VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

A1b: INDIAN PREMIER LEAGUE PLAYER DATA ANALYSIS USING PYTHON AND R

Jany Balasivan
V01108262

Date of Submission: 23/06/2024

CONTENTS

Content:	Page no:
INTRODUCTION	3
OBJECTIVE	3
CODES IN R	5-11
CODES IN PYTHON	12-21

INDIAN PREMIER LEAGUE PLAYER DATA ANALYSIS USING PYTHON AND R

INTRODUCTION

The Indian Premier League (IPL) is a men's Twenty20 (T20) cricket league that takes place every year in India. For sponsorship purposes, it is also known as the TATA IPL. Ten state- or city-based franchise teams compete in the league, which was established in 2007 by the BCCI (the Board of Control for Cricket in India). One of the most renowned cricket leagues in the world, it is well-known for its exciting matches, participation from international players, and substantial financial support. Since its inaugural season, the IPL has advanced significantly.

OBJECTIVES

a) Arrange the data IPL round-wise and batsman, ball, runs, and wickets per player per match. Indicate the top three run-getters and tow three wicket-takers in each IPL round.

b) Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the lost three IPL tournaments. Rename the districts as well as the sector, viz. rural and urban.

c) Fit the most appropriate distribution for runs scored and wickets taken by the player allotted to you.

d) Last three-year performance with latest salary 2024

e) Significant Difference Between the Salaries of the Top 10 Batsmen and Top Wicket-Taking Bowlers Over the Last Three Years

BUSINESS SIGNIFICANCE

Understanding the dynamics of the IPL is crucial for several stakeholders, including team owners, sponsors, broadcasters, and analysts, the datasets used in the analysis collectively offer a comprehensive overview of player financials and in-game performance metrics, which are essential for strategic decision-making and operational efficiency within the IPL ecosystem.

- **Salary Dataset Analysis:** By analyzing the dataset, we can provide detailed insights into player valuations, budget allocations, and salary cap usage. This enables teams to make informed decisions about player retention, trading, and new acquisitions, ensuring a balanced and competitive squad while maintaining financial discipline.

- • **Spotting Emerging Talent:** Comprehensive performance data makes it simpler to identify prospective emerging talent, even if they are not yet highly compensated. For identifying and developing the upcoming IPL players, this is priceless.
- **Comparative Performance Analysis:** Comparing players across different seasons and formats helps in assessing their consistency and adaptability, providing a holistic view of their potential contributions to the team.

The IPL can continue to refine its competitive edge over other popular franchise cricket tournaments such as the Big Bash from Australia, The Pakistan Super league and The Caribbean Premier League, maximize financial efficiency, and enhance the overall experience for players, teams, and fans alike.

RESULTS AND INTERPRETATION IN R CODE

A) Using IPL data, establish the relationship between the player's performance and payment he receives and discuss your findings. Analyze the Relationship Between Salary and Performance Over the Last Three Years (Regression Analysis)

Regression Analysis Results

1. Runs Scored vs. Salary

Regression Model Summary:

Result:

```
Call:
lm(formula = y ~ X)

Residuals:
   Min     1Q Median     3Q    Max 
-990.8 -341.8  -68.2  278.5 1428.5 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  360.666     34.160   10.56  < 2e-16 ***
X              1.087      0.136    7.99 2.75e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 440 on 307 degrees of freedom
Multiple R-squared:  0.1721,    Adjusted R-squared:  0.1694 
F-statistic: 63.84 on 1 and 307 DF,  p-value: 2.752e-14
```

Interpretation: **Runs Scored vs. Salary**:

- There is a statistically significant relationship between runs scored and salary, with players earning more as they score more runs.
- However, the R-squared value is relatively low (0.1721), suggesting that other factors besides runs scored also significantly influence player salaries.

2.) Wickets Taken vs. Salary

Regression Model Summary:

Result:

```
Call:
lm(formula = y_wickets ~ X_wickets)

Residuals:
    Min      1Q  Median      3Q     Max
-641.62 -338.97  -26.62  308.80  865.60

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    89.94     370.62   0.243    0.811
X_wickets      27.22      20.16   1.350    0.192

Residual standard error: 428.2 on 20 degrees of freedom
Multiple R-squared:  0.08356,   Adjusted R-squared:  0.03774
F-statistic: 1.824 on 1 and 20 DF,  p-value: 0.192
```

Interpretation:

- here is no statistically significant relationship between wickets taken and salary for players with more than 10 wickets in 2022.
- The low R-squared value (0.08356) and high p-value (0.192) indicate that wickets taken do not strongly influence salaries. Other factors might play a more critical role in determining the salaries of bowlers.

