Project#1 Report

110550097 王若禎

Dataset and Code Link :

https://github.com/Pandamachi/AIC/blob/main/Project1

I. Introduction

This dataset is about football player ability scores and try to use these scores to predict their position. In this project, I use five different supervised learning methods, such as KNN, Random Forest, SVM, Linear Discriminant Machine and CNN and one unsupervised learning method: K-Means to make predictions.

II. Dataset

《EA SPORTS FC 25》 is a kind of game that featuring the best players from around the globe in football, with all the match data captured to chart how they move, compete and win in every game.

I shared this dataset with my classmate 姚吝妙, we retrieve the top 2000 players' data, every player have eight columns, such as : Position, Overall, Pace, Shooting, Passing, Dribbling, Defending and Physicality. There are some explanations for these columns.

i. Position

There are twelve types in this column, the names and quantities of each item are shown in the following table :

CB	ST	CM	GK	CDM	RB	CAM	LB	LM	RM	RW	LW
336	278	253	206	179	139	128	119	98	96	72	66

ii. Overall

In this category it is the comprehensive score after Pace, Shooting, Passing, Dribbling, Defending and Physicality these six data. Can be used as a criterion for judging the overall ability of players.

iii. Pace

In this category, it is a combination of two different states in FIFA: sprint speed and acceleration. It is basically a value that reflects how fast a player is on the pitch.

iv. Shooting

Hitting the ball to score a goal. It is usually done using the feet or head, this category is ranking the player’s ability of scoring the goal.

v. Passing

It’s a swift one-two to get around an opponent or a mesmerizing diagonal ball to switch play to the opposite flank, allowing players to connect and combine with each other successfully.

vi. Dribbling

Staying on the ball while moving it around the pitch. Players can use different techniques to achieve this, like hiding and revealing.

III. Method

i. Data preprocessing

In this project, I use two kinds of methods to deal with data imbalance problem. The first one is Synthesized Minority Oversampling Technique(SMOTE), it randomly selects a small category sample and K samples of the same category near the sample, randomly select one of the K samples and generate a new sample of the same category between it and the selected sample. I use this method in supervised learning method.

The second method to deal with data imbalance is Principal Component Analysis(PCA), it works by transforming high-dimensional data into a lower-dimensional space while maximizing the variance of the data in the new space. I use this method in unsupervised learning methods and reduce data into two dimensions.

ii. Supervised Learning Algorithm

I use five supervised learning algorithms in this project.

k-nearest neighbors (KNN)

KNN is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point, it works by finding the K closest data points(neighbors) to a given query point and making predictions based on their values. For classification, it assigns the most common class among the neighbors and for regression, it takes the average (or weighted average) of the neighbors' values.

Random Forest

Random Forest is a powerful ensemble machine learning algorithm used for both classification and regression tasks. It is based on the concept of Decision Trees, but instead of using just one tree, it creates multiple trees and combines their outputs for more accurate and stable predictions.

Support Vector Machine (SVM)

SVM works by finding the optimal hyperplane that best separates data points belonging to different classes, it finds the best dividing line (or plane in higher dimensions) that maximizes the margin (distance) between the closest data points from each class and these closest points are called support vectors.

Linear Discriminant Analysis (LDA)

LDA is used to reduce the dimensionality of a dataset while maintaining the most important information for classification. LDA will compute the mean for each class first, and then calculate the scatter matrices, which measures the spread of data points within each class and between different class centers. After that, it will find the linear combination of features that maximize the separation between classes and project the data onto the new lower-dimensional space of classification.

Convolutional Neural Network (CNN)

CNN will reshape the data into a 2D matrix that CNNs can process and detect patterns in the ability metrics across multiple players. After that, filters/kernels will help identify which features contribute most to different player positions. The extracted features are flattened into a 1D vector and passed through dense (fully connected) layers for classification. In the end, it will use the final layer to predict probabilities of different positions.

In the unsupervised learning algorithm part, I choose K-Means as the method to predict the player's position.

K-Means

K-Means is used for clustering data into K groups. It finds patterns in data and groups similar data points together without requiring labeled outputs. First, it will choose the Number of Clusters (K) to decide how many clusters (groups) you want to divide the data into. Second, it will Pick K random points from the dataset as the initial cluster centers, after that, it will Compute the Euclidean distance between each point and the centroids and assign the point to the closest cluster and compute the new center of each cluster by taking the mean of all points in that cluster.

