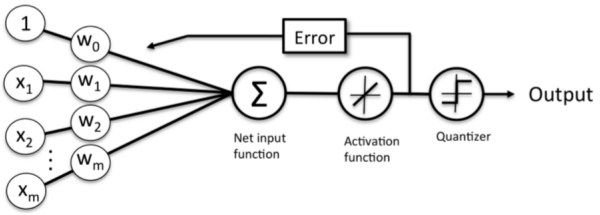
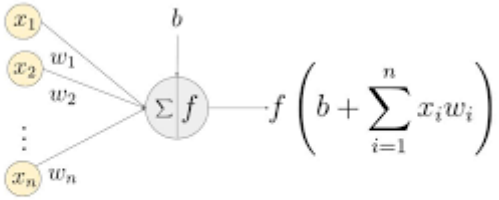
Theory

Το γραμμικό νευρωνικό δίκτυο(linear neural network) είναι αλγόριθμος μηχανικής μάθησης με δυνατότητα εφαρμογής κυρίως σε γραμμικού τύπου προβληματα αλλά και σαν πρώτη προσέγγιση σε κάποιο μη γραμμικό.Η γενική ιδέα είναι ο συνδυασμός (πολλών) χαρακτηριστικών εισόδων ,ο γραμμικός συνδυασμός αυτών και στη συνέχεια η έξοδος του νευρώνα αποτελεί είτε αυτό το άθροισμα είτε ,για ένα πρόβλημα ομαδοποίησης, περνά από την **συνάρτηση heaviside** με έξοδο 1 ή -1(δηλαδή μία από τις δύο κλάσεις).Σχηματικα:





An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

Η διαφορά με τον γνωστό και απλό γραμμικό αλγόριθμο του Perceptron είναι ,ότι προκειμένου να γίνει αναβάθμιση των βαρών γίνεται χρηση του αλγορίθμου **Gradient Descent**,ενώ στον Perceptron άλλη διαδικασία. Τα βήματα του αλγόριθμου συνοπτικά είναι:

1. Ορισμός βαρών **w,i** ανάλογος του αριθμού των εισόδων
2. Γραμμικός συνδυασμός των εισόδων (εισοδοι στον νευρώνα)(πολλές φορές πρόσθεση και επιπλέον όρου bias)
3. Υπολογισμός εξόδου από τον νευρώνα με βάση τη σχέση f(x)=x,οπου x:γραμ.συνδιασμος,ή κάποια αλλη για ομαδοποίηση αλλά μονο ακριβως πριν την έξοδο!(για πολλους νευρώνες στη σειρα)
4. Υπολογισμός συνάρτηση Loss για κάποιο αριθμό δειγμάτων
5. Update τα βάρη,διαδικασια γνωστή ως Back Propagation, και ξανα στο βήμα 2

Τέλος το *πολυστρωματικό νευρωνικό δίκτυο*(Multilayer Neural Network) ως θεωρία παρουσιάζεται σε άλλο notebook.Εδω γίνεται η εφαρμογή του με εισόδους μόνο τις αποδόσεις των εταιρειών.(3 inputs)

Παρακάτω θα περιγραφεί συνοπτικά η **διαδικασία** που ακολουθηθηκε προκειμένου να απαντηθούν καποια ερωτήματα και πιο κατω δίνεται αναλυτικα ο κωδικας με διαγράμματα και αποτελέσματα:

- Σε επόμενο notebook(κωδικα) φαίνεται η διαδικασία διαβάσματος της βάσης δεδομένων.Η βάση κατέβηκε απο την ιστοσελίδα του kaggle.com. Από τη βάση εξάγονται κάποια '.csv' αρχεία.Εδώ χρησιμοποιούνται αυτά που περιέχουν τις αποδόσεις και το αρχείο με τα αποτελέσματα των αγώνων('Results' το οποίο δημιουργήθηκε από την διαφορά των goal του κάθε αγώνα)

