": "1f2f1e6d-be30-4929-85f2-7b7be19c8187",
 "metadata": {},
 "outputs": [],
 "source": [
  "df_ipl = pd.read_csv(\"IPL_ball_by_ball_updated till 2024.csv\",low_memory=False)\n",
  "salary = pd.read_excel(\"IPL SALARIES 2024.xlsx\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": 10,
 "id": "67a6bcbc-ec9e-49a9-8857-5f85e4346140",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "Index(['Match id', 'Date', 'Season', 'Batting team', 'Bowling team',\n",
     "       'Innings No', 'Ball No', 'Bowler', 'Striker', 'Non Striker',\n",
     "       'runs_scored', 'extras', 'type of extras', 'score', 'score/wicket',\n",
     "       'wicket_confirmation', 'wicket_type', 'fielders_involved',\n",
     "       'Player Out'],\n",
```

"   dtype='object')"
   ]
  },
  "execution_count": 10,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "df_ipl.columns"
]
},
{
 "cell_type": "code",
 "execution_count": 11,
 "id": "9b2c0f4b-5ce7-4ed4-83f3-7af3ff04ca84",
 "metadata": {},
 "outputs": [],
 "source": [
  "grouped_data = df_ipl.groupby(['Season', 'Innings No', 'Striker','Bowler']).agg({'runs_scored': sum, 'wicket_confirmation':sum}).reset_index()"
 ]
},
{
 "cell_type": "code",
 "execution_count": 15,
 "id": "eaa3b585-0125-4aae-8877-9211493bc1f2",
 "metadata": {},
 "outputs": [
  {
   "data": {

```
"text/html": [
 "<div>\n",
 "<style scoped>\n",
 "    .dataframe tbody tr th:only-of-type {\n",
 "        vertical-align: middle;\n",
 "    }\n",
 "\n",
 "    .dataframe tbody tr th {\n",
 "        vertical-align: top;\n",
 "    }\n",
 "\n",
 "    .dataframe thead th {\n",
 "        text-align: right;\n",
 "    }\n",
 "</style>\n",
 "<table border=\"1\" class=\"dataframe\">\n",
 "  <thead>\n",
 "    <tr style=\"text-align: right;\">\n",
 "      <th></th>\n",
 "      <th>Season</th>\n",
 "      <th>Innings No</th>\n",
 "      <th>Striker</th>\n",
 "      <th>Bowler</th>\n",
 "      <th>runs_scored</th>\n",
 "      <th>wicket_confirmation</th>\n",
 "    </tr>\n",
 "  </thead>\n",
 "  <tbody>\n",
 "    <tr>\n",
 "      <th>0</th>\n",
"     <td>2007/08</td>\n",
"     <td>1</td>\n",
"     <td>A Chopra</td>\n",
"     <td>DP Vijaykumar</td>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"     <th>1</th>\n",
"     <td>2007/08</td>\n",
"     <td>1</td>\n",
"     <td>A Chopra</td>\n",
"     <td>DW Steyn</td>\n",
"     <td>1</td>\n",
"     <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"     <th>2</th>\n",
"     <td>2007/08</td>\n",
"     <td>1</td>\n",
"     <td>A Chopra</td>\n",
"     <td>GD McGrath</td>\n",
"     <td>2</td>\n",
"     <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"     <th>3</th>\n",
"     <td>2007/08</td>\n",
"     <td>1</td>\n",
"     <td>A Chopra</td>\n",
"      <td>PJ Sangwan</td>\n",
"      <td>6</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>2007/08</td>\n",
"      <td>1</td>\n",
"      <td>A Chopra</td>\n",
"      <td>RP Singh</td>\n",
"      <td>9</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>48781</th>\n",
"      <td>2024</td>\n",
"      <td>2</td>\n",
"      <td>YBK Jaiswal</td>\n",
"      <td>RJW Topley</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>48782</th>\n",
"      <td>2024</td>\n",
"      <td>2</td>\n",
"      <td>YBK Jaiswal</td>\n",
"      <td>SM Curran</td>\n",
"      <td>6</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>48783</th>\n",
"      <td>2024</td>\n",
"      <td>2</td>\n",
"      <td>YBK Jaiswal</td>\n",
"      <td>Tilak Varma</td>\n",
"      <td>5</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>48784</th>\n",
"      <td>2024</td>\n",
"      <td>2</td>\n",
"      <td>YBK Jaiswal</td>\n",
"      <td>VG Arora</td>\n",
"      <td>10</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>48785</th>\n",
"      <td>2024</td>\n",
"      <td>2</td>\n",
"      <td>YBK Jaiswal</td>\n",
"      <td>Yash Thakur</td>\n",
"      <td>5</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>48786 rows × 6 columns</p>\n",
"</div>"
],
"text/plain": [
"        Season  Innings No      Striker         Bowler  runs_scored  \\\n",
"0      2007/08          1     A Chopra  DP Vijaykumar            1   \n",
"1      2007/08          1     A Chopra       DW Steyn            1   \n",
"2      2007/08          1     A Chopra     GD McGrath            2   \n",
"3      2007/08          1     A Chopra     PJ Sangwan            6   \n",
