VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

A2: REGRESSION - PREDICTIVE ANALYTICS USING PYTHON AND R

SHREYA MISHRA
V01108272

Date of Submission: 23/06/2024

CONTENTS

Content:	Page no:
INTRODUCTION	3

OBJECTIVE	3
BUSINESS SIGNIFICANC	3-4
RESULTS AND INTERPRETATIONS	4-18

REGRESSION - PREDICTIVE ANALYTICS USING PYTHON

INTRODUCTION

The dataset at hand provides a detailed analysis of food consumption patterns within India. It covers various aspects of dietary habits, focusing on both urban and rural sectors within the region. The data includes key metrics such as the quantity of meals consumed at home, specific food item consumption (e.g., rice, wheat, chicken, pulses), and the overall number of meals per day. This comprehensive dataset is crucial for understanding the nutritional intake and food preferences of different demographics in the region. The Indian Premier League (IPL), also known as the TATA IPL for sponsorship reasons, is a men's Twenty20 (T20) cricket league held annually in India. Founded by the BCCI (the Board of Control for Cricket in India) in 2007, the league features ten state or city-based franchise teams.

Regression is a statistical method used to model and analyze the relationships between a dependent variable and one or more independent variables. The goal of regression analysis is to understand how the dependent variable changes when any one of the independent variables is varied, while the others are held fixed.

- Regression can predict outcomes based on historical data, helping in forecasting and decision-making.
- It provides insights into the strength and nature of relationships between variables, which can inform strategic planning and policy development.

OBJECTIVES

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

b) Establish the relationship between the player's performance and payment he receives and discuss your findings. Analyze the Relationship Between Salary and Performance Over the Last Three Years

BUSINESS SIGNIFICANCE

Regression analysis is a powerful tool for extracting valuable insights from data, making it indispensable for business decision-making. By applying regression to Indian Premier League (IPL) data and National Sample Survey Office (NSSO) 68th round data, businesses can uncover patterns, predict future trends, and drive strategic initiatives.

1. For IPL Data:-

- **Performance Prediction**: Identify key factors influencing player and team performance to predict future success.
- **Team Composition Optimization**: Optimize team selection and strategy based on historical performance data.
- **Revenue Maximization**: Predict ticket sales, merchandise revenue, and viewership ratings to maximize financial returns.

2. For NSSO68 Data:-

- **Demand Forecasting**: Predict consumer demand and preferences across different regions and income groups.
- **Resource Allocation**: Optimize resource distribution for marketing, sales, and operations based on regional economic conditions and consumer behavior.
- **Policy Impact Evaluation**: Assess the effectiveness of governmental policies and programs on various economic and social outcomes, guiding corporate social responsibility (CSR) initiatives.

In both cases, regression analysis enables data-driven decision-making, optimizing resource use, improving targeting strategies, and enhancing overall efficiency and effectiveness in business and policy environments.