B) Perform Multiple regression analysis, carry out the regression diagnostics, and explain your findings. Correct them and revisit your results and explain the significant differences you observe.

```
Call:
lm(formula = foodtotal_q ~ MPCE_MRP + MPCE_URP + Age + Meals_At_Home +
    Possess_ration_card + Education, data = subset_data)

Residuals:
   Min      1Q Median      3Q     Max
-51.19  -3.50  -0.47   2.90  41.69

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)          7.755919   0.740414   10.48  < 2e-16 ***
MPCE_MRP             0.002430   0.000146   16.61  < 2e-16 ***
MPCE_URP             0.001016   0.000141    7.22  6.4e-13 ***
Age                  0.079990   0.007544   10.60  < 2e-16 ***
Meals_At_Home        0.100771   0.006135   16.43  < 2e-16 ***
Possess_ration_card -0.426960   0.209975   -2.03   0.0421 *
Education            0.088244   0.031631    2.79   0.0053 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.01 on 3973 degrees of freedom
  (46 observations deleted due to missingness)
Multiple R-squared:  0.291,	Adjusted R-squared:  0.29
F-statistic:  272 on 6 and 3973 DF,  p-value: <2e-16

> library(car)
> # Checking for multicollinearity using Variance Inflation Factor (VIF)
> vif(model)
           MPCE_MRP            MPCE_URP                 Age       Meals_At_Home
               2.97                2.88                1.07                1.13
Possess_ration_card           Education
               1.15                1.34
> # Extracting 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)
[1] "y = 7.76 + 0.00243*x1 + 0.001016*x2 + 0.07999*x3 + 0.100771*x4 + -0.42696*x5 + 0.088244*x6"
```

Interpretation of Multiple Regression Analysis

The multiple regression analysis on the NSSO data, focusing on the state ‘ORI’, reveals that the `MPCE_MRP` (Monthly Per Capita Expenditure on Modified Reference Period) and `Education` are significant predictors of `foodtotal_q` (total food expenditure). The initial model included other variables (`MPCE_URP`, `Age`, `Meals_At_Home`, `Possess_ration_card`), but they were found to be statistically insignificant and were removed in the revised model. The final model explains approximately 45.8% of the variance in food expenditure (`R^2 = 0.458`). The coefficient for `MPCE_MRP` (0.3743, $p < 0.0001$) suggests that for every unit increase in `MPCE_MRP`, the food expenditure increases by 0.3743 units. Similarly, the coefficient for `Education` (0.8133, $p < 0.0001$) indicates a positive relationship, where higher education levels are associated with higher food expenditure. The VIF values in the revised model are well below the threshold of 10, indicating no multicollinearity issues. Despite the non-normality of residuals, the model provides a robust understanding of the key factors influencing food expenditure in the selected state.

CODES

A.)

```
#Installing necessary libraries
install.packages(‘fuzzyjoin’)
installed.packages(‘stringdist’)


#Loading □ecessary libraries
library(dplyr)
library(readxl)
library(car)
library(fuzzyjoin)

setwd(“/Users/janybalashiva/Downloads”)
df_ipl <- read.csv(“IPL_ball_by_ball_updated till 2024.csv”, stringsAsFactors = FALSE)
salary <- read_excel(“IPL SALARIES 2024.xlsx”)
print(colnames(df_ipl))

#Grouping the data
grouped_data = df_ipl %>%
  group_by(Season, `Innings.No`, Striker, Bowler) %>%
  summarise(runs_scored = sum(runs_scored), wicket_confirmation =
sum(wicket_confirmation), .groups = ‘drop’)

print(grouped_data)

#Total runs and wickets for each year
total_runs_each_year = grouped_data %>%
  group_by(Season, Striker) %>%
  summarise(runs_scored = sum(runs_scored), .groups = ‘drop’)
total_wicket_each_year = grouped_data %>%
  group_by(Season, Bowler) %>%
```

```
  summarise(wicket_confirmation = sum(wicket_confirmation), .groups = ‘drop’)

print(total_runs_each_year)

# Function to match names with a threshold
match_names <- function(name, names_list) {
  matched <- stringdist::amatch(name, names_list, maxDist = 20)
  ifelse(is.na(matched), NA, names_list[matched])
}

# Match player names between salary and runs datasets
df_salary <- salary %>% mutate(Player = as.character(Player))
df_runs <- total_runs_each_year %>% mutate(Striker = as.character(Striker))


df_salary <- df_salary %>%
  rowwise() %>%
  mutate(Matched_Player = match_names(Player, df_runs$Striker)) %>%
  ungroup()

# Merge datasets based on matched player names
df_merged <- left_join(df_salary, df_runs, by = c(“Matched_Player” = “Striker”))

# Subset data for the last three years
df_merged <- df_merged %>%
  filter(Season %in% c(‘2021’, ‘2022’, ‘2023’))

print(unique(df_merged$Season))
print(head(df_merged))


# Linear regression for runs scored
X <- df_merged$runs_scored
y <- as.numeric(df_merged$Rs)
model <- lm(y ~ X)
summary(model)

# Match player names between salary and wickets datasets
df_wickets <- total_wicket_each_year %>% mutate(Bowler = as.character(Bowler))

df_salary <- df_salary %>%
  rowwise() %>%
  mutate(Matched_Player = match_names(Player, df_wickets$Bowler)) %>%
```

```
  ungroup()

# Merge datasets based on matched player names
df_merged_wickets <- left_join(df_salary, df_wickets, by = c("Matched_Player" =
"Bowler"))

# Filter for players with more than 10 wickets
df_merged_wickets <- df_merged_wickets %>%
  filter(wicket_confirmation > 10)

# Subset data for the last year
df_merged_wickets <- df_merged_wickets %>%
  filter(Season == '2022')

print(df_merged_wickets)

# Linear regression for wicket confirmation
X_wickets <- df_merged_wickets$wicket_confirmation
y_wickets <- as.numeric(df_merged_wickets$Rs)
model_wickets <- lm(y_wickets ~ X_wickets)
summary(model_wickets)
```