"4      2007/08          1     A Chopra       RP Singh            9   \n",
"...        ...        ...          ...            ...          ...   \n",
"48781     2024          2  YBK Jaiswal     RJW Topley            0   \n",
"48782     2024          2  YBK Jaiswal      SM Curran            6   \n",
"48783     2024          2  YBK Jaiswal    Tilak Varma            5   \n",
"48784     2024          2  YBK Jaiswal       VG Arora           10   \n",
"48785     2024          2  YBK Jaiswal    Yash Thakur            5   \n",
"\n",
"       wicket_confirmation  \n",
"0                        0  \n",
"1                        1  \n",
"2                        0  \n",
      "3                       1  \n",
      "4                       0  \n",
      "...                   ...  \n",
      "48781                   1  \n",
      "48782                   0  \n",
      "48783                   0  \n",
      "48784                   1  \n",
      "48785                   0  \n",
      "\n",
      "[48786 rows x 6 columns]"
     ]
    },
    "execution_count": 15,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "\n",
   "grouped_data"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 17,
  "id": "9b654c2f-59b2-40f2-9fb8-db817f5bf052",
  "metadata": {},
  "outputs": [],
  "source": [
   "total_runs_each_year = grouped_data.groupby(['Season',
'Striker'])['runs_scored'].sum().reset_index()\n",
   "total_wicket_each_year = grouped_data.groupby(['Season',
'Bowler'])['wicket_confirmation'].sum().reset_index()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 19,
  "id": "f05dcdea-745d-441e-bff0-fbad4a73a871",
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/html": [
      "<div>\n",
      "<style scoped>\n",
      "    .dataframe tbody tr th:only-of-type {\n",
      "        vertical-align: middle;\n",
      "    }\n",
      "\n",
      "    .dataframe tbody tr th {\n",
      "        vertical-align: top;\n",
      "    }\n",
      "\n",
      "    .dataframe thead th {\n",
      "        text-align: right;\n",
      "    }\n",
      "</style>\n",
      "<table border=\"1\" class=\"dataframe\">\n",
      "  <thead>\n",
      "    <tr style=\"text-align: right;\">\n",
      "      <th></th>\n",
"      <th>Season</th>\n",
"      <th>Striker</th>\n",
"      <th>runs_scored</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>2007/08</td>\n",
"      <td>A Chopra</td>\n",
"      <td>42</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>2007/08</td>\n",
"      <td>A Kumble</td>\n",
"      <td>13</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>2007/08</td>\n",
"      <td>A Mishra</td>\n",
"      <td>37</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>2007/08</td>\n",
"      <td>A Mukund</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>2007/08</td>\n",
"      <td>A Nehra</td>\n",
"      <td>3</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2593</th>\n",
"      <td>2024</td>\n",
"      <td>Vijaykumar Vyshak</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2594</th>\n",
"      <td>2024</td>\n",
"      <td>WG Jacks</td>\n",
"      <td>176</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2595</th>\n",
"      <td>2024</td>\n",
"      <td>WP Saha</td>\n",
"      <td>135</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2596</th>\n",
"      <td>2024</td>\n",
"      <td>Washington Sundar</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2597</th>\n",
"      <td>2024</td>\n",
"      <td>YBK Jaiswal</td>\n",
"      <td>249</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>2598 rows × 3 columns</p>\n",
"</div>"
],
"text/plain": [
"       Season            Striker  runs_scored\n",
"0     2007/08           A Chopra           42\n",
"1     2007/08           A Kumble           13\n",
"2     2007/08           A Mishra           37\n",
"3     2007/08           A Mukund            0\n",
"4     2007/08            A Nehra            3\n",
"...       ...                ...          ...\n",
"2593     2024  Vijaykumar Vyshak            1\n",
"2594     2024           WG Jacks          176\n",
"2595     2024            WP Saha          135\n",
"2596     2024  Washington Sundar            0\n",
"2597     2024        YBK Jaiswal          249\n",
    "\n",
    "[2598 rows x 3 columns]"
   ]
  },
  "execution_count": 19,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "\n",