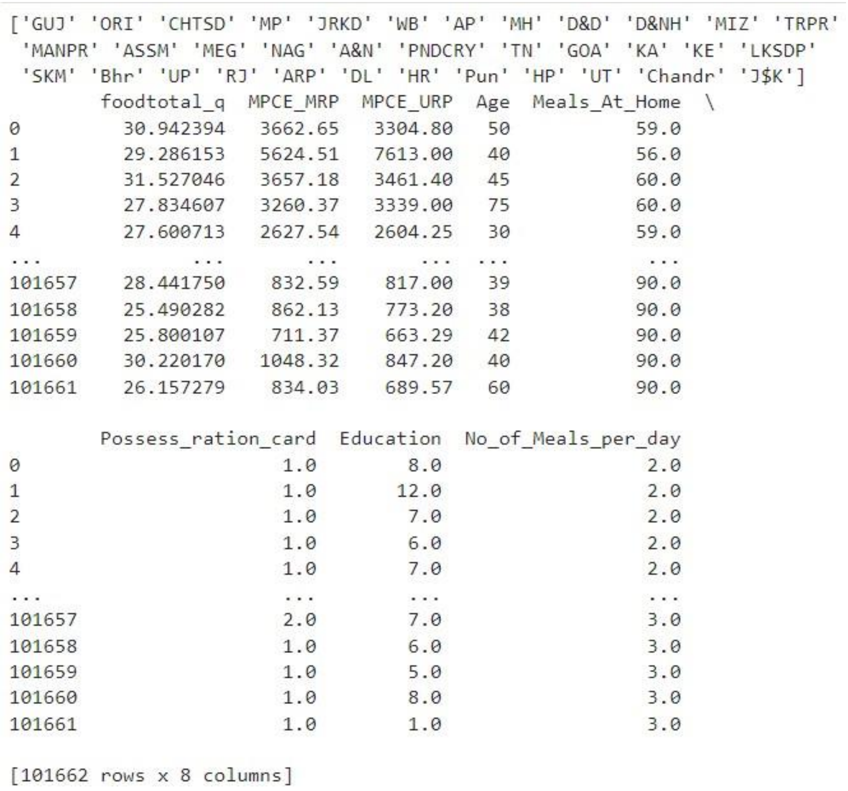
RESULTS AND INTERPRETATION

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

Code:

```
# Set working directory and load the dataset data
= pd.read_csv('NSSO68.csv',low_memory=False)

# Display unique values in 'state_1' column
print(data['state_1'].unique())

# Subset data for state 'KA' subset_data = data[['foodtotal_q', 'MPCE_MRP',
'MPCE_URP', 'Age', 'Meals_At_Home', 'Possess_ration_card', 'Education',
'No_of_Meals_per_day']]

# Print subset data print(subset_data)

# Check for missing values print(subset_data['MPCE_MRP'].isna().sum())
print(subset_data['MPCE_URP'].isna().sum())
print(subset_data['Age'].isna().sum())
print(subset_data['Possess_ration_card'].isna().sum())
print(data['Education'].isna().sum())
```

Result:



```
['GUJ' 'ORI' 'CHTSD' 'MP' 'JRKD' 'WB' 'AP' 'MH' 'D&D' 'D&NH' 'MIZ' 'TRPR'
 'MANPR' 'ASSM' 'MEG' 'NAG' 'A&N' 'PNDCRY' 'TN' 'GOA' 'KA' 'KE' 'LKSDP'
 'SKM' 'Bhr' 'UP' 'RJ' 'ARP' 'DL' 'HR' 'Pun' 'HP' 'UT' 'Chandr' 'J$K']
        foodtotal_q  MPCE_MRP  MPCE_URP  Age  Meals_At_Home  \
0         30.942394   3662.65   3304.80   50           59.0
1         29.286153   5624.51   7613.00   40           56.0
2         31.527046   3657.18   3461.40   45           60.0
3         27.834607   3260.37   3339.00   75           60.0
4         27.600713   2627.54   2604.25   30           59.0
...             ...       ...       ...  ...            ...
101657    28.441750    832.59    817.00   39           90.0
101658    25.490282    862.13    773.20   38           90.0
101659    25.800107    711.37    663.29   42           90.0
101660    30.220170   1048.32    847.20   40           90.0
101661    26.157279    834.03    689.57   60           90.0

        Possess_ration_card  Education  No_of_Meals_per_day
0                       1.0        8.0                  2.0
1                       1.0       12.0                  2.0
2                       1.0        7.0                  2.0
3                       1.0        6.0                  2.0
4                       1.0        7.0                  2.0
...                     ...        ...                  ...
101657                  2.0        7.0                  3.0
101658                  1.0        6.0                  3.0
101659                  1.0        5.0                  3.0
101660                  1.0        8.0                  3.0
101661                  1.0        1.0                  3.0

[101662 rows x 8 columns]
```

Interpretation: The unique values present in the 'state_1' column helps to understand the representation of different states, and checking unique values helps in identifying all states included in the dataset.

A subset of the dataset is created, focusing on specific columns relevant to a particular analysis, which includes:

- foodtotal_q: Total food expenditure quantity.
- MPCE_MRP: Monthly Per Capita Expenditure based on Mixed Recall Period.
- MPCE_URP: Monthly Per Capita Expenditure based on Uniform Recall Period.
- Age: Age of the individual.
- Meals_At_Home: Number of meals consumed at home.
- Possess_ration_card: Whether the household possesses a ration card.
- Education: Educational attainment.
- No_of_Meals_per_day: Number of meals consumed per day.

