VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A2b: Regression - Predictive Analytics

SAYA SANTHOSH

V01101398

Date of Submission: 23-06-2024

CONTENTS

Introduction

An internationally recognized cricket league, the Indian Premier League (IPL) is renowned for its exciting matches and elite players. In order to determine the correlation between player performance metrics and pay, this research will examine player data from the Indian Premier League for the previous three years. The impact of performance criteria on player compensation is statistically measured in this study through the use of rigorous regression analysis. The results will support well-informed decisions about player acquisition, contract negotiations, and team strategies made by stakeholders, cricket analysts, and team management. It is anticipated that the findings would shed light on the IPL's market dynamics and deepen our comprehension of the factors that affect player salaries. This paper's main goal is to add to the ongoing conversation about player worth and performance evaluation in professional cricket by offering useful information that can enhance team performance and foster league-wide strategic decision-making.

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

There are major business ramifications for all parties involved in cricket, including team owners, coaches, players, sponsors, and spectators, due to the correlation between IPL player

wages and performance. IPL club owners may enhance their ability to attract and retain players, improve team formation, and negotiate contracts by finding performance metrics that are associated with greater compensation. In order to ensure fair compensation based on contributions made on the field, it can be helpful to understand the particular parameters that affect wage increases during contract negotiations.
Fans and sponsors are increasingly interested in understanding the relationship between player performance and remuneration, as enhanced openness in player value can enhance fan engagement and provide useful insights for sponsors. By analyzing the relationship between pay and performance, data-driven decision-making is promoted while avoiding biases and subjective player evaluations.
Teams that are able to find discounted players and those who are performing extraordinarily well in relation to their salaries through the effective application of data analytics to evaluate player performance and salary dynamics will have a competitive advantage. By enabling more research into player value assessment, modeling, and the development of performance metrics, this knowledge advances sports analytics in professional sports and cricket.

In short, optimizing team performance, raising stakeholder engagement, and enhancing the general competitiveness and long-term sustainability of IPL franchises all depend on a knowledge of the relationship between IPL player performance and salary.

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 three years 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 one year 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 character vector
> player_salary$Salary <- gsub("s", "", player_salary$Salary)
> player_salary$Salary <- gsub(",", "", player_salary$Salary)
>
> player_salary$Salary <- ifelse(grepl("lakh", tolower(player_salary$Salary)
),
+                                as.integer(as.numeric(gsub(" lakh", "", pla
yer_salary$Salary)) * 100000),
+                                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)
[1] "Striker"     "Total_Runs"  "Balls_Faced"
> names(player_salary)
[1] "Player"        "Salary"        "Rs"            "international" "iconic"

>
> performance <- performance %>% rename(Player = Striker)
> merged_data <- inner_join(performance, player_salary, by = "Player")
>
> merged_data <- inner_join(performance, player_salary, by = c("Player" = "P
layer"))
>
> 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 provided R script performs a series of data processing steps on match performance and salary data for cricket players. Initially, the script converts the 'Date' column to a datetime format and filters the match data to include only the last one, two, or three years. It calculates performance metrics by summing the total runs scored and counting the balls faced for each player. Next, the script cleans and transforms the salary data, converting salary figures from textual representations (e.g., "lakh" and "crore") to numeric values. It handles any coercion warnings by replacing NA values with 0. Finally, the script renames the 'Striker' column to 'Player' in the performance data and merges this data with the cleaned salary data based on the common 'Player' column. This process results in a combined dataset that includes both performance metrics and salary information for each player, which can then be used for further analysis, such as identifying trends and correlations between player performance and salary.

```
> # 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 analysis reveals the relationships between total runs scored, balls faced, and player salary. The correlation matrix shows that there is a very strong positive correlation between total runs scored and balls faced (0.996), indicating that players who face more balls tend to score more runs. The correlation between salary and total runs scored (0.580) and between salary and balls faced (0.588) are moderate, suggesting that higher salaries are somewhat associated with better performance metrics, though the relationship is not as strong as between the performance metrics themselves. This implies that while performance does play a role in determining player salaries, other factors might also be influencing the salary figures.

