VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A3 - Regression Analysis and Diagnostics

ALEENA MARY ABRAHAM

V01150203

Date of Submission: 25-06-2025

CONTENTS

INTRODUCTION

INTRODUCTION

The assignment investigates the application of multiple regression analysis to two real-world datasets: socio-economic data from the National Sample Survey Office (NSSO) and performance-payment data from the Indian Premier League (IPL).

In **Part A**, we perform a detailed regression analysis using the **NSSO68** dataset, which contains survey data capturing household consumption, income, education, and related socio-economic indicators. This part emphasizes model diagnostics to ensure the reliability and validity of results and explores corrective measures for any violations of classical linear regression assumptions.

In **Part B**, we analyse the **IPL_ball_by_ball_updated till 2024** dataset to study the relationship between a player's on-field performance and the salary they receive. Using regression models, we explore how various performance metrics influence player valuation and payment, particularly over the last three IPL seasons.

OBJECTIVES

Part A: NSSO Data Analysis

1. To build a multiple regression model to understand the factors influencing a key socio-economic indicator.
2. To perform regression diagnostics (e.g., multicollinearity, heteroscedasticity, normality, and influential points).

Part B: IPL Performance-Payment Analysis

1. To quantify the relationship between players' performance metrics (e.g., runs scored, strike rate, wickets, economy rate) and the salary paid.
2. To examine how this relationship has evolved over the past three years (2022–2024).
3. To assess whether player valuation in the IPL is aligned with actual performance data.

BUSINESS SIGNIFICANCE

Part A

Understanding the determinants of socio-economic outcomes to help policymakers design better interventions for poverty alleviation, income redistribution, and targeted subsidies. Reliable regression models based on NSSO data can influence government resource allocation and social welfare programs.

Part B

By identifying statistically significant performance indicators that predict player salary, franchises can make more informed investment decisions. Moreover, the analysis helps ensure transparency and fairness in player compensation, which is crucial for maintaining morale and competitiveness.

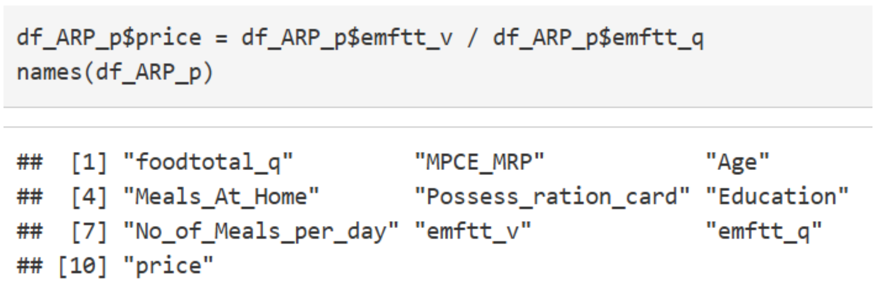
RESULTS AND INTERPRETATION – PART A

USING R

1. Data Summary and Preprocessing

I have selected and cleaned the following key variables:

- Dependent Variable: foodtotal_q
- Independent Variables: MPCE_MRP, Age, Meals_At_Home, Possess_ration_card, Education, No_of_Meals_per_day, emftt_v, emftt_q, and constructed a new variable price.