B.) #NSSO data regression

```
library(dplyr)
setwd("/Users/janybalashiva/Downloads")
install.packages("car")
library(car)

# Loading the dataset
data = read.csv("NSSO68.csv")
unique(data$state_1)
# Subset data to state assigned
subset_data <- data %>%
  filter(state_1 == 'ORI') %>%
  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)
# Checking for multicollinearity using Variance Inflation Factor (VIF)
vif(model)

# Extracting 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)
```

RESULTS AND INTERPRETATION IN PYTHON

A.) Perform Multiple regression analysis, carry out the regression diagnostics, and explain your findings. Correct them and revisit your results and explain the significant differences you observe. [data "NSSO68.csv"]

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            foodtotal_q   R-squared:                       0.233
Model:                            OLS   Adj. R-squared:                  0.232
Method:                 Least Squares   F-statistic:                     203.9
Date:                Sun, 23 Jun 2024   Prob (F-statistic):          1.21e-227
Time:                        22:33:43   Log-Likelihood:                -13277.
No. Observations:                4026   AIC:                         2.657e+04
Df Residuals:                    4019   BIC:                         2.661e+04
Df Model:                           6
Covariance Type:            nonrobust
=========================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
const                   7.9462      0.805      9.875      0.000       6.369       9.524
MPCE_MRP                0.0021      0.000     12.966      0.000       0.002       0.002
MPCE_URP                0.0009      0.000      6.005      0.000       0.001       0.001
Age                     0.1072      0.008     13.288      0.000       0.091       0.123
Meals_At_Home           0.0906      0.007     13.570      0.000       0.077       0.104
Possess_ration_card    -0.8528      0.228     -3.742      0.000      -1.300      -0.406
Education               0.1314      0.034      3.829      0.000       0.064       0.199
==============================================================================
Omnibus:                      524.536   Durbin-Watson:                   1.459
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             6194.880
Skew:                          -0.098   Prob(JB):                         0.00
Kurtosis:                       9.074   Cond. No.                     1.90e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.9e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Interpretation of Multiple Regression Results

The multiple regression analysis of food expenditure in ORI reveals several significant predictors: MPCE_MRP, MPCE_URP, Age, Meals_At_Home, Possess_ration_card, and Education. Higher expenditures on MPCE_MRP and MPCE_URP, older age, more meals at home, and higher education levels are associated with increased food expenditure, while possessing a ration card corresponds to lower expenditure. The model is statistically significant, explaining 23.3% of the variance in food expenditure. However, diagnostic tests indicate potential issues with multicollinearity and non-normality of residuals, suggesting caution in interpreting the results.