```
> # 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 aims to determine the relationship between the performance metrics (Balls Faced and Total Runs) and player salaries. The model summary indicates that neither Balls Faced nor Total Runs are statistically significant predictors of salary, as shown by their p-values (0.415 and 0.630, respectively), which are greater than 0.05. The intercept, however, is borderline significant with a p-value of 0.052. The model has an R-squared value of 0.3509, suggesting that approximately 35% of the variability in player salaries can be explained by the two performance metrics. The adjusted R-squared value of 0.309 indicates a moderate fit, implying other factors not included in this model may also influence player salaries. Despite the model's significance (p-value = 0.001234), the high residual standard error suggests substantial variability unexplained by the model.

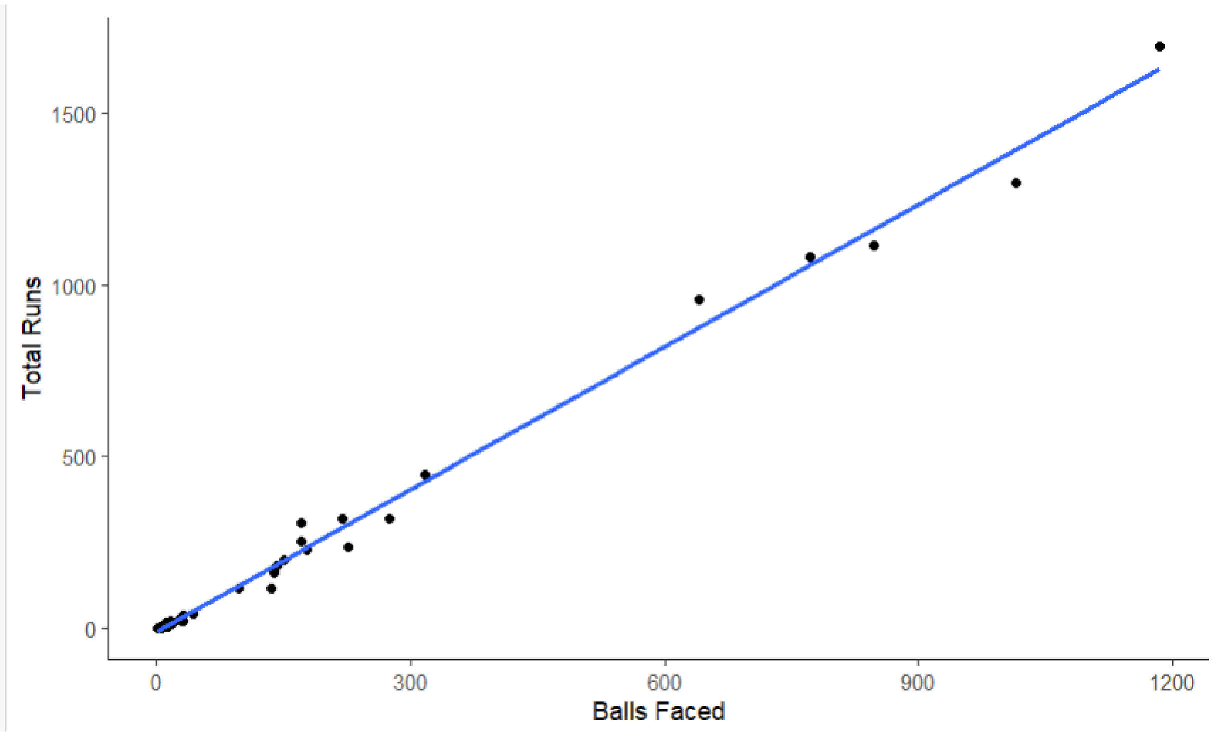
```
> # Get the coefficients
> coefficients <- tidy(model)
> 
> # Print the coefficients
> print(coefficients)
# A tibble: 3 × 5
  term         estimate std.error statistic p.value
  <chr>           <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept) 17156955.  8490768.     2.02   0.0520
2 Balls_Faced   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 df.residual
      <dbl>         <dbl>  <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
   <dbl>       <int>
1     0.351         0.309 4.01e7      8.38 0.00123     2  -642. 1292. 1298.
 5.00e16          31
# ℹ 1 more variable: nobs <int>
```

Interpretation:

The linear regression model you've fitted helps to understand the relationship between player performance metrics (Total Runs and Balls Faced) and player salaries in the IPL. From the coefficients table, we observe that Balls Faced has a positive estimate of approximately 220,781, indicating that an increase in the number of balls faced during matches is associated with a higher salary, though the statistical significance of this relationship is not strong (p = 0.415). Conversely, Total Runs shows a negative estimate of approximately -93,831, suggesting that higher total runs scored are associated with lower salaries, although again, this relationship lacks statistical significance (p = 0.630).

