PUBG_v3

April 28, 2021

1 Import

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from scipy import stats

import matplotlib.pyplot as plt

from sklearn import metrics
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import pairwise_distances_argmin_min

from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDRegressor
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import Ridge
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.svm import LinearSVR
from sklearn.cluster import KMeans

from xgboost import XGBRegressor

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GroupShuffleSplit
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from datetime import datetime

import random
import warnings
warnings.filterwarnings('ignore')
```

2 Load data

[2]:
```
train_data = pd.read_csv('../Datasets/train_V2.csv')
test_data = pd.read_csv('../Datasets/test_V2.csv')
```

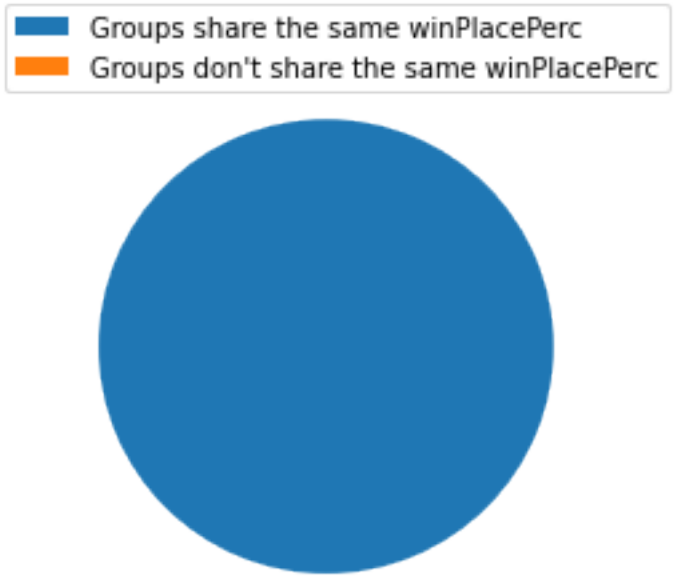
[25]:
```
train_data.shape[0]+test_data.shape[0]
```

[25]: 6381140

3 Check if each team share the same WPP

[15]:
```
unique = train_data.groupby('groupId').winPlacePerc.nunique() == 1
all_unique = sum(unique)/(train_data['groupId'].unique()).shape[0]
```

[31]:
```
patches, texts = plt.pie([all_unique,1-all_unique],radius=0.8 )
plt.legend(patches, ['Groups share the same winPlacePerc','Groups don\'t share␣
 ↪the same winPlacePerc'], loc="upper right")
plt.tight_layout()
plt.savefig('compareWPP')
plt.show()
```



Before splitting the database, I thought that if we wanted to use the "Kaggle trick," it might not be possible to completely randomly split the data set unless the initial

training set and the testing set had overlapping groupids, which indicated that they were completely randomly split.

[3]:
```
#Get the groupId in training set
train_groupId = train_data['groupId'].unique()

#Get the groupId in testing set
test_groupId = test_data['groupId'].unique()

#Check for overlap
intersection = np.intersect1d(train_groupId,test_groupId,assume_unique=True)

print(intersection)
```

```
[]
```

The results show that the groupids of the two datasets do not overlap. Therefore, we cannot divide the training set and the test set completely at random. We need to split them by groupId.

4 Data cleansing

[4]:
```
is_NaN = train_data.isnull()
row_has_NaN = is_NaN.any(axis=1)
rows_with_NaN = train_data[row_has_NaN]
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    display(rows_with_NaN)
```

```
                     Id         groupId         matchId  assists  boosts  \
2744604  f70c74418bb064  12dfbede33f92b  224a123c53e008        0       0

         damageDealt  DBNOs  headshotKills  heals  killPlace  killPoints  \
2744604          0.0      0              0      0          1           0

         kills  killStreaks  longestKill  matchDuration matchType  maxPlace  \
2744604      0            0          0.0              9  solo-fpp         1

         numGroups  rankPoints  revives  rideDistance  roadKills  \
2744604          1        1574        0           0.0          0

         swimDistance  teamKills  vehicleDestroys  walkDistance  \
2744604           0.0          0                0           0.0

         weaponsAcquired  winPoints  winPlacePerc
2744604                0          0           NaN
```

[4]:
```
#Drop the missing value
train_data = train_data.drop(2744604)
```

```
#Reset the index
train_data = train_data.reset_index(drop=True)

#Delete dankPoints
train_data = train_data.drop(["rankPoints"],axis=1)
```

5 Split the dataset based on groupId

[3]:
```
train_inds, test_inds = next(GroupShuffleSplit(test_size=.20, n_splits=2,␣
 ↪random_state = 7).split(train_data, groups=train_data['groupId']))

train = train_data.iloc[train_inds]
test = train_data.iloc[test_inds]

train = train.reset_index(drop=True)
test = test.reset_index(drop=True)
```

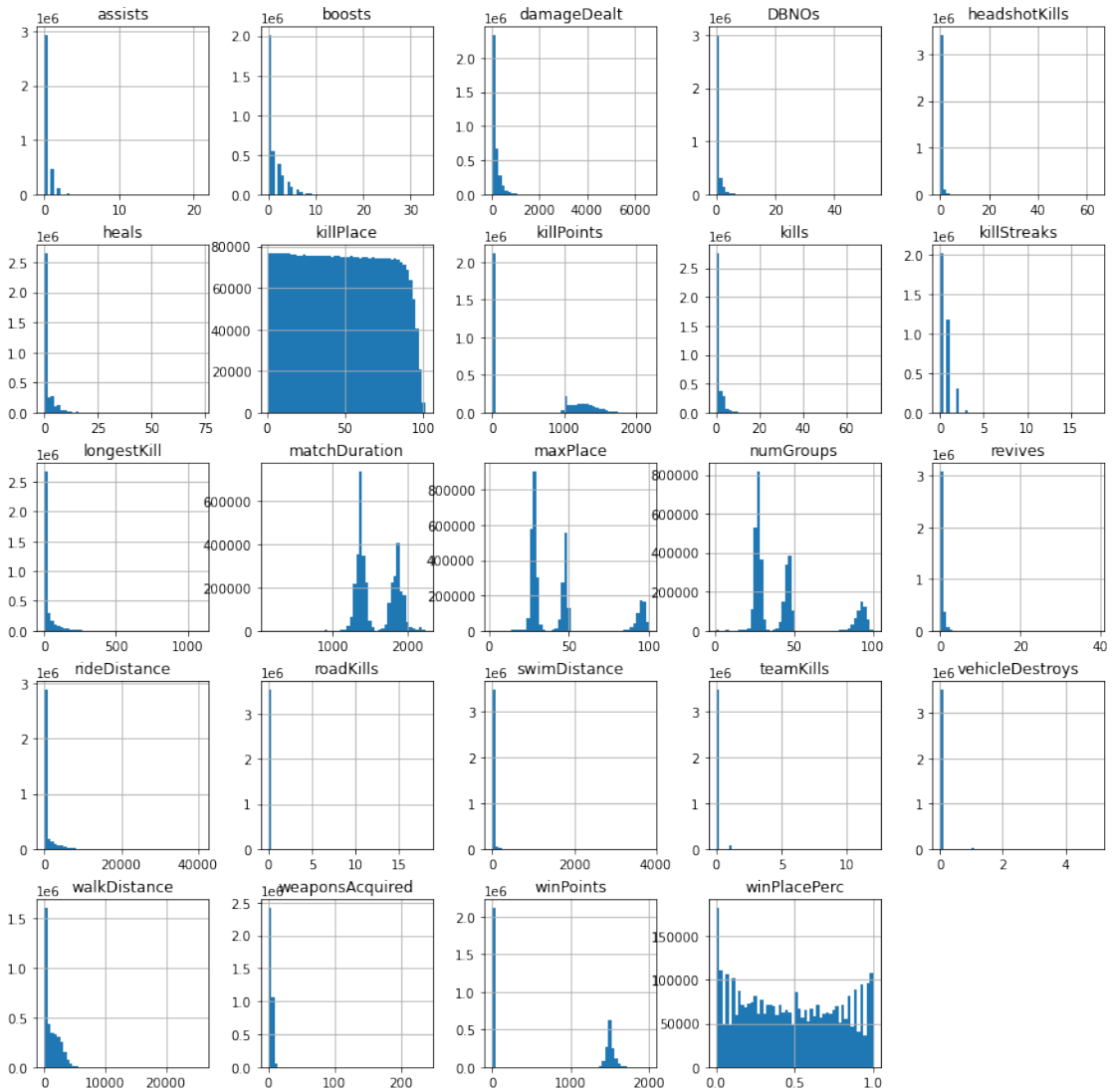
[30]:
```
patches, texts = plt.pie([train.shape[0],test.shape[0]],radius=0.8 )
plt.legend(patches, ['Number of players in the new training set','Number of␣
 ↪players in the new test set'], loc="upper right")
plt.tight_layout()
plt.savefig('train_test')
plt.show()
```

[5]:
```
#Check intersection
train_groupId = train['groupId'].unique()
test_groupId = test['groupId'].unique()
intersection= np.intersect1d(train_groupId,test_groupId,assume_unique=True)
print(intersection)
```

```
[]
```

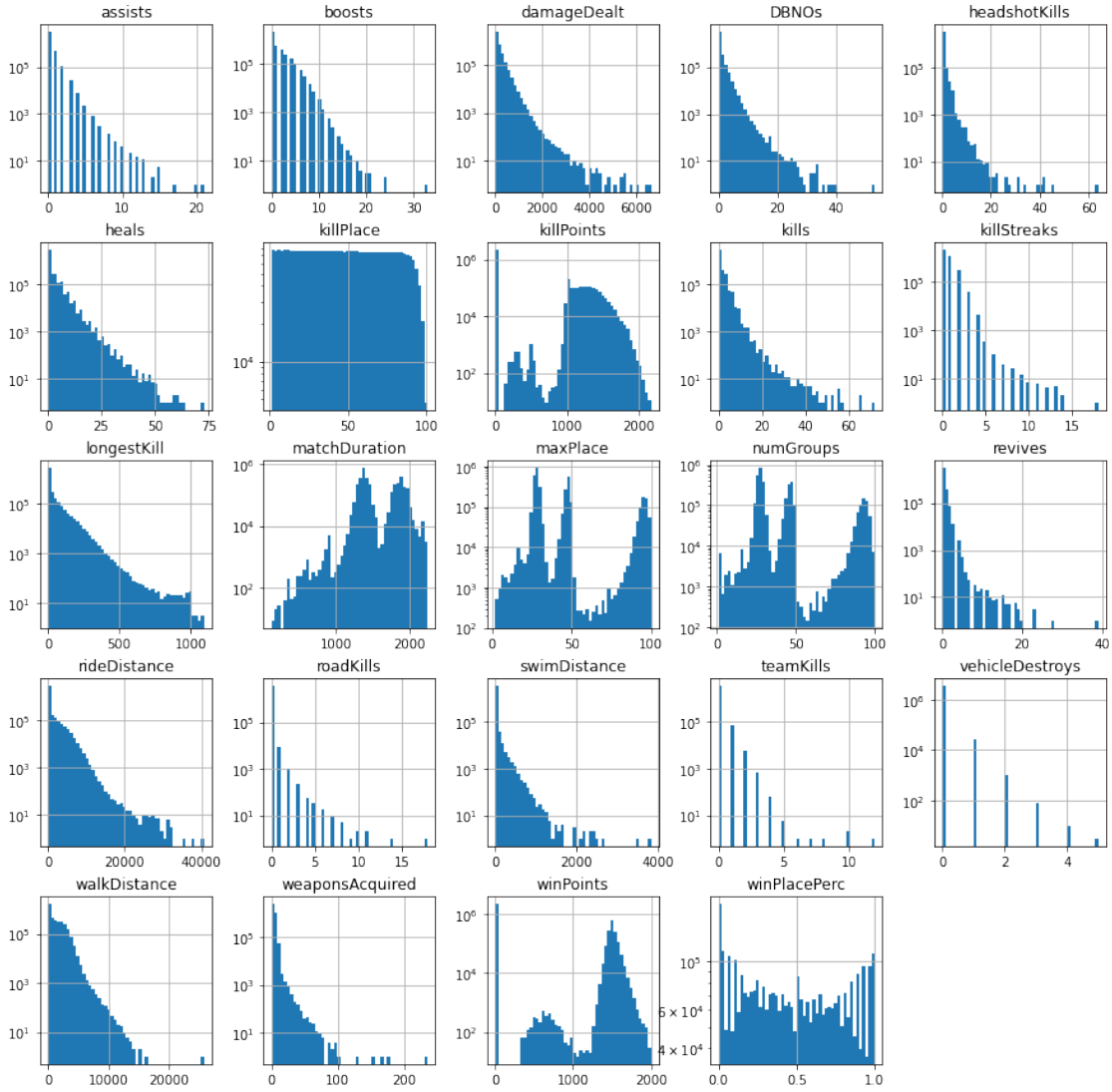
Histogram with normal scale

[9]:
```
train.hist(bins=50, layout=(5,5),figsize=(15, 15))
plt.savefig("hist_normal.png")
plt.show()
```

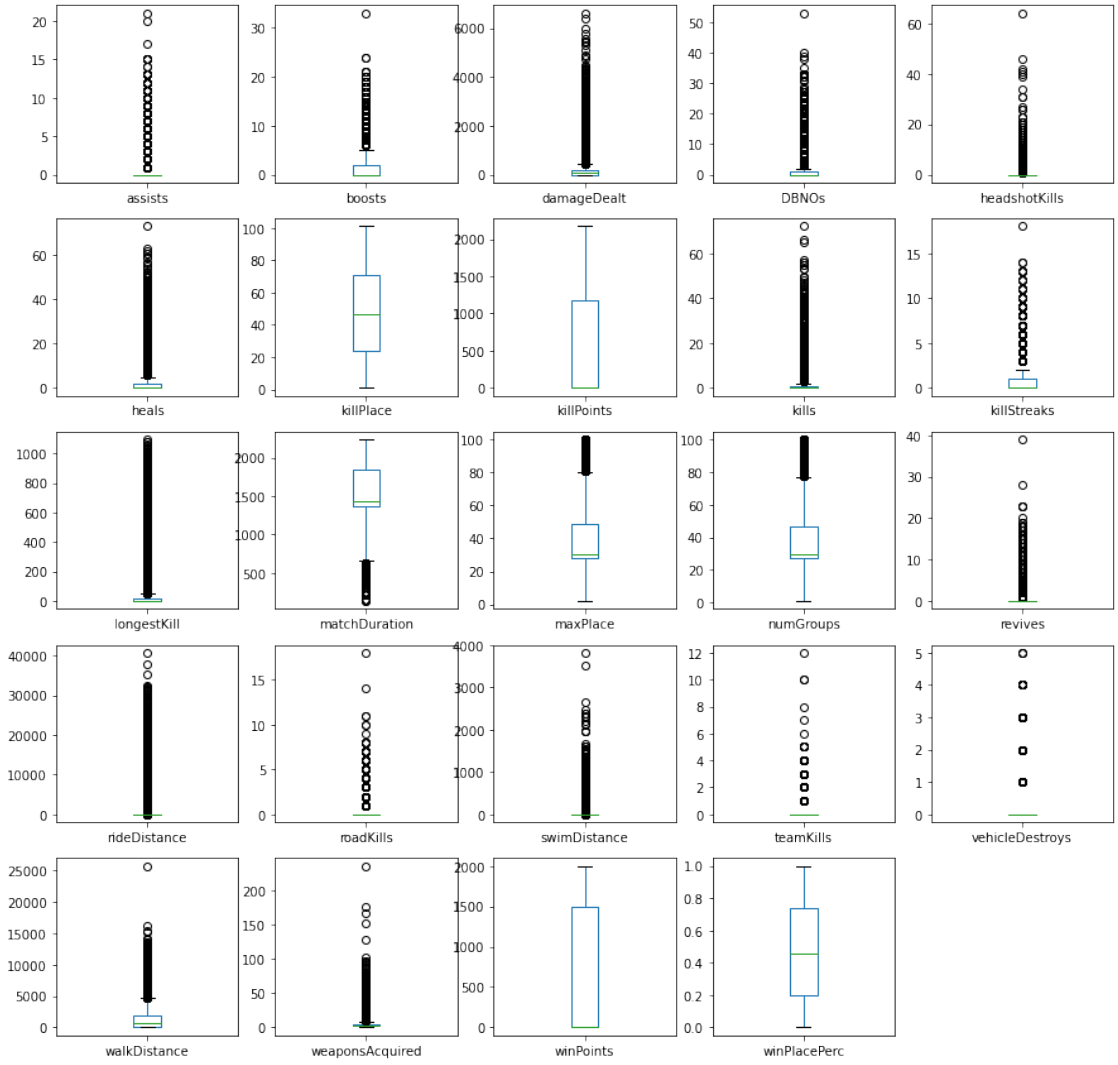


Histogram with log scale

[18]:
```
train.hist(bins=50, layout=(5,5),figsize=(15, 15),log=True)
plt.savefig("hist_log.png")
plt.show()
```



[19]:
```
train.plot(kind='box', subplots=True, layout=(5,5), sharex=False, sharey=False,␣
↪figsize=(15, 15))
plt.savefig("box_plot.png")
plt.show()
```



6 Data scaling

Scaled feature:

1. damageDealt * 1/100
2. killPlace * 1/100
3. killPoints * 1/1000
4. longestKill * 1/100
5. matchDuration * 1/1000
6. maxPlace * 1/100
7. numGroups * 1/100
8. rideDistance * 1/1000
9. swimDistance * 1/100
10. walkDistance * 1/1000
11. weaponsAcquired * 1/10

12. winPoints * 1/1000

[4]:
```
def scale(dataset):

    #Maintain the original data
    dataset = dataset.copy()



    dataset['damageDealt'] = dataset['damageDealt'] * 1/100
    dataset['killPlace'] = dataset['killPlace'] * 1/100
    dataset['killPoints'] = dataset['killPoints'] * 1/1000
    dataset['longestKill'] = dataset['longestKill'] * 1/100
    dataset['matchDuration'] = dataset['matchDuration'] * 1/1000
    dataset['maxPlace'] = dataset['maxPlace'] * 1/100
    dataset['numGroups'] = dataset['numGroups'] * 1/100
    dataset['rideDistance'] = dataset['rideDistance'] * 1/1000
    dataset['swimDistance'] = dataset['swimDistance'] * 1/100
    dataset['walkDistance'] = dataset['walkDistance'] * 1/1000
    dataset['weaponsAcquired'] = dataset['weaponsAcquired'] * 1/10
    dataset['winPoints'] = dataset['winPoints'] * 1/1000



    return dataset
```

[5]:
```
scaled_train = scale(train)
scaled_test = scale(test)
```

7 Cluster Analysis without matchType

[130]:
```
def getAverageWPP(y,n_clusters):

    #Initialize a list for storing the means with n_clusters 0s
    means = [0]*n_clusters

    #Get the mean in each cluster
    for i in range(n_clusters):
        means[i] = y[y["cluster"] == i]["winPlacePerc"].mean()

    return means
```

[21]:
```
#Do the clustering without Ids and matchType
cluster_train = scaled_train.drop(['Id','groupId','matchId','matchType'],axis=1)

#Split X and y
cluster_train_y = pd.DataFrame({'winPlacePerc':cluster_train['winPlacePerc']})
```

```
cluster_train_X = cluster_train.drop(['winPlacePerc'],axis=1)
```

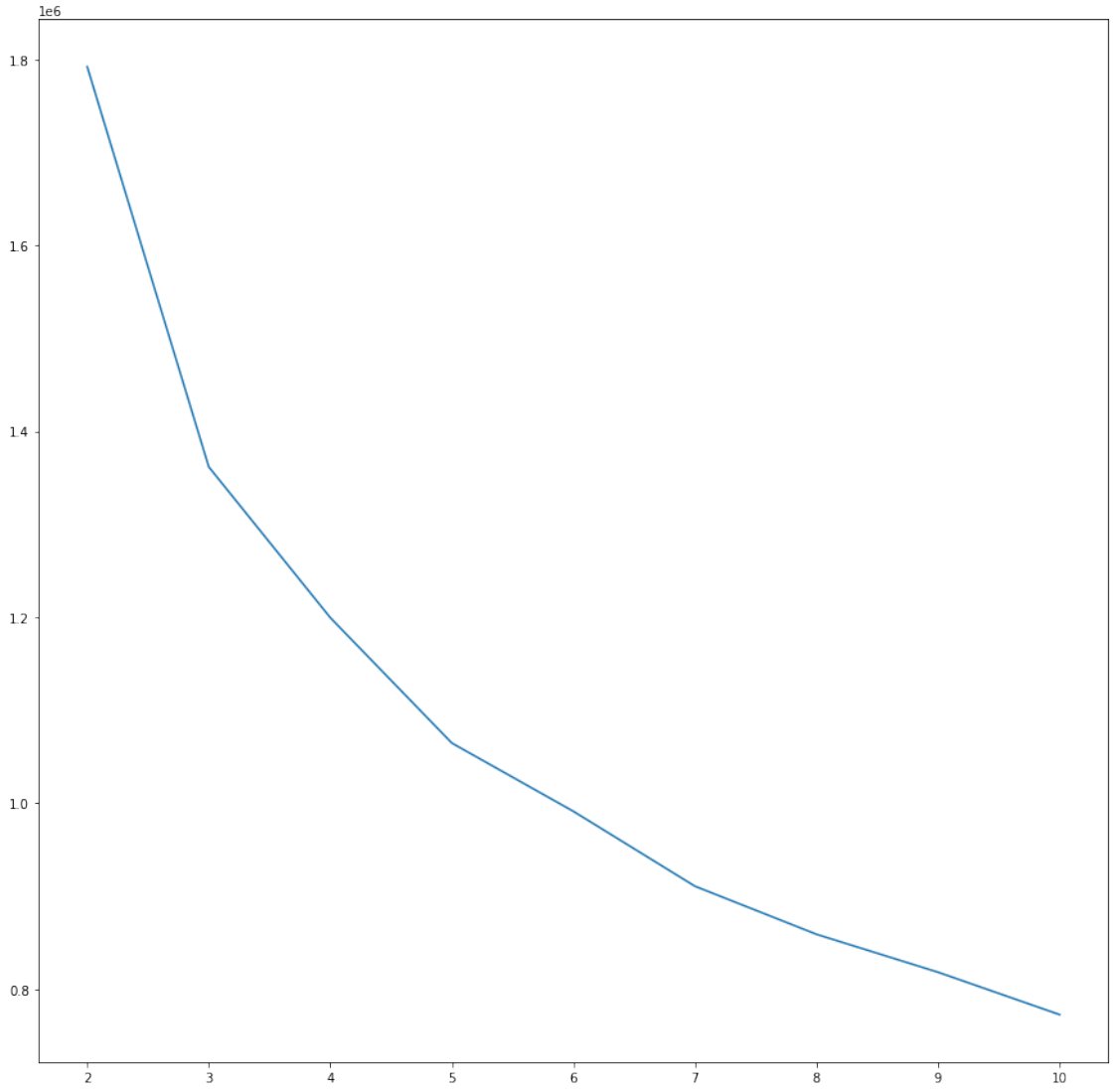
[22]:
```
#Two different initializations for comparing
kmeans_1 = KMeans(random_state=10,n_init = 10)
kmeans_2 = KMeans(random_state=20,n_init = 10)
```

[24]:
```
#Used to do score VS n_clustet plotting
scores_1 = []
Xs_1 = []

#Used to store best number of clusters, best score and best average WPP
best_n_clusters = 0
best_score = 0
best_average_Wpp = 0

for i in range(2,11):

    #Assign number of clusters
    kmeans_1.n_clusters = i

    #Train the model
    kmeans_1.fit(cluster_train_X)

    #Get the score
    score = metrics.calinski_harabasz_score(cluster_train_X,kmeans_1.labels_)

    #Add a cluster column to store labels
    cluster_train_X["cluster"] = kmeans_1.labels_

    #Add a cluster column to store labels
    cluster_train_y["cluster"] = kmeans_1.labels_

    #Get the average WPP
    average_Wpp = getAverageWPP(cluster_train_y,i)

    #If the score for current n_cluster is higher, then replace the best values
    if(score>best_score):
        best_score = score
        best_n_clusters = i
        best_average_Wpp = average_Wpp

    #Print the values
    print("n_clusters: ",i,",Score: ",score, ",Average WPP: ",average_Wpp)

    #Append the n_cluster
    Xs_1.append(i)
```

```
    #Append the score
    scores_1.append(score)

    #Delete the cluster column
    cluster_train_X = cluster_train_X.drop(["cluster"],axis=1)
    cluster_train_y = cluster_train_y.drop(["cluster"],axis=1)

#Print the best values
print("best_n_clusters: ",best_n_clusters,", best_score: ", best_score,",␣
 ↪best_average_Wpp: ",best_average_Wpp)
```

```
n_clusters:  2 ,Score:  1792350.6600233421 ,Average WPP:  [0.3885809058510306,
0.7902486079468457]
n_clusters:  3 ,Score:  1361777.1768941977 ,Average WPP:  [0.3781745289941825,
0.7831715901878191, 0.7514025662088565]
n_clusters:  4 ,Score:  1199510.042944904 ,Average WPP:  [0.83818786181057,
0.7063607561904349, 0.7722001728451454, 0.3196042182852584]
n_clusters:  5 ,Score:  1064626.7720802578 ,Average WPP:  [0.3150624118346602,
0.8575468230673395, 0.7712809620486166, 0.7234073156733456, 0.6715303817191675]
n_clusters:  6 ,Score:  991162.9409712646 ,Average WPP:  [0.31658642832316264,
0.796570012062977, 0.598012363836051, 0.7110271264240066, 0.8660359213952776,
0.7236795176430855]
n_clusters:  7 ,Score:  910609.208638535 ,Average WPP:  [0.30020823155956194,
0.6842178163080022, 0.514828706043734, 0.7195347274540805, 0.8714653874402203,
0.7775738498035812, 0.8087189476426153]
n_clusters:  8 ,Score:  858880.9589554905 ,Average WPP:  [0.400989896807283,
0.8092546527439258, 0.6750424887296048, 0.8031464138066088, 0.7186749317708392,
0.8922692654520599, 0.7504206889460409, 0.2796561534042405]
n_clusters:  9 ,Score:  818085.3530788469 ,Average WPP:  [0.2933571965690047,
0.8481669672023471, 0.6651799972201261, 0.49159870318708265, 0.7171684463497012,
0.8886048564238456, 0.30474691329717607, 0.7625451972481996, 0.8096720414424383]
n_clusters:  10 ,Score:  772452.9600767847 ,Average WPP:  [0.7771048131229971,
0.29392404866830063, 0.8932416752688889, 0.6174022064764464, 0.4753761757489192,
0.713809645868275, 0.8325386516184435, 0.8104266339431802, 0.8032103245551714,
0.3051964040487451]
best_n_clusters:  2 , best_score:  1792350.6600233421 , best_average_Wpp:
[0.3885809058510306, 0.7902486079468457]
```

[25]:
```
#Plot the line
plt.figure(figsize=(15,15))
plt.plot(Xs_1,scores_1)
plt.xticks(Xs_1)
plt.savefig("Kmeans_1.png")
plt.show()
```



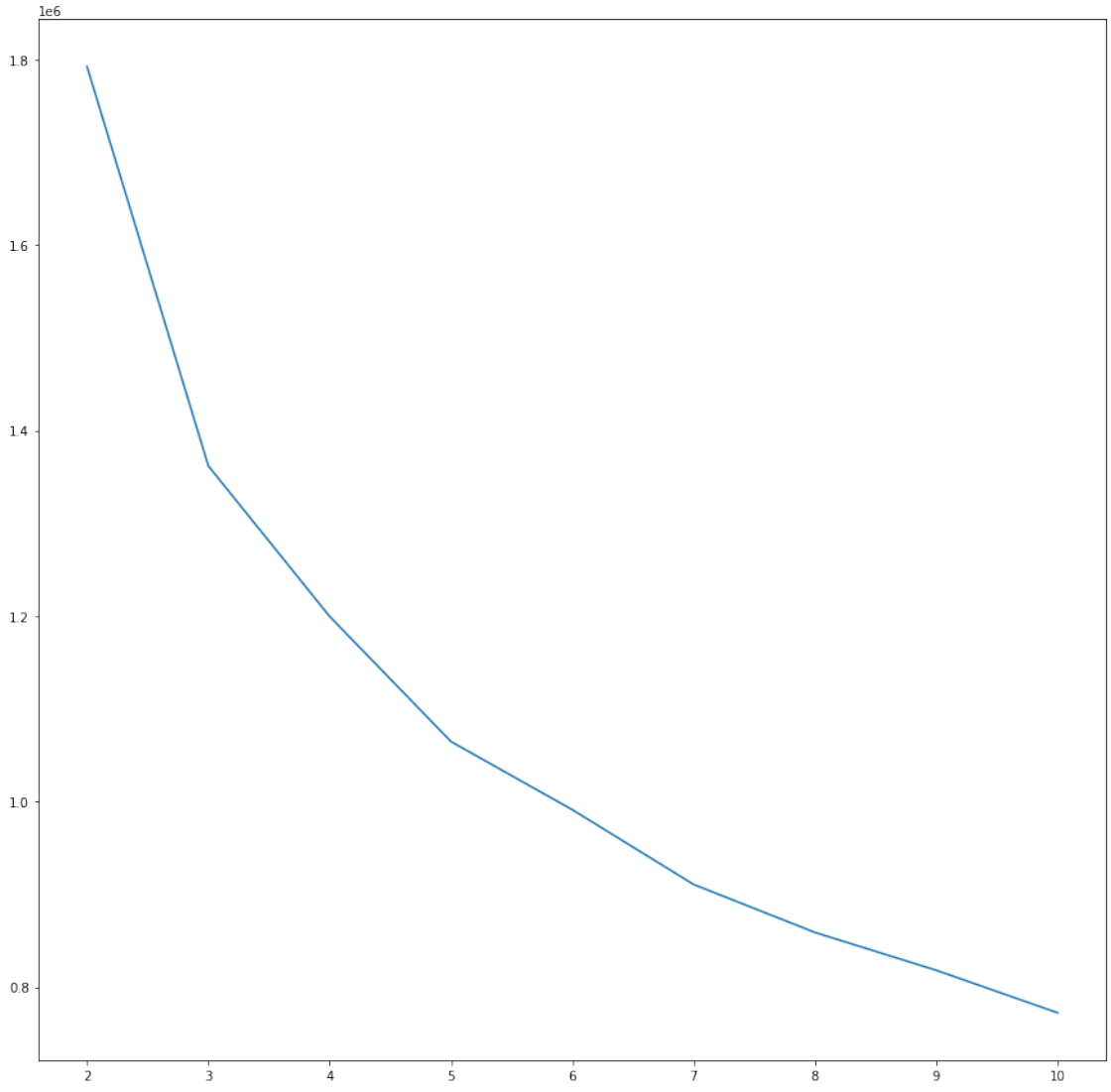
[26]:
```
#Used to do score VS n_clustet plotting
scores_2 = []
Xs_2 = []

#Used to store best number of clusters, best score and best average WPP
best_n_clusters = 0
best_score = 0
best_average_Wpp = 0

for i in range(2,11):

    #Assign number of clusters
    kmeans_2.n_clusters = i
```

```
    #Train the model
    kmeans_2.fit(cluster_train_X)

    #Get the score
    score = metrics.calinski_harabasz_score(cluster_train_X,kmeans_2.labels_)

    #Add a cluster column to store labels
    cluster_train_X["cluster"] = kmeans_2.labels_

    #Add a cluster column to store labels
    cluster_train_y["cluster"] = kmeans_2.labels_

    #Get the average WPP
    average_Wpp = getAverageWPP(cluster_train_y,i)

    #If the score for current n_cluster is higher, then replace the best values
    if(score>best_score):
        best_score = score
        best_n_clusters = i
        best_average_Wpp = average_Wpp

    #Print the values
    print("n_clusters: ",i,",Score: ",score, ",Average WPP: ",average_Wpp)

    #Append the n_cluster
    Xs_2.append(i)

    #Append the score
    scores_2.append(score)

    #Delete the cluster column
    cluster_train_X = cluster_train_X.drop(["cluster"],axis=1)
    cluster_train_y = cluster_train_y.drop(["cluster"],axis=1)

#Print the best values
print("best_n_clusters: ",best_n_clusters,", best_score: ", best_score,",␣
 ↪best_average_Wpp: ",best_average_Wpp)
```

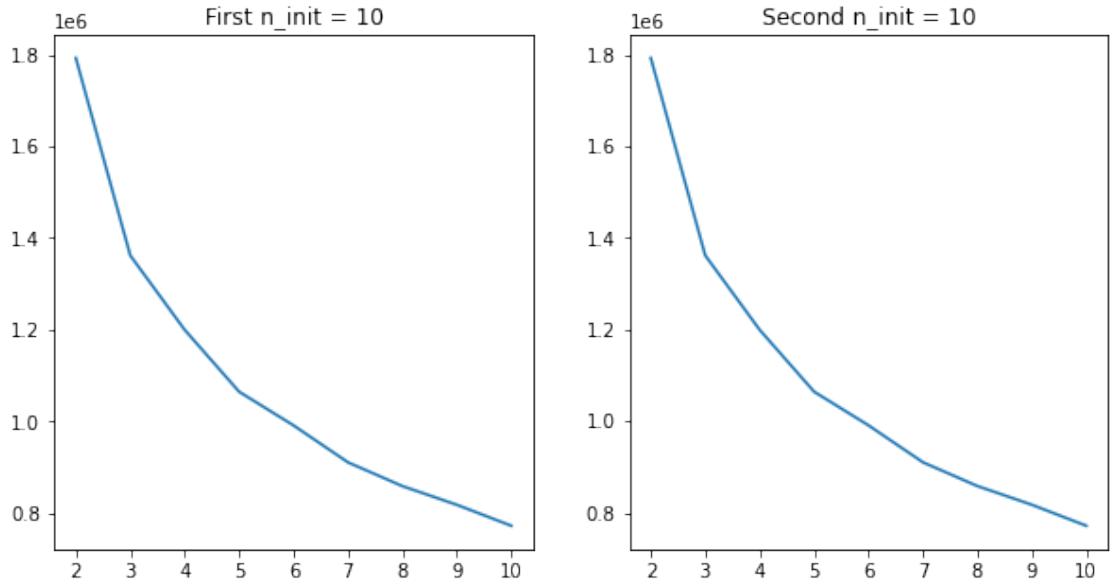
```
n_clusters:  2 ,Score:  1792349.6123606144 ,Average WPP:  [0.3890161459671143,
0.7906434722860924]
n_clusters:  3 ,Score:  1361779.2525691588 ,Average WPP:  [0.751092638708308,
0.3781039433760255, 0.7830249180831271]
n_clusters:  4 ,Score:  1199509.4283519469 ,Average WPP:  [0.32024102025118345,
0.7066015005474952, 0.83998074868226, 0.772040263777624]
n_clusters:  5 ,Score:  1064626.2218547056 ,Average WPP:  [0.7233112613149626,
0.3155909391338181, 0.857640647774139, 0.6734635085005738, 0.7712826209051771]
n_clusters:  6 ,Score:  991158.1977729003 ,Average WPP:  [0.31669195363275987,
```

```
0.8661258828558276, 0.5993759574381162, 0.7967923117116553, 0.7104084062934758,
0.7240362485993944]
n_clusters:  7 ,Score:  910609.7986223887 ,Average WPP:  [0.719525009417195,
0.30023668703870765, 0.5148132589722532, 0.6841971054946302, 0.8087242194114812,
0.7776412316917581, 0.8714591175926393]
n_clusters:  8 ,Score:  858880.6466607748 ,Average WPP:  [0.2796430784760554,
0.8031175675234687, 0.8922677736324943, 0.6750446238526574, 0.8092550759402741,
0.7503817424353163, 0.7186806468861572, 0.4009731400574724]
n_clusters:  9 ,Score:  818085.1387593657 ,Average WPP:  [0.2932423550636139,
0.8487038108392527, 0.7171700267245228, 0.8095942592615825, 0.30464000914500405,
0.7620038985163444, 0.4922597217480576, 0.8882603802433024, 0.6649202270513154]
n_clusters:  10 ,Score:  772303.3075732323 ,Average WPP:  [0.7749863381963252,
0.2939147145487552, 0.8923500108530498, 0.8364390709534076, 0.8014531593662628,
0.7147460596256066, 0.3052004522978602, 0.6248279880464741, 0.47897391907653475,
0.8131379782675238]
best_n_clusters:  2 , best_score:  1792349.6123606144 , best_average_Wpp:
[0.3890161459671143, 0.7906434722860924]
```

[27]:
```
#Plot the line
plt.figure(figsize=(15,15))
plt.plot(Xs_2,scores_2)
plt.xticks(Xs_2)
plt.savefig("Kmeans_2.png")
plt.show()
```



[28]:
```
plt.figure(figsize = (10,5))
ax = plt.subplot(1,2,1)
ax.plot(Xs_1,scores_1)
ax.set_title("First n_init = 10")
plt.xticks(Xs_1)
ax = plt.subplot(1,2,2)
ax.plot(Xs_2,scores_2)
ax.set_title("Second n_init = 10")
plt.xticks(Xs_1)
plt.savefig("combined_1.png")
plt.show()
```



Investigating 2 clusters

[8]:
```
def distinguish_clusters(kmeans,X_train_temp,y_train_temp,n_clusters,title):

    X_train_temp = X_train_temp.copy()
    y_train_temp = y_train_temp.copy()

    #Set the n_clusters
    kmeans.n_clusters = n_clusters

    #Get the clusters
    kmeans.fit(X_train_temp)

    #Get average wpp
    y_train_temp["cluster"] = kmeans.labels_
    average_Wpp = getAverageWPP(y_train_temp,n_clusters)

    #Get the centroids
    centroids = kmeans.cluster_centers_

    #Extract the centroids
    average_data = pd.DataFrame(centroids)

    xticks = []
    for i in range(23):
        xticks.append(i)

    #Plot the features
```

```
    average_data.T.plot.line(figsize = (15,10),title = title,xticks=xticks)
    locs, labels=plt.xticks()

    plt.xticks(locs,['as', 'bs', 'dD', 'DBNOs', 'hKs', 'hs','kP', 'kPs', 'ks',␣
 ↪'kSs', 'lK','mD', 'mP', 'nGs', 'rs','rD', 'rKs', 'sD', 'tKs','vDs', 'wD',␣
 ↪'wA', 'wPs'])

    #Here are the full names
#     plt.xticks(locs,['assists', 'boosts', 'damageDealt', 'DBNOs',␣
 ↪'headshotKills', 'heals',
#         'killPlace', 'killPoints', 'kills', 'killStreaks', 'longestKill',
#         'matchDuration', 'maxPlace', 'numGroups', 'revives',
#         'rideDistance', 'roadKills', 'swimDistance', 'teamKills',
#         'vehicleDestroys', 'walkDistance', 'weaponsAcquired', 'winPoints'])

    legends = []
    for i in range(n_clusters):
        legends.append("Cluster "+str(i+1)+"- average_Wpp "+str(average_Wpp[i]))
    plt.legend(legends)
    plt.show()
```

8 Dictionary for the x-axis

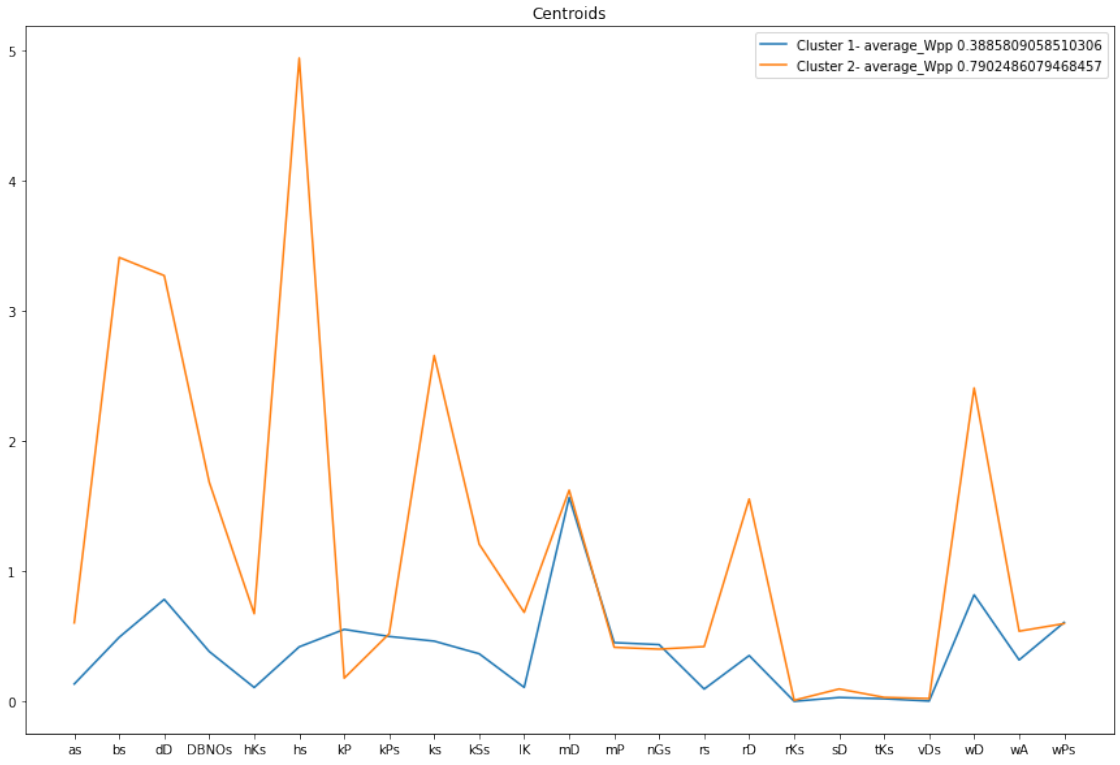
'as' = 'assists', 'bs' = 'boosts', 'dD' = 'damageDealt', 'DBNOs' = 'DBNOs', 'hKs' = 'headshotKills', 'hs' = 'heals'

'kP' = 'killPlace', 'kPs' = 'killPoints', 'ks' = 'kills', 'kSs' = 'killStreaks', 'lK' = 'longestKill','mD' = 'matchDuration'

'mP' = 'maxPlace', 'nGs' = 'numGroups', 'rs' = 'revives','rD' = 'rideDistance', 'rKs' = 'roadKills', 'sD' = 'swimDistance'

'tKs' = 'teamKills','vDs' = 'vehicleDestroys', 'wD' = 'walkDistance', 'wA' = 'weaponsAcquired', 'wPs' = 'winPoints'

[30]:
```
distinguish_clusters(kmeans_1,cluster_train_X,cluster_train_y,2,"Centroids")
```



8.1 Find the centroids together with most common matchtype, dominance of that type (eg 0.75 vs. 0.25) and average wPP

[11]:
```
def get_centroids(X_out,X_with,kmeans,n_clusters,y_train_temp):

