Summary of data:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import json

import sys

print (sys.version)

lol= pd.read\_csv("D://CS989//ML//lolgame.csv")

Champion=pd.read\_csv("D://CS989//ML//games.csv")

print(Champion.shape)

print(lol.groupby('winner').firstBlood.describe())

print(Champion.agg(["count","max"]))

print(Champion.groupby('t1\_inhibitorKills').describe())

print(Champion.groupby('t1\_baronKills').describe())

corr = lol.corr()

sns.heatmap(corr)

plt.show()

sns.countplot(x='winner',data=lol);plt.xlabel('overall winning rate for two team')

plt.show()

sns.countplot(x='firstBlood',data=lol,);plt.xlabel('overall first blood rate for two team')

plt.show()

sns.countplot(x='firstTower',data=lol);plt.xlabel('overall first tower rate for two team')

plt.show()

sns.countplot(x='firstInhibitor',data=lol);plt.xlabel('overall first inhibitor rate for two team')

plt.show()

sns.countplot(x='firstBaron',data=lol);plt.xlabel('overall kill firstBaron rate for two team')

plt.show()

sns.barplot(x=Champion["t1\_towerKills"],y=Champion["winner"],data=lol);plt.xlabel('tower destroyed amount')

plt.show()

sns.barplot(x=Champion["t1\_inhibitorKills"],y=Champion["winner"],data=lol);plt.xlabel('inhibitor destroyed amount')

plt.show()

sns.barplot(x=Champion["t1\_baronKills"],y=Champion["winner"],data=lol);plt.xlabel('baron slain amount')

plt.show()

plt.xlim([0,6000])

num\_bins =60

plt.hist(Champion.gameDuration,num\_bins, facecolor='green', alpha=0.5)

plt.show()

Champion=pd.read\_csv("D://CS989//ML//games.csv")

def IDconvert(ID,dic):

role = dic['name'][ID]

return role

Championname = pd.read\_json('D:/CS989/ML/LOL/champion\_info\_2.json')

information = pd.read\_json((Championname['data']).to\_json(), orient='index')

information.set\_index(['id'],inplace=True)

information.head()

Picklist = ['t1\_champ1id','t1\_champ2id','t1\_champ3id','t1\_champ4id','t1\_champ5id', 't2\_champ1id','t2\_champ2id','t2\_champ3id','t2\_champ4id','t2\_champ5id']

Banlist = ['t1\_ban1','t1\_ban2','t1\_ban3','t1\_ban4','t1\_ban5', 't2\_ban1','t2\_ban2','t2\_ban3','t2\_ban4','t2\_ban5']

for c in Picklist:

Champion[c] = Champion[c].apply(lambda x:IDconvert (x, information))

for c in Banlist:

Champion[c] = Champion[c].apply(lambda x:IDconvert (x, information))

print(Champion[Picklist].head())

print(Champion[Banlist].head())

Pick = pd.concat([Champion['t1\_champ1id'],Champion['t1\_champ2id'],Champion['t1\_champ3id'],Champion['t1\_champ4id'],Champion['t1\_champ5id'],Champion['t2\_champ1id'],Champion['t2\_champ2id'],Champion['t2\_champ3id'],Champion['t2\_champ4id'],Champion['t2\_champ5id']],ignore\_index=False)

Totalpick = sorted(Pick,reverse=True,key=id)

Ban = pd.concat([Champion['t1\_ban1'],Champion['t1\_ban2'],Champion['t1\_ban3'],Champion['t1\_ban4'],Champion['t1\_ban5'],Champion['t2\_ban1'],Champion['t2\_ban2'],Champion['t2\_ban3'],Champion['t2\_ban4'],Champion['t2\_ban5']],ignore\_index=False)

Totalban= sorted(Ban,reverse=True,key=id)

print("Total picklist",Pick.head(10))

print("Total banlist",Ban.head(10))

fig, (fig1, fig2) = plt.subplots(1,2, sharey=False, figsize=(23,30))

sns.countplot(y=Totalpick, data=Champion, ax=fig1,color='b',palette="viridis",orient='h')

sns.countplot(y=Totalban, data=Champion, ax=fig2,color='r',palette="viridis",orient='h')

plt.ylabel('Left graph for Picklist,right graph for Banlist')

K-Means:

from sklearn import datasets

from sklearn import metrics

from sklearn import cluster

from sklearn.preprocessing import scale

Champion=pd.read\_csv("D://CS989//ML//games.csv")

x = Champion[['firstBlood','firstTower','firstBaron','firstInhibitor','firstDragon', 'firstRiftHerald']]

data=scale(x)

n=len(np.unique(Champion.winner))

kmeans = cluster.KMeans(n\_clusters=n)

kmeans.fit(data)

print("completeness\_score:",metrics.completeness\_score(lol.winner, kmeans.labels\_))

print("homogeneits\_score:",metrics.homogeneity\_score(lol.winner, kmeans.labels\_) )

kmeans = cluster.KMeans(n\_clusters=5)

kmeans.fit(data)

print("completeness\_score:",metrics.completeness\_score(lol.winner, kmeans.labels\_))

print("homogeneits\_score:",metrics.homogeneity\_score(lol.winner, kmeans.labels\_) )

kmeans = cluster.KMeans(n\_clusters=10)

kmeans.fit(data)

print("completeness\_score:",metrics.completeness\_score(lol.winner, kmeans.labels\_))

print("homogeneits\_score:",metrics.homogeneity\_score(lol.winner, kmeans.labels\_) )

Logistic Regression:

#Logistic regression

from sklearn.linear\_model import LogisticRegression

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

from sklearn.metrics import classification\_report

x = Champion[['firstBlood','firstTower','firstBaron','firstInhibitor','firstDragon', 'firstRiftHerald']]

y = Champion['winner']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.8, random\_state=123)

logmodel = LogisticRegression()

logmodel.fit(x\_train, y\_train)

predictions = logmodel.predict(x\_test)

print(classification\_report(y\_test, predictions))