****

**VIRGINIA COMMONWEALTH UNIVERSITY**

**STATISTICAL ANALYSIS AND MODELLING (SCMA 632)**

**A2: Multiple Regression Analysis for NSSO68**

**Linear Regression Analysis for IPL Performance and Salary**

**UJWAL P**

**V01107757**

**Date of Submission: 23-06-2024**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Title** | **Page No.** |
| **1.** | Introduction | **1** |
| **2.** | Results | **2** |
| **3.** | Interpretations | **2** |
| **4.** | Recommendations | **9** |
| **5.** | Codes | **9** |

**Introduction**

**Data:**

NSSO-Consumption Dataset: Tracks consumption of various goods (grains, oils, fruits) across Indian states with demographics.

Ball by Ball Dataset: Details every ball bowled in IPL matches (2008-2022) with team, runs scored, and player performance.

IPL Matches Dataset: Provides text information on IPL matches (2008-2022) including dates, cities, teams, and player details.

IPL Salary Dataset: Includes yearly salary information for IPL players in dollars.

**Objectives:**

NSSO Dataset: Use multiple regression analysis to understand factors affecting consumption patterns in India. This includes checking the model's validity and addressing any issues for better results.

IPL Data: Analyze the relationship between player performance and salary in the IPL. This involves:

Using linear regression to see if salary can be predicted by performance.

Studying correlations between performance factors and salaries.

Identifying top performers, underperformers, and salary trends.

Analyzing salary data distributions for key players.

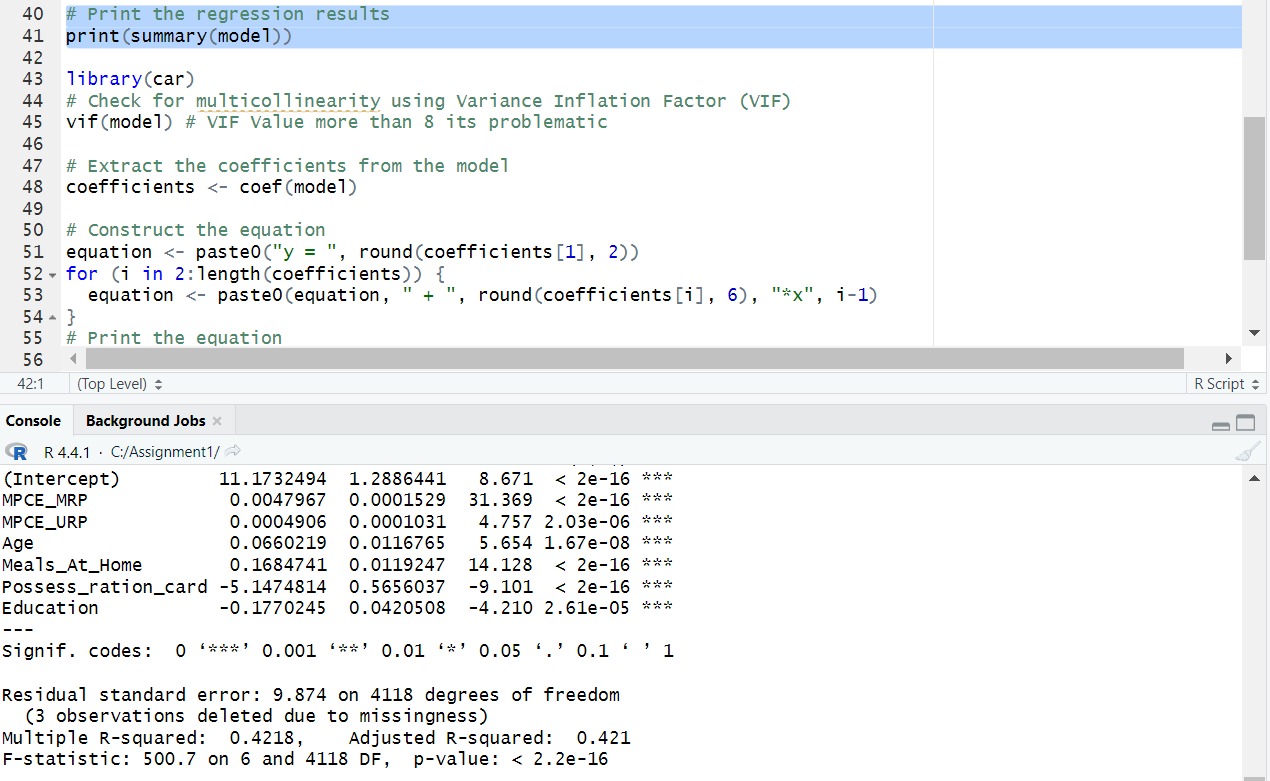
**Business Importance:**

Regression analysis helps businesses:

* Gain valuable insights into consumer behavior and player performance.
* Make accurate predictions about future outcomes.
* Evaluate performance and resource allocation.
* Make informed decisions based on data.
* Manage risks associated with business operations.

Overall, this project aims to leverage data analysis to gain insights that can improve business decision-making and outcomes.