    #Maintain the original data
    X = X_out.copy()
    X_with = X_with.copy()

    #Train kmeans
    kmeans.n_clusters = n_clusters
    kmeans.fit(X)

    #Get the centroids
    centroids = kmeans.cluster_centers_

    #Get the most common matchtype and dominance
    most_common_matchTypes = []
    dominance = []
    n_members = []
    X_with["cluster"] = kmeans.labels_
```

```
    for i in range(n_clusters):
        temp_cluster = X_with[X_with["cluster"] == i]
        most_common_type = temp_cluster['matchType'].mode()[0]
        most_common_type_count = temp_cluster['matchType'].
↪value_counts()[most_common_type]
        most_common_matchTypes.append(most_common_type)
        filtered_temp_cluster = temp_cluster[temp_cluster['matchType'] !=␣
↪most_common_type]
        filtered_common_type = filtered_temp_cluster['matchType'].mode()[0]
        filtered_type_count = filtered_temp_cluster['matchType'].
↪value_counts()[filtered_common_type]
        dominance.append((most_common_type_count-filtered_type_count)/
↪most_common_type_count)
        n_members.append(temp_cluster.shape[0])

    #Extract the centroids
    average_data = pd.DataFrame(centroids)
    average_data.columns = X_out.columns

    #Get the average WPP
    y_train_temp["cluster"] = kmeans.labels_
    average_Wpp = getAverageWPP(y_train_temp,n_clusters)

    #Add columns for most common matchtype, dominance and average WPP
    average_data["most_common"] = most_common_matchTypes
    average_data["dominance"] = dominance
    average_data["averageWpp"] = average_Wpp
    average_data["n_member"] = n_members

    average_data = average_data.sort_values('averageWpp',ascending=False)

    index = []
    for i in range(n_clusters):
        index.append("cluster_"+str(i+1))
    average_data.index = index

    return average_data
```

[47]:
```
centroids = get_centroids(cluster_train_X,scaled_train.
↪drop(['Id','groupId','matchId','winPlacePerc'],axis=1),kmeans_1,2,cluster_train_y)
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    display(centroids)
```

	assists	boosts	damageDealt	DBNOs	headshotKills	heals	\
cluster_1	0.604816	3.411910	3.272558	1.685508	0.674949	4.943793	
cluster_2	0.135101	0.494329	0.785060	0.384668	0.107998	0.420353	

```
           killPlace  killPoints     kills  killStreaks  longestKill  \
cluster_1   0.179964    0.521178  2.658613     1.208826     0.685203
cluster_2   0.554631    0.500281  0.464391     0.367322     0.109013

           matchDuration  maxPlace  numGroups   revives  rideDistance  \
cluster_1       1.624293  0.416862   0.402758  0.422380      1.556393
cluster_2       1.567636  0.452748   0.437524  0.096101      0.353750

           roadKills  swimDistance  teamKills  vehicleDestroys  walkDistance  \
cluster_1   0.009878      0.096933   0.032273         0.023116      2.409428
cluster_2   0.001820      0.031436   0.021581         0.003881      0.820336

           weaponsAcquired  winPoints most_common  dominance  averageWpp  \
cluster_1         0.540588   0.599156   squad-fpp   0.478670    0.790249
cluster_2         0.319716   0.607845   squad-fpp   0.418728    0.388581

           n_member
cluster_1    745302
cluster_2   2810992
```

9 Cluster Analysis with matchType

[8]:
```
#Do the clustering without Ids and matchType
mt_cluster_train = scaled_train.drop(['Id','groupId','matchId'],axis=1)

mt_cluster_train = pd.get_dummies(data = mt_cluster_train,columns =␣
 ↪["matchType"])

#Split X and y
mt_cluster_train_y = pd.DataFrame({'winPlacePerc':
 ↪mt_cluster_train['winPlacePerc']})
mt_cluster_train_X = mt_cluster_train.drop(['winPlacePerc'],axis=1)
```

[9]:
```
#Two different initializations for comparing
kmeans_3 = KMeans(random_state=10,n_init = 10)
kmeans_4 = KMeans(random_state=20,n_init = 10)
```

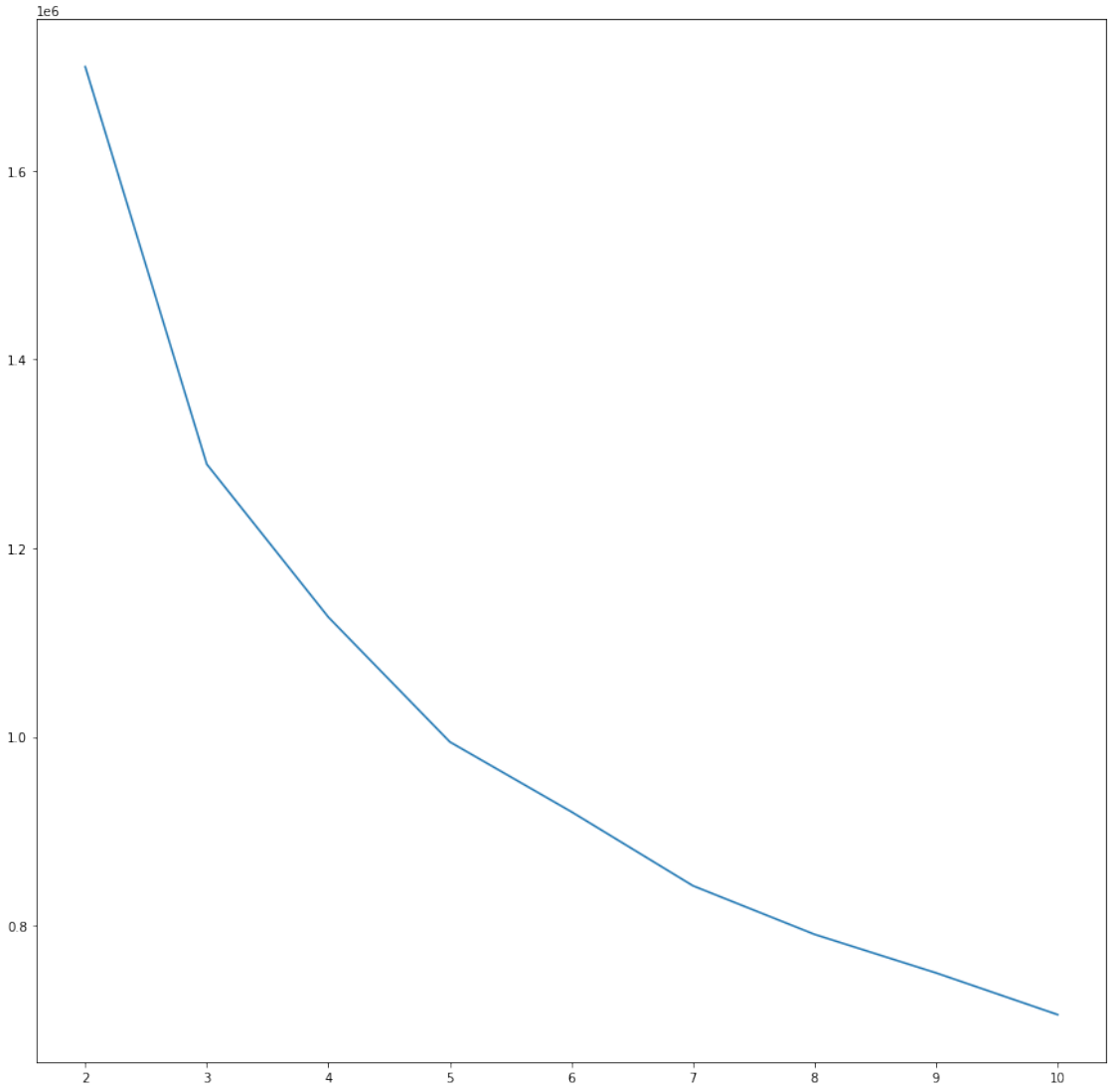
[37]:
```
#Used to do score VS n_clustet plotting
scores_3 = []
Xs_3 = []

#Used to store best number of clusters, best score and best average WPP
best_n_clusters = 0
best_score = 0
best_average_Wpp = 0
```

```
for i in range(2,11):

    #Assign number of clusters
    kmeans_3.n_clusters = i

    #Train the model
    kmeans_3.fit(mt_cluster_train_X)

    #Get the score
    score = metrics.calinski_harabasz_score(mt_cluster_train_X,kmeans_3.labels_)

    #Add a cluster column to store labels
    mt_cluster_train_X["cluster"] = kmeans_3.labels_

    #Add a cluster column to store labels
    mt_cluster_train_y["cluster"] = kmeans_3.labels_

    #Get the average WPP
    average_Wpp = getAverageWPP(mt_cluster_train_y,i)

    #If the score for current n_cluster is higher, then replace the best values
    if(score>best_score):
        best_score = score
        best_n_clusters = i
        best_average_Wpp = average_Wpp

    #Print the values
    print("n_clusters: ",i,",Score: ",score, ",Average WPP: ",average_Wpp)

    #Append the n_cluster
    Xs_3.append(i)

    #Append the score
    scores_3.append(score)

    #Delete the cluster column
    mt_cluster_train_X = mt_cluster_train_X.drop(["cluster"],axis=1)
    mt_cluster_train_y = mt_cluster_train_y.drop(["cluster"],axis=1)

#Print the best values
print("best_n_clusters: ",best_n_clusters,", best_score: ", best_score,",␣
 ↪best_average_Wpp: ",best_average_Wpp)
```

```
n_clusters:  2 ,Score:  1710187.8306510723 ,Average WPP:  [0.3886266562516232,
0.7902517285120596]
n_clusters:  3 ,Score:  1289059.4983299738 ,Average WPP:  [0.37818522220865347,
0.7513208877667753, 0.7831412805603016]
n_clusters:  4 ,Score:  1127081.5218089276 ,Average WPP:  [0.7061533351529624,
```

```
0.3198724953700752, 0.8393875812952073, 0.7720869991708986]
n_clusters:  5 ,Score:  994844.6757062995 ,Average WPP:  [0.31522873492564746,
0.6715137007960994, 0.771242955218717, 0.7233379129415962, 0.8578700943746352]
n_clusters:  6 ,Score:  920808.7053186877 ,Average WPP:  [0.31646769190668605,
0.8658495884526913, 0.7965375556983275, 0.7109547776868332, 0.72394522903329,
0.5972537543914803]
n_clusters:  7 ,Score:  842440.2297277856 ,Average WPP:  [0.30042875713970524,
0.513841753167081, 0.8713221755306036, 0.7193164694844639, 0.7783391383392742,
0.6839982412122718, 0.8087213732206348]
n_clusters:  8 ,Score:  790965.3167852819 ,Average WPP:  [0.2798124312078916,
0.4012657524741296, 0.8093338357023144, 0.7186231852503565, 0.8038997264200579,
0.8921601613009849, 0.7501792448065597, 0.6747336481254135]
n_clusters:  9 ,Score:  750142.05113529 ,Average WPP:  [0.2934827238365691,
0.4919051571001942, 0.8096616444215213, 0.7171066089715363, 0.30482722000594686,
0.7624972679827791, 0.6650365393936986, 0.8884190559644818, 0.8486203587270804]
n_clusters:  10 ,Score:  705935.8638051467 ,Average WPP:  [0.6167281000226297,
0.3054819818861749, 0.8104244273219492, 0.7137991714780763, 0.8321550856380739,
0.29426370988972034, 0.4746158876216466, 0.7780661525169091, 0.8934853801951089,
0.802723009837438]
best_n_clusters:  2 , best_score:  1710187.8306510723 , best_average_Wpp:
[0.3886266562516232, 0.7902517285120596]
```

[38]:
```
#Plot the line
plt.figure(figsize=(15,15))
plt.plot(Xs_3,scores_3)
plt.xticks(Xs_3)
plt.savefig("Kmeans_3.png")
plt.show()
```



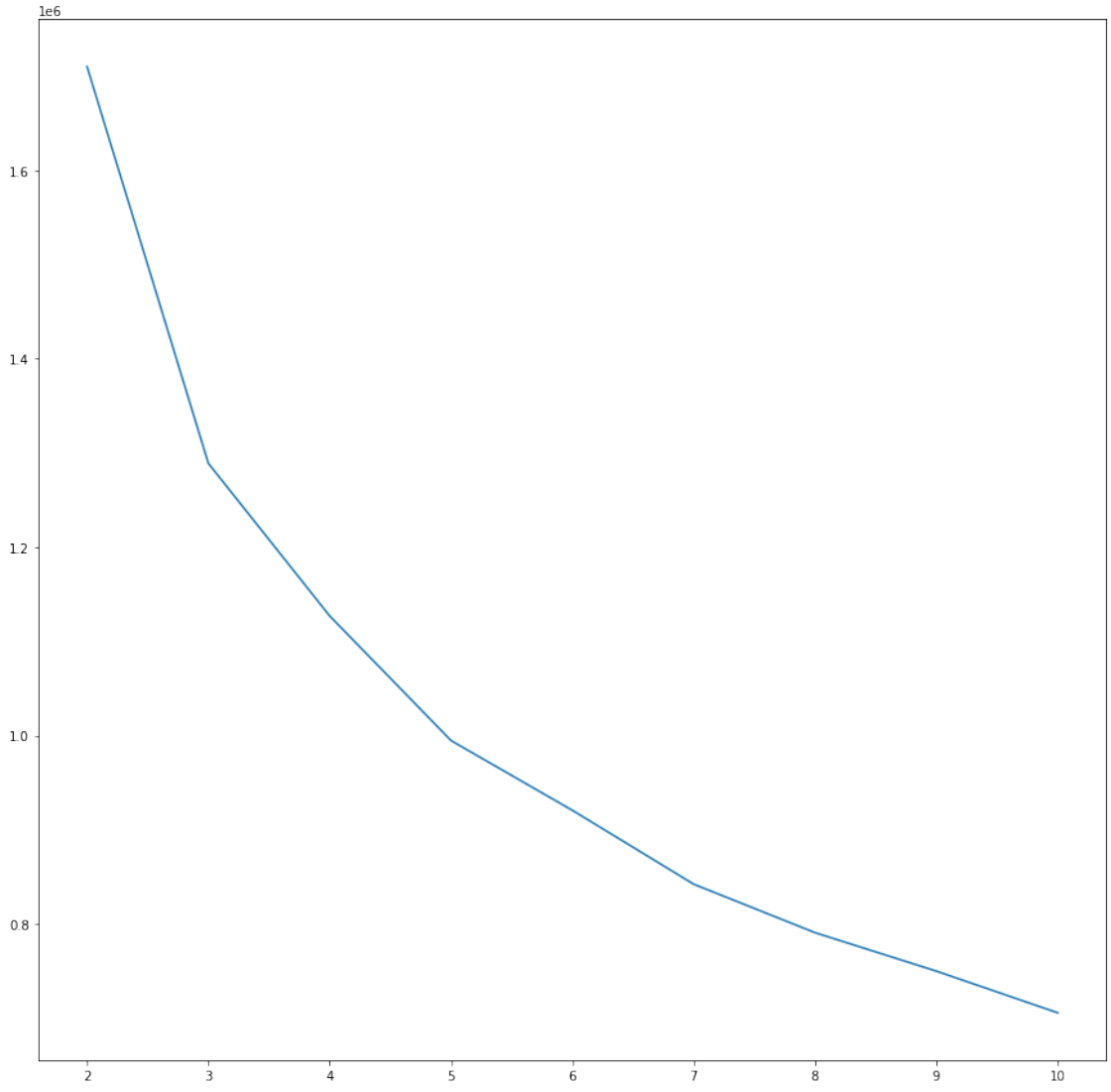
[39]:
```
#Used to do score VS n_clustet plotting
scores_4 = []
Xs_4 = []

#Used to store best number of clusters, best score and best average WPP
best_n_clusters = 0
best_score = 0
best_average_Wpp = 0

for i in range(2,11):

    #Assign number of clusters
    kmeans_4.n_clusters = i
```

```
    #Train the model
    kmeans_4.fit(mt_cluster_train_X)

    #Get the score
    score = metrics.calinski_harabasz_score(mt_cluster_train_X,kmeans_4.labels_)

    #Add a cluster column to store labels
    mt_cluster_train_X["cluster"] = kmeans_4.labels_

    #Add a cluster column to store labels
    mt_cluster_train_y["cluster"] = kmeans_4.labels_

    #Get the average WPP
    average_Wpp = getAverageWPP(mt_cluster_train_y,i)

    #If the score for current n_cluster is higher, then replace the best values
    if(score>best_score):
        best_score = score
        best_n_clusters = i
        best_average_Wpp = average_Wpp

    #Print the values
    print("n_clusters: ",i,",Score: ",score, ",Average WPP: ",average_Wpp)

    #Append the n_cluster
    Xs_4.append(i)

    #Append the score
    scores_4.append(score)

    #Delete the cluster column
    mt_cluster_train_X = mt_cluster_train_X.drop(["cluster"],axis=1)
    mt_cluster_train_y = mt_cluster_train_y.drop(["cluster"],axis=1)

#Print the best values
print("best_n_clusters: ",best_n_clusters,", best_score: ", best_score,",␣
 ↪best_average_Wpp: ",best_average_Wpp)
```

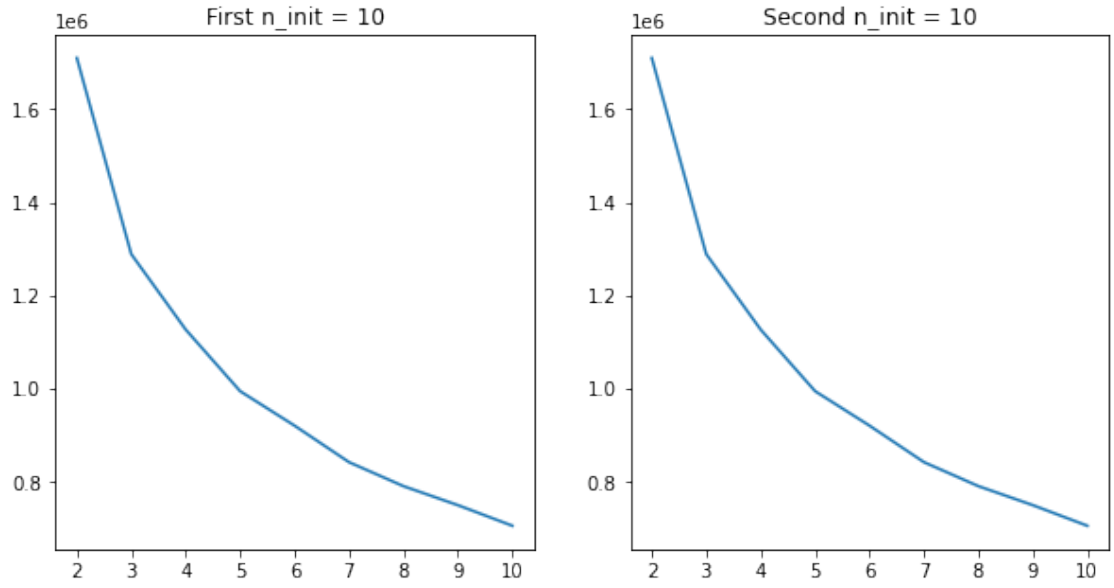
```
n_clusters:  2 ,Score:  1710187.9341075295 ,Average WPP:  [0.3889449317082806,
0.7905525506945659]
n_clusters:  3 ,Score:  1289060.0474784407 ,Average WPP:  [0.37819765006194783,
0.7831612814969082, 0.7513070498286142]
n_clusters:  4 ,Score:  1127081.2784272882 ,Average WPP:  [0.7720631518080785,
0.3199804711629512, 0.7062281068440426, 0.8395921198910564]
n_clusters:  5 ,Score:  994844.6345018139 ,Average WPP:  [0.7232260149845223,
0.3155811161379444, 0.8577959129455273, 0.6730228173263727, 0.7712583811122495]
n_clusters:  6 ,Score:  920806.0610967163 ,Average WPP:  [0.723899761736675,
```

```
0.31663172902745845, 0.59579965109502, 0.8660074806548917, 0.7117466445928842,
0.7959838677652115]
n_clusters:  7 ,Score:  842440.5980713853 ,Average WPP:  [0.7193097353998156,
0.3004433327796437, 0.871304514997703, 0.6839783239815023, 0.8087241844110178,
0.5137819578395955, 0.7783921955997475]
n_clusters:  8 ,Score:  790965.3609779464 ,Average WPP:  [0.7501589367495978,
0.2798056323352863, 0.8093355241445475, 0.6747184226333265, 0.892159960601814,
0.8039166303843027, 0.7186252064527986, 0.4012717261165576]
n_clusters:  9 ,Score:  750141.2652087045 ,Average WPP:  [0.29331880328471754,
0.49179327110939147, 0.6649556951560271, 0.8882718059622038, 0.7171566506636544,
0.3046599217917255, 0.8095952970545437, 0.8486289069212828, 0.7620978803467404]
n_clusters:  10 ,Score:  705937.3356403147 ,Average WPP:  [0.6172968722484862,
0.3048155365418092, 0.7758602886174436, 0.8924930141429441, 0.8104457555026771,
0.2935476183570557, 0.4759471914105742, 0.8340393276582332, 0.7138774451853527,
0.8019493489075683]
best_n_clusters:  2 , best_score:  1710187.9341075295 , best_average_Wpp:
[0.3889449317082806, 0.7905525506945659]
```

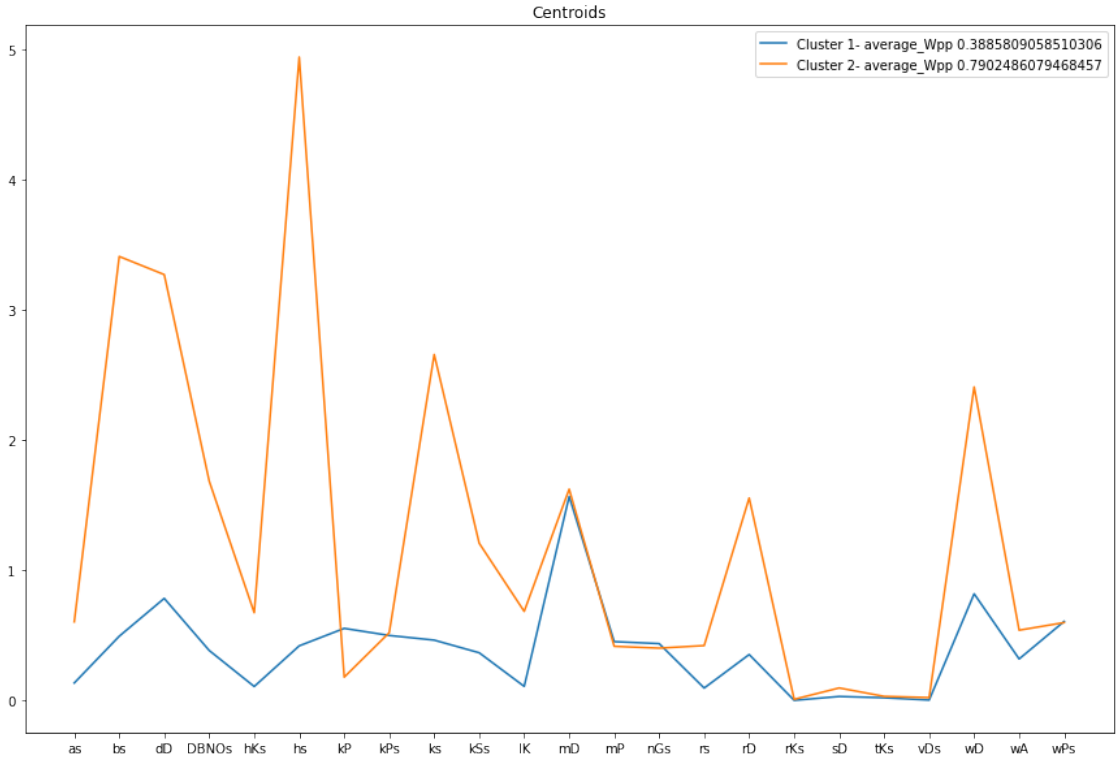
[40]:
```
#Plot the line
plt.figure(figsize=(15,15))
plt.plot(Xs_4,scores_4)
plt.xticks(Xs_4)
plt.savefig("Kmeans_4.png")
plt.show()
```



[41]:
```
plt.figure(figsize = (10,5))
ax = plt.subplot(1,2,1)
ax.plot(Xs_3,scores_3)
ax.set_title("First n_init = 10")
plt.xticks(Xs_3)
ax = plt.subplot(1,2,2)
ax.plot(Xs_4,scores_4)
ax.set_title("Second n_init = 10")
plt.xticks(Xs_4)
plt.savefig("combined_2.png")
plt.show()
```



[21]:
```
distinguish_clusters(kmeans_3,mt_cluster_train_X[[c for c in mt_cluster_train_X.
↪columns if c.lower()[:9] != 'matchtype']],mt_cluster_train_y,2,"Centroids")
```



[22]:
```
centroids = get_centroids(mt_cluster_train_X[[c for c in mt_cluster_train_X.
 ↪columns if c.lower()[:9] != 'matchtype']],scaled_train.
 ↪drop(['Id','groupId','matchId','winPlacePerc'],axis=1),kmeans_3,2,mt_cluster_train_y)
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    display(centroids)
```

	assists	boosts	damageDealt	DBNOs	headshotKills	heals	\
cluster_1	0.604816	3.411910	3.272558	1.685508	0.674949	4.943793	
cluster_2	0.135101	0.494329	0.785060	0.384668	0.107998	0.420353	

	killPlace	killPoints	kills	killStreaks	longestKill	\
cluster_1	0.179964	0.521178	2.658613	1.208826	0.685203	
cluster_2	0.554631	0.500281	0.464391	0.367322	0.109013	

	matchDuration	maxPlace	numGroups	revives	rideDistance	\
cluster_1	1.624293	0.416862	0.402758	0.422380	1.556393	
cluster_2	1.567636	0.452748	0.437524	0.096101	0.353750	

	roadKills	swimDistance	teamKills	vehicleDestroys	walkDistance	\
cluster_1	0.009878	0.096933	0.032273	0.023116	2.409428	
cluster_2	0.001820	0.031436	0.021581	0.003881	0.820336	

	weaponsAcquired	winPoints	most_common	dominance	averageWpp	\
cluster_1	0.540588	0.599156	squad-fpp	0.478670	0.790249	
cluster_2	0.319716	0.607845	squad-fpp	0.418728	0.388581	

	n_member
cluster_1	745302
cluster_2	2810992

10 Cluster Analysis with Kaggle Trick and matchType

[24]:
```
#Do the clustering without Ids and matchType
mt_cluster_train = scaled_train.drop(['Id','matchId'],axis=1)

mt_cluster_train = pd.get_dummies(data = mt_cluster_train,columns =␣
 ↪["matchType"])

mt_cluster_train = mt_cluster_train.groupby("groupId").transform('mean')
#Split X and y
mt_cluster_train_y = pd.DataFrame({'winPlacePerc':
 ↪mt_cluster_train['winPlacePerc']})
mt_cluster_train_X = mt_cluster_train.drop(['winPlacePerc'],axis=1)
```

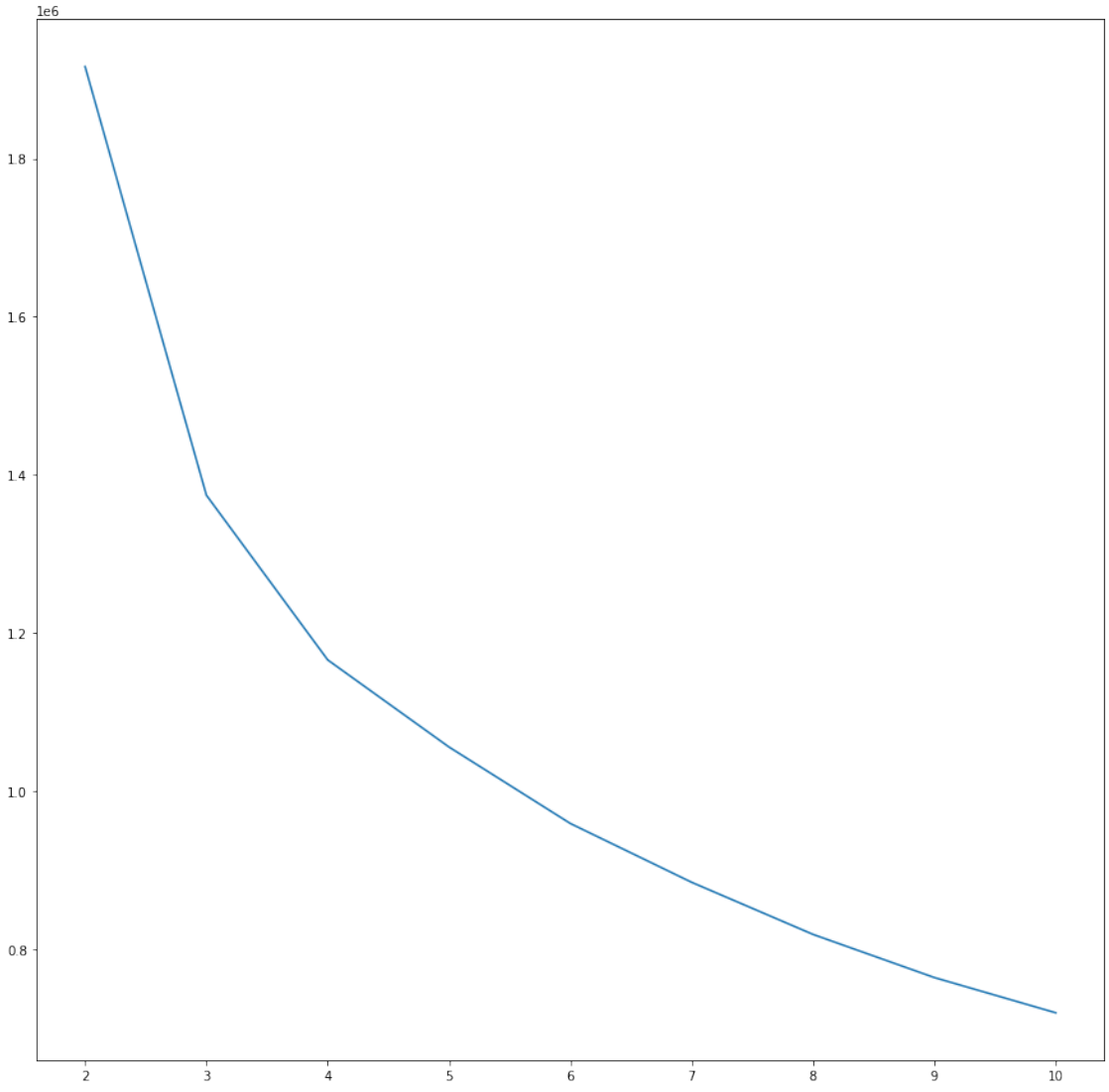
[25]:
```
#Two different initializations for comparing
kmeans_3 = KMeans(random_state=10,n_init = 10)
kmeans_4 = KMeans(random_state=20,n_init = 10)
```

[26]:
```
#Used to do score VS n_clustet plotting
scores_3 = []
Xs_3 = []

#Used to store best number of clusters, best score and best average WPP
best_n_clusters = 0
best_score = 0
best_average_Wpp = 0

for i in range(2,11):

    #Assign number of clusters
    kmeans_3.n_clusters = i

    #Train the model
    kmeans_3.fit(mt_cluster_train_X)

    #Get the score
    score = metrics.calinski_harabasz_score(mt_cluster_train_X,kmeans_3.labels_)

    #Add a cluster column to store labels
    mt_cluster_train_X["cluster"] = kmeans_3.labels_

    #Add a cluster column to store labels
    mt_cluster_train_y["cluster"] = kmeans_3.labels_

    #Get the average WPP
    average_Wpp = getAverageWPP(mt_cluster_train_y,i)

    #If the score for current n_cluster is higher, then replace the best values
    if(score>best_score):
        best_score = score
        best_n_clusters = i
        best_average_Wpp = average_Wpp

    #Print the values
    print("n_clusters: ",i,",Score: ",score, ",Average WPP: ",average_Wpp)

    #Append the n_cluster
    Xs_3.append(i)

    #Append the score
    scores_3.append(score)

    #Delete the cluster column
    mt_cluster_train_X = mt_cluster_train_X.drop(["cluster"],axis=1)
    mt_cluster_train_y = mt_cluster_train_y.drop(["cluster"],axis=1)
```

```
#Print the best values
print("best_n_clusters: ",best_n_clusters,", best_score: ", best_score,",␣
 ↪best_average_Wpp: ",best_average_Wpp)
```

```
n_clusters:  2 ,Score:  1916290.858605329 ,Average WPP:  [0.3640713375779634,
0.8122809114311556]
n_clusters:  3 ,Score:  1374292.4487806587 ,Average WPP:  [0.3042598194974294,
0.8555631995272949, 0.7375420225672885]
n_clusters:  4 ,Score:  1166014.0941148216 ,Average WPP:  [0.2909942472408201,
0.8852122342491946, 0.804614888523276, 0.7093313010090301]
n_clusters:  5 ,Score:  1055841.3212071992 ,Average WPP:  [0.7003455458877524,
0.28609242115381955, 0.8084904866985411, 0.7253503042137981, 0.8925881307884634]
n_clusters:  6 ,Score:  959191.3404727883 ,Average WPP:  [0.29777187988658754,
0.7258500503652664, 0.8968267120611985, 0.7274015217892552, 0.8102527295009663,
0.3083317880448458]
n_clusters:  7 ,Score:  884789.1443124753 ,Average WPP:  [0.7082981689882416,
0.9016252425100865, 0.8367819539052498, 0.72700897839986, 0.7457496984869071,
0.28512385458751216, 0.29628180654319236]
n_clusters:  8 ,Score:  818958.9555559459 ,Average WPP:  [0.2634076712342818,
0.8283755809939194, 0.6519110110654233, 0.8368585919007101, 0.9197099924254507,
0.7528481739852679, 0.7295743598614183, 0.27943217085892347]
n_clusters:  9 ,Score:  764385.3539111202 ,Average WPP:  [0.2565454837764349,
0.7459812906564822, 0.5170089498004815, 0.8605195965581866, 0.8384020094739466,
0.7323631410394418, 0.27062148308144485, 0.7320648313083178, 0.9196184090015496]
n_clusters:  10 ,Score:  719843.1947611735 ,Average WPP:  [0.5100645442019169,
0.8369837242115699, 0.8365032449093768, 0.7349070275835203, 0.2695482700677122,
0.7474710758993158, 0.8605370069791329, 0.2549097277891256, 0.6584689568934926,
0.9191146340835018]
best_n_clusters:  2 , best_score:  1916290.858605329 , best_average_Wpp:
[0.3640713375779634, 0.8122809114311556]
```

[27]:
```
#Plot the line
plt.figure(figsize=(15,15))
plt.plot(Xs_3,scores_3)
plt.xticks(Xs_3)
plt.savefig("Kmeans_5.png")
plt.show()
```



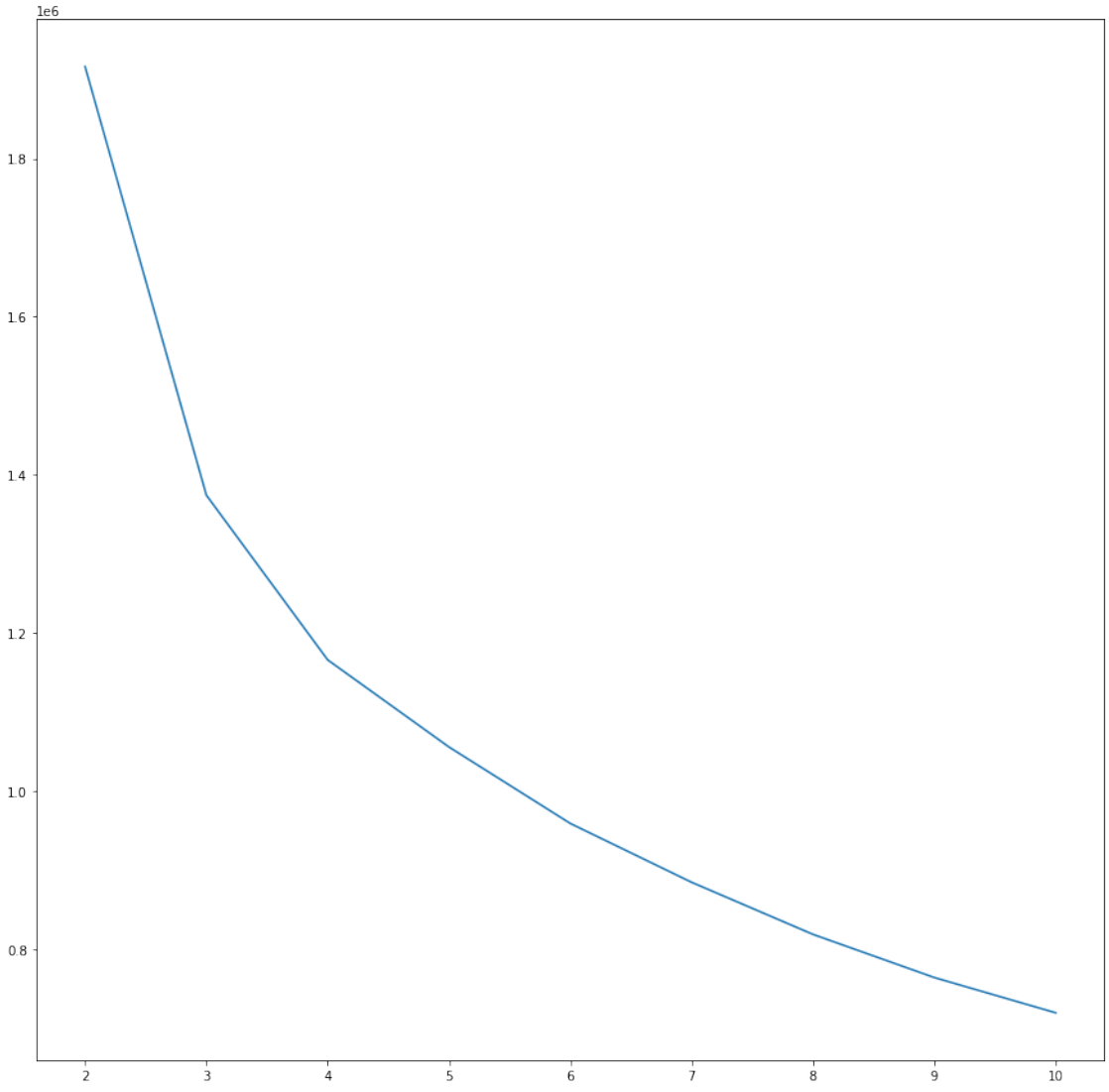
[28]:
```
#Used to do score VS n_clustet plotting
scores_4 = []
Xs_4 = []

