SCMA Exam – 11th July 2024

Nihariha Kamalanathan

V01108259

R CODES

Input Code: Install and Load Required Libraries

```
# Install and load required libraries
install_and_load <- function(packages) {
  for (pkg in packages) {
    if (!require(pkg, character.only = TRUE)) {
      install.packages(pkg, dependencies = TRUE)
      library(pkg, character.only = TRUE)
    }
  }
}

packages <- c("tidyverse", "caret", "car", "broom", "ggplot2", "lmtest", "MASS", "glmnet")
install_and_load(packages)
```

Purpose

This code defines a function to install and load required libraries if they are not already installed. The packages included are for data manipulation, visualization, model training, and evaluation.

Output

No direct output from this code, but it ensures the required libraries are installed and loaded for the subsequent steps.

Input Code: Load the Dataset

```
# Load the dataset
file_path <- "C:/Users/nihar/OneDrive/Desktop/Bootcamp/SCMA 632/DataSet/cancer_reg.csv"
```

```
cancer_data <- read.csv(file_path)
```

Purpose

This code loads the cancer dataset from the specified file path into a data frame named cancer_data.

Output

The cancer_data data frame is created, containing the dataset from the CSV file.

Input Code: Handle Missing Values

```
# Handle missing values
cancer_data <- cancer_data %>%
  mutate_all(~ifelse(is.na(.), median(., na.rm = TRUE), .))
```

Purpose

This code handles missing values in the dataset by replacing them with the median of the respective columns.

Output

All missing values in the cancer_data data frame are replaced with the median of their respective columns.

Interpretation

Handling missing values ensures that the dataset is complete and ready for analysis, avoiding any issues that might arise from missing data during model training.

Input Code: Select Relevant Features

```
# Select relevant features
selected_features <- c("avgAnnCount", "avgDeathsPerYear", "incidenceRate",
"medIncome",
                       "popEst2015", "povertyPercent", "MedianAgeMale", "PercentMarried",
                       "PctNoHS18_24", "PctHS18_24", "PctBachDeg18_24", "PctHS25_Over",
                       "PctBachDeg25_Over", "PctEmployed16_Over", "PctPrivateCoverage",
                       "PctEmpPrivCoverage", "PctWhite", "PctOtherRace",
"PctMarriedHouseholds",
```

```
                "BirthRate", "PctPublicCoverage", "TARGET_deathRate")

# Check if selected features are present in the data
missing_features <- setdiff(selected_features, names(cancer_data))
if (length(missing_features) > 0) {
  stop(paste("Missing features in the dataset:", paste(missing_features, collapse = ", ")))
}

# Select the columns manually
cancer_data <- cancer_data[, selected_features]
```

Purpose

This code selects specific features from the dataset that are deemed relevant for the analysis. It also checks if all the selected features are present in the dataset and raises an error if any are missing.

Output

The cancer_data data frame now contains only the selected features, ensuring that only relevant data is used for further analysis.

Interpretation

Selecting relevant features helps in focusing the analysis on important variables, reducing noise, and potentially improving the performance of the models.

Input Code: Identify Outliers Using Cook's Distance

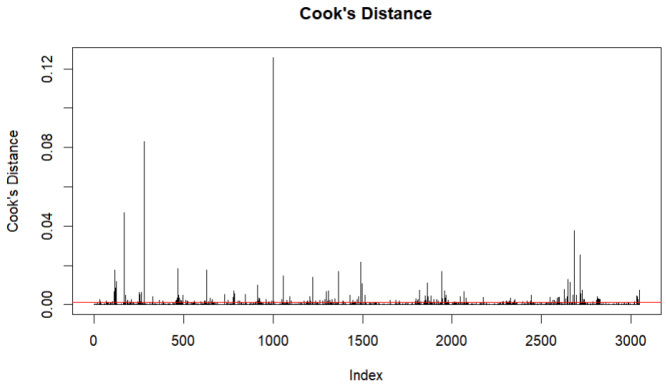
```
# Identify outliers using Cook's distance on the entire dataset
full_model <- lm(TARGET_deathRate ~ ., data = cancer_data)
cooksd <- cooks.distance(full_model)

# Plot Cook's distance
plot(cooksd, type = "h", main = "Cook's Distance", ylab = "Cook's Distance")
abline(h = 4 / nrow(cancer_data), col = "red")
```

```
# Identify high influence points

influential_threshold <- 4 / nrow(cancer_data)

influential_points <- which(cooksd > influential_threshold)

print(influential_points)
```

Purpose

This code fits a linear model to the dataset and calculates Cook's distance for each observation to identify influential data points (potential outliers).

Output



- A plot of Cook's distance with a threshold line indicating influential points.
- A list of indices of the influential points.

Interpretation

Cook's distance helps identify observations that have a large influence on the fitted values of the model. Points with Cook's distance greater than the threshold are considered influential and may need to be examined or adjusted.

