THEORY

Ο αλγοριθμος K-means ειναι ενας τροπος να πραγματοποιησουμε διαχωριμός των δεδομενων(παρατηρήσεων) *X* ,όπου X διάνυσμα των χαρακτηριστικών διαστασης M, σε συσταδες(ομάδες) ομοιων μεταξυ τους.Η καλυτερη διαμεριση του συνολου θεωρείται εκείνη που για συγκεκριμενο αριθμο ομαδων έχει σχετικα καλή μεταβολή των παρατηρησεων στο εσωτερικό της(variation).Ο k-means χρησιμοποιεί την απόσταση μεταξύ των παρατηρησεων(σημείων ή δειγματων) για να καθορίσει το πόσο διαφέρουν μεταξύ τους.Πιο συγκεκριμένα,εάν έχουμε ένα σύνολο N δειγματων(samples) και θελουμε να χωρίσουμε σε K ομάδες ο αλγοριθμος ακολουθεί τα εξής βήματα:

- 1)Επιλογή K παρατηρησεων ως **κεντρα βαρους** απο το συνολο N των δειγματων,συμβολιζονται ως mj
- 2)Υπολογισμος της απόστασης ολων των αλλων παρατηρησεων από αυτά τα κεντρα,και συγκεκριμένα της ευκλείδιας απόστασης:

$$||xi - mj||^2$$

- 3)Ευρεση νεων κεντρων βάρους mj',ως τον μεσο ορο της αποστασης στην καθε συστάδα και επανέλαβε απο το βημα 2,μέχρι ικανοποίηση ενός κριτηρίου

Γενικα στόχος του αλγόριθμου είναι να μειώσει το ανα-συσταδα-αθροισμα-τετραγώνων(within-cluster sum-of-squares) ή αδράνεια(inertia):

$$\sum_{n=1}^{N} ||xi - mj||^2$$

Αξίζει να σημειωθεί πως για πολλές διαστάσεις των δεδομένων *X* (M μεγαλο) η ευκλείδια απόσταση γίνεται υπολογιστικα δύσκολη

Στη συνέχεια παρουσιάζεται η ανάλυση των δεδομένων των στοιχηματικών εταιρειών,και συγκεκριμένα των αποδόδεσων της καθε στοιχηματικής εταιρείας σε 3 ομάδες(3-clusters).Τα βήματα που ακολουθούνται περιγράφονται συνοπτικα και στη συνέχεια παρουσιάζονται αναλυτικά με κώδικα:

Δεδομένα

- Διαβάζονται τα δεδομένα με από το αρχείο 'final_merged_table_features+odds+result.csv' που περιέχει όλες τις αποδόσεις των στοιχειματικων και τα αποτελεσματα για τον κάθε αγώνα.Τα δεδομέμα αυτά προέκυψαν από προηγούμενη επεξεργασία της βασης δεδομένων
- Χωρίζουμε τις 4 στοιχηματικές εταιρείες , 'B365','BW','LB','IW' σε 4 DataFrames,ετοιμα για επεξεργασια

Προεπεξεργασία(Preprocessing)

- Κάνουμε drop ολοκηρη της σειρα(δείγμα) στις Nan values.Σημειώνεται εδώ πως θα μπορούσε να είχε ακολουθηθεί άλλη στρατηγική και αντί να διώχναμε ολόκληρη τη σειρά ,να αντικαθιστούσαμε την NaN value με τον μεσο όρο ή τον διάμεσο της εκάστοτε στήλης που παρουσιαζεται
- Φιτάρουμε τον Kmeans μία φορα στην εταιρεία και παρατηρούμε πως οι ακραίες τιμες μπαινουμε σε αναμενόμενες κλάσεις,οπότε αποφασίζουμε να φιλτράρουμε τις ακραίες τιμές με βάση τη φορμουλα:

$$LowLimit = Q1 - 1.5IQR,$$

$$UpLimit = Q3 + 1.5IQR$$

,όπου Q1,Q3,IQR τα 25%-quantile,75%-quantile,inter-quantile-range αντίστοιχα.Αυτά είναι τα όρια που η κάθε τιμή του πίνακα αποδόσεων θα γίνεται δεκτή και θα χρησιμοπιέιται για φιλτράρισμα!

Εκπαίδευση και αποτελέσματα

- Εκπαιδεύουμε τον kmeans σε κάθε μία από τις 4 εταιρείες
- Παρουσιάζουμε διαγράμματα των αποδόσεων σε σχέση a)το πραγματικό αποτέλεσμα b)την ομαδοποίηση του kmeans
- Παρουσιάζουμε τις συχνότητες με τις οποίες εμφανίζεται το κάθε τελικό αποτέλεσμα μέσα στην κάθε συστάδα!

Στη συνέχεια παρουσιάζεται ο κωδικας για την υλοποίηση των παραπάνω

Practise

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
```

read_Data

In [2]:
```
df=pd.read_csv('./final_merged_table_features+odds+result.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	date	home_team_api_id	away_team_api_id	B365H	B365D	B365A	BWH
0	0	2008-08-17 00:00:00	9987	9993	1.73	3.40	5.0	1.75
1	1	2008-11-15 00:00:00	9987	9999	1.25	5.25	10.0	1.23
2	2	2008-11-29 00:00:00	9987	9984	1.73	3.40	4.5	1.65
3	3	2008-12-13 00:00:00	9987	9986	1.53	4.00	6.0	1.55
4	4	2009-01-24 00:00:00	9987	9998	1.44	4.00	6.5	1.40