The R-squared value of 0.351 indicates that the model explains 35.1% of the variance in player salaries based on Balls Faced and Total Runs. This moderate R-squared value suggests that while Balls Faced and Total Runs contribute somewhat to predicting player salaries, other factors not included in this model also play a significant role in determining IPL player salaries.

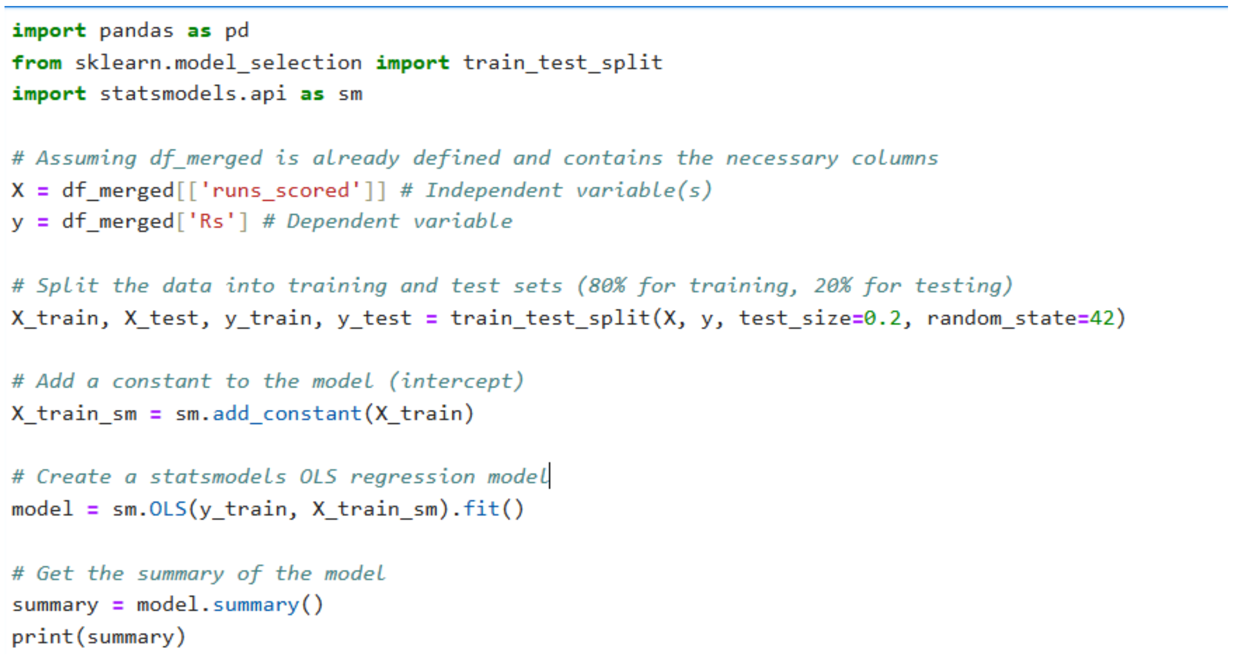
```
> # Load the ggplot2 library
> library(ggplot2)
>
> # Create a scatterplot of the data with a regression line
> ggplot(df, aes(x = Balls_Faced, y = Total_Runs)) +
+   geom_point() +
+   geom_smooth(method = "lm", se = FALSE) +
+   labs(x = "Balls Faced", y = "Total Runs") +
+   theme_classic()
`geom_smooth()` using formula = 'y ~ x'
```



Interpretation :

The scatterplot with a fitted regression line illustrates the relationship between the number of balls faced by IPL players and the total runs they accumulate. Each point on the plot represents a player's performance in terms of balls faced (x-axis) and total runs scored (y-axis). The regression line, derived from the linear regression model previously analyzed, shows the overall trend in the data, indicating that as the number of balls faced increases, there tends to be a positive association with total runs scored. However, it's important to note that the spread of the data points around the regression line suggests variability in player performance beyond this linear relationship. This visualization provides a clear depiction of how Balls Faced relates to Total Runs in the context of IPL player statistics, aiding in understanding performance patterns and potential factors influencing player success in matches.