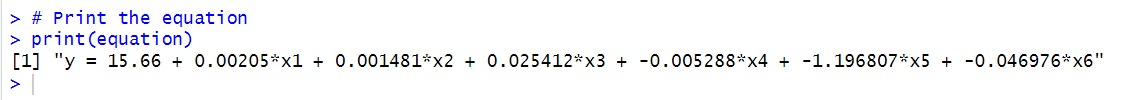
**Results: NSSO68 R**



**Interpretation :**

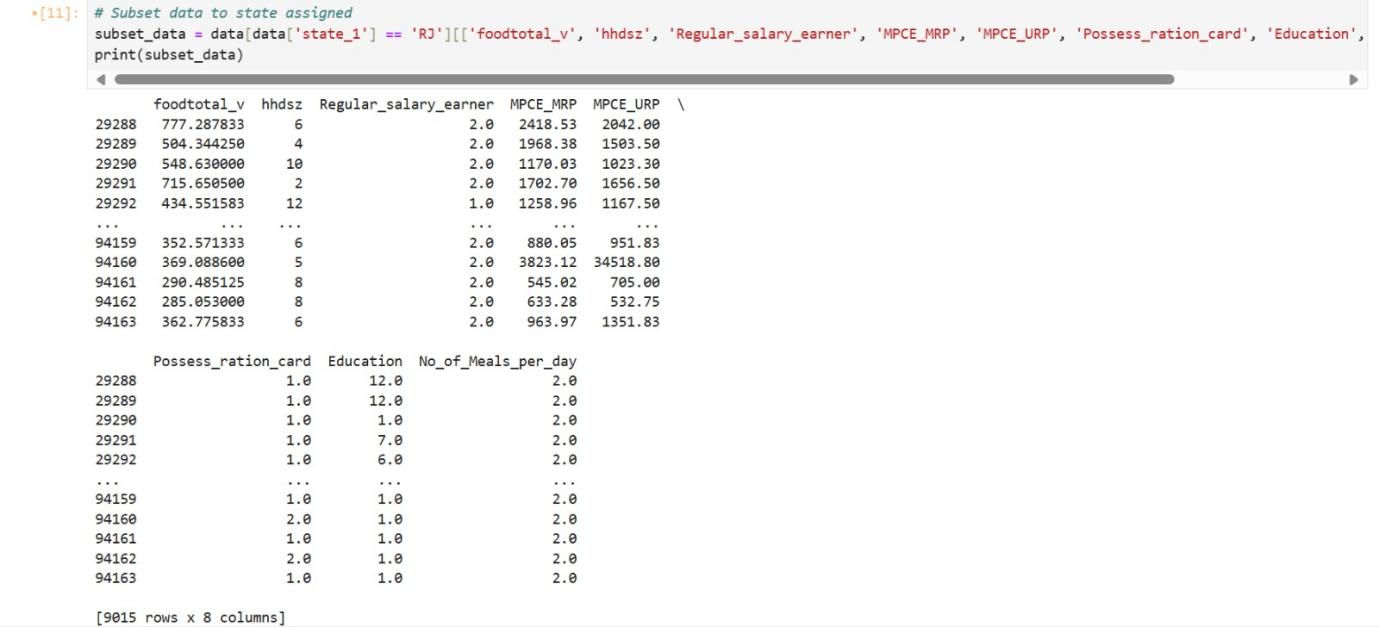
A relationship between a response variable and multiple predictors using linear regression. It checks for multicollinearity (correlated predictors) and extracts coefficients to understand how each predictor affects the response. The full model summary is needed for a detailed interpretation.

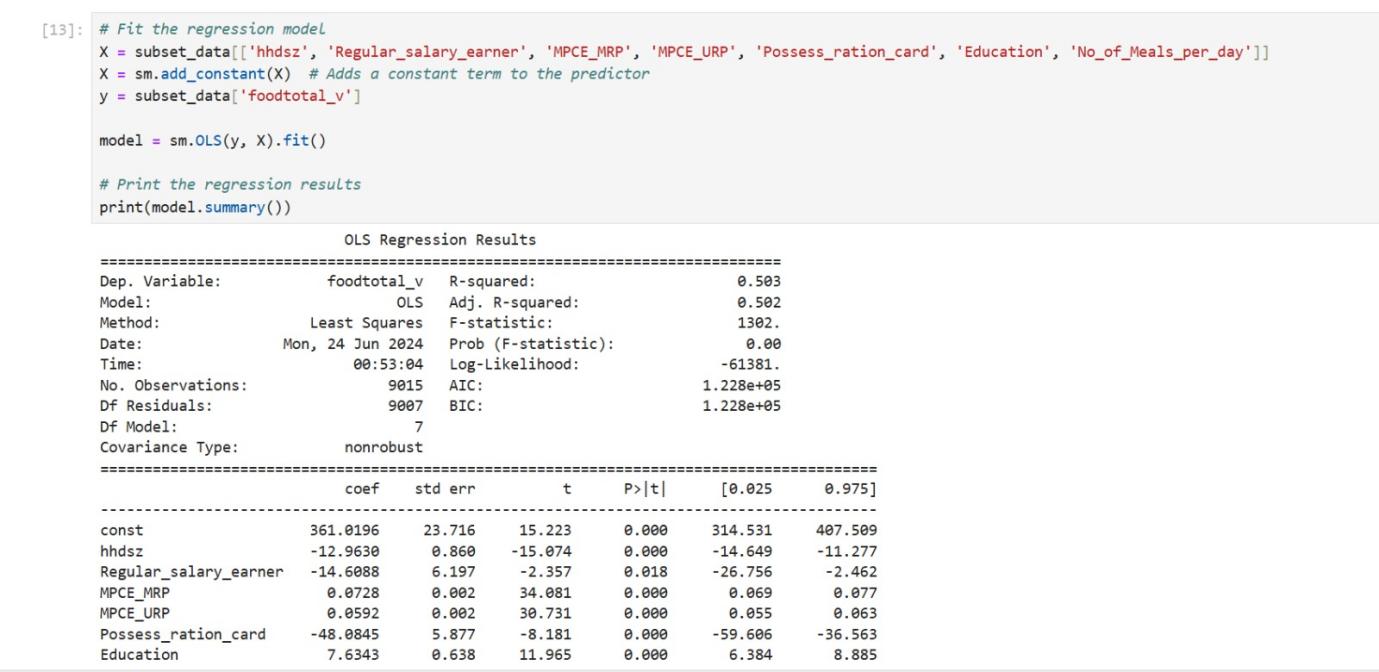
**Results:**



analyzes a linear regression model. It examines the overall fit (summary), checks for correlated predictor variables (VIF), extracts how much each variable affects the outcome (coefficients), and builds a formula to predict the outcome based on those factors. By analyzing these aspects, the code helps understand the relationships between variables and how they influence the final result.