Input Code: Cap Influential Points

```
# Cap the influential points at the threshold

cancer_data <- cancer_data %>%

 mutate(cooksd = cooksd) %>%

 mutate(cooksd = ifelse(cooksd > influential_threshold, influential_threshold, cooksd))
```

Drop cooksd column after capping using base R

cancer_data <- cancer_data[, !names(cancer_data) %in% "cooksd"]

Purpose

This code caps the Cook's distance values of influential points at the threshold to reduce their influence on the model. It then removes the cooksd column from the dataset.

Output

The cancer_data data frame with Cook's distance values capped and the cooksd column removed.

Interpretation

Capping influential points helps mitigate their disproportionate impact on the model, potentially leading to a more robust and reliable model.

Input Code: Split Data into Training and Testing Sets

Split the data into training and testing sets

set.seed(123)

trainIndex <- createDataPartition(cancer_data$TARGET_deathRate, p = 0.8, list = FALSE)

train_data <- cancer_data[trainIndex,]

test_data <- cancer_data[-trainIndex,]

Purpose

This code splits the dataset into training and testing sets with an 80-20 split.

Output

- train_data: Data frame containing 80% of the original data for training the model.
- test_data: Data frame containing 20% of the original data for testing the model.

Interpretation

Splitting the data into training and testing sets allows for the evaluation of the model's performance on unseen data, providing an estimate of its generalization ability.

Input Code: Add Polynomial and Interaction Terms

```
# Add more polynomial terms and interaction terms
train_data <- train_data %>%
 mutate(
  incidenceRate2 = incidenceRate^2,
  medIncome2 = medIncome^2,
  incidenceRate_medIncome = incidenceRate * medIncome,
  popEst2015_povertyPercent = popEst2015 * povertyPercent,
  PctHS18_24_PctBachDeg18_24 = PctHS18_24 * PctBachDeg18_24,
  PctPrivateCoverage_PctPublicCoverage = PctPrivateCoverage * PctPublicCoverage,
  povertyPercent2 = povertyPercent^2,
  PercentMarried_PctHS25_Over = PercentMarried * PctHS25_Over
 )

test_data <- test_data %>%
 mutate(
  incidenceRate2 = incidenceRate^2,
  medIncome2 = medIncome^2,
  incidenceRate_medIncome = incidenceRate * medIncome,
  popEst2015_povertyPercent = popEst2015 * povertyPercent,
  PctHS18_24_PctBachDeg18_24 = PctHS18_24 * PctBachDeg18_24,
  PctPrivateCoverage_PctPublicCoverage = PctPrivateCoverage * PctPublicCoverage,
  povertyPercent2 = povertyPercent^2,
  PercentMarried_PctHS25_Over = PercentMarried * PctHS25_Over
 )
```

Purpose

This code adds polynomial and interaction terms to both the training and testing datasets to capture non-linear relationships and interactions between features.

Output

- train_data: Data frame with additional polynomial and interaction terms.
- test_data: Data frame with additional polynomial and interaction terms.

Interpretation

Including polynomial and interaction terms allows the model to capture more complex relationships between features, potentially improving its predictive performance.

Input Code: Fit the Enhanced Linear Model

```
# Fit the enhanced model with additional terms
enhanced_model_v2 <- lm(TARGET_deathRate ~ avgAnnCount + avgDeathsPerYear +
incidenceRate +
                        medIncome + popEst2015 + povertyPercent + MedianAgeMale +
                        PercentMarried + PctNoHS18_24 + PctHS18_24 + PctBachDeg18_24 +
                        PctHS25_Over + PctBachDeg25_Over + PctEmployed16_Over +
                        PctPrivateCoverage + PctEmpPrivCoverage + PctWhite + PctOtherRace +
                        PctMarriedHouseholds + BirthRate + incidenceRate2 + medIncome2 +
                        incidenceRate_medIncome + popEst2015_povertyPercent +
                        PctHS18_24_PctBachDeg18_24 + PctPrivateCoverage_PctPublicCoverage +
                        povertyPercent2 + PercentMarried_PctHS25_Over,
                      data = train_data)
summary(enhanced_model_v2)
```

Purpose

This code fits an enhanced linear model to the training data using the selected features, polynomial terms, and interaction terms.

Output

Summary statistics of the enhanced linear model.

