Codes

Python Code (Exported from IPYNB)

```
# %%
# ## 1. Suppress Warnings
import warnings
warnings.filterwarnings("ignore", category=RuntimeWarning)

# %%
# ## 2. Import Required Libraries
import pandas as pd
import os
import scipy.stats as st
from fuzzywuzzy import process

# %%
# ## 3. Set Paths for Dataset Files
BASE = os.getcwd()
DATASETS_DIR = os.path.join(BASE, 'datasets')

PERF_DATASET_NAME = 'IPL_ball_by_ball_updated till 2024.csv'
PERF_DATASET_PATH = os.path.join(DATASETS_DIR, PERF_DATASET_NAME)

SALARY_DATASET_NAME = 'IPL SALARIES 2024.xlsx'
SALARY_DATASET_PATH = os.path.join(DATASETS_DIR, SALARY_DATASET_NAME)

# %%
# ## 4. Read IPL Performance Data
df = pd.read_csv(PERF_DATASET_PATH, low_memory=False)

# %%
df.head(3)  # Preview the first few rows

# %%
df.columns  # Check column names

# %%
df.info()  # Summary info about columns and types

# %%
# ## 5. Drop Unnecessary Columns
df.drop([
    "Batting team", "Bowling team", "Ball No", "Non Striker", "extras", "score",
    "score/wicket", "type of extras", "wicket_type", "fielders_involved", "Player
Out"
], axis=1, inplace=True)

# %%
# ## 6. Extract Year from Date Column
```

```
df["Year"] = pd.to_datetime(df["Date"], format="%d-%m-%Y").dt.year

# %%
# ## 7. Aggregate Runs and Wickets by Year, Innings, Player, Match
a1 = df.groupby(['Year', 'Innings No', 'Striker', 'Bowler']) \
        .agg({"runs_scored": "sum", "wicket_confirmation": "sum"}) \
        .reset_index()

# %%
runs = a1.groupby(['Year', 'Innings No', 'Striker']) \
         .agg({"runs_scored": "sum"}).reset_index()
wickets = a1.groupby(['Year', 'Innings No', 'Bowler']) \
            .agg({"wicket_confirmation": "sum"}).reset_index()

# %%
# ## 8. Identify Top 3 Players Each Year
seasons = runs["Year"].unique()
print(seasons)

# %%
for season in seasons:
    runs_season = runs[runs["Year"] == season]
    wickets_season = wickets[wickets["Year"] == season]

    print(f"Year: {season}\n")
    print("Top 3 Run Scorers:")
    print(runs_season.sort_values(by="runs_scored", ascending=False).head(3)
[['Striker', 'runs_scored']])

    print("\nTop 3 Wicket Takers:")
    print(wickets_season.sort_values(by="wicket_confirmation",
ascending=False).head(3)[['Bowler', 'wicket_confirmation']])
    print("\n" + "="*50 + "\n")

# %%
# ## 9. Prepare Data for Distribution Fitting
runs_per_year = df.groupby(['Year', 'Striker'])
[['runs_scored']].sum().reset_index()
wickets_per_year = df.groupby(['Year', 'Bowler'])
[['wicket_confirmation']].sum().reset_index()
runs_per_year.sort_values(['Year', 'runs_scored'], ascending=False, inplace=True)
wickets_per_year.sort_values(['Year', 'wicket_confirmation'], ascending=False,
inplace=True)

# %%
last_3_seasons = runs_per_year['Year'].unique().tolist()[:3]

# %%
# ## 10. Dictionary for Top 3 Players (last 3 seasons)
top_3_dict = {}
for season in last_3_seasons:
    top_3_dict[season] = {
        "batsmen": {k: [] for k in runs_per_year[runs_per_year["Year"] == season]
["Striker"].to_list()[:3]},
```

```
        "bowlers": {k: [] for k in wickets_per_year[wickets_per_year["Year"] ==
season]["Bowler"].to_list()[:3]}
    }

# %%
# ## 11. List of Candidate Distributions for Fitting
distribs = [
    "alpha", "beta", "betaprime", "burr12", "crystalball", "dgamma", "dweibull",
"erlang",
    "exponnorm", "f", "fatiguelife", "gamma", "gengamma", "gumbel_l", "johnsonsb",
    "kappa4", "lognorm", "nct", "norm", "norminvgauss", "powernorm", "rice",
    "recipinvgauss", "t", "trapezoid", "truncnorm"
]

# %%
# ## 12. Function to Get Best Fit Distribution
def get_best_distrib(data):
    results = []
    params = {}
    for distrib in distribs:
        dist = getattr(st, distrib)
        param = dist.fit(data)
        params[distrib] = param
        _, p = st.kstest(data, distrib, args=param)
        results.append((distrib, p))

    best_distrib, best_p = max(results, key=lambda x: x[1])

    print(f"Best Fitting Distribution: {best_distrib}")
    print(f"Best P-Value: {best_p}")
    print(f"Params for Best Fit: {str(params[best_distrib])}")

    return [best_distrib, best_p, params[best_distrib]]

# %%
# ## 13. Fit Distributions for Top 3 Players Each Season
runs = df.groupby(['Year', 'Striker','Match id'])
[['runs_scored']].sum().reset_index()
wickets = df.groupby(['Year', 'Bowler', 'Match id'])
[['wicket_confirmation']].sum().reset_index()

for year in top_3_dict.keys():
    for striker in top_3_dict[year]["batsmen"].keys():
        print("************************\n")
        print(f"Year: {year} | Batsman: {striker}\n")
        top_3_dict[year]["batsmen"][striker] =
get_best_distrib(runs[runs['Striker'] == striker]['runs_scored'])
        print('\n\n')

    for bowler in top_3_dict[year]["bowlers"].keys():
        print("************************\n")
        print(f"Year: {year} | Bowler: {striker}\n")
        top_3_dict[year]["bowlers"][bowler] =
get_best_distrib(wickets[wickets['Bowler'] == bowler]['wicket_confirmation'])
```

```
        print('\n\n')

# %%
# ## 14. Assigned Player Distribution Fit (N Pooran)
my_player = "N Pooran"
my_data = runs[runs['Striker'] == my_player]['runs_scored']
get_best_distrib(my_data)

# %%
# ## 15. Total Stats by Year
total_runs_per_year = df.groupby(['Year', 'Striker'])
[['runs_scored']].sum().reset_index().sort_values(['Year', 'runs_scored'],
ascending=False)
total_wickets_per_year = df.groupby(['Year', 'Bowler'])
[['wicket_confirmation']].sum().reset_index().sort_values(['Year',
'wicket_confirmation'], ascending=False)

# %%
# ## 16. Load Salary Dataset
salary_df = pd.read_excel(SALARY_DATASET_PATH)

# %%
# ## 17. Filter 2024 Data
total_runs_2024 = total_runs_per_year[total_runs_per_year['Year'] == 2024]
total_wickets_2024 = total_wickets_per_year[total_wickets_per_year['Year'] ==
2024]

# %%
# ## 18. Fuzzy Matching for Player Names
def match_names(name, names_list, threshold=80):
    if not name or not isinstance(name, str):
        return None

    result = process.extractOne(name, names_list)
    if result:
        match, score = result # type: ignore
        return match if score >= threshold else None
    return None

# %%
# ## 19. Correlation for Batsmen
df_striker_salary = salary_df.copy()
df_striker_salary['Matched Player'] = df_striker_salary['Player'].apply(lambda x:
match_names(x, total_runs_2024['Striker'].tolist()))
df_striker_merged = pd.merge(df_striker_salary, total_runs_2024, left_on='Matched
Player', right_on='Striker')

corr_striker = df_striker_merged['Rs'].corr(df_striker_merged['runs_scored'])
print('Correlation between Salary and Runs in 2024:', corr_striker)

# %%
# ## 20. Correlation for Bowlers
df_bowler_salary = salary_df.copy()
df_bowler_salary['Matched Player'] = df_bowler_salary['Player'].apply(lambda x:
```

```
match_names(x, total_wickets_2024['Bowler'].tolist()))
df_bowler_merged = pd.merge(df_bowler_salary, total_wickets_2024, left_on='Matched Player', right_on='Bowler')

corr_bowler = df_bowler_merged['Rs'].corr(df_bowler_merged['wicket_confirmation'])
print('Correlation between Salary and Wickets in 2024:', corr_bowler)
```

R Code

```
# Set the base directory for the project
BASE <- "C:\\Users\\ujjwa\\Documents\\VCU\\Pre-Course\\SCMA632\\Assignments\\A2\\R"
setwd(BASE)  # Change working directory to where your data files are located
getwd()  # Confirm working directory

# Define a helper function to install packages if not already installed
install <- function(pkg) {
  if (!require(pkg, character.only = TRUE)) {
    install.packages(pkg, dependencies = TRUE, quiet = TRUE)
  }
}

# Define a helper function to load packages
load <- function(pkg) {
  library(pkg, character.only = TRUE, quietly = TRUE)
}

# Required packages for this analysis
pkgs <- c("dplyr", "readr", "readxl", "lubridate", "stringdist", "stats", "fitdistrplus")
lapply(pkgs, install)
lapply(pkgs, load)

# --- Data Import ---

# Read in the IPL performance and salary datasets

# Define file paths
perf_path <- "datasets/IPL_ball_by_ball_updated till 2024.csv"
salary_path <- "datasets/IPL SALARIES 2024.xlsx"

# Read the datasets
df <- read_csv(perf_path)
salary_df <- read_excel(salary_path)

# --- Data Preprocessing ---

# Select only relevant columns and extract the year from the date column
columns_to_select <- c("Match id", "Date", "Season", "Innings No",
                       "Bowler", "Striker", "runs_scored", "wicket_confirmation")
```

```r
df <- df %>% dplyr::select(all_of(columns_to_select))
df$Year <- year(dmy(df$Date))  # Extract year from the date

# --- Aggregate Data ---

# Summarize total runs by each batsman
runs <- df %>%
  group_by(Year, `Innings No`, Striker) %>%
  summarise(runs_scored = sum(runs_scored), .groups = 'drop')

# Summarize total wickets by each bowler
wickets <- df %>%
  group_by(Year, `Innings No`, Bowler) %>%
  summarise(wicket_confirmation = sum(wicket_confirmation), .groups = 'drop')

# --- Identify Top 3 Performers Each Year ---

years <- unique(runs$Year)
for (yr in years) {
  cat("Year:", yr, "\n\nTop 3 Run Scorers:\n")
  print(runs %>% filter(Year == yr) %>%
          group_by(Striker) %>%
          summarise(runs = sum(runs_scored), .groups = 'drop') %>%
          arrange(desc(runs)) %>% head(3))

  cat("\nTop 3 Wicket Takers:\n")
  print(wickets %>% filter(Year == yr) %>%
          group_by(Bowler) %>%
          summarise(wickets = sum(wicket_confirmation), .groups = 'drop') %>%
          arrange(desc(wickets)) %>% head(3))

  cat("\n", strrep("=", 50), "\n\n")
}

# --- Name Matching Between Datasets (Fuzzy Matching) ---

match_names <- function(name, choices, threshold = 0.2) {
  if (is.na(name)) return(NA)
  dists <- stringdist(name, choices, method = "jw")
  min_dist <- min(dists)
  if (min_dist <= threshold) return(choices[which.min(dists)])
  return(NA)
}

# --- Correlation Between Salary and Performance (2024) ---

runs_2024 <- df %>%
  filter(Year == 2024) %>%
  group_by(Striker) %>%
  summarise(runs_scored = sum(runs_scored), .groups = 'drop')

wickets_2024 <- df %>%
  filter(Year == 2024) %>%
  group_by(Bowler) %>%
```

```
  summarise(wicket_confirmation = sum(wicket_confirmation), .groups = 'drop')

salary_df$Matched_Striker <- sapply(salary_df$Player, match_names, choices =
runs_2024$Striker)
salary_df$Matched_Bowler <- sapply(salary_df$Player, match_names, choices =
wickets_2024$Bowler)

striker_merged <- merge(salary_df, runs_2024, by.x = "Matched_Striker", by.y =
"Striker")
bowler_merged <- merge(salary_df, wickets_2024, by.x = "Matched_Bowler", by.y =
"Bowler")

cor_striker <- cor(striker_merged$Rs, striker_merged$runs_scored, use =
"complete.obs")
cor_bowler <- cor(bowler_merged$Rs, bowler_merged$wicket_confirmation, use =
"complete.obs")

cat("\nCorrelation between Salary and Runs in 2024:", cor_striker)
cat("\nCorrelation between Salary and Wickets in 2024:", cor_bowler)

# --- Distribution Fitting for Assigned Player: N Pooran ---

n_pooran_runs <- df %>%
  group_by(Year, Striker, `Match id`) %>%
  summarise(runs_scored = sum(runs_scored), .groups = 'drop') %>%
  filter(Striker == "N Pooran") %>%
  pull(runs_scored)

n_pooran_runs_pos <- n_pooran_runs[n_pooran_runs > 0]

fit_norm <- fitdist(n_pooran_runs, "norm")
fit_gamma <- fitdist(n_pooran_runs, "gamma")
fit_exp <- fitdist(n_pooran_runs, "exp")
fit_lnorm <- fitdist(n_pooran_runs_pos, "lnorm")

gof <- gofstat(list(fit_norm, fit_gamma, fit_exp))
gof_lnorm <- if (length(n_pooran_runs_pos) == length(n_pooran_runs))
gofstat(list(fit_lnorm)) else NULL

print(gof)
cat("\nLognormal fit (positive data only):\n")
print(gof_lnorm)

ks <- ks.test(n_pooran_runs, "pgamma", shape = fit_gamma$estimate["shape"], rate =
fit_gamma$estimate["rate"])
print(ks)

# --- Distribution Fitting for Top 3 Batsmen and Bowlers (Last 3 Seasons) ---

get_best_fit <- function(data) {
  if (length(data) < 2) return(list(best_dist = "Too few data", gof = NULL))
  data_pos <- data[data > 0]
  use_pos <- all(data > 0)
  data_used <- if (use_pos) data else data_pos
```

```
  fits <- list(
    norm = tryCatch(fitdist(data_used, "norm"), error = function(e) NULL),
    gamma = tryCatch(fitdist(data_used, "gamma"), error = function(e) NULL),
    exp = tryCatch(fitdist(data_used, "exp"), error = function(e) NULL)
  )
  if (all(data_used > 0)) {
    fits$lnorm <- tryCatch(fitdist(data_used, "lnorm"), error = function(e) NULL)
  }

  fits <- Filter(Negate(is.null), fits)

  if (length(fits) <= 1) return(list(best_dist = names(fits), gof = NULL))

  lengths <- sapply(fits, function(f) length(f$data))
  mode_len <- as.numeric(names(which.max(table(lengths))))
  fits_same <- fits[lengths == mode_len]

  if (length(fits_same) > 1) {
    gof <- tryCatch(gofstat(fits_same), error = function(e) NULL)
    if (!is.null(gof)) {
      best <- names(which.max(gof$ks))
      return(list(best_dist = best, gof = gof))
    }
  }

  return(list(best_dist = "N/A", gof = NULL))
}

last_3_seasons <- sort(unique(df$Year), decreasing = TRUE)[1:3]

df_runs <- df %>%
  group_by(Year, Striker, `Match id`) %>%
  summarise(runs_scored = sum(runs_scored), .groups = 'drop')

df_wickets <- df %>%
  group_by(Year, Bowler, `Match id`) %>%
  summarise(wicket_confirmation = sum(wicket_confirmation), .groups = 'drop')

for (year in last_3_seasons) {
  cat("\n===== Year:", year, "=====")

  top_batsmen <- df_runs %>%
    filter(Year == year) %>%
    group_by(Striker) %>%
    summarise(total_runs = sum(runs_scored), .groups = 'drop') %>%
    arrange(desc(total_runs)) %>%
    slice(1:3) %>% pull(Striker)

  top_bowlers <- df_wickets %>%
    filter(Year == year) %>%
    group_by(Bowler) %>%
    summarise(total_wickets = sum(wicket_confirmation), .groups = 'drop') %>%
    arrange(desc(total_wickets)) %>%
```

```
    slice(1:3) %>% pull(Bowler)

  cat("\nTop 3 Batsmen Distribution Fits:\n")
  for (batsman in top_batsmen) {
    player_data <- df_runs %>% filter(Year == year, Striker == batsman) %>%
pull(runs_scored)
    result <- get_best_fit(player_data)
    cat("-", batsman, ": Best Fit:", result$best_dist, "\n")
  }

  cat("\nTop 3 Bowlers Distribution Fits:\n")
  for (bowler in top_bowlers) {
    player_data <- df_wickets %>% filter(Year == year, Bowler == bowler) %>%
pull(wicket_confirmation)
    result <- get_best_fit(player_data)
    cat("-", bowler, ": Best Fit:", result$best_dist, "\n")
  }
}
```