#Used to store best number of clusters, best score and best average WPP
best_n_clusters = 0
best_score = 0
best_average_Wpp = 0

for i in range(2,11):

    #Assign number of clusters
    kmeans_4.n_clusters = i
```

```
    #Train the model
    kmeans_4.fit(mt_cluster_train_X)

    #Get the score
    score = metrics.calinski_harabasz_score(mt_cluster_train_X,kmeans_4.labels_)

    #Add a cluster column to store labels
    mt_cluster_train_X["cluster"] = kmeans_4.labels_

    #Add a cluster column to store labels
    mt_cluster_train_y["cluster"] = kmeans_4.labels_

    #Get the average WPP
    average_Wpp = getAverageWPP(mt_cluster_train_y,i)

    #If the score for current n_cluster is higher, then replace the best values
    if(score>best_score):
        best_score = score
        best_n_clusters = i
        best_average_Wpp = average_Wpp

    #Print the values
    print("n_clusters: ",i,",Score: ",score, ",Average WPP: ",average_Wpp)

    #Append the n_cluster
    Xs_4.append(i)

    #Append the score
    scores_4.append(score)

    #Delete the cluster column
    mt_cluster_train_X = mt_cluster_train_X.drop(["cluster"],axis=1)
    mt_cluster_train_y = mt_cluster_train_y.drop(["cluster"],axis=1)

#Print the best values
print("best_n_clusters: ",best_n_clusters,", best_score: ", best_score,",␣
 ↪best_average_Wpp: ",best_average_Wpp)
```

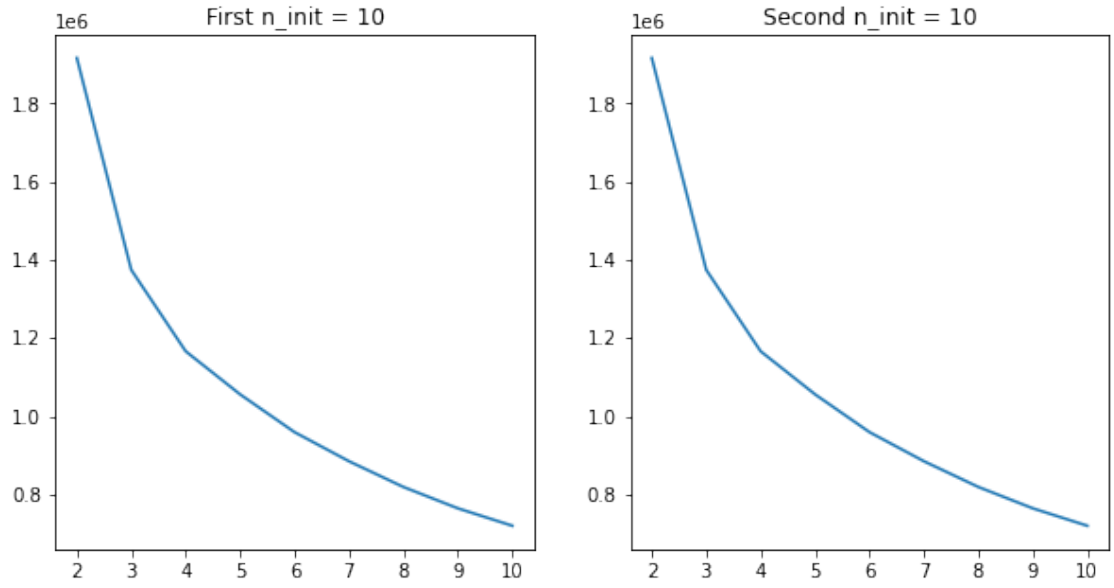
```
n_clusters:  2 ,Score:  1916290.1625511814 ,Average WPP:  [0.3640931932092816,
0.8123035924171162]
n_clusters:  3 ,Score:  1374288.1885960547 ,Average WPP:  [0.3038609830874633,
0.8557990050175137, 0.7368553324783602]
n_clusters:  4 ,Score:  1166014.3386481656 ,Average WPP:  [0.29096225747790877,
0.8851391154806112, 0.709268454942704, 0.8046224814746447]
n_clusters:  5 ,Score:  1055841.1528627013 ,Average WPP:  [0.28655092025988566,
0.8086497616919869, 0.7253036760347773, 0.7012069556994908, 0.8927742661430832]
n_clusters:  6 ,Score:  959191.1093305381 ,Average WPP:  [0.29723204211060744,
```

```
0.8100865282530788, 0.7275794003232312, 0.7247987069110952, 0.30789660682343695,
0.8963904972758694]
n_clusters:  7 ,Score:  884789.8577816094 ,Average WPP:  [0.7461748870708692,
0.708104742822329, 0.7270258183331206, 0.2849684748451972, 0.9014741702548742,
0.8369786214108657, 0.29614878144062085]
n_clusters:  8 ,Score:  818958.8318446432 ,Average WPP:  [0.8277210312636583,
0.752748531134918, 0.6520238457796379, 0.2633114231221987, 0.729611088591338,
0.8367202789426864, 0.9197256392151402, 0.2793715946441546]
n_clusters:  9 ,Score:  764385.4211554696 ,Average WPP:  [0.256318433147487,
0.8600226171910487, 0.7324729179904153, 0.2703980206005667, 0.8384289018300067,
0.9196051502491684, 0.7318215868287556, 0.745945752292025, 0.5159628529288789]
n_clusters:  10 ,Score:  719842.1508459456 ,Average WPP:  [0.255098241682737,
0.8370217581118695, 0.7350935556349896, 0.8607385847922532, 0.6579207603063753,
0.2697076170691926, 0.7480306243929036, 0.5105795167069992, 0.8362304074046774,
0.919152823614745]
best_n_clusters:  2 , best_score:  1916290.1625511814 , best_average_Wpp:
[0.3640931932092816, 0.8123035924171162]
```

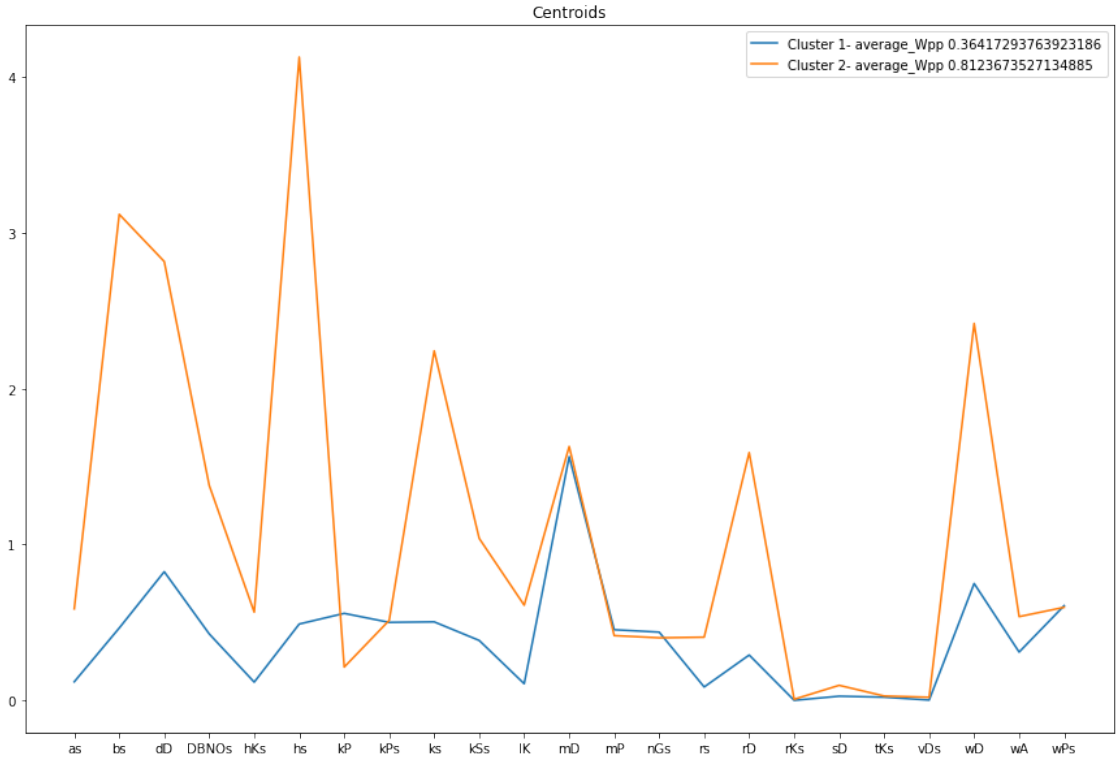
[29]:
```
#Plot the line
plt.figure(figsize=(15,15))
plt.plot(Xs_4,scores_4)
plt.xticks(Xs_4)
plt.savefig("Kmeans_6.png")
plt.show()
```



[30]:
```
plt.figure(figsize = (10,5))
ax = plt.subplot(1,2,1)
ax.plot(Xs_3,scores_3)
ax.set_title("First n_init = 10")
plt.xticks(Xs_3)
ax = plt.subplot(1,2,2)
ax.plot(Xs_4,scores_4)
ax.set_title("Second n_init = 10")
plt.xticks(Xs_4)
plt.savefig("combined_3.png")
plt.show()
```



[31]:
```
distinguish_clusters(kmeans_3,mt_cluster_train_X[[c for c in mt_cluster_train_X.
↪columns if c.lower()[:9] != 'matchtype']],mt_cluster_train_y,2,"Centroids")
```



[32]:
```
centroids = get_centroids(mt_cluster_train_X[[c for c in mt_cluster_train_X.
 ↪columns if c.lower()[:9] != 'matchtype']],scaled_train.
 ↪drop(['Id','groupId','matchId','winPlacePerc'],axis=1),kmeans_3,2,mt_cluster_train_y)
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    display(centroids)
```

	assists	boosts	damageDealt	DBNOs	headshotKills	heals	killPlace	killPoints	kills	killStreaks	longestKill	matchDuration	maxPlace	numGroups	revives	rideDistance	roadKills	swimDistance	teamKills	vehicleDestroys	walkDistance	weaponsAcquired	winPoints	most_common	dominance	averageWpp	n_member
cluster_1	0.588057	3.117071	2.814719	1.379293	0.566957	4.125092	0.214942	0.514417	2.242005	1.039928	0.611688	1.629205	0.416697	0.402611	0.406629	1.589989	0.008790	0.098015	0.029557	0.021558	2.417757	0.538308	0.598191	squad-fpp	0.491311	0.812367	861603
cluster_2	0.120644	0.465389	0.826259	0.427522	0.118550	0.490849	0.559268	0.501557	0.504800	0.385730	0.108175	1.563680	0.454313	0.439037	0.087370	0.292326	0.001827	0.028329	0.021997	0.003567	0.750680	0.311131	0.608519	squad-fpp	0.411453	0.364173	2694691

10.1 Cluster Analysis For Each Match Type

[20]:
```
#Extract each matchType
matchType_crashfpp=scaled_train[scaled_train['matchType']=='crashfpp']
matchType_crashtpp=scaled_train[scaled_train['matchType']=='crashtpp']
matchType_duo=scaled_train[scaled_train['matchType']=='duo']
matchType_duo_fpp=scaled_train[scaled_train['matchType']=='duo-fpp']
matchType_flarefpp=scaled_train[scaled_train['matchType']=='flarefpp']
matchType_flaretpp=scaled_train[scaled_train['matchType']=='flaretpp']
matchType_normal_duo=scaled_train[scaled_train['matchType']=='normal-duo']
matchType_normal_duo_fpp=scaled_train[scaled_train['matchType']=='normal-duo-fpp']
matchType_normal_solo=scaled_train[scaled_train['matchType']=='normal-solo']
matchType_normal_solo_fpp=scaled_train[scaled_train['matchType']=='normal-solo-fpp']
matchType_normal_squad=scaled_train[scaled_train['matchType']=='normal-squad']
matchType_normal_squad_fpp=scaled_train[scaled_train['matchType']=='normal-squad-fpp']
matchType_solo=scaled_train[scaled_train['matchType']=='solo']
matchType_solo_fpp=scaled_train[scaled_train['matchType']=='solo-fpp']
```

```
matchType_squad=scaled_train[scaled_train['matchType']=='squad']
matchType_squad_fpp=scaled_train[scaled_train['matchType']=='squad-fpp']
```

[10]:
```
#Add all match types to a list
allTypes =␣
 ↪[matchType_crashfpp,matchType_crashtpp,matchType_duo,matchType_duo_fpp,matchType_flarefpp,m

#Add all names of match types to a list
names =␣
 ↪["crashfpp","crashtpp","duo","duo_fpp","flarefpp","flaretpp","normal_duo","normal_duo_fpp",
```

[35]:
```
for index in range(len(allTypes)):
    #Reset the indices
    allTypes[index] = allTypes[index].reset_index(drop=True)
```

[13]:
```
kmeans_mt = KMeans(random_state=0,n_init = 50)
```

[37]:
```
plt.figure(figsize = (20,20))
plt.subplots_adjust(hspace=.5,wspace=.5)

#Used to allocate position for sub plots
p = 1

#Used to store the best number of clusters for each type
n_clusters_collection = []

for index in range(len(allTypes)):

    #Get the name
    name = names[index]

    #Get the dataframe with specific type, without changing the original data
    single_type = allTypes[index].copy()

    #Extract X and y
    single_y = single_type["winPlacePerc"]
    single_X = single_type.drop(["winPlacePerc"],axis=1)

    #Delete matchType column
    single_X_temp = single_X.drop(['Id','groupId','matchId','matchType'],axis=1)

    #Convert numpy array to pandas dataframe
    single_y_temp = pd.DataFrame({"winPlacePerc": single_y})

    scores = []
    Xs = []
```

```
    best_n_clusters = 0
    best_score = 0
    best_average_Wpp = 0
    print(name+":")
    for i in range(2,11):

        #Set the number of clusters
        kmeans_mt.n_clusters = i

        #Train the kmeans_mt
        kmeans_mt.fit(single_X_temp)

        #Get the score
        score = metrics.calinski_harabasz_score(single_X_temp,kmeans_mt.labels_)

        #Add a cluster column
        single_X_temp["cluster"] = kmeans_mt.labels_
        single_y_temp["cluster"] = kmeans_mt.labels_

        #Calculate the average WPP
        average_Wpp = getAverageWPP(single_y_temp,i)

        #If current score is larger than the best score, then replace the values
        if(score>best_score):
            best_score = score
            best_n_clusters = i
            best_average_Wpp = average_Wpp

        print("n_clusters: ",i,",Score: ",score, ",Average WPP: ",average_Wpp)

        #Append X and scores with current value
        Xs.append(i)
        scores.append(score)

        #Delete the cluster column
        single_X_temp = single_X_temp.drop(["cluster"],axis=1)
        single_y_temp = single_y_temp.drop(["cluster"],axis=1)

    print("best_n_clusters: ",best_n_clusters,", best_score: ", best_score,",␣
↪best_average_Wpp: ",best_average_Wpp,"\n")

    #Append the best n_clusters
    n_clusters_collection.append(best_n_clusters)

    #Plot the line
    ax = plt.subplot(4,4,p)
    plt.xticks(Xs)
```

```
    ax.plot(Xs,scores)
    ax.set_title(name)
    p+=1

plt.savefig("Individual_analysis.png")
plt.show()
```

```
crashfpp:
n_clusters:  2 ,Score:  2132.859710527504 ,Average WPP:  [0.759188740245262,
0.3221620516047829]
n_clusters:  3 ,Score:  2080.0936429806734 ,Average WPP:  [0.30491379657603224,
0.7532017150395779, 0.7036134171907757]
n_clusters:  4 ,Score:  1950.6437177225255 ,Average WPP:  [0.7120583743842365,
0.2913783333333333, 0.7889735294117648, 0.6944931603773585]
n_clusters:  5 ,Score:  1770.596379910717 ,Average WPP:  [0.253904432696218,
0.7228820408163265, 0.7158677198975235, 0.596153744493392, 0.8175401442307693]
n_clusters:  6 ,Score:  1620.335382399982 ,Average WPP:  [0.47476501501501506,
0.7137726051924799, 0.7518423999999999, 0.8421627345844505, 0.6755265350877193,
0.248855033557047]
n_clusters:  7 ,Score:  1515.5152680162803 ,Average WPP:  [0.8441707395498392,
0.24122633272058824, 0.6272645833333333, 0.4701076682316119, 0.7475706766917293,
0.7911408114558471, 0.685068267223382]
n_clusters:  8 ,Score:  1418.1942862023536 ,Average WPP:  [0.2203383733055266,
0.6906257111597374, 0.8381658385093168, 0.5647947916666667, 0.7972400000000001,
0.6375864864864865, 0.4062869491525423, 0.7615596153846153]
n_clusters:  9 ,Score:  1357.906302925264 ,Average WPP:  [0.7813677631578947,
0.21899407756813416, 0.7586787878787877, 0.8148431870669746, 0.6592120358514725,
0.6259755274261604, 0.8662487179487178, 0.5717989539748954, 0.4007765625]
n_clusters:  10 ,Score:  1284.337262437394 ,Average WPP:  [0.20873673580786029,
0.8658751633986929, 0.7781217241379311, 0.7572571428571427, 0.8137399509803922,
0.625481974248927, 0.393345471349353, 0.5683211293260473, 0.5692372767857143,
0.7464161904761905]
best_n_clusters:  2 , best_score:  2132.859710527504 , best_average_Wpp:
[0.759188740245262, 0.3221620516047829]

crashtpp:
n_clusters:  2 ,Score:  119.65091581586663 ,Average WPP:  [0.38042304147465433,
0.8112230769230769]
n_clusters:  3 ,Score:  99.46400759245593 ,Average WPP:  [0.8579344827586207,
0.3342470588235294, 0.7257121212121213]
n_clusters:  4 ,Score:  95.38666776461312 ,Average WPP:  [0.7736611111111111,
0.33100944444444447, 0.6891545454545455, 0.9049722222222223]
n_clusters:  5 ,Score:  94.29105757983727 ,Average WPP:  [0.28743124999999997,
0.9133882352941176, 0.6074409090909091, 0.8194625, 0.6748255319148936]
n_clusters:  6 ,Score:  89.36724060123522 ,Average WPP:  [0.27646478873239433,
0.9306833333333334, 0.6074836065573771, 0.8194625, 0.6565404761904761,
0.8370565217391305]
n_clusters:  7 ,Score:  87.43720709573158 ,Average WPP:  [0.40966612903225813,
```

0.8370565217391305, 0.2331039603960396, 0.8194625, 0.6563324324324324,
0.9306833333333334, 0.6813800000000001]
n_clusters: 8 ,Score: 83.73620821474702 ,Average WPP: [0.6546393939393939,
0.8194625, 0.209225, 0.8409388888888889, 0.7193423076923078, 0.4022189655172414,
0.9306833333333334, 0.6563324324324324]
n_clusters: 9 ,Score: 79.04046636512744 ,Average WPP: [0.7909318181818182,
0.6589571428571427, 0.209225, 0.8194625, 0.9306833333333334, 0.648936111111111,
0.798576923076923, 0.8422599999999999, 0.40389180327868857]
n_clusters: 10 ,Score: 76.22868488332183 ,Average WPP: [0.18689887640449437,
0.8315823529411764, 0.9018333333333333, 0.927, 0.3919320754716981, 0.62155,
0.9306833333333334, 0.6547714285714286, 0.6831791666666667, 0.6275470588235295]
best_n_clusters: 2 , best_score: 119.65091581586663 , best_average_Wpp:
[0.38042304147465433, 0.8112230769230769]

duo:
n_clusters: 2 ,Score: 132500.657252623 ,Average WPP: [0.8069829131599803,
0.3958567618586151]
n_clusters: 3 ,Score: 100741.19077087368 ,Average WPP: [0.763073874433887,
0.807902238825884, 0.38578655159243]
n_clusters: 4 ,Score: 89238.743922756 ,Average WPP: [0.7155666941760926,
0.7875717039757079, 0.32613478378310473, 0.8706351221039449]
n_clusters: 5 ,Score: 81375.63935752802 ,Average WPP: [0.32122686942166145,
0.7860767947572238, 0.6731342296346721, 0.8887416849429024, 0.7393050342781132]
n_clusters: 6 ,Score: 75763.46783083731 ,Average WPP: [0.3224827915843851,
0.8933601249710715, 0.6101356628377743, 0.8149039148351649, 0.7394118250479089,
0.7178902361751152]
n_clusters: 7 ,Score: 69542.33801743845 ,Average WPP: [0.735283633817609,
0.4602072172344923, 0.31832607233238297, 0.913151241110962, 0.8145031569090828,
0.8197644509886411, 0.6912638400467584]
n_clusters: 8 ,Score: 65237.21995628208 ,Average WPP: [0.4152628029949565,
0.2837601199544698, 0.8085101190777806, 0.7395424140975646, 0.7370046649414328,
0.8270191887125221, 0.6906828430557863, 0.9209427116583069]
n_clusters: 9 ,Score: 61804.86587480015 ,Average WPP: [0.7563584546793195,
0.2945887078585692, 0.8271661700703047, 0.7345257908550905, 0.5258798404932645,
0.6771218054898249, 0.31298406395337985, 0.8572376916817925, 0.9230970173985088]
n_clusters: 10 ,Score: 58685.009303414416 ,Average WPP: [0.29397473109691163,
0.880769081779053, 0.7874840763561677, 0.7711158621880666, 0.81530140311804,
0.3109261437777287, 0.9272162920097929, 0.6565817476361265, 0.4300979540880022,
0.7278949174690509]
best_n_clusters: 2 , best_score: 132500.657252623 , best_average_Wpp:
[0.8069829131599803, 0.3958567618586151]

duo_fpp:
n_clusters: 2 ,Score: 444539.83191625506 ,Average WPP: [0.8095440493928445,
0.40105553789998327]
n_clusters: 3 ,Score: 332346.0498526078 ,Average WPP: [0.7735367840185475,
0.38740012553163905, 0.7968634236490358]
n_clusters: 4 ,Score: 292661.41823965235 ,Average WPP: [0.785009791825613,

```
0.33252759057284714, 0.7046743585751806, 0.8769762818352989]
n_clusters:  5 ,Score:  262273.5889906712 ,Average WPP:  [0.3319737839115912,
0.8079432765217391, 0.8859794049593547, 0.5919244995383658, 0.746705341897789]
n_clusters:  6 ,Score:  243604.75063309283 ,Average WPP:  [0.607723180197189,
0.3298247951233557, 0.88973192572758, 0.8229906296743063, 0.7499029065337136,
0.7190246801210105]
n_clusters:  7 ,Score:  225287.37137305594 ,Average WPP:  [0.34562004260670937,
0.7193822514258044, 0.8931621099348158, 0.8231112575207598, 0.7533812598425197,
0.33322733019776174, 0.624271130070188]
n_clusters:  8 ,Score:  212527.30007003527 ,Average WPP:  [0.5448778464131674,
0.32333615126497994, 0.8935490519294903, 0.8352025671713242,
0.33630357894514185, 0.6852935885285032, 0.822368264005533, 0.744884444673027]
n_clusters:  9 ,Score:  202639.56878992607 ,Average WPP:  [0.3189849767936029,
0.8543932869771055, 0.7787146447311319, 0.6768664409860696, 0.5111737410756306,
0.7431574731241771, 0.922230541727672, 0.3038514458879062, 0.8333717571297148]
n_clusters:  10 ,Score:  191847.0869545228 ,Average WPP:  [0.7790471934552453,
0.3031002273627202, 0.6476260557243959, 0.787792805067044, 0.43187099469464085,
0.8814876850088537, 0.9247278170114367, 0.738104834640487, 0.82556870604782,
0.316743436909298]
best_n_clusters:  2 , best_score:  444539.83191625506 , best_average_Wpp:
[0.8095440493928445, 0.40105553789998327]

flarefpp:
n_clusters:  2 ,Score:  284.68310861831804 ,Average WPP:  [0.7542113821138211,
0.370935368956743]
n_clusters:  3 ,Score:  233.38825366412723 ,Average WPP:  [0.7235978260869566,
0.33616396648044694, 0.91712]
n_clusters:  4 ,Score:  228.35886056184748 ,Average WPP:  [0.9127578947368422,
0.26386894197952215, 0.707974, 0.6980277777777777]
n_clusters:  5 ,Score:  202.9161478824859 ,Average WPP:  [0.2546389090909091,
0.7014096153846153, 0.7027579710144927, 0.9127578947368422, 0.6555831683168316]
n_clusters:  6 ,Score:  184.97743038919413 ,Average WPP:  [0.6156391891891893,
0.7260000000000001, 0.7134157303370786, 0.23915378787878788, 0.9127578947368422,
0.694338596491228]
n_clusters:  7 ,Score:  171.4527251708784 ,Average WPP:  [0.22635163934426228,
0.8551724137931035, 0.6976560606060604, 0.9302900000000001, 0.6869400000000001,
0.7030543859649123, 0.49509538461538466]
n_clusters:  8 ,Score:  163.97479528096352 ,Average WPP:  [0.7260000000000001,
0.22391516393442623, 0.7458408163265305, 0.9364333333333335, 0.6509309523809524,
0.49394461538461537, 0.6904261538461538, 0.868103448275862]
n_clusters:  9 ,Score:  155.09952400209744 ,Average WPP:  [0.22051957446808512,
0.9168000000000001, 0.797125641025641, 0.7260000000000001, 0.84,
0.6184896551724137, 0.6509976744186047, 0.47188833333333335, 0.7485020408163264]
n_clusters:  10 ,Score:  147.88458200946212 ,Average WPP:  [0.7609148936170211,
0.2207637931034483, 0.9168000000000001, 0.5426, 0.45305555555555554,
0.7470966666666666, 0.84, 0.7902424242424242, 0.617420754716981,
0.7091999999999998]
best_n_clusters:  2 , best_score:  284.68310861831804 , best_average_Wpp:
```

[0.7542113821138211, 0.370935368956743]

flaretpp:
n_clusters: 2 ,Score: 1230.2851916380607 ,Average WPP: [0.31774108425865444,
0.777791568627451]
n_clusters: 3 ,Score: 958.1356484684813 ,Average WPP: [0.2951292971468337,
0.7694248691099477, 0.7437576576576576]
n_clusters: 4 ,Score: 880.2665110260192 ,Average WPP: [0.7607916167664671,
0.660663671875, 0.8417108571428571, 0.22790556023588882]
n_clusters: 5 ,Score: 782.3183360427055 ,Average WPP: [0.6218855704697986,
0.6656229508196722, 0.8495335526315789, 0.7694835714285714, 0.22507993019197206]
n_clusters: 6 ,Score: 715.378076029159 ,Average WPP: [0.6128937086092716,
0.7807244239631336, 0.6581999999999999, 0.22748210251954823, 0.7717037383177568,
0.9132048780487805]
n_clusters: 7 ,Score: 673.3816739703934 ,Average WPP: [0.5317629441624366,
0.21677202268431003, 0.8473891891891892, 0.7616934065934063, 0.6404191176470589,
0.6120382550335571, 0.9171688888888888]
n_clusters: 8 ,Score: 628.1766707025889 ,Average WPP: [0.20867945344129557,
0.6405045918367348, 0.856024107142857, 0.8150656000000002, 0.44339377593361,
0.6103883802816902, 0.91385, 0.756664705882353]
n_clusters: 9 ,Score: 590.0419706081764 ,Average WPP: [0.7521915662650602,
0.6146554913294798, 0.7654342105263158, 0.2934488599348534, 0.20995243619489556,
0.8094519685039371, 0.9365249999999999, 0.60865625, 0.8992327868852459]
n_clusters: 10 ,Score: 557.6978460018894 ,Average WPP: [0.7508909090909092,
0.5980032894736842, 0.7804770491803278, 0.2436616600790514, 0.5577071428571428,
0.8068605504587156, 0.9365249999999999, 0.6355335443037975, 0.8998396825396825,
0.16402939632545932]
best_n_clusters: 2 , best_score: 1230.2851916380607 , best_average_Wpp:
[0.31774108425865444, 0.777791568627451]

normal_duo:
n_clusters: 2 ,Score: 251.4727882572557 ,Average WPP: [0.4949563909774436,
0.6217233333333334]
n_clusters: 3 ,Score: 185.561346299911 ,Average WPP: [0.511225925925926,
0.7914529411764706, 0.4808663865546219]
n_clusters: 4 ,Score: 158.817127646323 ,Average WPP: [0.7525333333333334,
0.48494152542372887, 0.6921545454545455, 0.5148520000000001]
n_clusters: 5 ,Score: 146.67781666133308 ,Average WPP: [0.5173214285714286,
0.5572250000000001, 0.7632625, 0.4679451923076923, 0.7626333333333333]
n_clusters: 6 ,Score: 134.86445388129442 ,Average WPP: [0.5762909090909092,
0.5173214285714286, 0.4587063157894737, 0.7632625, 0.5520115384615385,
0.7626333333333333]
n_clusters: 7 ,Score: 124.58156473490062 ,Average WPP: [0.5602400000000001,
0.7632625, 0.5146791666666667, 0.5173214285714286, 0.7626333333333333,
0.7626343750000001, 0.3319636363636364]
n_clusters: 8 ,Score: 117.39879567076434 ,Average WPP: [0.77275,
0.5298086956521739, 0.341231884057971, 0.6026272727272728, 0.8055499999999999,
0.7868607142857142, 0.5762909090909092, 0.5296181818181819]

n_clusters: 9 ,Score: 110.28045662528714 ,Average WPP: [0.5940391304347826, 0.8474142857142857, 0.3399285714285714, 0.8055499999999999, 0.4810625, 0.7745068965517242, 0.34565, 0.6364, 0.59106]
n_clusters: 10 ,Score: 103.65354027840965 ,Average WPP: [0.6127772727272728, 0.8474142857142857, 0.31514590163934425, 0.8055499999999999, 0.4810625, 0.7647882352941175, 0.34565, 0.6364, 0.5369, 0.6500538461538461]
best_n_clusters: 2 , best_score: 251.4727882572557 , best_average_Wpp: [0.4949563909774436, 0.6217233333333334]

normal_duo_fpp:
n_clusters: 2 ,Score: 2549.4477131316194 ,Average WPP: [0.4919794777940459, 0.6929175757575757]
n_clusters: 3 ,Score: 2202.168392457188 ,Average WPP: [0.42874268712809266, 0.7116732510288066, 0.7085511088709677]
n_clusters: 4 ,Score: 1977.6607582743318 ,Average WPP: [0.43725434580884626, 0.8010793388429751, 0.698198086124402, 0.6632004474272931]
n_clusters: 5 ,Score: 1823.1784210514932 ,Average WPP: [0.3630837334437086, 0.8349933333333333, 0.6902010135135135, 0.6786569892473119, 0.6402625368731564]
n_clusters: 6 ,Score: 1705.8123639112673 ,Average WPP: [0.5545291203235592, 0.6809651408450703, 0.8513754385964913, 0.3619172461752434, 0.7421685962373373, 0.6437956]
n_clusters: 7 ,Score: 1587.3814844067988 ,Average WPP: [0.7362563545150501, 0.3608364370546318, 0.6856982456140351, 0.8513754385964913, 0.6633933789954338, 0.5484646680942183, 0.646473640167364]
n_clusters: 8 ,Score: 1502.5701764393666 ,Average WPP: [0.5382160651920839, 0.7562166666666665, 0.340207852077001, 0.6760649867374006, 0.7223007894736841, 0.6182473498233215, 0.8641758620689657, 0.6677684210526317]
n_clusters: 9 ,Score: 1417.9832623840084 ,Average WPP: [0.2941698412698413, 0.8641758620689657, 0.53671853188929, 0.6523237654320988, 0.8315646892655367, 0.762815238095238, 0.6231807285546417, 0.6677684210526317, 0.5827360465116278]
n_clusters: 10 ,Score: 1354.5045067668498 ,Average WPP: [0.8444956923076924, 0.29980435820895524, 0.8747157894736843, 0.5943236842105264, 0.6819619047619049, 0.6435114457831326, 0.5354259303721489, 0.7487462686567165, 0.629829512195122, 0.7862734693877552]
best_n_clusters: 2 , best_score: 2549.4477131316194 , best_average_Wpp: [0.4919794777940459, 0.6929175757575757]

normal_solo:
n_clusters: 2 ,Score: 250.68510481932898 ,Average WPP: [0.7380166666666665, 0.4846567251461989]
n_clusters: 3 ,Score: 254.6741810182629 ,Average WPP: [0.45544830508474576, 0.623167619047619, 0.8419000000000001]
n_clusters: 4 ,Score: 236.30420272190472 ,Average WPP: [0.4414222222222222, 0.8923733333333334, 0.5492600000000001, 0.698751851851852]
n_clusters: 5 ,Score: 213.18132333677357 ,Average WPP: [0.5453223684210527, 0.814, 0.7124354166666667, 0.4414222222222222, 0.9444333333333333]
n_clusters: 6 ,Score: 206.2476627029193 ,Average WPP: [0.6314619047619048, 0.4414222222222222, 0.556327027027027, 0.8245384615384614, 0.7373181818181818,

```
0.9444333333333333]
n_clusters:  7 ,Score:  188.4903627526518 ,Average WPP:  [0.5121803571428573,
0.6314619047619048, 0.4300878048780488, 0.8245384615384614, 0.7534,
0.6075463414634147, 0.9444333333333333]
n_clusters:  8 ,Score:  175.83485721103057 ,Average WPP:  [0.9444333333333333,
0.5567219512195122, 0.4300878048780488, 0.8793583333333332, 0.63917,
0.5170583333333334, 0.6438315789473684, 0.7532785714285714]
n_clusters:  9 ,Score:  165.53181452478393 ,Average WPP:  [0.4300878048780488,
0.56427, 0.5108239130434783, 0.9444333333333333, 0.63917, 0.8793583333333332,
0.5688681818181819, 0.5858857142857142, 0.8452636363636364]
n_clusters:  10 ,Score:  154.68702495899012 ,Average WPP:  [0.4521322580645161,
0.56427, 0.528348717948718, 0.9444333333333333, 0.63917, 0.8793583333333332,
0.5813142857142858, 0.5858857142857142, 0.8452636363636364, 0.4114305555555556]
best_n_clusters:  3 , best_score:  254.6741810182629 , best_average_Wpp:
[0.45544830508474576, 0.623167619047619, 0.8419000000000001]

normal_solo_fpp:
n_clusters:  2 ,Score:  1577.2555836503602 ,Average WPP:  [0.7214859416445624,
0.48326100104275294]
n_clusters:  3 ,Score:  1621.1583309253517 ,Average WPP:  [0.61975625,
0.4774075757575758, 0.8301031249999999]
n_clusters:  4 ,Score:  1459.7878353661536 ,Average WPP:  [0.48259780219780213,
0.7993055865921787, 0.5343516539440204, 0.8622333333333332]
n_clusters:  5 ,Score:  1342.9978678720863 ,Average WPP:  [0.4837189892802451,
0.695875, 0.9481416666666665, 0.49094070796460176, 0.8179420454545455]
n_clusters:  6 ,Score:  1224.5291946311509 ,Average WPP:  [0.6888385650224216,
0.4872814685314686, 0.8204538461538463, 0.5385348314606742, 0.9481416666666665,
0.4858352755905512]
n_clusters:  7 ,Score:  1159.1332450196337 ,Average WPP:  [0.5576772093023256,
0.47222782608695646, 0.8094032258064516, 0.5706000000000001, 0.8021333333333334,
0.9481416666666665, 0.4984942528735632]
n_clusters:  8 ,Score:  1110.691285537699 ,Average WPP:  [0.5619419811320755,
0.4715980701754387, 0.8292473684210527, 0.9481416666666665, 0.7848042553191489,
0.5018602362204724, 0.809102, 0.535724358974359]
n_clusters:  9 ,Score:  1039.6483447077583 ,Average WPP:  [0.4553810606060606,
0.6451337579617834, 0.8052269230769231, 0.50922, 0.9481416666666665,
0.5838411764705882, 0.5258, 0.8198204301075268, 0.8777900000000001]
n_clusters:  10 ,Score:  982.1509938881 ,Average WPP:  [0.5026148936170213,
0.8279656565656565, 0.44012590361445786, 1.0, 0.8271666666666667,
0.6266503105590062, 0.5454191780821919, 0.9308555555555557, 0.8051914893617022,
0.5624483333333334]
best_n_clusters:  3 , best_score:  1621.1583309253517 , best_average_Wpp:
[0.61975625, 0.4774075757575758, 0.8301031249999999]