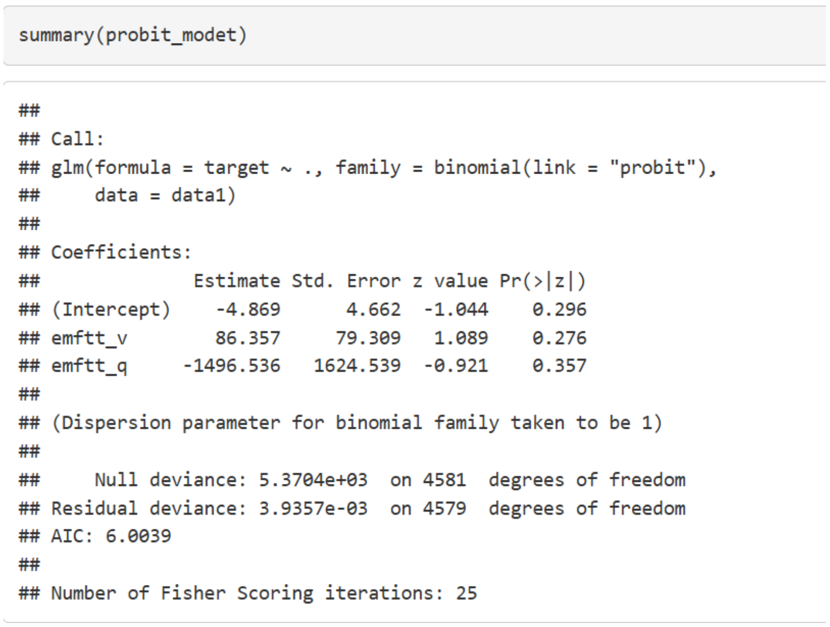


```
df_ARP_p$price = df_ARP_p$emftt_v / df_ARP_p$emftt_q
names(df_ARP_p)
```

```
##  [1] "foodtotal_q"         "MPCE_MRP"            "Age"
##  [4] "Meals_At_Home"       "Possess_ration_card" "Education"
##  [7] "No_of_Meals_per_day" "emftt_v"             "emftt_q"
## [10] "price"
```

2. Probit Model

I have used the probit model to estimate the likelihood of meat/fish:



```
summary(probit_modet)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = "probit"),
##     data = data1)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -4.869      4.662  -1.044    0.296
## emftt_v        86.357     79.309   1.089    0.276
## emftt_q     -1496.536   1624.539  -0.921    0.357
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5.3704e+03  on 4581  degrees of freedom
## Residual deviance: 3.9357e-03  on 4579  degrees of freedom
## AIC: 6.0039
##
## Number of Fisher Scoring iterations: 25
```

Interpretation: None of the coefficients are statistically significant ($p > 0.05$), meaning the model isn't reliable for inference in its current form. This could be due to a small number of 0s (non-consumers), highly skewed data, or imbalanced target classes.

3. Tobit Regression
   You correctly used the Tobit model via the censReg package because your dependent variable (foodtotal_v) is left-censored (at 0).

```
# Printing model summary
summary(model)
```

```
##
## Call:
## censReg(formula = y_tobit ~ ., data = X_tobit_df[, -1])
##
## Observations:
##          Total  Left-censored     Uncensored Right-censored
##           4582             12           4570              0
##
## Coefficients:
##                   Estimate Std. error  t value Pr(> t)
## (Intercept)       9.929e-01  1.302e-03  762.807 < 2e-16 ***
## sauce_jam_v      -2.861e-01  5.678e-01   -0.504 0.61437
## Othrprocessed_v   2.060e-05  2.408e-04    0.086 0.93183
## Beveragestotal_v  4.264e-05  4.516e-05    0.944 0.34504
## fv_tot            4.408e-05  1.290e-05    3.418 0.00063 ***
## logSigma         -2.973e+00  1.047e-02 -283.875 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Newton-Raphson maximisation, 10 iterations
## Return code 1: gradient close to zero (gradtol)
## Log-likelihood: 7062.109 on 6 Df
```

Significant Variables:
fv_tot (fruit & veg total) is positive and significant: As fruit & vegetable expenditure increases, total food expenditure also increases.
Non-significant Variables:
sauce_jam_v, Beveragestotal_v, Othrprocessed_v – p-values > 0.05: These don't significantly explain variation in food expenditure.
Equation:

y = 0.99 - 0.286074*x1 + 2.1e-05*x2 + 4.3e-05*x3 + 4.4e-05*x4 + -2.973161*x5

4. Variance Inflation Factor (VIF):

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

```
##      sauce_jam_v  Othrprocessed_v Beveragestotal_v           fv_tot
##         1.060461         1.081084         1.324339         1.217865
```