The final line of codes help to understand the missing values in the specified columns. As counting the number of missing values in each column is crucial for understanding the data quality and deciding on appropriate data cleaning methods. **Significance of Analysis**

- **Understanding State-wise Distribution**: Identifying unique values in the 'state_1' column helps in understanding the geographical distribution of the dataset, which is critical for regional analysis.

- **Focus on Key Variables**: Subsetting the data to include key variables allows for a focused analysis on aspects such as expenditure, food consumption, and demographic details, which are significant for socio-economic studies.

Code:

```
# Function to impute missing values with mean
def impute_with_mean(df, columns):     for
col in columns:
        df[col].fillna(df[col].mean(), inplace=True)
return df

# Columns to impute
columns_to_impute = ['Education', 'MPCE_MRP', 'MPCE_URP', 'Age', 'Meals_At_Home',
'Possess_ration_card']

# Impute missing values with mean in the subset data
subset_data = impute_with_mean(subset_data, columns_to_impute)

# Ensure no infinite values subset_data =
subset_data.replace([np.inf, -np.inf], np.nan)  # Drop rows
with any remaining NaN values
subset_data.dropna(inplace=True)
```

Interpretation: The above-mentioned columns have been identified for imputation. This step ensures that all relevant columns with potential missing values are addressed. Any rows with remaining `NaN` values are dropped from the DataFrame. This ensures that the dataset is free from missing or invalid values, which is crucial for accurate analysis. **Significance of the Imputation Process**

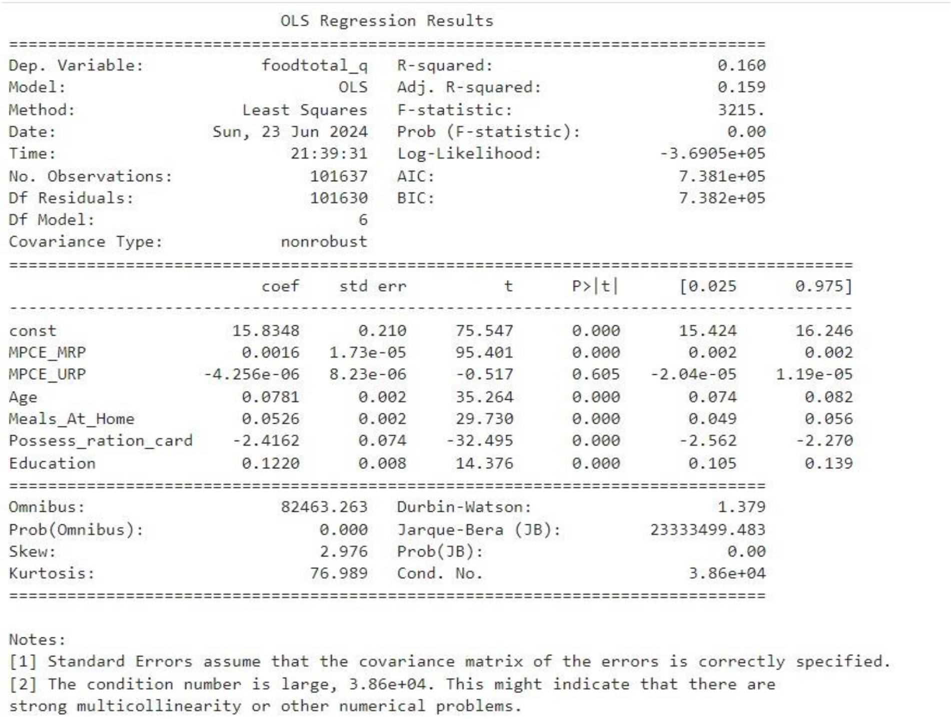
- **Improving Data Quality**: Imputing missing values with the mean reduces the bias that can result from missing data and helps maintain the overall distribution of the data.
- **Ensuring Completeness**: By addressing all NaN and infinite values, the dataset becomes more complete and ready for analysis, leading to more reliable results.