normal_squad:
n_clusters:  2 ,Score:  337.59327000586666 ,Average WPP:  [0.4952034188034188,
0.7550170212765956]
n_clusters:  3 ,Score:  279.61078721982955 ,Average WPP:  [0.6267217391304348,
```

0.8151294117647058, 0.47677024221453285]
n_clusters: 4 ,Score: 289.03003961857365 ,Average WPP: [0.6236989583333333,
0.764065625, 0.46088104089219334, 1.0]
n_clusters: 5 ,Score: 270.7084542643844 ,Average WPP: [0.46151570881226056,
0.8192307692307691, 0.5968385416666667, 0.7370407407407407, 1.0]
n_clusters: 6 ,Score: 250.82108870037374 ,Average WPP: [0.46329076923076923,
0.85715, 0.7006217391304348, 1.0, 0.5888357894736842, 0.8192307692307691]
n_clusters: 7 ,Score: 236.16504441312136 ,Average WPP: [0.4612185328185328,
0.8192307692307691, 1.0, 0.7037304347826089, 0.5978886792452829,
0.5912977777777778, 0.9285749999999999]
n_clusters: 8 ,Score: 230.96942129771995 ,Average WPP: [0.7037304347826089,
0.43265844155844163, 1.0, 0.5916666666666667, 0.7863636363636364,
0.5932041666666666, 0.9285749999999999, 0.6892487804878049]
n_clusters: 9 ,Score: 220.97218174848487 ,Average WPP: [0.5594096774193548,
0.9285749999999999, 0.4222838150289018, 0.6410567567567567, 1.0,
0.7037304347826089, 0.7863636363636364, 0.738025, 0.46301538461538466]
n_clusters: 10 ,Score: 214.82220273699306 ,Average WPP: [0.7037304347826089,
0.5734819999999999, 0.5558333333333334, 0.7863636363636364, 0.9285749999999999,
0.6410567567567567, 1.0, 0.6688806451612904, 0.38763053435114503,
0.48961249999999995]
best_n_clusters: 2 , best_score: 337.59327000586666 , best_average_Wpp:
[0.4952034188034188, 0.7550170212765956]

normal_squad_fpp:
n_clusters: 2 ,Score: 7865.493244915583 ,Average WPP: [0.6327917441860464,
0.46615331751472017]
n_clusters: 3 ,Score: 5941.070749522682 ,Average WPP: [0.45107986928104576,
0.6892038681948424, 0.594061429433052]
n_clusters: 4 ,Score: 5650.963085351038 ,Average WPP: [0.5833690533562822,
0.6156822061482821, 0.40525, 0.687302496099844]
n_clusters: 5 ,Score: 5104.8050549145855 ,Average WPP: [0.4005353191489362,
0.7072305172413794, 0.6511845454545454, 0.5598373569198752, 0.6150714768883878]
n_clusters: 6 ,Score: 4755.560841816579 ,Average WPP: [0.3575282550077042,
0.6217423324742267, 0.716465965583174, 0.5644229868228404, 0.5422414838337183,
0.6465538306451613]
n_clusters: 7 ,Score: 4538.393214445618 ,Average WPP: [0.3511057102502018,
0.6172992385786802, 0.6948864285714286, 0.751806818181818, 0.5397583926453143,
0.5517434375000001, 0.6121120663650075]
n_clusters: 8 ,Score: 4311.395312905485 ,Average WPP: [0.3432741489361702,
0.7785623376623376, 0.6075121739130435, 0.5339752389762565, 0.6181805068226122,
0.6420223711340206, 0.6870470588235295, 0.5472308366533865]
n_clusters: 9 ,Score: 4101.956548774572 ,Average WPP: [0.46354649606299214,
0.5073592680608365, 0.605901166180758, 0.3448798784458433, 0.7119523519645822,
0.6870470588235295, 0.5874498181818182, 0.7960717741935486, 0.6575605011933173]
n_clusters: 10 ,Score: 3959.5675223995445 ,Average WPP: [0.46143287509819325,
0.6871020408163265, 0.5029376648754273, 0.6036512643678161, 0.34196702269692925,
0.6725928961748634, 0.7223098214285715, 0.5801134532990574, 0.7978554621848739,
0.598804054054054]

best_n_clusters: 2 , best_score: 7865.493244915583 , best_average_Wpp:
[0.6327917441860464, 0.46615331751472017]

solo:
n_clusters: 2 ,Score: 73255.26266074822 ,Average WPP: [0.3915982966058232,
0.8290304240594861]
n_clusters: 3 ,Score: 56542.944216137774 ,Average WPP: [0.8216163342530548,
0.37720367241273756, 0.8011652908067541]
n_clusters: 4 ,Score: 51100.76747758277 ,Average WPP: [0.32938070824256555,
0.8985481653506734, 0.8148120017913121, 0.7343665996737738]
n_clusters: 5 ,Score: 48117.029295678825 ,Average WPP: [0.3231960466859629,
0.9080288852278967, 0.8215319580078124, 0.7492975414522584, 0.7190612180129619]
n_clusters: 6 ,Score: 45000.46300550175 ,Average WPP: [0.6822157436989706,
0.31238679016830123, 0.7502659761150378, 0.9111155699591377, 0.8442344884488449,
0.7367804515805318]
n_clusters: 7 ,Score: 41873.54781587222 ,Average WPP: [0.7265378387572287,
0.4862610905440171, 0.8133979182056277, 0.30252838128973414, 0.9187424670258222,
0.747630346930995, 0.8455465838509316]
n_clusters: 8 ,Score: 39406.587275698075 ,Average WPP: [0.838729605387972,
0.3005225303854218, 0.8465505376344087, 0.7227392184392185, 0.9166038102216524,
0.6201664072272411, 0.34362114205694727, 0.7478419070196233]
n_clusters: 9 ,Score: 37426.15616644893 ,Average WPP: [0.49543494846537095,
0.9442344176285413, 0.32214569263995063, 0.8448842755985163, 0.7501297200386153,
0.27866285965882326, 0.7764422748963467, 0.7300131560449858, 0.8451718328840969]
n_clusters: 10 ,Score: 35892.21525306852 ,Average WPP: [0.32253274436090223,
0.8531909268854346, 0.6703655181150241, 0.4998867621094649, 0.7489217010022273,
0.9474580231065469, 0.7877449438202246, 0.8366016415868672, 0.2790695211579012,
0.8418366638441999]
best_n_clusters: 2 , best_score: 73255.26266074822 , best_average_Wpp:
[0.3915982966058232, 0.8290304240594861]

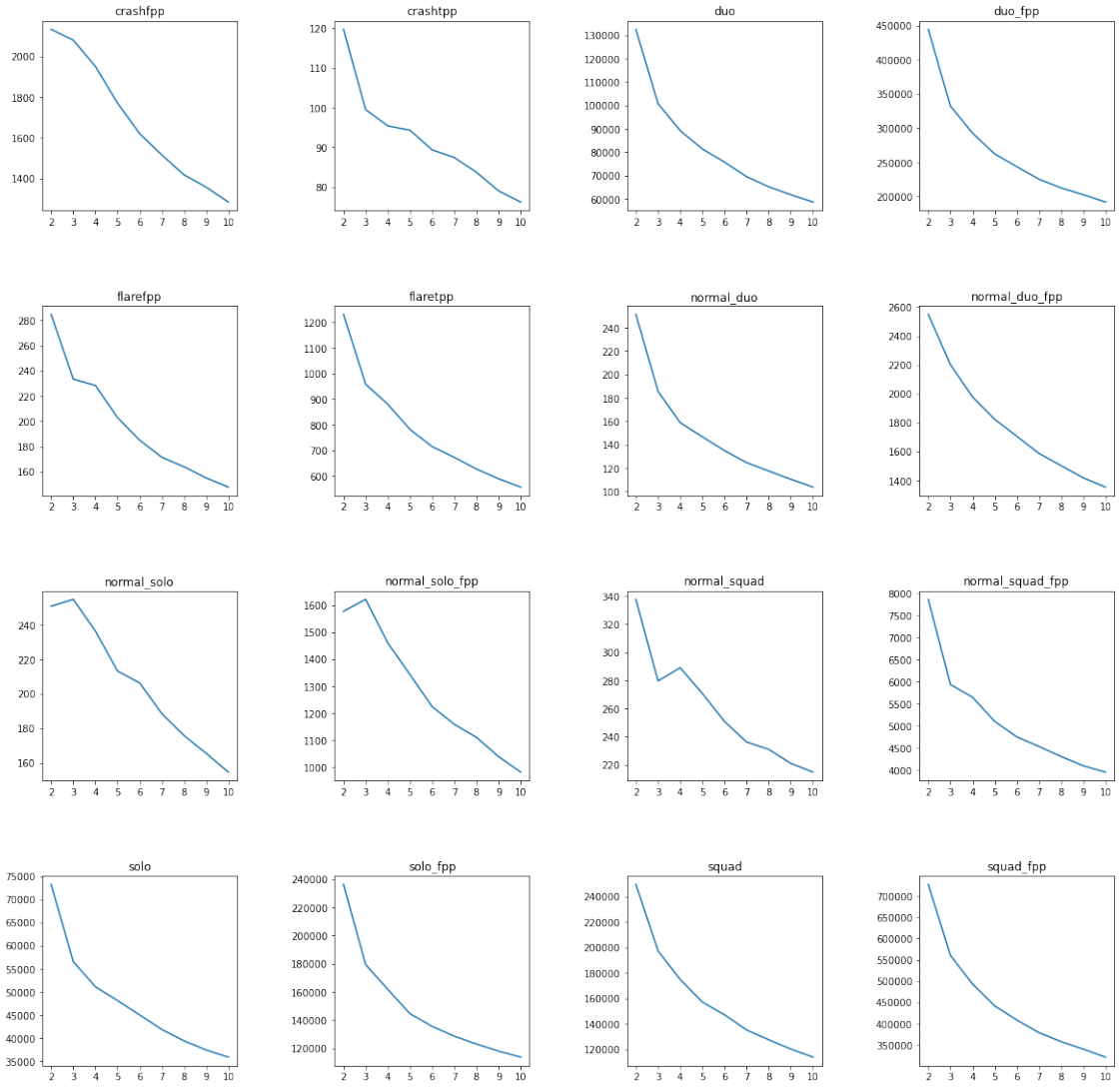
solo_fpp:
n_clusters: 2 ,Score: 236328.74199192724 ,Average WPP: [0.8327608083366773,
0.4141886355972406]
n_clusters: 3 ,Score: 179422.20965185255 ,Average WPP: [0.39801950144826825,
0.8165937195920082, 0.8312136176807492]
n_clusters: 4 ,Score: 161819.24615592512 ,Average WPP: [0.3539329178251949,
0.9046252702863253, 0.8241499879701659, 0.7342198596308811]
n_clusters: 5 ,Score: 144587.57961357656 ,Average WPP: [0.3417086578523106,
0.7526503215810091, 0.6917263162727115, 0.9099549251345895, 0.8613660180817609]
n_clusters: 6 ,Score: 135623.88841932028 ,Average WPP: [0.6818657671032518,
0.33860224085831664, 0.8650816192560175, 0.9125601237569061, 0.7419930091904311,
0.7720175173394985]
n_clusters: 7 ,Score: 128681.12953311906 ,Average WPP: [0.4946216045675942,
0.3292138064360411, 0.922365971796444, 0.867980636202979, 0.7663754207485557,
0.825438194782388, 0.7280977722772277]
n_clusters: 8 ,Score: 123085.57401705543 ,Average WPP: [0.6227609473730723,
0.9167366069403493, 0.7650171319542077, 0.7245905239106906, 0.8682273072060682,

0.35311143215469715, 0.8565954687118064, 0.3375634860213095]
n_clusters: 9 ,Score: 118046.00015704651 ,Average WPP: [0.31600318296174856,
0.7326325430947884, 0.8290021376206721, 0.9440864700780572, 0.7674710561925788,
0.8139433848773655, 0.5065972994686162, 0.8662079548660084, 0.33227070636748246]
n_clusters: 10 ,Score: 113808.06921807323 ,Average WPP: [0.8417295781312562,
0.332424941498285, 0.5094941843739402, 0.8657845256215119, 0.6617073582367998,
0.31527137235276714, 0.8218984015594543, 0.7667859181185918, 0.8526984838619664,
0.9487535999018767]
best_n_clusters: 2 , best_score: 236328.74199192724 , best_average_Wpp:
[0.8327608083366773, 0.4141886355972406]

squad:
n_clusters: 2 ,Score: 249477.65870643815 ,Average WPP: [0.7678304773315529,
0.3644227054522414]
n_clusters: 3 ,Score: 197312.6021848475 ,Average WPP: [0.7594287495606944,
0.3569825073071379, 0.729434045070204]
n_clusters: 4 ,Score: 174901.913886372 ,Average WPP: [0.6935375023372031,
0.7878756561300163, 0.7485325723334765, 0.29337733443471914]
n_clusters: 5 ,Score: 157197.5922377535 ,Average WPP: [0.7451307913669065,
0.2871322846419294, 0.6459597761691499, 0.8357332800312013, 0.6967333827929265]
n_clusters: 6 ,Score: 147204.84619383144 ,Average WPP: [0.2919842054318137,
0.6986033300949914, 0.6889612075198842, 0.7689782820573039, 0.5535873541325103,
0.8489274186908192]
n_clusters: 7 ,Score: 135288.50764650488 ,Average WPP: [0.26070923559773873,
0.6768094137313433, 0.7080960984930198, 0.5112986346757155, 0.8607405763662684,
0.6982322728944464, 0.7793977512583212]
n_clusters: 8 ,Score: 127626.85462919934 ,Average WPP: [0.7825208106169298,
0.2511680914664244, 0.6514376091621339, 0.6955642467073563, 0.8864551798783631,
0.775909932802966, 0.710333708250351, 0.3507446280341919]
n_clusters: 9 ,Score: 120251.23781644605 ,Average WPP: [0.2724582618818834,
0.6908771595330738, 0.6422599101334936, 0.8820454524033932, 0.8253212434571684,
0.7817451772464963, 0.4545211035975115, 0.7258797614106012, 0.2635953299116555]
n_clusters: 10 ,Score: 114018.09180322764 ,Average WPP: [0.8898353703251038,
0.26386182255004653, 0.6848855813652959, 0.7200013233584691, 0.7695428777949114,
0.741718090251333, 0.843114302812687, 0.27295794338227525, 0.6179120001162519,
0.34972921542061824]
best_n_clusters: 2 , best_score: 249477.65870643815 , best_average_Wpp:
[0.7678304773315529, 0.3644227054522414]

squad_fpp:
n_clusters: 2 ,Score: 727270.1507302539 ,Average WPP: [0.778118222574964,
0.37731190460366676]
n_clusters: 3 ,Score: 560490.8386678381 ,Average WPP: [0.7600856147707289,
0.36663556949153187, 0.7413622804779201]
n_clusters: 4 ,Score: 492859.0712697686 ,Average WPP: [0.7098422928682212,
0.30675990790832725, 0.7676968643870065, 0.7875423801844917]
n_clusters: 5 ,Score: 441583.2323776081 ,Average WPP: [0.3062771317659251,
0.7244587319090894, 0.8480141231129372, 0.7733399801336767, 0.5353253842200165]

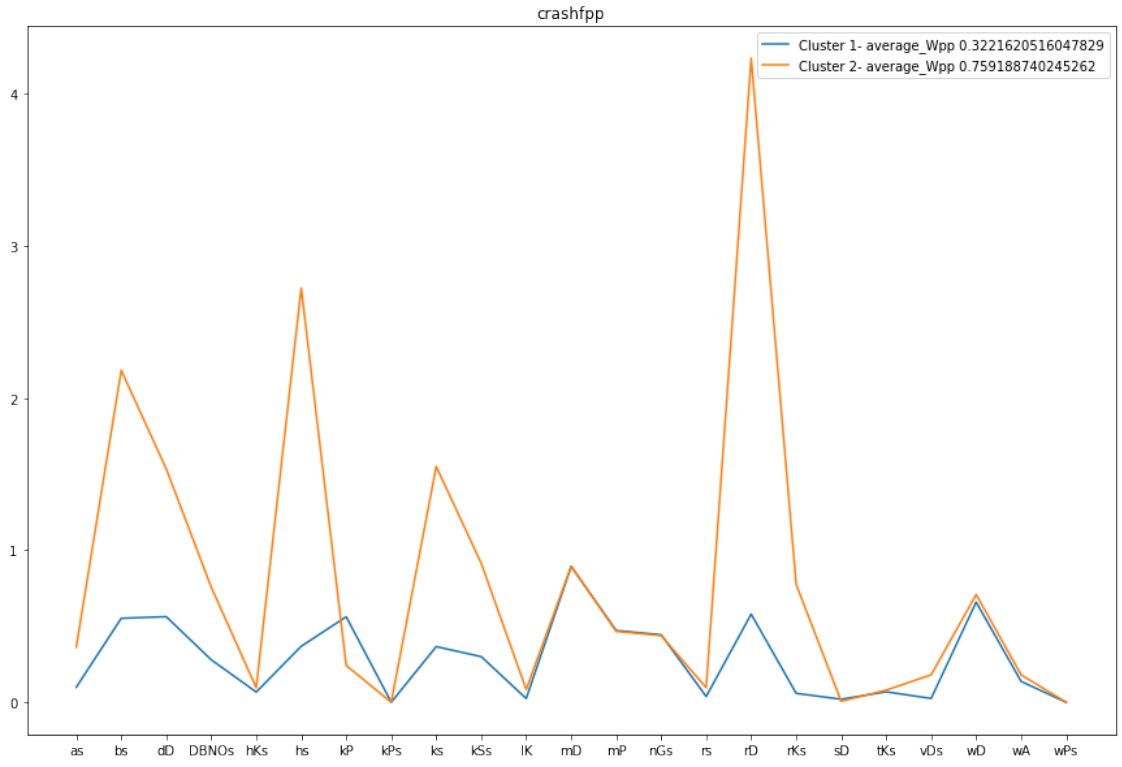
```
n_clusters:  6 ,Score:  408341.21718856797 ,Average WPP:  [0.7154403874202919,
0.5440590114204884, 0.8490014173810981, 0.7835183393425239, 0.7076320496924174,
0.3049042778503038]
n_clusters:  7 ,Score:  378715.51439274254 ,Average WPP:  [0.7633968754376137,
0.8568565547856787, 0.28025004672765996, 0.7990944068784611, 0.7127989337985847,
0.47834838100466653, 0.6715689878718842]
n_clusters:  8 ,Score:  357521.0496557742 ,Average WPP:  [0.3521722564245447,
0.7457121932902286, 0.8822952253189508, 0.7665696501163352, 0.666453151266051,
0.2666249119780968, 0.7121953338522748, 0.7974102116687654]
n_clusters:  9 ,Score:  339993.44104838534 ,Average WPP:  [0.28596515823929286,
0.4396026177654818, 0.8806494646680942, 0.6531555321846096, 0.7988641657430301,
0.8253734915092182, 0.27602087980129375, 0.7557239264538164, 0.7106491370438975]
n_clusters:  10 ,Score:  321273.51591303985 ,Average WPP:  [0.42725706961383364,
0.28623356575520637, 0.7894756300827901, 0.8120954754800545, 0.7069920324820524,
0.6040467308879286, 0.80044128146134, 0.7675455369383607, 0.27636914895401693,
0.8840944423203921]
best_n_clusters:  2 , best_score:  727270.1507302539 , best_average_Wpp:
[0.778118222574964, 0.37731190460366676]
```

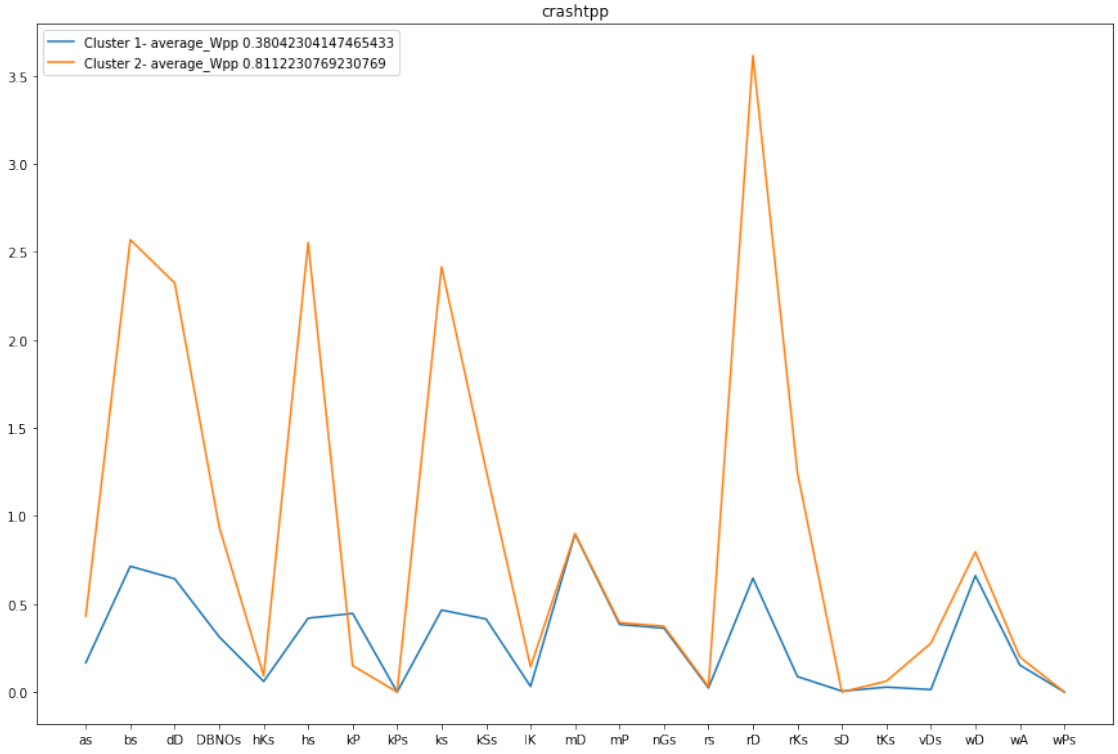


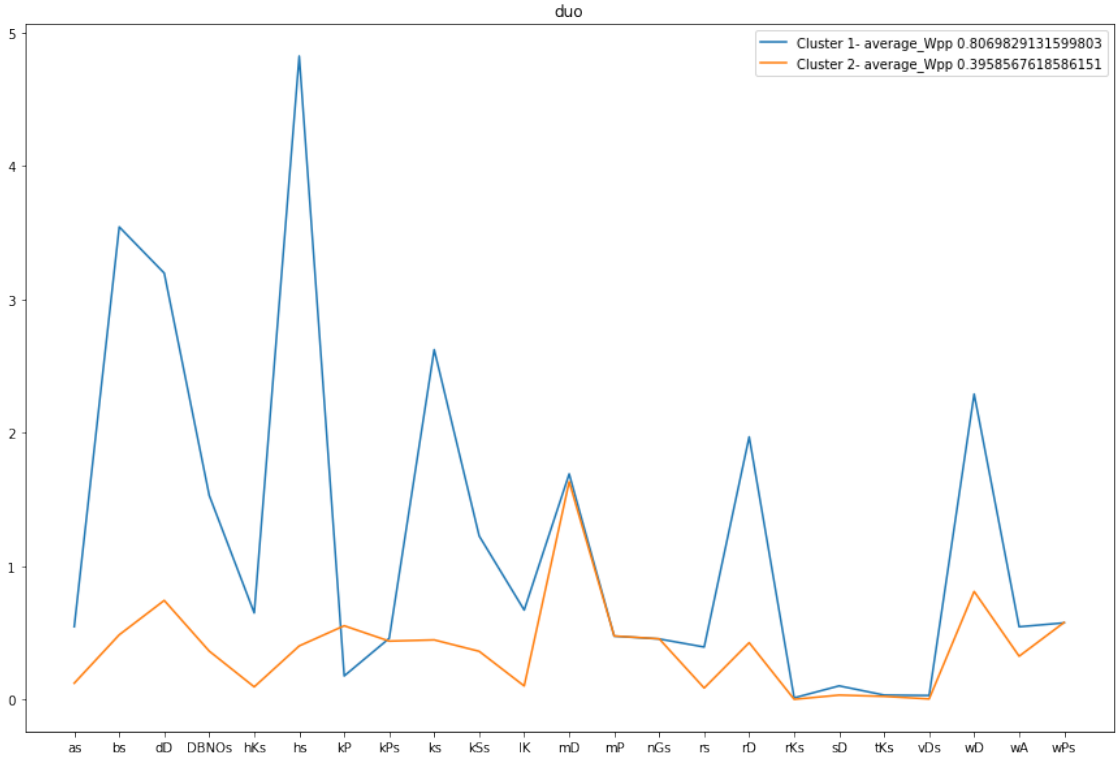
[15]:
```
types_with_two =␣
↪[matchType_crashfpp,matchType_crashtpp,matchType_duo,matchType_duo_fpp,matchType_flarefpp,m

names_with_two =␣
↪["crashfpp","crashtpp","duo","duo_fpp","flarefpp","flaretpp","normal_duo","normal_duo_fpp",
```

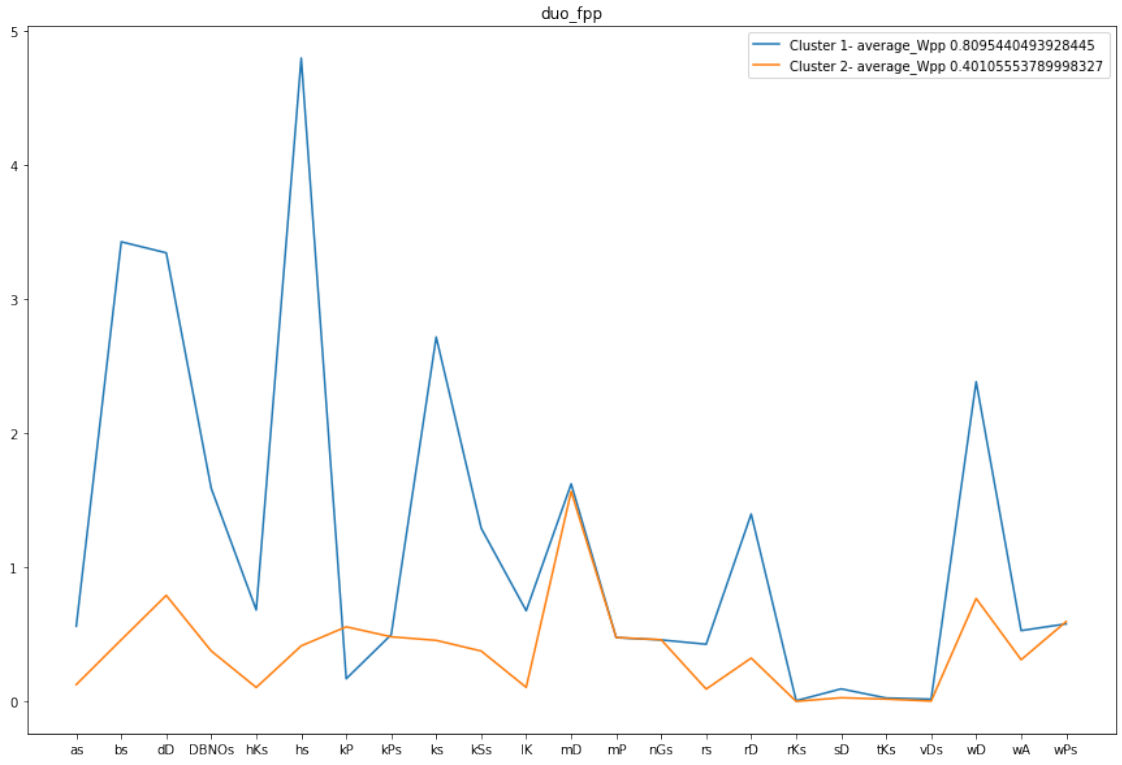
[40]:
```
for i in range(len(types_with_two)):
    #Get the name
    name = names_with_two[i]

    #Get the dataframe with specific type, without changing the original data
    single_type = types_with_two[i].copy()
```

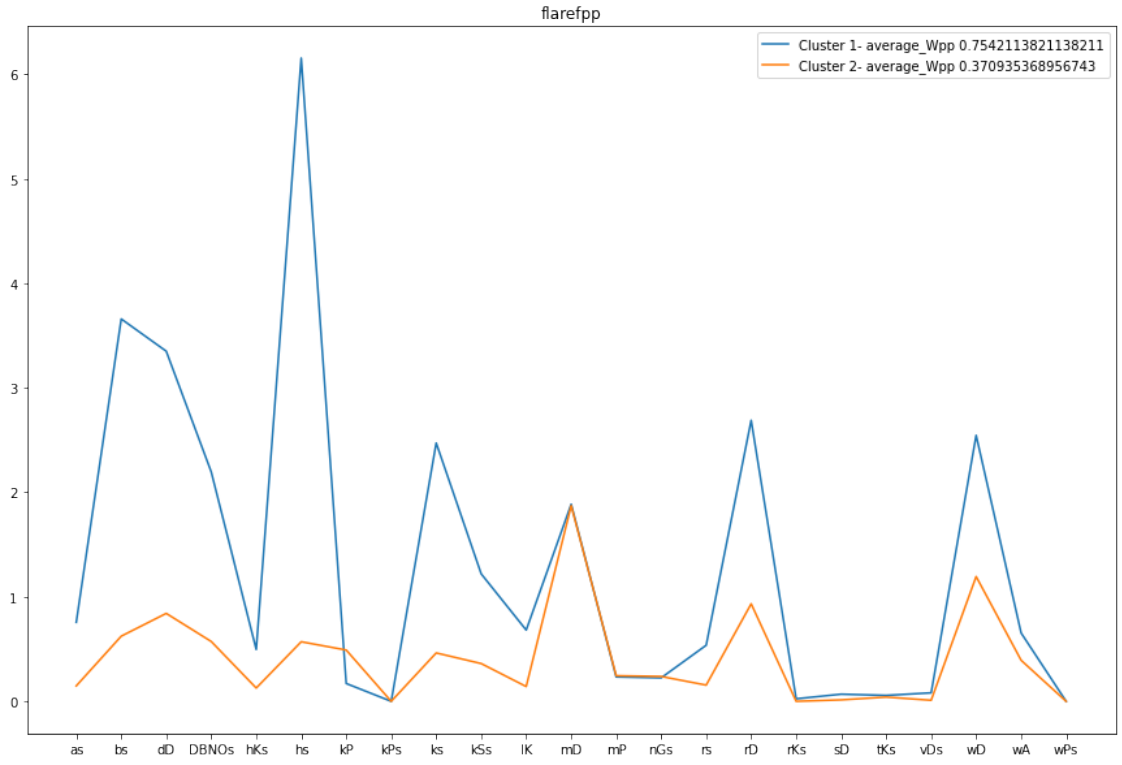
```
    #Extract X and y
    single_y = single_type["winPlacePerc"]
    single_X = single_type.
↪drop(['Id','groupId','matchId',"winPlacePerc"],axis=1)

    #Delete matchType column
    single_X_temp = single_X.drop(["matchType"],axis=1)

    #Convert numpy array to pandas dataframe
    single_y_temp = pd.DataFrame({"winPlacePerc": single_y})