5 rows × 34 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25979 entries, 0 to 25978
Data columns (total 34 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Unnamed: 0                    25979 non-null  int64  
 1   date                          25979 non-null  object 
 2   home_team_api_id              25979 non-null  int64  
 3   away_team_api_id              25979 non-null  int64  
 4   B365H                         22592 non-null  float64
 5   B365D                         22592 non-null  float64
 6   B365A                         22592 non-null  float64
 7   BWH                           22575 non-null  float64
 8   BWD                           22575 non-null  float64
 9   BWA                           22575 non-null  float64
 10  IWH                           22520 non-null  float64
 11  IWD                           22520 non-null  float64
 12  IWA                           22520 non-null  float64
 13  LBH                           22556 non-null  float64
 14  LBD                           22556 non-null  float64
 15  LBA                           22556 non-null  float64
 16  team_api_id                   25979 non-null  int64  
 17  buildUpPlaySpeed              25979 non-null  float64
 18  buildUpPlayPassing            25979 non-null  float64
 19  chanceCreationPassing         25979 non-null  float64
 20  chanceCreationCrossing        25979 non-null  float64
 21  chanceCreationShooting        25979 non-null  float64
 22  defencePressure               25979 non-null  float64
 23  defenceAggression             25979 non-null  float64
 24  defenceTeamWidth              25979 non-null  float64
 25  buildUpPlaySpeed_away         25979 non-null  float64
 26  buildUpPlayPassing_away       25979 non-null  float64
 27  chanceCreationPassing_away    25979 non-null  float64
 28  chanceCreationCrossing_away   25979 non-null  float64
 29  chanceCreationShooting_away   25979 non-null  float64
 30  defencePressure_away          25979 non-null  float64
 31  defenceAggression_away        25979 non-null  float64
 32  defenceTeamWidth_away         25979 non-null  float64
 33  Result                        25979 non-null  object 
dtypes: float64(28), int64(4), object(2)
memory usage: 6.7+ MB
```

we drop rows with nan values

In [6]:
```
df=df.dropna()
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22467 entries, 0 to 24556
Data columns (total 34 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Unnamed: 0                   22467 non-null  int64  
 1   date                         22467 non-null  object 
 2   home_team_api_id             22467 non-null  int64  
 3   away_team_api_id             22467 non-null  int64  
 4   B365H                        22467 non-null  float64
 5   B365D                        22467 non-null  float64
 6   B365A                        22467 non-null  float64
 7   BWH                          22467 non-null  float64
 8   BWD                          22467 non-null  float64
 9   BWA                          22467 non-null  float64
 10  IWH                          22467 non-null  float64
 11  IWD                          22467 non-null  float64
 12  IWA                          22467 non-null  float64
 13  LBH                          22467 non-null  float64
 14  LBD                          22467 non-null  float64
 15  LBA                          22467 non-null  float64
 16  team_api_id                  22467 non-null  int64  
 17  buildUpPlaySpeed             22467 non-null  float64
 18  buildUpPlayPassing           22467 non-null  float64
 19  chanceCreationPassing        22467 non-null  float64
 20  chanceCreationCrossing       22467 non-null  float64
 21  chanceCreationShooting       22467 non-null  float64
 22  defencePressure              22467 non-null  float64
 23  defenceAggression            22467 non-null  float64
 24  defenceTeamWidth             22467 non-null  float64
 25  buildUpPlaySpeed_away        22467 non-null  float64
 26  buildUpPlayPassing_away      22467 non-null  float64
 27  chanceCreationPassing_away   22467 non-null  float64
 28  chanceCreationCrossing_away  22467 non-null  float64
 29  chanceCreationShooting_away  22467 non-null  float64
 30  defencePressure_away         22467 non-null  float64
 31  defenceAggression_away       22467 non-null  float64
 32  defenceTeamWidth_away        22467 non-null  float64
 33  Result                       22467 non-null  object 
dtypes: float64(28), int64(4), object(2)
memory usage: 6.0+ MB
```

In [8]:

```
odds_columns=['B365H',
       'B365D', 'B365A', 'BWH', 'BWD', 'BWA', 'IWH', 'IWD', 'IWA', 'LBH',
       'LBD', 'LBA']
odds_cols_B365=['B365H','B365D', 'B365A']
odds_cols_BW=['BWH', 'BWD', 'BWA']
odds_cols_IW=[ 'IWH', 'IWD', 'IWA']
odds_cols_LB=['LBH','LBD', 'LBA']
```

In [9]:
```
odds=df[odds_columns]
odds.head()
```

Out[9]:

	B365H	B365D	B365A	BWH	BWD	BWA	IWH	IWD	IWA	LBH	LBD	LBA
0	1.73	3.40	5.0	1.75	3.35	4.20	1.85	3.2	3.5	1.80	3.30	3.75
1	1.25	5.25	10.0	1.23	5.00	10.00	1.30	4.2	8.0	1.25	4.50	10.00
2	1.73	3.40	4.5	1.65	3.45	4.90	1.65	3.4	4.2	1.72	3.40	4.00
3	1.53	4.00	6.0	1.55	3.55	5.65	1.55	3.5	4.8	1.50	3.50	6.00
4	1.44	4.00	6.5	1.40	3.85	7.10	1.45	3.8	5.4	1.40	3.75	7.00

In [10]:
```
odds_B365=df[odds_cols_B365]
odds_BW=df[odds_cols_BW]
odds_IW=df[odds_cols_IW]
odds_LB=df[odds_cols_LB]
```

In [11]:
```
odds_BW.shape,odds_IW.shape
```

Out[11]:
```
((22467, 3), (22467, 3))
```

clustering

In [12]:
```
from sklearn.cluster import KMeans
```

In [13]:
```
kmeans=KMeans(n_clusters=3,init='k-means++',
    n_init=10,
    max_iter=300,
    tol=0.0001,
    verbose=0,
    random_state=1,
    copy_x=True,
    algorithm='full',)
```

In [14]:
```
kmeans.fit(odds_B365)
```

Out[14]:
```
KMeans(algorithm='full', n_clusters=3, random_state=1)
```

In [15]:
```
len(kmeans.labels_)
```

Out[15]:
```
22467
```

In [16]:
```
odds_B365
```

Out[16]:

	B365H	B365D	B365A
0	1.73	3.40	5.00
1	1.25	5.25	10.00
2	1.73	3.40	4.50
3	1.53	4.00	6.00
4	1.44	4.00	6.50
...	...	...	...
24552	4.00	3.60	1.91
24553	1.91	3.50	4.00
24554	3.30	3.40	2.20
24555	2.10	3.30	3.75
24556	3.50	3.25	2.20

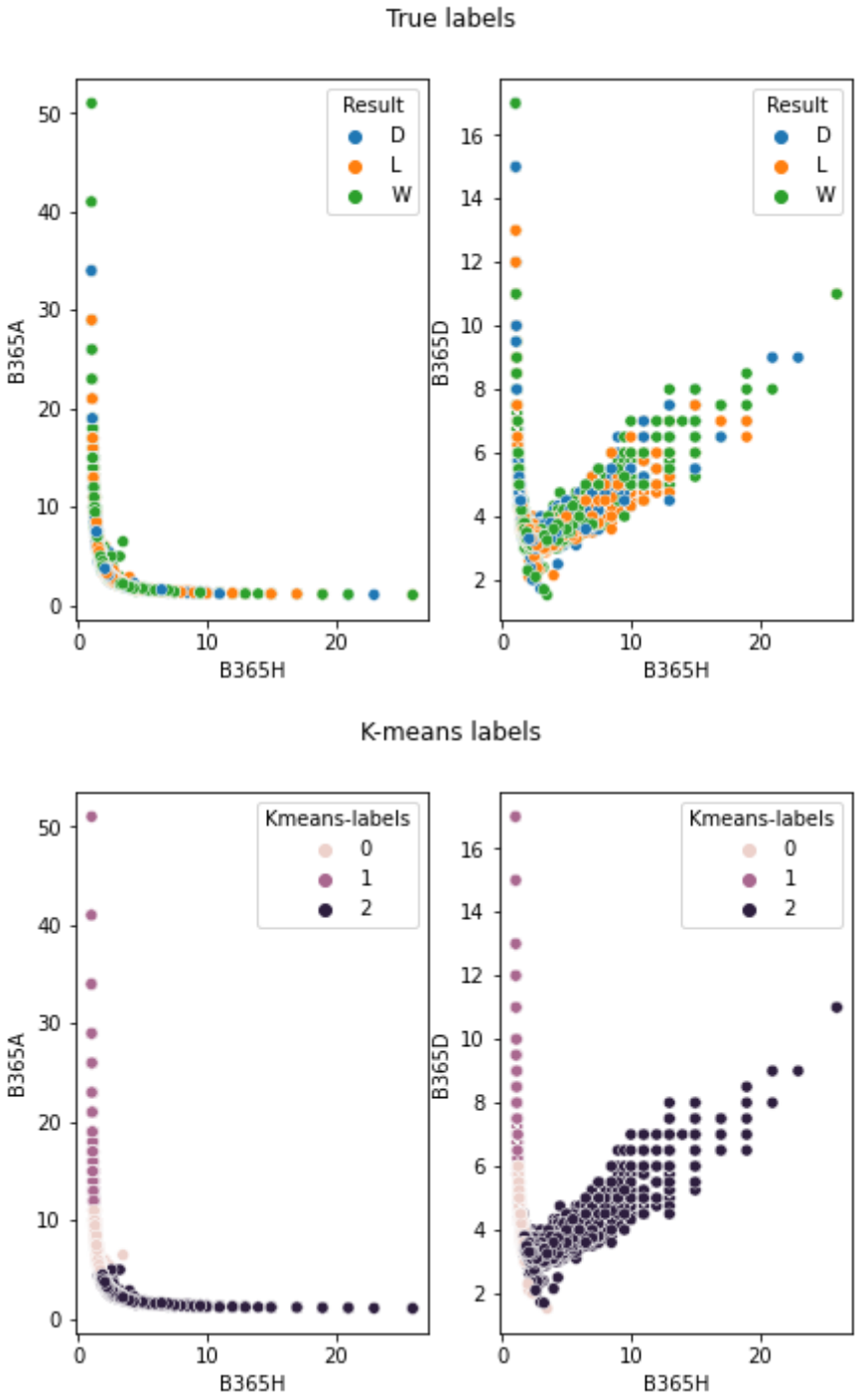
22467 rows × 3 columns

In [17]:
```
import seaborn as sns
```

In [18]:
```
#sns.pairplot(data=odds_B365,hue='Kmeans-labels',corner=True)
```

In [19]:
```
def plot_kmeans_and_results(odds,kmeans_labels):
    odds_topic=odds.copy()
    odds_topic_columns=odds_topic.columns
    odds_topic['Result']=df['Result'].loc[odds_topic.index]

    home=odds_topic.columns[0]
    dual=odds_topic.columns[1]
    away=odds_topic.columns[2]
    #sns.pairplot(data=odds_topic,hue='Result',corner=True)
    #plt.figure(figsize=(7,7))
    fig,(ax1,ax2)=plt.subplots(1,2,figsize=(7,5))
    #sns.scatterplot(x='B365H',y='B365A',hue='Result',data=odds_topic)
    sns.scatterplot(x=home,y=away,hue='Result',data=odds_topic,ax=ax1)

    #sns.scatterplot(x='B365H',y='B365D',hue='Result',data=odds_topic)
    sns.scatterplot(x=home,y=dual,hue='Result',data=odds_topic,ax=ax2)
    fig.suptitle('True labels')
    plt.show()

    odds_topic=odds_topic.drop('Result',axis=1)
    #plt.figure(figsize=(7,7))
    fig,(ax1,ax2)=plt.subplots(1,2,figsize=(7,5))
    odds_topic['Kmeans-labels']=kmeans_labels
    #sns.pairplot(data=odds_topic,hue='Kmeans-labels',corner=True)
    #sns.scatterplot(x='B365H',y='B365A',hue='Kmeans-labels',data=odds_to
    sns.scatterplot(x=home,y=away,hue='Kmeans-labels',data=odds_topic,ax=
    #plt.title('K-means labels')
    #plt.show()
    sns.scatterplot(x=home,y=dual,hue='Kmeans-labels',data=odds_topic,ax=