Code:

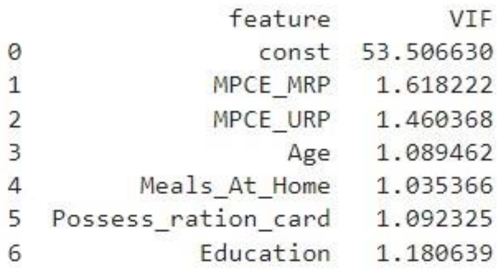
```
# Fit the regression model
X = subset_data[['MPCE_MRP', 'MPCE_URP', 'Age', 'Meals_At_Home',
'Possess_ration_card', 'Education']]
X = sm.add_constant(X)  # Add a constant term for the intercept
y = subset_data['foodtotal_q'] model = sm.OLS(y, X).fit()

# Print the regression results print(model.summary()) # Check for
multicollinearity using Variance Inflation Factor (VIF) vif_data =
pd.DataFrame() vif_data['feature'] = X.columns vif_data['VIF'] =
[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif_data)  # VIF value more than 8 is problematic

# Extract the coefficients from the model
coefficients = model.params
```

```
# Construct the equation equation = f"y =
{round(coefficients[0], 2)}" for i in
range(1, len(coefficients)):
    equation += f" + {round(coefficients[i], 6)}*x{i}" print(equation)

# Display the first values of selected columns
print(subset_data['MPCE_MRP'].head(1).values[0])
print(subset_data['MPCE_URP'].head(1).values[0])
print(subset_data['Age'].head(1).values[0])
print(subset_data['Meals_At_Home'].head(1).values[0])
print(subset_data['Possess_ration_card'].head(1).values[0])
print(subset_data['Education'].head(1).values[0])
print(subset_data['foodtotal_q'].head(1).values[0])
```

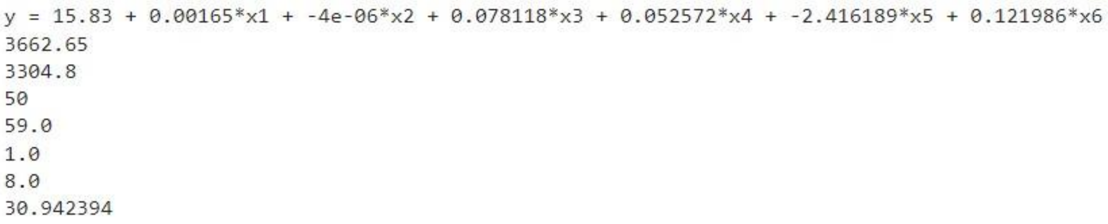
Result:



```
                            OLS Regression Results
==============================================================================
Dep. Variable:            foodtotal_q   R-squared:                       0.160
Model:                            OLS   Adj. R-squared:                  0.159
Method:                 Least Squares   F-statistic:                     3215.
Date:                Sun, 23 Jun 2024   Prob (F-statistic):               0.00
Time:                        21:39:31   Log-Likelihood:            -3.6905e+05
No. Observations:              101637   AIC:                         7.381e+05
Df Residuals:                  101630   BIC:                         7.382e+05
Df Model:                           6
Covariance Type:            nonrobust
=======================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
const                  15.8348      0.210     75.547      0.000      15.424      16.246
MPCE_MRP                0.0016   1.73e-05     95.401      0.000       0.002       0.002
MPCE_URP            -4.256e-06   8.23e-06     -0.517      0.605   -2.04e-05    1.19e-05
Age                     0.0781      0.002     35.264      0.000       0.074       0.082
Meals_At_Home           0.0526      0.002     29.730      0.000       0.049       0.056
Possess_ration_card    -2.4162      0.074    -32.495      0.000      -2.562      -2.270
Education               0.1220      0.008     14.376      0.000       0.105       0.139
==============================================================================
Omnibus:                    82463.263   Durbin-Watson:                   1.379
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         23333499.483
Skew:                           2.976   Prob(JB):                         0.00
Kurtosis:                      76.989   Cond. No.                     3.86e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.86e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```