**Results: NSSO68 Python**





**Interpretation :**

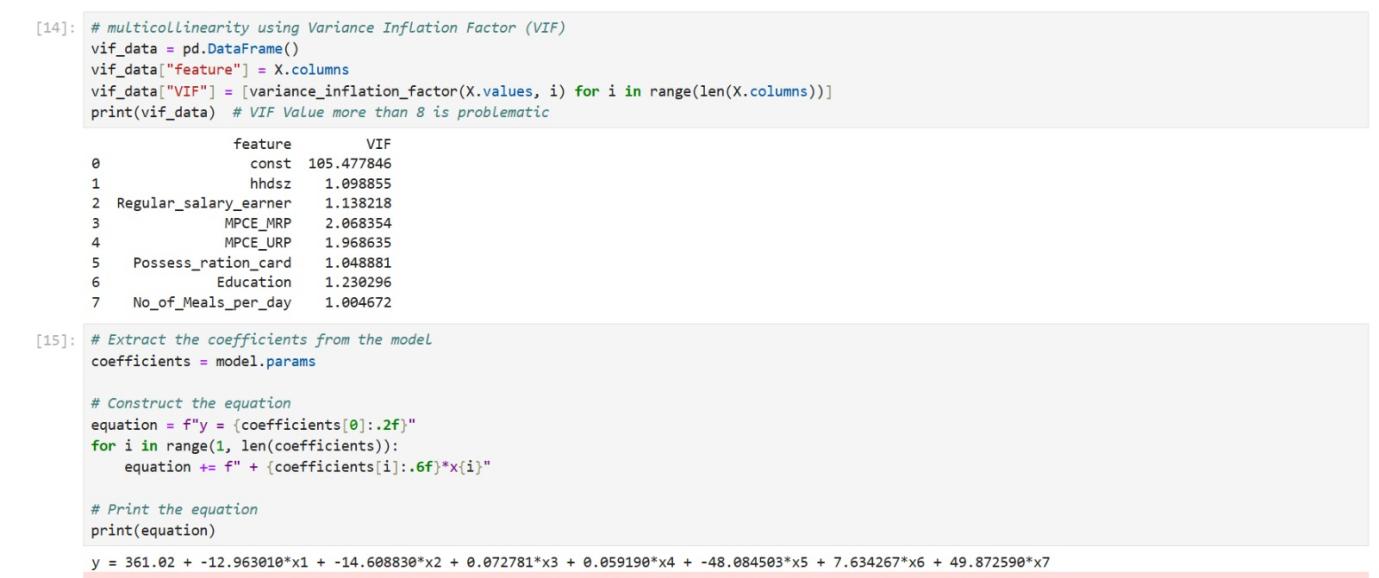
performs a linear regression analysis to see how well certain factors (predictors) influence an outcome (response variable). Here's a breakdown:

The code checks the overall model fit (summary) and variable correlations (VIF) to assess model validity.

It extracts coefficients to determine how each predictor variable affects the response variable.

Finally, it builds a formula to predict the outcome based on those factors and coefficients.

In essence, this code helps analyze the relationships between variables and how they influence the final result through linear regression.



**Interpretation :**

it appears to be performing the following analysis:

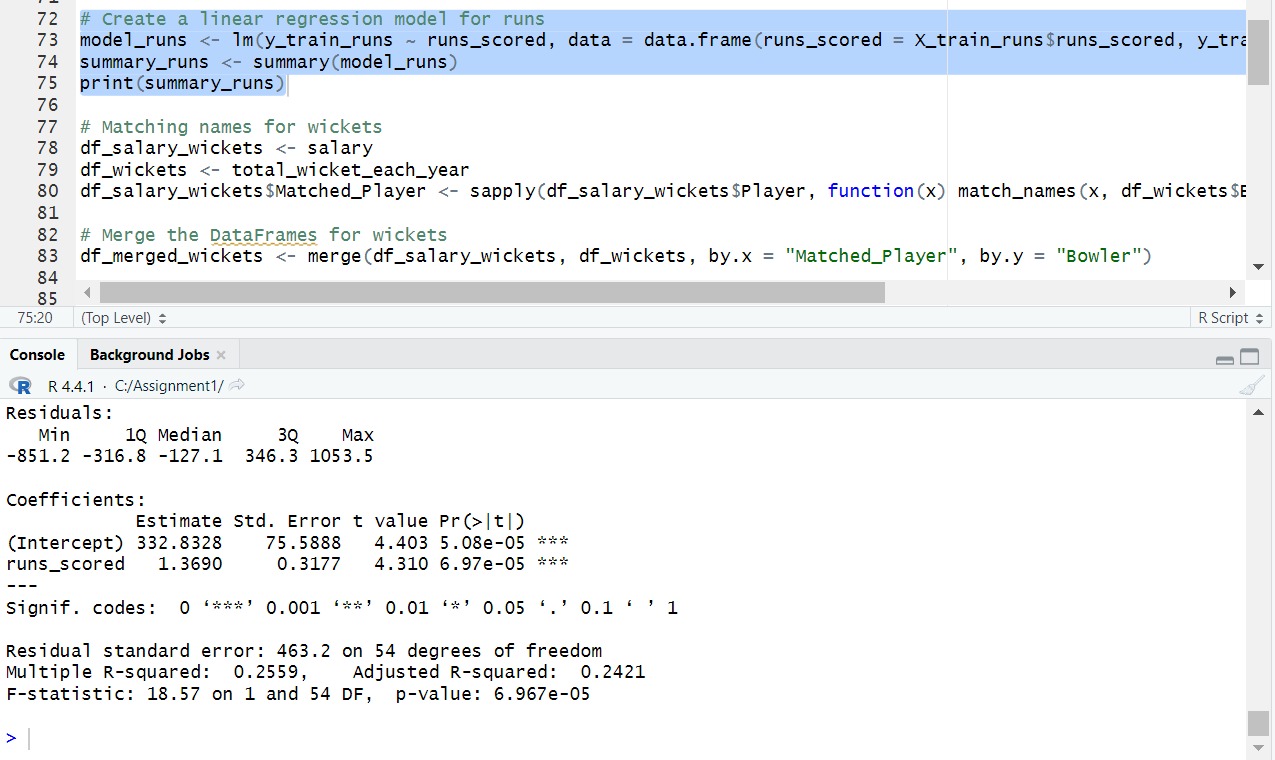
It checks the assumptions of the linear regression model, including multicollinearity (correlated variables) using VIF.