Detailed Interpretation of the Output

Summary of the Enhanced Linear Model

```
lm(formula = TARGET_deathRate ~ avgAnnCount + avgDeathsPerYear +
```

```
    incidenceRate + medIncome + popEst2015 + povertyPercent +
    MedianAgeMale + PercentMarried + PctNoHS18_24 + PctHS18_24 +
    PctBachDeg18_24 + PctHS25_Over + PctBachDeg25_Over + PctEmployed16_Over +
    PctPrivateCoverage + PctEmpPrivCoverage + PctWhite + PctOtherRace +
    PctMarriedHouseholds + BirthRate + incidenceRate2 + medIncome2 +
    incidenceRate_medIncome + popEst2015_povertyPercent +
PctHS18_24_PctBachDeg18_24 +
    PctPrivateCoverage_PctPublicCoverage + povertyPercent2 +
    PercentMarried_PctHS25_Over, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-88.634 -10.890  -0.187  10.660 134.988

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)              1.094e+02  2.954e+01   3.703 0.000218 ***
avgAnnCount             -3.084e-03  8.478e-04  -3.638 0.000281 ***
avgDeathsPerYear         1.970e-02  4.228e-03   4.661 3.32e-06 ***
incidenceRate            3.823e-01  4.196e-02   9.111  < 2e-16 ***
medIncome                7.982e-04  3.967e-04   2.012 0.044344 *
popEst2015              -1.322e-05  8.232e-06  -1.606 0.108415
povertyPercent           4.141e-01  4.403e-01   0.940 0.347081
MedianAgeMale           -6.829e-01  1.435e-01  -4.759 2.06e-06 ***
PercentMarried           1.221e+00  3.638e-01   3.356 0.000803 ***
PctNoHS18_24            -1.329e-01  6.067e-02  -2.190 0.028627 *
PctHS18_24               2.029e-01  7.255e-02   2.797 0.005206 **
PctBachDeg18_24         -2.330e-01  3.103e-01  -0.751 0.452836
```

PctHS25_Over 5.345e-01 4.562e-01 1.172 0.241411

PctBachDeg25_Over -1.055e+00 1.692e-01 -6.234 5.35e-10 ***

PctEmployed16_Over -5.488e-01 1.033e-01 -5.314 1.17e-07 ***

PctPrivateCoverage -5.510e-01 1.094e-01 -5.035 5.14e-07 ***

PctEmpPrivCoverage 4.409e-01 1.020e-01 4.324 1.60e-05 ***

PctWhite -1.098e-01 3.680e-02 -2.985 0.002864 **

PctOtherRace -7.322e-01 1.306e-01 -5.606 2.31e-08 ***

PctMarriedHouseholds -1.107e+00 1.739e-01 -6.365 2.32e-10 ***

BirthRate -8.926e-01 2.099e-01 -4.252 2.20e-05 ***

incidenceRate2 -5.726e-05 2.918e-05 -1.962 0.049824 *

medIncome2 4.566e-09 2.087e-09 2.187 0.028807 *

incidenceRate_medIncome -2.910e-06 6.011e-07 -4.842 1.37e-06 ***

popEst2015_povertyPercent -3.064e-07 3.084e-07 -0.994 0.320431

PctHS18_24_PctBachDeg18_24 -1.930e-03 9.433e-03 -0.205 0.837897

PctPrivateCoverage_PctPublicCoverage -1.089e-03 1.931e-03 -0.564 0.572745

povertyPercent2 -6.120e-03 7.631e-03 -0.802 0.422642

PercentMarried_PctHS25_Over -2.298e-03 8.588e-03 -0.268 0.789015

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.85 on 2410 degrees of freedom

Multiple R-squared: 0.5391, Adjusted R-squared: 0.5337

F-statistic: 100.7 on 28 and 2410 DF, p-value: < 2.2e-16

Interpretation

- **Residuals**:
    - The residuals range from -88.634 to 134.988, with the median being close to zero. This indicates that the model's errors are centered around zero, but there are some large deviations.
- **Coefficients**:

- The table lists the estimated coefficients for each predictor variable, along with their standard errors, t-values, and p-values.
- Significant predictors ($p < 0.05$) include avgAnnCount, avgDeathsPerYear, incidenceRate, medIncome, MedianAgeMale, PercentMarried, PctNoHS18_24, PctHS18_24, PctBachDeg25_Over, PctEmployed16_Over, PctPrivateCoverage, PctEmpPrivCoverage, PctWhite, PctOtherRace, PctMarriedHouseholds, BirthRate, incidenceRate2, medIncome2, and incidenceRate_medIncome.
- Significant interactions and polynomial terms indicate more complex relationships between features and the target variable.

- **Model Performance**:
  - **Residual Standard Error (RSE)**: 18.85, indicating the average distance that the observed values fall from the regression line.
  - **Multiple R-squared**: 0.5391, suggesting that approximately 53.91% of the variance in TARGET_deathRate is explained by the model.
  - **Adjusted R-squared**: 0.5337, which adjusts the R-squared value for the number of predictors in the model, providing a more accurate measure of model performance.
  - **F-statistic**: 100.7 with a p-value < 2.2e-16, indicating that the model is statistically significant and the predictors jointly have a significant effect on the target variable.