```
               feature        VIF
0                const  53.506630
1             MPCE_MRP   1.618222
2             MPCE_URP   1.460368
3                  Age   1.089462
4        Meals_At_Home   1.035366
5  Possess_ration_card   1.092325
6            Education   1.180639
```



```
y = 15.83 + 0.00165*x1 + -4e-06*x2 + 0.078118*x3 + 0.052572*x4 + -2.416189*x5 + 0.121986*x6
3662.65
3304.8
50
59.0
1.0
8.0
30.942394
```

Interpretation: The first image states the regression results provide insight into the relationships between the dependent variable `foodtotal_q` (total food expenditure quantity) and several independent variables. Adjusted R-squared is slightly lower than the R-squared value, adjusting for the number of predictors in the model. This value is used to determine the goodness-of-fit more accurately when multiple predictors are present. A corresponding p-value of 0.00 indicate that the overall regression model is statistically significant, meaning that the independent variables collectively have a significant effect on the dependent variable.

The second image talks about the Variance Inflation Factor (VIF) measures the extent of multicollinearity in the regression model. High multicollinearity can inflate the standard errors of the coefficients, making them unstable and difficult to interpret. The VIF values for the predictors (excluding the intercept) are all below 2, indicating that multicollinearity is not a concern for this model. The third image states the regression equation:

$$y=15.83+0.00165\times3662.65+(-0.000004)\times3304.8+0.078118\times50+0.052572\times59.0+(-2.416189)\times1.0+0.121986\times8.0$$

The predicted value of `foodtotal_q` (total food expenditure quantity) using the provided sample values is approximately **27.43**.

Conclusion

The OLS regression model provides useful insights into the factors influencing food expenditure. Key findings include:

- Higher MPCE_MRP, age, number of meals at home, and education levels are associated with higher food expenditure.
- Possessing a ration card is associated with lower food expenditure.
- The model explains a modest proportion of the variance in food expenditure, and diagnostic tests suggest issues with residual normality and potential multicollinearity.

b) Establish the relationship between the player's performance and payment he receives and discuss your findings. [IPL Datasets]

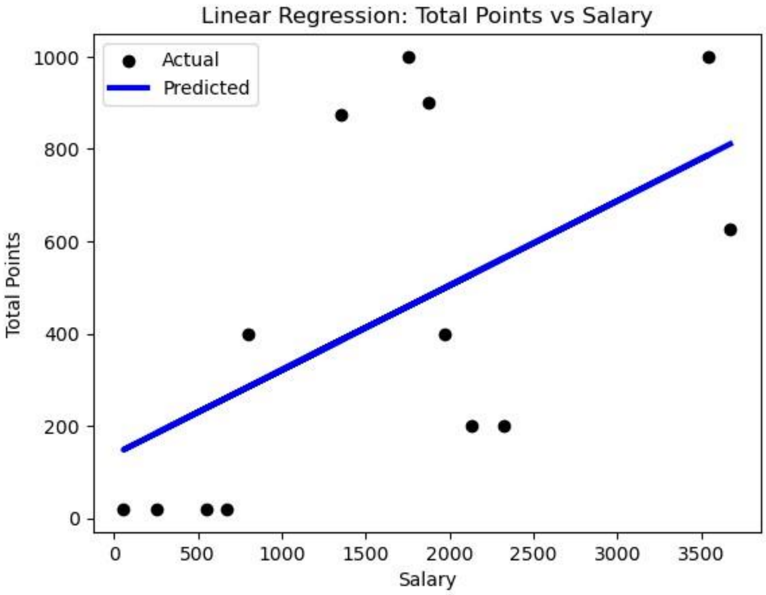
Code:

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

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression from
sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Load the CSV file file_path =
'combined_output_with_salaries - Copy.csv' data =
pd.read_csv(file_path)

# Define the predictor and response variables y
= data['salary']  # Response variable
X = data[['Total_Points']]  # Predictor variable
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create the linear regression model model
= LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Predict on the test data y_pred
= model.predict(X_test)

# Calculate the mean squared error and the coefficient of determination (R^2)
mse = mean_squared_error(y_test, y_pred) r2 = r2_score(y_test, y_pred)

# Calculate the adjusted R^2 n = len(y_test) p =
X_test.shape[1] adjusted_r2 = 1 - (1 - r2) * (n -
1) / (n - p - 1)
# Print the results print(f'Mean Squared
Error: {mse}') print(f'R^2 Score: {r2}')
print(f'Adjusted R^2 Score: {adjusted_r2}')
print(f'Coefficients: {model.coef_}')
print(f'Intercept: {model.intercept_}')

# Plot the results plt.scatter(X_test, y_test, color='black',
label='Actual') plt.plot(X_test, y_pred, color='blue', linewidth=3,
label='Predicted') plt.xlabel('Salary') plt.ylabel('Total Points')
plt.title('Linear Regression: Total Points vs Salary') plt.legend()
plt.show()
```