    #plt.title('K-means labels')
    fig.suptitle('K-means labels')
    plt.show()
    del(odds_topic)
```

In [20]:
```
plot_kmeans_and_results(odds_B365,kmeans.labels_)
```



drop outliers

In [21]:
```
odds_B365.quantile(0.25),odds_B365.quantile(0.75),odds_B365.quantile(0.75
```

Out[21]:
```
(B365H    1.67
 B365D    3.30
 B365A    2.50
 Name: 0.25, dtype: float64,
 B365H    2.80
 B365D    4.00
 B365A    5.25
 Name: 0.75, dtype: float64,
 B365H    1.13
 B365D    0.70
 B365A    2.75
 dtype: float64)
```

In [22]:
```
odds_B365.describe()
```

Out[22]:

	B365H	B365D	B365A
count	22467.000000	22467.000000	22467.000000
mean	2.628185	3.841820	4.665438
std	1.794432	1.119239	3.736650
min	1.040000	1.530000	1.080000
25%	1.670000	3.300000	2.500000
50%	2.100000	3.500000	3.500000
75%	2.800000	4.000000	5.250000
max	26.000000	17.000000	51.000000

In [23]:
```
def IQR(odds):
    iqr=odds.quantile(0.75)-odds.quantile(0.25)
    return iqr
def filter_by_iqr(odds):
    q1=odds.quantile(0.25)
    q3=odds.quantile(0.75)
    booleans1=(odds>q1-1.5*IQR(odds))
    booleans2=(odds<q3+1.5*IQR(odds))
    #print(booleans2.index)
    #print(odds[(booleans1)&(booleans2)])
    return odds[(booleans1)&(booleans2)].dropna()
```

In [24]:
```
odds_B365_filtered=filter_by_iqr(odds_B365)

odds_BW_filtered=filter_by_iqr(odds_BW)

odds_LB_filtered=filter_by_iqr(odds_LB)

odds_IW_filtered=filter_by_iqr(odds_IW)
```

In [25]:
```
odds_B365_filtered.describe()#,odds_BW_filtered.describe(),odds_LB_filter
```

Out[25]:

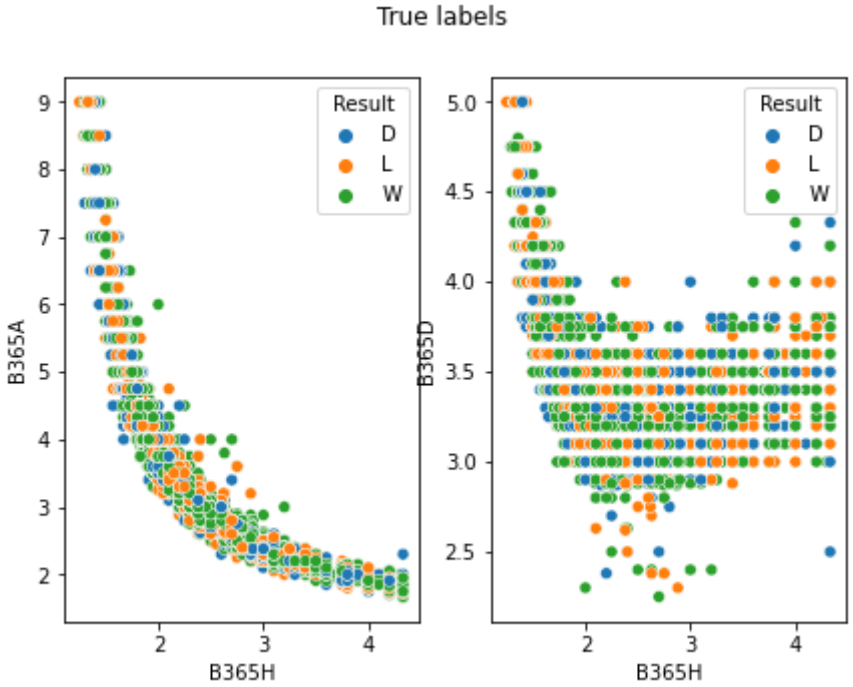
	B365H	B365D	B365A
count	18138.000000	18138.000000	18138.000000
mean	2.261065	3.486744	3.959591
std	0.667243	0.395649	1.684212
min	1.250000	2.250000	1.670000
25%	1.750000	3.250000	2.750000
50%	2.100000	3.400000	3.500000
75%	2.600000	3.600000	4.750000
max	4.330000	5.000000	9.000000

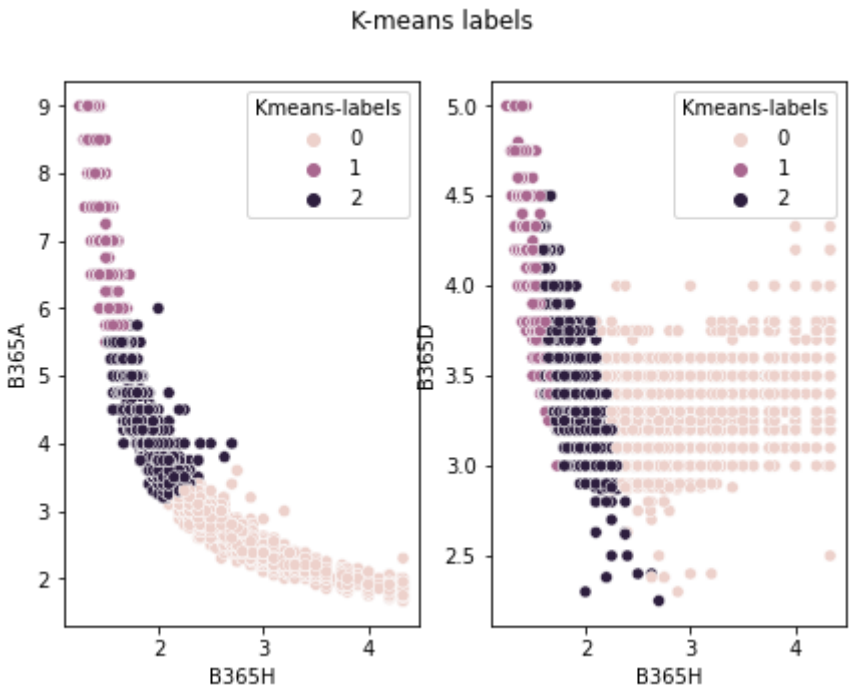
train Kmean again

In [26]:
```
kmeans.fit(odds_B365_filtered)
```

Out[26]:
```
KMeans(algorithm='full', n_clusters=3, random_state=1)
```

In [27]:
```
plot_kmeans_and_results(odds_B365_filtered,kmeans.labels_)
```



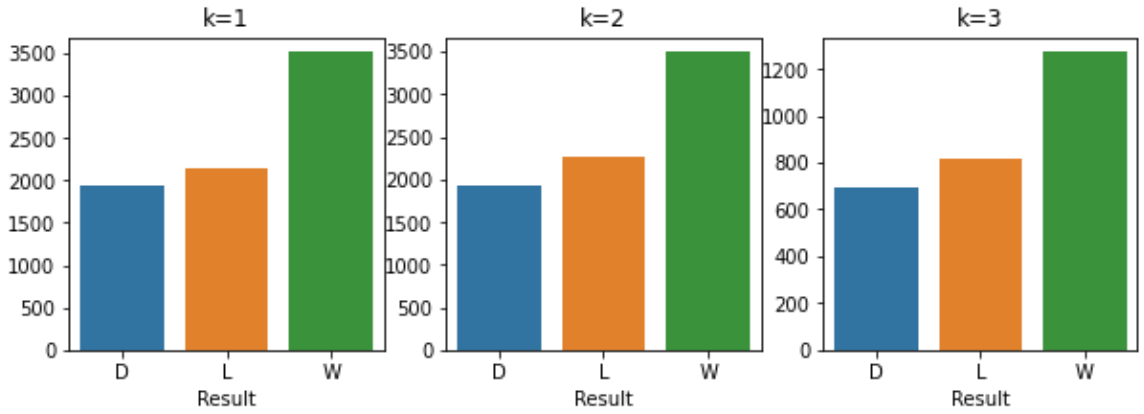


In [119…