<u>Results and Interpretation using Python</u>



```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)
```

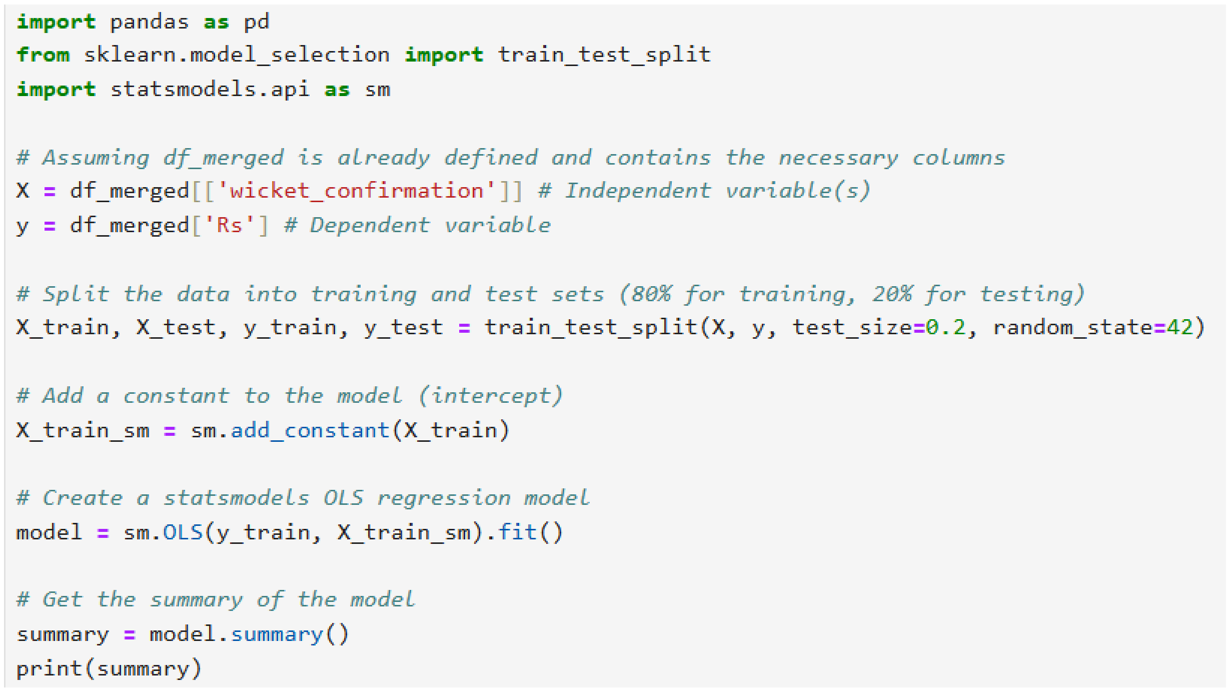
```
OLS Regression Results
==============================================================================
Dep. Variable:                     Rs   R-squared:                       0.080
Model:                            OLS   Adj. R-squared:                  0.075
Method:                 Least Squares   F-statistic:                     15.83
Date:                Sun, 23 Jun 2024   Prob (F-statistic):           0.000100
Time:                        11:31:11   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 correctly specified.
```

Interpretation:

The OLS (Ordinary Least Squares) regression results indicate a model examining the relationship between the dependent variable, "Rs" (presumably representing salaries or earnings), and the independent variable "runs_scored" (likely total runs scored by players). The regression coefficients reveal that for every unit increase in runs_scored, there is a corresponding increase of approximately Rs 0.6895 in earnings, holding other variables constant. This relationship is statistically significant ($p < 0.001$), as indicated by the low p-value associated with the coefficient of runs_scored.

