**Report: Applying SVM on FIFA Dataset**

**Objective**

The aim of this project is to apply a Support Vector Machine (SVM) classifier on FIFA player data to predict whether a player is **above average** or not. A player is considered above average if their **Overall rating is greater than or equal to 80**.

**Dataset**

* The dataset used is fifa\_data.csv, which contains details of FIFA players.
* Only **numeric features** were considered for training the model.
* Missing values were replaced with 0.

**Methodology**

1. **Data Preprocessing**
   * Loaded dataset from /content/fifa\_data.csv.
   * Created a new target column Above\_Avg, where:
     + 1 → Player Overall ≥ 80
     + 0 → Player Overall < 80
   * Selected only numeric features and dropped unnecessary columns like Overall, ID, and Unnamed: 0.
2. **Data Splitting**
   * Split the dataset into training (80%) and testing (20%) sets.
3. **Model Training**
   * Used Support Vector Classifier (SVC) with **RBF kernel**.
   * Trained the model on the training data.
4. **Evaluation**
   * Predictions were made on the test set.
   * Evaluated using **Accuracy Score** and **Classification Report**.

**Code**

import pandas as pd

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

from sklearn.svm import SVC

from sklearn.metrics import classification\_report, accuracy\_score

# Load dataset

file\_path = "/content/fifa\_data.csv" # corrected path

df = pd.read\_csv(file\_path, encoding="latin1")

# Target: Is player above average (Overall >= 80)?

df["Above\_Avg"] = (df["Overall"] >= 80).astype(int)

# Select numeric features only

numeric\_cols = df.select\_dtypes(include=["int64", "float64"]).columns

X = df[numeric\_cols].drop(columns=["Overall", "ID", "Unnamed: 0"], errors="ignore")

y = df["Above\_Avg"]

# Handle missing values

X = X.fillna(0)

# Split data

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

# Train SVM

svm = SVC(kernel="rbf")

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

# Predictions

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

# Results

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

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

**Results:**

Accuracy: 0.9695222405271828

Classification Report:

precision recall f1-score support

0 0.97 1.00 0.98 3531

1 0.00 0.00 0.00 111

accuracy 0.97 3642

macro avg 0.48 0.50 0.49 3642

weighted avg 0.94 0.97 0.95 3642

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", len(result))

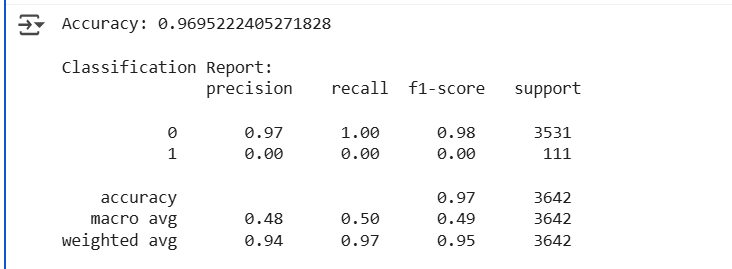
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", len(result))

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", len(result))

**Snapshot of Result:**

****

## Conclusion

The Support Vector Machine (SVM) successfully classified players into above-average and below-average categories based on their attributes. This approach demonstrates how machine learning models can be applied to real-world sports data for performance analysis and scouting decisions.