    distinguish_clusters(kmeans_mt,single_X_temp,single_y_temp,2,name)
```

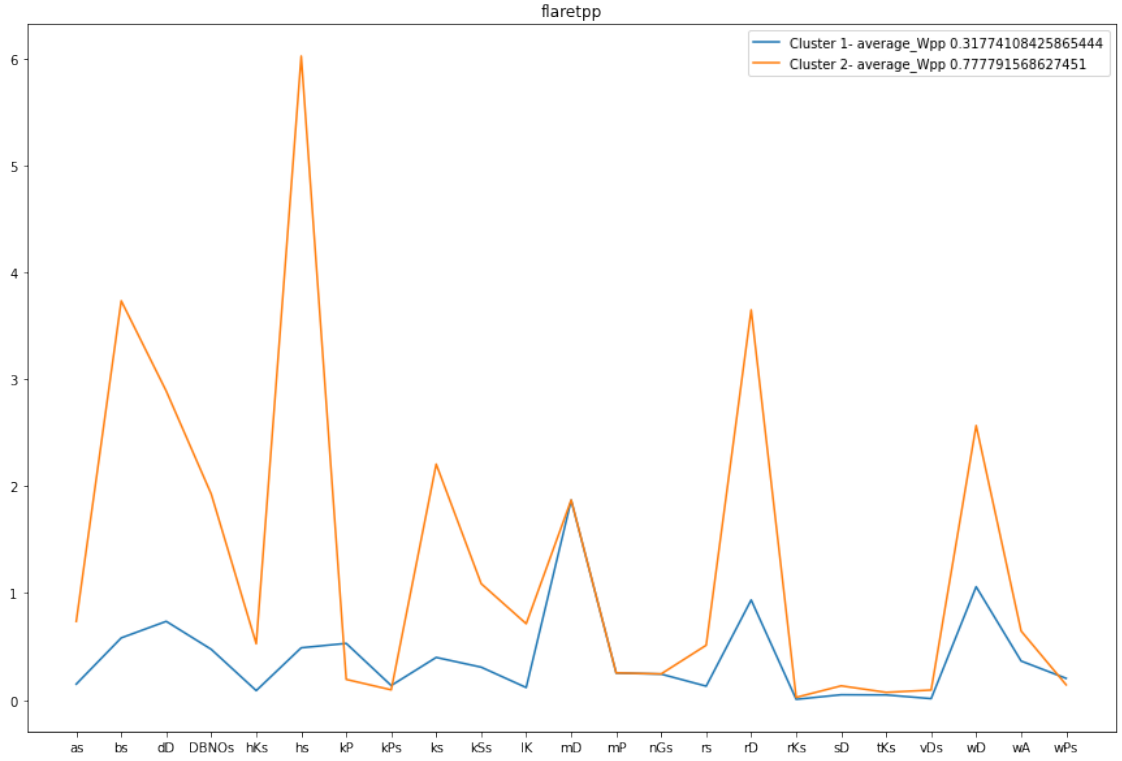


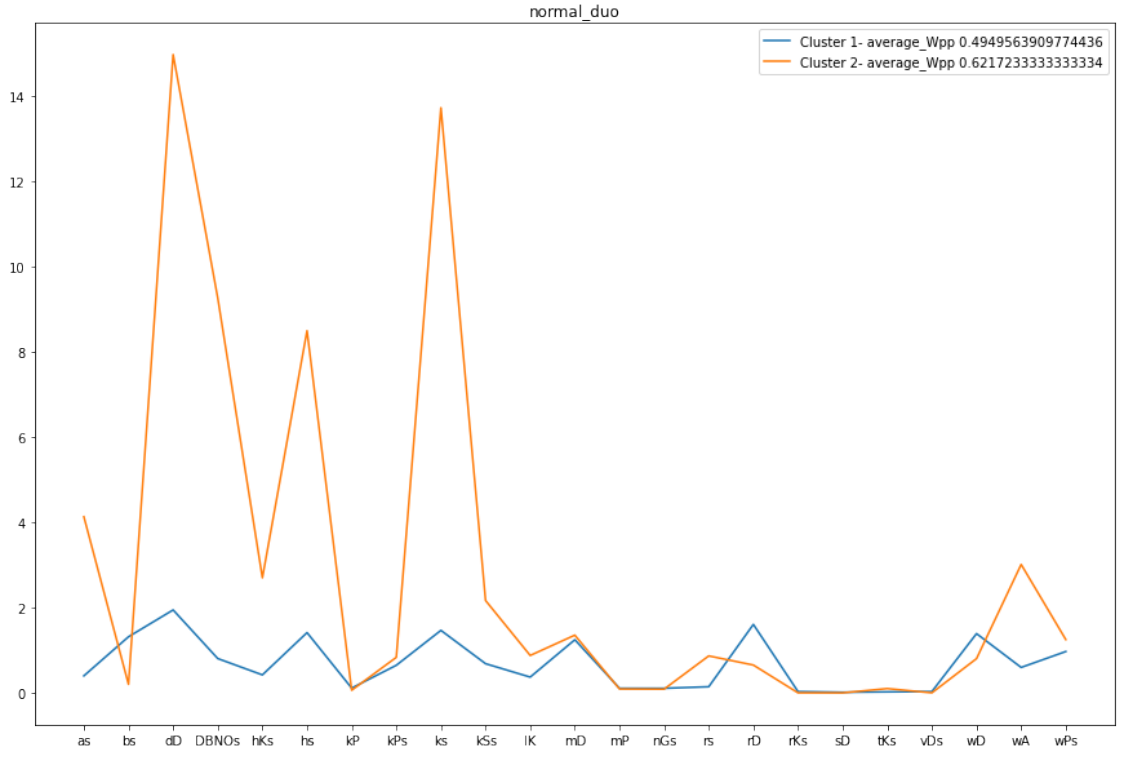

crashtpp
Cluster 1- average_Wpp 0.38042304147465433
Cluster 2- average_Wpp 0.8112230769230769
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
as bs dD DBNOs hKs hs kP kPs ks kSs lK mD mP nGs rs rD rKs sD tKs vDs wD wA wPs

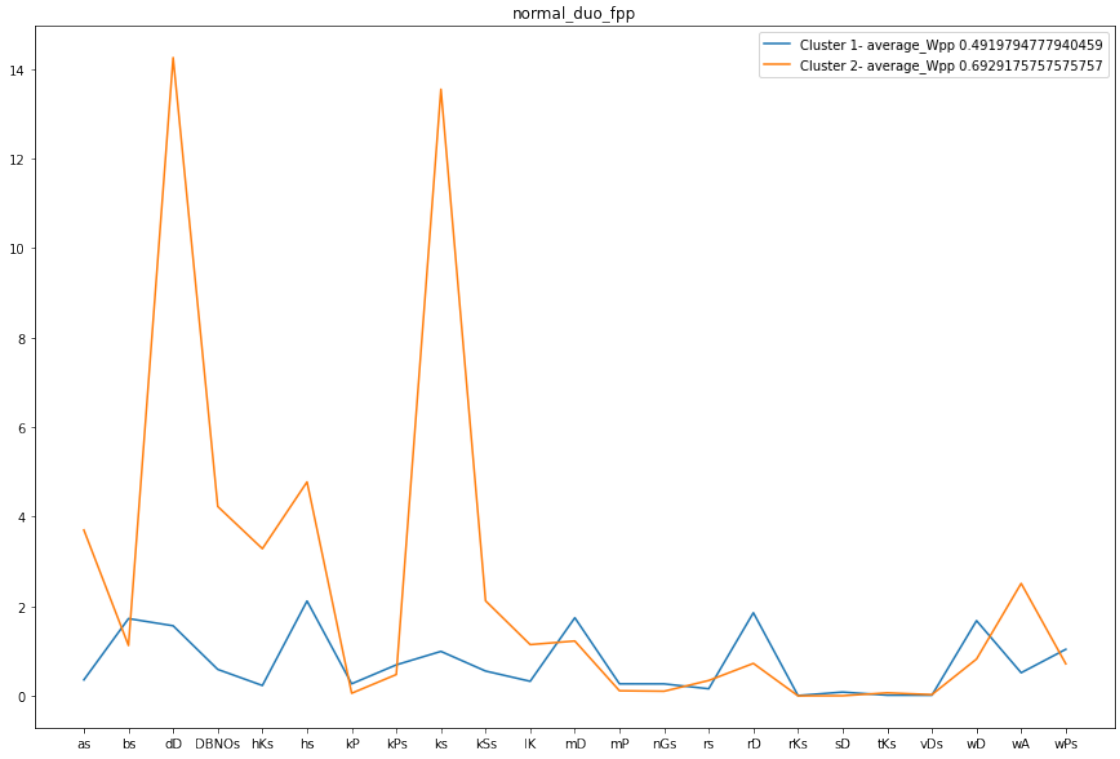

duo
Cluster 1- average_Wpp 0.8069829131599803
Cluster 2- average_Wpp 0.3958567618586151
5
4
3
2
1
0
as bs dD DBNOs hKs hs kP kPs ks kSs lK mD mP nGs rs rD rKs sD tKs vDs wD wA wPs

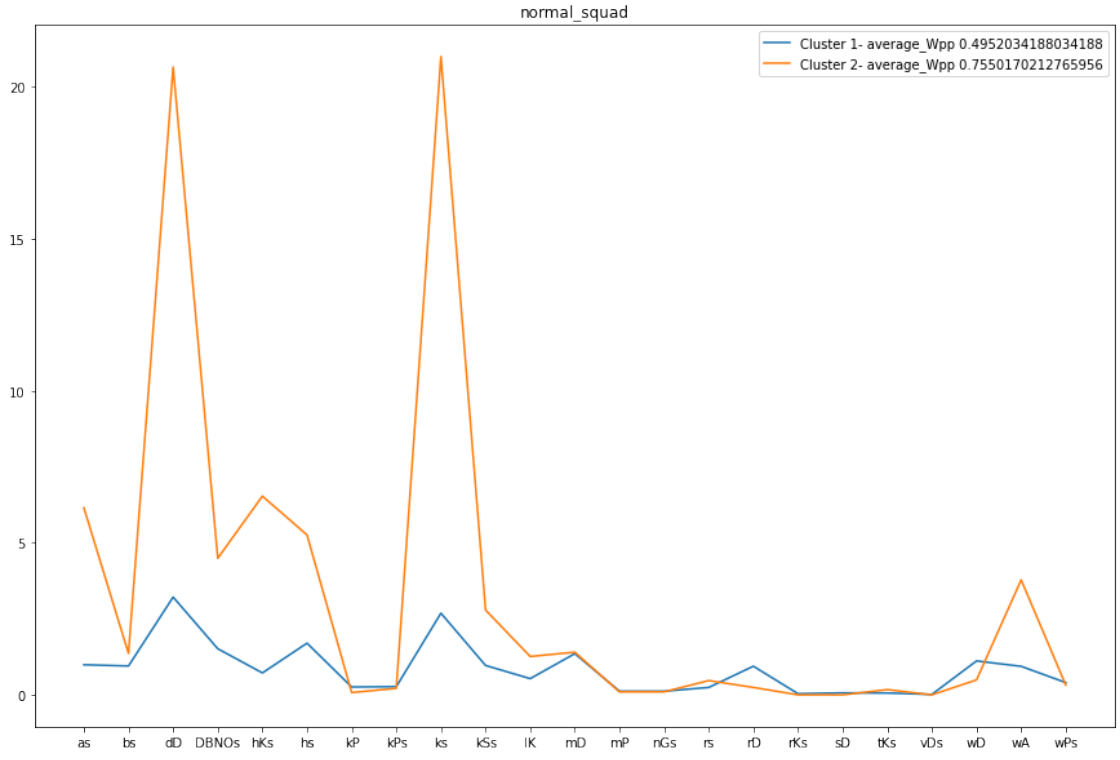

duo_fpp
Cluster 1- average_Wpp 0.8095440493928445
Cluster 2- average_Wpp 0.40105553789998327
0
1
2
3
4
5
as
bs
dD
DBNOs
hKs
hs
kP
kPs
ks
kSs
lK
mD
mP
nGs
rs
rD
rKs
sD
tKs
vDs
wD
wA
wPs

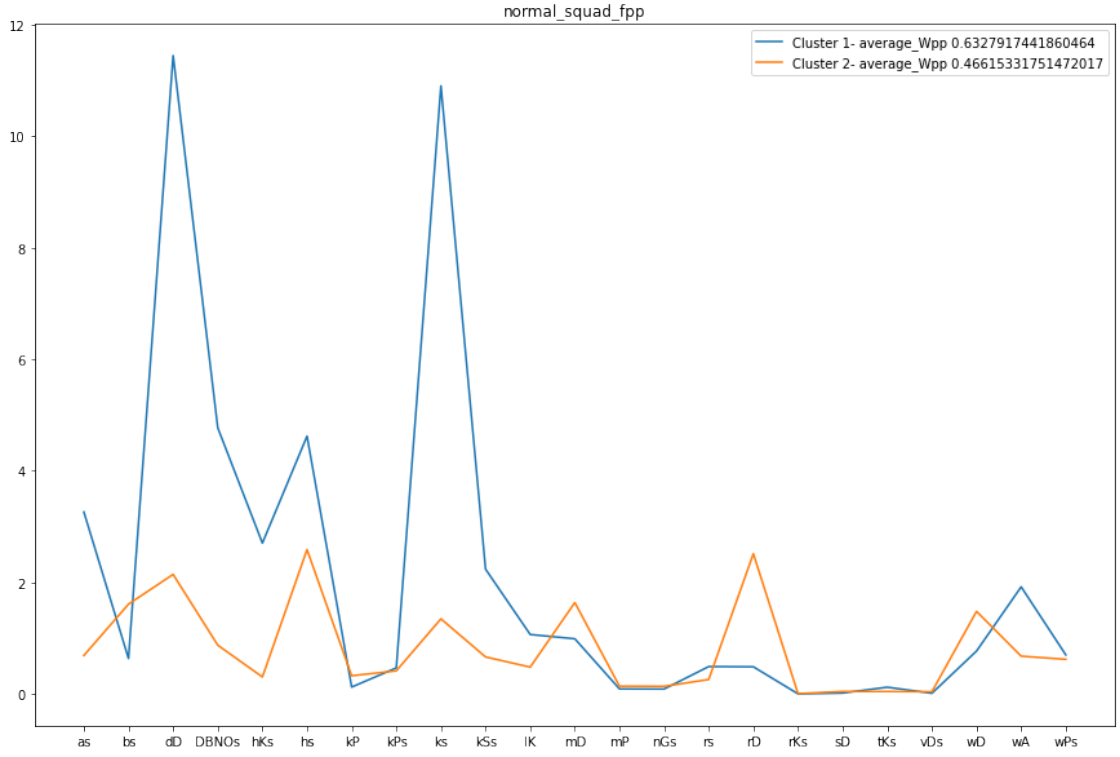

flarefpp
Cluster 1- average_Wpp 0.7542113821138211
Cluster 2- average_Wpp 0.370935368956743
0
1
2
3
4
5
6
as
bs
dD
DBNOs
hKs
hs
kP
kPs
ks
kSs
lK
mD
mP
nGs
rs
rD
rKs
sD
tKs
vDs
wD
wA
wPs

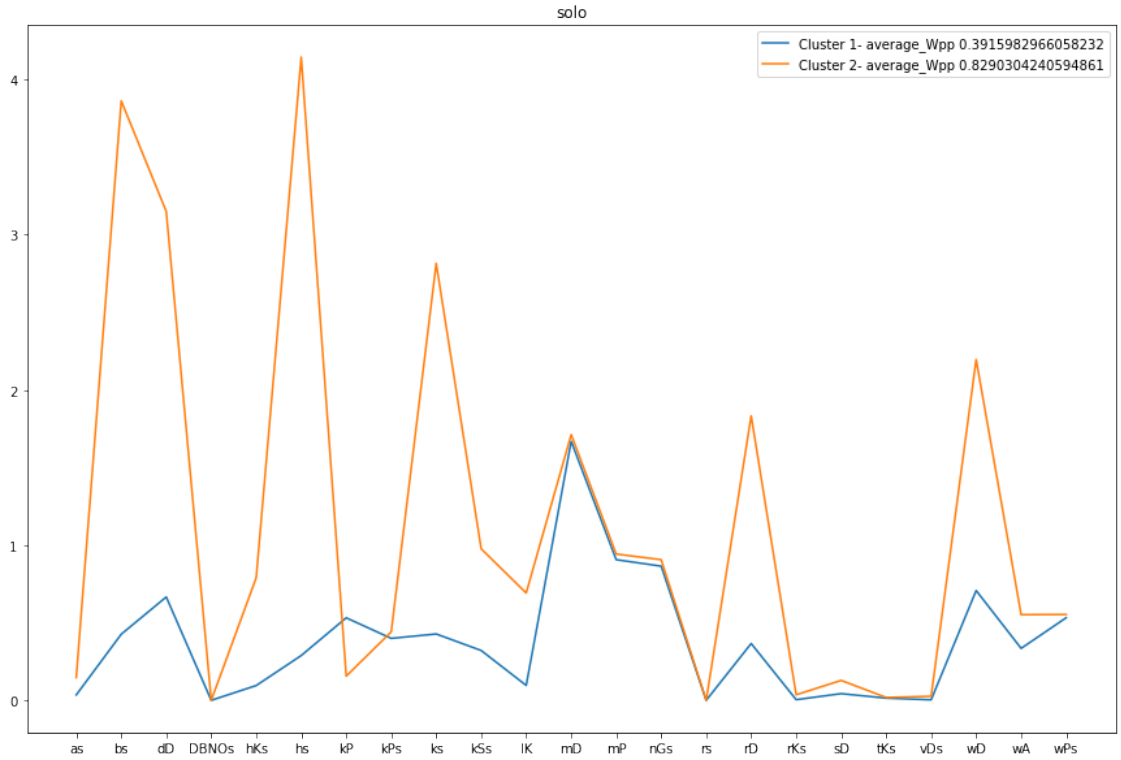

flaretpp
Cluster 1- average_Wpp 0.31774108425865444
Cluster 2- average_Wpp 0.777791568627451
0
1
2
3
4
5
6
as
bs
dD
DBNOs
hKs
hs
kP
kPs
ks
kSs
lK
mD
mP
nGs
rs
rD
rKs
sD
tKs
vDs
wD
wA
wPs

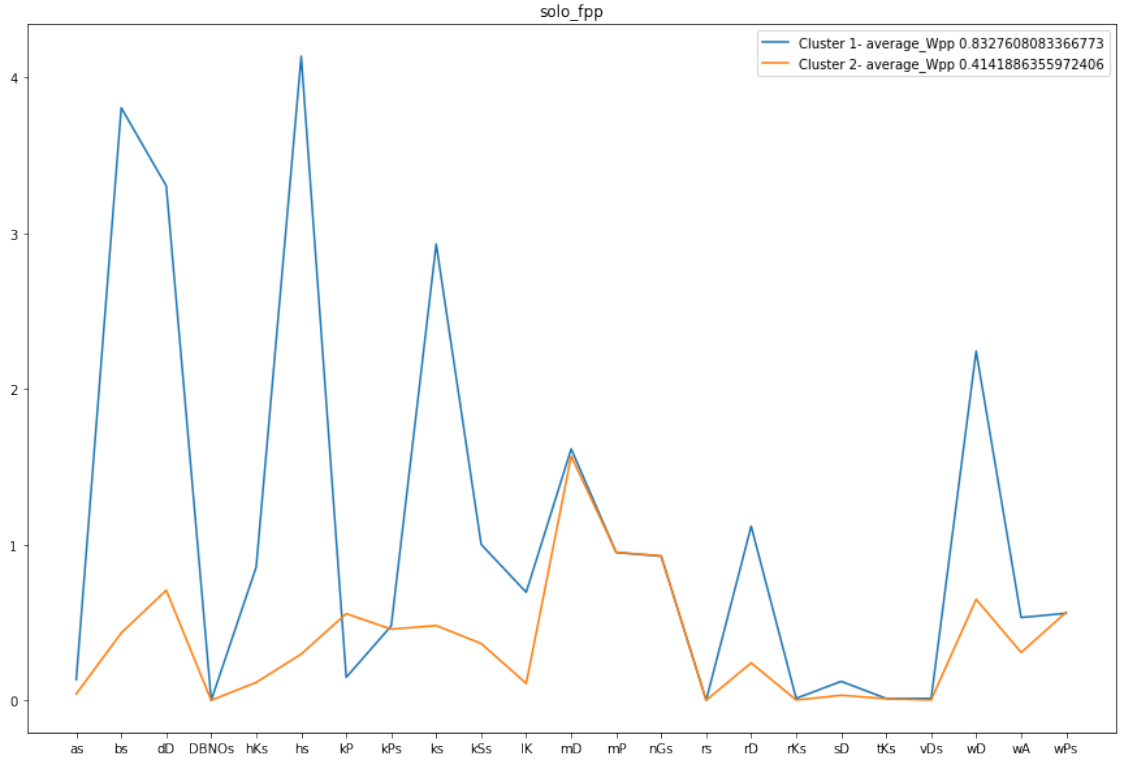

normal_duo
Cluster 1- average_Wpp 0.4949563909774436
Cluster 2- average_Wpp 0.6217233333333334
0
2
4
6
8
10
12
14
as
bs
dD
DBNOs
hKs
hs
kP
kPs
ks
kSs
lK
mD
mP
nGs
rs
rD
rKs
sD
tKs
vDs
wD
wA
wPs

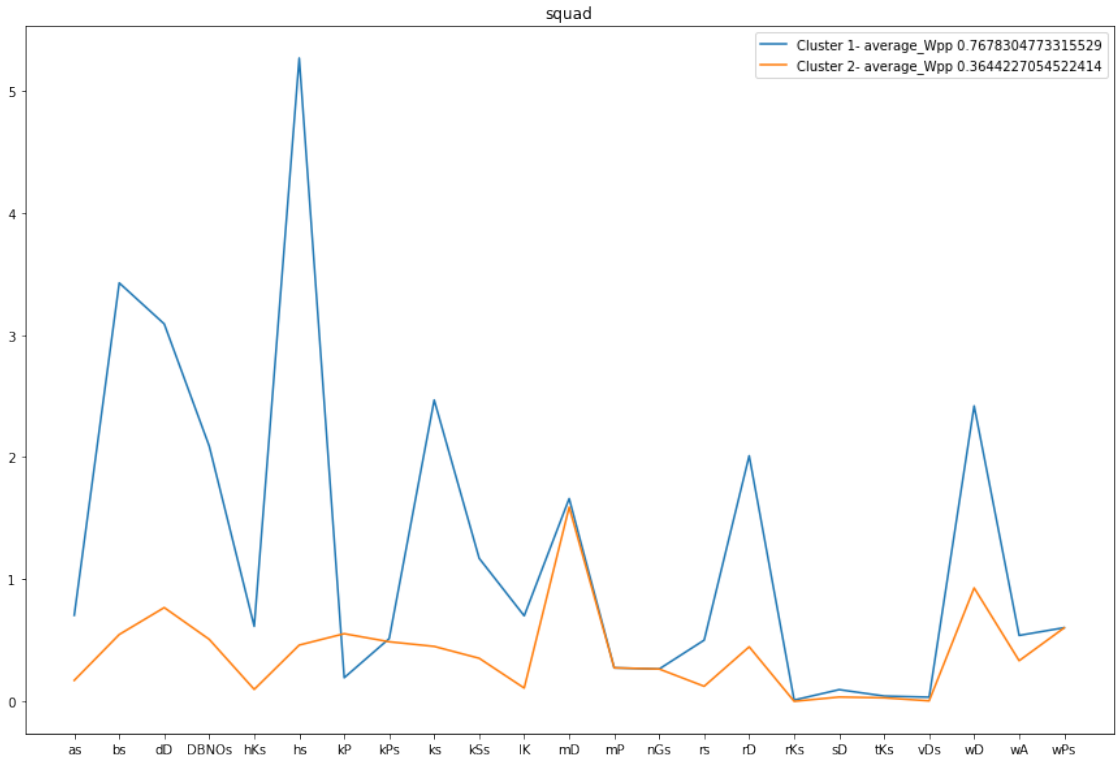

normal_duo_fpp
Cluster 1- average_Wpp 0.4919794777940459
Cluster 2- average_Wpp 0.6929175757575757
0
2
4
6
8
10
12
14
as
bs
dD
DBNOs
hKs
hs
kP
kPs
ks
kSs
lK
mD
mP
nGs
rs
rD
rKs
sD
tKs
vDs
wD
wA
wPs

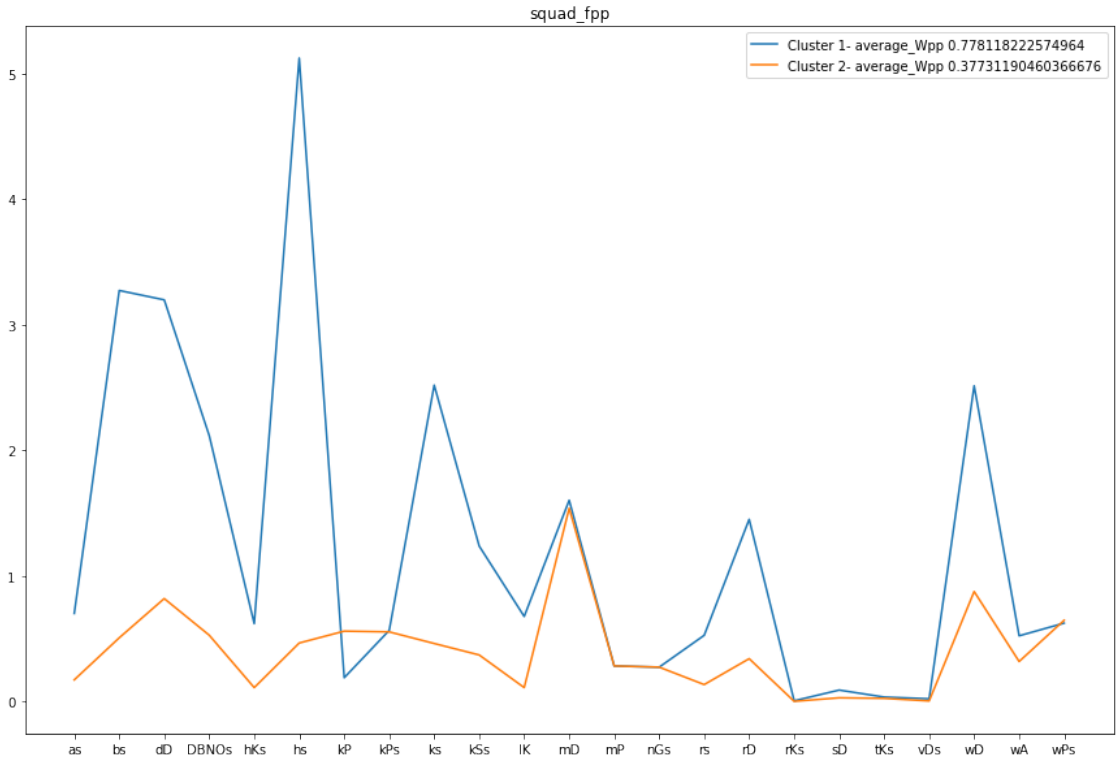

normal_squad
Cluster 1- average_Wpp 0.4952034188034188
Cluster 2- average_Wpp 0.7550170212765956
0
5
10
15
20
as
bs
dD
DBNOs
hKs
hs
kP
kPs
ks
kSs
lK
mD
mP
nGs
rs
rD
rKs
sD
tKs
vDs
wD
wA
wPs


normal_squad_fpp
Cluster 1- average_Wpp 0.6327917441860464
Cluster 2- average_Wpp 0.46615331751472017
12
10
8
6
4
2
0
as
bs
dD
DBNOs
hKs
hs
kP
kPs
ks
kSs
lK
mD
mP
nGs
rs
rD
rKs
sD
tKs
vDs
wD
wA
wPs


solo
Cluster 1- average_Wpp 0.3915982966058232
Cluster 2- average_Wpp 0.8290304240594861
4
3
2
1
0
as
bs
dD
DBNOs
hKs
hs
kP
kPs
ks
kSs
lK
mD
mP
nGs
rs
rD
rKs
sD
tKs
vDs
wD
wA
wPs


solo_fpp
Cluster 1- average_Wpp 0.8327608083366773
Cluster 2- average_Wpp 0.4141886355972406
4
3
2
1
0
as
bs
dD
DBNOs
hKs
hs
kP
kPs
ks
kSs
lK
mD
mP
nGs
rs
rD
rKs
sD
tKs
vDs
wD
wA
wPs


squad
Cluster 1- average_Wpp 0.7678304773315529
Cluster 2- average_Wpp 0.3644227054522414
0
1
2
3
4
5
as
bs
dD
DBNOs
hKs
hs
kP
kPs
ks
kSs
lK
mD
mP
nGs
rs
rD
rKs
sD
tKs
vDs
wD
wA
wPs

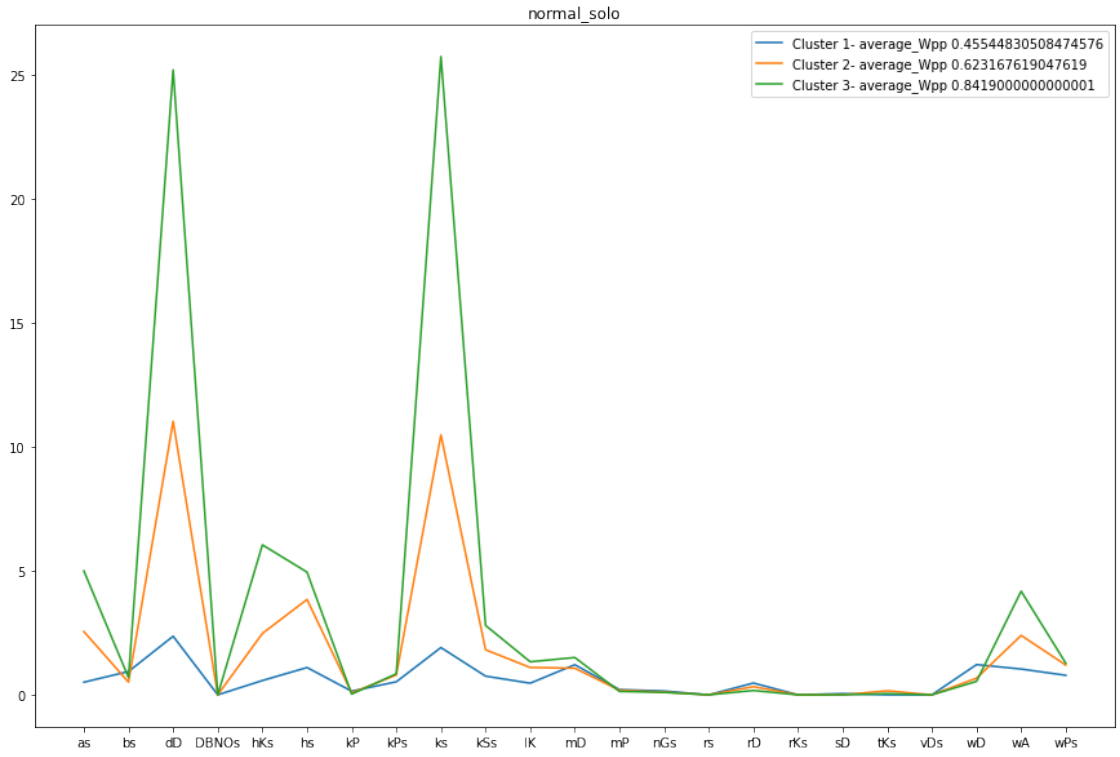

squad_fpp
Cluster 1- average_Wpp 0.778118222574964
Cluster 2- average_Wpp 0.37731190460366676
0
1
2
3
4
5
as
bs
dD
DBNOs
hKs
hs
kP
kPs
ks
kSs
lK
mD
mP
nGs
rs
rD
rKs
sD
tKs
vDs
wD
wA
wPs

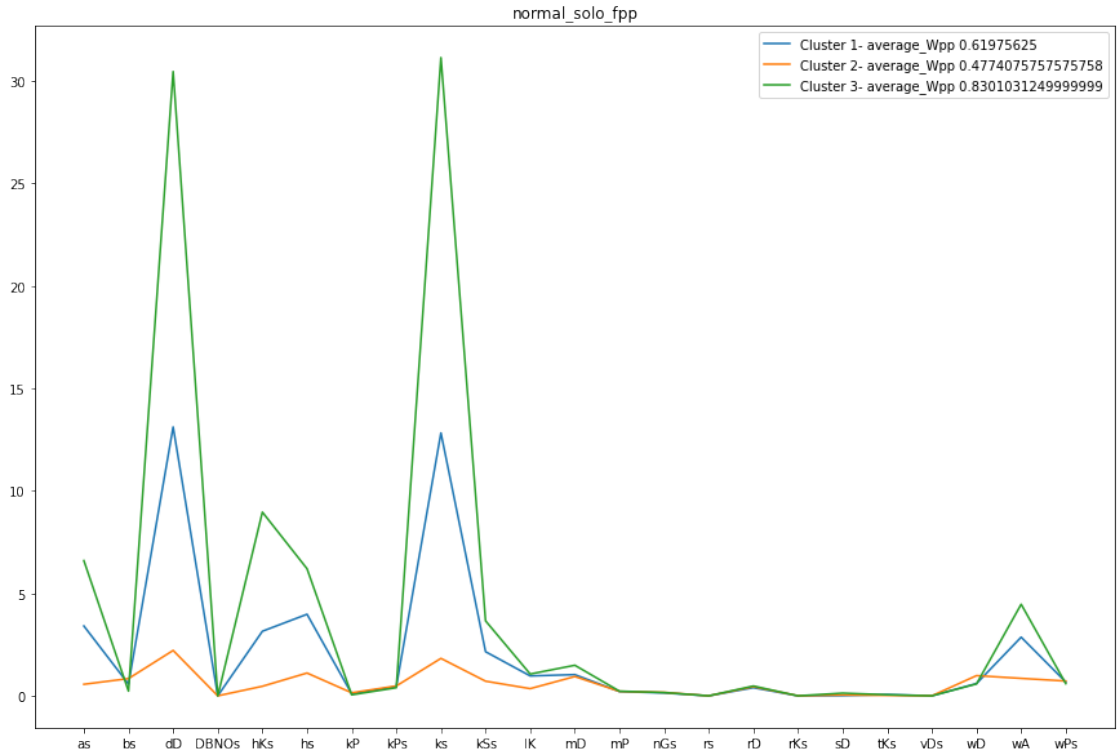
10.1.1 Comparison between normal_solo and normal_solo_fpp

[128]:
```
def get_centroids_single(X_out,kmeans,n_clusters,y_train_temp):

    #Maintain the original data
    X = X_out.copy()
    y_train_temp = y_train_temp.copy()

    #Train kmeans
    kmeans.n_clusters = n_clusters
    kmeans.fit(X)

    #Get the centroids
    centroids = kmeans.cluster_centers_

    #Get the most common matchtype and dominance
    n_members = []
    X_out["cluster"] = kmeans.labels_

    for i in range(n_clusters):
        temp_cluster = X_out[X_out["cluster"] == i]
        n_members.append(temp_cluster.shape[0])

    X_out = X_out.drop(['cluster'],axis=1)
    #Extract the centroids
    average_data = pd.DataFrame(centroids)
    average_data.columns = X_out.columns

    #Get the average WPP
    y_train_temp["cluster"] = kmeans.labels_
    average_Wpp = getAverageWPP(y_train_temp,n_clusters)

    #Add columns for most common matchtype, dominance and average WPP
    average_data["averageWpp"] = average_Wpp
    average_data["n_member"] = n_members

    average_data = average_data.sort_values('averageWpp',ascending=False)

    index = []
    for i in range(n_clusters):
        index.append("cluster_"+str(i+1))
    average_data.index = index

    return average_data
```

[24]:
```
types_with_three = [matchType_normal_solo,matchType_normal_solo_fpp]

names_with_three = ["normal_solo","normal_solo_fpp"]
```

[14]:
```
for i in range(len(types_with_three)):
    #Get the name
    name = names_with_three[i]

    #Get the dataframe with specific type, without changing the original data
    single_type = types_with_three[i].copy()

    #Extract X and y
    single_y = single_type["winPlacePerc"]
    single_X = single_type.
↪drop(['Id','groupId','matchId',"winPlacePerc"],axis=1)

    #Delete matchType column
    single_X_temp = single_X.drop(["matchType"],axis=1)

    #Convert numpy array to pandas dataframe
    single_y_temp = pd.DataFrame({"winPlacePerc": single_y})

    distinguish_clusters(kmeans_mt,single_X_temp,single_y_temp,3,name)
```





[39]:
```
for i in range(len(types_with_three)):
    #Get the name
    name = names_with_three[i]

    #Get the dataframe with specific type, without changing the original data
    single_type = types_with_three[i].copy()

    #Extract X and y
    single_y = single_type["winPlacePerc"]
    single_X = single_type.
↪drop(['Id','groupId','matchId',"winPlacePerc"],axis=1)

    #Delete matchType column
    single_X_temp = single_X.drop(["matchType"],axis=1)

    #Convert numpy array to pandas dataframe
    single_y_temp = pd.DataFrame({"winPlacePerc": single_y})

    print(name)
    with pd.option_context('display.max_rows', None, 'display.max_columns',␣
↪None):
        display(get_centroids_single(single_X_temp,kmeans,3,single_y_temp))
```

normal_solo

	assists	boosts	damageDealt	DBNOs	headshotKills	heals
cluster_1	5.000000	0.700000	25.213000	0.0	6.050000	4.950000
cluster_2	2.552381	0.514286	11.032390	0.0	2.476190	3.847619
cluster_3	0.508475	0.940678	2.363378	0.0	0.576271	1.101695

	killPlace	killPoints	kills	killStreaks	longestKill
cluster_1	0.028500	0.850000	25.750000	2.800000	1.332410
cluster_2	0.078000	0.800000	10.485714	1.819048	1.103699
cluster_3	0.146949	0.525424	1.906780	0.754237	0.468796

	matchDuration	maxPlace	numGroups	revives	rideDistance
cluster_1	1.505200	0.138000	0.099500	0.0	0.175195
cluster_2	1.072333	0.185238	0.109238	0.0	0.328385
cluster_3	1.215254	0.209915	0.153475	0.0	0.475163

	roadKills	swimDistance	teamKills	vehicleDestroys
cluster_1	0.0	-1.040834e-17	5.000000e-02	0.0
cluster_2	0.0	1.734723e-17	1.619048e-01	0.0
cluster_3	0.0	4.823814e-02	-1.387779e-16	0.0

	walkDistance	weaponsAcquired	winPoints	averageWpp	n_member
cluster_1	0.543443	4.180000	1.275000	0.841900	20
cluster_2	0.670332	2.397143	1.200000	0.623168	105
cluster_3	1.219670	1.040678	0.788136	0.455448	118

normal_solo_fpp

	assists	boosts	damageDealt	DBNOs	headshotKills	heals
cluster_1	6.593750	0.229167	30.446771	0.0	8.958333	6.197917
cluster_2	3.409396	0.574944	13.124499	0.0	3.154362	3.986577
cluster_3	0.564943	0.839849	2.220423	0.0	0.466583	1.110971

	killPlace	killPoints	kills	killStreaks	longestKill
cluster_1	0.048333	0.395833	31.125000	3.656250	1.062852
cluster_2	0.098389	0.449664	12.832215	2.152125	0.971810
cluster_3	0.149369	0.480454	1.833544	0.716267	0.351158

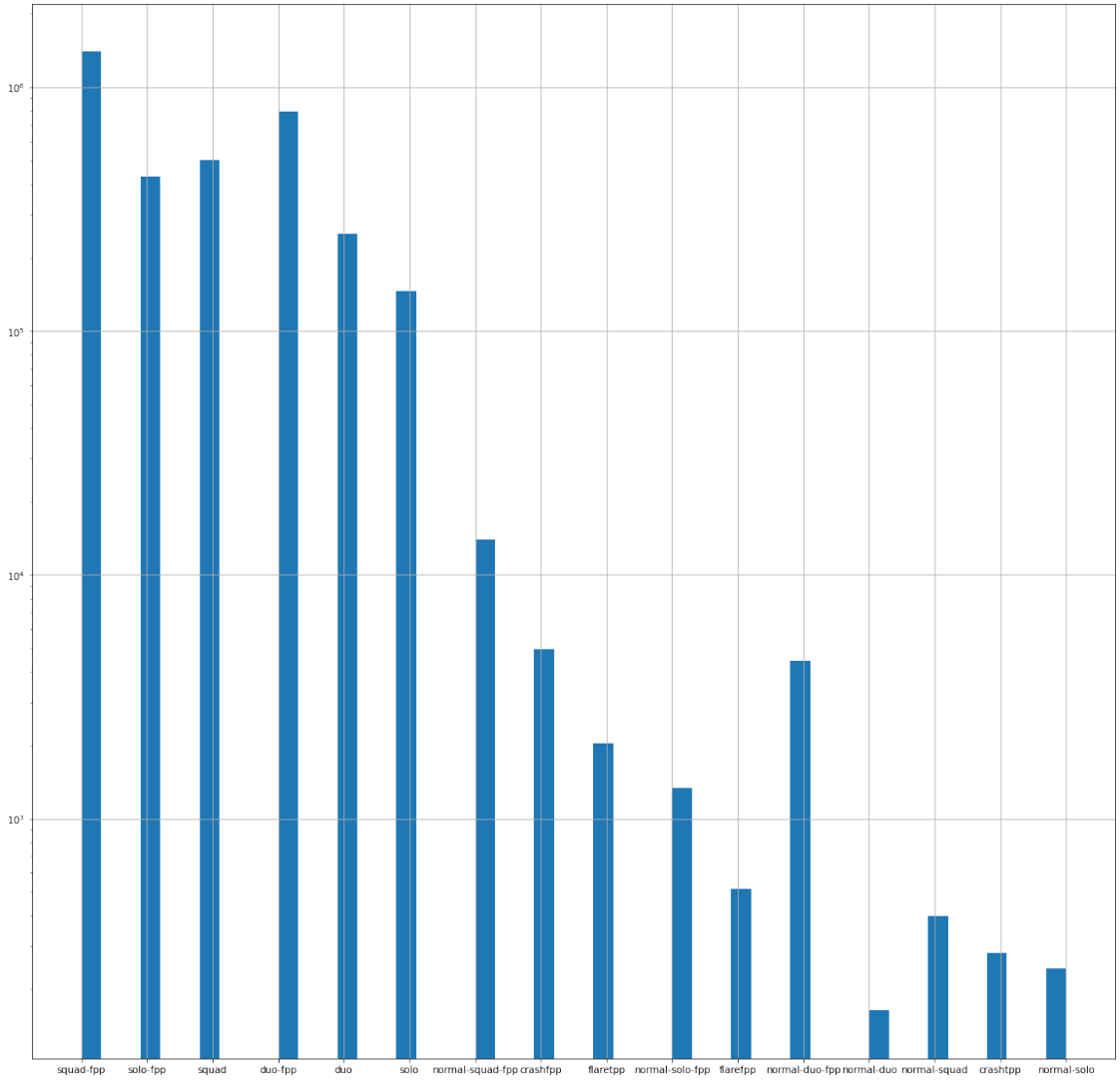
	matchDuration	maxPlace	numGroups	revives	rideDistance
cluster_1	1.490708	0.227187	0.160625	0.0	0.478400
cluster_2	1.034492	0.209306	0.141633	0.0	0.388631
cluster_3	0.941536	0.217781	0.168638	0.0	0.451418

	roadKills	swimDistance	teamKills	vehicleDestroys
cluster_1	1.301043e-18	0.131979	0.052083	0.000000e+00
cluster_2	-1.734723e-18	0.012051	0.062640	6.505213e-19
cluster_3	3.783102e-03	0.044994	0.022699	2.522068e-03

	walkDistance	weaponsAcquired	winPoints	averageWpp	n_member
cluster_1	0.581878	4.465625	0.593750	0.830103	96
cluster_2	0.597798	2.859060	0.674497	0.619756	448
cluster_3	0.985647	0.855485	0.720681	0.477408	792

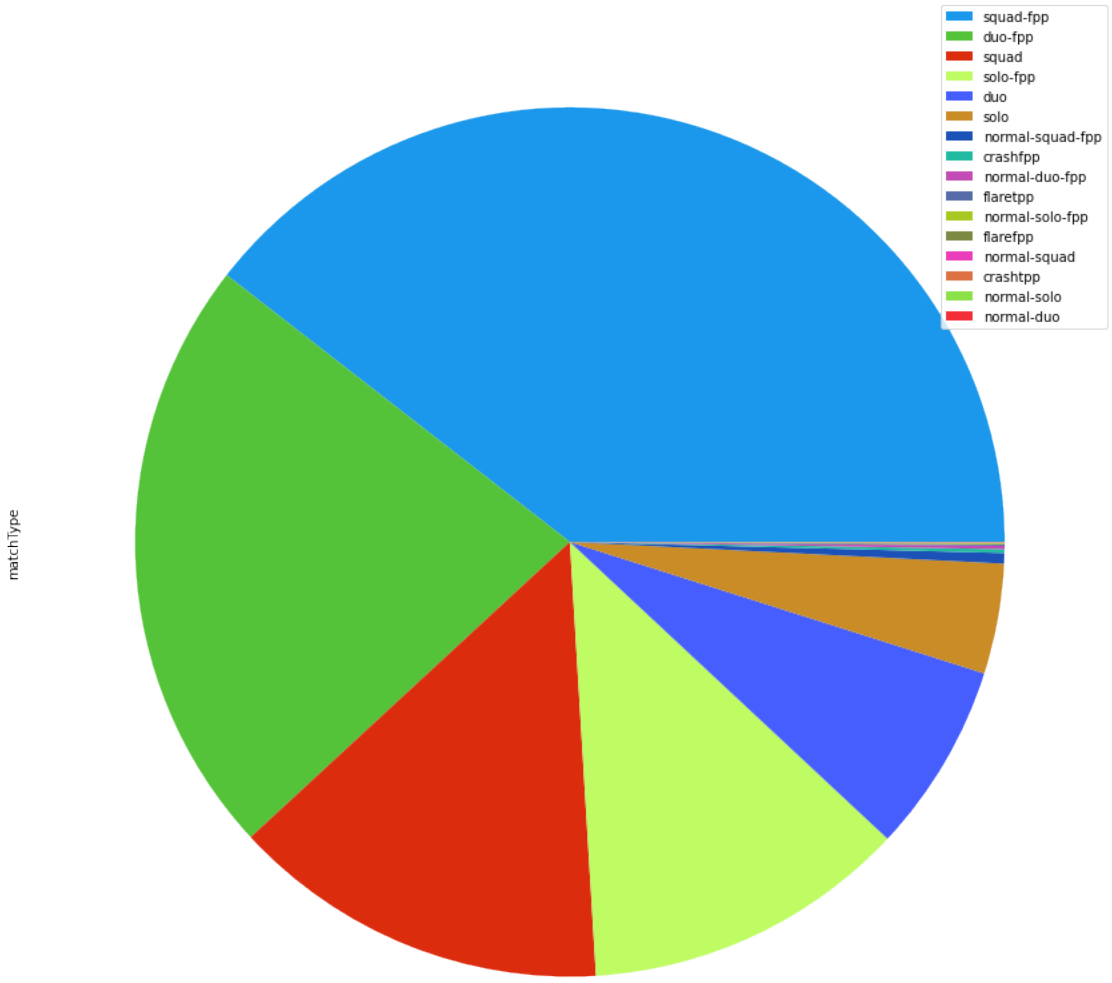
10.1.2 View matchType counts

[21]:
```
scaled_train['matchType'].astype('category').
 ↪hist(bins=50,figsize=(20,20),log=True)
plt.savefig("matchType_counts.png")
plt.show()
```

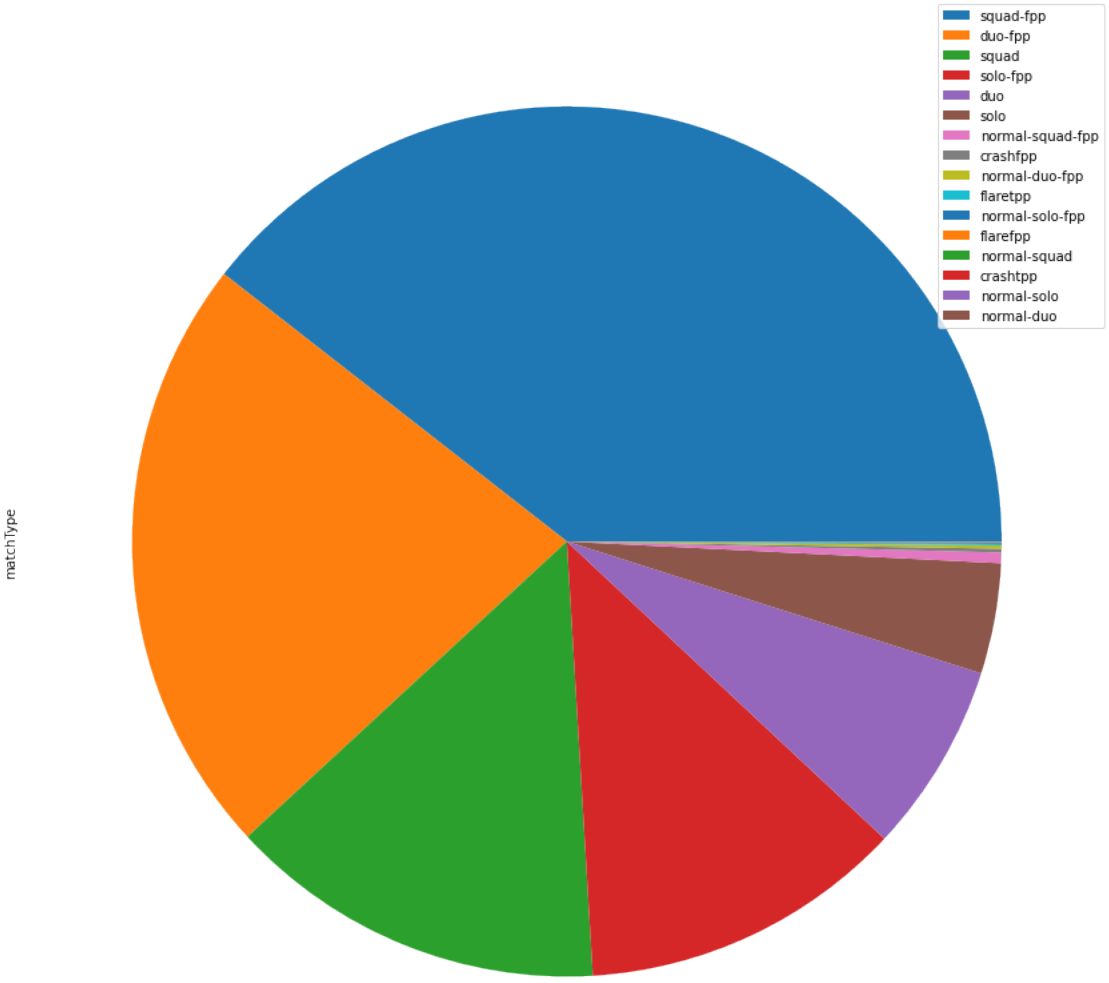


[19]:
```
from matplotlib import cm
cs=np.random.rand(18,3)
```

```
scaled_train['matchType'].value_counts().plot.
 ↪pie(figsize=(15,15),labels=None,legend=True,colors=cs)
plt.savefig("matchType_counts.png")
plt.show()
```



[29]:

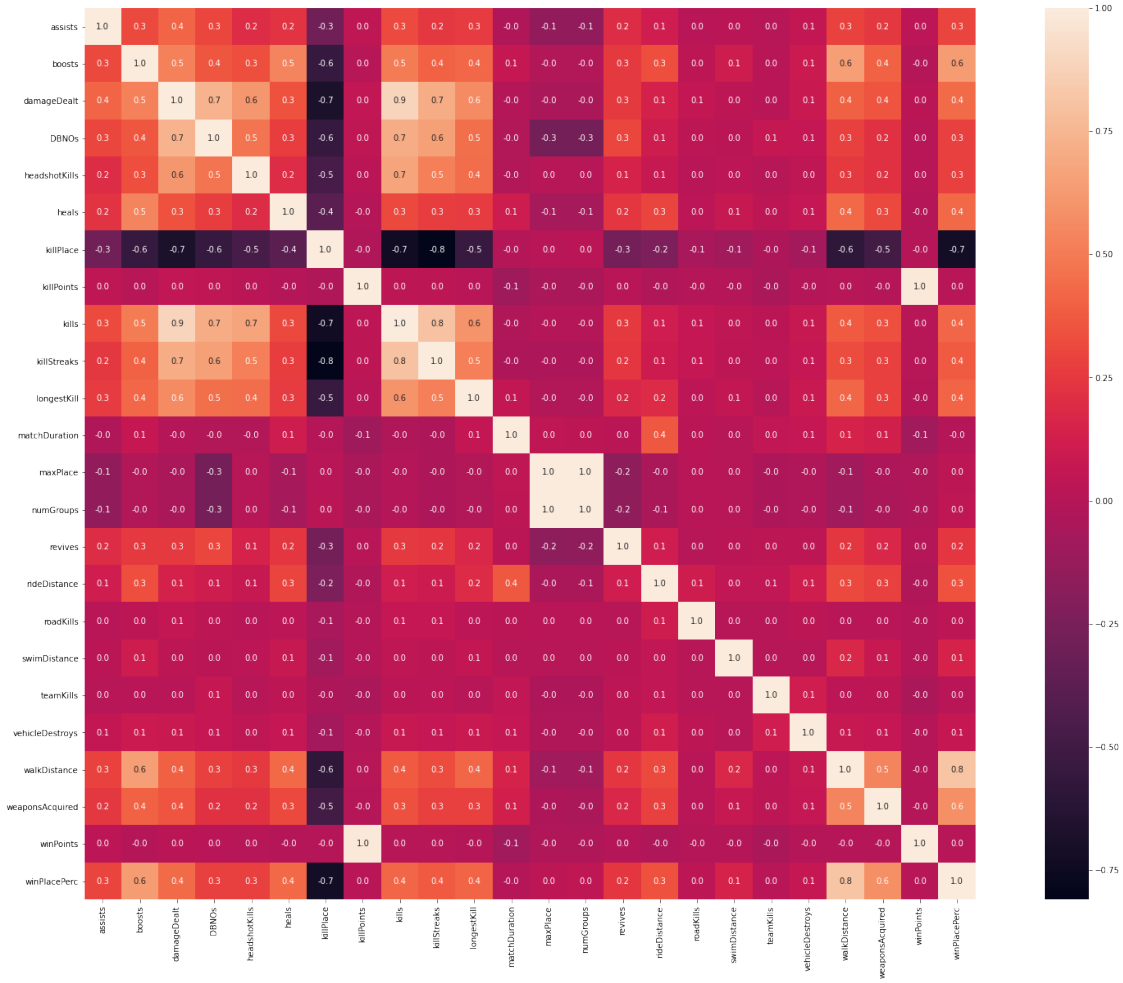


11 Plot Correlation Heat Map

The correlation heat map is the same as the previous one.

[42]:
```
def plot_heat_map(ds):
    dataset_ = ds.copy()
    corr = dataset_.corr()
    plt.subplots(figsize=(30,20))
    sns.heatmap( corr, square=True, annot=True, fmt=".01f" )
    plt.savefig("heat_map.png")
    plt.show()
```

[43]:
```
plot_heat_map(scaled_train)
```



12 Models

13 XGBoost hyperparameter tuning

[67]:
```
def timer(start_time=None):
    if not start_time:
        start_time=datetime.now()
        return start_time
    elif start_time:
        thour, temp_sec = divmod((datetime.now()-start_time).
↪total_seconds(),3600)
        tmin,tsec=divmod(temp_sec,60)
        print('\n Time taken: %i hours %i minutes and %s seconds.
↪'%(thour,tmin,round(tsec,2)))
```

[68]:
```
def XGBhyperParameterTuning(X_train, y_train):

    xgb_model = XGBRegressor()

    param_tuning = {
        'learning_rate': [0.01, 0.1, 0.5],
        'max_depth': [3, 5, 7, 10],
        'min_child_weight': [1, 3, 5, 7],
        'colsample_bytree': [0.5, 0.7],
        'n_estimators' : [100, 200, 500],
        'objective': ['reg:squarederror'],
        'tree_method':['gpu_hist'],
        'gpu_id':[0]
    }

    randomsearch = RandomizedSearchCV(estimator = xgb_model,
                                      param_distributions = param_tuning,
                                      n_iter = 5,
                                      scoring = 'neg_mean_absolute_error',␣
 ↪#MAE
                                      cv = 5,
                                      n_jobs = 1,
                                      verbose = 1)

    start_time = timer(None)
    randomsearch.fit(X_train,y_train)
    timer(start_time)

    return randomsearch.best_params_
```

[69]:
```
def random_sample(dictionary):

    new_dict = dictionary.copy()
    for key in new_dict:
        new_dict[key] = random.choice(new_dict[key])
    return new_dict
```

[70]:
```
def custom_XGBhyperParameterTuning(X_train, y_train, has_groupId, groupId =␣
 ↪None):