Preprocessing

- Στα δεδομένα υπάρχουν missing values που αποτυπώνονται ως Nan.Αυτές έχουν αφεραιθεί από το κάθε dataset.Σημειώνονται πως μια άλλη εναλλακτική πρακτική θα ήταν να αντικαθιστόυσαμε αυτές τις τιμές με τον μέσο όρο(mean) της στήλης στην οποία βρίσκονται ή την διάμεσο(median)
- Οπτικοποίηση των δεδομένων(μόνο για την BET365)
- Χωρισμός των δεδομένων σε Training ,Test set και κανονικοποίηση(StandarScaler)προκειμένου να γίνει εκπαίδευση των μοντέλων

Εκπαίδευση

- Εκπαίδευση των μοντέλων στο trainning set,και στη συνέχεια επανεκπαίδευση με χρηση της μεθόδου crossvilidation.Στην αρχή γίνεται μια επίδειξη στην εταιρεία bet365 και στο τέλος ,με χρήση συνάρτησης,εκπαιδεύονται μοντέλα και στις άλλες 3 εταιρείες
- Σε όλη τη ροή του κώδικα φαίνονται στοιχεία για τα δεδομένα ,για τα μοντέλα,η καμπύλη Loss curve κατα τη φάση της εκπαίδευσης,confusion matrix(όπου φαίνεται τι προβλέπει το μοντέλο σε σχέση με τις σωστές ετικέτες/στόχους),οο χρονος που χρειάστηκε να εκπαιδευτεί το μη γραμμικο perceptron(προφανώς όσο περισσότερα layers,τόσο περισσότερα βάρη και άρα περισσότεροσ χρόνος εκ).Στο τέλος παρουσιάζονται κάποια αποτελέσματα όπως το ποια εταιρεία έχει καλύτερο crossvilidation score(mean score).
- Σημειώνεται πως εκπαιδεύτηκαν 2 μη γραμμικά μοντέλα στην εταιρεία bet365 ,1 με πολλα βάρη και 1 με λιγότερα για να φανεί η επίδραση της πολυπλοκότητας του δικτύου

Χρησιμοποιήθηκαν τα μοντέλα από την ανοιχτή κοινότητα και συγκεκριμένα για τα μοντέλα των νευρωνικών οι βιβλιοθήκες του sklearn σε python,περισσότερες πληροφορίες στο https://scikit-learn.org

Practise

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

Preprocessing

Bet365

In [2]:
```
file_name1='bet365-bets.csv'
file_name2='results.csv'

stream= os.popen('pwd') #put path of file_name
working_dir_path=stream.read().split('\n')[0] #read the command pwd from
absolute_path_1=working_dir_path + "/" + file_name1
absolute_path_2=working_dir_path + "/" + file_name2


x_df=pd.read_csv(absolute_path_1)
y_df=pd.read_csv(absolute_path_2)
```