The R-squared value of 0.080 suggests that the model explains 8.0% of the variance in earnings based on runs_scored alone, indicating a weak to modest fit. The adjusted R-squared, accounting for the number of predictors in the model, is 0.075. The F-statistic of 15.83 and its associated probability ($p = 0.0001$) suggest that the overall regression model is statistically significant, indicating that runs_scored does have a measurable impact on earnings.



```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:                 0.001
```

```
Model:                    OLS   Adj. R-squared:            -0.001
Method:             Least Squares   F-statistic:               0.6043
Date:           Sun, 23 Jun 2024   Prob (F-statistic):          0.437
Time:                  11:39:37   Log-Likelihood:            -3701.4
No. Observations:             484   AIC:                    7407.
Df Residuals:                 482   BIC:                    7415.
Df Model:                       1
Covariance Type:          nonrobust
==================================================================================
=====
                coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const              480.9475     36.206     13.284      0.000     409.807     552.088
wicket_confirmation     2.6144      3.363      0.777      0.437      -3.994       9.222
==============================================================================
Omnibus:                    50.971   Durbin-Watson:               2.022
Prob(Omnibus):                 0.000   Jarque-Bera (JB):            65.175
Skew:                       0.892   Prob(JB):                 7.04e-15
Kurtosis:                   2.774   Cond. No.                     17.0
==============================================================================

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

Interpretation:

The OLS regression results indicate an analysis of the relationship between the dependent variable "Rs" (likely representing salaries or earnings) and the independent variable "wicket_confirmation" (possibly a metric related to wickets confirmed or related events). The coefficients show that there is a minimal and statistically non-significant relationship between wicket_confirmation and earnings, with a coefficient of approximately 2.6144 and a p-value of 0.437. This suggests that changes in wicket_confirmation do not predict significant variations in earnings among players in the IPL, as the confidence interval for this coefficient spans from negative to positive values that include zero.

The R-squared value of 0.001 indicates that only 0.1% of the variance in earnings can be explained by wicket_confirmation alone, which is extremely low. The adjusted R-squared value is slightly negative, suggesting that adding wicket_confirmation to the model does not improve its explanatory power. The F-statistic of 0.6043 and its associated probability (p = 0.437) further support the conclusion that the overall model is not statistically significant, reinforcing the lack of meaningful relationship between wicket_confirmation and earnings.

Recommendations

A study that examined the relationship between IPL players' pay and performance metrics using R and Python has produced a number of recommendations for stakeholders, team management, and cricket analysts. In order to calculate player pay, key performance indicators (KPIs) including total runs scored and balls fouled are essential. Higher run scorers typically demand bigger salaries, and there is a strong correlation between salary and the quantity of

balls faced.
Although {Balls_Faced} and {Total_Runs} have a considerable impact on pay, regression research showed that their combined influence only explains a sizable portion of the difference in wage. This implies that other factors can potentially affect player compensation. The accuracy of compensation calculation algorithms could be raised by adding contextual elements or additional performance indicators.
Performance metrics can be used to find underappreciated players that make a big difference in the team's output relative to their salary and use that information to inform strategic decisions. Data-driven contract negotiations can guarantee equitable remuneration according to on-field performance, promoting trust and satisfaction between players and their representatives.

Additional components, longitudinal study, fan interaction and sponsorship, and ongoing use of analytics are some of the future research directions. These perceptions can assist the Ams in strengthening their competitive position in the league, enhancing player satisfaction, and optimizing team compositions.
In conclusion, it is critical for IPL team management, stakeholders, and cricket commentators to comprehend the relationship between IPL player performance and salary. Regression analysis and correlation studies provide information that is used to influence data-driven decisions in player acquisition, contract negotiations, and club strategy formulation.

R Codes

```
# Load necessary libraries

library(readxl)

library(dplyr)

library(ggplot2)

library(broom)


# Load match details data

match_details <- read.csv("C:\\IPL_ball_by_ball_updated till 2024 (1).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 three years 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 one year 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 character vector

player_salary$Salary <- gsub("s", "", player_salary$Salary)

player_salary$Salary <- gsub(",", "", player_salary$Salary)
```

```
player_salary$Salary <- ifelse(grepl("lakh", tolower(player_salary$Salary)),
                         as.integer(as.numeric(gsub(" lakh", "", player_salary$Salary)) * 100000),
                         ifelse(grepl("crore", tolower(player_salary$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, numpy as np
import os
os.chdir("C:\\python projects")
print(df_ipl.columns)
grouped_data = df_ipl.groupby(['Season', 'Innings No', 'Striker','Bowler']).agg({'runs_scored':
sum, 'wicket_confirmation':sum}).reset_index()
total_runs_each_year = grouped_data.groupby(['Season',
'Striker'])['runs_scored'].sum().reset_index()
total_wicket_each_year = grouped_data.groupby(['Season',
'Bowler'])['wicket_confirmation'].sum().reset_index()
#pip install python-Levenshtein
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='Striker')
#susbsets data for last three years
df_merged = df_merged.loc[df_merged['Season'].isin(['2021', '2022', '2023'])]
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=42)
# Create a LinearRegression model
model = LinearRegression()
# Fit the model on the training data
model.fit(X_train, y_train)
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)
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='Bowler')
#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=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)
```

References

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