```
def frequencies_inside_kmeans(odds,kmeans):
    odds_with_clusters=odds.copy()
    col1=odds_with_clusters.columns[0]
    odds_with_clusters['clusters']=kmeans.labels_
    odds_with_clusters['Result']=df['Result'].loc[odds_with_clusters.inde
    grouped=odds_with_clusters.groupby(['clusters','Result']).count()[col

    fig,(ax)=plt.subplots(1,3,figsize=(10,3))
    for i in range(3):


        sns.barplot(x=grouped[i].index,y=grouped[i],ax=ax[i])
        ax[i].set_ylabel('')
        ax[i].set_title(f'k={i+1}')
        #plt.ylabel('')
        #plt.ylabel('')
        #plt.title(f'k={i+1}')
        #plt.show()
    del(odds_with_clusters)
```

In [120…

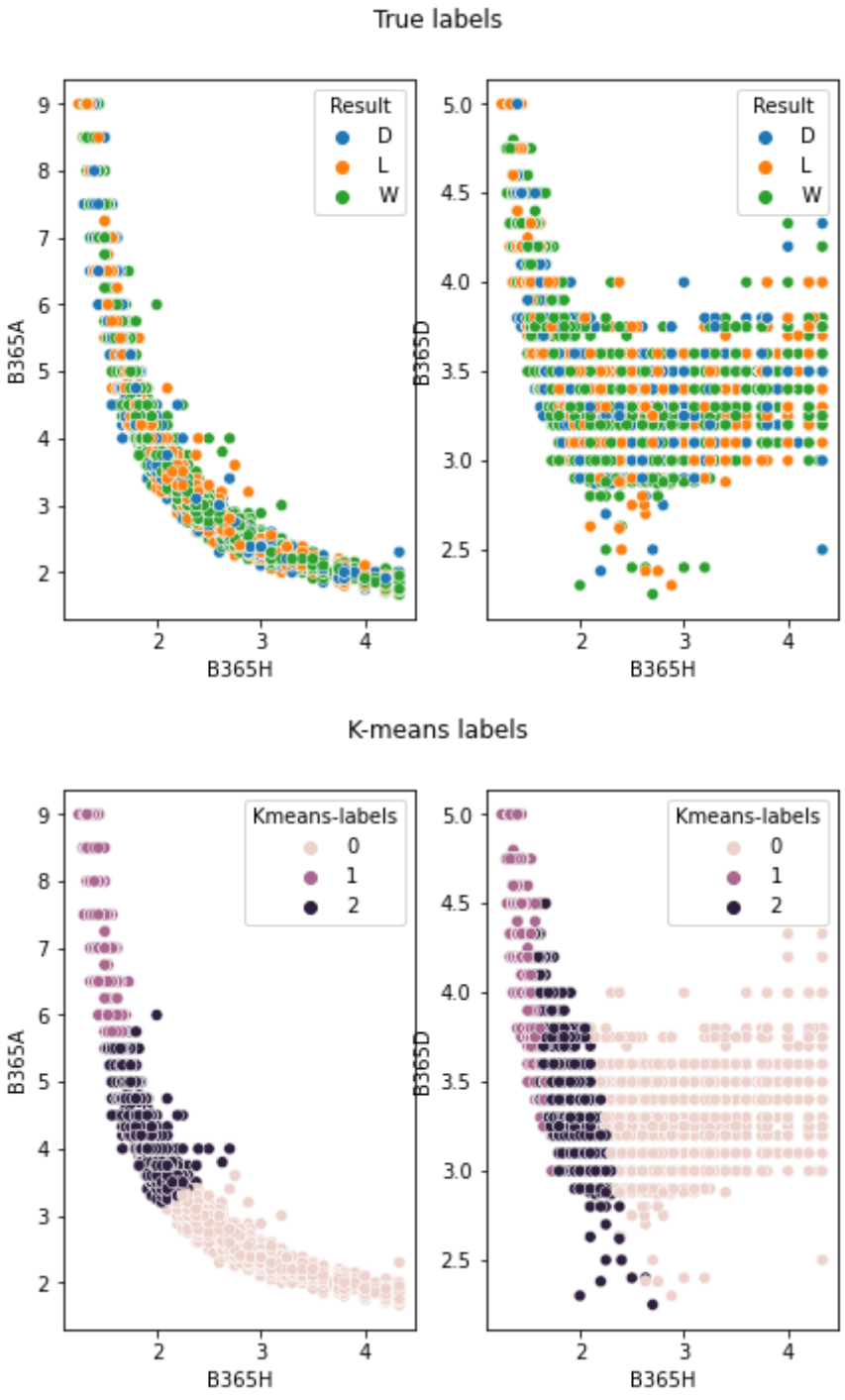
```
frequencies_inside_kmeans(odds_IW_filtered,kmeans)
```



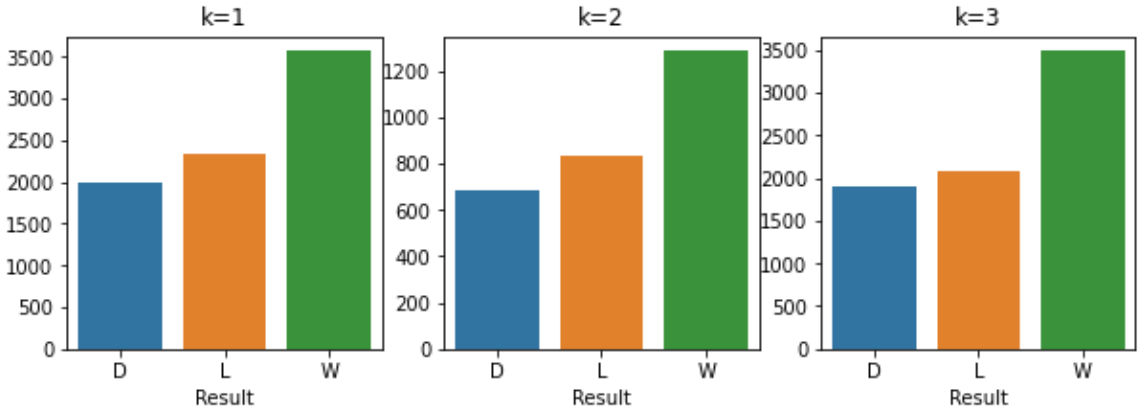
In [121…

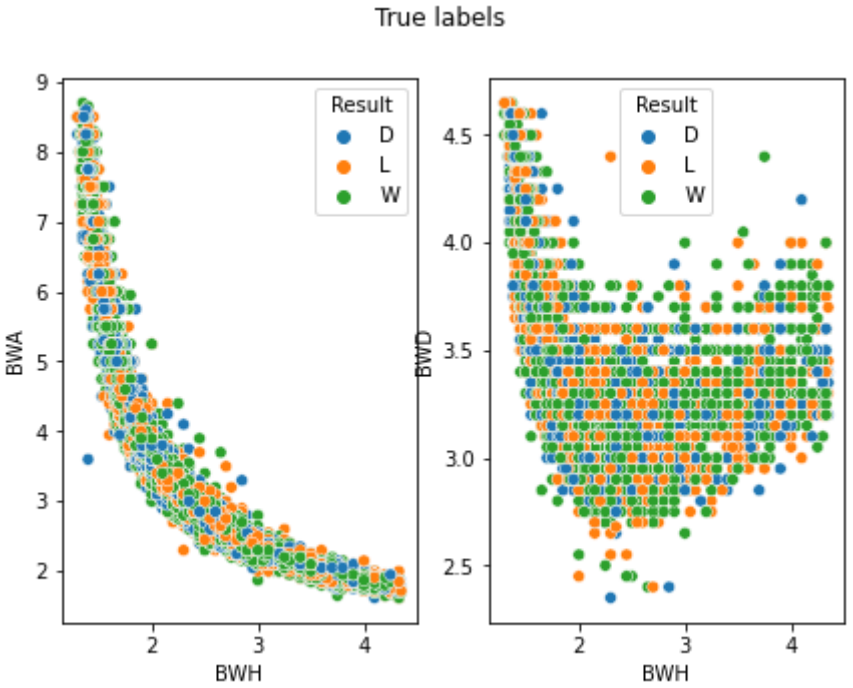
```
companies=['B365','BW','LB','IW']
i=0