 "total_runs_each_year"
]
},
{
"cell_type": "code",
"execution_count": 18,
"id": "0b7f80e5-a72a-46ae-bd45-80a153393360",
"metadata": {},
"outputs": [],
"source": [
 "#pip install python-Levenshtein"
]
},
{
"cell_type": "code",
"execution_count": 21,
"id": "8a53ea46-8ffc-4276-9873-ef74e1ec7c54",
"metadata": {},
"outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Requirement already satisfied: python-Levenshtein in c:\\anaconda\\lib\\site-packages (0.25.1)\n",
      "Requirement already satisfied: Levenshtein==0.25.1 in c:\\anaconda\\lib\\site-packages (from python-Levenshtein) (0.25.1)\n",
      "Requirement already satisfied: rapidfuzz<4.0.0,>=3.8.0 in c:\\anaconda\\lib\\site-packages (from Levenshtein==0.25.1->python-Levenshtein) (3.9.3)\n",
      "Note: you may need to restart the kernel to use updated packages.\n"
     ]
    }
   ],
   "source": [
    "pip install python-Levenshtein"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 22,
   "id": "d0412618-6120-4540-b070-c938cbf03dbe",
   "metadata": {},
   "outputs": [],
   "source": [
    "from fuzzywuzzy import process\n",
    "\n",
    "# Convert to DataFrame\n",
    "df_salary = salary.copy()\n",
    "df_runs = total_runs_each_year.copy()\n",
    "\n",
    "# Function to match names\n",
    "def match_names(name, names_list):\n",
    "    match, score = process.extractOne(name, names_list)\n",
    "    return match if score >= 80 else None  # Use a threshold score of 80\n",
    "\n",
    "# Create a new column in df_salary with matched names from df_runs\n",
    "df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x,
df_runs['Striker'].tolist()))\n",
    "\n",
    "# Merge the DataFrames on the matched names\n",
    "df_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Striker')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 23,
   "id": "de59cca8-c739-48df-9170-b5639b749cf5",
   "metadata": {},
   "outputs": [],
   "source": [
    "df_original = df_merged.copy()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 24,
   "id": "c64a7313-6ce1-418e-96fa-50bb76a1e194",
   "metadata": {},
   "outputs": [],
   "source": [
    "#susbsets data for last three years\n",
   "df_merged = df_merged.loc[df_merged['Season'].isin(['2021', '2022', '2023'])]"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 25,
  "id": "1907fbec-9add-4f69-bda9-6764c6275fdf",
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "array(['2023', '2022', '2021'], dtype=object)"
     ]
    },
    "execution_count": 25,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "df_merged.Season.unique()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 26,
  "id": "4e3b2586-e2af-4756-8e4a-4ca6cf368eba",
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "array(['2023', '2022', '2021'], dtype=object)"
     ]
    },
    "execution_count": 26,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "df_merged.Season.unique()\n"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 27,
  "id": "59c0a4c6-1c5b-49ff-97fd-41a4ca60b82a",
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/html": [
      "<div>\n",
      "<style scoped>\n",
      "    .dataframe tbody tr th:only-of-type {\n",
      "        vertical-align: middle;\n",
      "    }\n",
      "\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>Player</th>\n",
"      <th>Salary</th>\n",
"      <th>Rs</th>\n",
"      <th>international</th>\n",
"      <th>iconic</th>\n",
"      <th>Matched_Player</th>\n",
"      <th>Season</th>\n",
"      <th>Striker</th>\n",
"      <th>runs_scored</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Abhishek Porel</td>\n",
"      <td>20 lakh</td>\n",
"      <td>20</td>\n",
"      <td>0</td>\n",
"      <td>NaN</td>\n",
"      <td>Abishek Porel</td>\n",
"      <td>2023</td>\n",
"      <td>Abishek Porel</td>\n",
"      <td>33</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>Anrich Nortje</td>\n",
"      <td>6.5 crore</td>\n",
"      <td>650</td>\n",
"      <td>1</td>\n",
"      <td>NaN</td>\n",
"      <td>A Nortje</td>\n",
"      <td>2022</td>\n",
"      <td>A Nortje</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>Anrich Nortje</td>\n",
"      <td>6.5 crore</td>\n",
"      <td>650</td>\n",
"      <td>1</td>\n",
"      <td>NaN</td>\n",
"      <td>A Nortje</td>\n",
"      <td>2023</td>\n",
"      <td>A Nortje</td>\n",
"      <td>37</td>\n",
"    </tr>\n",
