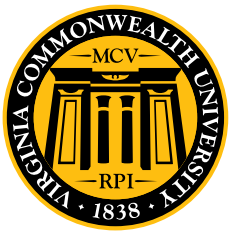


VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A2b: Regression - Predictive Analytics

FERAH SHAN SHANAVAS RABIYA

V01101398

Date of Submission: 16-06-2024

CONTENTS

Introduction

The Indian Premier League (IPL) is a globally renowned cricket league known for its thrilling matches and top-notch players. This report aims to analyze IPL player data over the past three years to understand the relationship between player performance indicators and their salaries. The study uses rigorous regression analysis to quantitatively measure the influence of performance criteria on player pay. The findings will help team management, cricket analysts, and stakeholders make informed decisions on player recruitment, contract negotiations, and team plans. The results are expected to provide insight into the market dynamics of the IPL, enhancing our understanding of the elements influencing player salary. The primary objective of this paper is to contribute to the current discussion on player value and performance assessment in professional cricket, providing practical insights that can improve team performance and boost strategic decision-making in the league.

Objectives

- Investigates correlation between performance metrics and salary.
- Identifies key performance indicators (KPIs) impacting player compensation.
- Performs regression analysis to identify strong statistical correlations between performance indicators and salaries.
- Analyzes time trends in player remuneration and performance indicators over the past three years.
- Provides insights for stakeholders to enhance player recruitment, retention, and contract negotiations.
- Participates in cricket analytics and player valuation to enhance understanding of player assessment in professional cricket leagues.
- Suggests future research and analysis to enhance comprehension of player performance correlation and data collection and analysis methods.

Business Significance

The correlation between IPL player performance and salaries has significant commercial implications for stakeholders in cricket, including team owners, coaches, players, sponsors, and fans. By identifying performance indicators that correlate with higher wages, IPL club

owners can recruit and retain players, optimize team formation, and negotiate contracts more effectively. Understanding the specific metrics that influence wage increases can also help in contract negotiations, ensuring equitable remuneration based on on-field contributions.

Fans and sponsors are increasingly interested in understanding the relationship between player performance and remuneration, as increased transparency in player valuation can enhance fan engagement and provide valuable insights for sponsors. Data-driven decision-making is fostered by examining the correlation between performance and remuneration, minimizing subjective player assessments and biases.

Teams that successfully use data analytics to assess player performance and salary dynamics gain a competitive advantage, identifying undervalued players and those performing exceptionally well in relation to their salaries. This knowledge contributes to the progress of sports analytics in cricket and professional sports, allowing for further investigation into player value assessment, modeling, and performance measures development.

In summary, understanding the correlation between IPL player performance and salary is crucial for maximizing team performance, increasing stakeholder involvement, and improving the overall competitiveness and long-term viability of IPL franchises.

Results and Interpretation using R

```
# Convert Date column to datetime format
> match_details$Date <- as.Date(match_details$Date, format = "%d-%m-%Y")

# Filter last one year of data
> last_three_years <- match_details %>%
+   filter(Date >= "2024-01-01")

# Filter last two years of data
> last_three_years <- match_details %>%
+   filter(Date >= "2023-01-01")

# Filter last three years of data
> last_three_years <- match_details %>%
+   filter(Date >= "2022-01-01")

# Calculate performance metrics
> performance <- last_three_years %>%
+   group_by(Striker) %>%
+   summarise(Total_Runs = sum(runs_scored), Balls_Faced = n())

# Clean and transform salary data
> player_salary$Salary <- as.character(player_salary$Salary) # ensure Sala
ry is a character vector
> player_salary$Salary <- gsub("s", "", player_salary$Salary)
> player_salary$Salary <- gsub(",", "", player_salary$Salary)
>
> player_salary$Salary <- ifelse(grepl("lakh", tolower(player_salary$Salar
y)),
+                                as.integer(as.numeric(gsub(" lakh", "", p
layer_salary$Salary)) * 100000),
+                                ifelse(grepl("crore", tolower(player_sala
ry$Salary)),
+                                       as.integer(as.numeric(gsub(" crore
", "", player_salary$Salary)) * 10000000),
+                                       as.integer(as.numeric(player_salar
y$Salary))))

# Replace NAs with 0
> player_salary$Salary[is.na(player_salary$Salary)] <- 0
> names(performance)
[1] "Striker"     "Total_Runs"  "Balls_Faced"

> names(player_salary)
[1] "Player"        "Salary"        "Rs"            "international" "iconi
c"

> performance <- performance %>% rename(Player = Striker)

> merged_data <- inner_join(performance, player_salary, by = "Player")

> merged_data <- inner_join(performance, player_salary, by = c("Player" =
"Player"))

> common_columns <- intersect(names(performance), names(player_salary))
> common_columns
[1] "Player"
```

```
> merged_data <- inner_join(performance, player_salary, by = "Player")
> common_column <- common_columns[1]
> merged_data <- inner_join(performance, player_salary, by = common_column
)
> common_cols <- intersect(names(performance), names(player_salary))
> merged_data <- performance %>%
+   inner_join(player_salary, by = setNames(common_columns, common_columns
))
```