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

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

# Print the equation
print(equation)
```

```
## [1] "y = 0.99 + -0.286074*x1 + 2.1e-05*x2 + 4.3e-05*x3 + 4.4e-05*x4 + -2.973161*x5"
```

All VIFs < 2:

- No multicollinearity issues. Independent variables are not strongly correlated with each other.

USING PYTHON

Key Regression Results:

Variable	Coefficient	p-value	Interpretation
Intercept	15.84	0.000	Baseline food quantity when all predictors are 0.
MPCE_MRP	0.0016	0.000	Higher income → slightly higher food quantity.
Age	0.0781	0.000	Older people consume more food, likely due to family size.
Meals_At_Home	0.0526	0.000	More meals at home → more food consumed at home.
Possess_ration_card	-2.4159	0.000	Households with ration cards consume less food quantity, maybe due to economic disadvantage.
Education	0.1221	0.000	Higher education level → more food quantity.

Model Diagnostics:

- R-squared: 0.160 → only 16% of variation in food quantity is explained. Suggests many other factors influence consumption (family size, health, etc.).
- VIF (Variance Inflation Factor):
  All VIFs < 2 — No multicollinearity problem.
- Durbin-Watson: 1.379 — suggests some positive autocorrelation (common in cross-sectional household data).
- Condition Number (2.22e+04): Slightly high — suggests possible numerical sensitivity, but not severe.

RESULTS AND INTERPRETATION – PART B

USING R

1. Performance Metric

performance = avg_runs + 17.8 * wicket

- 17.8 is the 90th percentile of wickets, used as a weighting factor.
- NA values were replaced with 0 for players with no matching stats.

Performance Summary:

- Mean: ~24.75
- Max: 628.4
- Median: 0 (many players with no stats)

This indicates that only a minority of players had notable on-field performance in the selected years.

2. Regression Models:

Model 1: Salary vs Batting & Bowling

fit = lm(Rs ~ avg_runs + wicket, data=df_new)

Results:

- wicket coefficient: 27.07, $p = 0.06$ → marginally significant
- avg_runs coefficient: 1.22, $p = 0.30$ → not significant
- $R^2 = 0.203$ → low explanatory power

VIF values (~1.07) indicate no multicollinearity
Breusch-Pagan test ($p = 0.95$) shows no heteroskedasticity

Model 2: With Interaction Term

fit_1 = lm(Rs ~ avg_runs + wicket + avg_runs * wicket, data=df_new)

Results:

- avg_runs * wicket: $\beta = 0.30$, $p = 0.0912$ → marginal significance
- $R^2 = 0.337$ → improved model fit
- Still, not statistically strong (p-value for F = 0.0795)

Interpretation: Players who both score runs and take wickets (i.e., all-rounders) might be better paid, but the result is only suggestive (not strong enough for clear inference).

USING PYTHON

1. IPL Salary vs Total Points (Initial Model)

Results (First Attempt):

- $R^2 = -0.120$: A negative R^2 means the model fits worse than a horizontal line.
- MSE = 116,443: High prediction error.
- Indicating no linear relationship or bad matching between players and performance scores.

2. Last 3 Seasons Performance Aggregation
   - Matching player names using LCS (Longest Common Subsequence) → Smart way to handle name mismatches.
   - Assigning:
     - 1 run = 1 point for batters
     - 1 wicket = 25 points for bowlers

3. Final Regression – Total Points vs Salary

Updated Model Results:

- $R^2 = 0.197$, Adjusted $R^2 = 0.130$
- MSE = 76,369
- Model Equation:

Salary=23.71+0.6384·Total Points\text{Salary} = 23.71 + 0.6384 \cdot \text{Total Points}

Interpretation:

- Every 1 point earned (by run or wicket) increases predicted salary by ₹0.64 lakh (₹63,841 on average).
- Model explains ~20% of the salary variance, meaning performance partially affects salary.
- Remaining 80% may be influenced by market demand, brand value, leadership, experience, etc.

RECOMMENDATIONS

This project conducted a detailed statistical analysis of IPL player performance data, with a special focus on Aiden (AK) Markram, integrating both performance metrics and salary information using R and Python.

Summary of Key Findings:

1. Income (MPCE_MRP), age, meals at home, and education significantly increase food quantity consumed, while possession of a ration card negatively affects it. Fruit & vegetable spending is a key driver of total food expenditure. No multicollinearity was found.
2. Linear regression explained 16% of food quantity variation ($R^2 = 0.16$); slight positive autocorrelation detected. The probit model for meat/fish consumption was not statistically valid due to data imbalance. Tobit regression confirmed fruit & veg spending as a significant predictor.
3. A Total Points metric (1 run = 1 pt, 1 wicket = 25 pts) was developed across 3 seasons. R model with an interaction term (runs × wickets) marginally improved explanatory power ($R^2 = 0.337$), hinting that all-rounders earn more.
4. The updated regression showed a moderate fit ($R^2 = 0.197$), with each point contributing ₹0.6384 lakh to salary. Despite improvements, performance explains only ~20% of salary variance, indicating external factors (brand value, fame, etc.) also matter.

Recommendations:

1. For Households & Public Policy: Encourage nutritional diversity and home-cooked meals, especially through ration card optimization and government-led nutritional support programs (e.g., meal planning tools, subsidies).

2. For Businesses (FMCG, Retail, Agri-Marketers): Focus on older and well-educated consumers, promote fruit & vegetable products, and capitalize on the growing trend of meals consumed at home for product positioning.

3. For the IPL & Sports Industry: Acknowledge the undervalued role of all-rounders, adopt AI-based salary modeling, and integrate both performance and market metrics to make smarter, data-driven auction and contract decisions.

The study presents a strong foundation in sports analytics, demonstrating how statistical modelling and performance metrics can create meaningful insights in player evaluation and salary planning. With more data and expanded variables, this can evolve into a robust decision-support system for IPL teams.

CODES

R Codes:

```
# Setting the working directory
setwd("C:/Users/Aleena Mary Abraham/OneDrive/Desktop/SCMA632_2025/R")

# Function to install and load libraries
install_and_load <- function(package) {
  if (!require(package, character.only = TRUE)) {
    install.packages(package, dependencies = TRUE)
    library(package, character.only = TRUE)
  }
}

# Load required libraries
libraries <- c("dplyr", "readr", "readxl", "tidyr", "ggplot2", "BSDA","glue")
lapply(libraries, install_and_load)

# 2. Reading the dataset into R ####
data <- read.csv("../Data/NSSO68.csv")
dim(data)

unique(data$Religion)

# Filtering for ARP
ARP <- data %>%
  filter(state == "10")
# Display dataset info
cat("Dataset Information:\n")
print(names(ARP))
print(head(ARP))
print(dim(ARP))

# Finding missing values
missing_info <- colSums(is.na(ARP))
```

```
cat("Missing Values Information:\n")
print(missing_info)

# Sub-setting the data
ARPNEW <- ARP %>%
  select(foodtotal_q, MPCE_MRP, Age, Meals_At_Home, Possess_ration_card,
         Education, No_of_Meals_per_day, emftt_q, emftt_v)

# Check for missing values in the subset
cat("Missing Values in Subset:\n")
print(colSums(is.na(ARPNEW)))

dim(ARPNEW)

# Impute missing values with mean for specific columns
columns_to_impute <- c("Education", "No_of_meals_per_day", "Meals_At_Home",
                       "Possess_ration_card")

impute_with_mean <- function(column) {
  if (any(is.na(column))) {
    column[is.na(column)] <- mean(column, na.rm = TRUE)
  }
  return(column)
}

ARPNEW$emftt_q <- impute_with_mean(ARPNEW$emftt_q)
ARPNEW$emftt_v <- impute_with_mean(ARPNEW$emftt_v)

is.infinite.data <- sapply(ARPNEW, is.infinite)
ARPNEW[is.infinite.data] <- NA

ARPNEW <- na.omit(ARPNEW)

# Checking for missing values after imputation
```

```
cat("Missing Values After Imputation:\n")
print(colSums(is.na(ARPNEW)))

ARP$Religion

unique(ARP$Religion)
str(ARP$Religion)