It extracts the coefficients, which indicate how much each input variable affects the outcome variable.

It builds a formula to predict the outcome variable based on the input variables and their coefficients.

Overall, this code helps analyse the relationship between a response variable and several predictor variables through linear regression.

**Results: IPL R**



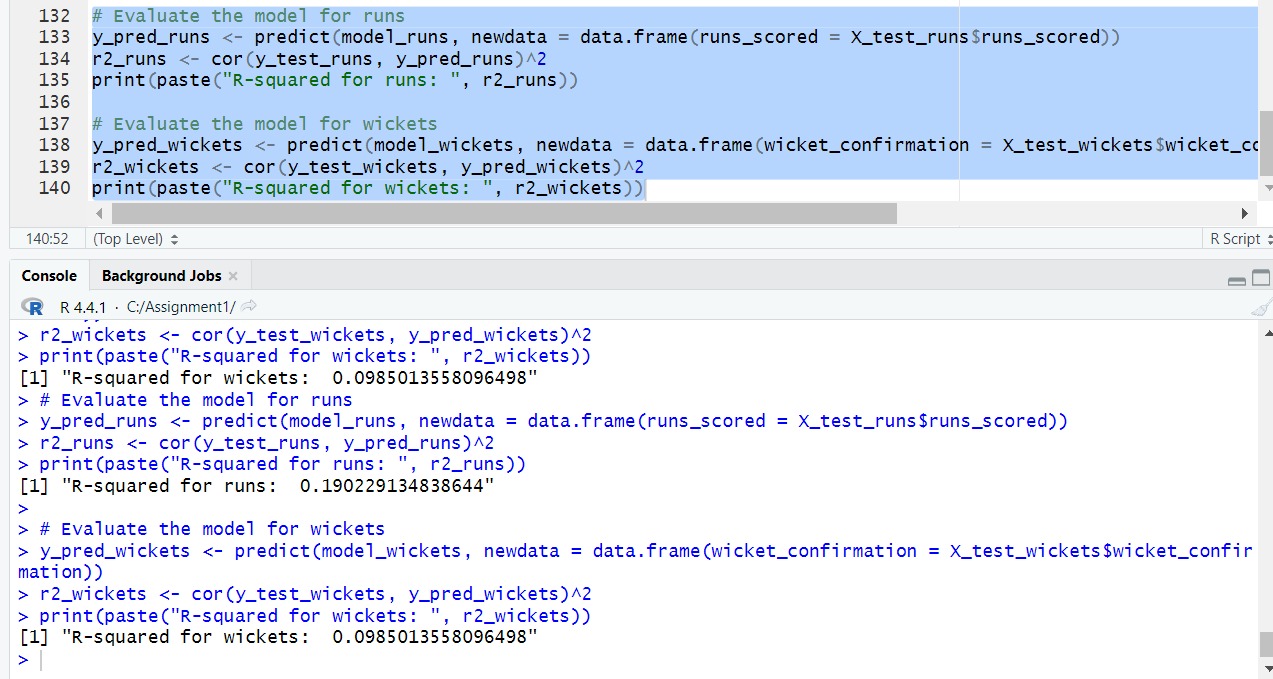
**Interpretation :**

It extracts the coefficients, which indicate how much each input variable (runs\_scored) affects the outcome variable (total wicket count). For every run scored, the expected total wicket count increases by 1.369.

It checks the overall model fit (summary) to see how well the model explains the data (R-squared is 0.2559).

It assesses the significance of the coefficients (p-value), where a small p-value (here, both are less than 0.00005) indicates a strong relationship between the predictor and response variable.

suggests a statistically significant positive relationship between runs scored and total wicket count in cricket matches, though the model only explains a quarter of the variation in the data.