IV. Experiment

According to CB, ST, CM, GK, CDM, RB, CAM, LB, LM, RM, RW and LW these twelve types, I further classify them into four different types and re-predict based on the new non-class.

Forward	ST, RW, LW
Midfielder	CM, CDM, CAM, LM, RM
Defender	CB, RB, LB
Goalkeeper	GK

In this project, we use the top 2000 players in FC25 ranking and use six different data combinations to find the prediction result.

Experiment Data

1. 2000 players and 12 categories

2. 1000 players and 12 categories

3. 2000 players and 4 categories

4. 1000 players and 4 categories

5. 2000 players and 4 categories and SMOTE

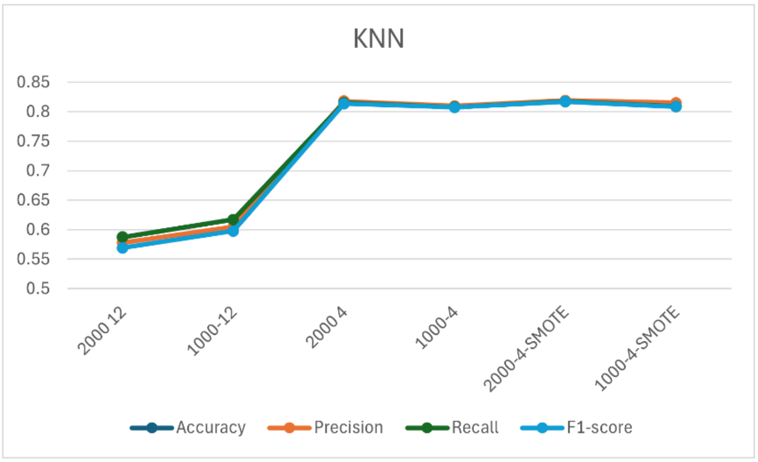
6. 1000 players and 4 categories and SMOTE

v. Analysis

I use 5-fold cross-validation to select the best result and conduct comparison of performance across various algorithms for different datasets.

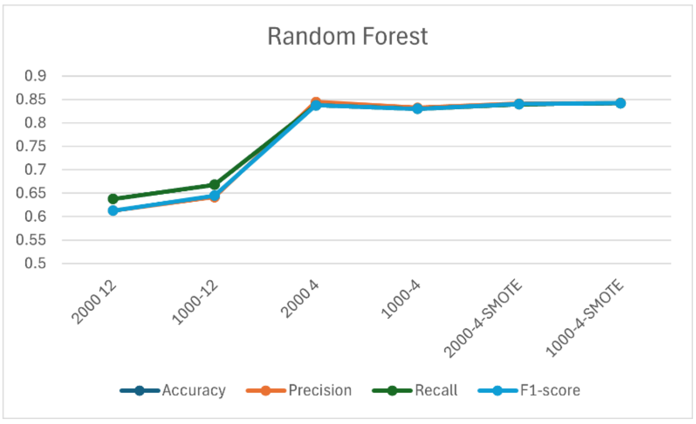
K-Nearest Neighbors (KNN)

In this method, we observe that in the 12-type position scenario, 1000 records yield better results than 2000 records, whereas in the 4-type position scenario, 2000 records outperform 1000 records. Additionally, I applied SMOTE for oversampling, but it did not lead to significant improvement in the results.



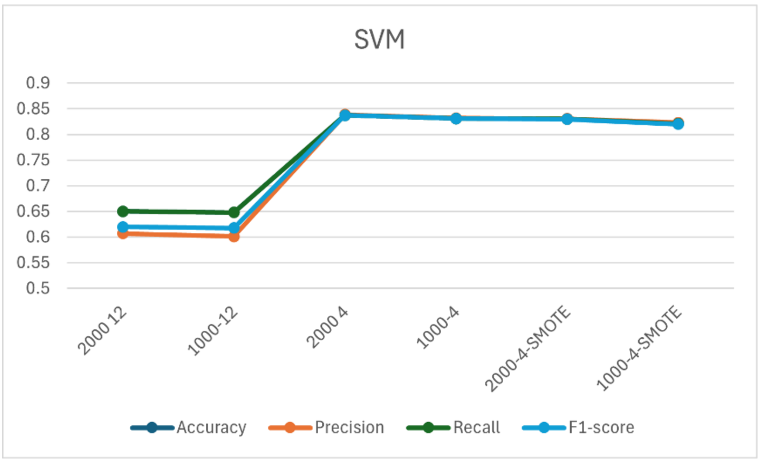
Random Forest

In this method, we observe a pattern similar to KNN: in the 12-type position scenario, 1000 records yield better results than 2000 records, while in the 4-type position scenario, 2000 records perform better than 1000 records. Additionally, applying for SMOTE does not lead to significant improvement. Compared to KNN, this method achieves better results in the 12-type position prediction.



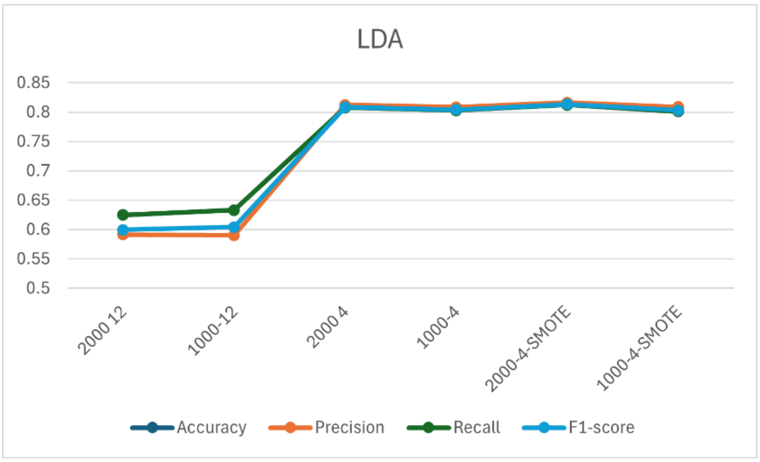
Support Vector Machine (SVM)

In this section, we observe that whether in the 4-type position or the 12-type position, 2000 records yield better results. However, using SMOTE does not improve the prediction results. The performance is better than KNN and nearly identical to that of Random Forest.



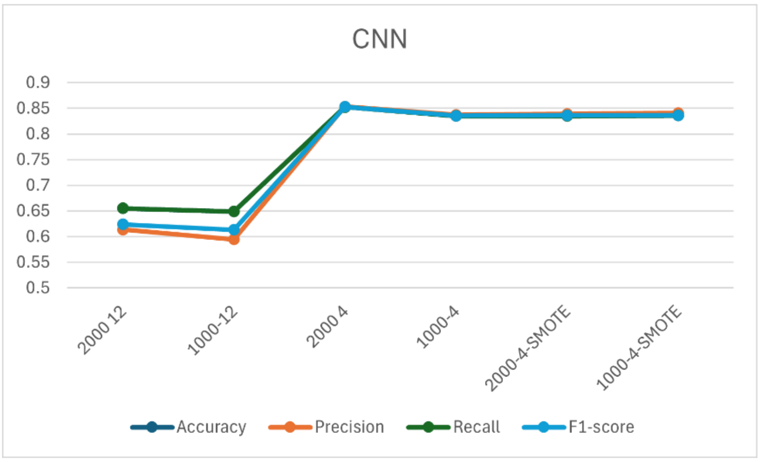
Linear Discriminant Analysis (LDA)

In this method, the situation differs from other methods; specifically, SMOTE has a positive impact on the prediction results. Moreover, the difference between 2000 records and 1000 records is not as significant as in other methods.



Convolutional Neural Network (CNN)

In this method, the results improve significantly compared to other methods in both the 4-type and 12-type position predictions. However, it performs the worst in 12-type prediction while achieving the best results in 4-type prediction. I also applied SMOTE to address the data imbalance issue, but the improvement remains substantial.



K-Means

In this project, I used K-Means for position prediction and compared its performance with and without PCA. Without PCA, the model performed poorly in the 12-type classification but did better in the 4-type scenario. After applying PCA, inertia and silhouette scores improved, indicating tighter clusters.

However, accuracy dropped in the 12-type classification due to information loss, while the 4-type classification benefited from better-structured data.

Although PCA improved clustering quality, it did not necessarily enhance predictive accuracy for more detailed classifications, highlighting the challenge of using unsupervised learning for this task compared to supervised methods.

Without PCA

K-Means	Accuracy	Precision	Recall	F1-score	Inertia	Silhouette Score
2000-12	0.1125	0.0862	0.1125	0.0929	3177.3747	0.1723
1000-12	0.1230	0.0997	0.1230	0.1046	3177.3747	0.1908
2000-4	0.3200	0.5047	0.3200	0.3833	5330.7426	0.2180
1000-4	0.3030	0.4794	0.3030	0.3631	5330.7426	0.2250

With PCA

K-Means + PCA	Accuracy	Precision	Recall	F1-score	Inertia	Silhouette Score
2000-12	0.0725	0.0763	0.0725	0.0708	715.0845	0.3095
1000-12	0.0640	0.0708	0.0640	0.0644	715.0845	0.3288
2000-4	0.3550	0.3910	0.3550	0.3710	1991.5750	0.3651
1000-4	0.3350	0.3730	0.3350	0.3510	1991.5750	0.3792

VI. Discuss

i. Based on your experiments, are the results and observed behaviors what you expect?

1. Since this project focuses on position prediction, I initially expected Random Forest to perform well. As anticipated, both Random Forest and CNN achieved the best results in this study.

2. Before the experiment, I assumed that using SMOTE and PCA would significantly impact the results. However, I found that data imbalance was not a major issue, so SMOTE had little effect. On the other hand, PCA noticeably improved the results.

ii. Discuss factors that affect the performance, including dataset characteristics.

There are some factors that will affect the performance, such as:

1. Prediction categories (i.e. 4-types, 12types)

2. Numbers of data

3. Different data preprocess methods

4. Learning algorithms (i.e. supervised learning, unsupervised learning)

iii Describe experiments that you would do if there were more time available.

If more time were available, I would explore using different score combinations for prediction. For example, I would investigate whether excluding Pace affects the results. Additionally, I would experiment with more learning methods to determine which yields the best predictions. Another area of interest is incorporating players' physical attributes, such as height and weight, as these factors could also influence the prediction outcomes.

iv. Indicate what you have learned from the experiments as well as your remaining questions.

In this project, I found that determining which data to use is a crucial challenge, as it impacts every step that follows. It was difficult to find a topic that was both interesting and feasible within the given time.

Additionally, I learned about various supervised and unsupervised learning methods, gaining a clear understanding of their differences. This experience also helped me understand how to select the most suitable algorithm to achieve the best prediction results.

Appendix

KNN

KNN	Accuracy	Precision	Recall	F1-score
2000-12	0.5875	0.578	0.5875	0.5692
1000-12	0.617	0.6048	0.617	0.5977
2000-4	0.815	0.8177	0.815	0.8138
1000-4	0.808	0.8098	0.808	0.8073
2000-4-SMOTE	0.8175	0.8191	0.8175	0.8172
1000-4-SMOTE	0.809	0.815	0.809	0.8086

Random Forest

RF	Accuracy	Precision	Recall	F1-score
2000 12	0.6375	0.6125	0.6375	0.6126
1000-12	0.668	0.6415	0.668	0.6447
2000 4	0.8375	0.845	0.8375	0.8377
1000-4	0.83	0.8325	0.83	0.83
2000-4-SMOTE	0.84	0.8412	0.84	0.8402
1000-4-SMOTE	0.842	0.8426	0.842	0.8421

SVM

SVM	Accuracy	Precision	Recall	F1-score
2000-12	0.65	0.607	0.65	0.6199
1000-12	0.648	0.601	0.648	0.6177
2000-4	0.8375	0.8388	0.8375	0.8374
1000-4	0.831	0.8318	0.831	0.8311
2000-4-SMOTE	0.83	0.8307	0.83	0.8297
1000-4-SMOTE	0.821	0.823	0.821	0.8201

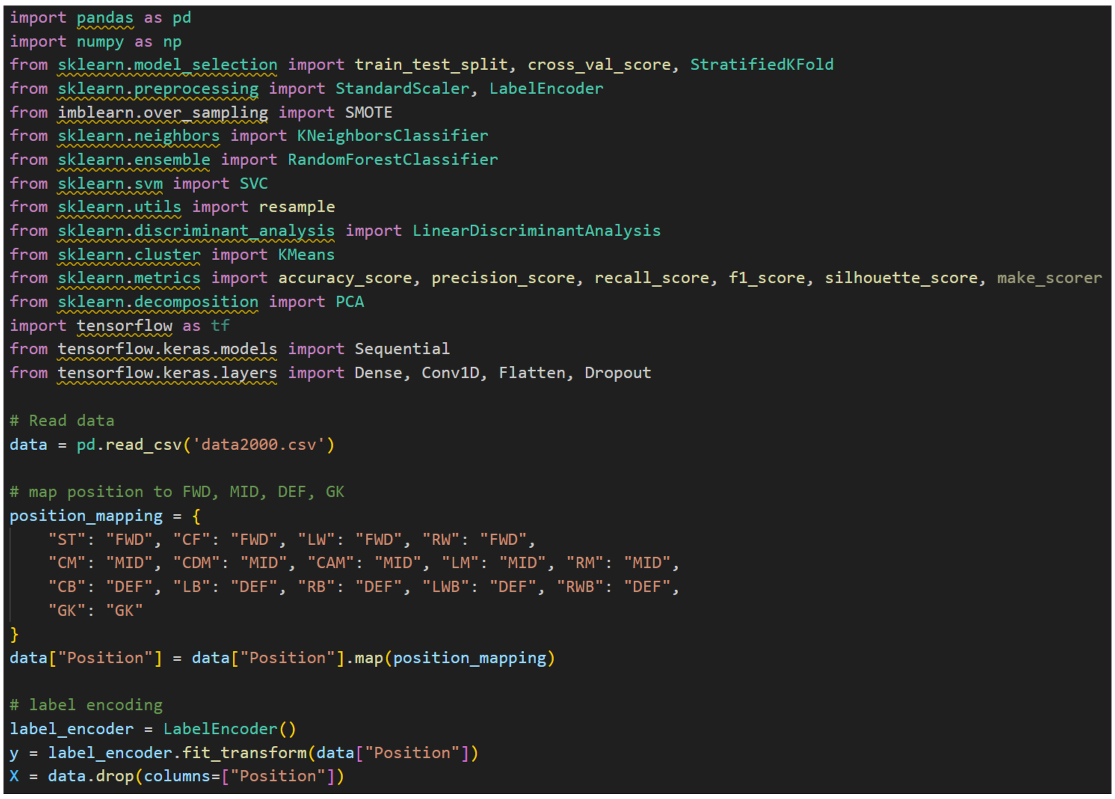
LDA

LDA	Accuracy	Precision	Recall	F1-score
2000 12	0.625	0.5913	0.625	0.5996
1000-12	0.633	0.5902	0.633	0.6041
2000 4	0.8075	0.8125	0.8075	0.8083
1000-4	0.803	0.8084	0.803	0.8043
2000-4-SMOTE	0.8125	0.8164	0.8125	0.8136
1000-4-SMOTE	0.801	0.8089	0.801	0.8029

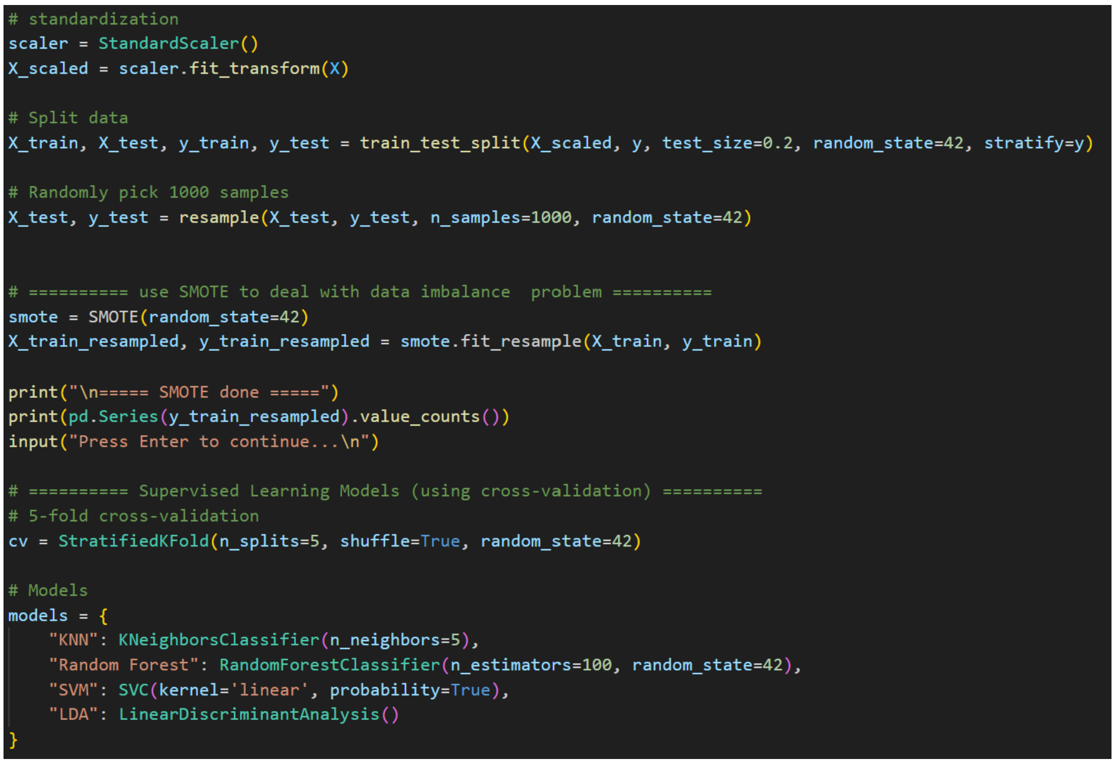
CNN

CNN	Accuracy	Precision	Recall	F1-score
2000 12	0.655	0.6138	0.655	0.6239
1000-12	0.649	0.5945	0.649	0.6131
2000 4	0.8525	0.8534	0.8525	0.8528
1000-4	0.835	0.8375	0.835	0.8357
2000-4-SMOTE	0.835	0.8392	0.835	0.8362
1000-4-SMOTE	0.836	0.841	0.836	0.8362

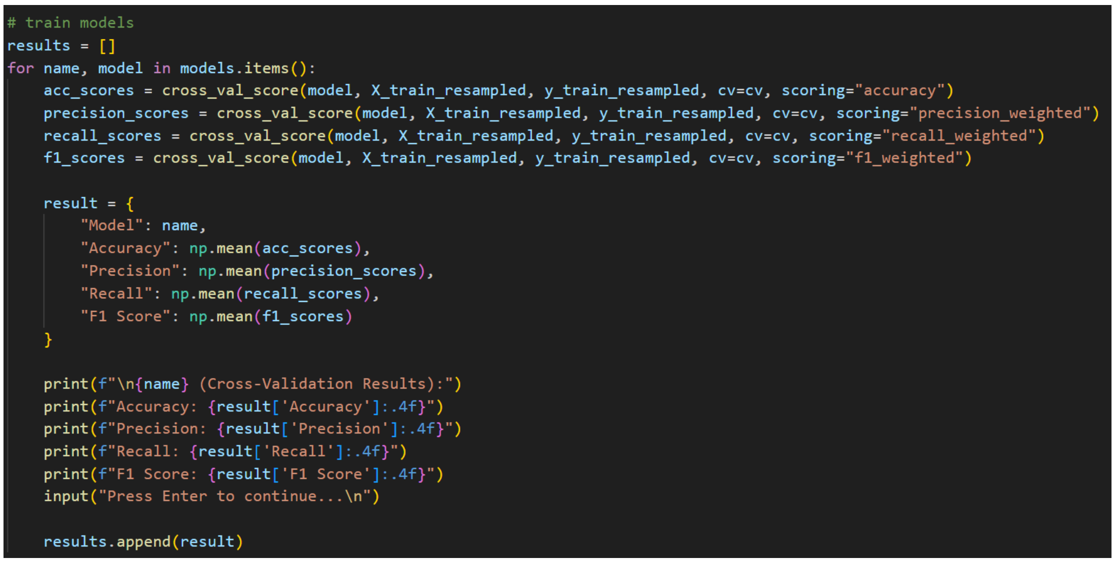
Code



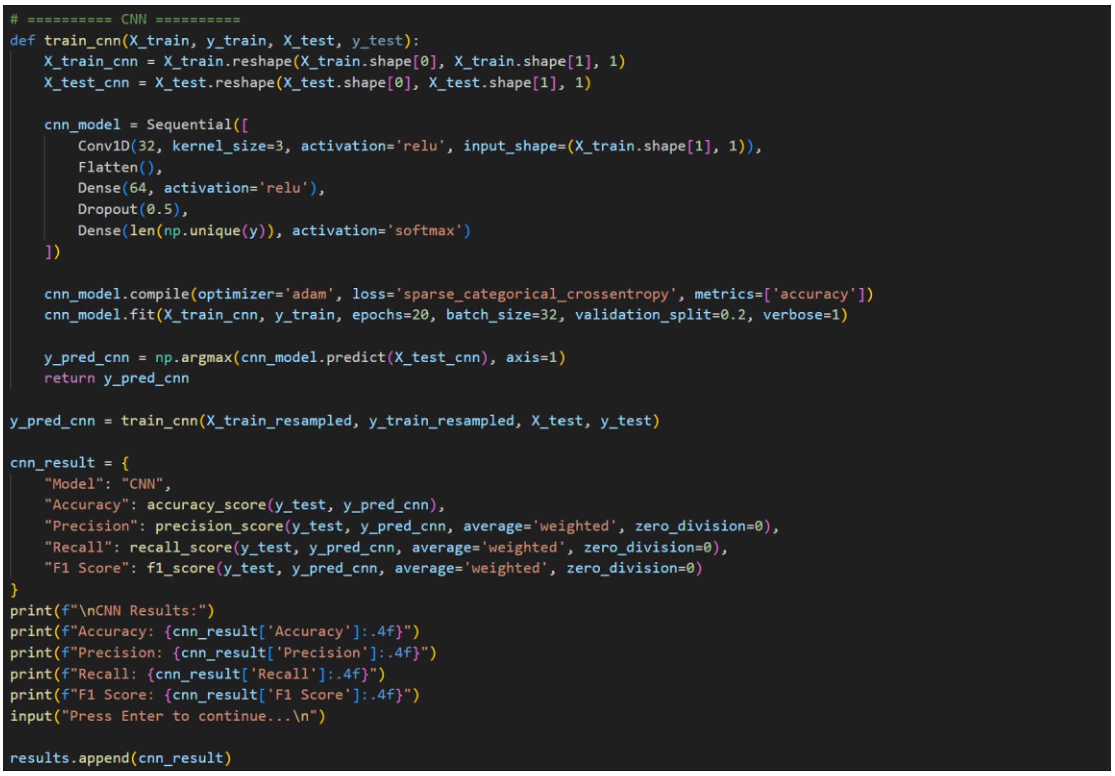
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
from sklearn.preprocessing import StandardScaler, LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.utils import resample
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, silhouette_score, make_scorer
from sklearn.decomposition import PCA
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv1D, Flatten, Dropout

# Read data
data = pd.read_csv('data2000.csv')

# map position to FWD, MID, DEF, GK
position_mapping = {
    "ST": "FWD", "CF": "FWD", "LW": "FWD", "RW": "FWD",
    "CM": "MID", "CDM": "MID", "CAM": "MID", "LM": "MID", "RM": "MID",
    "CB": "DEF", "LB": "DEF", "RB": "DEF", "LWB": "DEF", "RWB": "DEF",
    "GK": "GK"
}
data["Position"] = data["Position"].map(position_mapping)

# label encoding
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(data["Position"])
X = data.drop(columns=["Position"])
```



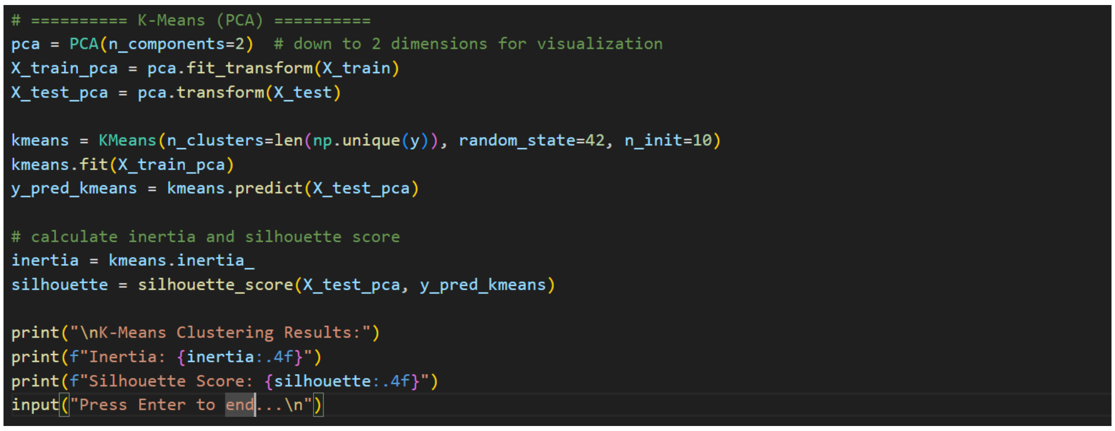
```python
# standardization
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

# Randomly pick 1000 samples
X_test, y_test = resample(X_test, y_test, n_samples=1000, random_state=42)


# ========== use SMOTE to deal with data imbalance  problem ==========
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

print("\n===== SMOTE done =====")
print(pd.Series(y_train_resampled).value_counts())
input("Press Enter to continue...\n")

# ========== Supervised Learning Models (using cross-validation) ==========
# 5-fold cross-validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Models
models = {
    "KNN": KNeighborsClassifier(n_neighbors=5),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "SVM": SVC(kernel='linear', probability=True),
    "LDA": LinearDiscriminantAnalysis()
}
```



```python
# train models
results = []
for name, model in models.items():
    acc_scores = cross_val_score(model, X_train_resampled, y_train_resampled, cv=cv, scoring="accuracy")
    precision_scores = cross_val_score(model, X_train_resampled, y_train_resampled, cv=cv, scoring="precision_weighted")
    recall_scores = cross_val_score(model, X_train_resampled, y_train_resampled, cv=cv, scoring="recall_weighted")
    f1_scores = cross_val_score(model, X_train_resampled, y_train_resampled, cv=cv, scoring="f1_weighted")

    result = {
        "Model": name,
        "Accuracy": np.mean(acc_scores),
        "Precision": np.mean(precision_scores),
        "Recall": np.mean(recall_scores),
        "F1 Score": np.mean(f1_scores)
    }

    print(f"\n{name} (Cross-Validation Results):")
    print(f"Accuracy: {result['Accuracy']:.4f}")
    print(f"Precision: {result['Precision']:.4f}")
    print(f"Recall: {result['Recall']:.4f}")
    print(f"F1 Score: {result['F1 Score']:.4f}")
    input("Press Enter to continue...\n")

    results.append(result)
```



```python
# ========== CNN ==========
def train_cnn(X_train, y_train, X_test, y_test):
    X_train_cnn = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
    X_test_cnn = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

    cnn_model = Sequential([
        Conv1D(32, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)),
        Flatten(),
        Dense(64, activation='relu'),
        Dropout(0.5),
        Dense(len(np.unique(y)), activation='softmax')
    ])

    cnn_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    cnn_model.fit(X_train_cnn, y_train, epochs=20, batch_size=32, validation_split=0.2, verbose=1)

    y_pred_cnn = np.argmax(cnn_model.predict(X_test_cnn), axis=1)
    return y_pred_cnn

y_pred_cnn = train_cnn(X_train_resampled, y_train_resampled, X_test, y_test)

cnn_result = {
    "Model": "CNN",
    "Accuracy": accuracy_score(y_test, y_pred_cnn),
    "Precision": precision_score(y_test, y_pred_cnn, average='weighted', zero_division=0),
    "Recall": recall_score(y_test, y_pred_cnn, average='weighted', zero_division=0),
    "F1 Score": f1_score(y_test, y_pred_cnn, average='weighted', zero_division=0)
}
print(f"\nCNN Results:")
print(f"Accuracy: {cnn_result['Accuracy']:.4f}")
print(f"Precision: {cnn_result['Precision']:.4f}")
print(f"Recall: {cnn_result['Recall']:.4f}")
print(f"F1 Score: {cnn_result['F1 Score']:.4f}")
input("Press Enter to continue...\n")

results.append(cnn_result)
```



```python
# ========== K-Means (PCA) ==========
pca = PCA(n_components=2)  # down to 2 dimensions for visualization
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

kmeans = KMeans(n_clusters=len(np.unique(y)), random_state=42, n_init=10)
kmeans.fit(X_train_pca)
y_pred_kmeans = kmeans.predict(X_test_pca)

# calculate inertia and silhouette score
inertia = kmeans.inertia_
silhouette = silhouette_score(X_test_pca, y_pred_kmeans)

print("\nK-Means Clustering Results:")
print(f"Inertia: {inertia:.4f}")
print(f"Silhouette Score: {silhouette:.4f}")
input("Press Enter to end...\n")
```

Reference

1. A beginner's guide to supervised learning with Python

https://www.geeksforgeeks.org/a-beginners-guide-to-supervised-learning-with-python/

2. FC 25 Player Ratings Reveal

https://www.ea.com/games/ea-sports-fc/ratings