**Statistical Testing**

1. **Player Performance Evaluation**

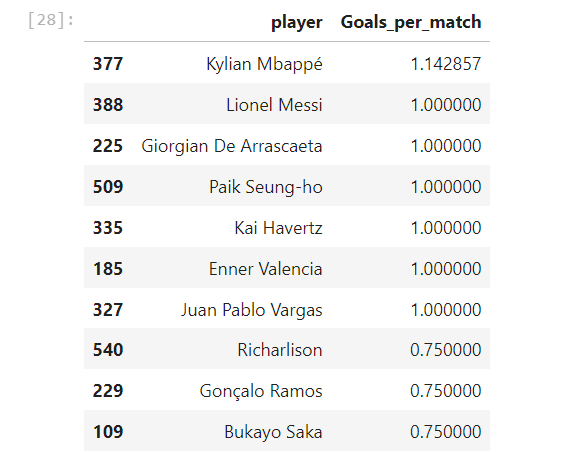
# Calculate goals per match

players['Goals\_per\_match'] = players['goals'] / players['games']

# Display top scorers based on goals per match

top\_scorers = players.sort\_values(by='Goals\_per\_match', ascending=False).head(10)

top\_scorers[['player', 'Goals\_per\_match']]

****

1. **Predicting Match Outcomes**

# Since matches have no result we use a proxy for Match\_Result based on goals

# This is a simplified approach to adjust it as needed for our data.

teams['Match\_Result'] = (teams['goals'] > teams['goals'].mean()).astype(int)

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# Calculate Pass Accuracy

teams['Pass\_Accuracy'] = (teams['passes\_completed'] / teams['passes']) \* 100

# Prepare data for training

features = teams[['goals', 'possession', 'Pass\_Accuracy']]

labels = teams['Match\_Result']

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, labels, test\_size=0.2, random\_state=42)

# Train the model

model = LogisticRegression()

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

# Test the model

predictions = model.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, predictions))



1. **Improved Tactical Understanding**

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

# Calculate Pass Accuracy

teams['Pass\_Accuracy'] = (teams['passes\_completed'] / teams['passes']) \* 100

# Apply clustering (excluding Goals\_Conceded if not available)

kmeans = KMeans(n\_clusters=4, random\_state=0).fit(teams[['possession', 'Pass\_Accuracy']])

teams['Play\_Style'] = kmeans.labels\_

# Visualize clusters

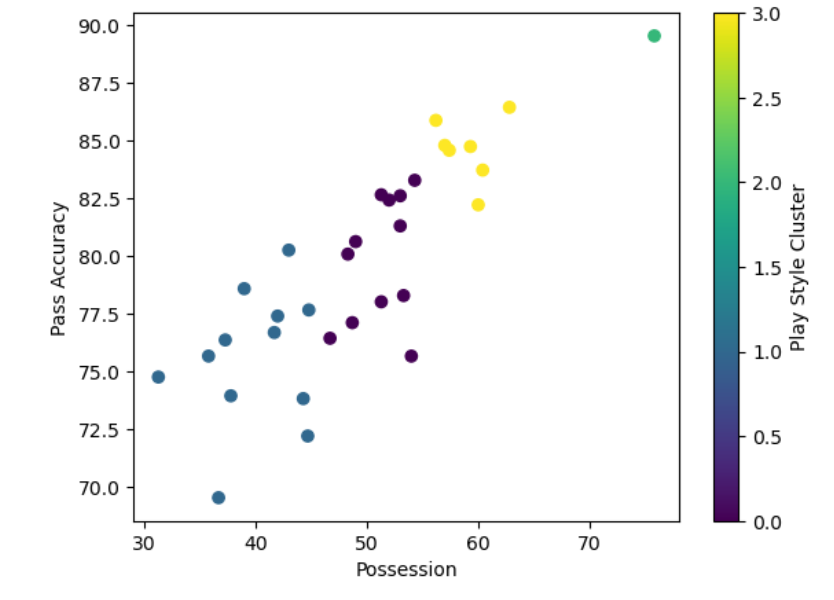
plt.scatter(teams['possession'], teams['Pass\_Accuracy'], c=teams['Play\_Style'], cmap='viridis')

plt.xlabel('Possession')

plt.ylabel('Pass Accuracy')

plt.colorbar(label='Play Style Cluster')

plt.show()

****

1. **Team Management and Player Selection**

# Check players with position as 'Forward'

forwards = players[players['position'] == 'FW']

print("Total forwards:", len(forwards))

# Check players with Goals per Match greater than 0.3

high\_scorers = players[players['Goals\_per\_match'] > 0.3]

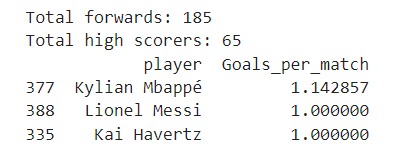
print("Total high scorers:", len(high\_scorers))

# Try a lower threshold for Goals per Match

selected\_players = players[(players['position'] == 'FW') & (players['Goals\_per\_match'] > 0.1)]

selected\_players = selected\_players.sort\_values(by='Goals\_per\_match', ascending=False).head(3)

print(selected\_players[['player', 'Goals\_per\_match']])

****

1. **World Cup Wins Analysis**

# Count the number of World Cups each country has won

world\_cup\_wins = world\_cups['Winner'].value\_counts()

print(world\_cup\_wins)

# Plot world cup wins

world\_cup\_wins.plot(kind='bar', title="World Cup Wins by Country")

plt.xlabel("Country")

plt.ylabel("World Cups Won")

plt.show()