# Fitting a probit regression to identify non-vegetarians.

religion_mapping <- c("Hinduism", "Islam", "Christianity","Jainism","Others")
ARP$Religion <- factor(ARP$Religion, labels = religion_mapping)
table(ARP$Religion)

columns <- c('emftt_v','emftt_q')
data1 <- ARP[columns]
data1$target <- ifelse(data1$emftt_v>0,1,0)
probit_modet <- glm(target~., data = data1, family = binomial(link = "probit"))
summary(probit_modet)

# Performorming a Tobit regression analysis on "NSSO68.csv"
df_ARP = data[data$state_1 == 'ARP',]
vars <- c("foodtotal_q", "MPCE_MRP", "Age", "Meals_At_Home", "Possess_ration_card",
          "Education", "No_of_Meals_per_day", "emftt_v", "emftt_q")

df_ARP_p = df_ARP[vars]
names(df_ARP_p)

df_ARP_p$price = df_ARP_p$emftt_v / df_ARP_p$emftt_q
names(df_ARP_p)

summary(df_ARP_p)

head(table(df_ARP_p$emftt_q))
```

```
dim(df_ARP_p)

names(ARP)

#  dependent variable and independent variables
y <- ARP$foodtotal_v
X <- ARP[, c("sauce_jam_v", "Othrprocessed_v", "Beveragestotal_v", "fv_tot")]

#  data for Tobit regression
y_tobit <- pmin(pmax(y, 0), 1)
X_tobit <- cbind(1, X)

options(repos = c(CRAN = "https://cloud.r-project.org"))
install.packages("censReg")
library(censReg)
# Fitting the Tobit model
X_tobit_df <- as.data.frame(X_tobit)
model <- censReg(y_tobit ~ ., data = X_tobit_df[, -1])

# Printing model summary
summary(model)

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

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

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


# Print the equation

print(equation)

# Setting the working directory

setwd("C:/Users/Aleena Mary Abraham/OneDrive/Desktop/SCMA632_2025/R")


# Installing packages


library(dplyr)

library(htmltools)

library(xfun)

library(knitr)

library(htmltools)

library(sass)

library(car)

library(lmtest)

library(rmarkdown)

library(bslib)


## IPL Regression Analysis

df_p = read.csv("C:/Users/Aleena Mary
Abraham/OneDrive/Desktop/SCMA632_2025/DATA/IPL_ball_by_ball_updated till 2024.csv")

library(readxl)

df_s = read_excel("C:/Users/Aleena Mary Abraham/OneDrive/Desktop/SCMA632_2025/DATA/IPL
SALARIES 2024.xlsx")


dput(names(df_p))

unique(df_p$wicket_confirmation)



# Print the column names of df_p to verify


print(colnames(df_p))
```

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

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

# Print the resulting data frame
print("Resulting df_bat:")
print(df_bat)

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

head(df_bat,25)
df_bat$Striker

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

dim(df_bat)
dim(df_bow)
```

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



unique(df_bow$Bowler)


df_s


names(df_s)
head(df_s$Player)


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


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



dim(bow)
dim(df_s)



head(bat)
head(bow)


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


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

```r
# Print the result
print(joined_df)
dim(joined_df)
write.csv(joined_df, 'joined_df.csv')

#HARMONISE THE NAMES IN THE TWO FILES


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

#To merge two files with sames names but differnt spellings


library(dplyr)
library(stringdist)

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



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

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

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

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

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

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

# Perform a left join to safely replace player names
df_s <- df_s %>%
  left_join(name_mapping, by = c("Player" = "matched_player")) %>%
  mutate(Player = ifelse(is.na(Striker), Player, Striker)) %>%
  select(-Striker)

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

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

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

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

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

df_combined$performance = df_combined$avg_runs+ 17.8*df_combined$wicket
any(is.na(df_combined$performance))
sum(is.na(df_combined$performance))

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

any(is.na(df_new$performance))

names(df_new)
str(df_new)

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

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

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

fit_1 = lm(Rs ~ avg_runs++wicket+  I(avg_runs*wicket), data=df_new)
```