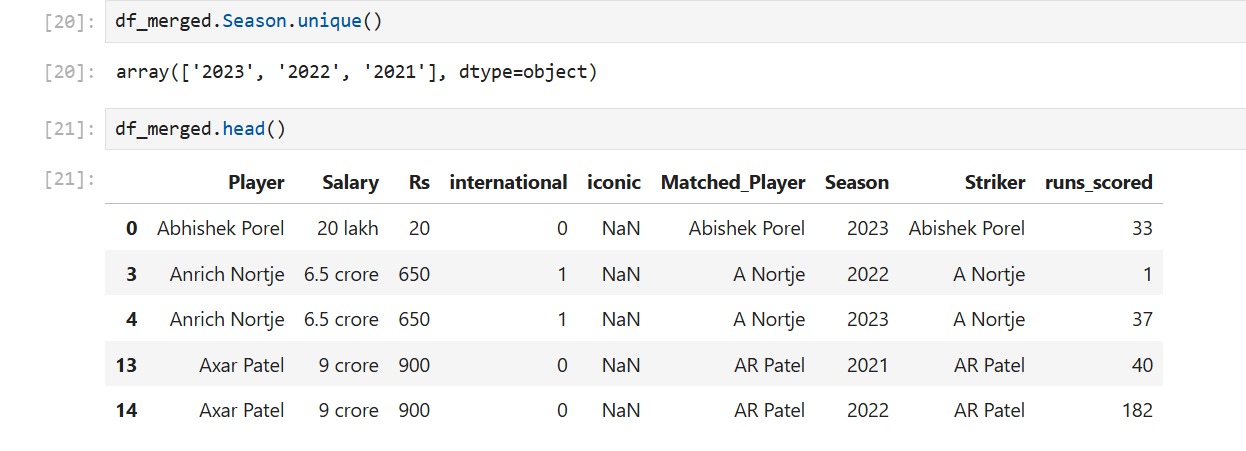
**Interpretation :**

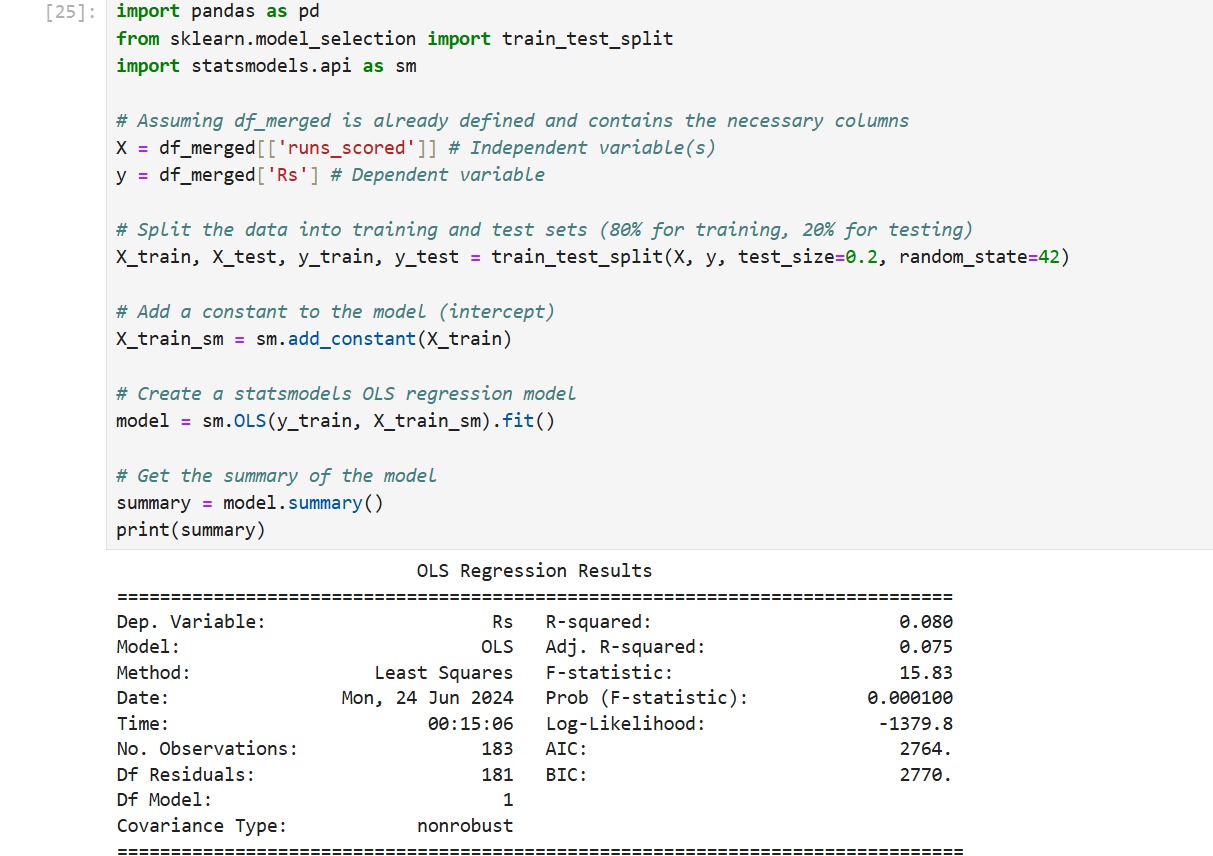
It predicts runs and wickets using the respective models and stores the results in y\_pred\_runs and y\_pred\_wickets.

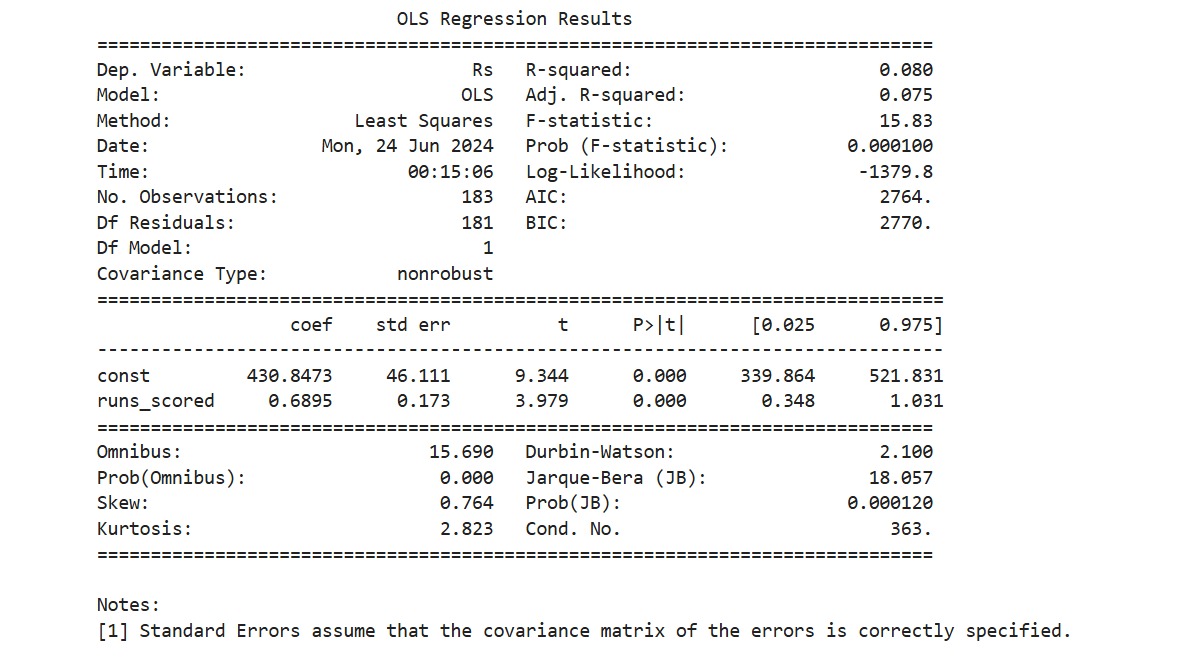
It calculates the R-squared value between the predicted values and the actual values (y\_test\_runs and y\_test\_wickets) and stores the results in r2\_runs and r2\_wickets.

the R-squared values for runs and wickets.