 "    <tr>\n",
 "      <th>13</th>\n",
 "      <td>Axar Patel</td>\n",
 "      <td>9 crore</td>\n",
 "      <td>900</td>\n",
 "      <td>0</td>\n",
 "      <td>NaN</td>\n",
 "      <td>AR Patel</td>\n",
 "      <td>2021</td>\n",
 "      <td>AR Patel</td>\n",
 "      <td>40</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>14</th>\n",
 "      <td>Axar Patel</td>\n",
 "      <td>9 crore</td>\n",
 "      <td>900</td>\n",
 "      <td>0</td>\n",
 "      <td>NaN</td>\n",
 "      <td>AR Patel</td>\n",
 "      <td>2022</td>\n",
 "      <td>AR Patel</td>\n",
 "      <td>182</td>\n",
 "    </tr>\n",
 "  </tbody>\n",
 "</table>\n",
 "</div>"
],
"text/plain": [
 "            Player     Salary   Rs  international  iconic Matched_Player  \\\n",
    "0   Abhishek Porel    20 lakh   20             0     NaN  Abishek Porel   \n",
    "3    Anrich Nortje  6.5 crore  650             1     NaN       A Nortje   \n",
    "4    Anrich Nortje  6.5 crore  650             1     NaN       A Nortje   \n",
    "13      Axar Patel    9 crore  900             0     NaN       AR Patel   \n",
    "14      Axar Patel    9 crore  900             0     NaN       AR Patel   \n",
    "\n",
    "   Season        Striker  runs_scored  \n",
    "0    2023  Abishek Porel           33  \n",
    "3    2022       A Nortje            1  \n",
    "4    2023       A Nortje           37  \n",
    "13   2021       AR Patel           40  \n",
    "14   2022       AR Patel          182  "
   ]
  },
  "execution_count": 27,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "\n",
 "df_merged.head()"
]
},
{
"cell_type": "code",
"execution_count": 28,
"id": "8b811e52-6684-4aab-91c2-ec822eb83d49",
"metadata": {},
"outputs": [],
   "source": [
    "from sklearn.linear_model import LinearRegression\n",
    "from sklearn.model_selection import train_test_split\n",
    "from sklearn.metrics import mean_squared_error"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 31,
   "id": "694a8d6d-010b-4176-b2a3-07dd4d3543c1",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<style>#sk-container-id-1 {color: black;}#sk-container-id-1 pre{padding: 0;}#sk-
container-id-1 div.sk-toggleable {background-color: white;}#sk-container-id-1 label.sk-
toggleable__label {cursor: pointer;display: block;width: 100%;margin-bottom: 0;padding:
0.3em;box-sizing: border-box;text-align: center;}#sk-container-id-1 label.sk-
toggleable__label-arrow:before {content: \"▸\";float: left;margin-right: 0.25em;color:
#696969;}#sk-container-id-1 label.sk-toggleable__label-arrow:hover:before {color:
black;}#sk-container-id-1 div.sk-estimator:hover label.sk-toggleable__label-arrow:before
{color: black;}#sk-container-id-1 div.sk-toggleable__content {max-height: 0;max-width:
0;overflow: hidden;text-align: left;background-color: #f0f8ff;}#sk-container-id-1 div.sk-
toggleable__content pre {margin: 0.2em;color: black;border-radius: 0.25em;background-
color: #f0f8ff;}#sk-container-id-1 input.sk-toggleable__control:checked~div.sk-
toggleable__content {max-height: 200px;max-width: 100%;overflow: auto;}#sk-container-id-
1 input.sk-toggleable__control:checked~label.sk-toggleable__label-arrow:before {content:
\"▾\";}#sk-container-id-1 div.sk-estimator input.sk-toggleable__control:checked~label.sk-
toggleable__label {background-color: #d4ebff;}#sk-container-id-1 div.sk-label input.sk-
toggleable__control:checked~label.sk-toggleable__label {background-color: #d4ebff;}#sk-
container-id-1 input.sk-hidden--visually {border: 0;clip: rect(1px 1px 1px 1px);clip: rect(1px,
1px, 1px, 1px);height: 1px;margin: -1px;overflow: hidden;padding: 0;position:
absolute;width: 1px;}#sk-container-id-1 div.sk-estimator {font-family:
monospace;background-color: #f0f8ff;border: 1px dotted black;border-radius: 0.25em;box-
sizing: border-box;margin-bottom: 0.5em;}#sk-container-id-1 div.sk-estimator:hover
{background-color: #d4ebff;}#sk-container-id-1 div.sk-parallel-item::after {content:
\"\";width: 100%;border-bottom: 1px solid gray;flex-grow: 1;}#sk-container-id-1 div.sk-
```