for data in [odds_B365_filtered,odds_BW_filtered,odds_LB_filtered,odds_IW
    print(f'=======start training for {companies[i]} company=======')
    kmeans.fit(data)
    print(f'plots for {companies[i]} company')
    plot_kmeans_and_results(data,kmeans.labels_)
    print(f"center of clusters:{kmeans.cluster_centers_}")
    i+=1
    print("frequencies on each cluster:")
    frequencies_inside_kmeans(data,kmeans)
```

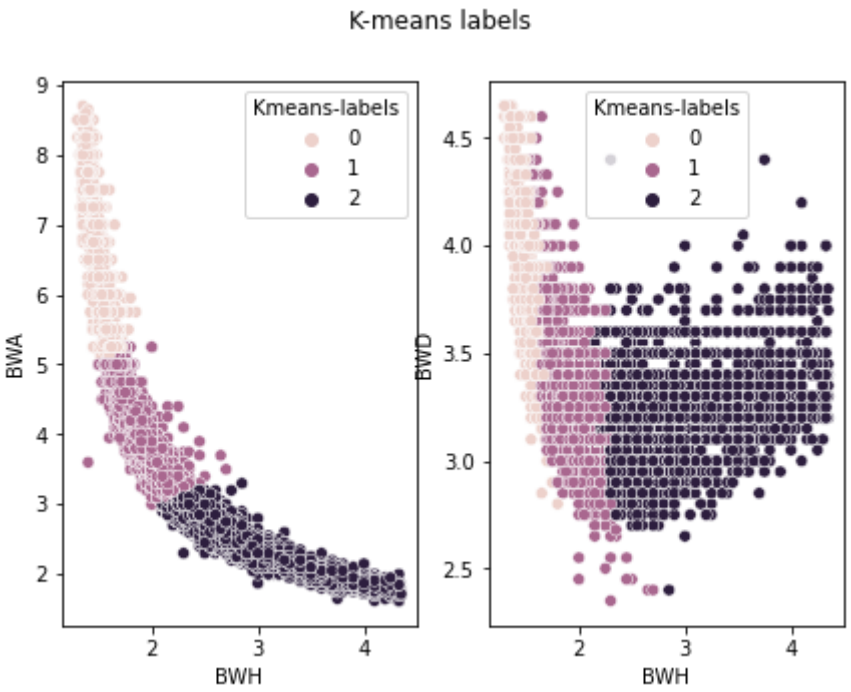
```
=======start training for B365 company=======
plots for B365 company
```



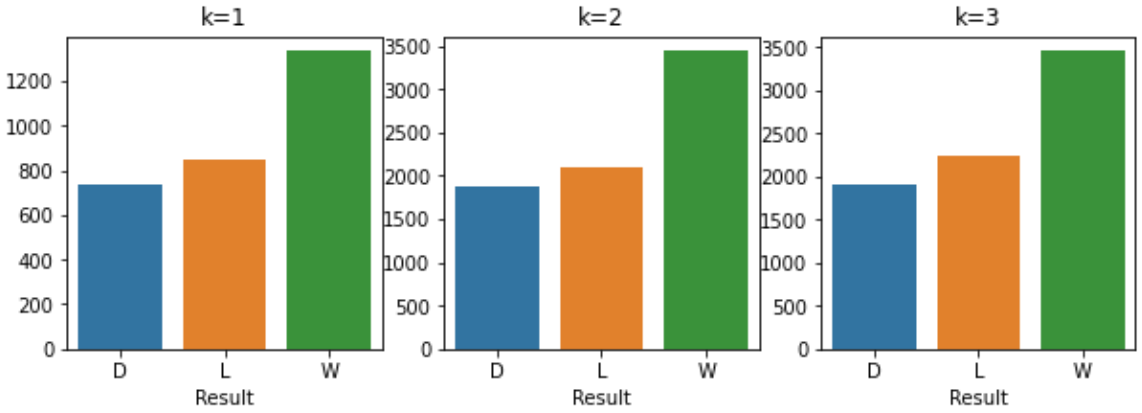
```
center of clusters:[[2.85586841 3.28760269 2.60970969]
 [1.47671674 4.17878398 7.1961731 ]
 [1.92583848 3.43789509 4.17402468]]
frequencies on each cluster:
=======start training for BW company=======
plots for BW company
```

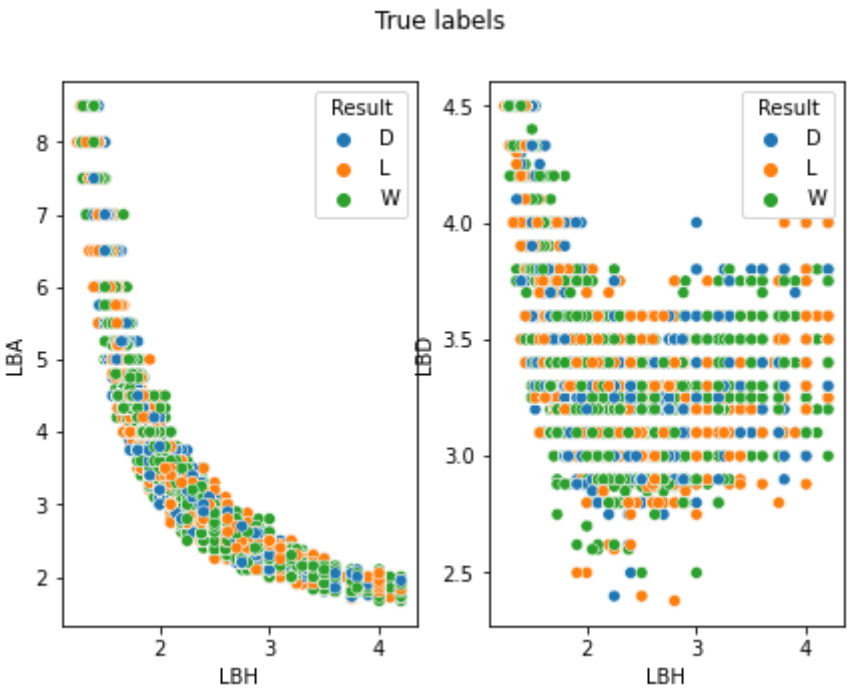


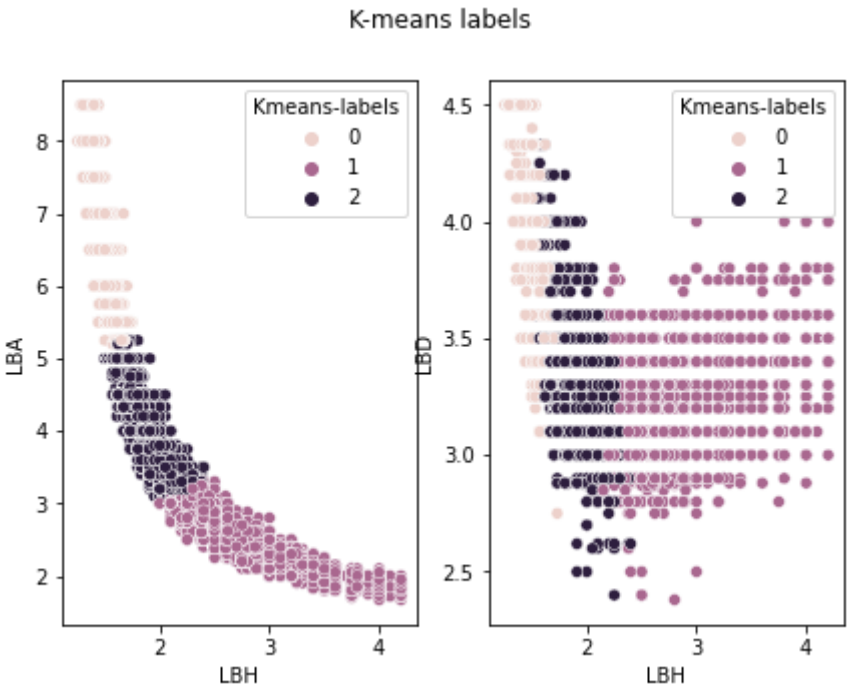




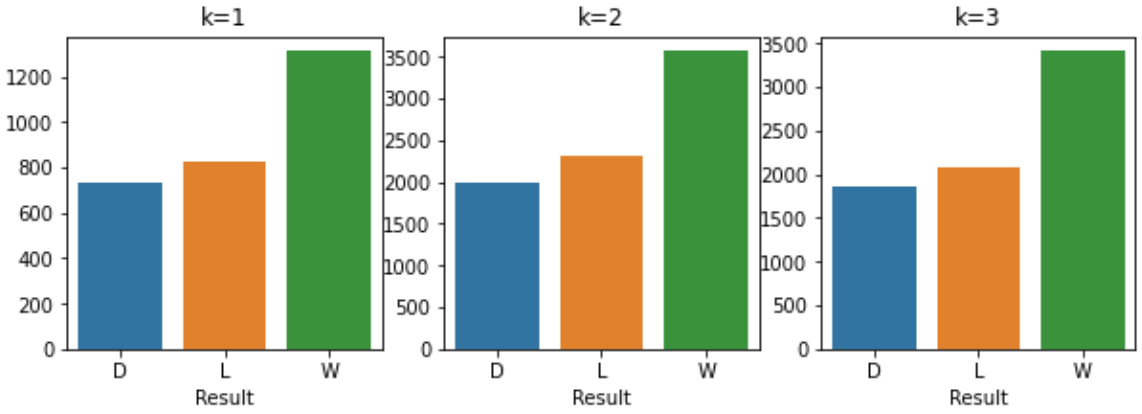

