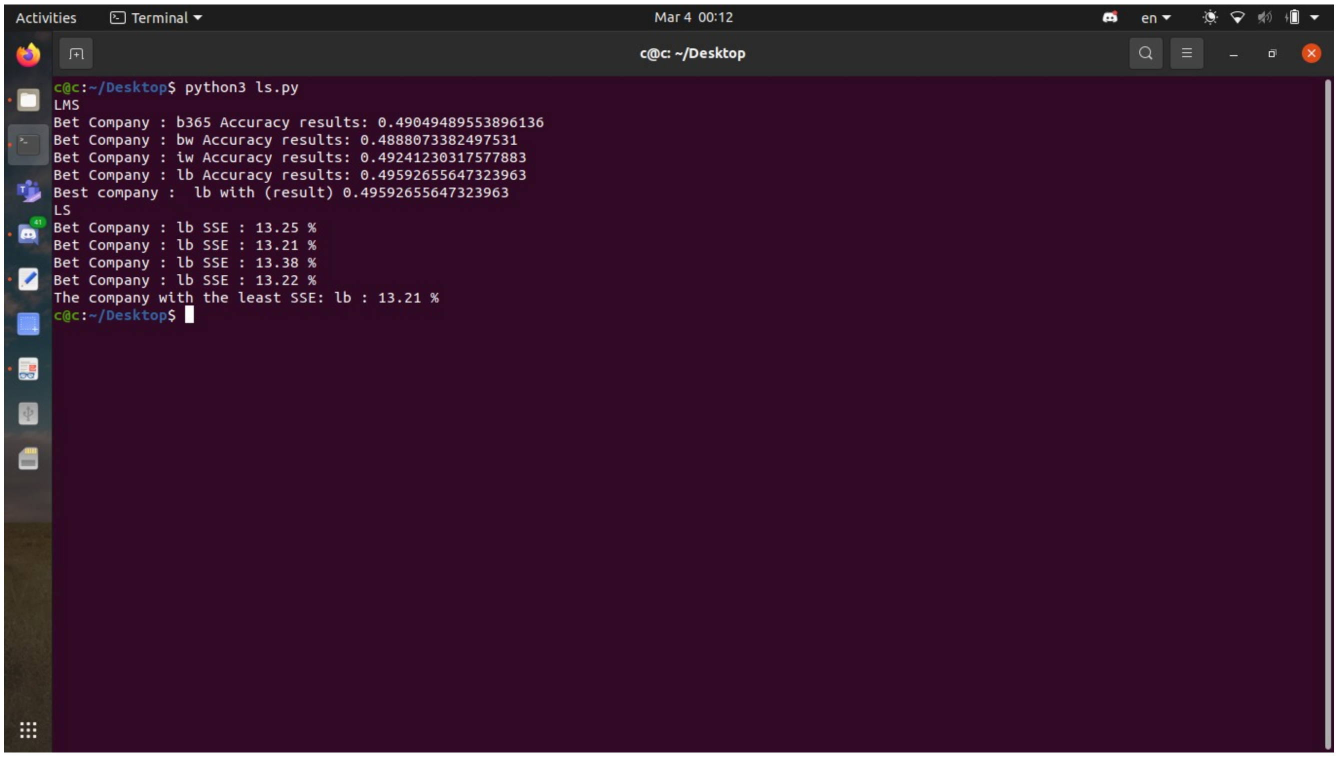
Τα ποτέλέσμτα εμφανίζονται με την σεια των ερωτημάτων
LMS για το πρώτο ερώτημα
Και Ls για το δεύτερο ερώτημα

Το πρόγραμμα γράφτηκε και εκτέλεστηκε σε Ubuntu20.04
Με τις νεότερες εκδόσεις των sklearn, numpy, sqlite
Το pandas απαιτεί την έκδοση 1.2.0(δεν λειτουργει με νεότερες εκδόσεις)



```
c@c:~/Desktop$ python3 ls.py
LMS
Bet Company : b365 Accuracy results: 0.49049489553896136
Bet Company : bw Accuracy results: 0.4888073382497531
Bet Company : iw Accuracy results: 0.49241230317577883
Bet Company : lb Accuracy results: 0.49592655647323963
Best company :  lb with (result) 0.49592655647323963
LS
Bet Company : lb SSE : 13.25 %
Bet Company : lb SSE : 13.21 %
Bet Company : lb SSE : 13.38 %
Bet Company : lb SSE : 13.22 %
The company with the least SSE: lb : 13.21 %
c@c:~/Desktop$
```

Κώδικας

```
import sqlite3
import pandas as pd
import numpy as np
import sklearn
import os.path
from sklearn.model_selection import KFold
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LinearRegression



def LMS(x,y):
    x = np.hstack((np.ones((len(x), 1)), x))
    w = np.random.rand(x.shape[1], 3)
    out , out[np.arange(len(y)), y] = np.zeros((len(y), 3)),1

    for t in range(len(out)):
        i = int(Results[t])
        out[t,i] = 1
    loss = []
    for i in range(100): #increase the value and get more accurate results (high temperatures)
        predictions = x.dot(w)

        diffs = predictions - out
        l = np.mean(np.square(diffs))
        loss.append(l)
        Gradient = x.T @ diffs
        w -= 0.000001*Gradient
    return w


connection = sqlite3.connect('database.sqlite')

match = pd.read_sql("select id, home_team_goal, away_team_goal, B365H, B365D, "
                    "B365A, BWH, BWD, BWA, IWH, IWD, IWA, LBH, LBD, LBA from Match",
connection, index_col="id").dropna(axis='rows')
attributes = pd.read_sql("Select id, team_fifa_api_id, team_api_id, date, buildUpPlaySpeed,
buildUpPlaySpeedClass, buildUpPlayDribbling, buildUpPlayDribblingClass, buildUpPlayPassing,
buildUpPlayPassingClass, chanceCreationPassing, chanceCreationPassingClass,
chanceCreationCrossing, chanceCreationCrossingClass, chanceCreationShooting ,
chanceCreationShootingClass , chanceCreationPositioningClass , defencePressure ,
defencePressureClass , defenceAggression, defenceAggressionClass , defenceTeamWidth,