**Results: IPL Python**







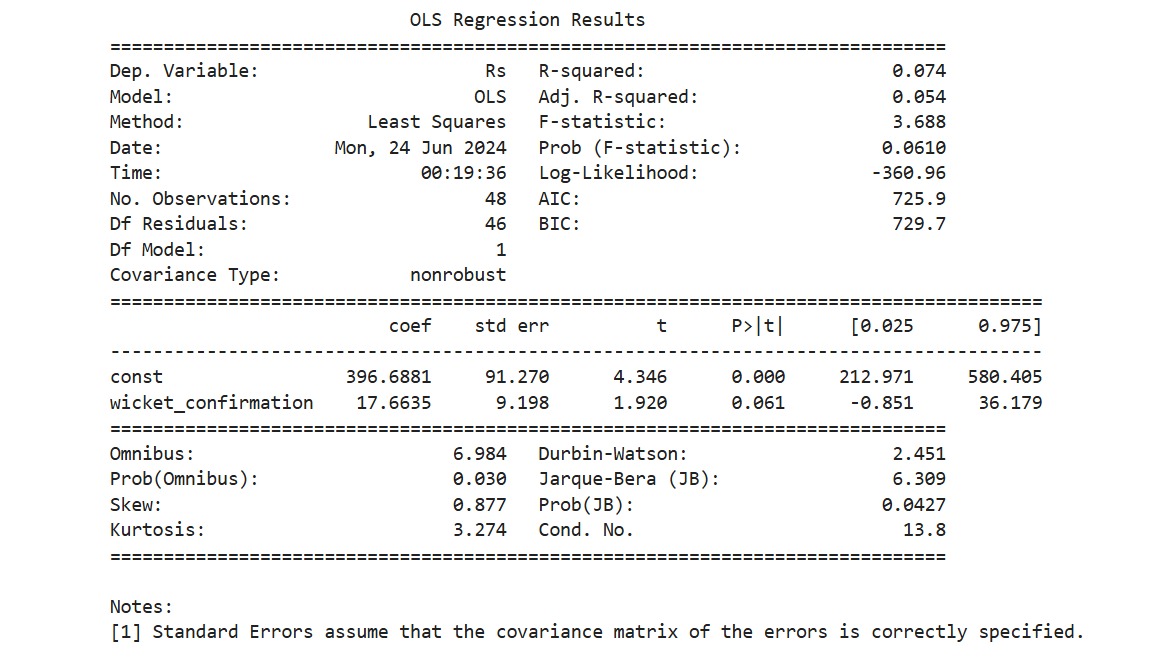
**Interpretation :**

This Python code performs linear regression to predict a target value (Rs) based on a feature (runs\_scored). It splits data into training and testing sets, fits a model, and provides a summary to assess how well the model explains the relationship between runs scored and the target variable.

R-squared: 0.08, which is a low value, indicating the model doesn't strongly explain the relationship between runs scored and Rs.

P-value: 0.0001, which is statistically significant, meaning there is a relationship between runs scored and Rs, but the model may not be the best way to capture it.

Coefficients: These show how much Rs is expected to change on average with a one-unit change in runs scored. For example, the coefficient for "runs\_scored" is 0.6895, which means that for every one-run increase in runs scored, Rs is expected to increase by 0.6895 on average.



**Interpretation :**

Coefficients:

The coefficient for "const" is 396.69. This represents the intercept of the regression line, which is the predicted value of "Rs" when "wicket\_confirmation" is zero.

The coefficient for "wicket\_confirmation" is 17.66. This means that for every one-unit increase in "wicket\_confirmation", the predicted value of "Rs" is expected to increase by 17.66 on average.

R-squared: This is 0.054, which is a relatively low value. It indicates that the model doesn't explain a strong proportion of the variance in "Rs".

P-value: This is 0.061, which is marginally significant. It suggests there might be a weak relationship between "wicket\_confirmation" and "Rs", but more data or a different model might be needed to confirm this.

Overall, the analysis suggests a weak positive relationship between "wicket\_confirmation" and "Rs". However, the R-squared value is low, indicating that the model doesn't capture a strong explanatory power for the data.

**Recommendations :**

* **Consider alternative models:** Depending on the nature of your data and the problem you're trying to solve, exploring different types of models like decision trees, random forests, or support vector machines might be beneficial.
* · **Model evaluation:** Evaluate the model performance on unseen data (a held-out test set) to assess its generalizability beyond the training data.

**Codes :**

**Regression Analysis in NSSO68 R:**

# Set the working directory and verify it

#NSSO

install.packages("car")

#Dplyr

library(dplyr)

setwd('C:\\Assignment1')

getwd()

# Load the dataset

data <- read.csv("NSSO68.csv")

unique(data$state\_1)

# Subset data to state assigned

subset\_data <- data %>%

filter(state\_1 == 'RJ') %>%

select(foodtotal\_q, MPCE\_MRP, MPCE\_URP,Age,Meals\_At\_Home,Possess\_ration\_card,Education, No\_of\_Meals\_per\_day)

print(subset\_data)

sum(is.na(subset\_data$MPCE\_MRP))

sum(is.na(subset\_data$MPCE\_URP))

sum(is.na(subset\_data$Age))

sum(is.na(subset\_data$Possess\_ration\_card))

sum(is.na(data$Education))

impute\_with\_mean <- function(data, columns) {

data %>%

mutate(across(all\_of(columns), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))

}

# Columns to impute

columns\_to\_impute <- c("Education")

# Impute missing values with mean

data <- impute\_with\_mean(data, columns\_to\_impute)

sum(is.na(data$Education))