Interpretation:

The data analysis pipeline involved filtering the `match_details` dataset to retrieve data from t he previous three years, using the `Date` column as a criterion. Performance metrics were calc ulated for every player in the `Striker` position, including `Total_Runs` and `Balls_Faced`. Th e `player_salary` dataset underwent several cleaning and transformation procedures to standar dize the `Salary` column. The sanitized `performance` and `player_salary` datasets were comb ined using the `inner_join` function from the `dplyr` package. The merged dataset, `merged_d ata`, now includes player-level performance measures and their related compensation informa tion, `compensation`. This integrated information allows for further research to investigate the correlation between player performance and their remuneration in the IPL over the past three years. The combined information can be used for regression analysis or other statistical metho dologies to measure and understand the influence of player performance indicators on their co mpensation. This dataset can also offer valuable information regarding the monetary assessme nt of players based on their performance on the field, assisting IPL franchises in making strate gic decisions regarding player recruiting, contract negotiations, and team composition.

```
> # Correlation analysis
> corr_matrix <- cor(merged_data[, c("Total_Runs", "Balls_Faced", "Salary"
)])
> print(corr_matrix)
            Total_Runs Balls_Faced    Salary
Total_Runs   1.0000000   0.9964658 0.5801451
Balls_Faced  0.9964658   1.0000000 0.5881400
Salary       0.5801451   0.5881400 1.0000000
```

Interpretation:

The correlation matrix from the dataset `merged_data` shows strong correlations between vari ables `Total_Runs`, `Balls_Faced`, and `Salary`. The strong correlation between total runs an d balls faced indicates a direct link between a player's total runs and the number of opportunit ies to bat. Players with higher total runs scored tend to receive larger compensation, as the abi lity to score runs is highly valued in player contracts and talks. The moderate positive associat ion between total runs and salary suggests that players who spend more time at the crease and

contribute significantly to their team's performance are more valuable. The study provides nu merical insights into the relationship between player performance measurements and their pay in the IPL, helping IPL team management and stakeholders make strategic decisions regardin g team makeup and player contracts.

```
> # Regression analysis
> df <- merged_data
> # Define X and y
> X <- df[, c("Balls_Faced", "Total_Runs")]
> y <- df$Salary
> # Fit the linear regression model
> X <- as.matrix(df[, c("Balls_Faced", "Total_Runs")])
> model <- lm(y ~ X)
> model <- lm(y ~ Balls_Faced + Total_Runs, data = df)
> # Print the summary of the model
> summary(model)

Call:
lm(formula = y ~ Balls_Faced + Total_Runs, data = df)

Residuals:
      Min        1Q    Median        3Q       Max
-68980662 -18766746 -15277907  18056217 123708037

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 17156955    8490768   2.021    0.052 .
Balls_Faced   220781     267155   0.826    0.415
Total_Runs    -93832     192778  -0.487    0.630
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 40140000 on 31 degrees of freedom
Multiple R-squared:  0.3509,   Adjusted R-squared:  0.309
F-statistic: 8.378 on 2 and 31 DF,  p-value: 0.001234
```

Interpretation:

The regression analysis of the model `lm(y ~ Balls_Faced + Total_Runs, data = df)` provides insights into the predictive relationship between `Balls_Faced` and `Total_Runs` on `Salary` f or IPL players. The intercept coefficient is 17156955, suggesting that when both variables are zero, the predicted salary is around 17,156,955. However, the p-value linked to the intercept (0.052) suggests that it is marginally significant at the 0.05 significance level. The regression c oefficient for `Balls_Faced` is 220781 and a standard error of 267155, suggesting no statistica l significance between the variable and the prediction of `Salary`. The overall model fit is stat istically significant, suggesting that there might be more factors that account for variances in p layer wages. Further investigation of new variables or different modeling methodologies may