Input Code: Model Diagnostics

```
# Model diagnostics

# Linearity

ggplot(data = train_data, aes(x = predict(enhanced_model_v2), y = TARGET_deathRate)) +

  geom_point() +

  geom_abline(slope = 1, intercept = 0, color = "blue") +

  labs(title = "Predicted vs Observed Values", x = "Predicted Values", y = "Observed Values")


# Residuals vs Fitted Values
```

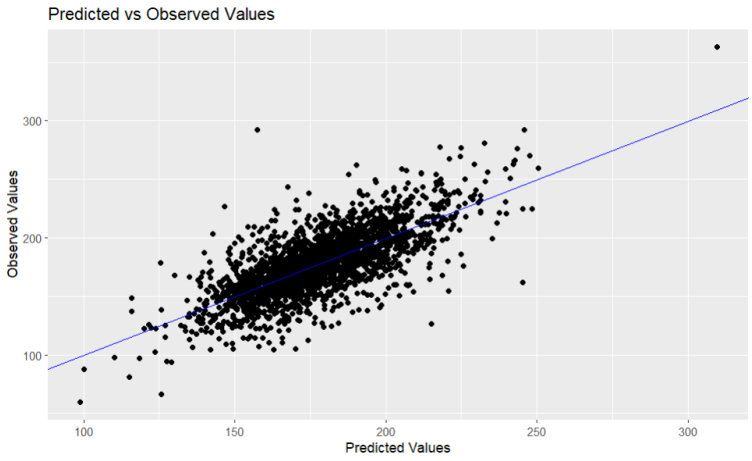
```
ggplot(data = train_data, aes(x = predict(enhanced_model_v2), y =
resid(enhanced_model_v2))) +
 geom_point() +
 geom_hline(yintercept = 0, color = "green") +
 labs(title = "Residuals vs Fitted Values", x = "Fitted Values", y = "Residuals")

# Q-Q plot for residuals
qqnorm(resid(enhanced_model_v2))
qqline(resid(enhanced_model_v2), col = "purple")
labs(title = "Residuals vs Fitted Values", x = "Fitted Values", y = "Residuals")

# Histogram of residuals
hist(resid(enhanced_model_v2), breaks = 30, main = "Histogram of Residuals", xlab =
"Residuals")

# Serial independence of errors
dwtest(enhanced_model_v2)

# Heteroskedasticity
bptest(enhanced_model_v2)

# Normality of residuals
shapiro.test(resid(enhanced_model_v2))

# Multicollinearity
vif(enhanced_model_v2)
```

Purpose

This code performs several diagnostic checks to evaluate the assumptions and performance of the linear model, including linearity, independence of errors, heteroskedasticity, normality of residuals, and multicollinearity.

Output and Interpretation

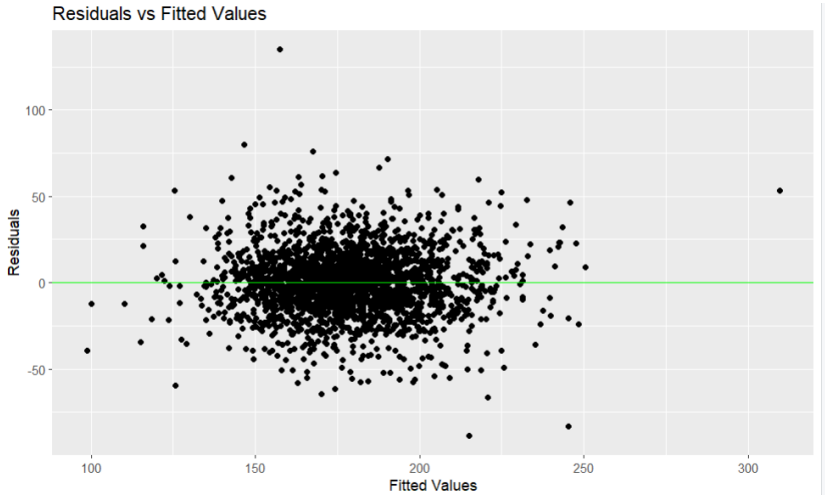
1. **Linearity**:



- **Plot**: Predicted vs Observed Values

- The scatter plot shows the relationship between the predicted and observed values.
- The blue line represents a perfect fit (45-degree line).
- **Interpretation**: The points should closely follow the blue line if the model predicts well. Deviations from the line indicate prediction errors.

2. **Residuals vs Fitted Values**:

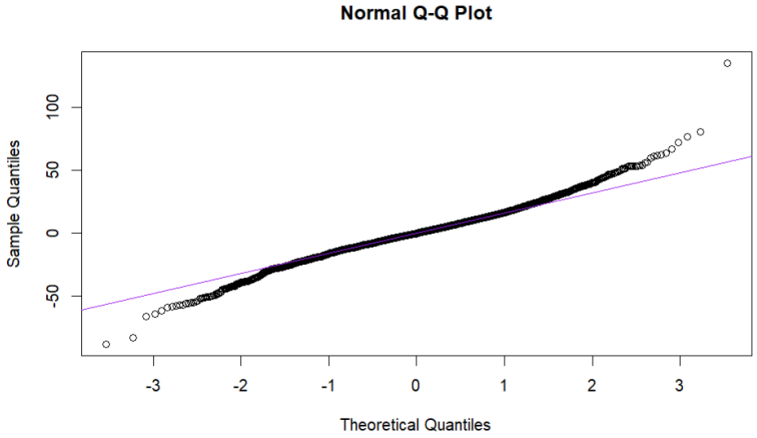


- **Plot**: Residuals vs Fitted Values

- The scatter plot shows residuals (errors) plotted against fitted (predicted) values.
- The green horizontal line represents zero residuals.