```
center of clusters:[[1.5061265  3.93669402 6.46955556]
 [1.94066991 3.34416261 3.91723393]
 [2.84484681 3.24305325 2.51301907]]
frequencies on each cluster:
=======start training for LB company=======
plots for LB company
```



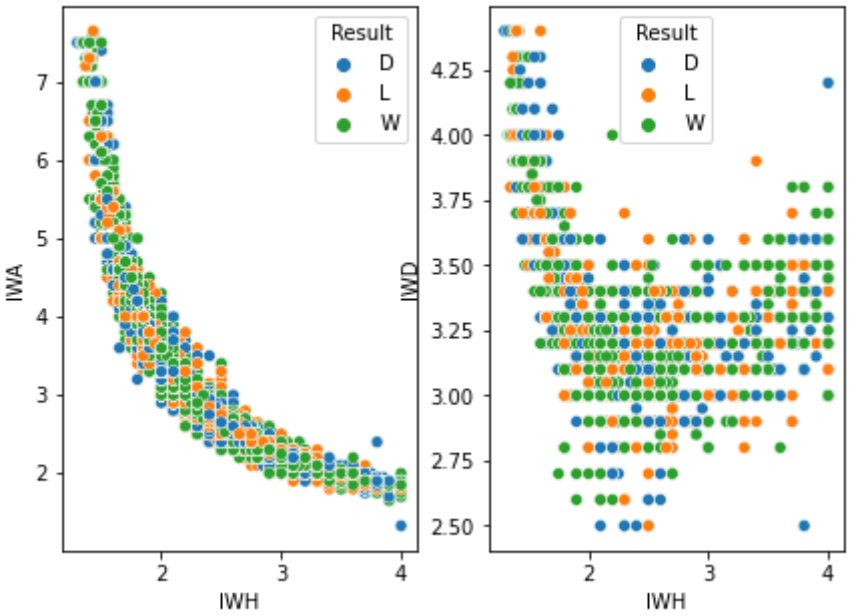




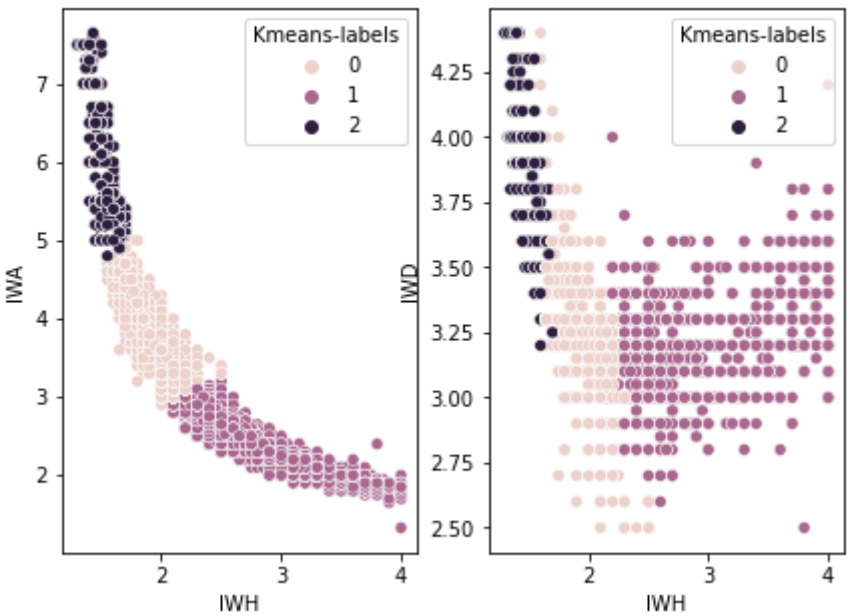