    X_train = X_train.copy()
    y_train = y_train.copy()
    xgb_model = XGBRegressor()
    param_tuning = {
        'learning_rate': [0.01, 0.1, 0.5],
        'max_depth': [3, 5, 7, 10],
        'min_child_weight': [1, 3, 5, 7],
```

```
        'colsample_bytree': [0.5, 0.7],
        'n_estimators' : [100, 200, 500],
        'objective': ['reg:squarederror'],
        'tree_method':['gpu_hist'],
        'gpu_id':[0]
    }

    start_time = timer(None)

    #Extract the validation set

    if has_groupId:

        temp_train = pd.concat([X_train,y_train],axis=1)
        train_inds, validate_inds = next(GroupShuffleSplit(test_size=.20,␣
↪n_splits=2, random_state = 7).split(temp_train, groups=groupId))

        train = temp_train.iloc[train_inds]
        validate = temp_train.iloc[validate_inds]
        train = train.reset_index(drop=True)
        validate = validate.reset_index(drop=True)
        y_train = train['winPlacePerc']
        X_train = train.drop(['winPlacePerc'],axis=1)
        y_validate = validate['winPlacePerc']
        X_validate = validate.drop(['winPlacePerc'],axis=1)

    else:

        X_train,X_validate,y_train,y_validate = train_test_split(X_train,␣
↪y_train, test_size = 0.2, random_state = 7)
        X_train = X_train.reset_index(drop=True)
        X_validate = X_validate.reset_index(drop=True)
        y_train = y_train.reset_index(drop=True)
        y_validate = y_validate.reset_index(drop=True)

    #Do the hyperparameter tuning
    best_feature = None
    best_mae = 100

    for i in range(10):

        random_feature = random_sample(param_tuning)
        xgb_model.set_params(**random_feature)
        xgb_model.fit(X_train,y_train)

        preds = xgb_model.predict(X_validate)
        mae = mean_absolute_error(preds,y_validate)
```

```
        print("Fit ",i+1," features: ", random_feature," MAE:", mae)

        if mae < best_mae:
            best_mae = mae
            best_feature = random_feature


    timer(start_time)

    return best_feature
```

13.1 Approach 1

Train on all data

[71]:
```
scaled_train_1 = scaled_train.drop(['Id','groupId','matchId'],axis=1)
scaled_train_1 = pd.get_dummies(data = scaled_train_1,columns=["matchType"])
scaled_train_1_y = scaled_train_1["winPlacePerc"]
scaled_train_1_X = scaled_train_1.drop(["winPlacePerc"],axis=1)
```

[72]:
```
scaled_test_1 = scaled_test.drop(['Id','groupId','matchId'],axis=1)
scaled_test_1 = pd.get_dummies(data = scaled_test_1,columns=["matchType"])
scaled_test_1_y = scaled_test_1["winPlacePerc"]
scaled_test_1_X = scaled_test_1.drop(["winPlacePerc"],axis=1)
```

[13]:
```
custom_XGBhyperParameterTuning(scaled_train_1_X,scaled_train_1_y,True,scaled_train["groupId"])
```

```
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07816944689676836
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06083776999844778
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06083776999844778
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05774865790795198
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.057462033262283545
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0588717581271869
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
```

```
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05992441339833259
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05811216393856049
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.058636111417038776
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09197491584131845

 Time taken: 0 hours 3 minutes and 52.3 seconds.
```

[13]:
```
{'learning_rate': 0.1,
 'max_depth': 5,
 'min_child_weight': 5,
 'colsample_bytree': 0.7,
 'n_estimators': 500,
 'objective': 'reg:squarederror',
 'tree_method': 'gpu_hist',
 'gpu_id': 0}
```

[73]:
```
xgboost_1 = XGBRegressor().set_params(**{'learning_rate': 0.1,
 'max_depth': 5,
 'min_child_weight': 5,
 'colsample_bytree': 0.7,
 'n_estimators': 500,
 'objective': 'reg:squarederror',
 'tree_method': 'gpu_hist',
 'gpu_id': 0})
```

[74]:
```
xgboost_1.fit(scaled_train_1_X,scaled_train_1_y)
```

[74]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.7, gamma=0, gpu_id=0,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.1, max_delta_step=0, max_depth=5,
             min_child_weight=5, missing=nan, monotone_constraints='()',
             n_estimators=500, n_jobs=12, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='gpu_hist', validate_parameters=1, verbosity=None)
```

[75]:
```
mean_absolute_error(xgboost_1.predict(scaled_test_1_X),scaled_test_1_y)
```

[75]:
```
0.057524919396732314
```

13.2 Approach 1

Train on all data - "Kaggle winner trick"

[76]:
```
scaled_train_1_k = scaled_train.drop(['Id','matchId'],axis=1)
scaled_train_1_k = pd.get_dummies(data = scaled_train_1_k,columns=["matchType"])
scaled_train_1_k = scaled_train_1_k.groupby("groupId").mean()
scaled_train_1_k_y = scaled_train_1_k["winPlacePerc"]
scaled_train_1_k_X = scaled_train_1_k.drop(["winPlacePerc"],axis=1)
```

[77]:
```
scaled_test_1_k = scaled_test.drop(['Id','matchId'],axis=1)
scaled_test_1_k = pd.get_dummies(data = scaled_test_1_k,columns=["matchType"])
scaled_test_1_k = scaled_test_1_k.groupby("groupId").transform('mean')
scaled_test_1_k_y = scaled_test_1_k["winPlacePerc"]
scaled_test_1_k_X = scaled_test_1_k.drop(["winPlacePerc"],axis=1)
```

[78]:
```
parameter_1_k = XGBhyperParameterTuning(scaled_train_1_k_X,scaled_train_1_k_y)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  6.4min finished


 Time taken: 0 hours 6 minutes and 46.68 seconds.
```

[79]:
```
xgboost_1_k = XGBRegressor().set_params(**parameter_1_k)
```

[80]:
```
xgboost_1_k.fit(scaled_train_1_k_X,scaled_train_1_k_y)
```

[80]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.7, gamma=0, gpu_id=0,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.5, max_delta_step=0, max_depth=7,
             min_child_weight=5, missing=nan, monotone_constraints='()',
             n_estimators=500, n_jobs=12, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='gpu_hist', validate_parameters=1, verbosity=None)
```

[81]:
```
mean_absolute_error(xgboost_1_k.predict(scaled_test_1_k_X),scaled_test_1_k_y)
```

[81]:
```
0.0505256542603541
```

14 Significance test of approach 1 - with/without kaggle

[85]:
```
set_1 = np.absolute(xgboost_1.predict(scaled_test_1_X)-scaled_test_1_y)
```

[86]:
```
set_2 = np.absolute(xgboost_1_k.predict(scaled_test_1_k_X)-scaled_test_1_k_y)
```

[87]:
```
stats.ttest_ind(set_1,set_2)
```

[87]:
```
Ttest_indResult(statistic=89.6495803783327, pvalue=0.0)
```

14.1 Approach 2

Train on each match type - with "Kaggle winner trick"

[88]:
```
#Extract each matchType
train_crashfpp=scaled_train[scaled_train['matchType']=='crashfpp']
train_crashtpp=scaled_train[scaled_train['matchType']=='crashtpp']
train_duo=scaled_train[scaled_train['matchType']=='duo']
train_duo_fpp=scaled_train[scaled_train['matchType']=='duo-fpp']
train_flarefpp=scaled_train[scaled_train['matchType']=='flarefpp']
train_flaretpp=scaled_train[scaled_train['matchType']=='flaretpp']
train_normal_duo=scaled_train[scaled_train['matchType']=='normal-duo']
train_normal_duo_fpp=scaled_train[scaled_train['matchType']=='normal-duo-fpp']
train_normal_solo=scaled_train[scaled_train['matchType']=='normal-solo']
train_normal_solo_fpp=scaled_train[scaled_train['matchType']=='normal-solo-fpp']
train_normal_squad=scaled_train[scaled_train['matchType']=='normal-squad']
train_normal_squad_fpp=scaled_train[scaled_train['matchType']=='normal-squad-fpp']
train_solo=scaled_train[scaled_train['matchType']=='solo']
train_solo_fpp=scaled_train[scaled_train['matchType']=='solo-fpp']
train_squad=scaled_train[scaled_train['matchType']=='squad']
train_squad_fpp=scaled_train[scaled_train['matchType']=='squad-fpp']
```

[89]:
```
#Extract each matchType
test_crashfpp=scaled_test[scaled_test['matchType']=='crashfpp']
test_crashtpp=scaled_test[scaled_test['matchType']=='crashtpp']
test_duo=scaled_test[scaled_test['matchType']=='duo']
test_duo_fpp=scaled_test[scaled_test['matchType']=='duo-fpp']
test_flarefpp=scaled_test[scaled_test['matchType']=='flarefpp']
test_flaretpp=scaled_test[scaled_test['matchType']=='flaretpp']
test_normal_duo=scaled_test[scaled_test['matchType']=='normal-duo']
test_normal_duo_fpp=scaled_test[scaled_test['matchType']=='normal-duo-fpp']
test_normal_solo=scaled_test[scaled_test['matchType']=='normal-solo']
test_normal_solo_fpp=scaled_test[scaled_test['matchType']=='normal-solo-fpp']
test_normal_squad=scaled_test[scaled_test['matchType']=='normal-squad']
test_normal_squad_fpp=scaled_test[scaled_test['matchType']=='normal-squad-fpp']
test_solo=scaled_test[scaled_test['matchType']=='solo']
test_solo_fpp=scaled_test[scaled_test['matchType']=='solo-fpp']
test_squad=scaled_test[scaled_test['matchType']=='squad']
test_squad_fpp=scaled_test[scaled_test['matchType']=='squad-fpp']
```

[90]:
```
#Add all match types to a list
train_allTypes =␣
↪[train_crashfpp,train_crashtpp,train_duo,train_duo_fpp,train_flarefpp,train_flaretpp,train_
```

```
test_allTypes =␣
 ↪[test_crashfpp,test_crashtpp,test_duo,test_duo_fpp,test_flarefpp,test_flaretpp,test_normal_
```

With "Kaggle winner trick"

[55]:
```
num_test = scaled_test.shape[0]
sum_error = 0
set_3 = np.array([])
for index in range(len(train_allTypes)):

    temp_train = train_allTypes[index]
    temp_test = test_allTypes[index]

    temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
    temp_train = temp_train.groupby("groupId").mean()
    temp_train_y = temp_train["winPlacePerc"]
    temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

    temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
    temp_test = temp_test.groupby("groupId").transform('mean')
    temp_test_y = temp_test["winPlacePerc"]
    temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

    temp_parameter = XGBhyperParameterTuning(temp_train_X,temp_train_y)
    temp_xgb = XGBRegressor().set_params(**temp_parameter)
    temp_xgb.fit(temp_train_X,temp_train_y)
    sum_error += mean_absolute_error(temp_xgb.
 ↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]
    set_3 = np.append(set_3,np.absolute(temp_xgb.
 ↪predict(temp_test_X)-temp_test_y))

print(sum_error/num_test)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   21.5s finished


 Time taken: 0 hours 0 minutes and 22.66 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    8.6s finished


 Time taken: 0 hours 0 minutes and 8.85 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   40.3s finished
```

```
 Time taken: 0 hours 0 minutes and 42.44 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  4.6min finished


 Time taken: 0 hours 4 minutes and 42.62 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    8.7s finished


 Time taken: 0 hours 0 minutes and 9.3 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   19.5s finished


 Time taken: 0 hours 0 minutes and 21.01 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    6.9s finished


 Time taken: 0 hours 0 minutes and 7.18 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   25.6s finished


 Time taken: 0 hours 0 minutes and 26.92 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    6.9s finished


 Time taken: 0 hours 0 minutes and 7.59 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   30.5s finished


 Time taken: 0 hours 0 minutes and 31.89 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    7.2s finished
```

```
 Time taken: 0 hours 0 minutes and 7.5 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   10.2s finished


 Time taken: 0 hours 0 minutes and 10.95 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  3.5min finished


 Time taken: 0 hours 3 minutes and 38.74 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  1.7min finished


 Time taken: 0 hours 1 minutes and 45.88 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  1.4min finished


 Time taken: 0 hours 1 minutes and 27.75 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  1.3min finished


 Time taken: 0 hours 1 minutes and 20.52 seconds.
0.05047853237746263
```

[63]:
```
num_test = scaled_test.shape[0]
sum_error = 0
set_3 = np.array([])
for index in range(len(train_allTypes)):

    temp_train = train_allTypes[index]
    temp_test = test_allTypes[index]

    temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
    temp_train = temp_train.groupby("groupId").mean()
    temp_train_y = temp_train["winPlacePerc"]
    temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)
```

```
    temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
    temp_test = temp_test.groupby("groupId").transform('mean')
    temp_test_y = temp_test["winPlacePerc"]
    temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

    temp_parameter =␣
 ↪custom_XGBhyperParameterTuning(temp_train_X,temp_train_y,False)
    temp_xgb = XGBRegressor().set_params(**temp_parameter)
    temp_xgb.fit(temp_train_X,temp_train_y)
    sum_error += mean_absolute_error(temp_xgb.
 ↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]
    set_3 = np.append(set_3,np.absolute(temp_xgb.
 ↪predict(temp_test_X)-temp_test_y))

print(sum_error/num_test)
```

```
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07612513641003821
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07153136870245233
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.08289374595302222
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07826349397904306
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07248294599944068
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07440258247387228
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07616621061212073
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0687456605391144
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08462233170956072
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.0704565986601105

 Time taken: 0 hours 0 minutes and 6.33 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13812060101562076
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1147130803156782
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11125668544841033
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10710825420419375
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11900701096786392
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12176461543700208
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10904578761760558
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11288106318668081
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09681416920865021
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13179501076715963

 Time taken: 0 hours 0 minutes and 4.58 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05482428504271033
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04838934891897755
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
```

```
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0492881149016964
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05085273211944196
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05060762640696794
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.054278108199573856
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04939471146903024
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05407636376566389
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04720089698409791
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12056985346398676

 Time taken: 0 hours 0 minutes and 25.94 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04470792262073838
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11289392056429787
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.046173952526165324
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11003016991017915
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08076226614653559
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
```

```
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05436063875934094
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.044727247395054126
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.058269218677133916
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.043178832181287465
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11649960819776375

 Time taken: 0 hours 0 minutes and 55.38 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11893237502328281
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09447416605795252
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09844048323795712
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10128615201999401
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11900616351818216
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09459174831817889
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10129597516224302
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10227534653112806
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
```

```
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10423553371676082
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09435632866333271

 Time taken: 0 hours 0 minutes and 4.37 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.15256768674871565
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09547068949739193
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1561139058370506
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09600334221599376
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10814091372933007
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0785085731916723
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.08621711715025977
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09153923589754684
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07475113937245534
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08438028873074636

 Time taken: 0 hours 0 minutes and 6.87 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.1858088869796859
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.18724102705319723
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13297400429447492
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1503529541651408
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.20925991309483846
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.15742404813170432
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.19916240777704453
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.126670502295759
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.13535044079356723
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.18529547740485933

 Time taken: 0 hours 0 minutes and 3.76 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.17094564201234463
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.15925450622768195
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09984671532116299
```

```
Fit 4 features: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.10473435641062662
Fit 5 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.10949821976766215
Fit 6 features: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.09451400088499945
Fit 7 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.09294707996679615
Fit 8 features: {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.10313333817571893
Fit 9 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.1290281754707023
Fit 10 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.09400018097192622

 Time taken: 0 hours 0 minutes and 14.19 seconds.
Fit 1 features: {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.12303137085437775
Fit 2 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.12336673820086885
Fit 3 features: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.12314727041789461
Fit 4 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.1223075037424905
Fit 5 features: {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.1124910267455237
Fit 6 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.12421275648117062
```

```
Fit 7 features: {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.16365691734586446
Fit 8 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.13114398708752223
Fit 9 features: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.13313483833653586
Fit 10 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.11459663568428584

 Time taken: 0 hours 0 minutes and 5.86 seconds.
Fit 1 features: {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.1352256385863668
Fit 2 features: {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.2038439054337526
Fit 3 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.1494375298217412
Fit 4 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.14283083762120743
Fit 5 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.11913655992249171
Fit 6 features: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.1308827154045533
Fit 7 features: {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.12495726012487442
Fit 8 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.13599611865713834
Fit 9 features: {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
```

```
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1815616985895695
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.14515893438339236

 Time taken: 0 hours 0 minutes and 11.3 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.28518460565328596
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.2782512522280216
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.28946722599983216
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.28220298508644104
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.27301177990913394
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.28363702925384043
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.28425324919939043
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.25466864931702615
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.28055054830074305
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2845025411349535

 Time taken: 0 hours 0 minutes and 3.87 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
```

```
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12826166423293778
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1284701316398018
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.14153214575369868
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12258295335982726
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.14163809908405528
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.18178058770360234
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.13564800893444262
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1331143595376187
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1367547157218674
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12483350078192405

 Time taken: 0 hours 0 minutes and 10.18 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04443514693464364
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04357240308008356
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05002285481166547
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05518908334893882
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
```

```
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.055009444154132184
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.045463533358541
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0639764004860469
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0639764004860469
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04122468413790919
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0639764004860469

 Time taken: 0 hours 0 minutes and 48.15 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.03938614747789844
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0606574108098699
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.040343760508365696
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04096178154568487
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04094224261404976
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04086166273903191
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.039106507968769126
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
```

```
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.03949122659432899
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06068359330790411
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.039769979554058456

 Time taken: 0 hours 0 minutes and 59.4 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06553446330702595
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06470413542795206
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06464129669548385
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07313071114332842
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08771843197350407
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.067437308479777
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06789902852881156
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13323498192212885
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0654196217658179
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08734537258556335

 Time taken: 0 hours 0 minutes and 30.86 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
```

```
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05832423264058174
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13093593284068158
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.061544099614353844
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05763663819172616
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05803187931561145
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.059216046365894705
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07701953442830628
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06196387410457631
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13430375623103538
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.059741843206779596

 Time taken: 0 hours 0 minutes and 35.16 seconds.
0.04944479233074204
```

[69]:
```
stats.ttest_ind(set_2,set_3)
```

[69]:
```
Ttest_indResult(statistic=-4.892376603554839, pvalue=9.963424765441378e-07)
```

14.1.1 Different model selection for approach 2

According to the the counts below:

Use XGBoost for duo, duo-fpp, normal-duo-fpp, solo, solo-fpp, squad, squad-fpp.

Use SGDRegressor for normal-squad-fpp

Use SVR(kernel = 'linear') for crashfpp, crashtpp, flarefpp, flaretpp, normal-duo, normal-duo-fpp, normal-solo, normal-solo-fpp, normal-squad

(As suggested by scikit-learn cheat sheet)

https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

[24]:
```
pd.DataFrame({'counts': scaled_train.groupby(('matchType')).size(),
                            'Percentage': scaled_train.groupby(('matchType')).
↪size() / len(train_data)})
```

[24]:

matchType	counts	Percentage
crashfpp	4972	0.001118
crashtpp	282	0.000063
duo	251120	0.056470
duo-fpp	796987	0.179220
flarefpp	516	0.000116
flaretpp	2041	0.000459
normal-duo	163	0.000037
normal-duo-fpp	4428	0.000996
normal-solo	243	0.000055
normal-solo-fpp	1336	0.000300
normal-squad	398	0.000089
normal-squad-fpp	13959	0.003139
solo	146105	0.032855
solo-fpp	429670	0.096621
squad	500754	0.112606
squad-fpp	1403320	0.315568

[18]:
```
#Extract each matchType
train_crashfpp=scaled_train[scaled_train['matchType']=='crashfpp']
train_crashtpp=scaled_train[scaled_train['matchType']=='crashtpp']
train_duo=scaled_train[scaled_train['matchType']=='duo']
train_duo_fpp=scaled_train[scaled_train['matchType']=='duo-fpp']
train_flarefpp=scaled_train[scaled_train['matchType']=='flarefpp']
train_flaretpp=scaled_train[scaled_train['matchType']=='flaretpp']
train_normal_duo=scaled_train[scaled_train['matchType']=='normal-duo']
train_normal_duo_fpp=scaled_train[scaled_train['matchType']=='normal-duo-fpp']
train_normal_solo=scaled_train[scaled_train['matchType']=='normal-solo']
train_normal_solo_fpp=scaled_train[scaled_train['matchType']=='normal-solo-fpp']
train_normal_squad=scaled_train[scaled_train['matchType']=='normal-squad']
train_normal_squad_fpp=scaled_train[scaled_train['matchType']=='normal-squad-fpp']
train_solo=scaled_train[scaled_train['matchType']=='solo']
train_solo_fpp=scaled_train[scaled_train['matchType']=='solo-fpp']
train_squad=scaled_train[scaled_train['matchType']=='squad']
train_squad_fpp=scaled_train[scaled_train['matchType']=='squad-fpp']
```

[19]:
```
#Extract each matchType
test_crashfpp=scaled_test[scaled_test['matchType']=='crashfpp']
test_crashtpp=scaled_test[scaled_test['matchType']=='crashtpp']
test_duo=scaled_test[scaled_test['matchType']=='duo']
test_duo_fpp=scaled_test[scaled_test['matchType']=='duo-fpp']
test_flarefpp=scaled_test[scaled_test['matchType']=='flarefpp']
test_flaretpp=scaled_test[scaled_test['matchType']=='flaretpp']
test_normal_duo=scaled_test[scaled_test['matchType']=='normal-duo']
test_normal_duo_fpp=scaled_test[scaled_test['matchType']=='normal-duo-fpp']
test_normal_solo=scaled_test[scaled_test['matchType']=='normal-solo']
test_normal_solo_fpp=scaled_test[scaled_test['matchType']=='normal-solo-fpp']
test_normal_squad=scaled_test[scaled_test['matchType']=='normal-squad']
test_normal_squad_fpp=scaled_test[scaled_test['matchType']=='normal-squad-fpp']
test_solo=scaled_test[scaled_test['matchType']=='solo']
test_solo_fpp=scaled_test[scaled_test['matchType']=='solo-fpp']
test_squad=scaled_test[scaled_test['matchType']=='squad']
test_squad_fpp=scaled_test[scaled_test['matchType']=='squad-fpp']
```

[29]:
```
train_xgb_allTypes =␣
↪[train_duo,train_duo_fpp,train_normal_duo_fpp,train_solo,train_solo_fpp,train_squad,train_s
test_xgb_allTypes =␣
↪[test_duo,test_duo_fpp,test_normal_duo_fpp,test_solo,test_solo_fpp,test_squad,test_squad_fp
```

[30]:
```
train_sgd_allTypes = [train_normal_squad_fpp]
test_sgd_allTypes = [test_normal_squad_fpp]
```

[35]:
```
train_svr_allTypes =␣
↪[train_crashfpp,train_crashtpp,train_flarefpp,train_flaretpp,train_normal_duo,train_normal_
test_svr_allTypes =␣
↪[test_crashfpp,test_crashtpp,test_flarefpp,test_flaretpp,test_normal_duo,test_normal_duo_fp
```

[38]:
```
num_test = scaled_test.shape[0]
err_xgb = 0
err_sgd = 0
err_svr = 0

#XGB
for index in range(len(train_xgb_allTypes)):
    temp_train = train_xgb_allTypes[index]
    temp_test = test_xgb_allTypes[index]

    temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
    temp_train = temp_train.groupby("groupId").mean()
    temp_train_y = temp_train["winPlacePerc"]
    temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

    temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
```

```
    temp_test = temp_test.groupby("groupId").transform('mean')
    temp_test_y = temp_test["winPlacePerc"]
    temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

    temp_xgb = XGBRegressor(objective="reg:squarederror")
    temp_xgb.fit(temp_train_X,temp_train_y)
    err_xgb += mean_absolute_error(temp_xgb.
 ↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]

#SGD
temp_train = train_sgd_allTypes[0]
temp_test = test_sgd_allTypes[0]

temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
temp_train = temp_train.groupby("groupId").mean()
temp_train_y = temp_train["winPlacePerc"]
temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
temp_test = temp_test.groupby("groupId").transform('mean')
temp_test_y = temp_test["winPlacePerc"]
temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

temp_sgd = SGDRegressor()
temp_sgd.fit(temp_train_X,temp_train_y)
err_sgd += mean_absolute_error(temp_sgd.
 ↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]

#SVR
for index in range(len(train_svr_allTypes)):
    temp_train = train_svr_allTypes[index]
    temp_test = test_svr_allTypes[index]

    temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
    temp_train = temp_train.groupby("groupId").mean()
    temp_train_y = temp_train["winPlacePerc"]
    temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

    temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
    temp_test = temp_test.groupby("groupId").transform('mean')
    temp_test_y = temp_test["winPlacePerc"]
    temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

    temp_svr = LinearSVR(loss = 'epsilon_insensitive') #L1 loss
    temp_svr.fit(temp_train_X,temp_train_y)
    err_svr += mean_absolute_error(temp_svr.
 ↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]
```

```
(err_xgb+err_sgd+err_svr)/num_test
```

[38]:
```
0.05858968283308794
```

14.2 Approach 3

Train on different cluster, using kaggle winner trick.

[64]:
```
def approach_3(train,test,n_clusters,kmeans,first_kt):
    train = train.copy()
    test = test.copy()

    if first_kt:
        train_d = train.drop(['Id','matchId',"matchType"],axis=1)
        train_d = train_d.groupby("groupId").transform('mean')
        train_y = train_d["winPlacePerc"]
        train_X = train_d.drop(["winPlacePerc"],axis=1)

        test_d = test.drop(['Id','matchId',"matchType"],axis=1)
        test_d = test_d.groupby("groupId").transform('mean')
        test_y = test_d["winPlacePerc"]
        test_X = test_d.drop(["winPlacePerc"],axis=1)
    else:
        train_d = train.drop(['Id','matchId','groupId',"matchType"],axis=1)
        train_y = train_d["winPlacePerc"]
        train_X = train_d.drop(["winPlacePerc"],axis=1)

        test_d = test.drop(['Id','matchId','groupId',"matchType"],axis=1)
        test_y = test_d["winPlacePerc"]
        test_X = test_d.drop(["winPlacePerc"],axis=1)

    kmeans.n_clusters = n_clusters
    kmeans.fit(train_X)

    train["cluster"] = kmeans.labels_
    test["cluster"] = kmeans.predict(test_X)

    train_n = train.drop(['Id','matchId','matchType'],axis=1)
    test_n = test.drop(['Id','matchId','matchType'],axis=1)

    train_n["matchType"] = train["matchType"]
    test_n["matchType"] = test["matchType"]
    train_n = pd.get_dummies(data = train_n,columns=["matchType"])
    test_n = pd.get_dummies(data = test_n,columns=["matchType"])

    num_test = test.shape[0]
    sum_error = 0
```

```
    temp_set = np.array([])
    for i in range(n_clusters):

        temp_train_n = train_n[train_n["cluster"] == i]
        temp_test_n = test_n[test_n["cluster"] == i]

        temp_train_n.drop(["cluster"],axis=1)
        temp_test_n.drop(["cluster"],axis=1)

        temp_train_n = temp_train_n.groupby("groupId").mean()
        temp_train_n_y = temp_train_n["winPlacePerc"]
        temp_train_n_X = temp_train_n.drop(["winPlacePerc"],axis=1)

        temp_test_n = temp_test_n.groupby("groupId").transform('mean')
        temp_test_n_y = temp_test_n["winPlacePerc"]
        temp_test_n_X = temp_test_n.drop(["winPlacePerc"],axis=1)

        temp_parameter =␣
↪custom_XGBhyperParameterTuning(temp_train_n_X,temp_train_n_y,False)
        temp_xgb = XGBRegressor().set_params(**temp_parameter)
        temp_xgb.fit(temp_train_n_X,temp_train_n_y)
        sum_error += mean_absolute_error(temp_xgb.
↪predict(temp_test_n_X),temp_test_n_y)*temp_test_n.shape[0]
        temp_set = np.append(temp_set,np.absolute(temp_xgb.
↪predict(temp_test_n_X)-temp_test_n_y))

    return sum_error/num_test,temp_set
```

[58]:
```
kmeans = KMeans(random_state=10,n_init = 10)
```

kaggle trick applied before clustering

[59]:
```
mae,set_4 = approach_3(scaled_train,scaled_test,2,kmeans,True)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  4.1min finished


 Time taken: 0 hours 4 minutes and 18.4 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  2.3min finished


 Time taken: 0 hours 2 minutes and 23.68 seconds.
```

[60]:
```
mae
```

[60]:
```
0.05145748508669292
```

[61]:
```
stats.ttest_ind(set_2,set_4)
```

[61]:
```
Ttest_indResult(statistic=-33.11426767990838, pvalue=2.1932557689840957e-240)
```

[62]:
```
stats.ttest_ind(set_3,set_4)
```

[62]:
```
Ttest_indResult(statistic=-13.646251764903981, pvalue=2.1358035452754248e-42)
```

[65]:
```
mae,set_4 = approach_3(scaled_train,scaled_test,2,kmeans,True)
```

```
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.045357470843814515
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04519997577054853
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.044902192334544073
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04539555931243005
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04719145814294863
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04500963380585551
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04612848172795308
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07643639862859815
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0529023563382154
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.047008666205741095
```

```
 Time taken: 0 hours 2 minutes and 56.5 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.061696271246890255
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06339182831717698
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06232799592958938
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08216897322260658
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06460549031234435
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07074631459421582
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06376048313930724
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.061073377646533505
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06434621843649745
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06339182831717698

 Time taken: 0 hours 1 minutes and 0.41 seconds.
```

[66]:
```
mae
```

[66]:
```
0.05051294522035498
```

[67]:
```
stats.ttest_ind(set_2,set_4)
```

[67]:
```
Ttest_indResult(statistic=-20.101950969967373, pvalue=7.260149002223263e-90)
```

[68]:
```
stats.ttest_ind(set_3,set_4)
```

[68]:
```
Ttest_indResult(statistic=-15.200651427583807, pvalue=3.5277775764109265e-52)
```

14.2.1 Train on the clusters within each matchType

[88]:
```
train_two =␣
 ↪[train_crashfpp,train_crashtpp,train_duo,train_duo_fpp,train_flarefpp,train_flaretpp,train_
test_two =␣
 ↪[test_crashfpp,test_crashtpp,test_duo,test_duo_fpp,test_flarefpp,test_flaretpp,test_normal_
```

[91]:
```
train_three = [train_normal_solo,train_normal_solo_fpp]
test_three = [test_normal_solo,test_normal_solo_fpp]
```

[92]:
```
kmeans = KMeans(random_state=10,n_init = 10)
```

[93]:
```
num = scaled_test.shape[0]
sum_err = 0
set_5 = np.array([])
for i in range(len(train_two)):

    train_single = train_two[i]
    test_single = test_two[i]
    mae,temp_set = approach_3(train_single,test_single,2,kmeans,True)
    sum_err += test_single.shape[0]*mae
    set_5 = np.append(set_5,temp_set)

for i in range(len(train_three)):

    train_single = train_three[i]
    test_single = test_three[i]
    mae,temp_set = approach_3(train_single,test_single,3,kmeans,True)
    sum_err += test_single.shape[0]*mae
    set_5 = np.append(set_5,temp_set)

print(sum_err/num)
```

```
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09187945363998413
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09130576115910599
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.08347530674073754
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.08299432755191151
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.09249321075718578
Fit 6 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.08323012282162177
Fit 7 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.0866807767128363
Fit 8 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.0880106373291481
Fit 9 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.08558289678364266
Fit 10 features: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.08580721296496507

 Time taken: 0 hours 0 minutes and 6.84 seconds.
Fit 1 features: {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.13180246216177532
Fit 2 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.05907197026150858
Fit 3 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.06676212924962825
Fit 4 features: {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.13063631566644127
Fit 5 features: {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.07183046774326014
Fit 6 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.07257007447945728
Fit 7 features: {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.1312037678106544
Fit 8 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.06657189621544768
Fit 9 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
```

```
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06812742615355967
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09043244363348099

 Time taken: 0 hours 0 minutes and 7.68 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0939446112060547
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1183443149471283
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09766709083557129
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10725496254920959
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13658571086883545
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11943915690422058
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10256868475914001
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12500314868927004
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10616435612678528
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10287264789581299

 Time taken: 0 hours 0 minutes and 3.47 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10245607315529796
```

```
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1469993253778009
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11461735449862831
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11227822072497184
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10172779363875879
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10930799182197626
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10245607315529796
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10547752134852549
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10484759330705684
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10660602507258163

 Time taken: 0 hours 0 minutes and 2.74 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.047064294897217036
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06317295802260811
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11975166705209178
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
```

```
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04578719808446934
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11348439280975714
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04679100494241054
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04496112966876481
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.046073962904120275
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04399438439793425
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11812654507987279

 Time taken: 0 hours 0 minutes and 26.86 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0618258728968698
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06095949259810599
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07068974824292144
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06232385746946388
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06175064155359607
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06600804770645713
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06155707399028916
```

```
Fit 8 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.06028112869619331
Fit 9 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.062116725238804585
Fit 10 features: {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.13819941479609812

 Time taken: 0 hours 0 minutes and 20.98 seconds.
Fit 1 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.05422139790237517
Fit 2 features: {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.06378354847988847
Fit 3 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.055223617456098606
Fit 4 features: {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.1363226673320269
Fit 5 features: {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.13589699880012396
Fit 6 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.05520038588030012
Fit 7 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.0565984376299649
Fit 8 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.055489391202383484
Fit 9 features: {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.13834858772479003
Fit 10 features: {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.13835056118551253
```

```
 Time taken: 0 hours 0 minutes and 14.6 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06275543288866664
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04410414177555562
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10539987343879233
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04436393097996034
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04224329966241983
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.041679507992309996
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04113218433600756
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10540308390061717
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.041871496378198376
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.045278280137661976

 Time taken: 0 hours 1 minutes and 24.95 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06949800178805988
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12734757702085708
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
```

```
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13418194824324714
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0839109049787124
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12734757702085708
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07792320306433571
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10731790124641524
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13335284266736772
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06873903779387475
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10304720104336738

 Time taken: 0 hours 0 minutes and 4.45 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13086131859620412
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09706131440798445
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11289142196178441
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09255230184396113
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.12466251004536948
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12466251004536948
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1015090277194977
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1352546791156133
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10715008598963423
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1352546791156133

 Time taken: 0 hours 0 minutes and 3.02 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1310739935926489
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0847505202347463
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08846907389582813
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.072425240548096
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07500847685555549
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0767600465814155
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08544097435314497
```

```
Fit 8 features: {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.07270255607680694
Fit 9 features: {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.08009755655270623
Fit 10 features: {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.07061623109898253

 Time taken: 0 hours 0 minutes and 5.58 seconds.
Fit 1 features: {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.0933466250371933
Fit 2 features: {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.16112472071886064
Fit 3 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.09674487021088601
Fit 4 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.10078391294240951
Fit 5 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.10158175471782685
Fit 6 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.09331568209171295
Fit 7 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.09685977170705797
Fit 8 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.10515339763879776
Fit 9 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.0979028436589241
Fit 10 features: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.09295530183792114
```

```
 Time taken: 0 hours 0 minutes and 4.74 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.13140256789539542
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.15469475274937494
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.18202693252989224
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.14193920830041168
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12470477910744293
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1413500412583351
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12211249554029534
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11564610138301339
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.15117665351799556
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.17684342817578996

 Time taken: 0 hours 0 minutes and 5.43 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.3723470582485199
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.4768168940067291
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.319707357776165
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
```

```
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.5045665516376495
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.369906814289093
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.31971542675495146
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2990989698886871
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.3009300171256065
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.29828220744132994
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.3066169931411743

 Time taken: 0 hours 0 minutes and 2.53 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0944176604683924
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10538633203996071
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10327628719820474
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.16867980593422907
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09501307529854884
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10276723498107077
```

```
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10087871972103056
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09000400171119281
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11006469784206767
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.17874820865069646

 Time taken: 0 hours 0 minutes and 11.9 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.20068796759049098
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.18955728302246486
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1830452233228928
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.18220678231838422
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.18732466549659385
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.18947270953655243
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.20259008486079863
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.2015395573782615
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.19614457698296278
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.21060531982641953
```

```
 Time taken: 0 hours 0 minutes and 3.74 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.2884131853310267
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.3195329124259949
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.31423639253060026
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.2786788401842118
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.2706386221289635
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.28432722342650096
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.31660366136233015
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.28618873906771347
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.2706869345855713
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2933853773117066

 Time taken: 0 hours 0 minutes and 4.9 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.3204589595397313
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.3432785762627919
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.35464797516663865
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
```

```
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2790559175888697
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.29095339874426523
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.2781497978448868
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.3068311770757039
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.3204589595397313
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.34709184467792503
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.3426434159278869

 Time taken: 0 hours 0 minutes and 4.24 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11835633621809566
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.15879694286213228
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1295178353720201
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11244806905104306
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13233693649779368
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11668520331921334
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
```

```
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11818978348226032
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11992850995063178
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12547666510528158
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11332043350293002

 Time taken: 0 hours 0 minutes and 15.94 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.17872390298619367
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2123423376607402
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.17862997146900866
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.19572407694415958
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.19790324848658286
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.17280305868091198
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.19632621790562704
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.19632621790562704
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
```

```
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.17872390298619367
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.18873955896185748

 Time taken: 0 hours 0 minutes and 12.61 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04048995643999185
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08066700199508703
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.038746643284866274
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04593595594439519
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04166809484106594
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.038902950960271306
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11539493415143712
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04213593897546428
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.039960688767201095
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.041309599285131345

 Time taken: 0 hours 0 minutes and 44.72 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.1404360636096296
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05950227054967875
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1404360636096296
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06008514544664538
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05809101645275189
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06821632982695748
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.060083670516499176
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06706606982189504
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06821632982695748
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.058856599135848474

 Time taken: 0 hours 0 minutes and 32.82 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.03748168051966286
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.03801559469206356
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.03701296255576662
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0377472040890415
```

```
Fit 5 features: {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.037527707406058965
Fit 6 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.0374440908382239
Fit 7 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.03621996839418054
Fit 8 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.037459872572320745
Fit 9 features: {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.06207227165138864
Fit 10 features: {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.1135817982473125