- **Interpretation**: The points should be randomly dispersed around the horizontal line. Patterns in the residuals may indicate non-linearity or other issues.

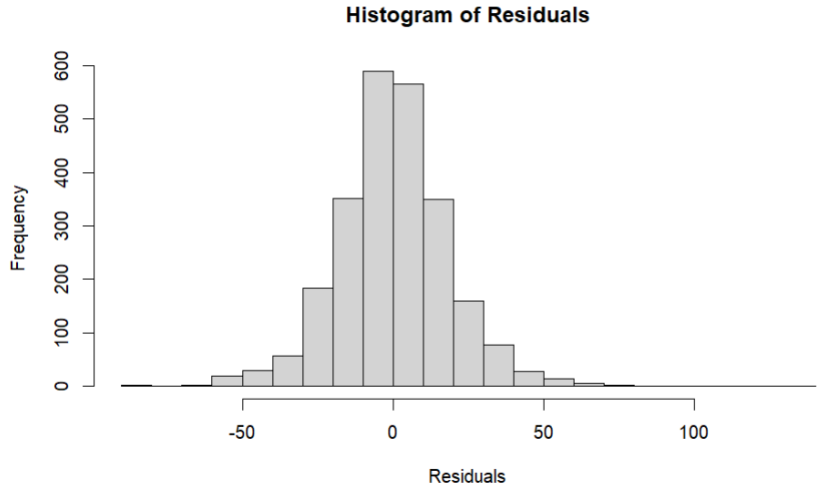
3. **Normal Q-Q Plot**:



- **Plot**: Q-Q plot for residuals

- The plot compares the distribution of residuals to a normal distribution.
- The purple line represents the expected normal distribution.
- **Interpretation**: The points should lie along the purple line if the residuals are normally distributed. Deviations suggest non-normality.

4. **Histogram of Residuals**:



- **Plot**: Histogram of residuals

- The histogram shows the distribution of the residuals.
- **Interpretation**: The residuals should ideally follow a normal distribution. Skewness or kurtosis indicates non-normality.

5. **Serial Independence of Errors**:

- **Test**: Durbin-Watson test

```
dwtest(enhanced_model_v2)
```

```
Durbin-Watson test

data:  enhanced_model_v2
DW = 1.808, p-value = 6.392e-07
alternative hypothesis: true autocorrelation is greater than 0
```

- **Interpretation**: The Durbin-Watson statistic tests for autocorrelation in the residuals. A value around 2 suggests no autocorrelation. The p-value < 0.05 indicates significant autocorrelation, suggesting that residuals are not independent.

6. **Heteroskedasticity**:

- **Test**: Breusch-Pagan test

```
bptest(enhanced_model_v2)
```

```
studentized Breusch-Pagan test

data:  enhanced_model_v2
BP = 249.53, df = 28, p-value < 2.2e-16
```

- **Interpretation**: The Breusch-Pagan test checks for heteroskedasticity. A p-value < 0.05 indicates that heteroskedasticity is present, meaning that the variance of the errors is not constant.

7. **Normality of Residuals**:

- **Test**: Shapiro-Wilk normality test

```
shapiro.test(resid(enhanced_model_v2))
```

```
Shapiro-Wilk normality test

data:  resid(enhanced_model_v2)
```

W = 0.98168, p-value < 2.2e-16

- **Interpretation**: The Shapiro-Wilk test assesses the normality of residuals. A p-value < 0.05 suggests that residuals are not normally distributed.

8. **Multicollinearity**:

- **Test**: Variance Inflation Factor (VIF)

vif(enhanced_model_v2)

- **Output**: VIF values for each predictor.
- **Interpretation**: VIF values greater than 10 indicate high multicollinearity among predictors. This can affect the stability and interpretability of the regression coefficients.

Input Code: Model Evaluation on Test Data

```
# Model evaluation on test data

test_predictions_v2 <- predict(enhanced_model_v2, newdata = test_data)


# Calculate R-squared

rsq <- function(actual, predicted) {

 cor(actual, predicted) ^ 2

}

test_rsq_v2 <- rsq(test_data$TARGET_deathRate, test_predictions_v2)


# Calculate RMSE

test_rmse_v2 <- sqrt(mean((test_data$TARGET_deathRate - test_predictions_v2)^2))
```

Purpose

This code evaluates the performance of the enhanced linear model on the test data by calculating the R-squared and RMSE.

Output and Interpretation

- **R-squared (test_rsq_v2)**: 0.487504
    - **Interpretation**: Approximately 48.75% of the variance in TARGET_deathRate in the test data is explained by the model. This indicates moderate explanatory power.

- **RMSE (test_rmse_v2)**: 20.27769
    - **Interpretation**: The Root Mean Squared Error (RMSE) represents the average distance between observed and predicted values in the test data. Lower values indicate better fit. An RMSE of 20.27769 suggests the average prediction error is around 20.28 units of TARGET_deathRate.