label:hover label.sk-toggleable__label {background-color: #d4ebff;}#sk-container-id-1 div.sk-serial::before {content: \"\";position: absolute;border-left: 1px solid gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: 0;}#sk-container-id-1 div.sk-serial {display: flex;flex-direction: column;align-items: center;background-color: white;padding-right: 0.2em;padding-left: 0.2em;position: relative;}#sk-container-id-1 div.sk-item {position: relative;z-index: 1;}#sk-container-id-1 div.sk-parallel {display: flex;align-items: stretch;justify-content: center;background-color: white;position: relative;}#sk-container-id-1 div.sk-item::before, #sk-container-id-1 div.sk-parallel-item::before {content: \"\";position: absolute;border-left: 1px solid gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: -1;}#sk-container-id-1 div.sk-parallel-item {display: flex;flex-direction: column;z-index: 1;position: relative;background-color: white;}#sk-container-id-1 div.sk-parallel-item:first-child::after {align-self: flex-end;width: 50%;}#sk-container-id-1 div.sk-parallel-item:last-child::after {align-self: flex-start;width: 50%;}#sk-container-id-1 div.sk-parallel-item:only-child::after {width: 0;}#sk-container-id-1 div.sk-dashed-wrapped {border: 1px dashed gray;margin: 0 0.4em 0.5em 0.4em;box-sizing: border-box;padding-bottom: 0.4em;background-color: white;}#sk-container-id-1 div.sk-label label {font-family: monospace;font-weight: bold;display: inline-block;line-height: 1.2em;}#sk-container-id-1 div.sk-label-container {text-align: center;}#sk-container-id-1 div.sk-container {/* jupyter's `normalize.less` sets `[hidden] { display: none; }` but bootstrap.min.css set `[hidden] { display: none !important; }` so we also need the `!important` here to be able to override the default hidden behavior on the sphinx rendered scikit-learn.org. See: https://github.com/scikit-learn/scikit-learn/issues/21755 */display: inline-block !important;position: relative;}#sk-container-id-1 div.sk-text-repr-fallback {display: none;}</style><div id=\"sk-container-id-1\" class=\"sk-top-container\"><div class=\"sk-text-repr-fallback\"><pre>LinearRegression()</pre><b>In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. <br />On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.</b></div><div class=\"sk-container\" hidden><div class=\"sk-item\"><div class=\"sk-estimator sk-toggleable\"><input class=\"sk-toggleable__control sk-hidden--visually\" id=\"sk-estimator-id-1\" type=\"checkbox\" checked><label for=\"sk-estimator-id-1\" class=\"sk-toggleable__label sk-toggleable__label-arrow\">LinearRegression</label><div class=\"sk-toggleable__content\"><pre>LinearRegression()</pre></div></div></div></div></div>"
    ],
    "text/plain": [
     "LinearRegression()"
    ]
   },
   "execution_count": 31,
   "metadata": {},
   "output_type": "execute_result"
  }

```
   ],
   "source": [
    "import pandas as pd\n",
    "from sklearn.linear_model import LinearRegression\n",
    "from sklearn.metrics import r2_score, mean_absolute_percentage_error\n",
    "X = df_merged[['runs_scored']] # Independent variable(s)\n",
    "y = df_merged['Rs'] # Dependent variable\n",
    "# Split the data into training and test sets (80% for training, 20% for testing)\n",
    "X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)\n",
    "# Create a LinearRegression model\n",
    "model = LinearRegression()\n",
    "# Fit the model on the training data\n",
    "model.fit(X_train, y_train)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 32,
   "id": "74842fbf-9bb9-48b7-a426-a0e6c7cb482a",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>runs_scored</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>33</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>37</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>13</th>\n",
      "     <td>40</td>\n",
      "    </tr>\n",
      "    <tr>\n",
      "     <th>14</th>\n",
      "     <td>182</td>\n",
      "    </tr>\n",
      "  </tbody>\n",
      "</table>\n",
      "</div>"
     ],
     "text/plain": [
      "    runs_scored\n",
      "0           33\n",
      "3            1\n",
      "4           37\n",
      "13          40\n",
      "14         182"
     ]
    },
    "execution_count": 32,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "X.head()"
  ]
 },
 {
  "cell_type": "code",
   "execution_count": 33,
   "id": "1b530bd8-25c9-4e15-9f0d-2eb3a2c6ed31",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "                            OLS Regression Results                            \n",
"==============================================================================\n",
      "Dep. Variable:                     Rs   R-squared:                       0.080\n",
      "Model:                            OLS   Adj. R-squared:                  0.075\n",
      "Method:                 Least Squares   F-statistic:                     15.83\n",
      "Date:                Wed, 26 Jun 2024   Prob (F-statistic):           0.000100\n",
      "Time:                        00:05:44   Log-Likelihood:                -1379.8\n",
      "No. Observations:                 183   AIC:                             2764.\n",
      "Df Residuals:                     181   BIC:                             2770.\n",
      "Df Model:                           1                                         \n",
      "Covariance Type:            nonrobust                                         \n",
"==============================================================================\n",
      "                  coef    std err          t      P>|t|      [0.025      0.975]\n",
      "------------------------------------------------------------------------------\n",
      "const          430.8473     46.111      9.344      0.000     339.864     521.831\n",
      "runs_scored      0.6895      0.173      3.979      0.000       0.348       1.031\n",
"==============================================================================\n",
      "Omnibus:                       15.690   Durbin-Watson:                   2.100\n",
      "Prob(Omnibus):                 0.000   Jarque-Bera (JB):             18.057\n",
      "Skew:                          0.764   Prob(JB):                  0.000120\n",
      "Kurtosis:                      2.823   Cond. No.                       363.\n",
"==============================================================================\n",
      "\n",
      "Notes:\n",
      "[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.\n"
     ]
    }
   ],
   "source": [
    "import pandas as pd\n",
    "from sklearn.model_selection import train_test_split\n",
    "import statsmodels.api as sm\n",
    "\n",
    "# Assuming df_merged is already defined and contains the necessary columns\n",
    "X = df_merged[['runs_scored']] # Independent variable(s)\n",
    "y = df_merged['Rs'] # Dependent variable\n",
    "\n",
    "# Split the data into training and test sets (80% for training, 20% for testing)\n",
    "X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)\n",
    "\n",
    "# Add a constant to the model (intercept)\n",
    "X_train_sm = sm.add_constant(X_train)\n",
    "\n",
    "# Create a statsmodels OLS regression model\n",
    "model = sm.OLS(y_train, X_train_sm).fit()\n",
    "\n",
   "# Get the summary of the model\n",
   "summary = model.summary()\n",
   "print(summary)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 34,
  "id": "c7153517-a492-4b8f-b7ab-e0a5b6caca65",
  "metadata": {},
  "outputs": [],
  "source": [
   "from fuzzywuzzy import process\n",
   "\n",
   "# Convert to DataFrame\n",
   "df_salary = salary.copy()\n",
   "df_runs = total_wicket_each_year.copy()\n",
   "\n",
   "# Function to match names\n",
   "def match_names(name, names_list):\n",
   "    match, score = process.extractOne(name, names_list)\n",
   "    return match if score >= 80 else None  # Use a threshold score of 80\n",
   "\n",
   "# Create a new column in df_salary with matched names from df_runs\n",
   "df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x,
df_runs['Bowler'].tolist()))\n",
   "\n",
   "# Merge the DataFrames on the matched names\n",
   "df_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Bowler')"
  ]
 },
{
 "cell_type": "code",
 "execution_count": 35,
 "id": "80deb77d-6417-4c77-94ab-67a16866368b",
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/html": [
     "<div>\n",
     "<style scoped>\n",
     "    .dataframe tbody tr th:only-of-type {\n",
     "        vertical-align: middle;\n",
     "    }\n",
     "\n",
     "    .dataframe tbody tr th {\n",
     "        vertical-align: top;\n",
     "    }\n",
     "\n",
     "    .dataframe thead th {\n",
     "        text-align: right;\n",
     "    }\n",
     "</style>\n",
     "<table border=\"1\" class=\"dataframe\">\n",
     "  <thead>\n",
     "    <tr style=\"text-align: right;\">\n",
     "      <th></th>\n",
     "      <th>Player</th>\n",
     "      <th>Salary</th>\n",
     "      <th>Rs</th>\n",
"      <th>international</th>\n",