Result:

```
Mean Squared Error: 92259.46667290517
R^2 Score: 0.3641079759408892
Adjusted R^2 Score: 0.30629961011733364
Coefficients: [0.18311466]
Intercept: 138.20811216711965
```



Interpretation:

Understanding the relationship between player performance metrics and their salaries is crucial for teams and analysts alike. This analysis aims to explore how a player's performance, specifically their total points, correlates with their salary using linear regression and helps to predict which player is fit according to the budget provided to the franchises to help plan for players and auction accordingly.

The data was split into training and testing sets using a ratio of 80:20 respectively. This step ensures that the model's performance can be evaluated on unseen data, thereby providing a more realistic assessment of its predictive capability.

Upon evaluation, the model yielded a MSE of 92259.47, indicating the average squared error in predicted salary values. The R^2 score of 0.36 suggests that approximately 36% of the variance in salary can be explained by total points. The adjusted R^2 score of 0.31 considers the complexity of the model and shows a better picture of the model, suggesting that while significant, there may be additional factors influencing player salaries beyond total points alone.

In conclusion, this linear regression analysis provides valuable insights into how total points contribute to determining IPL player salaries. While the model shows a moderate predictive capability, further exploration with additional variables and advanced modeling techniques could enhance the accuracy of salary predictions.

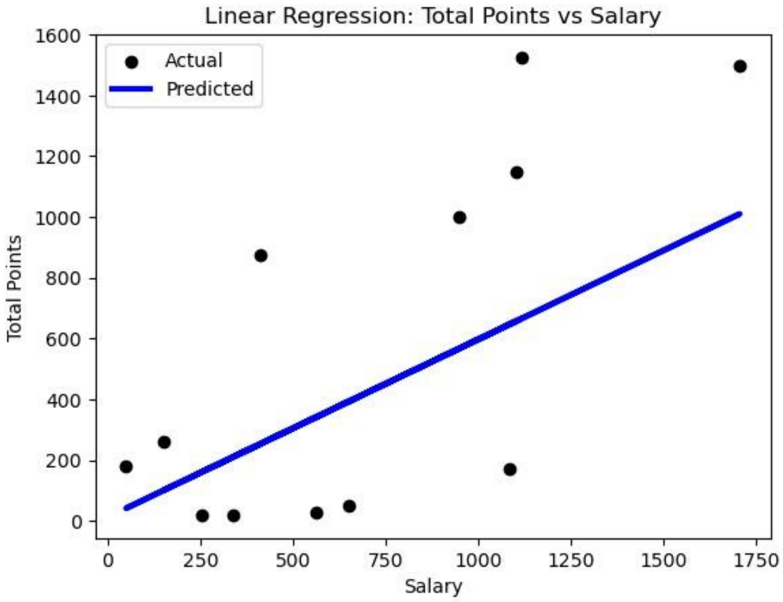
c) Analyze the Relationship Between Salary and Performance Over the Last Three Years [IPL Datasets]