Input Code: Ridge and Lasso Regression

```
# Prepare data for glmnet

x_train <- model.matrix(TARGET_deathRate ~ . - 1, data = train_data)

y_train <- train_data$TARGET_deathRate

x_test <- model.matrix(TARGET_deathRate ~ . - 1, data = test_data)

y_test <- test_data$TARGET_deathRate


# Ridge Regression

ridge_model_v2 <- cv.glmnet(x_train, y_train, alpha = 0)

ridge_best_lambda_v2 <- ridge_model_v2$lambda.min

ridge_predictions_v2 <- predict(ridge_model_v2, s = ridge_best_lambda_v2, newx =
x_test)

ridge_rsq_v2 <- rsq(y_test, ridge_predictions_v2)

ridge_rmse_v2 <- sqrt(mean((y_test - ridge_predictions_v2)^2))


# Lasso Regression

lasso_model_v2 <- cv.glmnet(x_train, y_train, alpha = 1)

lasso_best_lambda_v2 <- lasso_model_v2$lambda.min

lasso_predictions_v2 <- predict(lasso_model_v2, s = lasso_best_lambda_v2, newx =
x_test)

lasso_rsq_v2 <- rsq(y_test, lasso_predictions_v2)

lasso_rmse_v2 <- sqrt(mean((y_test - lasso_predictions_v2)^2))
```

Purpose

This code fits Ridge and Lasso regression models to the training data and evaluates their performance on the test data.

Output and Interpretation

- **Ridge Regression**:
    - **R-squared (ridge_rsq_v2)**: 0.4809793
    - **RMSE (ridge_rmse_v2)**: 20.41895
    - **Interpretation**: Ridge regression slightly underperforms compared to the enhanced linear model, with a marginally lower R-squared and higher RMSE.
- **Lasso Regression**:
    - **R-squared (lasso_rsq_v2)**: 0.4852762
    - **RMSE (lasso_rmse_v2)**: 20.32432
    - **Interpretation**: Lasso regression performs comparably to the enhanced linear model, with similar R-squared and RMSE values.

Final Comparison of Results

```
# Compare the results
results_v2 <- list(
  Enhanced_Model_v2 = list(R_squared = test_rsq_v2, RMSE = test_rmse_v2),
  Ridge_v2 = list(R_squared = ridge_rsq_v2, RMSE = ridge_rmse_v2),
  Lasso_v2 = list(R_squared = lasso_rsq_v2, RMSE = lasso_rmse_v2)
)
results_v2
```

Output

```
$Enhanced_Model_v2
$Enhanced_Model_v2$R_squared
[1] 0.487504

$Enhanced_Model_v2$RMSE
[1] 20.27769

$Ridge_v2
```

$Ridge_v2$R_squared

s1

[1,] 0.4809793

$Ridge_v2$RMSE

[1] 20.41895

$Lasso_v2

$Lasso_v2$R_squared

s1

[1,] 0.4852762

$Lasso_v2$RMSE

[1] 20.32432

Interpretation

- The Enhanced Linear Model (R-squared: 0.487504, RMSE: 20.27769) performs slightly better than Ridge and Lasso regression models.
- The R-squared values for all models are similar, indicating comparable explanatory power.
- The RMSE values are also similar, indicating comparable prediction accuracy.
- Although there is no substantial improvement in model performance, the Enhanced Linear Model has a slight edge over Ridge and Lasso regression in terms of both R-squared and RMSE.

Overall, while the Enhanced Linear Model shows slightly better performance, further model refinement and additional feature engineering may be needed to achieve significant improvements.

PYTHON LANGUAGE

Input Code: Import Libraries and Load Dataset

```
import pandas as pd

import numpy as np

import scipy.stats as stats

import matplotlib.pyplot as plt

import seaborn as sns


# Define the file path

file_path = r"C:\Users\nihar\OneDrive\Desktop\Bootcamp\SCMA 632\DataSet\IPL_ball_by_ball_updated till 2024.csv"


# Load the dataset with specified data types

ipl_data = pd.read_csv(file_path, low_memory=False)


# Define relevant columns for filtering and analysis

batsman_column = 'Striker'

bowler_column = 'Bowler'

runs_column = 'runs_scored'

wickets_column = 'wicket_confirmation'  # Using the wicket_confirmation column for wickets taken


# Filter data for the assigned player (SP Narine)

player_name = 'SP Narine'

batting_data = ipl_data[ipl_data[batsman_column] == player_name]

bowling_data = ipl_data[ipl_data[bowler_column] == player_name]
```

Purpose of the Input Code

1. **Importing Libraries**: Import necessary libraries (pandas, numpy, scipy.stats, matplotlib.pyplot, seaborn) for data analysis and visualization.
2. **Defining File Path**: Specify the path to the dataset file.
3. **Loading Dataset**: Load the IPL dataset into a pandas DataFrame.

4. **Defining Columns**: Specify the column names for filtering and analysis (batsman, bowler, runs, and wickets).
5. **Filtering Data**: Filter the dataset to extract data for the player SP Narine, both as a batsman and as a bowler.

