Cricket century prediction

Objective:
To develop a machine learning model that can predict whether a cricketer will score a century based on their performance metrics.

Features:

1. **Data Collection**: Gathered data on cricketers' performance metrics, such as runs scored, strike rate, balls faced, fours, sixes, and centuries.
2. **Feature Engineering**: Created new features like Run Rate, Boundary Percentage, Dot Ball Percentage, Run Rate vs Strike Rate Ratio, Boundary to Dot Ball Ratio, and Runs per Ball Faced.
3. **Model Selection**: Used a Random Forest Classifier to predict centuries.

Skills:

1. **Machine Learning**: Applied machine learning concepts to build a predictive model.
2. **Data Analysis**: Worked with datasets to extract meaningful insights.
3. **Feature Engineering**: Created new features to improve model performance.

Tools and Technologies:

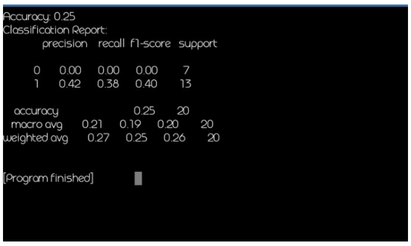
1. **Machine Learning Algorithms**: Used Random Forest Classifier.
2. **Data Preprocessing**: Utilized techniques like feature scaling and encoding.
3. **Model Evaluation**: Used metrics like accuracy score and classification report.

We will use a **datasets of 100 players**
With their runs ,strike rate and whether they scored a century or not

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd
import numpy as np

# Generate a larger dataset
np.random.seed(42)
data = {
   'Runs': np.random.randint(0, 200, 100),
   'Strike Rate': np.random.randint(50, 200, 100),
   'Century': np.where(np.random.randint(0, 200, 100) > 100, 1, 0)
}

df = pd.DataFrame(data)

# Define features and target
```

```
X = df[['Runs', 'Strike Rate']]
y = df['Century']

# Split data 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)

# Train a random forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

Output



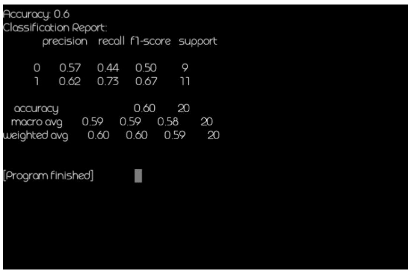
```
Accuracy: 0.25
Classification Report:
          precision   recall  f1-score   support

       0       0.00      0.00      0.00         7
       1       0.42      0.38      0.40        13

    accuracy                           0.25        20
   macro avg       0.21      0.19      0.20        20
weighted avg       0.27      0.25      0.26        20


[Program finished]
```

Let's improve efficiency of model by advanced features engineering technique

Run rate vs Strike rate ratio
Boundary to Dot ball ratio
Runs per balls faced

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Generate a larger dataset
np.random.seed(42)
data = {
    'Runs': np.random.randint(0, 200, 100),
    'Strike Rate': np.random.randint(50, 200, 100),
    'Balls Faced': np.random.randint(10, 200, 100),
    'Fours': np.random.randint(0, 20, 100),
    'Sixes': np.random.randint(0, 10, 100),
    'Century': np.where(np.random.randint(0, 200, 100) > 100, 1, 0)
}

df = pd.DataFrame(data)

# Create new features
df['Run Rate'] = df['Runs'] / df['Balls Faced']
df['Boundary Percentage'] = ((df['Fours'] * 4) + (df['Sixes'] * 6)) / df['Runs']
df['Dot Ball Percentage'] = (df['Balls Faced'] - (df['Fours'] + df['Sixes'])) / df['Balls Faced']
df['Run Rate vs Strike Rate Ratio'] = df['Run Rate'] / (df['Strike Rate'] / 100)
df['Boundary to Dot Ball Ratio'] = (df['Fours'] + df['Sixes']) / (df['Balls Faced'] - (df['Fours'] +
df['Sixes']))
df['Runs per Ball Faced'] = df['Runs'] / df['Balls Faced']

# Define features and target
X = df[['Runs', 'Strike Rate', 'Run Rate', 'Boundary Percentage', 'Dot Ball Percentage', 'Run
Rate vs Strike Rate Ratio', 'Boundary to Dot Ball Ratio', 'Runs per Ball Faced']]
y = df['Century']

# Split data 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)

# Train a random forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

Output



```
Accuracy: 0.6
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.44      0.50         9
           1       0.62      0.73      0.67        11

    accuracy                           0.60        20
   macro avg       0.59      0.59      0.58        20
weighted avg       0.60      0.60      0.59        20


[Program finished]
```

Outcomes:

1. **Predictive Insights**: The model provides predictions on century scoring potential.
2. **Performance Evaluation**: The model's performance was evaluated using metrics like accuracy score and classification report.
3. **Model Improvement**: Identified areas for improvement, such as tuning hyperparameters and experimenting with different algorithms.