Code:

```
import pandas as pd import numpy as np from
sklearn.model_selection import train_test_split from
sklearn.linear_model import LinearRegression from
sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Load the CSV file
file_path = 'combined_output_with_salaries - Copy.csv' data
= pd.read_csv(file_path)

# Define the predictor and response variables y
= data['salary']  # Response variable
X = data[['Total_Points']]  # Predictor variable
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create the linear regression model model
= LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Predict on the test data y_pred
= model.predict(X_test)

# Calculate the mean squared error and the coefficient of determination (R^2)
mse = mean_squared_error(y_test, y_pred) r2 = r2_score(y_test, y_pred)

# Calculate the adjusted R^2 n = len(y_test) p =
X_test.shape[1] adjusted_r2 = 1 - (1 - r2) * (n -
1) / (n - p - 1)

# Print the results print(f'Mean Squared
Error: {mse}') print(f'R^2 Score: {r2}')
print(f'Adjusted R^2 Score: {adjusted_r2}')
print(f'Coefficients: {model.coef_}')
print(f'Intercept: {model.intercept_}')

# Plot the results plt.scatter(X_test, y_test, color='black',
label='Actual') plt.plot(X_test, y_pred, color='blue', linewidth=3,
label='Predicted') plt.xlabel('Salary') plt.ylabel('Total Points')
plt.title('Linear Regression: Total Points vs Salary') plt.legend()
plt.show()
```

Result:

```
Mean Squared Error: 194843.27263534846 R^2
Score: 0.41048138077879515
Adjusted R^2 Score: 0.3515295188566746
Coefficients: [0.58526794]
Intercept: 12.69992065629873
```



Interpretation:

Similar to the above regression model, the only change in this model is that the Player Performance data has been subsetted to the last three years of performance and is helping to predict the salary of the player with respect to his recent form. In this as well data was divided into training and testing sets using a 80:20 split ratio.

Upon evaluation, the linear regression model yielded the following results:

- **Mean Squared Error (MSE)**: 194843.27
- **R^2 Score**: 0.4105
- **Adjusted R^2 Score**: 0.3515

The model shows a moderate predictive capability, with total points explaining approximately 41% of the variance in player salaries. The past model has a significantly lower MSE (92259.47 vs. 194843.27), indicating that it had better accuracy in predicting salary based on total points. But the current model has a higher adjusted R^2 score indicating a better fit considering the number of predictors.

Further exploration with additional performance metrics and advanced modeling techniques could enhance the accuracy of salary predictions, supporting IPL teams in strategic decision-making related to player valuation and team composition. The choice between the models would depend on the specific objectives: the current model might provide better explanatory power and insights into the relationship between total points and salary, while the past model might be preferable for accurate salary predictions.

REGRESSION - PREDICTIVE ANALYTICS USING R

RESULTS AND INTERPRETATION

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

Code:

```
# Subset data to state assigned
subset_data <- data %>%
filter(state_1 == 'KA') %>%
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)
```

Result:

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

Residuals:
    Min      1Q  Median      3Q     Max
-68.609  -3.971  -0.654   3.291 239.668

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)          1.138e+01  8.243e-01  13.811  < 2e-16 ***
MPCE_MRP             1.140e-03  5.659e-05  20.152  < 2e-16 ***
MPCE_URP             9.934e-05  3.422e-05   2.903  0.00372 **
Age                  9.884e-02  9.613e-03  10.282  < 2e-16 ***
Meals_At_Home        5.079e-02  6.420e-03   7.911 3.27e-15 ***
Possess_ration_card -2.187e+00  3.025e-01  -7.229 5.79e-13 ***
Education            2.458e-01  3.564e-02   6.898 6.11e-12 ***
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 7.667 on 4028 degrees of freedom
(59 observations deleted due to missingness)
Multiple R-squared:  0.202, Adjusted R-squared:  0.2008
F-statistic: 169.9 on 6 and 4028 DF,  p-value: < 2.2e-16