 Time taken: 0 hours 1 minutes and 16.34 seconds.
Fit 1 features: {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.08029609098459897
Fit 2 features: {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.07768157789998315
Fit 3 features: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.051162555110242494
Fit 4 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.051155831613707264
Fit 5 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.059320843618843114
Fit 6 features: {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.05512518675864879
Fit 7 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.050888129957069136
```

```
Fit 8 features: {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.05446150264388573
Fit 9 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.05101895573081631
Fit 10 features: {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.05174395090393675

 Time taken: 0 hours 0 minutes and 58.71 seconds.
Fit 1 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.059626361389305306
Fit 2 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.060424432537413725
Fit 3 features: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.05963007893032031
Fit 4 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.060620081956987225
Fit 5 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.05964899433145786
Fit 6 features: {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.0637401853202052
Fit 7 features: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.06017212399311554
Fit 8 features: {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.07639366947880265
Fit 9 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.06747874945090744
Fit 10 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.06817567785253122
```

```
 Time taken: 0 hours 1 minutes and 9.91 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1451813892333114
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07628179395577213
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07768257650403394
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08851936685646758
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0782535168233
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08807131550606542
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.14265493618510106
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07878889895528758
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09780246665136226
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07839315086350815

 Time taken: 0 hours 1 minutes and 7.02 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05808731699635269
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
```

```
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05609715323270765
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05730682361778427
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.055728628297923036
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06107640726532623
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06063939835865376
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05445186816840309
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05375661822464722
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.055549265379962826
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05479670740316737

 Time taken: 0 hours 1 minutes and 22.62 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07069601955613869
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13744310639680724
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06924500540416599
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08534393921522057
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.0710304543624615
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0755325452112886
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06777559464825929
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09278907921106562
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07109255066253294
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.088147873497725

 Time taken: 0 hours 0 minutes and 53.15 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.20742480928488075
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.18949686046093706
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12559221724048258
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2008195372471586
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.21166269163489343
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.21458947154581548
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.16642512259930375
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12591837105229498
```

```
Fit 9 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.16590676717311148
Fit 10 features: {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.20075021075885743

 Time taken: 0 hours 0 minutes and 4.51 seconds.
Fit 1 features: {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.1980663615290324
Fit 2 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.1860106098683675
Fit 3 features: {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.19005340037981672
Fit 4 features: {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.21800767956415815
Fit 5 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.19268615564028424
Fit 6 features: {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.1593189853254954
Fit 7 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.17411400874773664
Fit 8 features: {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.19304133035977683
Fit 9 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.20177927015463515
Fit 10 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.17027627724647526

 Time taken: 0 hours 0 minutes and 2.86 seconds.
Fit 1 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
```

```
0.3604298902893066
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.29591349495887753
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.3250865055465698
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.3257611884021759
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2423567844772339
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.29665387405395505
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2901836885356903
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.30898068679809565
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2875032020950318
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.36712995304107665

 Time taken: 0 hours 0 minutes and 3.96 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.15707049910491844
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1585394507119975
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1657449332949581
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
```

```
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.155085633248387
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1628105321428907
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1704769066644466
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.16887967269383078
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1704052830598486
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.16568824706359173
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.15220292403308422

 Time taken: 0 hours 0 minutes and 7.7 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0759077907037735
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.084810209941864
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11226558862686158
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11840540833473204
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09314194984436033
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09701428369522093
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
```

```
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07872110970020293
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0512040144920349
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09824336833953856
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05860815427303313

 Time taken: 0 hours 0 minutes and 3.44 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12844768124592715
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1293638806834303
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12846357834421354
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1099482622785815
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11292274177444393
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1530231534522155
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10837505894323876
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.18643509697996336
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1940856358396596
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.11158310289691234

 Time taken: 0 hours 0 minutes and 3.57 seconds.
0.04979613968338985
```

[97]:
```
stats.ttest_ind(set_2,set_5)
```

[97]:
```
Ttest_indResult(statistic=-9.902933668324893, pvalue=4.047967698538534e-23)
```

[98]:
```
stats.ttest_ind(set_3,set_5)
```

[98]:
```
Ttest_indResult(statistic=-5.013931374148071, pvalue=5.333411462438877e-07)
```

15 with threshold

[81]:
```
train_two =␣
 ↪[train_crashfpp,train_duo,train_duo_fpp,train_flaretpp,train_normal_duo_fpp,train_normal_sq
test_two =␣
 ↪[test_crashfpp,test_duo,test_duo_fpp,test_flaretpp,test_normal_duo_fpp,test_normal_squad_fp
```

[82]:
```
train_three = [train_normal_solo_fpp]
test_three = [test_normal_solo_fpp]
```

[83]:
```
train_dont =␣
 ↪[train_crashtpp,train_flarefpp,train_normal_duo,train_normal_solo,train_normal_squad]
test_dont =␣
 ↪[test_crashtpp,test_flarefpp,test_normal_duo,test_normal_solo,test_normal_squad]
```

[84]:
```
kmeans = KMeans(random_state=10,n_init = 10)
```

[86]:
```
num = scaled_test.shape[0]
sum_err = 0
set_6 = np.array([])
for i in range(len(train_two)):

    train_single = train_two[i]
    test_single = test_two[i]
    mae,temp_set = approach_3(train_single,test_single,2,kmeans,True)
    sum_err += test_single.shape[0]*mae
    set_6 = np.append(set_6,temp_set)

for i in range(len(train_three)):

    train_single = train_three[i]
    test_single = test_three[i]
    mae,temp_set = approach_3(train_single,test_single,3,kmeans,True)
    sum_err += test_single.shape[0]*mae
```

```
    set_6 = np.append(set_6,temp_set)

for i in range(len(train_dont)):

    temp_train = train_dont[i]
    temp_test = test_dont[i]

    temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
    temp_train = temp_train.groupby("groupId").mean()
    temp_train_y = temp_train["winPlacePerc"]
    temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

    temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
    temp_test = temp_test.groupby("groupId").transform('mean')
    temp_test_y = temp_test["winPlacePerc"]
    temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

    temp_parameter =␣
 ↪custom_XGBhyperParameterTuning(temp_train_X,temp_train_y,False)
    temp_xgb = XGBRegressor().set_params(**temp_parameter)
    temp_xgb.fit(temp_train_X,temp_train_y)
    sum_err += mean_absolute_error(temp_xgb.
 ↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]
    set_6 = np.append(set_6,np.absolute(temp_xgb.
 ↪predict(temp_test_X)-temp_test_y))

print(sum_err/num)
```

```
Fit 1 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.09056123005332016
Fit 2 features: {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.14286520917985496
Fit 3 features: {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.09011821937793638
Fit 4 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.08453271202180446
Fit 5 features: {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.08299432755191151
Fit 6 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
```

```
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0880106373291481
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1408845393050589
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.08334386424227458
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13547596672267448
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13547596672267448

 Time taken: 0 hours 0 minutes and 7.55 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0990289082237945
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06645307891695597
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0726692777663418
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0735703381837272
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07243199114983333
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06757686094610192
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09665753210701072
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.0767879165695321
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09609356640663344
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06940401394311164

 Time taken: 0 hours 0 minutes and 21.89 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07120199338723736
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05007895349261084
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11349829812348615
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04551046347564493
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04399438439793425
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06093927997804066
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04958267073092667
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.047902116687468375
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.045496828246720966
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04574937805243538
```

```
 Time taken: 0 hours 1 minutes and 13.11 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06095949259810599
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06739282502444366
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.061358802204141805
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06135138222637498
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1394998586724369
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06078009007699672
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06645455278411683
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06836765421988628
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06669735119273504
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06203670916748192

 Time taken: 0 hours 0 minutes and 20.41 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13446635133374132
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05683615900836811
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.06105459538172715
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.054572987733786285
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05423984540373146
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05789282086880551
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.14070706219112064
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05406149699515699
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.056688675736696434
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0545759427618495

 Time taken: 0 hours 0 minutes and 20.34 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04135894290468451
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04272960495687033
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04382872517954512
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04531182449654792
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.041679507992309996
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1097022651594116
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
```

```
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04179663430979093
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.040339664729769194
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04180847092826895
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05483887388950296

 Time taken: 0 hours 0 minutes and 56.98 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13295706143926928
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06944197607631417
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0722022307450405
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08811915474936768
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08882047745923738
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07285744443288124
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07316938649536224
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.072842718077159
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07503535979507944
```

```
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07270255607680694

 Time taken: 0 hours 0 minutes and 9.76 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10732091043114662
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10943872697472572
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.16103085904955866
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11659193879604342
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.08735434221863746
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09785149363040926
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1142231055402756
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09498520480990409
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09051936327934265
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09424788986444474

 Time taken: 0 hours 0 minutes and 3.52 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1542869270487083
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
```

```
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09652991393793295
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11587623306318795
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1634652877231296
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09986602335740685
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10062718777001459
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10538633203996071
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11595636756403162
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.15590184867296286
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12421860760887748

 Time taken: 0 hours 0 minutes and 12.91 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.19127333093912172
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.23513390542727253
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.17899973635001062
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.18661067612476837
```

```
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.19573648319030418
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.19549506047077667
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2004751513750125
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.22788784095201736
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.18725857573105736
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2049462195738768

 Time taken: 0 hours 0 minutes and 5.77 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.15887423588896574
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1116566058588629
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11474456438322359
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.13221770378290582
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12968915070342316
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11383944723283483
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11479552310908289
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
```

```
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11907795641228724
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12934248124954673
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11967918407130262

 Time taken: 0 hours 0 minutes and 29.36 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.19232572667712952
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2391185103429244
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.184932344802754
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.18306731899256673
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.18406427226698638
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2393870974729525
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.18332525537120015
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.181438462780426
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.19240275650329935
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.19314272375778865
```

```
 Time taken: 0 hours 0 minutes and 5.92 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.040592227585680235
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.038902950960271306
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.060184268841362173
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04095422420812349
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05044692133125308
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.040905463873607883
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.03798495332294779
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0413560393719639
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0403020578464432
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0406074015177714

 Time taken: 0 hours 0 minutes and 52.39 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08698653426366333
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06969745582301132
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.0686654542846442
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06830685527558794
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05866261092241932
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.14046313049906314
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05871122205410833
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05828295903580993
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.058819367619994775
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06048075294805455

 Time taken: 0 hours 0 minutes and 39.7 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.071468070110515
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.036615431192222304
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.037013833585348474
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04095406246791216
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10036815885733642
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.04130900871037241
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.03534835337548692
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.039451693154680216
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.03559616452637493
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.039451693154680216

 Time taken: 0 hours 2 minutes and 25.46 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08029609098459897
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.051806039813931136
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.050311368145045963
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.14243560545582631
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05045397889482475
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05356517105858541
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05946015605019341
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.051382176811261886
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
```

```
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05194505739632526
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05140479177077056

 Time taken: 0 hours 1 minutes and 9.8 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07141375793976099
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06255398849995444
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06308044443565305
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08147785953746688
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05990525041322822
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.0610489739436388
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12480420794807726
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.060718103431264395
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06312281005243335
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.05964770080943892

 Time taken: 0 hours 0 minutes and 44.6 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
```

```
0.10033481088158974
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07763264518917935
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10187820516320564
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.14051258243275208
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08755158533549352
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07879134132637387
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07806720025220369
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1393841711617969
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08846465758074137
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0797784197148449

 Time taken: 0 hours 0 minutes and 32.46 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05499731562893069
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05495583472741545
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.054428736445051355
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
```

```
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12725836521589032
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.05419102863200341
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12140188086085603
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1163962169422906
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0652708507730382
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.055303133196082904
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.053391238998038576

 Time taken: 0 hours 0 minutes and 35.87 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.14299211244243454
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06958853137069224
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13744310639680724
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07072929926745965
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07064211602370081
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06928354742951848
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
```

```
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0909929204407483
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07157210357533157
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09665915901630609
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13636622313640015

 Time taken: 0 hours 0 minutes and 45.2 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.19069246064520726
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.19337745600159711
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.16614716778445662
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1632309815707521
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.18306423975239675
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.17535174137514842
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2012484207326033
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.16910879731579562
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.18024468687299905
```

```
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.16133568795064657

 Time taken: 0 hours 0 minutes and 6.54 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06000265262126921
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06363353026866912
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.14175379343032835
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12185285158157347
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06299983329772949
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.09342469529151914
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06606178662776946
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06764464445114135
Fit  9  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07966452134609223
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07966452134609223

 Time taken: 0 hours 0 minutes and 4.72 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11508780501649298
```

```
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11772205313937419
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.18965325175811504
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11736468709131768
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11791567040373539
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1446506952992801
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12030595956465297
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11189242492647009
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.137960055491431
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1292441609900573

 Time taken: 0 hours 0 minutes and 6.07 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.15379835443761614
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10112148916478508
Fit  3  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10078689820700219
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11030516152329467
```

```
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13795557783003207
Fit  6  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.13132025807919326
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1028375610931052
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10616521967925407
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.18053419562445747
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11266377263786614

 Time taken: 0 hours 0 minutes and 4.7 seconds.
Fit  1  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.11466841563068586
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.10356640993521132
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.13121781765119783
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.15440932118317177
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.09435632866333271
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.10541923578186281
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.11603175968877201
```

```
Fit 8 features: {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.09742351775556032
Fit 9 features: {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.12257128483180342
Fit 10 features: {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.11917120004440176

 Time taken: 0 hours 0 minutes and 3.83 seconds.
Fit 1 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.18724102705319723
Fit 2 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.16028843170669343
Fit 3 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.19551083302895228
Fit 4 features: {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.16361911109222307
Fit 5 features: {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.12765012373261983
Fit 6 features: {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.1612043895284335
Fit 7 features: {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.1537186727391349
Fit 8 features: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0} MAE: 0.1303300463292334
Fit 9 features: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.15313917908006244
Fit 10 features: {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0} MAE:
0.139080061374108

 Time taken: 0 hours 0 minutes and 4.22 seconds.
Fit 1 features: {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
```

```
3, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1435833840233939
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.16661430567877636
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.16570611301830837
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.16659253465652468
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12303137085437775
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.1529241750434467
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.12673204865796225
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.16323158667360035
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.12952170795542853
Fit  10  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
3, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.1242641543327059

 Time taken: 0 hours 0 minutes and 6.03 seconds.
Fit  1  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.28055054830074305
Fit  2  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.27490226389527317
Fit  3  features:  {'learning_rate': 0.01, 'max_depth': 7, 'min_child_weight':
```

```
7, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.27788361380338666
Fit  4  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
3, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2744185368949175
Fit  5  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.27140068406701084
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.26365271189093586
Fit  7  features:  {'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2628941715455055
Fit  8  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.27646417016088964
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2737452587640285
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.2868453652085364

 Time taken: 0 hours 0 minutes and 5.71 seconds.
0.04968187740460711
```

[94]:
```
stats.ttest_ind(set_2,set_6)
```

[94]:
```
Ttest_indResult(statistic=-8.263216738503965, pvalue=1.4189394763435278e-16)
```

[95]:
```
stats.ttest_ind(set_3,set_6)
```

[95]:
```
Ttest_indResult(statistic=-3.3813325058326744, pvalue=0.000721367445481827)
```

[96]:
```
stats.ttest_ind(set_5,set_6)
```

[96]:
```
Ttest_indResult(statistic=1.6242733455530836, pvalue=0.10431766020713831)
```

16 SGDRegressor hyperparameter tuning

[7]:
```
def timer(start_time=None):
    if not start_time:
        start_time=datetime.now()
        return start_time
    elif start_time:
        thour, temp_sec = divmod((datetime.now()-start_time).
↪total_seconds(),3600)
        tmin,tsec=divmod(temp_sec,60)
        print('\n Time taken: %i hours %i minutes and %s seconds.
↪'%(thour,tmin,round(tsec,2)))
```

[8]:
```
def SGDhyperParameterTuning(X_train, y_train):

    sgd_model = SGDRegressor()

    param_tuning = {
        'loss':['squared_loss'],
        'penalty' : ['elasticnet'],
        'alpha' : [0.0001, 0.001, 0.01],
        'l1_ratio' : [0.15,0.35,0.55],
        'max_iter':[1000,2000,5000],
        'eta0' : [0.1,0.01,0.001]
    }

    randomsearch = RandomizedSearchCV(estimator = sgd_model,
                                      param_distributions = param_tuning,
                                      n_iter = 5,
                                      scoring = 'neg_mean_absolute_error',␣
↪#MAE
                                      cv = 5,
                                      n_jobs = 1,
                                      verbose = 1)

    start_time = timer(None)
    randomsearch.fit(X_train,y_train)
    timer(start_time)

    return randomsearch.best_params_
```

[9]:
```
def random_sample(dictionary):

    new_dict = dictionary.copy()
    for key in new_dict:
        new_dict[key] = random.choice(new_dict[key])
    return new_dict
```

[10]:
```
def custom_SGDhyperParameterTuning(X_train, y_train, has_groupId, groupId =␣
↪None):

    X_train = X_train.copy()
    y_train = y_train.copy()
    sgd_model = SGDRegressor()
    param_tuning = {
        'loss':['squared_loss'],
        'penalty' : ['elasticnet'],
        'alpha' : [0.0001, 0.001, 0.01],
        'l1_ratio' : [0.15,0.35,0.55],
        'max_iter':[1000,2000,5000],
        'eta0' : [0.1,0.01,0.001]
    }

    start_time = timer(None)

    #Extract the validation set
    print("Start Splitting...")
    if has_groupId:

        temp_train = pd.concat([X_train,y_train],axis=1)
        train_inds, validate_inds = next(GroupShuffleSplit(test_size=.20,␣
↪n_splits=2, random_state = 7).split(temp_train, groups=groupId))

        train = temp_train.iloc[train_inds]
        validate = temp_train.iloc[validate_inds]
        train = train.reset_index(drop=True)
        validate = validate.reset_index(drop=True)
        y_train = train['winPlacePerc']
        X_train = train.drop(['winPlacePerc'],axis=1)
        y_validate = validate['winPlacePerc']
        X_validate = validate.drop(['winPlacePerc'],axis=1)

    else:

        X_train,X_validate,y_train,y_validate = train_test_split(X_train,␣
↪y_train, test_size = 0.2, random_state = 7)
        X_train = X_train.reset_index(drop=True)
        X_validate = X_validate.reset_index(drop=True)
        y_train = y_train.reset_index(drop=True)
        y_validate = y_validate.reset_index(drop=True)

    #Do the hyperparameter tuning
    best_feature = None
    best_mae = 100
```

```
    print("Start validation...")
    for i in range(10):

        random_feature = random_sample(param_tuning)
        sgd_model.set_params(**random_feature)

        print("Start training...",random_feature)
        sgd_model.fit(X_train,y_train)

        preds = sgd_model.predict(X_validate)
        mae = mean_absolute_error(preds,y_validate)

        print(" MAE:", mae)

        if mae < best_mae:
            best_mae = mae
            best_feature = random_feature


    timer(start_time)

    return best_feature
```

16.1 Approach 1

Train on all data

[11]:
```
scaled_train_1 = scaled_train.drop(['Id','groupId','matchId'],axis=1)
scaled_train_1 = pd.get_dummies(data = scaled_train_1,columns=["matchType"])
scaled_train_1_y = scaled_train_1["winPlacePerc"]
scaled_train_1_X = scaled_train_1.drop(["winPlacePerc"],axis=1)
```

[12]:
```
scaled_test_1 = scaled_test.drop(['Id','groupId','matchId'],axis=1)
scaled_test_1 = pd.get_dummies(data = scaled_test_1,columns=["matchType"])
scaled_test_1_y = scaled_test_1["winPlacePerc"]
scaled_test_1_X = scaled_test_1.drop(["winPlacePerc"],axis=1)
```

[13]:
```
custom_SGDhyperParameterTuning(scaled_train_1_X,scaled_train_1_y,True,scaled_train["groupId"])
```

```
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.09269226168717268
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 0.09175578860207272
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
```

```
0.01, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.11293003571523456
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.10098053770708296
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.11411631191729643
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.11423898864617779
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.09243003244064031
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.11418909050921619
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.08999620506533187
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.09139022200250317

 Time taken: 0 hours 1 minutes and 29.98 seconds.
```

[13]:
```
{'loss': 'squared_loss',
 'penalty': 'elasticnet',
 'alpha': 0.0001,
 'l1_ratio': 0.15,
 'max_iter': 1000,
 'eta0': 0.01}
```

[14]:
```
sgd_1 = SGDRegressor().set_params(**{'loss': 'squared_loss',
 'penalty': 'elasticnet',
 'alpha': 0.0001,
 'l1_ratio': 0.15,
 'max_iter': 1000,
 'eta0': 0.01})
```

[15]:
```
sgd_1.fit(scaled_train_1_X,scaled_train_1_y)
```

[15]:
```
SGDRegressor(penalty='elasticnet')
```

[16]:
```
mean_absolute_error(sgd_1.predict(scaled_test_1_X),scaled_test_1_y)
```

[16]:
```
0.09027456341269986
```

16.2 Approach 1

Train on all data - "Kaggle winner trick"

[17]:
```
scaled_train_1_k = scaled_train.drop(['Id','matchId'],axis=1)
scaled_train_1_k = pd.get_dummies(data = scaled_train_1_k,columns=["matchType"])
scaled_train_1_k = scaled_train_1_k.groupby("groupId").mean()
scaled_train_1_k_y = scaled_train_1_k["winPlacePerc"]
scaled_train_1_k_X = scaled_train_1_k.drop(["winPlacePerc"],axis=1)
```

[18]:
```
scaled_test_1_k = scaled_test.drop(['Id','matchId'],axis=1)
scaled_test_1_k = pd.get_dummies(data = scaled_test_1_k,columns=["matchType"])
scaled_test_1_k = scaled_test_1_k.groupby("groupId").transform('mean')
scaled_test_1_k_y = scaled_test_1_k["winPlacePerc"]
scaled_test_1_k_X = scaled_test_1_k.drop(["winPlacePerc"],axis=1)
```

[32]:
```
parameter_1_k = SGDhyperParameterTuning(scaled_train_1_k_X,scaled_train_1_k_y)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  1.4min finished


 Time taken: 0 hours 1 minutes and 27.02 seconds.
```

[33]:
```
sgd_1_k = SGDRegressor().set_params(**parameter_1_k)
```

[34]:
```
sgd_1_k.fit(scaled_train_1_k_X,scaled_train_1_k_y)
```

[34]:
```
SGDRegressor(max_iter=2000, penalty='elasticnet')
```

[35]:
```
mean_absolute_error(sgd_1_k.predict(scaled_test_1_k_X),scaled_test_1_k_y)
```

[35]:
```
0.07480550582764552
```

17 Significance test of approach 1 - with/without kaggle

[23]:
```
set_1 = np.absolute(sgd_1.predict(scaled_test_1_X)-scaled_test_1_y)
```

[36]:
```
set_2 = np.absolute(sgd_1_k.predict(scaled_test_1_k_X)-scaled_test_1_k_y)
```

[25]:
```
stats.ttest_ind(set_1,set_2)
```

[25]:
```
Ttest_indResult(statistic=134.51653408206255, pvalue=0.0)
```

17.1 Approach 2

Train on each match type - with "Kaggle winner trick"

[26]:
```
#Extract each matchType
train_crashfpp=scaled_train[scaled_train['matchType']=='crashfpp']
train_crashtpp=scaled_train[scaled_train['matchType']=='crashtpp']
train_duo=scaled_train[scaled_train['matchType']=='duo']
train_duo_fpp=scaled_train[scaled_train['matchType']=='duo-fpp']
train_flarefpp=scaled_train[scaled_train['matchType']=='flarefpp']
train_flaretpp=scaled_train[scaled_train['matchType']=='flaretpp']
train_normal_duo=scaled_train[scaled_train['matchType']=='normal-duo']
train_normal_duo_fpp=scaled_train[scaled_train['matchType']=='normal-duo-fpp']
train_normal_solo=scaled_train[scaled_train['matchType']=='normal-solo']
train_normal_solo_fpp=scaled_train[scaled_train['matchType']=='normal-solo-fpp']
train_normal_squad=scaled_train[scaled_train['matchType']=='normal-squad']
train_normal_squad_fpp=scaled_train[scaled_train['matchType']=='normal-squad-fpp']
train_solo=scaled_train[scaled_train['matchType']=='solo']
train_solo_fpp=scaled_train[scaled_train['matchType']=='solo-fpp']
train_squad=scaled_train[scaled_train['matchType']=='squad']
train_squad_fpp=scaled_train[scaled_train['matchType']=='squad-fpp']
```

[27]:
```
#Extract each matchType
test_crashfpp=scaled_test[scaled_test['matchType']=='crashfpp']
test_crashtpp=scaled_test[scaled_test['matchType']=='crashtpp']
test_duo=scaled_test[scaled_test['matchType']=='duo']
test_duo_fpp=scaled_test[scaled_test['matchType']=='duo-fpp']
test_flarefpp=scaled_test[scaled_test['matchType']=='flarefpp']
test_flaretpp=scaled_test[scaled_test['matchType']=='flaretpp']
test_normal_duo=scaled_test[scaled_test['matchType']=='normal-duo']
test_normal_duo_fpp=scaled_test[scaled_test['matchType']=='normal-duo-fpp']
test_normal_solo=scaled_test[scaled_test['matchType']=='normal-solo']
test_normal_solo_fpp=scaled_test[scaled_test['matchType']=='normal-solo-fpp']
test_normal_squad=scaled_test[scaled_test['matchType']=='normal-squad']
test_normal_squad_fpp=scaled_test[scaled_test['matchType']=='normal-squad-fpp']
test_solo=scaled_test[scaled_test['matchType']=='solo']
test_solo_fpp=scaled_test[scaled_test['matchType']=='solo-fpp']
test_squad=scaled_test[scaled_test['matchType']=='squad']
test_squad_fpp=scaled_test[scaled_test['matchType']=='squad-fpp']
```

[28]:
```
#Add all match types to a list
train_allTypes =␣
 ↪[train_crashfpp,train_crashtpp,train_duo,train_duo_fpp,train_flarefpp,train_flaretpp,train_
test_allTypes =␣
 ↪[test_crashfpp,test_crashtpp,test_duo,test_duo_fpp,test_flarefpp,test_flaretpp,test_normal_
```

With "Kaggle winner trick"

[31]:
```
num_test = scaled_test.shape[0]
sum_error = 0
set_3 = np.array([])
for index in range(len(train_allTypes)):
```

```
    temp_train = train_allTypes[index]
    temp_test = test_allTypes[index]

    temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
    temp_train = temp_train.groupby("groupId").mean()
    temp_train_y = temp_train["winPlacePerc"]
    temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

    temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
    temp_test = temp_test.groupby("groupId").transform('mean')
    temp_test_y = temp_test["winPlacePerc"]
    temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

    temp_parameter = SGDhyperParameterTuning(temp_train_X,temp_train_y)
    temp_SGD = SGDRegressor().set_params(**temp_parameter)
    temp_SGD.fit(temp_train_X,temp_train_y)
    sum_error += mean_absolute_error(temp_SGD.
 ↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]
    set_3 = np.append(set_3,np.absolute(temp_SGD.
 ↪predict(temp_test_X)-temp_test_y))

print(sum_error/num_test)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.16 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished


 Time taken: 0 hours 0 minutes and 0.09 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    5.4s finished


 Time taken: 0 hours 0 minutes and 5.8 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   16.0s finished


 Time taken: 0 hours 0 minutes and 16.87 seconds.
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.08 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


 Time taken: 0 hours 0 minutes and 0.15 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.07 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


 Time taken: 0 hours 0 minutes and 0.16 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.09 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


 Time taken: 0 hours 0 minutes and 0.16 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.07 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.1s finished


 Time taken: 0 hours 0 minutes and 0.22 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    5.5s finished
```

```
 Time taken: 0 hours 0 minutes and 5.83 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   17.8s finished


 Time taken: 0 hours 0 minutes and 18.77 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    6.4s finished


 Time taken: 0 hours 0 minutes and 6.81 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   16.4s finished


 Time taken: 0 hours 0 minutes and 17.55 seconds.
0.07301519409102036
```

[29]:

```
num_test = scaled_test.shape[0]
sum_error = 0
set_3 = np.array([])
for index in range(len(train_allTypes)):

    temp_train = train_allTypes[index]
    temp_test = test_allTypes[index]

    temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
    temp_train = temp_train.groupby("groupId").mean()
    temp_train_y = temp_train["winPlacePerc"]
    temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

    temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
    temp_test = temp_test.groupby("groupId").transform('mean')
    temp_test_y = temp_test["winPlacePerc"]
    temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

    temp_parameter =␣
↪custom_SGDhyperParameterTuning(temp_train_X,temp_train_y,False)
    temp_SGD = SGDRegressor().set_params(**temp_parameter)
    temp_SGD.fit(temp_train_X,temp_train_y)
    sum_error += mean_absolute_error(temp_SGD.
↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]
```

```
    set_3 = np.append(set_3,np.absolute(temp_SGD.
 ↪predict(temp_test_X)-temp_test_y))

print(sum_error/num_test)
```

```
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 0.12550827781754817
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.11014491622951106
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 0.11425225641824707
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 0.09604298514187476
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 0.09785653424900116
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.12578855972164346
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.12563134601874631
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 0.09951963311101121
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 0.11957974155888045
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.11312899403266224

 Time taken: 0 hours 0 minutes and 0.06 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.10188624284037465
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 0.11744515363762074
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
```

```
0.0001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 0.10471944493784033
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.09124060100717568
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 0.09834023808391994
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.10201004320505083
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.10177413931284462
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 0.0946921549588893
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.09262469471465803
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 0.08818484555416556

 Time taken: 0 hours 0 minutes and 0.03 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.07375798252520904
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.09949212418956413
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.07625581747945646
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 0.07512477020482931
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.0973877557512233
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.07655225212505547
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.09915324312046066
```

```
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.10453707403295288
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.09918460127062224
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.10450925449111247

 Time taken: 0 hours 0 minutes and 1.85 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 0.06820075832005246
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.07015603985870644
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.08450315008291624
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.08453251357731473
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 0.07054635091273605
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.06748495905455475
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.06732725470109674
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.06883181255519377
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 0.06794933108982591
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.10450120948185668

 Time taken: 0 hours 0 minutes and 5.93 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
```

```
0.0001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 563.7302024854022
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.13989399599409916
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 5064.881099995718
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.1259129572667202
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.14039841732055822
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.12739399414206162
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.12623701995688735
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.13886851122950697
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.1410517399136544
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.13882617268959788

 Time taken: 0 hours 0 minutes and 0.03 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 0.15037969516403227
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.12697066317935451
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 8049.741901722922
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.12573492653227475
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 307900.73194874014
```

```
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.12633556368647883
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 58.71313447196737
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.12511907905044953
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.10891684210472896
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.10895759680487394

 Time taken: 0 hours 0 minutes and 0.06 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.14829829656254037
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 1052415383087.8044
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.12185781988610134
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 2141087161056.96
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.1909501771946782
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.1403152118034726
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.1440578210195537
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 2194767895988.4297
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.18714903310070838
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.1}
```

```
 MAE: 483798700507.24524

 Time taken: 0 hours 0 minutes and 0.03 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.1595353364995421
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 43361563840.995346
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 28506245373.82782
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 45603781449.090416
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 75897171923.24994
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 263704220155.59848
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.16019802280578
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 924647566839.7064
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 45314441386.65656
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 622847554295.5321

 Time taken: 0 hours 0 minutes and 0.09 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.25161665992710275
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.21039651393456033
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.2071457518570384
```

```
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.21209951259129348
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.1773505026094042
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.21247351263336423
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.20792535399049114
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.2040313345519819
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 5705072104868.84
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.20950674593650356

 Time taken: 0 hours 0 minutes and 0.02 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.331072075736684
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.2148263366574435
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 783533951760.4512
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 333853411271.69305
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 1642031060712.8384
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.2142811767159906
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.4091260944747794
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.1}
```

```
 MAE: 343119944313.88965
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.20898864993255906
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.2187115837464737

 Time taken: 0 hours 0 minutes and 0.12 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.2965583812283459
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.2854054633655833
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.3013751882834923
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.3129778306804721
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 1742095578892.6108
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.29683427843185195
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.3153984240420026
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.29610563666983547
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.3015213340036005
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.29477910570258425

 Time taken: 0 hours 0 minutes and 0.03 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 258997610.21932566
```

```
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 9527377.050930187
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 920093732.3932134
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.19513698269218469
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 0.20692609229761308
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.19530679159138797
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 227915585.30519617
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 2756788534.2476325
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 426816048.7988914
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 0.17795056791763228

 Time taken: 0 hours 0 minutes and 0.3 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 0.1071856439523348
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.07974796435784172
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.10218895476546652
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.09925361003101948
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.07849826531854043
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.01}
```

```
 MAE: 0.07639390509009869
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.09256900040854413
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.09826189116377977
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.09990306441861409
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 0.08154014022319721

 Time taken: 0 hours 0 minutes and 2.28 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.06947547839663754
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.08427162307816673
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.08251268531090925
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.06834279761013334
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.08365108165272359
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.08657768771379612
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 0.07147850071459468
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 0.08826251814313629
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.08263393848240082
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 0.10820611064430487
```

```
 Time taken: 0 hours 0 minutes and 6.68 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.08518012371530873
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.10409402522667204
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.1}
 MAE: 0.0987074604093258
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 0.10330159782730383
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 5000, 'eta0': 0.01}
 MAE: 0.10345495674976386
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 0.08444764732399516
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.1}
 MAE: 0.08790230659694447
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.15, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.0849167085864345
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.10488171265027672
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.01}
 MAE: 0.08683936564599523

 Time taken: 0 hours 0 minutes and 2.23 seconds.
Start Splitting…
Start validation…
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.35, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.10266816814888201
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.1}
 MAE: 0.07938131109357008
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.15, 'max_iter': 1000, 'eta0': 0.001}
 MAE: 0.09837900425239861
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.15, 'max_iter': 5000, 'eta0': 0.001}
```

```
 MAE: 0.0911052886400083
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.01, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.0989514842893563
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.09062410143206014
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.07708898356239115
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 1000, 'eta0': 0.01}
 MAE: 0.07744865242133825
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.0001, 'l1_ratio': 0.55, 'max_iter': 2000, 'eta0': 0.001}
 MAE: 0.09062627773897024
Start training… {'loss': 'squared_loss', 'penalty': 'elasticnet', 'alpha':
0.001, 'l1_ratio': 0.35, 'max_iter': 5000, 'eta0': 0.001}
 MAE: 0.09178719245403079

 Time taken: 0 hours 0 minutes and 6.37 seconds.
0.07705889630682686
```

[37]:
```
stats.ttest_ind(set_2,set_3)
```

[37]:
```
Ttest_indResult(statistic=18.40340519912551, pvalue=1.253768399802096e-75)
```

17.1.1 Different model selection for approach 2

According to the the counts below:

Use XGBoost for duo, duo-fpp, normal-duo-fpp, solo, solo-fpp, squad, squad-fpp.

Use SGDRegressor for normal-squad-fpp

Use SVR(kernel = 'linear') for crashfpp, crashtpp, flarefpp, flaretpp, normal-duo, normal-duo-fpp, normal-solo, normal-solo-fpp, normal-squad

(As suggested by scikit-learn cheat sheet)

https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

[24]:
```
pd.DataFrame({'counts': scaled_train.groupby(('matchType')).size(),
                              'Percentage': scaled_train.groupby(('matchType')).
 ↪size() / len(train_data)})
```

[24]:

matchType	counts	Percentage
crashfpp	4972	0.001118
crashtpp	282	0.000063
duo	251120	0.056470

```
duo-fpp             796987    0.179220
flarefpp               516    0.000116
flaretpp              2041    0.000459
normal-duo             163    0.000037
normal-duo-fpp        4428    0.000996
normal-solo            243    0.000055
normal-solo-fpp       1336    0.000300
normal-squad           398    0.000089
normal-squad-fpp     13959    0.003139
solo                146105    0.032855
solo-fpp            429670    0.096621
squad               500754    0.112606
squad-fpp          1403320    0.315568
```

[38]:
```
#Extract each matchType
train_crashfpp=scaled_train[scaled_train['matchType']=='crashfpp']
train_crashtpp=scaled_train[scaled_train['matchType']=='crashtpp']
train_duo=scaled_train[scaled_train['matchType']=='duo']
train_duo_fpp=scaled_train[scaled_train['matchType']=='duo-fpp']
train_flarefpp=scaled_train[scaled_train['matchType']=='flarefpp']
train_flaretpp=scaled_train[scaled_train['matchType']=='flaretpp']
train_normal_duo=scaled_train[scaled_train['matchType']=='normal-duo']
train_normal_duo_fpp=scaled_train[scaled_train['matchType']=='normal-duo-fpp']
train_normal_solo=scaled_train[scaled_train['matchType']=='normal-solo']
train_normal_solo_fpp=scaled_train[scaled_train['matchType']=='normal-solo-fpp']
train_normal_squad=scaled_train[scaled_train['matchType']=='normal-squad']
train_normal_squad_fpp=scaled_train[scaled_train['matchType']=='normal-squad-fpp']
train_solo=scaled_train[scaled_train['matchType']=='solo']
train_solo_fpp=scaled_train[scaled_train['matchType']=='solo-fpp']
train_squad=scaled_train[scaled_train['matchType']=='squad']
train_squad_fpp=scaled_train[scaled_train['matchType']=='squad-fpp']
```

[39]:
```
#Extract each matchType
test_crashfpp=scaled_test[scaled_test['matchType']=='crashfpp']
test_crashtpp=scaled_test[scaled_test['matchType']=='crashtpp']
test_duo=scaled_test[scaled_test['matchType']=='duo']
test_duo_fpp=scaled_test[scaled_test['matchType']=='duo-fpp']
test_flarefpp=scaled_test[scaled_test['matchType']=='flarefpp']
test_flaretpp=scaled_test[scaled_test['matchType']=='flaretpp']
test_normal_duo=scaled_test[scaled_test['matchType']=='normal-duo']
test_normal_duo_fpp=scaled_test[scaled_test['matchType']=='normal-duo-fpp']
test_normal_solo=scaled_test[scaled_test['matchType']=='normal-solo']
test_normal_solo_fpp=scaled_test[scaled_test['matchType']=='normal-solo-fpp']
test_normal_squad=scaled_test[scaled_test['matchType']=='normal-squad']
test_normal_squad_fpp=scaled_test[scaled_test['matchType']=='normal-squad-fpp']
test_solo=scaled_test[scaled_test['matchType']=='solo']
test_solo_fpp=scaled_test[scaled_test['matchType']=='solo-fpp']
```

```
test_squad=scaled_test[scaled_test['matchType']=='squad']
test_squad_fpp=scaled_test[scaled_test['matchType']=='squad-fpp']
```

[40]:
```
train_xgb_allTypes =␣
↪[train_duo,train_duo_fpp,train_normal_duo_fpp,train_solo,train_solo_fpp,train_squad,train_s
test_xgb_allTypes =␣
↪[test_duo,test_duo_fpp,test_normal_duo_fpp,test_solo,test_solo_fpp,test_squad,test_squad_fp
```

[41]:
```
train_sgd_allTypes = [train_normal_squad_fpp]
test_sgd_allTypes = [test_normal_squad_fpp]
```

[42]:
```
train_svr_allTypes =␣
↪[train_crashfpp,train_crashtpp,train_flarefpp,train_flaretpp,train_normal_duo,train_normal_
test_svr_allTypes =␣
↪[test_crashfpp,test_crashtpp,test_flarefpp,test_flaretpp,test_normal_duo,test_normal_duo_fp
```

[38]:
```
num_test = scaled_test.shape[0]
err_SGD = 0
err_sgd = 0
err_svr = 0

#SGD
for index in range(len(train_SGD_allTypes)):
    temp_train = train_SGD_allTypes[index]
    temp_test = test_SGD_allTypes[index]

    temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
    temp_train = temp_train.groupby("groupId").mean()
    temp_train_y = temp_train["winPlacePerc"]
    temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

    temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
    temp_test = temp_test.groupby("groupId").transform('mean')
    temp_test_y = temp_test["winPlacePerc"]
    temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

    temp_SGD = SGDRegressor(objective="reg:squarederror")
    temp_SGD.fit(temp_train_X,temp_train_y)
    err_SGD += mean_absolute_error(temp_SGD.
↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]

#SGD
temp_train = train_sgd_allTypes[0]
temp_test = test_sgd_allTypes[0]

temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
temp_train = temp_train.groupby("groupId").mean()
```

```
temp_train_y = temp_train["winPlacePerc"]
temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
temp_test = temp_test.groupby("groupId").transform('mean')
temp_test_y = temp_test["winPlacePerc"]
temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

temp_sgd = SGDRegressor()
temp_sgd.fit(temp_train_X,temp_train_y)
err_sgd += mean_absolute_error(temp_sgd.
 ↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]