defenceTeamWidthClass , defenceDefenderLineClass from
Team_Attributes",connection,index_col="id").dropna(axis='rows')


Guest_goal = match['home_team_goal'].tolist()
Home_goal = match['away_team_goal'].tolist()

b365 = match[['B365H','B365D','B365A']].values.tolist()
bw = match[['BWH','BWD','BWA']].values.tolist()
iw = match[['IWH','IWD','IWA']].values.tolist()
lb = match[['LBH','LBD','LBA']].values.tolist()


Results = []

companies = ["b365","bw","iw","lb"]


for i in range(len(Home_goal)):
    if Home_goal[i]>Guest_goal[i]:
        Results.append(1)
    elif Home_goal[i]<Guest_goal[i]:
        Results.append(0)
    else:
        Results.append(-1)
```

```
print("LMS")
max = -1
for count , companypred in enumerate([b365, bw, iw, lb]):
    kf = KFold(n_splits=10, shuffle = True)
    Acc = []
    Arr1 = np.array(companypred)
    Arr2 = np.array(Results)
    for train_index, test_index in kf.split(companypred):
        Xque , Yque = Arr1[train_index], Arr2[train_index] #X_train, Y_train
        X_test, Y_test = Arr1[test_index], Arr2[test_index]
        res = LMS(Xque,Yque)
        test_bias = np.hstack((np.ones((len(X_test),1)),X_test))
        estimates = np.dot(test_bias,res).argmax(axis=1)
        accuracy = np.mean(estimates==Y_test)
        Acc.append(accuracy)
    print("Bet Company :", companies[count], "Accuracy results:", np.mean(Acc))
    if(max < np.mean(Acc)):
        max = np.mean(Acc)
        i= count

print("Best company : ",companies[i],"with (result)",max)



#2nd
print("LS")
minimum = 100#increase the value and get more accurate results (high temperatures)
for cound , companypred in enumerate([b365, bw, iw, lb]):
    sse_arr = []
    kf = KFold(n_splits=10, shuffle=True)
    Arr1 = np.array(companypred)
    Arr2 = np.array(Results)
    for train_index, test_index in kf.split(companypred):
        Xque, Yque = Arr1[train_index], Arr2[train_index]
        X_test, Y_test = Arr1[test_index], Arr2[test_index]
        mdl = LinearRegression()

        ovr = OneVsRestClassifier(mdl).fit(Xque,Yque)
        predictions = ovr.predict(X_test)
        sse = np.sum(np.square(Y_test - predictions))
        sse_arr.append(sse/100)
    print("Bet Company :", companies[count], "SSE :", min(sse_arr),"%")
    if minimum > min(sse_arr):
        minimum = min(sse_arr)
        i = count
print("The company with the least SSE:",companies[i],":",minimum,"%")
```