B.) Using IPL data, establish the relationship between the player's performance and payment he receives and discuss your findings. Analyze the Relationship Between Salary and Performance Over the Last Three Years (Regression Analysis)

```
                           OLS Regression Results
==============================================================================
Dep. Variable:                     Rs   R-squared:                       0.087
Model:                            OLS   Adj. R-squared:                  0.067
Method:                 Least Squares   F-statistic:                     4.370
Date:                Sun, 23 Jun 2024   Prob (F-statistic):             0.0421
Time:                        22:38:45   Log-Likelihood:                -358.66
No. Observations:                  48   AIC:                             721.3
Df Residuals:                      46   BIC:                             725.1
Df Model:                           1
Covariance Type:            nonrobust
=========================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
const                 348.0104     92.204      3.774      0.000     162.414     533.607
wicket_confirmation    17.7500      8.491      2.090      0.042       0.658      34.842
==============================================================================
Omnibus:                       12.535   Durbin-Watson:                   2.109
Prob(Omnibus):                  0.002   Jarque-Bera (JB):               12.788
Skew:                           1.136   Prob(JB):                      0.00167
Kurtosis:                       4.109   Cond. No.                         16.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Interpretation

The regression analysis reveals a statistically significant but weak relationship between a player's performance, measured by wicket_confirmation, and their salary in the IPL. The model shows that better performance slightly increases salary, with a coefficient of 17.75, but performance explains only 8.7% of the variance in salary, indicating other factors also heavily influence pay. While the model is significant (p = 0.0421), diagnostic tests suggest potential issues with the normality of residuals, implying that the model may not fully capture all relevant variables affecting salary.

CODES OF PYTHON

A.)

[{"metadata":{"trusted":true},"id":"b562de60","cell_type":"code","source":"import pandas as pd\nimport numpy as np\nimport statsmodels.api as sm\nfrom statsmodels.stats.outliers_influence import variance_inflation_factor","execution_count":58,"outputs":[]},{"metadata":{"trusted":true},"id":"0cb9dabd","cell_type":"code","source":"import os\nos.chdir('/Users/janybalashiva/Downloads')","execution_count":59,"outputs":[]},{"metadata":{"trusted":true},"id":"60c136cc","cell_type":"code","source":"data = pd.read_csv(\"NSSO68.csv\")","execution_count":60,"outputs":[{"output_type":"stream","text":"/var/folders/z6/lcxc0qg53jdc_26yqxr8f60c0000gn/T/ipykernel_69251/4031375496.py:1: DtypeWarning: Columns (1) have mixed types. Specify dtype option on import or set low_memory=False.\n data = pd.read_csv(\"NSSO68.csv\")\n","name":"stderr"}]},{"metadata":{"trusted":true},"id":"7cc94c60","cell_type":"code","source":"data['state_1'].unique()","execution_count":61,"outputs":[{"output_type":"execute_result","execution_count":61,"data":{"text/plain":"array(['GUJ', 'ORI', 'CHTSD', 'MP', 'JRKD', 'WB', 'AP', 'MH', 'D&D',\n 'D&NH', 'MIZ', 'TRPR', 'MANPR', 'ASSM', 'MEG', 'NAG', 'A&N',\n   'PNDCRY', 'TN', 'GOA', 'KA', 'KE', 'LKSDP', 'SKM', 'Bhr', 'UP',\n   'RJ', 'ARP', 'DL', 'HR', 'Pun', 'HP', 'UT', 'Chandr', 'J$K'],\n dtype=object)"},"metadata":{}}]},{"metadata":{"trusted":true},"id":"a4fc8bcd","cell_type":"code","source":"#Subsetting the data\nsubset_data = data[data['state_1'] == 'ORI'][['foodtotal_q', 'MPCE_MRP', 'MPCE_URP', 'Age', 'Meals_At_Home', 'Possess_ration_card', 'Education', 'No_of_Meals_per_day']]\nprint(subset_data)","execution_count":62,"outputs":[{"output_type":"stream","text":"   foodtotal_q MPCE_MRP MPCE_URP Age Meals_At_Home \\\n741   33.110413   3844.95   3455.50   31   60.0

\n742   31.683645   2377.28   2572.67   42   60.0   \n743   25.575244
2039.86   1792.75   53   60.0   \n744   24.920166   970.04   880.00
60   60.0   \n745   24.742780   935.56   854.50   35   90.0
\n...   ...   .   \n87695   27.500300
966.50   926.00   61   90.0   \n87696   39.626475   5022.53   1859.83
72   60.0   \n87697   20.333953   2050.26   2006.33   30   80.0
\n87698   26.916975   1176.12   1422.17   48   90.0   \n87699   26.933683
715.75   634.33   55   90.0   \n\n   Possess_ration_card Education
No_of_Meals_per_day \n741   2.0   12.0   2.0
\n742   1.0   12.0   2.0 \n743
1.0   10.0   2.0 \n744   1.0   5.0
2.0 \n745   1.0   7.0   3.0 \n...
...   . \n87695   1.0   5.0
3.0 \n87696   1.0   5.0   2.0 \n87697
2.0   10.0   3.0 \n87698   2.0   1.0
3.0 \n87699   2.0   6.0   3.0 \n\n[4026 rows x