```
center of clusters:[[1.48198745 3.92811018 6.54433403]
 [2.75778838 3.23755183 2.52955742]
 [1.91408464 3.34588532 3.93850785]]
frequencies on each cluster:
=======start training for IW company=======
plots for IW company
```



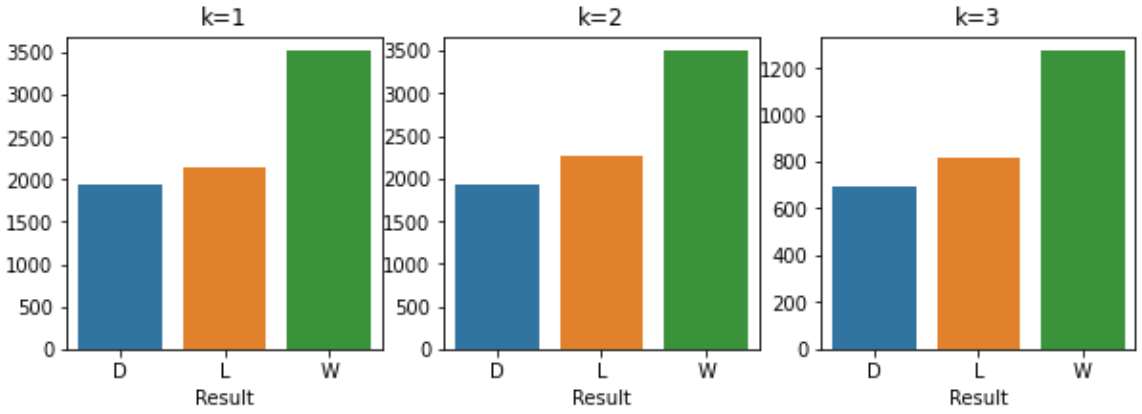
True labels



K-means labels




```
center of clusters:[[1.92378796 3.29198574 3.7586414 ]
 [2.74410914 3.19382001 2.48613832]
 [1.4987149  3.85068395 6.07712383]]
frequencies on each cluster:
```



Φαινεται πως τα παιχνιδια που ήρθαν νικη για την ομάδα στην έδρα της επικρατούν σε κάθε cluster και σε όλες τις στοιχηματικές εταιρείες.Αυτό μπορει να δικαιολογηθεί και πάλι με το ότι το 46% του dataset αποτελέιται απο HOME-WIN παρατηρήσεις!

In [ ]: