**Project - Sports Analytics and Performance Monitoring Using Machine Learning**

**Project by:**

**Ms. Suyesha Santosh Patil**

**Overview:**

This project uses **machine learning (ML)** to help athletes and sports teams improve performance and optimize game strategies. By analyzing data from **wearables**, **GPS trackers**, and **video footage**, ML models provide insights to guide training and make better decisions during games.

**Key Objectives:**

1. **Predict Performance**: Using historical and real-time data, ML models help predict how well athletes will perform.
2. **Real-Time Adjustments**: During games or training, the system provides real-time suggestions to improve player performance.
3. **Personalized Training**: Training programs are tailored for each athlete based on their unique data, such as fitness levels or game performance.

**Methodology:**

* **Data Collection**: Gather data from wearables, GPS, and videos.
* **Model Development**: Build ML models to predict performance outcomes.
* **Real-Time Monitoring**: Track athlete performance and provide real-time adjustments.
* **Personalized Programs**: Create custom training plans for individual athletes.
* **Power BI Dashboard**: A **Power BI dashboard** was developed to visualize the results and insights for better decision-making.

**Tools Used:**

* **Pandas** for data handling
* **Matplotlib/Seaborn** for creating charts
* **TensorFlow/Keras** for building models
* **OpenCV** for video analysis

**Conclusion:**

By applying machine learning and visualizing data with **Power BI**, this project helps athletes and teams make data-driven decisions to improve performance and strategies during games.

Code->

1.Creating a csv file🡪

import csv

# Define the header and sample data

header = [

"Player Name", "Match Date", "Opponent", "Runs Scored",

"Balls Faced", "Fours", "Sixes", "Strike Rate",

"Wickets Taken", "Bowling Overs", "Runs Conceded",

"Economy Rate", "Role", "Performance Score", "Win Probability"

]

data = [

["Virat Kohli", "2024-09-15", "Australia", 85, 105, 7, 1, 80.95, 0, 0, 0, 0.00, "Batsman", 90, 0.75],

["Rohit Sharma", "2024-09-15", "Australia", 45, 50, 4, 2, 90.00, 0, 0, 0, 0.00, "Batsman", 70, 0.60],

["Jasprit Bumrah", "2024-09-15", "Australia", 0, 0, 0, 0, 0.00, 3, 10, 30, 3.00, "Bowler", 85, 0.70],

["Shreyas Iyer", "2024-09-15", "Australia", 60, 70, 5, 1, 85.71, 0, 0, 0, 0.00, "Batsman", 80, 0.65],

["KL Rahul", "2024-09-14", "South Africa", 70, 80, 6, 3, 87.50, 1, 10, 45, 4.50, "Batsman", 75, 0.68],

["Ravindra Jadeja", "2024-09-14", "South Africa", 25, 30, 2, 0, 83.33, 2, 8, 35, 4.38, "All-Rounder", 78, 0.72],

["Hardik Pandya", "2024-09-14", "South Africa", 40, 45, 3, 1, 88.89, 1, 7, 50, 7.14, "All-Rounder", 80, 0.69],

["Shubman Gill", "2024-09-13", "New Zealand", 90, 95, 8, 4, 94.74, 0, 0, 0, 0.00, "Batsman", 88, 0.74],

["Bhuvneshwar Kumar", "2024-09-13", "New Zealand", 0, 0, 0, 0, 0.00, 2, 10, 25, 2.50, "Bowler", 82, 0.65],

["Ishan Kishan", "2024-09-13", "New Zealand", 30, 35, 3, 1, 85.71, 0, 0, 0, 0.00, "Wicket-Keeper", 72, 0.60],

# Add more records as needed

]

# Write data to CSV file

with open('cricket\_stats\_with\_predictions.csv', mode='w', newline='') as file:

writer = csv.writer(file)

writer.writerow(header) # Write header

writer.writerows(data) # Write data

print("CSV file 'cricket\_stats\_with\_predictions.csv' created successfully.")

df.head()

2.to check which player played best in the match🡪

import pandas as pd

# Load the CSV file into a DataFrame

df = pd.read\_csv('cricket\_stats\_with\_predictions.csv')

# Display the DataFrame (optional)

print("DataFrame:")

print(df)

# Find the player with the highest runs scored

best\_batsman = df.loc[df['Runs Scored'].idxmax()]

# Print the best player details

print("\nBest Player in the Match:")

print(best\_batsman)

Output🡪

DataFrame:

Player Name Match Date Opponent Runs Scored Balls Faced \

0 Virat Kohli 2024-09-15 Australia 85 105

1 Rohit Sharma 2024-09-15 Australia 45 50

2 Jasprit Bumrah 2024-09-15 Australia 0 0

3 Shreyas Iyer 2024-09-15 Australia 60 70

4 KL Rahul 2024-09-14 South Africa 70 80

5 Ravindra Jadeja 2024-09-14 South Africa 25 30

6 Hardik Pandya 2024-09-14 South Africa 40 45

7 Shubman Gill 2024-09-13 New Zealand 90 95

8 Bhuvneshwar Kumar 2024-09-13 New Zealand 0 0

9 Ishan Kishan 2024-09-13 New Zealand 30 35

Fours Sixes Strike Rate Wickets Taken Bowling Overs Runs Conceded \

0 7 1 80.95 0 0 0

1 4 2 90.00 0 0 0

2 0 0 0.00 3 10 30

3 5 1 85.71 0 0 0

4 6 3 87.50 1 10 45

5 2 0 83.33 2 8 35

6 3 1 88.89 1 7 50

7 8 4 94.74 0 0 0

8 0 0 0.00 2 10 25

9 3 1 85.71 0 0 0

Economy Rate Role Performance Score Win Probability

0 0.00 Batsman 90 0.75

1 0.00 Batsman 70 0.60

2 3.00 Bowler 85 0.70

3 0.00 Batsman 80 0.65

4 4.50 Batsman 75 0.68

5 4.38 All-Rounder 78 0.72

6 7.14 All-Rounder 80 0.69

7 0.00 Batsman 88 0.74

8 2.50 Bowler 82 0.65

9 0.00 Wicket-Keeper 72 0.60

Best Player in the Match:

Player Name Shubman Gill

Match Date 2024-09-13

Opponent New Zealand

Runs Scored 90

Balls Faced 95

Fours 8

Sixes 4

Strike Rate 94.74

Wickets Taken 0

Bowling Overs 0

Runs Conceded 0

Economy Rate 0.0

Role Batsman

Performance Score 88

Win Probability 0.74

Name: 7, dtype: object

1. Model Development

We'll create a machine learning model to predict a player's performance score based on their match statistics.

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Split data into features and target variable

X = df[['Runs Scored', 'Balls Faced', 'Fours', 'Sixes', 'Wickets Taken', 'Bowling Overs', 'Runs Conceded']]

y = df['Performance Score']

# Split the dataset 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)

# Initialize and train the model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

print(f'Mean Squared Error: {mse:.2f}')

Output-> Mean Squared Error: 279.83

1. Real-Time Monitoring and Adjustments🡪

To provide real-time suggestions, you would typically set up a system that continuously inputs data and outputs recommendations based on the model’s

def predict\_performance(runs\_scored, balls\_faced, fours, sixes, wickets\_taken, bowling\_overs, runs\_conceded):

    input\_data = np.array([[runs\_scored, balls\_faced, fours, sixes, wickets\_taken, bowling\_overs, runs\_conceded]])

    prediction = model.predict(input\_data)

    return prediction[0]

def real\_time\_monitoring():

    print("Real-Time Monitoring of Player Performance")

    print("Please enter the following statistics:")

    # Input current match statistics

    runs\_scored = int(input("Runs Scored: "))

    balls\_faced = int(input("Balls Faced: "))

    fours = int(input("Fours: "))

    sixes = int(input("Sixes: "))

    wickets\_taken = int(input("Wickets Taken: "))

    bowling\_overs = int(input("Bowling Overs: "))

    runs\_conceded = int(input("Runs Conceded: "))

    # Predict performance

    predicted\_score = predict\_performance(runs\_scored, balls\_faced, fours, sixes, wickets\_taken, bowling\_overs, runs\_conceded)

    # Provide recommendations

    print(f"\nPredicted Performance Score: {predicted\_score:.2f}")

    if predicted\_score > 75:

        print("Recommendation: Maintain current performance and keep attacking.")

    elif 50 < predicted\_score <= 75:

        print("Recommendation: Focus on consolidating and building partnerships.")

    else:

        print("Recommendation: Review your strategies and focus on improving key metrics.")

# Example usage

real\_time\_monitoring()

Output-> through user input

Real-Time Monitoring of Player Performance

Please enter the following statistics:

Runs Scored: 20

Balls Faced: 5

Fours: 9

Sixes: 4

Wickets Taken: 2

Bowling Overs: 9

Runs Conceded: 6

1. Visualizing Results🡪

import matplotlib.pyplot as plt

import seaborn as sns

# Visualize the predictions vs actual performance

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

sns.scatterplot(x=y\_test, y=y\_pred)

plt.xlabel('Actual Performance')

plt.ylabel('Predicted Performance')

plt.title('Predicted vs Actual Performance')

plt.plot([50, 100], [50, 100], color='red') # Line for perfect prediction

plt.show()

output->



Connecting power Bi and google colab🡪

1. # Download the file

#from google.colab import files

#files.download('cricket\_stats\_with\_predictions.csv')

2.goto drive save this file in drive

Run this code in colab🡪

3. from google.colab import drive

import pandas as pd

# Mount Google Drive to Colab

drive.mount('/content/drive')

# Define the file path in Google Drive

csv\_file\_path = '/content/drive/MyDrive/cricket\_stats\_with\_predictions.csv'

# Convert data to DataFrame

df = pd.DataFrame(data)

# Save DataFrame as CSV in Google Drive

df.to\_csv(csv\_file\_path, index=False)

print(f"Data saved to {csv\_file\_path}")

### Mount Google Drive in Google Colab

You need to store your generated data (e.g., a CSV file) in Google Drive.

### Step 2: Make the CSV File Publicly Accessible

To allow Power BI to access the CSV file, you need to make the file publicly accessible or share it with a specific link.

1. Go to [Google Drive](https://drive.google.com).
2. Locate the file (colab\_data.csv), right-click it, and select **Share**.
3. Change the sharing settings to **Anyone with the link can view**.
4. Copy the link to the file.

Now, the file is accessible via a direct URL, which you’ll use in Power BI.

### Step 3: Import CSV into Power BI

1. **Open Power BI Desktop**.
2. Click on **Get Data** → **Web**.
3. Paste the link you copied from Google Drive but modify it slightly:
   * Replace the https://drive.google.com/file/d/FILE\_ID/view?usp=sharing link with https://drive.google.com/uc?id=FILE\_ID&export=download.

For example, if your Google Drive link is:(this below link is of our projects)

bash

Copy code

https://drive.google.com/file/d/1r1AMFRDeZ\_6G5Niwf08t0HDGLTYGQrgW/view?usp=sharing

The modified link will be:

bash

Copy code->

https://drive.google.com/uc?id=1r1AMFRDeZ\_6G5Niwf08t0HDGLTYGQrgW&export=download

1. Power BI will import the CSV data.

### Step 4: Enable Scheduled Refresh in Power BI

To ensure your Power BI dashboard reflects real-time changes from the Colab program, enable the refresh option:

1. **Upload to Power BI Service**:
   * After creating the Power BI report, save and publish it to Power BI Service (cloud).
   * Click on **File** → **Publish** → **Publish to Power BI Service**.
   * Name of our file saved is🡪 criket\_dashboard
2. **Configure Scheduled Refresh**:
   * Go to Power BI Service (https://app.powerbi.com).
   * Navigate to **Datasets**.
   * Find your dataset and click on the **Schedule Refresh** option.
   * Set up the refresh frequency (e.g., every hour or once a day).

Power BI will now automatically pull updated data from the Google Drive CSV file at the scheduled intervals.

### Step 5: Sync Google Colab Data Updates with Power BI

Whenever your Google Colab program modifies the CSV file, those updates will be reflected in Power BI during the next refresh cycle.

For example, if you update your CSV file in Google Colab:

python

Copy code

# Update your data in Google Colab

df['Runs Scored'] = [100, 50, 0, 0, 0, 0, 0, 0, 0, 0] # Update runs scored with a list of 10 values to match the DataFrame length. You can change these values as needed.

df.to\_csv(csv\_file\_path, index=False)

print(f"Updated data saved to {csv\_file\_path}")

The next time Power BI refreshes its data from the file, these updates will be reflected in the dashboard.

### Recap:

1. **Mount Google Drive in Google Colab** and save your data (CSV file).
2. **Share the Google Drive CSV file** and modify the link for Power BI.
3. **Connect Power BI** to the CSV file via the **Get Data → Web** option.
4. **Enable Scheduled Refresh** in Power BI Service to keep data up-to-date.

This workflow keeps your Google Colab data in sync with your Power BI dashboard.

\*\*\*\* goto powerbi on a 🡪 cricket\_dashboard \*\*\*\*