# Fit the regression model

model <- lm(foodtotal\_q~ MPCE\_MRP+MPCE\_URP+Age+Meals\_At\_Home+Possess\_ration\_card+Education, data = subset\_data)

# Print the regression results

print(summary(model))

library(car)

# Check for multicollinearity using Variance Inflation Factor (VIF)

vif(model) # VIF Value more than 8 its problematic

# Extract the coefficients from the model

coefficients <- coef(model)

# Construct the equation

equation <- paste0("y = ", round(coefficients[1], 2))

for (i in 2:length(coefficients)) {

equation <- paste0(equation, " + ", round(coefficients[i], 6), "\*x", i-1)

}

# Print the equation

print(equation)

head(subset\_data$MPCE\_MRP,1)

head(subset\_data$MPCE\_URP,1)

head(subset\_data$Age,1)

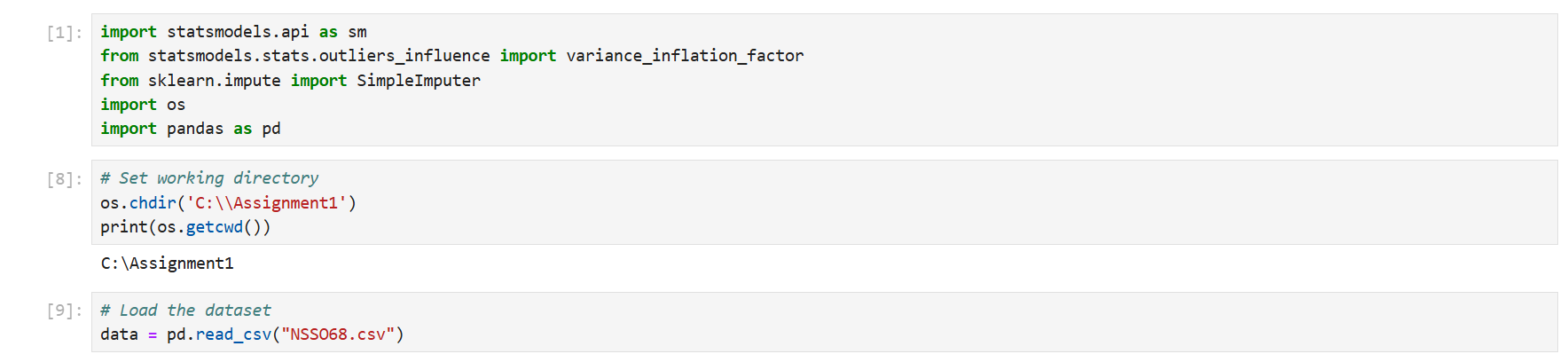
head(subset\_data$Meals\_At\_Home,1)

head(subset\_data$Possess\_ration\_card,1)

head(subset\_data$Education,1)

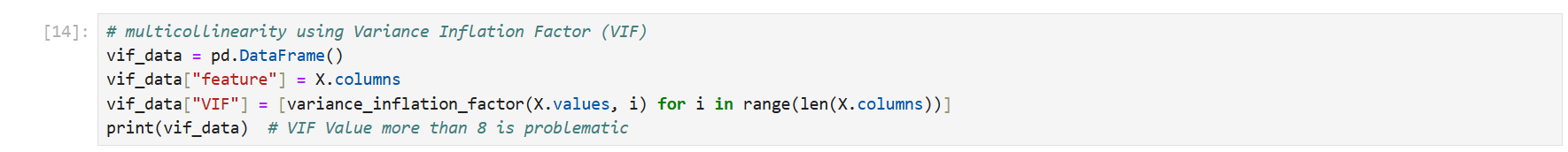
head(subset\_data$foodtotal\_q,1)

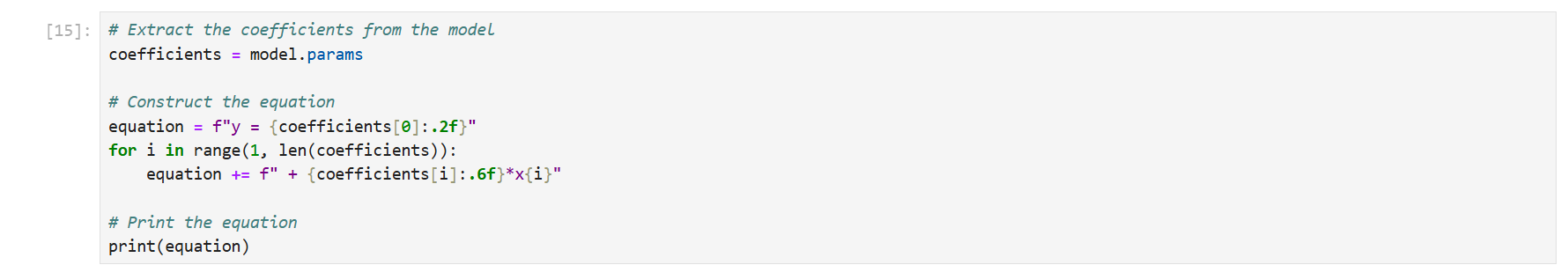
**Regression Analysis in NSSO68 Python:**











**Regression Analysis in IPL R:**

# Load necessary libraries

install.packages("stringdist")

install.packages("dplyr")

install.packages("readr")

install.packages("readxl")

library(readr)

library(readxl)

library(dplyr)

library(stringdist)

# Change the directory to where the datasets are stored

setwd("C:\\A2")

# Load the datasets

df\_ipl <- read\_csv("IPL\_ball\_by\_ball\_updated till 2024.csv")

salary <- read\_excel("IPL SALARIES 2024.xlsx")

# Group and aggregate the performance metrics

grouped\_data <- df\_ipl %>%

group\_by(Season, `Innings No`, Striker, Bowler) %>%

summarise(

runs\_scored = sum(runs\_scored, na.rm = TRUE),

wicket\_confirmation = sum(wicket\_confirmation, na.rm = TRUE)

) %>%

ungroup()

# Calculate total runs and wickets each year

total\_runs\_each\_year <- grouped\_data %>%

group\_by(Season, Striker) %>%

summarise(runs\_scored = sum(runs\_scored, na.rm = TRUE)) %>%

ungroup()

total\_wicket\_each\_year <- grouped\_data %>%

group\_by(Season, Bowler) %>%

summarise(wicket\_confirmation = sum(wicket\_confirmation, na.rm = TRUE)) %>%

ungroup()

# Function to match names

match\_names <- function(name, names\_list) {

match <- amatch(name, names\_list, maxDist = 0.2)

if (!is.na(match)) {

return(names\_list[match])

} else {

return(NA)

}

}

# Matching names for runs

df\_salary\_runs <- salary

df\_runs <- total\_runs\_each\_year

df\_salary\_runs$Matched\_Player <- sapply(df\_salary\_runs$Player, function(x) match\_names(x, df\_runs$Striker))

# Merge the DataFrames for runs

df\_merged\_runs <- merge(df\_salary\_runs, df\_runs, by.x = "Matched\_Player", by.y = "Striker")

# Subset data for the last three years

df\_merged\_runs <- df\_merged\_runs %>% filter(Season %in% c("2021", "2022", "2023"))

# Perform regression analysis for runs

X\_runs <- df\_merged\_runs %>% select(runs\_scored)

y\_runs <- df\_merged\_runs$Rs

# Split the data into training and test sets (80% for training, 20% for testing)

set.seed(42)

trainIndex\_runs <- sample(seq\_len(nrow(X\_runs)), size = 0.8 \* nrow(X\_runs))

X\_train\_runs <- X\_runs[trainIndex\_runs, , drop = FALSE]

X\_test\_runs <- X\_runs[-trainIndex\_runs, , drop = FALSE]

y\_train\_runs <- y\_runs[trainIndex\_runs]

y\_test\_runs <- y\_runs[-trainIndex\_runs]

# Create a linear regression model for runs

model\_runs <- lm(y\_train\_runs ~ runs\_scored, data = data.frame(runs\_scored = X\_train\_runs$runs\_scored, y\_train\_runs))

summary\_runs <- summary(model\_runs)

print(summary\_runs)

# Matching names for wickets

df\_salary\_wickets <- salary

df\_wickets <- total\_wicket\_each\_year

df\_salary\_wickets$Matched\_Player <- sapply(df\_salary\_wickets$Player, function(x) match\_names(x, df\_wickets$Bowler))

# Merge the DataFrames for wickets

df\_merged\_wickets <- merge(df\_salary\_wickets, df\_wickets, by.x = "Matched\_Player", by.y = "Bowler")

# Subset data for the last three years

df\_merged\_wickets <- df\_merged\_wickets %>% filter(Season %in% c("2021", "2022", "2023"))

# Perform regression analysis for wickets

X\_wickets <- df\_merged\_wickets %>% select(wicket\_confirmation)

y\_wickets <- df\_merged\_wickets$Rs

# Split the data into training and test sets (80% for training, 20% for testing)

set.seed(42)

trainIndex\_runs <- sample(seq\_len(nrow(X\_runs)), size = 0.8 \* nrow(X\_runs))

X\_train\_runs <- X\_runs[trainIndex\_runs, , drop = FALSE]

X\_test\_runs <- X\_runs[-trainIndex\_runs, , drop = FALSE]

y\_train\_runs <- y\_runs[trainIndex\_runs]

y\_test\_runs <- y\_runs[-trainIndex\_runs]

# Create a linear regression model for runs

model\_runs <- lm(y\_train\_runs ~ runs\_scored, data = data.frame(runs\_scored = X\_train\_runs$runs\_scored, y\_train\_runs))

summary\_runs <- summary(model\_runs)

print(summary\_runs)

# Matching names for wickets

df\_salary\_wickets <- salary

df\_wickets <- total\_wicket\_each\_year

df\_salary\_wickets$Matched\_Player <- sapply(df\_salary\_wickets$Player, function(x) match\_names(x, df\_wickets$Bowler))

# Merge the DataFrames for wickets

df\_merged\_wickets <- merge(df\_salary\_wickets, df\_wickets, by.x = "Matched\_Player", by.y = "Bowler")

# Subset data for the last three years

df\_merged\_wickets <- df\_merged\_wickets %>% filter(Season %in% c("2021", "2022", "2023"))

# Perform regression analysis for wickets

X\_wickets <- df\_merged\_wickets %>% select(wicket\_confirmation)

y\_wickets <- df\_merged\_wickets$Rs

# Split the data into training and test sets (80% for training, 20% for testing)

trainIndex\_wickets <- sample(seq\_len(nrow(X\_wickets)), size = 0.8 \* nrow(X\_wickets))

X\_train\_wickets <- X\_wickets[trainIndex\_wickets, , drop = FALSE]

X\_test\_wickets <- X\_wickets[-trainIndex\_wickets, , drop = FALSE]

y\_train\_wickets <- y\_wickets[trainIndex\_wickets]

y\_test\_wickets <- y\_wickets[-trainIndex\_wickets]

# Create a linear regression model for wickets

model\_wickets <- lm(y\_train\_wickets ~ wicket\_confirmation, data = data.frame(wicket\_confirmation = X\_train\_wickets$wicket\_confirmation, y\_train\_wickets))

summary\_wickets <- summary(model\_wickets)

print(summary\_wickets)

# Evaluate the model for runs

y\_pred\_runs <- predict(model\_runs, newdata = data.frame(runs\_scored = X\_test\_runs$runs\_scored))

r2\_runs <- cor(y\_test\_runs, y\_pred\_runs)^2

print(paste("R-squared for runs: ", r2\_runs))

# Evaluate the model for wickets

y\_pred\_wickets <- predict(model\_wickets, newdata = data.frame(wicket\_confirmation = X\_test\_wickets$wicket\_confirmation))

r2\_wickets <- cor(y\_test\_wickets, y\_pred\_wickets)^2

print(paste("R-squared for wickets: ", r2\_wickets))

**Regression Analysis in IPL Python:**