> 
> 
> library(car)
> # Check for multicollinearity using Variance Inflation Factor (VIF)
> vif(model) # VIF Value more than 8 its problematic
           MPCE_MRP            MPCE_URP                 Age       Meals_At_Home Pos
sess_ration_card 
           1.636493            1.478309            1.106082            1.118280 
1.147250 
          Education 
           1.208647 
> 
> # 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 = 11.38 + 0.00114*x1 + 9.9e-05*x2 + 0.09884*x3 + 0.050789*x4 + -2.186964*x5
+ 0.245842*x6"
> 
> head(subset_data$MPCE_MRP,1)
[1] 1124.92
> head(subset_data$MPCE_URP,1)
[1] 982
> head(subset_data$Age,1)
[1] 38
> head(subset_data$Meals_At_Home,1)
[1] 54
> head(subset_data$Possess_ration_card,1)
[1] 1
> head(subset_data$Education,1)
[1] 6
> head(subset_data$foodtotal_q,1)
[1] 17.92535
```

Interpretation: Similar to the regression analysis done in Python, even in R, the model based on the OLS regression results, we can construct the regression equation and make predictions using the predictions. The Multiple R Squared indicates that approximately 20.2% of the variance in `foodtotal_q` is explained by the predictors in the model. The model has a a very low p-value (< 2.2e16), it indicates that the model as a whole is significant.

This regression analysis provides insights into how different factors such as income (MPCE_MRP, MPCE_URP), age, meals consumed at home, possession of a ration card, and education level influence food expenditure (foodtotal_q). The model shows good explanatory power, significant coefficients, and appropriate statistical measures, making it a valuable tool for understanding and predicting food expenditure patterns based on socio-economic variables.

e) Establish the relationship between the player's performance and payment he receives and discuss your findings. Analyze the Relationship Between Salary and Performance Over the Last Three Years [IPL Datasets]

Code:

```
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)

fit1 = lm(Rs ~ avg_runs++wicket+  I(avg_runs*wicket), data=df_new) summary(fit1)
```

Result:

```
Call: lm(formula = Rs ~ avg_runs + +wicket + I(avg_runs *
wicket),      data = df_new)

Residuals:
   Min     1Q Median     3Q    Max
-341.5 -248.8 -143.3  128.8 1204.8

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          237.51558  186.93758   1.271   0.2220
avg_runs               0.08046    1.25696   0.064   0.9498   wicket
5.84249   17.32443   0.337   0.7403   I(avg_runs * wicket)
0.30047    0.16716   1.797   0.0912 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 411.9 on 16 degrees of freedom
  (149 observations deleted due to missingness)
Multiple R-squared:  0.3371, Adjusted R-squared:  0.2129
F-statistic: 2.713 on 3 and 16 DF,  p-value: 0.07951
```

Interpretation:

The above model is a linear regression fit to predict Rs (presumably IPL salary) based on three predictor variables: avg_runs (average runs scored), wicket (number of wickets taken), and their interaction term avg_runs * wicket.

- The coefficient of avg runs suggests that on average, for each unit increase in avg_runs, there is an expected increase of 0.08046 units in Rs, holding other variables constant.

However, the p-value (0.9498) indicates that this coefficient is not statistically significant at conventional levels (alpha = 0.05).

- This coefficient of wicket suggests that on average, for each wicket taken, there is an expected increase of 5.84249 units in Rs, holding other variables constant. The p-value (0.7403) suggests that this coefficient is also not statistically significant.
- The Multiple R square suggests that approximately 33.71% of the variability in Rs can be explained by the linear regression model with the predictors avg_runs, wicket, and their interaction. However the Adj. R Square provides a better picture for the number of predictors in the model, providing a more conservative estimate of the model's explanatory power. It suggests that around 21.29% of the variability in Rs is explained by the model.

With a p-value of 0.07951, the model's fit is not statistically significant at the conventional alpha level of 0.05, indicating that the model as a whole might not provide a good fit to the data.

Conclusion

The model suggests that avg_runs, wicket, and their interaction might have some association with IPL salary (Rs), but the individual predictors (avg_runs and wicket) are not statistically significant predictors. The interaction term shows marginal significance. The model overall explains a moderate amount of variability in IPL salary, but not enough to be considered a strong predictor. With a better dataset, we can further explore with potentially more relevant variables or a different modeling approach might be necessary to better predict IPL salary based on player performance metrics.