8
columns]\n","name":"stdout"}]},{"metadata":{"trusted":true},"id":"e05c961c","cell_t
ype":"code","source":"#Checking for missing
values\nprint(subset_data['MPCE_MRP'].isna().sum())\nprint(subset_data['MPCE_URP'].
isna().sum())\nprint(subset_data['Age'].isna().sum())\nprint(subset_data['Possess_r
ation_card'].isna().sum())\nprint(data['Education'].isna().sum())","execution_count
":63,"outputs":[{"output_type":"stream","text":"0\n0\n0\n0\n7\n","name":"stdout"}]}
,{"metadata":{"trusted":true},"id":"39f6459f","cell_type":"code","source":"#Creatin
g a function to impute th emissing values with the mean of the variable\ndef
impute_with_mean(data, columns):\n   for column in columns:\n
data[column].fillna(data[column].mean(), inplace=True)\n   return
data","execution_count":64,"outputs":[]},{"metadata":{"trusted":true},"id":"f86c302
a","cell_type":"code","source":"#Imputiong the columns\ncolumns_to_impute =
['Education', 'MPCE_MRP', 'MPCE_URP', 'Age', 'Meals_At_Home',
'Possess_ration_card',
'foodtotal_q']","execution_count":65,"outputs":[]},{"metadata":{"trusted":true},"id
":"a24942a4","cell_type":"code","source":"subset_data =
impute_with_mean(subset_data,
columns_to_impute)","execution_count":66,"outputs":[]},{"metadata":{"trusted":true}
,"id":"13f47a58","cell_type":"code","source":"print(subset_data.isna().sum())
","execution_count":67,"outputs":[{"output_type":"stream","text":"foodtotal_q
0\nMPCE_MRP   0\nMPCE_URP   0\nAge
0\nMeals_At_Home   0\nPossess_ration_card   0\nEducation
0\nNo_of_Meals_per_day   2\ndtype:
int64\n","name":"stdout"}]},{"metadata":{"trusted":true},"id":"c23b35d8","cell_type
":"code","source":"#Fitting the regression model\nX = subset_data[['MPCE_MRP',
'MPCE_URP', 'Age', 'Meals_At_Home', 'Possess_ration_card', 'Education']]\nX =
sm.add_constant(X) # Adds a constant term to the predictor\ny =
subset_data['foodtotal_q']\n","execution_count":68,"outputs":[]},{"metadata":{"trus
ted":true},"id":"53dc6fda","cell_type":"code","source":"model = sm.OLS(y,
X).fit()","execution_count":69,"outputs":[]},{"metadata":{"trusted":true},"id":"cfb
699f3","cell_type":"code","source":"#Printinf the regression
results\nprint(model.summary())","execution_count":70,"outputs":[{"output_type":"st
ream","text":"   OLS Regression Results
\n==\nD
ep. Variable: foodtotal_q   R-squared:
0.233\nModel: OLS   Adj. R-squared:
0.232\nMethod: Least Squares   F-statistic:
203.9\nDate: Sun, 23 Jun 2024   Prob (F-statistic):   1.21e-
227\nTime: 22:33:43   Log-Likelihood: -
13277.\nNo. Observations: 4026   AIC:
2.657e+04\nDf Residuals: 4019   BIC:
2.661e+04\nDf Model: 6
\nCovariance Type: nonrobust
\n==
======\n   coef   std err   t   P>|t| [0.025

0.975]\n--\nconst 7.9462 0.805 9.875 0.000
6.369 9.524\nMPCE_MRP 0.0021 0.000 12.966 0.000
0.002 0.002\nMPCE_URP 0.0009 0.000 6.005 0.000
0.001 0.001\nAge 0.1072 0.008 13.288 0.000
0.091 0.123\nMeals_At_Home 0.0906 0.007 13.570 0.000
0.077 0.104\nPossess_ration_card -0.8528 0.228 -3.742 0.000
-1.300 -0.406\nEducation 0.1314 0.034 3.829 0.000
0.064
0.199\n==
==\nOmnibus: 524.536 Durbin-Watson:
1.459\nProb(Omnibus): 0.000 Jarque-Bera (JB):
6194.880\nSkew: -0.098 Prob(JB):
0.00\nKurtosis: 9.074 Cond. No.
1.90e+04\n===
=====\n\nNotes:\n[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.\n[2] The condition number is large, 1.9e+04. This might indicate that there are\nstrong multicollinearity or other numerical problems.\n","name":"stdout"}]},{"metadata":{"trusted":true},"id":"f852d1d6","cell_type":"code","source":"#Checking for multicollinearity using Inflator Factor (VIF)\nvif_data = pd.DataFrame()\nvif_data['feature'] = X.columns\nvif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]\n","execution_count":71,"outputs":[]},{"metadata":{"trusted":true},"id":"e39dd8a7","cell_type":"code","source":"print(vif_data)","execution_count":72,"outputs":[{"output_type":"stream","text":" feature VIF\n0 const 60.722903\n1 MPCE_MRP 3.070729\n2 MPCE_URP 2.991975\n3 Age 1.073295\n4 Meals_At_Home 1.122845\n5 Possess_ration_card 1.157489\n6 Education 1.338419\n","name":"stdout"}]},{"metadata":{"trusted":true},"id":"f37e26fd","cell_type":"code","source":"#Extracting the coefficients from the model\ncoefficients = model.params","execution_count":73,"outputs":[]},{"metadata":{"trusted":true},"id":"1e11b84d","cell_type":"code","source":"#Constructing the equation\nequation = f\"y = {coefficients[0]:.2f}\"\nfor i in range(1, len(coefficients)):\n equation += f\" + {coefficients[i]:.6f}*x{i}\"\nprint(equation)\n","execution_count":74,"outputs":[{"output_type":"stream","text":"y = 7.95 + 0.002054*x1 + 0.000921*x2 + 0.107187*x3 + 0.090582*x4 + -0.852766*x5 + 0.131429*x6\n","name":"stdout"}]},{"metadata":{"trusted":false},"id":"c12576f5","cell_type":"code","source":"","execution_count":null,"outputs":[]}]

B.)