In [3]:
```
x_df.info(),y_df.info()
```

Out[3]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22592 entries, 0 to 22591
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   B365H   22592 non-null  float64
 1   B365D   22592 non-null  float64
 2   B365A   22592 non-null  float64
dtypes: float64(3)
memory usage: 529.6 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22592 entries, 0 to 22591
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Result  22592 non-null  object
dtypes: object(1)
memory usage: 176.6+ KB
(None, None)
```

In [4]:
```
df=pd.concat([x_df,y_df],axis=1)
```

In [5]:
```
df.head()
```

Out[5]:

	B365H	B365D	B365A	Result
0	1.73	3.40	5.00	D
1	1.95	3.20	3.60	D
2	2.38	3.30	2.75	L
3	1.44	3.75	7.50	W
4	5.00	3.50	1.65	L

In [6]:
```
from sklearn import preprocessing
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
```

In [7]:
```
label_encoder=preprocessing.LabelEncoder()
y_enc=label_encoder.fit_transform(y_df['Result'])
```

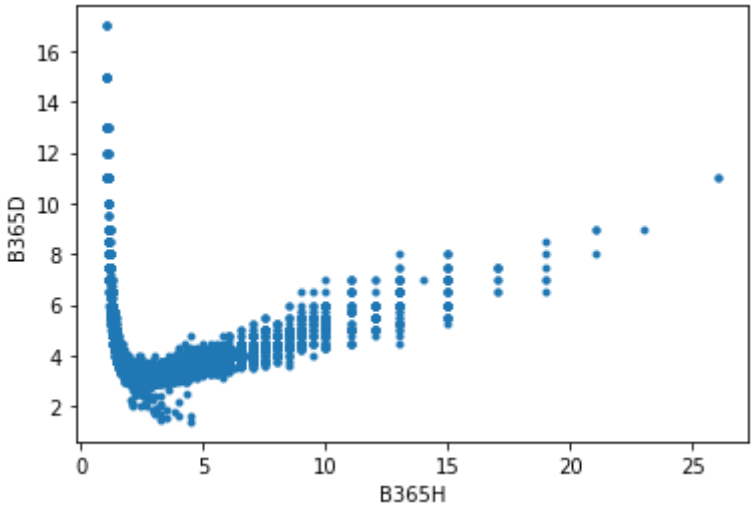
encoded --> D:0,L:1,W:2

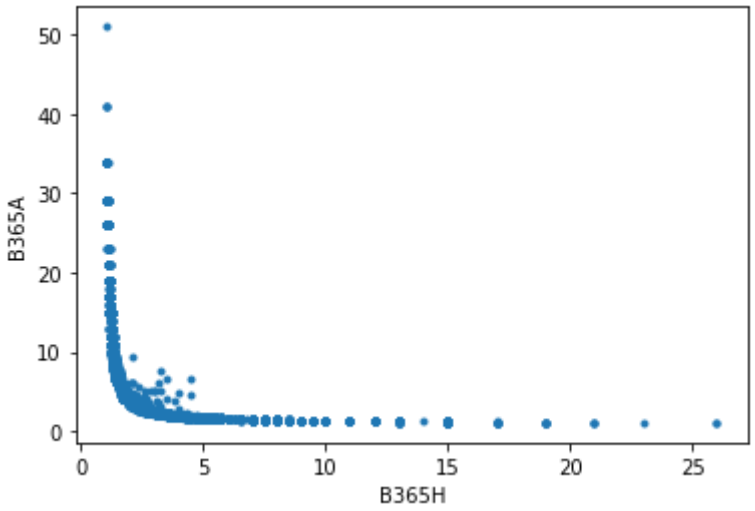
In [8]:
```
draw=(y_enc==0).sum()/len(y_enc)
loss=(y_enc==1).sum()/len(y_enc)
win=(y_enc==2).sum()/len(y_enc)
```

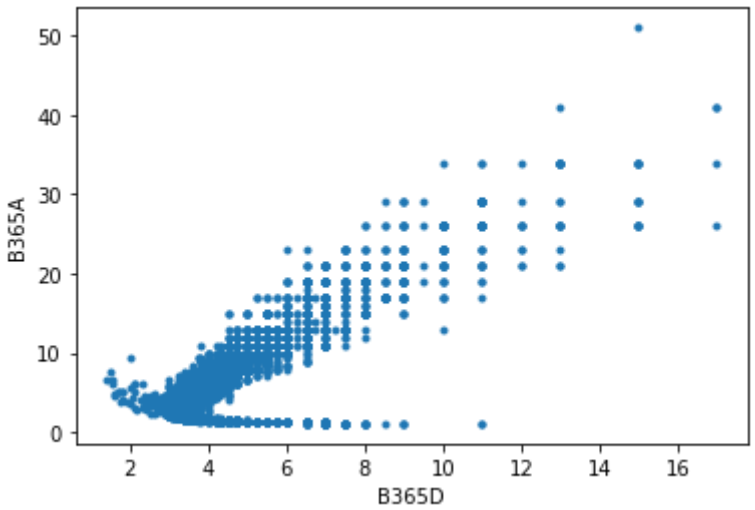
unbalanced data set so if we predict win(home) all the time we get 46% accuracy!!

data visualization

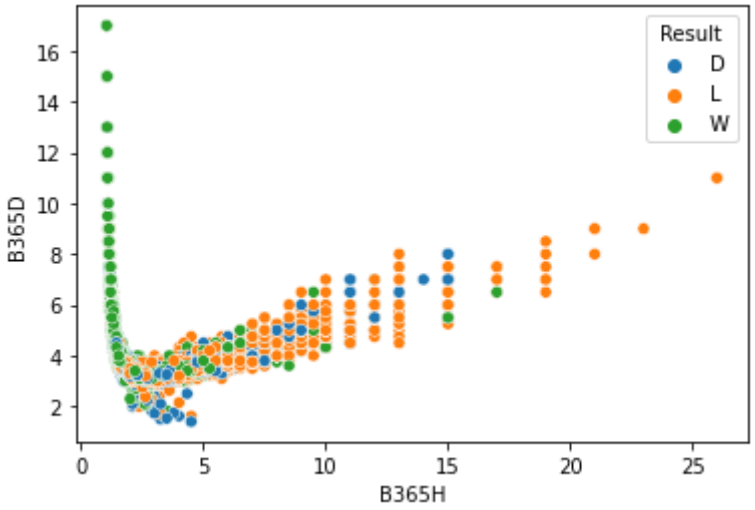
In [10]:
```
plt.plot(x_df['B365H'],x_df['B365D'],".")
plt.xlabel('B365H')
plt.ylabel('B365D')
plt.show()
plt.plot(x_df['B365H'],x_df['B365A'],".")
plt.xlabel('B365H')
plt.ylabel('B365A')
plt.show()
plt.plot(x_df['B365D'],x_df['B365A'],".")
plt.xlabel('B365D')
plt.ylabel('B365A')
plt.show()
```

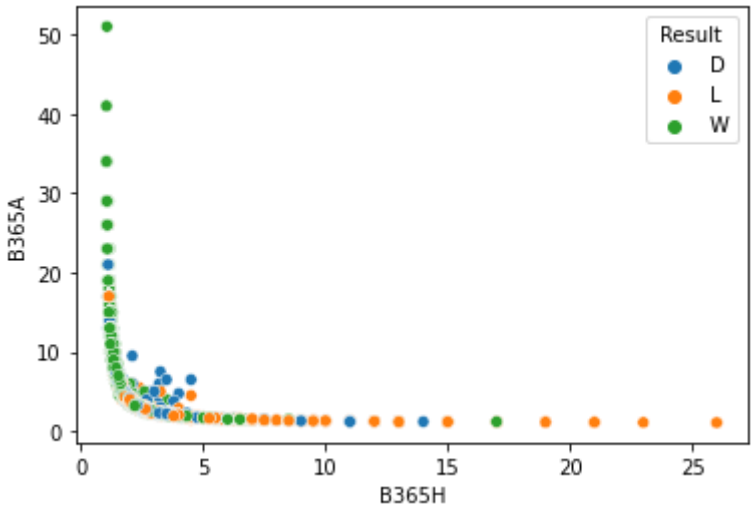


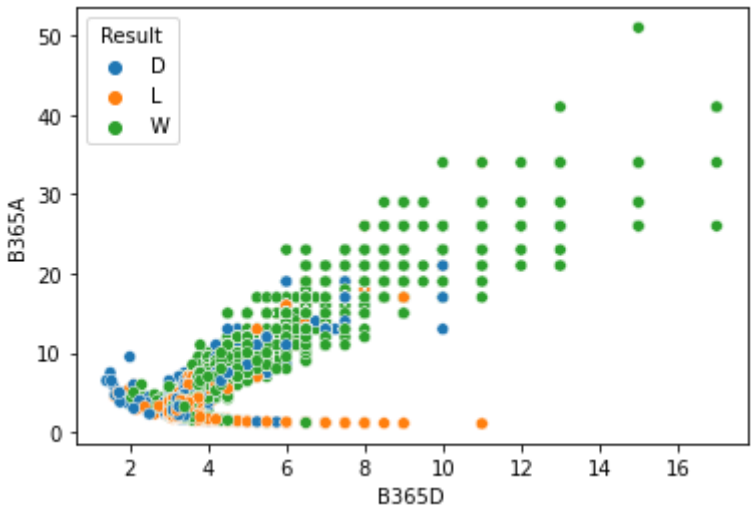




In [11]:
```
sns.scatterplot(x='B365H',y='B365D',hue='Result',data=df)
plt.show()

sns.scatterplot(x='B365H',y='B365A',hue='Result',data=df)
plt.show()

sns.scatterplot(x='B365D',y='B365A',hue='Result',data=df)
plt.show()
```







Training

Linear Neural Network(using SGDClassidier)

In [9]:
```
X_train,X_test,y_train,y_test=train_test_split(np.array(x_df),y_enc,test_

(y_train==0).sum(),(y_train==1).sum(),(y_train==2).sum()
```

Out[9]:
```
(5093, 5900, 9339)
```

In [12]:
```
from sklearn.linear_model import SGDClassifier
sgd = SGDClassifier(max_iter=1000, tol=1e-3,loss='log')
```

In [13]:
```
%%time
sgd.fit(X_train,y_train)
```

Out[13]:
```
CPU times: user 772 ms, sys: 4.07 ms, total: 776 ms
Wall time: 873 ms
SGDClassifier(loss='log')
```

In [14]:
```
print('linear neuro single training score:')
sgd.score(X_train,y_train)
```

Out[14]:
```
linear neuro single training score:
0.5301987015542002
```

In [ ]:
```
linear_cm_train=ConfusionMatrixDisplay(confusion_matrix(y_train,sgd.predi
linear_cm_train.plot()
plt.title("train set confusion matrix")
```

In [16]:
```
linear_cv_score=cross_val_score(estimator=sgd,X=X_train,y=y_train,cv=10,n
```

In [17]:
```
print('linear neuron cross validation mean and std:')
linear_cv_score.mean(),linear_cv_score.std()
```

Out[17]:
```
linear neuron cross validation mean and std:
(0.5152971786563977, 0.02202549916383954)
```

In [15]:
```
print('linear neuro test score:')
sgd.score(X_test,y_test)
```

Out[15]:
```
linear neuro test score:
0.5283185840707965
```

In [ ]:
```
linear_cm_test=ConfusionMatrixDisplay(confusion_matrix(y_test,sgd.predict
linear_cm_test.plot()
plt.title("test set confusion matrix")
```

MultiLayer Perceptron

In [18]:
```
from sklearn.neural_network import MLPClassifier
```

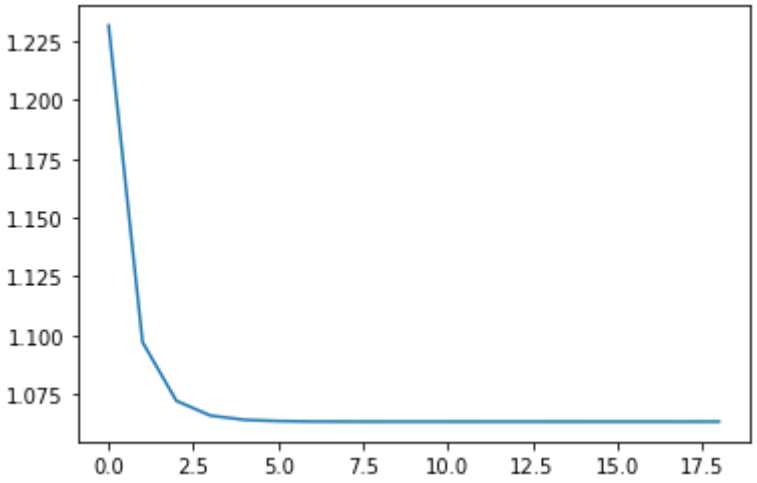
In [39]:
```
#mlp1=MLPClassifier(hidden_layer_sizes=(50,8),activation='relu',solver='s

mlp2=MLPClassifier(hidden_layer_sizes=(100,2,3),
                   activation='relu',
                   solver='sgd',
                   max_iter=900,
                   learning_rate_init=0.001,
                   tol=0.00001)
```

In [40]:
```
mlp2.fit(X_train,y_train)
```

Out[40]:
```
MLPClassifier(hidden_layer_sizes=(100, 2, 3), max_iter=900, solver='sgd',
              tol=1e-05)
```

In [41]:
```
plt.plot(mlp2.loss_curve_)
```

Out[41]:
```
[<matplotlib.lines.Line2D at 0x7f3851723fa0>]
```



In [42]:
```
print('predicted probabilities are:')
mlp2.predict_proba(np.array(X_train[0:10]))
```

Out[42]:
```
predicted probabilities are:
array([[0.25040721, 0.29001766, 0.45957513],
       [0.25040721, 0.29001766, 0.45957513],
       [0.25040721, 0.29001766, 0.45957513],
       [0.25040721, 0.29001766, 0.45957513],
       [0.25040721, 0.29001766, 0.45957513],
       [0.25040721, 0.29001766, 0.45957513],
       [0.25040721, 0.29001766, 0.45957513],
       [0.25040721, 0.29001766, 0.45957513],
       [0.25040721, 0.29001766, 0.45957513],
       [0.25040721, 0.29001766, 0.45957513]])
```

In [43]:
```
print('training set score:')
mlp2.score(X_train,y_train)
```

Out[43]:
```
training set score:
0.4593252016525674
```

In [44]:
```
print('coefficients of neuronrs are :')
mlp2.coefs_[0].shape,mlp2.coefs_[1].shape,mlp2.coefs_[2].shape#,mlp.coefs
```

Out[44]:
```
coefficients of neuronrs are :
((3, 100), (100, 2), (2, 3))
```

In [45]:
```
%%time
mlp_cv_score=cross_val_score(estimator=mlp2,X=X_train,y=y_train,cv=10,n_j
```

```
CPU times: user 90.7 ms, sys: 74.3 ms, total: 165 ms
Wall time: 4min 51s
```

In [46]:
```
print('MLP cross validation mean score and standar deviation are:')
mlp_cv_score.mean(),mlp_cv_score.std()
```

Out[46]:
```
MLP cross validation mean score and standar deviation are:
(0.5102314998203197, 0.03383315629048947)
```

In [49]:
```
test_pred=mlp2.predict(X_test)
print('test score:',mlp2.score(X_test,y_test))
```

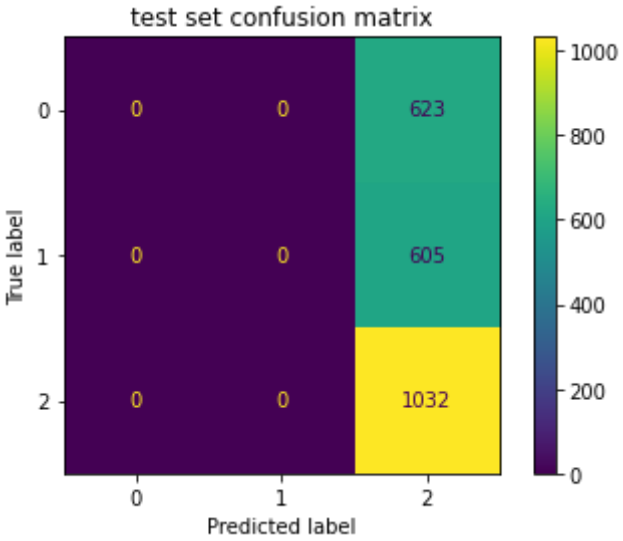
```
test score: 0.45663716814159294
```

In [50]:
```
confusion_matrix(y_test,test_pred)
```

Out[50]:
```
array([[   0,    0,  623],
       [   0,    0,  605],
       [   0,    0, 1032]])
```

In [51]:
```
ds_test=ConfusionMatrixDisplay(confusion_matrix(y_test,test_pred))
ds_test.plot()
plt.title("test set confusion matrix")
```

Out[51]:
```
Text(0.5, 1.0, 'test set confusion matrix')
```



Όπως βλέπουμε το μοντέλο δεν προβλέπει ποτε την κλαση 0(ισοπαλλια)και 1(ισοπαλλια),παρά την κλαση 2(home win) που είναι και η κλαση με τις περισσότερες παρατηρήσεις στο dataset.Με λίγα λόγια το πιο πολύπλοκο μοντέλο,με παραμέτρους πολλές περισσότερες από τα βάρη έχει κάνει overfit στα δεδομένα

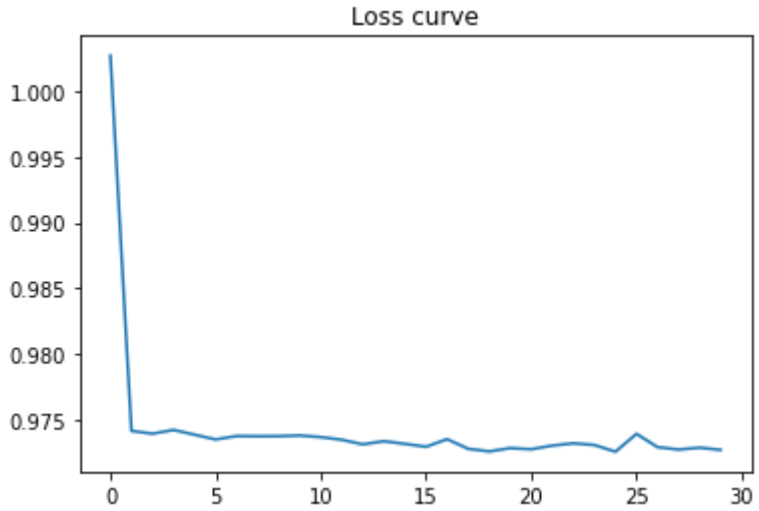
more simple MLP

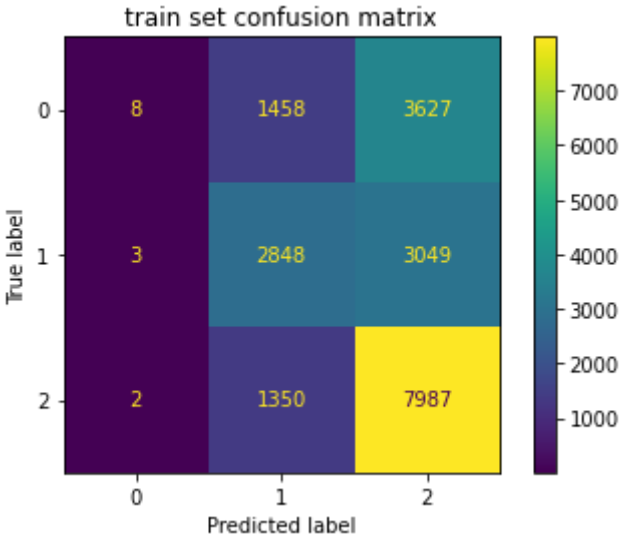
In [37]:
```
mlp1=MLPClassifier(hidden_layer_sizes=(50,8),activation='relu',solver='sg
```

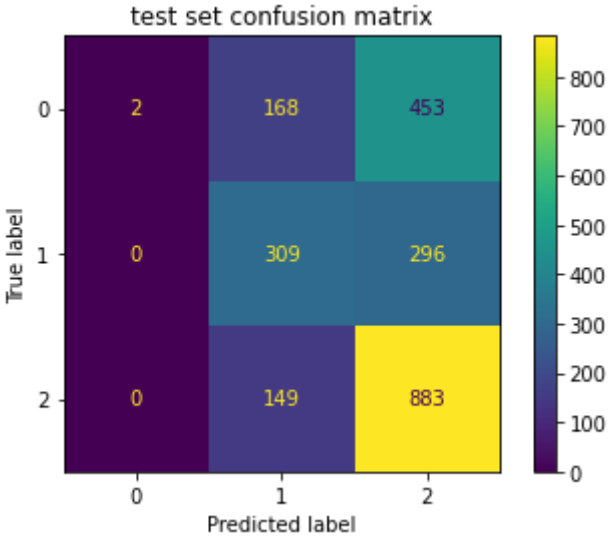
In [38]:
```
mlp1.fit(X_train,y_train)

plt.plot(mlp1.loss_curve_)
plt.title('Loss curve');


print('train_score:',mlp1.score(X_train,y_train))
print("test_score:",mlp1.score(X_test,y_test))

mlp1_cm_train=ConfusionMatrixDisplay(confusion_matrix(y_train,mlp1.predic
mlp1_cm_train.plot()
plt.title("train set confusion matrix")

mlp1_cm_test=ConfusionMatrixDisplay(confusion_matrix(y_test,mlp1.predict(
mlp1_cm_test.plot()
plt.title("test set confusion matrix")
```

```
train_score: 0.5332972653944521
test_score: 0.5283185840707965
```

Out[38]: Text(0.5, 1.0, 'test set confusion matrix')







Το πιο απλό μοντέλο με λιγότερες παραμέτρους και λιγότερα layers,στα ίδια δεδομένα,δείχνει καλύτερη απόδοση και σε crossvalidation και στο test set

Ακομα και ένα μη γραμμικό μοντέλο με περίπου **3x50x8x3=3600 βαροι και 2 hidden layers** πετυχαίνει περίπου **2% καλύτερη ακρίβεια στο test set από ότι το γραμμικό μοντέλο**

Train all companies and show some results

In [34]:
```
def calculate_cvScore(broker_name,linear_model,MLP_model):
    #file_name1='bet365-bets.csv'
    #read clean dataframes from cv(all nan dropped)
    print(f'----started for broker {broker_name}----')
    file_name1=broker_name
    #file_name2='results.csv'

    stream= os.popen('pwd') #put path of file_name
    working_dir_path=stream.read().split('\n')[0] #read the command pwd f
    absolute_path_1=working_dir_path + "/" + file_name1
    #absolute_path_2=working_dir_path + "/" + file_name2


    x_df=pd.read_csv(absolute_path_1)
    #x_df2.drop(x_df2.columns[0],axis=1)

    y_df=x_df['Result']
    x_df=x_df.drop('Result',axis=1)
    #print(x_df.head())
    y_enc=label_encoder.fit_transform(y_df)

    X_train,X_test,y_train,y_test = train_test_split(np.array(x_df),y_enc
    print("D,L,W : ",(y_train==0).sum(),(y_train==1).sum(),(y_train==2).s

    #model.fit(X_train,y_train)
    cv_score_linear=cross_val_score(estimator=linear_model,X=X_train,y=y_
    cv_score_MLP=cross_val_score(estimator=linear_model,X=X_train,y=y_tra

    print(f'----{broker_name}----')
    print('linear cross validation mean and std:',cv_score_linear.mean(),
    print('MLP cross validation mean and std:',cv_score_MLP.mean(),cv_sco
    return cv_score_linear,cv_score_MLP
```

In [52]:
```
#mlp=MLPClassifier(hidden_layer_sizes=(25,2,3),
#                   activation='relu',
#                   solver='sgd',
#                   max_iter=900,
#                   learning_rate_init=0.001,
#                   tol=0.00001)
mlp=MLPClassifier(hidden_layer_sizes=(50,8),
                  activation='relu',
                  solver='sgd',
                  max_iter=900,
                  learning_rate_init=0.001,
                  tol=0.00001)
```

In [53]:
```
for broker_name in ['BW-bets+Result.csv','IW-bets+Result.csv','LB-bets+Re
    calculate_cvScore(broker_name,sgd,mlp)
```

```
----started for broker BW-bets+Result.csv----
D,L,W :  5152 5785 9283
----BW-bets+Result.csv----
linear cross validation mean and std: 0.3805637982195845 0.08391975219640
846
MLP cross validation mean and std: 0.4133036597428289 0.07790347379831408
----started for broker IW-bets+Result.csv----
D,L,W :  5116 5822 9282
----IW-bets+Result.csv----
linear cross validation mean and std: 0.3226013847675569 0.07061859100869
892
MLP cross validation mean and std: 0.3941147378832838 0.07493828507593647
----started for broker LB-bets+Result.csv----
D,L,W :  5124 5828 9268
----LB-bets+Result.csv----
linear cross validation mean and std: 0.37942631058358056 0.0803774016071
0717
MLP cross validation mean and std: 0.37611275964391694 0.0825699463117989
4
```

Την καλύτερη ακρίβειας ως προς το cross validation δίνει η εταιρεία B365,και στα δύο μοντέλα(γραμμικό και μη)όπως φαίνεται από τα παραπανω,με ακριβεια (mean) κοντά στο 0.51.Επίσης τα γραμμικά μοντέλα,όταν δεν κάνουν overfit έχουν λίγο καλύτερη καρίβεια ταξινόμησης από το γραμμικό

In [ ]:

In [ ]:

In [ ]:

In [ ]: