**Core code**

XGB+SHAP visualization Part

import shap

import matplotlib.pyplot as plt

from xgboost import XGBClassifier

# XGB Part

final\_model = XGBClassifier(use\_label\_encoder=False,

eval\_metric="logloss", subsample=0.8,

colsample\_bytree=0.8, reg\_lambda=1,

min\_child\_weight=3, max\_depth=7)

final\_model.fit(csgo[x], csgo[y])

# creat SHAP explainer（tree-based explainer）

explainer = shap.Explainer(final\_model, csgo[x])

shap\_values = explainer(csgo[x]) # shap.Explanation

# summary plot

shap.summary\_plot(shap\_values, csgo[x], feature\_names=x, max\_display=20)

# Bar plot

shap.summary\_plot(shap\_values, csgo[x], plot\_type="bar", feature\_names=x, max\_display=20)

#Force plot

shap.force\_plot(explainer.expected\_value, shap\_values.values[0], csgo[x].iloc[0], matplotlib=True)

# Waterfall

shap.plots.waterfall(shap\_values[0])

# Decision Plot

# shap.plots.decision(shap\_values[0])

shap\_values = explainer(csgo[x])

shap.decision\_plot(

explainer.expected\_value,

shap\_values.values,

feature\_names=x )

# Heatmap Plot

shap.plots.heatmap(shap\_values[:N]) # N=sample

#dependence\_plot

feature\_names\_to\_plot = ["Metrics Name"]

for feature\_name in feature\_names\_to\_plot:

print(f" dependence\_plot ：{feature\_name}")

shap.dependence\_plot(feature\_name, shap\_values.values, csgo[x], feature\_names=x)