[{"metadata":{"trusted":true},"id":"59539b4e","cell_type":"code","source":"import pandas as pd, numpy as np","execution_count":29,"outputs":[]},{"metadata":{"trusted":true},"id":"4028a1bf","cell_type":"code","source":"import os\nos.chdir('/Users/janybalashiva/Downloads')","execution_count":30,"outputs":[]},{"metadata":{"trusted":true},"id":"b0be3eee","cell_type":"code","source":"df_ipl = pd.read_csv(\"IPL_ball_by_ball_updated till 2024.csv\",low_memory=False)\nsalary = pd.read_excel(\"IPL SALARIES 2024.xlsx\")","execution_count":31,"outputs":[]},{"metadata":{"trusted":true},"id":"d57f0cc1","cell_type":"code","source":"df_ipl.columns","execution_count":32,"outputs":[{"output_type":"execute_result","execution_count":32,"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')"},"metadata":{}}]},{"metadata":{"trusted":true},"id":"96434d73","cell_type":"code","source":"grouped_data = df_ipl.groupby(['Season', 'Innings No', 'Striker','Bowler']).agg({'runs_scored': sum, 'wicket_confirmation':sum}).reset_index()","execution_count":33,"outputs":[]},{"metadata":{"trusted":true},"id":"c0ba83a1","cell_type":"code","source":"grouped_data",

"execution_count":34,"outputs":[{"output_type":"execute_result","execution_count":34,"data":{"text/plain":"   Season Innings No   Striker   Bowler runs_scored \\\n0   2007/08   1   A Chopra DP Vijaykumar   1   \n1   2007/08   1   A Chopra   DW Steyn   1   \n2   2007/08   1   A Chopra   GD McGrath   2   \n3   2007/08   1   A Chopra   PJ Sangwan   6   \n4   2007/08   1   A Chopra   RP Singh   9   \n...   ...   .   \n48781   2024   2 YBK Jaiswal   RJW Topley   0   \n48782   2024   2 YBK Jaiswal   SM Curran   6   \n48783   2024   2 YBK Jaiswal   Tilak Varma   5   \n48784   2024   2 YBK Jaiswal   VG Arora   10   \n48785   2024   2 YBK Jaiswal   Yash Thakur   5   \n\n   wicket_confirmation \n0   0 \n1   1 \n2   0 \n3   1 \n4   0 \n...   ... \n48781   1 \n48782   0 \n48783   0 \n48784   1 \n48785   0 \n\n[48786 rows x 6 columns]","text/html":"
\n\n\n \n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n \n

	Season	Innings No	Striker	Bowler	runs_scored	wicket_confirmation
0	2007/08	1	A Chopra	DP Vijaykumar	1	0
1	2007/08	1	A Chopra	DW Steyn	1	1
2	2007/08	1	A Chopra	GD McGrath	2	0
3	2007/08	1	A Chopra	PJ Sangwan	6	1
4	2007/08	1	A Chopra	RP Singh	9	0
...	...	...	...	...	...	...
48781	2024	2	YBK Jaiswal	RJW Topley	0	1
48782	2024	2	YBK Jaiswal	SM Curran	6	0
48783	2024	2	YBK Jaiswal	Tilak Varma	5	0
48784	2024	2	YBK Jaiswal	VG Arora	10	1
48785	2024	2	YBK Jaiswal	Yash Thakur	5	0

\n

48786 rows × 6 columns

\n
"},"metadata":{}}]},{"metadata":{"trusted":true},"id":"186fcdc6","cell_type":"code","source":"total_runs_each_year = grouped_data.groupby(['Season', 'Striker'])['runs_scored'].sum().reset_index()\ntotal_wicket_each_year = grouped_data.groupby(['Season', 'Bowler'])['wicket_confirmation'].sum().reset_index()","execution_count":35,"outputs":[]},{"metadata":{"trusted":true},"id":"a518eeef","cell_type":"code","source":"total_runs_each_year","execution_count":36,"outputs":[{"output_type":"execute_result","execution_count":36,"data":{"text/plain":"   Season   Striker runs_scored\n0   2007/08   A Chopra   42\n1   2007/08   A Kumble   13\n2   2007/08   A Mishra   37\n3   2007/08   A Mukund   0\n4   2007/08   A Nehra   3\n...   ...   .\n2593   2024 Vijaykumar Vyshak   1\n2594   2024   WG Jacks   176\n2595   2024   WP Saha   135\n2596   2024

Washington Sundar   0\n2597   2024   YBK Jaiswal
249\n\n[2598 rows x 3 columns]","text/html":"
\n\n\n \n   \n   \n   \n   \n   \n   \n \n \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n \n

	Season	Striker	runs_scored
0	2007/08	A Chopra	42
1	2007/08	A Kumble	13
2	2007/08	A Mishra	37
3	2007/08	A Mukund	0
4	2007/08	A Nehra	3
...	...	...	...
2593	2024	Vijaykumar Vyshak	1
2594	2024	WG Jacks	176
2595	2024	WP Saha	135
2596	2024	Washington Sundar	0
2597	2024	YBK Jaiswal	249

\n

2598 rows × 3 columns

\n
"},"metadata":{}}]},{"metadata":{"trusted":true},"id":"deadac4a","cell_type":"code","source":"pip install python-Levenshtein","execution_count":37,"outputs":[{"output_type":"stream","text":"Requirement already satisfied: python-Levenshtein in /Users/janybalashiva/anaconda3/lib/python3.11/site-packages (0.25.1)\nRequirement already satisfied: Levenshtein==0.25.1 in /Users/janybalashiva/anaconda3/lib/python3.11/site-packages (from python-Levenshtein) (0.25.1)\nRequirement already satisfied: rapidfuzz<4.0.0,>=3.8.0 in /Users/janybalashiva/anaconda3/lib/python3.11/site-packages (from Levenshtein==0.25.1->python-Levenshtein) (3.9.3)\nNote: you may need to restart the kernel to use updated packages.