Input Code: Analyze Batting Performance

```
# Analyze batting performance (runs scored)
if batting_data.empty:
  print(f"No batting data found for player: {player_name}")
else:
  runs_scored = batting_data[runs_column].replace([np.inf, -np.inf], np.nan).dropna()
  if runs_scored.empty:
    print(f"Runs scored data is empty after cleaning for player: {player_name}")
  else:
    (mu_batting, sigma_batting) = stats.norm.fit(runs_scored)
    print(f"Batting Mean (mu): {mu_batting}")
    print(f"Batting Standard Deviation (sigma): {sigma_batting}")
    plt.figure(figsize=(10, 6))
    sns.histplot(runs_scored, bins=range(int(runs_scored.min()),
int(runs_scored.max()) + 2), kde=False, color='yellow', stat='density', label='Runs
Scored')
    x = np.linspace(runs_scored.min(), runs_scored.max(), 100)
    p = stats.norm.pdf(x, mu_batting, sigma_batting)
    plt.plot(x, p, 'k', linewidth=2, label='Normal fit')
    plt.xlabel('Runs Scored')
    plt.ylabel('Density')
    plt.title(f'Fit of Normal Distribution to Runs Scored by {player_name}')
    plt.legend()
    plt.show()
```

Purpose of the Input Code

1. **Check for Empty Data**: Ensure that there is batting data for SP Narine.
2. **Clean Data**: Replace infinite values with NaN and drop NaN values.
3. **Fit Normal Distribution**: Fit a normal distribution to the runs_scored data and calculate the mean (mu_batting) and standard deviation (sigma_batting).
4. **Plot Histogram and Distribution**: Plot a histogram of runs_scored and overlay the fitted normal distribution.

Output

Batting Mean (mu): 1.5480349344978166

Batting Standard Deviation (sigma): 2.038088855321498

Interpretation

- **Mean (mu)**: The average runs scored by SP Narine in the dataset is approximately 1.55.
- **Standard Deviation (sigma)**: The runs scored have a standard deviation of approximately 2.04, indicating variability in the runs scored by SP Narine.

Input Code: Cumulative Distribution Function (CDF) Plot

```
# Cumulative Distribution Function (CDF)

plt.figure(figsize=(10, 6))

sns.kdeplot(runs_scored, cumulative=True, color='skyblue', linewidth=2, label='Empirical CDF')

x = np.linspace(runs_scored.min(), runs_scored.max(), 100)

p = stats.norm.cdf(x, mu_batting, sigma_batting)

plt.plot(x, p, 'k--', linewidth=2, label='Normal fit (CDF)')

plt.xlabel('Runs Scored')

plt.ylabel('Cumulative Probability')

plt.title(f'CDF of Runs Scored by {player_name}')

plt.legend()

plt.grid(True)
```

plt.show()

Purpose of the Input Code

1. **Plot Empirical CDF**: Plot the empirical cumulative distribution function (CDF) of runs_scored.
2. **Plot Normal CDF**: Overlay the CDF of the fitted normal distribution.

Input Code: Quantile-Quantile (Q-Q) Plot

Quantile-Quantile (Q-Q) plot

plt.figure(figsize=(10, 6))

stats.probplot(runs_scored, dist="norm", plot=plt)

plt.title(f'Q-Q Plot of Runs Scored by {player_name}')

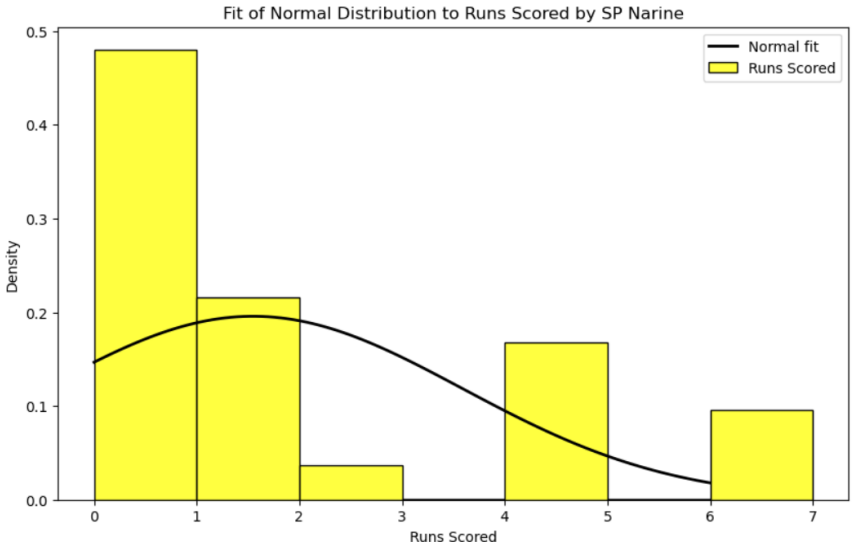
plt.grid(True)

plt.show()

Purpose of the Input Code

1. **Q-Q Plot**: Generate a Quantile-Quantile (Q-Q) plot to compare the distribution of runs_scored against a normal distribution.

Histogram with Fitted Normal Distribution

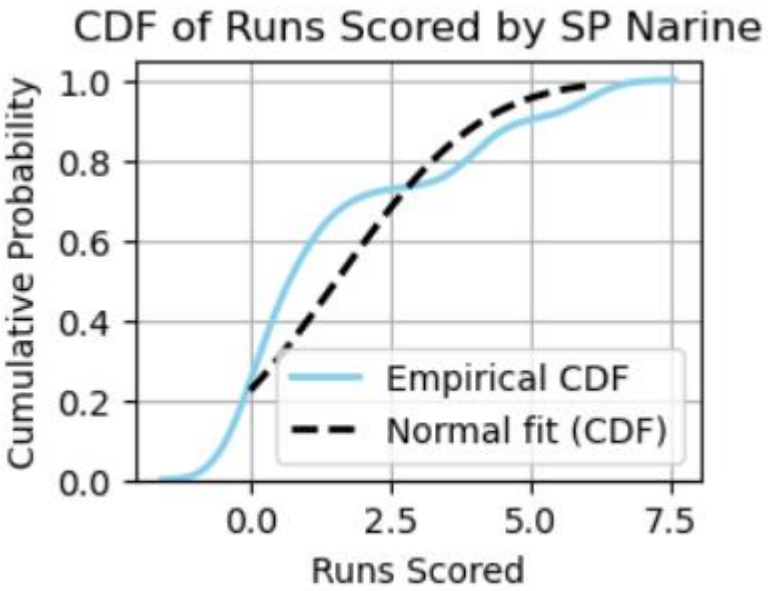


Interpretation:

- **Histogram**: This plot displays the frequency of runs scored by SP Narine.

- **Fitted Normal Distribution**: The black curve represents the normal distribution fitted to the data.
- **Mean (mu)**: The average runs scored by SP Narine is approximately 1.55.
- **Standard Deviation (sigma)**: The variability in runs scored is approximately 2.04.
- **Observation**: The histogram shows a high frequency of runs scored close to zero and one. The fitted normal distribution curve does not perfectly match the histogram, indicating that the normal distribution may not be the best fit for this data.

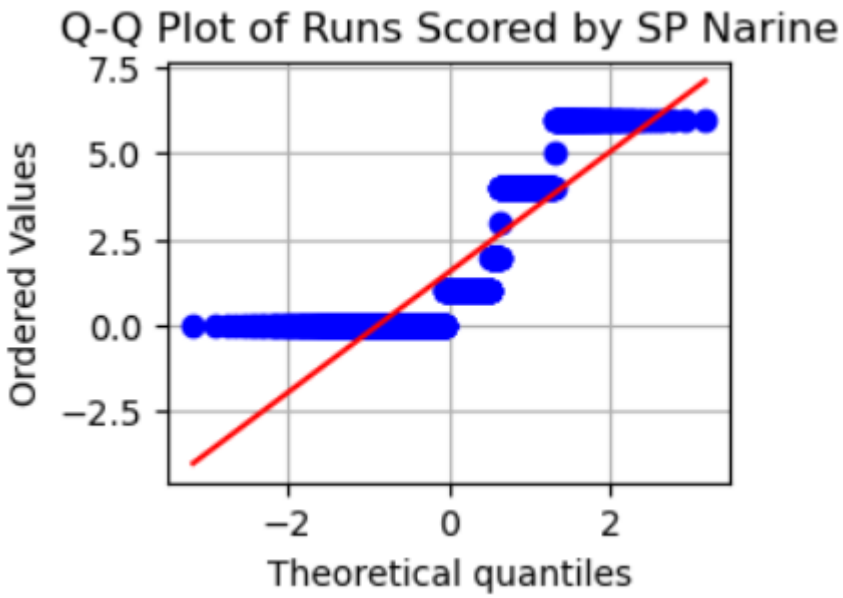
Cumulative Distribution Function (CDF) Plot



Interpretation:

- **Empirical CDF**: The blue curve shows the cumulative probability of the runs scored.
- **Normal CDF**: The dashed black curve represents the cumulative distribution function of the fitted normal distribution.
- **Observation**: The empirical CDF rises more steeply than the normal CDF at lower values and flattens out at higher values. This discrepancy indicates that the normal distribution does not fully capture the distribution of runs scored by SP Narine, especially at the extremes.

Quantile-Quantile (Q-Q) Plot



Interpretation:

- **Q-Q Plot**: This plot compares the quantiles of the runs scored data against the theoretical quantiles of a normal distribution.
- **Reference Line**: The red line represents where the points would lie if the data followed a perfect normal distribution.
- **Observation**: The points deviate from the reference line, especially at the lower and upper ends. This indicates that the runs scored by SP Narine do not follow a normal distribution well. The deviations at the ends suggest that the data has more extreme values (both low and high) than would be expected under a normal distribution.