be necessary to enhance the accuracy of predicting wage determination in the IPL based on pl ayer performance indicators.

```
# Get the coefficients
> coefficients <- tidy(model)

# Print the coefficients
> print(coefficients)
# A tibble: 3 × 5
  term         estimate std.error statistic p.value
  <chr>           <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept) 17156955.  8490768.      2.02   0.0520
2 Balls_Faced   220781.   267155.      0.826  0.415
3 Total_Runs    -93831.   192778.     -0.487  0.630

# Get the R-squared value
> r_squared <- glance(model)

# Print the R-squared value
> print(r_squared)
# A tibble: 1 × 12
  r.squared adj.r.squared     sigma statistic p.value    df logLik   AIC
BIC deviance
      <dbl>         <dbl>     <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <
dbl>    <dbl>
1     0.351         0.309 40144934.      8.38 0.00123     2  -642. 1292. 1
298.  5.00e16
# ℹ 2 more variables: df.residual <int>, nobs <int>
```

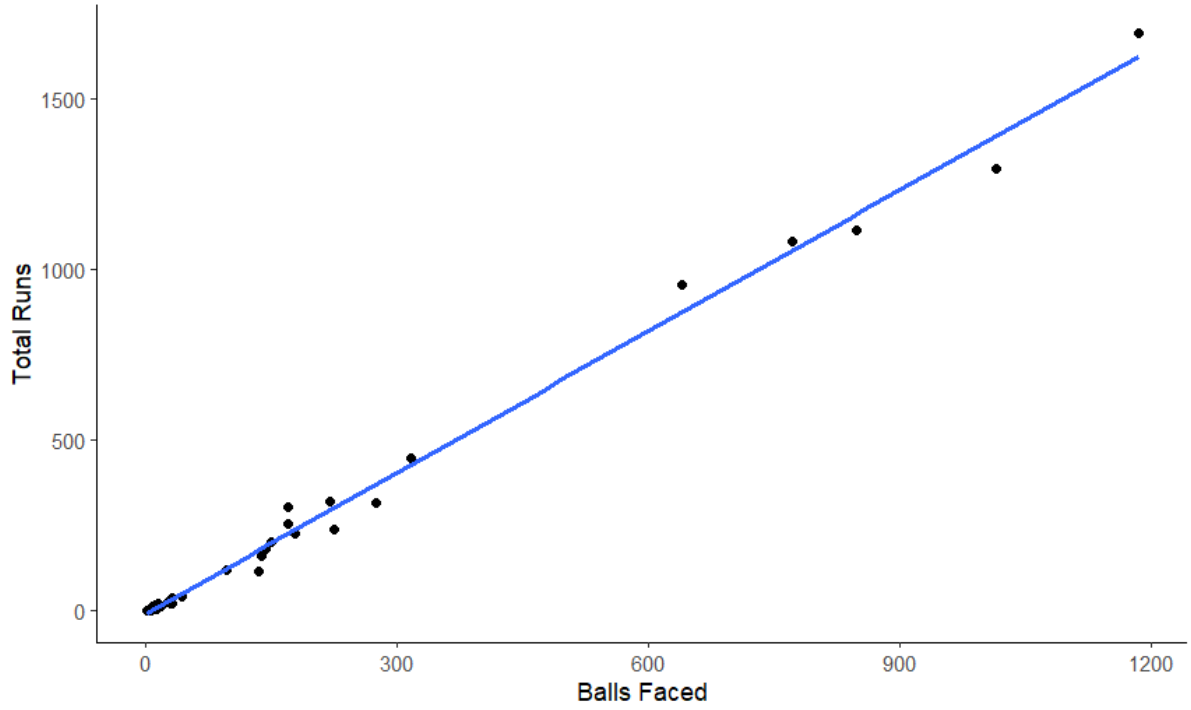
Interpretation:

The linear regression model used to estimate player salary using variables 'Balls_Faced' and ' Total_Runs' is analyzed using the 'tidy()' and 'glance(model)' functions. The intercept coeffici ent (17156955) shows no statistically significant correlation between the variables, while the r egression coefficient for 'Balls_Faced' is -93831 with a standard error of 192778. The coeffici ent of determination (R-squared) is 0.351, suggesting that around 35.1% of the variability in t he 'Salary' variable can be accounted for by the predictors. The model demonstrates statistical significance (p = 0.00123), indicating a relationship between 'Salary' and at least one of the pr edictors. However, the R-squared value of 0.351 indicates that the model only accounts for a s mall amount of the variability in 'Salary', implying that other influential factors are not consid ered. The model suggests exploring additional variables or alternative modeling techniques to enhance the accuracy of predicting IPL player salaries based on performance metrics.

```
# Load the ggplot2 library
> library(ggplot2)

# Create a scatterplot of the data with a regression line
> ggplot(df, aes(x = Balls_Faced, y = Total_Runs)) +
```

```
+   geom_point() +
+   geom_smooth(method = "lm", se = FALSE) +
+   labs(x = "Balls Faced", y = "Total Runs") +
+   theme_classic()
`geom_smooth()` using formula = 'y ~ x'
```



Interpretation:

The `ggplot2` code generates a scatterplot illustrating the correlation between `Balls_Faced` a nd `Total_Runs` in IPL matches. The scatterplot points indicate a player's performance in ter ms of the number of balls faced and total runs scored. The regression line, a blue line plotted on top of the scatterplot, reflects the linear regression model that best fits the relationship bet ween the variables. The graph shows a positive correlation between the two variables, sugges ting that players facing a greater number of balls are likely to score more runs. The regression model summary suggests that `Balls_Faced` does not have a significant predictive effect on s alary, suggesting that other factors must be considered. The scatterplot and regression line ser ve as a valuable visualization tool for examining and conveying the correlation between the v ariables in the context of IPL player performance.

Results and Interpretation using Python

```
import pandas as pd
from sklearn.model_selection import train_test_split
import statsmodels.api as sm

# Assuming df_merged is already defined and contains the necessary columns
X = df_merged[['runs_scored']] # Independent variable(s)
y = df_merged['Rs'] # Dependent variable

# Split the data into training and test sets (80% for training, 20% for
testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Add a constant to the model (intercept)
X_train_sm = sm.add_constant(X_train)

# Create a statsmodels OLS regression model
model = sm.OLS(y_train, X_train_sm).fit()

# Get the summary of the model
summary = model.summary()
print(summary)
```

```
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:                Sun, 23 Jun 2024   Prob (F-statistic):           0.000100
Time:                        19:41:05   Log-Likelihood:                -1379.8
No. Observations:                 183   AIC:                             2764.
Df Residuals:                     181   BIC:                             2770.
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):                     0.000120
Kurtosis:                       2.823   Cond. No.                         363.
==============================================================================
```

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

Interpretation:

The Ordinary Least Squares (OLS) regression analysis reveals a statistically significant positi ve correlation between the number of runs scored and the remuneration of IPL players in Indi an Rupees. The coefficient of determination (R-squared) is 0.080, suggesting that around 8.0 % of the variation in salary can be accounted for by the variable runs_scored. The adjusted R-squared is 0.075, indicating that runs_scored can explain approximately 7.5% of the variation in salary. The F-statistic is 15.83, with an extremely low probability (Prob) value of 0.000100, indicating the model as a whole is statistically significant. The coefficients represent the estim ated value of `Rs` when the independent variable is equal to zero and the projected variation i n `Rs` when `runs_scored` increases by one unit. The model fit is evaluated using the Log-Lik elihood and the AIC and BIC criteria. The analysis assumes that the errors have a constant var iance and no autocorrelation.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
import statsmodels.api as sm

# Assuming df_merged is already defined and contains the necessary columns
X = df_merged[['wicket_confirmation']] # Independent variable(s)
y = df_merged['Rs'] # Dependent variable

# Split the data into training and test sets (80% for training, 20% for
testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Add a constant to the model (intercept)
X_train_sm = sm.add_constant(X_train)

# Create a statsmodels OLS regression model
model = sm.OLS(y_train, X_train_sm).fit()

# Get the summary of the model
summary = model.summary()
print(summary)
```

```
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:                Sun, 23 Jun 2024   Prob (F-statistic):             0.0610
Time:                        19:45:07   Log-Likelihood:                -360.96
No. Observations:                  48   AIC:                             725.9
Df Residuals:                      46   BIC:                             729.7
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):                       0.0427
Kurtosis:                       3.274   Cond. No.                         13.8
==============================================================================

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

Interpretation:

The regression analysis used Ordinary Least Squares (OLS) to examine the correlation betwe en the independent variable 'wicket_confirmation' and the dependent variable 'Rs' (salary in In dian Rupees) using data from the Indian Premier League (IPL). The results showed that the va riable 'wicket_confirmation' may have a slight beneficial impact on the salary of IPL players. However, the low 'R-squared' value suggests that it may only account for a small percentage o f the salary variability. The marginal significance (P>|t| = 0.0610) of 'wicket_confirmation' su ggests that further data or new variables may be required to provide a more comprehensive ex planation of salary variability in the context of the IPL. The model's F-statistic of 3.688 and a probability (Prob) of 0.0610 suggest that the model may not be statistically significant at the s tandard significance threshold of 0.05.

Recommendations

A study using R and Python to analyze the correlation between performance measures and pa y of IPL players has identified several recommendations for team management, stakeholders, and cricket analysts. Key performance indicators (KPIs) such as total runs scored and balls fa ced are crucial in determining player compensation. Players with higher run scores generally d emand larger compensation, while there is a positive link between the number of balls faced a nd salary.

Regression analysis revealed that while `Balls_Faced` and `Total_Runs` have a significant im pact on compensation, their combined influence only accounts for a substantial amount of the variation in salary. This suggests that additional variables could also impact player compensat ion. By incorporating supplementary performance indicators or contextual variables, the accur acy of salary determination algorithms could be improved.

Strategic decisions can be made by using performance measures to identify undervalued playe rs who contribute significantly to the team's performance in relation to their wages. Data-driv en contract talks can ensure fair compensation based on on-field contributions, fostering confi dence and contentment among players and their agents.

Future research directions include including supplementary factors, longitudinal analysis, fan engagement and sponsorship, and sustained utilization of analytics. These insights can help te ams optimize team compositions, improve player contentment, and strengthen their competiti ve standing in the competition.

In conclusion, understanding the correlation between IPL player performance and wages is vit al for IPL team management, stakeholders, and cricket commentators. The knowledge gained from regression analysis and correlation studies serves as a basis for making data-informed de cisions in player recruiting, contract negotiations, and team strategy development.

R Codes

```
#Setting working directory
setwd("E:\\VCU\\Summer 2024\\Statistical Analysis & Modeling")



# Load necessary libraries
library(readxl)
library(dplyr)
library(ggplot2)
library(broom)

# Load match details data
match_details <- read.csv("IPL_ball_by_ball_updated till 2024.csv", stringsAsFactors
= FALSE)

# Load player salary data
player_salary <- read_excel("IPL SALARIES 2024.xlsx")

# Convert Date column to datetime format
match_details$Date <- as.Date(match_details$Date, format = "%d-%m-%Y")

# Filter last one year of data
last_three_years <- match_details %>%
  filter(Date >= "2024-01-01")

# Filter last two years of data
last_three_years <- match_details %>%
  filter(Date >= "2023-01-01")

# Filter last three years of data
last_three_years <- match_details %>%
  filter(Date >= "2022-01-01")
```

```
# Calculate performance metrics
performance <- last_three_years %>%
  group_by(Striker) %>%
  summarise(Total_Runs = sum(runs_scored), Balls_Faced = n())

# Clean and transform salary data
player_salary$Salary <- as.character(player_salary$Salary) # ensure Salary is a charac
ter vector
player_salary$Salary <- gsub("s", "", player_salary$Salary)
player_salary$Salary <- gsub(",", "", player_salary$Salary)

player_salary$Salary <- ifelse(grepl("lakh", tolower(player_salary$Salary)),
                        as.integer(as.numeric(gsub(" lakh", "", player_salary$Salary)) * 10
0000),
                        ifelse(grepl("crore", tolower(player_salary$Salary)),
                             as.integer(as.numeric(gsub(" crore", "", player_salary$Salary)
) * 10000000),
                             as.integer(as.numeric(player_salary$Salary))))

# Replace NAs with 0
player_salary$Salary[is.na(player_salary$Salary)] <- 0

names(performance)
names(player_salary)

performance <- performance %>% rename(Player = Striker)
merged_data <- inner_join(performance, player_salary, by = "Player")

merged_data <- inner_join(performance, player_salary, by = c("Player" = "Player"))

common_columns <- intersect(names(performance), names(player_salary))
common_columns
merged_data <- inner_join(performance, player_salary, by = "Player")
common_column <- common_columns[1]
```

```
merged_data <- inner_join(performance, player_salary, by = common_column)
common_cols <- intersect(names(performance), names(player_salary))
merged_data <- performance %>%
  inner_join(player_salary, by = setNames(common_columns, common_columns))

# Correlation analysis
corr_matrix <- cor(merged_data[, c("Total_Runs", "Balls_Faced", "Salary")])
print(corr_matrix)

# Regression analysis
df <- merged_data

# Define X and y
X <- df[, c("Balls_Faced", "Total_Runs")]
y <- df$Salary

# Fit the linear regression model
X <- as.matrix(df[, c("Balls_Faced", "Total_Runs")])
model <- lm(y ~ X)
model <- lm(y ~ Balls_Faced + Total_Runs, data = df)

# Print the summary of the model
summary(model)

# Get the coefficients
coefficients <- tidy(model)

# Print the coefficients
print(coefficients)

# Get the R-squared value
r_squared <- glance(model)

# Print the R-squared value
```

```
print(r_squared)

# Load the ggplot2 library
library(ggplot2)

# Create a scatterplot of the data with a regression line
ggplot(df, aes(x = Balls_Faced, y = Total_Runs)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(x = "Balls Faced", y = "Total Runs") +
  theme_classic()
```