#SVR
for index in range(len(train_svr_allTypes)):
    temp_train = train_svr_allTypes[index]
    temp_test = test_svr_allTypes[index]

    temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
    temp_train = temp_train.groupby("groupId").mean()
    temp_train_y = temp_train["winPlacePerc"]
    temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

    temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
    temp_test = temp_test.groupby("groupId").transform('mean')
    temp_test_y = temp_test["winPlacePerc"]
    temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

    temp_svr = LinearSVR(loss = 'epsilon_insensitive') #L1 loss
    temp_svr.fit(temp_train_X,temp_train_y)
    err_svr += mean_absolute_error(temp_svr.
 ↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]
(err_SGD+err_sgd+err_svr)/num_test
```

[38]:
```
0.05858968283308794
```

17.2 Approach 3

Train on different cluster, using kaggle winner trick.

[46]:
```
def approach_3(train,test,n_clusters,kmeans,first_kt):
    train = train.copy()
    test = test.copy()

    if first_kt:
        train_d = train.drop(['Id','matchId',"matchType"],axis=1)
        train_d = train_d.groupby("groupId").transform('mean')
```

```
        train_y = train_d["winPlacePerc"]
        train_X = train_d.drop(["winPlacePerc"],axis=1)

        test_d = test.drop(['Id','matchId',"matchType"],axis=1)
        test_d = test_d.groupby("groupId").transform('mean')
        test_y = test_d["winPlacePerc"]
        test_X = test_d.drop(["winPlacePerc"],axis=1)
    else:
        train_d = train.drop(['Id','matchId','groupId',"matchType"],axis=1)
        train_y = train_d["winPlacePerc"]
        train_X = train_d.drop(["winPlacePerc"],axis=1)

        test_d = test.drop(['Id','matchId','groupId',"matchType"],axis=1)
        test_y = test_d["winPlacePerc"]
        test_X = test_d.drop(["winPlacePerc"],axis=1)

    kmeans.n_clusters = n_clusters
    kmeans.fit(train_X)

    train["cluster"] = kmeans.labels_
    test["cluster"] = kmeans.predict(test_X)

    train_n = train.drop(['Id','matchId','matchType'],axis=1)
    test_n = test.drop(['Id','matchId','matchType'],axis=1)

    train_n["matchType"] = train["matchType"]
    test_n["matchType"] = test["matchType"]
    train_n = pd.get_dummies(data = train_n,columns=["matchType"])
    test_n = pd.get_dummies(data = test_n,columns=["matchType"])

    num_test = test.shape[0]
    sum_error = 0
    temp_set = np.array([])
    for i in range(n_clusters):

        temp_train_n = train_n[train_n["cluster"] == i]
        temp_test_n = test_n[test_n["cluster"] == i]

        temp_train_n.drop(["cluster"],axis=1)
        temp_test_n.drop(["cluster"],axis=1)

        temp_train_n = temp_train_n.groupby("groupId").mean()
        temp_train_n_y = temp_train_n["winPlacePerc"]
        temp_train_n_X = temp_train_n.drop(["winPlacePerc"],axis=1)

        temp_test_n = temp_test_n.groupby("groupId").transform('mean')
        temp_test_n_y = temp_test_n["winPlacePerc"]
```

```
        temp_test_n_X = temp_test_n.drop(["winPlacePerc"],axis=1)

        temp_parameter = SGDhyperParameterTuning(temp_train_n_X,temp_train_n_y)
        temp_SGD = SGDRegressor().set_params(**temp_parameter)
        temp_SGD.fit(temp_train_n_X,temp_train_n_y)
        sum_error += mean_absolute_error(temp_SGD.
↪predict(temp_test_n_X),temp_test_n_y)*temp_test_n.shape[0]
        temp_set = np.append(temp_set,np.absolute(temp_SGD.
↪predict(temp_test_n_X)-temp_test_n_y))

    return sum_error/num_test,temp_set
```

[47]:
```
kmeans = KMeans(random_state=10,n_init = 10)
```

kaggle trick applied before clustering

[48]:
```
mae,set_4 = approach_3(scaled_train,scaled_test,2,kmeans,True)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  1.0min finished


 Time taken: 0 hours 1 minutes and 5.4 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  1.4min finished


 Time taken: 0 hours 1 minutes and 23.97 seconds.
```

[49]:
```
mae
```

[49]:
```
0.06897970157366241
```

[50]:
```
stats.ttest_ind(set_2,set_4)
```

[50]:
```
Ttest_indResult(statistic=61.123410569539715, pvalue=0.0)
```

[51]:
```
stats.ttest_ind(set_3,set_4)
```

[51]:
```
Ttest_indResult(statistic=42.98734545499313, pvalue=0.0)
```

[65]:
```
mae,set_4 = approach_3(scaled_train,scaled_test,2,kmeans,True)
```

```
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.045357470843814515
Fit  2  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 7,
```

```
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04519997577054853
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.044902192334544073
Fit  4  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04539555931243005
Fit  5  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04719145814294863
Fit  6  features:  {'learning_rate': 0.5, 'max_depth': 5, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.04500963380585551
Fit  7  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.04612848172795308
Fit  8  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.5, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.07643639862859815
Fit  9  features:  {'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.0529023563382154
Fit  10  features:  {'learning_rate': 0.5, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.5, 'n_estimators': 500, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.047008666205741095

 Time taken: 0 hours 2 minutes and 56.5 seconds.
Fit  1  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.061696271246890255
Fit  2  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 7,
'colsample_bytree': 0.7, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06339182831717698
Fit  3  features:  {'learning_rate': 0.5, 'max_depth': 3, 'min_child_weight': 5,
'colsample_bytree': 0.7, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06232799592958938
Fit  4  features:  {'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight':
5, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.08216897322260658
Fit  5  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
```

```
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06460549031234435
Fit  6  features:  {'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 100, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.07074631459421582
Fit  7  features:  {'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 7,
'colsample_bytree': 0.5, 'n_estimators': 200, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06376048313930724
Fit  8  features:  {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight':
1, 'colsample_bytree': 0.7, 'n_estimators': 200, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.061073377646533505
Fit  9  features:  {'learning_rate': 0.5, 'max_depth': 7, 'min_child_weight': 1,
'colsample_bytree': 0.5, 'n_estimators': 500, 'objective': 'reg:squarederror',
'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE: 0.06434621843649745
Fit  10  features:  {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight':
7, 'colsample_bytree': 0.7, 'n_estimators': 100, 'objective':
'reg:squarederror', 'tree_method': 'gpu_hist', 'gpu_id': 0}  MAE:
0.06339182831717698

 Time taken: 0 hours 1 minutes and 0.41 seconds.
```

[66]:
```
mae
```

[66]:
```
0.05051294522035498
```

[67]:
```
stats.ttest_ind(set_2,set_4)
```

[67]:
```
Ttest_indResult(statistic=-20.101950969967373, pvalue=7.260149002223263e-90)
```

[68]:
```
stats.ttest_ind(set_3,set_4)
```

[68]:
```
Ttest_indResult(statistic=-15.200651427583807, pvalue=3.5277775764109265e-52)
```

17.2.1 Train on the clusters within each matchType

[52]:
```
train_two =␣
 ↪[train_crashfpp,train_crashtpp,train_duo,train_duo_fpp,train_flarefpp,train_flaretpp,train_
test_two =␣
 ↪[test_crashfpp,test_crashtpp,test_duo,test_duo_fpp,test_flarefpp,test_flaretpp,test_normal_
```

[53]:
```
train_three = [train_normal_solo,train_normal_solo_fpp]
test_three = [test_normal_solo,test_normal_solo_fpp]
```

[54]:
```
kmeans = KMeans(random_state=10,n_init = 10)
```

[55]:
```
num = scaled_test.shape[0]
sum_err = 0
set_5 = np.array([])
```

```
for i in range(len(train_two)):

    train_single = train_two[i]
    test_single = test_two[i]
    mae,temp_set = approach_3(train_single,test_single,2,kmeans,True)
    sum_err += test_single.shape[0]*mae
    set_5 = np.append(set_5,temp_set)

for i in range(len(train_three)):

    train_single = train_three[i]
    test_single = test_three[i]
    mae,temp_set = approach_3(train_single,test_single,3,kmeans,True)
    sum_err += test_single.shape[0]*mae
    set_5 = np.append(set_5,temp_set)

print(sum_err/num)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


 Time taken: 0 hours 0 minutes and 0.69 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.12 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished


 Time taken: 0 hours 0 minutes and 0.09 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.09 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    3.7s finished


 Time taken: 0 hours 0 minutes and 4.02 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  1.1min finished


 Time taken: 0 hours 1 minutes and 7.77 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  6.7min finished


 Time taken: 0 hours 6 minutes and 41.5 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   12.0s finished


 Time taken: 0 hours 0 minutes and 12.6 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.09 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.07 seconds.

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished

Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.1 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


 Time taken: 0 hours 0 minutes and 0.11 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.08 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

```
 Time taken: 0 hours 0 minutes and 0.08 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


 Time taken: 0 hours 0 minutes and 0.36 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.1 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished


 Time taken: 0 hours 0 minutes and 0.09 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.08 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    2.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished


 Time taken: 0 hours 0 minutes and 2.7 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.14 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    5.0s finished


 Time taken: 0 hours 0 minutes and 5.31 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.9s finished


 Time taken: 0 hours 0 minutes and 1.03 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   13.8s finished


 Time taken: 0 hours 0 minutes and 14.65 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    7.4s finished


 Time taken: 0 hours 0 minutes and 7.66 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    4.3s finished


 Time taken: 0 hours 0 minutes and 4.65 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    5.7s finished


 Time taken: 0 hours 0 minutes and 5.88 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   11.6s finished


 Time taken: 0 hours 0 minutes and 12.18 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  3.5min finished


 Time taken: 0 hours 3 minutes and 29.49 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits


 Time taken: 0 hours 0 minutes and 0.09 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished


 Time taken: 0 hours 0 minutes and 0.09 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits
```

```
 Time taken: 0 hours 0 minutes and 0.09 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


 Time taken: 0 hours 0 minutes and 0.33 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.09 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.12 seconds.
29114313.90931671
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
```

[56]:
```
stats.ttest_ind(set_2,set_5)
```

[56]:
```
Ttest_indResult(statistic=-3.3054336909266886, pvalue=0.0009483141176813204)
```

[57]:
```
stats.ttest_ind(set_3,set_5)
```

[57]:
```
Ttest_indResult(statistic=-3.305433691129948, pvalue=0.0009483141169935094)
```

18 with threshold

[58]:
```
train_two =␣
 ↪[train_crashfpp,train_duo,train_duo_fpp,train_flaretpp,train_normal_duo_fpp,train_normal_sq
test_two =␣
 ↪[test_crashfpp,test_duo,test_duo_fpp,test_flaretpp,test_normal_duo_fpp,test_normal_squad_fp
```

[59]:
```
train_three = [train_normal_solo_fpp]
test_three = [test_normal_solo_fpp]
```

[60]:
```
train_dont =␣
 ↪[train_crashtpp,train_flarefpp,train_normal_duo,train_normal_solo,train_normal_squad]
test_dont =␣
 ↪[test_crashtpp,test_flarefpp,test_normal_duo,test_normal_solo,test_normal_squad]
```

[61]:
```
kmeans = KMeans(random_state=10,n_init = 10)
```

[62]:
```
num = scaled_test.shape[0]
sum_err = 0
set_6 = np.array([])
for i in range(len(train_two)):

    train_single = train_two[i]
    test_single = test_two[i]
    mae,temp_set = approach_3(train_single,test_single,2,kmeans,True)
    sum_err += test_single.shape[0]*mae
    set_6 = np.append(set_6,temp_set)

for i in range(len(train_three)):

    train_single = train_three[i]
    test_single = test_three[i]
    mae,temp_set = approach_3(train_single,test_single,3,kmeans,True)
    sum_err += test_single.shape[0]*mae
    set_6 = np.append(set_6,temp_set)

for i in range(len(train_dont)):

    temp_train = train_dont[i]
    temp_test = test_dont[i]

    temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
    temp_train = temp_train.groupby("groupId").mean()
    temp_train_y = temp_train["winPlacePerc"]
    temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

    temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
    temp_test = temp_test.groupby("groupId").transform('mean')
    temp_test_y = temp_test["winPlacePerc"]
    temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

    temp_parameter = SGDhyperParameterTuning(temp_train_X,temp_train_y)
    temp_SGD = SGDRegressor().set_params(**temp_parameter)
    temp_SGD.fit(temp_train_X,temp_train_y)
    sum_err += mean_absolute_error(temp_SGD.
 ↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]
    set_6 = np.append(set_6,np.absolute(temp_SGD.
 ↪predict(temp_test_X)-temp_test_y))

print(sum_err/num)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.4s finished
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished


 Time taken: 0 hours 0 minutes and 0.5 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.12 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    3.4s finished


 Time taken: 0 hours 0 minutes and 3.67 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   59.9s finished


 Time taken: 0 hours 1 minutes and 0.03 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  1.6min finished


 Time taken: 0 hours 1 minutes and 33.85 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   12.0s finished


 Time taken: 0 hours 0 minutes and 12.71 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.11 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


 Time taken: 0 hours 0 minutes and 0.1 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.7s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
```

```
 Time taken: 0 hours 0 minutes and 0.83 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.11 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.1s finished

 Time taken: 0 hours 0 minutes and 0.19 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.15 seconds.

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished

Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    4.1s finished

 Time taken: 0 hours 0 minutes and 4.38 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    4.4s finished

 Time taken: 0 hours 0 minutes and 4.58 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   13.7s finished

 Time taken: 0 hours 0 minutes and 14.5 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:  6.6min finished

 Time taken: 0 hours 6 minutes and 36.76 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    3.6s finished

 Time taken: 0 hours 0 minutes and 3.88 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    1.4s finished


 Time taken: 0 hours 0 minutes and 1.61 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   10.7s finished


 Time taken: 0 hours 0 minutes and 11.33 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:   42.9s finished


 Time taken: 0 hours 0 minutes and 43.18 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


 Time taken: 0 hours 0 minutes and 0.36 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.08 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


 Time taken: 0 hours 0 minutes and 0.12 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.08 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
```

```
 Time taken: 0 hours 0 minutes and 0.08 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.08 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.09 seconds.
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 0.08 seconds.
0.06849072298530477

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s finished
```

[63]:
```
stats.ttest_ind(set_2,set_6)
```

[63]:
```
Ttest_indResult(statistic=66.33863966854169, pvalue=0.0)
```

[64]:
```
stats.ttest_ind(set_3,set_6)
```

[64]:
```
Ttest_indResult(statistic=48.25980090395253, pvalue=0.0)
```

[65]:
```
stats.ttest_ind(set_5,set_6)
```

[65]:
```
Ttest_indResult(statistic=3.3054336916436244, pvalue=0.0009483141152552755)
```

[66]:
```
stats.ttest_ind(set_4,set_6)
```

[66]:
```
Ttest_indResult(statistic=5.33195223259157, pvalue=9.717428680824867e-08)
```

[ ]:

[ ]:

[ ]:

[ ]:

19 SGDRegressor

19.1 Approach 1

Train on all data

[16]:
```
scaled_train_1 = scaled_train.drop(['Id','groupId','matchId'],axis=1)
scaled_train_1 = pd.get_dummies(data = scaled_train_1,columns=["matchType"])
scaled_train_1_y = scaled_train_1["winPlacePerc"]
```

```
scaled_train_1_X = scaled_train_1.drop(["winPlacePerc"],axis=1)
```

[17]:
```
scaled_test_1 = scaled_test.drop(['Id','groupId','matchId'],axis=1)
scaled_test_1 = pd.get_dummies(data = scaled_test_1,columns=["matchType"])
scaled_test_1_y = scaled_test_1["winPlacePerc"]
scaled_test_1_X = scaled_test_1.drop(["winPlacePerc"],axis=1)
```

[7]:
```
sgd_1 = SGDRegressor(penalty='elasticnet')
```

[102]:
```
sgd_1.fit(scaled_train_1_X,scaled_train_1_y)
```

[102]:
```
SGDRegressor(penalty='elasticnet')
```

[103]:
```
mean_absolute_error(sgd_1.predict(scaled_test_1_X),scaled_test_1_y)
```

[103]:
```
0.09028851144915745
```

19.2 Approach 1

Train on all data - "Kaggle winner trick"

[11]:
```
scaled_train_1_k = scaled_train.drop(['Id','matchId'],axis=1)
scaled_train_1_k = pd.get_dummies(data = scaled_train_1_k,columns=["matchType"])
scaled_train_1_k = scaled_train_1_k.groupby("groupId").mean()
scaled_train_1_k_y = scaled_train_1_k["winPlacePerc"]
scaled_train_1_k_X = scaled_train_1_k.drop(["winPlacePerc"],axis=1)
```

[12]:
```
scaled_test_1_k = scaled_test.drop(['Id','matchId'],axis=1)
scaled_test_1_k = pd.get_dummies(data = scaled_test_1_k,columns=["matchType"])
scaled_test_1_k = scaled_test_1_k.groupby("groupId").transform('mean')
scaled_test_1_k_y = scaled_test_1_k["winPlacePerc"]
scaled_test_1_k_X = scaled_test_1_k.drop(["winPlacePerc"],axis=1)
```

[82]:
```
sgd_1_k = SGDRegressor(penalty='elasticnet')
```

[83]:
```
sgd_1_k.fit(scaled_train_1_k_X,scaled_train_1_k_y)
```

[83]:
```
SGDRegressor(penalty='elasticnet')
```

[84]:
```
mean_absolute_error(sgd_1_k.predict(scaled_test_1_k_X),scaled_test_1_k_y)
```

[84]:
```
0.07469108180015249
```

19.3 Approach 2

Train on each match type - with "Kaggle winner trick"

[91]:
```
#Extract each matchType
train_crashfpp=scaled_train[scaled_train['matchType']=='crashfpp']
```

```
train_crashtpp=scaled_train[scaled_train['matchType']=='crashtpp']
train_duo=scaled_train[scaled_train['matchType']=='duo']
train_duo_fpp=scaled_train[scaled_train['matchType']=='duo-fpp']
train_flarefpp=scaled_train[scaled_train['matchType']=='flarefpp']
train_flaretpp=scaled_train[scaled_train['matchType']=='flaretpp']
train_normal_duo=scaled_train[scaled_train['matchType']=='normal-duo']
train_normal_duo_fpp=scaled_train[scaled_train['matchType']=='normal-duo-fpp']
train_normal_solo=scaled_train[scaled_train['matchType']=='normal-solo']
train_normal_solo_fpp=scaled_train[scaled_train['matchType']=='normal-solo-fpp']
train_normal_squad=scaled_train[scaled_train['matchType']=='normal-squad']
train_normal_squad_fpp=scaled_train[scaled_train['matchType']=='normal-squad-fpp']
train_solo=scaled_train[scaled_train['matchType']=='solo']
train_solo_fpp=scaled_train[scaled_train['matchType']=='solo-fpp']
train_squad=scaled_train[scaled_train['matchType']=='squad']
train_squad_fpp=scaled_train[scaled_train['matchType']=='squad-fpp']
```

[92]:
```
#Extract each matchType
test_crashfpp=scaled_test[scaled_test['matchType']=='crashfpp']
test_crashtpp=scaled_test[scaled_test['matchType']=='crashtpp']
test_duo=scaled_test[scaled_test['matchType']=='duo']
test_duo_fpp=scaled_test[scaled_test['matchType']=='duo-fpp']
test_flarefpp=scaled_test[scaled_test['matchType']=='flarefpp']
test_flaretpp=scaled_test[scaled_test['matchType']=='flaretpp']
test_normal_duo=scaled_test[scaled_test['matchType']=='normal-duo']
test_normal_duo_fpp=scaled_test[scaled_test['matchType']=='normal-duo-fpp']
test_normal_solo=scaled_test[scaled_test['matchType']=='normal-solo']
test_normal_solo_fpp=scaled_test[scaled_test['matchType']=='normal-solo-fpp']
test_normal_squad=scaled_test[scaled_test['matchType']=='normal-squad']
test_normal_squad_fpp=scaled_test[scaled_test['matchType']=='normal-squad-fpp']
test_solo=scaled_test[scaled_test['matchType']=='solo']
test_solo_fpp=scaled_test[scaled_test['matchType']=='solo-fpp']
test_squad=scaled_test[scaled_test['matchType']=='squad']
test_squad_fpp=scaled_test[scaled_test['matchType']=='squad-fpp']
```

[93]:
```
#Add all match types to a list
train_allTypes =␣
 ↪[train_crashfpp,train_crashtpp,train_duo,train_duo_fpp,train_flarefpp,train_flaretpp,train_
test_allTypes =␣
 ↪[test_crashfpp,test_crashtpp,test_duo,test_duo_fpp,test_flarefpp,test_flaretpp,test_normal_
```

With "Kaggle winner trick"

[54]:
```
num_test = scaled_test.shape[0]
sum_error = 0
for index in range(len(train_allTypes)):

    temp_train = train_allTypes[index]
    temp_test = test_allTypes[index]
```

```
    temp_train = temp_train.drop(['Id','matchId','matchType'],axis=1)
    temp_train = temp_train.groupby("groupId").mean()
    temp_train_y = temp_train["winPlacePerc"]
    temp_train_X = temp_train.drop(["winPlacePerc"],axis=1)

    temp_test = temp_test.drop(['Id','matchId','matchType'],axis=1)
    temp_test = temp_test.groupby("groupId").transform('mean')
    temp_test_y = temp_test["winPlacePerc"]
    temp_test_X = temp_test.drop(["winPlacePerc"],axis=1)

    temp_sgd = SGDRegressor(penalty='elasticnet')
    temp_sgd.fit(temp_train_X,temp_train_y)
    sum_error += mean_absolute_error(temp_sgd.
 ↪predict(temp_test_X),temp_test_y)*temp_test.shape[0]

print(sum_error/num_test)
```

```
0.07235442030928865
```

19.4 Approach 3

Train on different cluster, using kaggle winner trick.

[157]:
```
def approach_3(train,test,n_clusters,kmeans,first_kt):
    train = train.copy()
    test = test.copy()

    if first_kt:
        train_d = train.drop(['Id','matchId',"matchType"],axis=1)
        train_d = train_d.groupby("groupId").transform('mean')
        train_y = train_d["winPlacePerc"]
        train_X = train_d.drop(["winPlacePerc"],axis=1)

        test_d = test.drop(['Id','matchId',"matchType"],axis=1)
        test_d = test_d.groupby("groupId").transform('mean')
        test_y = test_d["winPlacePerc"]
        test_X = test_d.drop(["winPlacePerc"],axis=1)
    else:
        train_d = train.drop(['Id','matchId','groupId',"matchType"],axis=1)
        train_y = train_d["winPlacePerc"]
        train_X = train_d.drop(["winPlacePerc"],axis=1)

        test_d = test.drop(['Id','matchId','groupId',"matchType"],axis=1)
        test_y = test_d["winPlacePerc"]
        test_X = test_d.drop(["winPlacePerc"],axis=1)

    kmeans.n_clusters = n_clusters
```

```
    kmeans.fit(train_X)

    train["cluster"] = kmeans.labels_
    test["cluster"] = kmeans.predict(test_X)

    train_n = train.drop(['Id','matchId','matchType'],axis=1)
    test_n = test.drop(['Id','matchId','matchType'],axis=1)

    train_n["matchType"] = train["matchType"]
    test_n["matchType"] = test["matchType"]
    train_n = pd.get_dummies(data = train_n,columns=["matchType"])
    test_n = pd.get_dummies(data = test_n,columns=["matchType"])

    num_test = test.shape[0]
    sum_error = 0

    for i in range(n_clusters):

        temp_train_n = train_n[train_n["cluster"] == i]
        temp_test_n = test_n[test_n["cluster"] == i]

        temp_train_n.drop(["cluster"],axis=1)
        temp_test_n.drop(["cluster"],axis=1)

        temp_train_n = temp_train_n.groupby("groupId").mean()
        temp_train_n_y = temp_train_n["winPlacePerc"]
        temp_train_n_X = temp_train_n.drop(["winPlacePerc"],axis=1)

        temp_test_n = temp_test_n.groupby("groupId").transform('mean')
        temp_test_n_y = temp_test_n["winPlacePerc"]
        temp_test_n_X = temp_test_n.drop(["winPlacePerc"],axis=1)

        temp_xgb = SGDRegressor(penalty='elasticnet')
        temp_xgb.fit(temp_train_n_X,temp_train_n_y)
        print("cluster",i,mean_absolute_error(temp_xgb.
↪predict(temp_test_n_X),temp_test_n_y))
        sum_error += mean_absolute_error(temp_xgb.
↪predict(temp_test_n_X),temp_test_n_y)*temp_test_n.shape[0]
    print("MAE",sum_error/num_test,"\n")
    return sum_error/num_test
```

[56]:
```
kmeans = KMeans(random_state=10,n_init = 10)
```

kaggle trick applied before clustering

[92]:
```
approach_3(scaled_train,scaled_test,2,kmeans,True)
```

[92]: 0.06871192430450172

19.4.1 Train on corresponding clusters for each matchType

[24]:
```
pd.DataFrame({'counts': scaled_train.groupby(('matchType')).size(),
                            'Percentage': scaled_train.groupby(('matchType')).
 ↪size() / len(train_data)})
```

[24]:

matchType	counts	Percentage
crashfpp	4972	0.001118
crashtpp	282	0.000063
duo	251120	0.056470
duo-fpp	796987	0.179220
flarefpp	516	0.000116
flaretpp	2041	0.000459
normal-duo	163	0.000037
normal-duo-fpp	4428	0.000996
normal-solo	243	0.000055
normal-solo-fpp	1336	0.000300
normal-squad	398	0.000089
normal-squad-fpp	13959	0.003139
solo	146105	0.032855
solo-fpp	429670	0.096621
squad	500754	0.112606
squad-fpp	1403320	0.315568

[62]:
```
train_two =␣
 ↪[train_crashfpp,train_crashtpp,train_duo,train_duo_fpp,train_flarefpp,train_flaretpp,train_
test_two =␣
 ↪[test_crashfpp,test_crashtpp,test_duo,test_duo_fpp,test_flarefpp,test_flaretpp,test_normal_
```

[63]:
```
train_three = [train_normal_solo,train_normal_solo_fpp]
test_three = [test_normal_solo,test_normal_solo_fpp]
```

[105]:
```
kmeans = KMeans(random_state=10,n_init = 10)
```

[158]:
```
num = scaled_test.shape[0]
sum_err = 0

for i in range(len(train_two)):

    train_single = train_two[i]
    test_single = test_two[i]
    temp_err = approach_3(train_single,test_single,2,kmeans,True)

for i in range(len(train_three)):
```

```
    train_single = train_three[i]
    test_single = test_three[i]
    temp_err = approach_3(train_single,test_single,3,kmeans,True)
    sum_err += test_single.shape[0]*temp_err

print(sum_err/num)
```

```
cluster 0 0.12002610449166844
cluster 1 0.07917891873564317
MAE 0.09374723061364762

cluster 0 0.12277274433211408
cluster 1 0.09760521912133967
MAE 0.1055230922213586

cluster 0 0.06515048498242368
cluster 1 0.07697880907065403
MAE 0.06788116145631488

cluster 0 0.07040791888074734
cluster 1 0.0597070642349998
MAE 0.062084460375900964

cluster 0 0.09838480929876577
cluster 1 0.10106126906762912
MAE 0.09910029854390745

cluster 0 0.09186571196657012
cluster 1 0.10708344306031975
MAE 0.09567014474000753

cluster 0 0.14802505108046826
cluster 1 0.25910492967255755
MAE 0.15964603743647385

cluster 0 0.1381055042561134
cluster 1 16.748989162281404
MAE 5.909683724417444

cluster 0 0.16906661857401453
cluster 1 0.20151363415159915
MAE 0.17887641612966373

cluster 0 0.07732055151216383
cluster 1 0.0817098454316731
MAE 0.07813856853566967
```

```
cluster 0 0.054150840694132334
cluster 1 0.06958143543301364
MAE 0.05700192227987011

cluster 0 0.07129491194864956
cluster 1 0.09086475053155861
MAE 0.07652035075164097

cluster 0 0.06490601449341453
cluster 1 0.0803233972138564
MAE 0.06903717014735461

cluster 0 0.17213243412295726
cluster 1 0.23841221473799895
cluster 2 395700585988.83636
MAE 66744677154.90734

cluster 0 0.20502527153539596
cluster 1 558497869983.1604
cluster 2 1.7479065151794566
MAE 20984024017.378338

14371502.512005236
```

19.4.2 This MAE is caused by overfitting of normal_duo, normal_solo, normal_solo_fpp

[105]:
```
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    display(get_centroids_single(train_normal_duo.
↪drop(["Id","matchId","groupId","matchType","winPlacePerc"],axis=1),kmeans,2,pd.
↪DataFrame({"winPlacePerc": train_normal_duo["winPlacePerc"]})))
```

```
           assists    boosts  damageDealt     DBNOs  headshotKills     heals  \
cluster_1  4.133333  0.200000    14.981667  9.266667       2.700000  8.500000
cluster_2  0.398496  1.315789     1.945771  0.804511       0.421053  1.413534

           killPlace  killPoints      kills  killStreaks  longestKill  \
cluster_1   0.060000    0.833333  13.733333     2.166667     0.877053
cluster_2   0.111128    0.646617   1.466165     0.684211     0.369181

           matchDuration  maxPlace  numGroups   revives  rideDistance  \
cluster_1       1.353867  0.091000   0.086667  0.866667      0.652023
cluster_2       1.245902  0.107669   0.106692  0.142857      1.606109

              roadKills  swimDistance  teamKills  vehicleDestroys  \
cluster_1  3.469447e-18 -3.469447e-18   0.100000     3.469447e-18
cluster_2  3.007519e-02  1.402180e-02   0.022556     3.007519e-02
```

```
           walkDistance  weaponsAcquired  winPoints  averageWpp  n_member
cluster_1      0.803161         3.013333   1.250000    0.621723        30
cluster_2      1.389846         0.596241   0.969925    0.494956       133
```

[131]:
```
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    display(get_centroids_single(train_normal_solo.
↪drop(["Id","matchId","groupId","matchType","winPlacePerc"],axis=1),kmeans,3,pd.
↪DataFrame({"winPlacePerc": train_normal_solo["winPlacePerc"]})))
```

```
            assists    boosts  damageDealt  DBNOs  headshotKills     heals  \
cluster_1  5.000000  0.700000    25.213000    0.0       6.050000  4.950000
cluster_2  2.552381  0.514286    11.032390    0.0       2.476190  3.847619
cluster_3  0.508475  0.940678     2.363378    0.0       0.576271  1.101695

           killPlace  killPoints      kills  killStreaks  longestKill  \
cluster_1   0.028500    0.850000  25.750000     2.800000     1.332410
cluster_2   0.078000    0.800000  10.485714     1.819048     1.103699
cluster_3   0.146949    0.525424   1.906780     0.754237     0.468796

           matchDuration  maxPlace  numGroups  revives  rideDistance  \
cluster_1       1.505200  0.138000   0.099500      0.0      0.175195
cluster_2       1.072333  0.185238   0.109238      0.0      0.328385
cluster_3       1.215254  0.209915   0.153475      0.0      0.475163

           roadKills  swimDistance     teamKills  vehicleDestroys  \
cluster_1        0.0 -1.040834e-17  5.000000e-02              0.0
cluster_2        0.0  1.734723e-17  1.619048e-01              0.0
cluster_3        0.0  4.823814e-02 -1.387779e-16              0.0

           walkDistance  weaponsAcquired  winPoints  averageWpp  n_member
cluster_1      0.543443         4.180000   1.275000    0.841900        20
cluster_2      0.670332         2.397143   1.200000    0.623168       105
cluster_3      1.219670         1.040678   0.788136    0.455448       118
```

[132]:
```
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    display(get_centroids_single(train_normal_solo_fpp.
↪drop(["Id","matchId","groupId","matchType","winPlacePerc"],axis=1),kmeans,3,pd.
↪DataFrame({"winPlacePerc": train_normal_solo_fpp["winPlacePerc"]})))
```

```
            assists    boosts  damageDealt  DBNOs  headshotKills     heals  \
cluster_1  6.593750  0.229167    30.446771    0.0       8.958333  6.197917
cluster_2  3.409396  0.574944    13.124499    0.0       3.154362  3.986577
cluster_3  0.564943  0.839849     2.220423    0.0       0.466583  1.110971

           killPlace  killPoints      kills  killStreaks  longestKill  \
cluster_1   0.048333    0.395833  31.125000     3.656250     1.062852
cluster_2   0.098389    0.449664  12.832215     2.152125     0.971810
cluster_3   0.149369    0.480454   1.833544     0.716267     0.351158
```

```
           matchDuration  maxPlace  numGroups  revives  rideDistance  \
cluster_1       1.490708  0.227187   0.160625      0.0      0.478400
cluster_2       1.034492  0.209306   0.141633      0.0      0.388631
cluster_3       0.941536  0.217781   0.168638      0.0      0.451418

              roadKills  swimDistance  teamKills  vehicleDestroys  \
cluster_1  8.673617e-19      0.131979   0.052083     0.000000e+00
cluster_2 -1.734723e-18      0.012051   0.062640     8.673617e-19
cluster_3  3.783102e-03      0.044994   0.022699     2.522068e-03

           walkDistance  weaponsAcquired  winPoints  averageWpp  n_member
cluster_1      0.581878         4.465625   0.593750    0.830103        96
cluster_2      0.597798         2.859060   0.674497    0.619756       448
cluster_3      0.985647         0.855485   0.720681    0.477408       792
```

19.4.3 Train normal_solo, normal_solo_fpp, normal_duo without clustering

[137]:
```
train_two =␣
 ↪[train_crashfpp,train_crashtpp,train_duo,train_duo_fpp,train_flarefpp,train_flaretpp,train_
test_two =␣
 ↪[test_crashfpp,test_crashtpp,test_duo,test_duo_fpp,test_flarefpp,test_flaretpp,test_normal_
train_all = [train_normal_solo,train_normal_solo_fpp,train_normal_duo]
test_all = [test_normal_solo,test_normal_solo_fpp,test_normal_duo]
```

[144]:
```
num = scaled_test.shape[0]
sum_err = 0

for i in range(len(train_two)):

    train_single = train_two[i]
    test_single = test_two[i]
    temp_err = approach_3(train_single,test_single,2,kmeans,True)
    sum_err += test_single.shape[0]*temp_err

for i in range(len(train_all)):

    train_single = train_all[i]
    test_single = test_all[i]

    train_single_c = train_single.drop(["matchId", "Id","matchType"],axis=1)
    train_single_c = train_single_c.groupby("groupId").mean()
    train_single_c_y = train_single_c["winPlacePerc"]
    train_single_c_X = train_single_c.drop(["winPlacePerc"],axis=1)

    test_single_c = test_single.drop(["matchId", "Id","matchType"],axis=1)
    test_single_c = test_single_c.groupby("groupId").transform('mean')
    test_single_c_y = test_single_c["winPlacePerc"]
```

```
    test_single_c_X = test_single_c.drop(["winPlacePerc"],axis=1)
    temp_sgd = SGDRegressor(penalty='elasticnet')
    temp_sgd.fit(train_single_c_X,train_single_c_y)
    err = mean_absolute_error(temp_sgd.predict(test_single_c_X),test_single_c_y)
    sum_err += err*test_single.shape[0]

print(sum_err/num)
```

```
0.08684248030110803
```

20 Significance test

The T-test for the means of two independent samples of scores.

[88]:
```
display(scaled_test_1_X)
```

```
        assists  boosts  damageDealt  DBNOs  headshotKills  heals  killPlace  \
0             1       0       0.6800      0              0      0       0.47
1             0       2       0.6547      0              0      2       0.61
2             0       3       0.0000      0              0     13       0.49
3             0       0       2.0000      2              0      0       0.69
4             1       1       1.0000      1              0      1       0.30
…           …       …           …     …              …     …         …
890666        0       7       1.2620      0              1      7       0.16
890667        0       7       2.9410      0              0      6       0.06
890668        0       0       0.3010      0              0      0       0.57
890669        0       0       0.0000      0              0      0       0.74
890670        0       4       1.8040      1              1      2       0.11

        killPoints  kills  killStreaks  …  matchType_normal-duo  \
0            0.000      0            0  …                     0
1            1.095      0            0  …                     0
2            0.000      0            0  …                     0
3            0.000      0            0  …                     0
4            0.000      1            1  …                     0
…              …    …          …   …                   …
890666       0.000      2            1  …                     0
890667       1.016      3            1  …                     0
890668       1.364      0            0  …                     0
890669       1.029      0            0  …                     0
890670       0.000      2            1  …                     0

        matchType_normal-duo-fpp  matchType_normal-solo  \
0                              0                      0
1                              0                      0
2                              0                      0
3                              0                      0
4                              0                      0
```

```
…                            …                       …
890666                           0                       0
890667                           0                       0
890668                           0                       0
890669                           0                       0
890670                           0                       0

        matchType_normal-solo-fpp  matchType_normal-squad  \
0                               0                       0
1                               0                       0
2                               0                       0
3                               0                       0
4                               0                       0
…                             …                       …
890666                          0                       0
890667                          0                       0
890668                          0                       0
890669                          0                       0
890670                          0                       0

        matchType_normal-squad-fpp  matchType_solo  matchType_solo-fpp  \
0                                0               0                   0
1                                0               0                   0
2                                0               0                   0
3                                0               0                   0
4                                0               0                   0
…                              …               …                   …
890666                           0               0                   1
890667                           0               1                   0
890668                           0               0                   0
890669                           0               0                   0
890670                           0               0                   0

        matchType_squad  matchType_squad-fpp
0                     0                    0
1                     0                    1
2                     0                    1
3                     0                    1
4                     0                    1
…                   …                    …
890666                0                    0
890667                0                    0
890668                0                    1
890669                0                    1
890670                0                    1

[890671 rows x 39 columns]
```

[89]:
```
display(scaled_test_1_k_X)
```

```
        assists    boosts  damageDealt     DBNOs  headshotKills     heals  \
0          0.50  1.500000     1.073000  0.500000       0.000000  1.000000
1          0.00  2.000000     1.680980  1.600000       0.000000  1.400000
2          1.00  3.250000     1.659250  1.500000       0.500000  7.000000
3          0.00  0.500000     0.750000  0.750000       0.000000  0.250000
4          0.25  0.750000     2.227000  2.250000       0.250000  2.500000
…           …        …           …        …             …        …
890666     0.00  7.000000     1.262000  0.000000       1.000000  7.000000
890667     0.00  7.000000     2.941000  0.000000       0.000000  6.000000
890668     0.25  1.416667     2.281500  1.583333       0.416667  2.083333
890669     0.00  0.400000     1.194800  1.000000       0.200000  0.400000
890670     0.25  2.250000     0.774125  0.500000       0.250000  1.000000

        killPlace  killPoints  kills  killStreaks  …  matchType_normal-duo  \
0        0.325000      0.0000   1.00     0.500000  …                     0
1        0.426000      1.1900   1.00     0.800000  …                     0
2        0.217500      0.0000   2.00     1.000000  …                     0
3        0.675000      0.0000   0.00     0.000000  …                     0
4        0.207500      0.0000   2.25     1.250000  …                     0
…           …          …    …           …  …                   …
890666   0.160000      0.0000   2.00     1.000000  …                     0
890667   0.060000      1.0160   3.00     1.000000  …                     0
890668   0.383333      1.2075   1.50     0.833333  …                     0
890669   0.434000      1.1588   1.00     1.000000  …                     0
890670   0.197500      0.0000   1.25     0.750000  …                     0

        matchType_normal-duo-fpp  matchType_normal-solo  \
0                              0                      0
1                              0                      0
2                              0                      0
3                              0                      0
4                              0                      0
…                            …                    …
890666                         0                      0
890667                         0                      0
890668                         0                      0
890669                         0                      0
890670                         0                      0

        matchType_normal-solo-fpp  matchType_normal-squad  \
0                               0                       0
1                               0                       0
2                               0                       0
3                               0                       0
4                               0                       0
…                             …                     …
```

```
890666                          0                     0
890667                          0                     0
890668                          0                     0
890669                          0                     0
890670                          0                     0

        matchType_normal-squad-fpp  matchType_solo  matchType_solo-fpp  \
0                                 0               0                   0
1                                 0               0                   0
2                                 0               0                   0
3                                 0               0                   0
4                                 0               0                   0
…                               …               …                 …
890666                            0               0                   1
890667                            0               1                   0
890668                            0               0                   0
890669                            0               0                   0
890670                            0               0                   0

        matchType_squad  matchType_squad-fpp
0                     0                    0
1                     0                    1
2                     0                    1
3                     0                    1
4                     0                    1
…                   …                    …
890666                0                    0
890667                0                    0
890668                0                    1
890669                0                    1
890670                0                    1

[890671 rows x 39 columns]
```