"      <th>iconic</th>\n",
"      <th>Matched_Player</th>\n",
"      <th>Season</th>\n",
"      <th>Bowler</th>\n",
"      <th>wicket_confirmation</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Anrich Nortje</td>\n",
"      <td>6.5 crore</td>\n",
"      <td>650</td>\n",
"      <td>1</td>\n",
"      <td>NaN</td>\n",
"      <td>A Nortje</td>\n",
"      <td>2020/21</td>\n",
"      <td>A Nortje</td>\n",
"      <td>23</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Anrich Nortje</td>\n",
"      <td>6.5 crore</td>\n",
"      <td>650</td>\n",
"      <td>1</td>\n",
"      <td>NaN</td>\n",
"      <td>A Nortje</td>\n",
"      <td>2021</td>\n",
"      <td>A Nortje</td>\n",
"      <td>12</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>Anrich Nortje</td>\n",
"      <td>6.5 crore</td>\n",
"      <td>650</td>\n",
"      <td>1</td>\n",
"      <td>NaN</td>\n",
"      <td>A Nortje</td>\n",
"      <td>2023</td>\n",
"      <td>A Nortje</td>\n",
"      <td>11</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>6</th>\n",
"      <td>Axar Patel</td>\n",
"      <td>9 crore</td>\n",
"      <td>900</td>\n",
"      <td>0</td>\n",
"      <td>NaN</td>\n",
"      <td>AR Patel</td>\n",
"      <td>2014</td>\n",
"      <td>AR Patel</td>\n",
"      <td>19</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>7</th>\n",
"      <td>Axar Patel</td>\n",
"      <td>9 crore</td>\n",
"      <td>900</td>\n",
"      <td>0</td>\n",
"      <td>NaN</td>\n",
"      <td>AR Patel</td>\n",
"      <td>2015</td>\n",
"      <td>AR Patel</td>\n",
"      <td>14</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>589</th>\n",
"      <td>T. Natarajan</td>\n",
"      <td>3.2 crore</td>\n",
"      <td>320</td>\n",
"      <td>0</td>\n",
"      <td>NaN</td>\n",
"      <td>T Natarajan</td>\n",
"      <td>2020/21</td>\n",
"     <td>T Natarajan</td>\n",
"     <td>19</td>\n",
"    </tr>\n",
"    <tr>\n",
"     <th>591</th>\n",
"     <td>T. Natarajan</td>\n",
"     <td>3.2 crore</td>\n",
"     <td>320</td>\n",
"     <td>0</td>\n",
"     <td>NaN</td>\n",
"     <td>T Natarajan</td>\n",
"     <td>2022</td>\n",
"     <td>T Natarajan</td>\n",
"     <td>20</td>\n",
"    </tr>\n",
"    <tr>\n",
"     <th>592</th>\n",
"     <td>T. Natarajan</td>\n",
"     <td>3.2 crore</td>\n",
"     <td>320</td>\n",
"     <td>0</td>\n",
"     <td>NaN</td>\n",
"     <td>T Natarajan</td>\n",
"     <td>2023</td>\n",
"     <td>T Natarajan</td>\n",
"     <td>13</td>\n",
"    </tr>\n",
"    <tr>\n",
"     <th>593</th>\n",
"     <td>T. Natarajan</td>\n",
"      <td>3.2 crore</td>\n",
"      <td>320</td>\n",
"      <td>0</td>\n",
"      <td>NaN</td>\n",
"      <td>T Natarajan</td>\n",
"      <td>2024</td>\n",
"      <td>T Natarajan</td>\n",
"      <td>13</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>595</th>\n",
"      <td>Umran Malik</td>\n",
"      <td>4 crore</td>\n",
"      <td>400</td>\n",
"      <td>0</td>\n",
"      <td>NaN</td>\n",
"      <td>Umran Malik</td>\n",
"      <td>2022</td>\n",
"      <td>Umran Malik</td>\n",
"      <td>23</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>227 rows × 9 columns</p>\n",
"</div>"
],
"text/plain": [
"            Player     Salary   Rs  international  iconic Matched_Player  \\\n",
"1    Anrich Nortje  6.5 crore  650              1     NaN       A Nortje   \n",
"2    Anrich Nortje  6.5 crore  650              1     NaN       A Nortje   \n",
    "4    Anrich Nortje  6.5 crore  650              1     NaN       A Nortje   \n",
    "6       Axar Patel    9 crore  900              0     NaN       AR Patel   \n",
    "7       Axar Patel    9 crore  900              0     NaN       AR Patel   \n",
    "..             ...        ...  ...            ...     ...            ...   \n",
    "589   T. Natarajan  3.2 crore  320              0     NaN    T Natarajan   \n",
    "591   T. Natarajan  3.2 crore  320              0     NaN    T Natarajan   \n",
    "592   T. Natarajan  3.2 crore  320              0     NaN    T Natarajan   \n",
    "593   T. Natarajan  3.2 crore  320              0     NaN    T Natarajan   \n",
    "595    Umran Malik    4 crore  400              0     NaN    Umran Malik   \n",
    "\n",
    "      Season       Bowler  wicket_confirmation  \n",
    "1    2020/21     A Nortje                   23  \n",
    "2       2021     A Nortje                   12  \n",
    "4       2023     A Nortje                   11  \n",
    "6       2014     AR Patel                   19  \n",
    "7       2015     AR Patel                   14  \n",
    "..       ...          ...                  ...  \n",
    "589  2020/21  T Natarajan                   19  \n",
    "591     2022  T Natarajan                   20  \n",
    "592     2023  T Natarajan                   13  \n",
    "593     2024  T Natarajan                   13  \n",
    "595     2022  Umran Malik                   23  \n",
    "\n",
    "[227 rows x 9 columns]"
   ]
  },
  "execution_count": 35,
  "metadata": {},
  "output_type": "execute_result"
 }
```

```
  ],
  "source": [
   "df_merged[df_merged['wicket_confirmation']>10]"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 46,
  "id": "6dc8d22d-bf66-41c3-929d-3179b70e13c9",
  "metadata": {},
  "outputs": [],
  "source": [
   "#susbsets data for last three years\n",
   "df_merged = df_merged.loc[df_merged['Season'].isin(['2022'])]"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 48,
  "id": "1d391ad4-45b1-48d1-b40b-3bc1fe5efaae",
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "                            OLS Regression Results                            \n",
"==============================================================================\n",
     "Dep. Variable:                     Rs   R-squared:                       0.074\n",
      "Model:                            OLS   Adj. R-squared:                  0.054\n",
      "Method:                 Least Squares   F-statistic:                     3.688\n",
      "Date:                Wed, 26 Jun 2024   Prob (F-statistic):             0.0610\n",
      "Time:                        00:06:13   Log-Likelihood:                -360.96\n",
      "No. Observations:                  48   AIC:                             725.9\n",
      "Df Residuals:                      46   BIC:                             729.7\n",
      "Df Model:                           1                                         \n",
      "Covariance Type:            nonrobust                                         \n",
"========================================================================================\n",
      "                          coef    std err          t      P>|t|      [0.025      0.975]\n",
      "---------------------------------------------------------------------------------------\n",
      "const                 396.6881     91.270      4.346      0.000     212.971     580.405\n",
      "wicket_confirmation    17.6635      9.198      1.920      0.061      -0.851      36.179\n",
"==============================================================================\n",
      "Omnibus:                        6.984   Durbin-Watson:                   2.451\n",
      "Prob(Omnibus):                  0.030   Jarque-Bera (JB):                6.309\n",
      "Skew:                           0.877   Prob(JB):                       0.0427\n",
      "Kurtosis:                       3.274   Cond. No.                         13.8\n",
"==============================================================================\n",
      "\n",
      "Notes:\n",
      "[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.\n"
     ]
    }
   ],
   "source": [
  "import pandas as pd\n",
  "from sklearn.model_selection import train_test_split\n",
  "import statsmodels.api as sm\n",
  "\n",
  "# Assuming df_merged is already defined and contains the necessary columns\n",
  "X = df_merged[['wicket_confirmation']] # Independent variable(s)\n",
  "y = df_merged['Rs'] # Dependent variable\n",
  "\n",
  "# Split the data into training and test sets (80% for training, 20% for testing)\n",
  "X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)\n",
  "\n",
  "# Add a constant to the model (intercept)\n",
  "X_train_sm = sm.add_constant(X_train)\n",
  "\n",
  "# Create a statsmodels OLS regression model\n",
  "model = sm.OLS(y_train, X_train_sm).fit()\n",
  "\n",
  "# Get the summary of the model\n",
  "summary = model.summary()\n",
  "print(summary)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "id": "1dfa2799-8683-46e0-88b2-22c9d4aca283",
 "metadata": {},
 "outputs": [],
 "source": []
}
```

```
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.11.5"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```

REFERENCES

- Books
- Chrome
- Microsoft web browser