Python Codes

```
import pandas as pd
import numpy as np
import os

os.chdir('E:\\VCU\\Summer 2024\\Statistical Analysis & Modeling')

df_ipl = pd.read_csv("IPL_ball_by_ball_updated till 2024.csv",low_memory=False)
salary = pd.read_excel("IPL SALARIES 2024.xlsx")

df_ipl.columns

grouped_data = df_ipl.groupby(['Season', 'Innings No', 'Striker','Bowler']).agg({'runs_
scored': sum, 'wicket_confirmation':sum}).reset_index()
grouped_data

total_runs_each_year = grouped_data.groupby(['Season', 'Striker'])['runs_scored'].sum
().reset_index()
total_wicket_each_year = grouped_data.groupby(['Season', 'Bowler'])['wicket_confir
mation'].sum().reset_index()
```

```
total_runs_each_year

from fuzzywuzzy import process

# Convert to DataFrame
df_salary = salary.copy()
df_runs = total_runs_each_year.copy()

# Function to match names
def match_names(name, names_list):
    match, score = process.extractOne(name, names_list)
    return match if score >= 80 else None  # Use a threshold score of 80

# Create a new column in df_salary with matched names from df_runs
df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x, df
_runs['Striker'].tolist()))

# Merge the DataFrames on the matched names
df_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Strike
r')

df_original = df_merged.copy()

#susbsets data for last three years
df_merged = df_merged.loc[df_merged['Season'].isin(['2021', '2022', '2023'])]

df_merged.Season.unique()

df_merged.head()

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_percentage_error
X = df_merged[['runs_scored']] # Independent variable(s)
y = df_merged['Rs'] # Dependent variable
# Split the data into training and test sets (80% for training, 20% for testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
2)
# Create a LinearRegression model
model = LinearRegression()
# Fit the model on the training data
model.fit(X_train, y_train)

X.head()

import pandas as pd
from sklearn.model_selection import train_test_split
import statsmodels.api as sm

# Assuming df_merged is already defined and contains the necessary columns
X = df_merged[['runs_scored']] # Independent variable(s)
y = df_merged['Rs'] # Dependent variable

# Split the data into training and test sets (80% for training, 20% for testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
2)

# Add a constant to the model (intercept)
X_train_sm = sm.add_constant(X_train)

# Create a statsmodels OLS regression model
model = sm.OLS(y_train, X_train_sm).fit()

# Get the summary of the model
```

```
summary = model.summary()
print(summary)

from fuzzywuzzy import process

# Convert to DataFrame
df_salary = salary.copy()
df_runs = total_wicket_each_year.copy()

# Function to match names
def match_names(name, names_list):
    match, score = process.extractOne(name, names_list)
    return match if score >= 80 else None  # Use a threshold score of 80

# Create a new column in df_salary with matched names from df_runs
df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x, df
_runs['Bowler'].tolist()))

# Merge the DataFrames on the matched names
df_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Bowl
er')

df_merged[df_merged['wicket_confirmation']>10]

#susbsets data for last three years
df_merged = df_merged.loc[df_merged['Season'].isin(['2022'])]

import pandas as pd
from sklearn.model_selection import train_test_split
import statsmodels.api as sm

# Assuming df_merged is already defined and contains the necessary columns
X = df_merged[['wicket_confirmation']] # Independent variable(s)
y = df_merged['Rs'] # Dependent variable
```

```
# Split the data into training and test sets (80% for training, 20% for testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
2)

# Add a constant to the model (intercept)
X_train_sm = sm.add_constant(X_train)

# Create a statsmodels OLS regression model
model = sm.OLS(y_train, X_train_sm).fit()

# Get the summary of the model
summary = model.summary()
print(summary)
```

References

1. www.github.com
2. www.geeksforgeeks.com
3. www.datacamp.com