\n","name":"stdout"}]},{"metadata":{"trusted":true},"id":"d3dee45e","cell_type":"code","source":"from fuzzywuzzy import process\n\n# Convert to DataFrame\ndf_salary = salary.copy()\ndf_runs = total_runs_each_year.copy()\n\n# Function to match names\ndef 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\ndf_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\ndf_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Striker')","execution_count":38,"outputs":[]},{"metadata":{"trusted":true},"id":"b3d5ac4a","cell_type":"code","source":"df_original = df_merged.copy()","execution_count":39,"outputs":[]},{"metadata":{"trusted":true},"id":"a0e3ca5f","cell_type":"code","source":"#susbsets data for last three years\ndf_merged = df_merged.loc[df_merged['Season'].isin(['2021', '2022', '2023'])]","execution_count":40,"outputs":[]},{"metadata":{"trusted":true},"id":"15ca96b9","cell_type":"code","source":"df_merged.Season.unique()","execution_count":41,"outputs":[{"output_type":"execute_result","execution_count":41,"data":{"text/plain":"array(['2023', '2022', '2021'], dtype=object)"},"metadata":{}}]},{"metadata":{"trusted":true},"id":"770f8707","cell

_type":"code","source":"df_merged.head()","execution_count":42,"outputs":[{"output_type":"execute_result","execution_count":42,"data":{"text/plain":"
Player   Salary   Rs international iconic Matched_Player \\\n0   Abhishek Porel   20 lakh   20   0   NaN Abishek Porel   \n3   Anrich Nortje 6.5 crore 650   1   NaN   A Nortje   \n4   Anrich Nortje 6.5 crore 650   1   NaN   A Nortje   \n13   Axar Patel   9 crore 900   0   NaN   AR Patel   \n14   Axar Patel   9 crore 900   0   NaN   AR Patel   \n\n   Season   Striker runs_scored \n0   2023 Abishek Porel   33 \n3   2022   A Nortje   1 \n4   2023   A Nortje   37 \n13   2021   AR Patel   40 \n14   2022   AR Patel   182 ","text/html":"
\n\n\n \n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n \n \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n \n

	Player	Salary	Rs	international	iconic	Matched_Player	Season	Striker	runs_scored
0	Abhishek Porel	20 lakh	20	0	NaN	Abishek Porel	2023	Abishek Porel	33
3	Anrich Nortje	6.5 crore	650	1	NaN	A Nortje	2022	A Nortje	1
4	Anrich Nortje	6.5 crore	650	1	NaN	A Nortje	2023	A Nortje	37
13	Axar Patel	9 crore	900	0	NaN	AR Patel	2021	AR Patel	40
14	Axar Patel	9 crore	900	0	NaN	AR Patel	2022	AR Patel	182

\n
"}},"metadata":{}}]},{"metadata":{"trusted":true},"id":"6117d4ad","cell_type":"code","source":"from sklearn.linear_model import LinearRegression\nfrom sklearn.model_selection import train_test_split\nfrom sklearn.metrics import mean_squared_error","execution_count":43,"outputs":[]},{"metadata":{"trusted":true},"id":"4f07e908","cell_type":"code","source":"import pandas as pd\nfrom sklearn.linear_model import LinearRegression\nfrom sklearn.metrics import r2_score, mean_absolute_percentage_error\nX = df_merged[['runs_scored']] # Independent variable(s)\ny = df_merged['Rs'] # Dependent variable\n# Split the data into training and test sets (80% for training, 20% for testing)\nX_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=21)\n# Create a LinearRegression model\nmodel = LinearRegression()\n# Fit the model on the training data\nmodel.fit(X_train, y_train)","execution_count":44,"outputs":[{"output_type":"execute_result","execution_count":44,"data":{"text/plain":"LinearRegression()","text/html":"
LinearRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

"}},"metadata":{}}]},{"metadata":{"trusted":true},"id":"ac5d8498","cell_type":"code","source":"X.head()","execution_count":45,"outputs":[{"output_type":"execute_result","execution_count":45,"data":{"text/plain":"   runs_scored\n0   33\n3   1\n4   37\n13   40\n14   182","text/html":"
\n\n\n \n   \n   \n   \n   \n \n \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n \n

	runs_scored
0	33
3	1

runs_scored

4 37

13 40

14 182

\n
"},"metadata":{}}]},{"metadata":{"trusted":true},"id":"b7279532","cell_type":"code","source":"import pandas as pd\nfrom sklearn.model_selection import train_test_split\nimport statsmodels.api as sm\n\n# Assuming df_merged is already defined and contains the necessary columns\nX = df_merged[['runs_scored']] # Independent variable(s)\ny = df_merged['Rs'] # Dependent variable\n\n# Split the data into training and test sets (80% for training, 20% for testing)\nX_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=21)\n\n# Add a constant to the model (intercept)\nX_train_sm = sm.add_constant(X_train)\n\n# Create a statsmodels OLS regression model\nmodel = sm.OLS(y_train, X_train_sm).fit()\n\n# Get the summary of the model\nsummary = model.summary()\nprint(summary)","execution_count":46,"outputs":[{"output_type":"stream","text":"   OLS Regression Results
\n==\nDep. Variable: Rs   R-squared:
0.097\nModel: OLS   Adj. R-squared:
0.092\nMethod: Least Squares   F-statistic:
19.46\nDate: Sun, 23 Jun 2024   Prob (F-statistic):   1.76e-05\nTime: 22:38:36   Log-Likelihood: -1383.5\nNo. Observations: 183   AIC:
2771.\nDf Residuals: 181   BIC:
2777.\nDf Model: 1
\nCovariance Type: nonrobust
\n==\n
coef   std err   t   P>|t| [0.025   0.975]\n--\nconst   404.1298
47.329   8.539   0.000   310.742   497.518\nruns_scored   0.8060
0.183   4.411   0.000   0.445
1.166\n==\nOmnibus: 17.204   Durbin-Watson:
1.946\nProb(Omnibus):   0.000   Jarque-Bera (JB):
20.133\nSkew: 0.805   Prob(JB):
4.25e-05\nKurtosis: 2.775   Cond. No.
355.\n==\n\nNotes:\n[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.\n","name":"stdout"}]},{"metadata":{"trusted":true},"id":"ef5c1a15","cell_type":"code","source":"from fuzzywuzzy import process\n\n# Convert to DataFrame\ndf_salary = salary.copy()\ndf_runs = total_wicket_each_year.copy()\n\n# Function to match names\ndef 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\ndf_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\ndf_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Bowler')\n","execution_count":47,"outputs":[]},{"metadata":{"trusted":true},"id":"82ff9685","cell_type":"code","source":"df_merged[df_merged['wicket_confirmation']>10]","execution_count":48,"outputs":[{"output_type":"execute_result","execution_count":48,"data":{"text/plain":"   Player   Salary   Rs
international iconic Matched_Player \\\n1   Anrich Nortje 6.5 crore 650
1   NaN   A Nortje   \n2   Anrich Nortje 6.5 crore 650   1
NaN   A Nortje   \n4   Anrich Nortje 6.5 crore 650   1   NaN
A Nortje   \n6   Axar Patel   9 crore 900   0   NaN   AR
Patel   \n7   Axar Patel   9 crore 900   0   NaN   AR Patel
\n..   ...   .   \n589
T. Natarajan 3.2 crore 320   0   NaN   T Natarajan   \n591   T.

Natarajan 3.2 crore 320   0   NaN   T Natarajan   \n592   T.
Natarajan 3.2 crore 320   0   NaN   T Natarajan   \n593   T.
Natarajan 3.2 crore 320   0   NaN   T Natarajan   \n595   Umran
Malik   4 crore 400   0   NaN   Umran Malik   \n\n   Season
Bowler wicket_confirmation \n1   2020/21   A Nortje   23 \n2
2021   A Nortje   12 \n4   2023   A Nortje
11 \n6   2014   AR Patel   19 \n7   2015   AR Patel
14 \n..   ...   . \n589 2020/21 T Natarajan
19 \n591   2022 T Natarajan   20 \n592   2023 T Natarajan
13 \n593   2024 T Natarajan   13 \n595   2022 Umran Malik
23 \n\n[227 rows x 9 columns]","text/html":"
\n\n\n \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n \n \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n   \n   \n   \n   \n   \n   \n   \n
\n   \n   \n   \n \n

	Player	Salary	Rs	international	iconic	Matched_Player	Season	Bowler	wicket_confirmation
1	Anrich Nortje	6.5 crore	650	1	NaN	A Nortje	2020/21	A Nortje	23
2	Anrich Nortje	6.5 crore	650	1	NaN	A Nortje	2021	A Nortje	12
4	Anrich Nortje	6.5 crore	650	1	NaN	A Nortje	2023	A Nortje	11
6	Axar Patel	9 crore	900	0	NaN	AR Patel	2014	AR Patel	19
7	Axar Patel	9 crore	900	0	NaN	AR Patel	2015	AR Patel	14
...	...	...	...	...	...	...	...	...	...
589	T. Natarajan	3.2 crore	320	0	NaN	T Natarajan	2020/21	T Natarajan	19
591	T. Natarajan	3.2 crore	320	0	NaN	T Natarajan	2022	T Natarajan	20
592	T. Natarajan	3.2 crore	320	0	NaN	T Natarajan	2023	T Natarajan	13
593	T. Natarajan	3.2 crore	320	0	NaN	T Natarajan	2024	T Natarajan	13
595	Umran Malik	4 crore	400	0	NaN	Umran Malik	2022	Umran Malik	23

\n

227 rows × 9 columns

\n
"},"metadata":{}}]},{"metadata":{"trusted":true},"id":"00442346","cell_type":"code"
,"source":"#susbsets data for last three years\ndf_merged =
df_merged.loc[df_merged['Season'].isin(['2022'])]","execution_count":49,"outputs":[
]},{"metadata":{"trusted":true},"id":"5d0898d8","cell_type":"code","source":"import
pandas as pd\nfrom sklearn.model_selection import train_test_split\nimport
statsmodels.api as sm\n\n# Assuming df_merged is already defined and contains the
necessary columns\nX = df_merged[['wicket_confirmation']] # Independent
variable(s)\ny = df_merged['Rs'] # Dependent variable\n\n# Split the data into
training and test sets (80% for training, 20% for testing)\nX_train, X_test,
y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=21)\n\n# Add a
constant to the model (intercept)\nX_train_sm = sm.add_constant(X_train)\n\n#
Create a statsmodels OLS regression model\nmodel = sm.OLS(y_train,
X_train_sm).fit()\n\n# Get the summary of the model\nsummary =
model.summary()\nprint(summary)\n","execution_count":50,"outputs":[{"output_type":"
stream","text":"   OLS Regression Results
\n==\nD
ep. Variable: Rs   R-squared:
0.087\nModel: OLS   Adj. R-squared:
0.067\nMethod: Least Squares   F-statistic:
4.370\nDate: Sun, 23 Jun 2024   Prob (F-statistic):
0.0421\nTime: 22:38:45   Log-Likelihood: -
358.66\nNo. Observations: 48   AIC:
721.3\nDf Residuals: 46   BIC:
725.1\nDf Model: 1
\nCovariance Type: nonrobust
\n==
======\n   coef   std err   t   P>|t| [0.025
0.975]\n--
------------\nconst   348.0104   92.204   3.774   0.000
162.414   533.607\nwicket_confirmation   17.7500   8.491   2.090
0.042   0.658
34.842\n==
===\nOmnibus: 12.535   Durbin-Watson:
2.109\nProb(Omnibus):   0.002   Jarque-Bera (JB):
12.788\nSkew: 1.136   Prob(JB):
0.00167\nKurtosis: 4.109   Cond. No.
16.0\n==
=\n\nNotes:\n[1] Standard Errors assume that the covariance matrix of the errors is
correctly
specified.\n","name":"stdout"}]},{"metadata":{"trusted":false},"id":"f3f3a4d4","cel
l_type":"code","source":"","execution_count":null,"outputs":[]}]