ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

ΤΕΛΙΚΗ ΥΠΟΛΟΓΙΣΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΚΟΛΕΜΙ ΚΡΙΣΤΙΑΝ, Π18029

ΧΡΗΣΤΟΣ ΜΙΧΑΗΛ ΚΑΤΑΓΗΣ, Π18067

ΜΙΧΑΗΛ ΚΑΤΣΟΥΛΑΣ, Π18071

ΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ ΤΟΥ DATASET

Πρωτού ξεκινήσουμε να απαντάμε τα ερωτήματα που τίθονται από την εργασία είναι σημαντικό να φορτώσουμε τα δεδομένα της βάσης Sqlite που δίνεται στον σύνδεσμο https://www.kaggle.com/hugomathien/soccer και να τα επεξεργαστούμε με κατάλληλο τρόπο, όπως θα αναλύσουμε παρακάτω.

Ξεκινώντας, έχοντας κάνει import το library sqlite3, δημιουργούμε μια σύνδεση με τη βάση και εκτελούμε ένα query ώστε να πάρουμε τους πίνακες που αυτή περιέχει, όπως φαίνεται στις παρακάτω γραμμές κώδικα:

```
# establish connection with the database and execute query to get tables
connection = sqlite3.connect("database.sqlite")
cur = connection.cursor()
cur.execute("SELECT name FROM sqlite_master WHERE type = 'table';")

print("Database Tables: ")
print(cur.fetchall())
```

Εκτελώντας το, εκτυπώνονται οι πίνακες που περιέχονται στη βάση.

```
PS C:\Users\golem\Desktop\Pattern Recognition> python -u "c:\Users\golem\Desktop\Pattern Recognition\ergasia.p
y"
2021-09-19 12:50:58.182097: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dyn
amic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2021-09-19 12:50:58.189135: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror i
f you do not have a GPU set up on your machine.
TABLES:
[('sqlite_sequence',), ('Player_Attributes',), ('Player',), ('Match',), ('League',), ('Country',), ('Team',),
('Team_Attributes',)]
```

Εν συνεχεία, για να έχουμε πρόσβαση στους πίνακες Match και Team_Attributes, οι οποίοι είναι οι μόνοι που θα μας χρησιμεύσουν, μέσω της βιβλιοθήκης Pandas τους μετατρέπουμε σε DataFrames κάνοντας χρήση της συνάρτησης pandas.read_sql_query() δίνοντας της τα απαραίτητα ορίσματα.

Ύστερα, εκτελώντας διαδοχικά τα παρακάτω βήματα είμαστε πλέον σε θέση να ξεκινήσουμε το training. Εξάγουμε τη χρονολογία εγγραφής της κάθε ομάδας από το dataset, κρατάμε από τους πίνακες μόνο τις αναγκαίες στήλες που θα μας χρειαστούν για το training και τέλος αφαιρούμε όλες τις εγγραφές των οποίων τα διανύσματα προγνωστικών έχουν μηδενικές τιμές, όπως αναφέρετε και στην εκφώνηση της εργασίας.

Η διαδικασία που αναλύσαμε για την επεξεργασία των δεδομένων και τη φόρτωση των πινάκων φαίνεται στις παρακάτω γραμμές κώδικα:

```
# load only the tables we will use and convert them into Pandas DataFrames
match = pd.read_sql_query("SELECT * FROM Match", connection)
team_attributes = pd.read_sql_query("SELECT * FROM Team_Attributes", connection)

match['date'] = pd.to_datetime(match['date'])
team_attributes['date'] = pd.to_datetime(team_attributes['date'])
match['year'] = match['date'].dt.year
team_attributes['year'] = team_attributes['date'].dt.year

# keep only the columns that we'll need during the training phase
match_keep = [
    'year',
    'home_team_api_id', 'away_team_api_id',
    'home_team_goal', 'away_team_goal',
    'B365H', 'B365D', 'B365A',
    'BWH', 'BWD', 'BWA',
    'IWH', 'IWD', 'IWA',
    'LBH', 'LBD', 'LBA'
    ]
match = match[match_keep]
# remove all the records whose prediction vectors have value = 0
match = match.replace(np.nan, 0.0)
match = match.dropna()

team_attributes_keep = [
    'year', 'team_api_id',
    'buildUpPlaySpeed','buildUpPlayPassing',
    'chanceCreationPassing', 'chanceCreationCrossing', 'chanceCreationShooting',
    'defencePressure', 'defenceAggression','defenceTeamWidth'
    ]
team_attributes = team_attributes[team_attributes_keep]
# remove all the records whose prediction vectors have value = 0
team_attributes = team_attributes.replace(np.nan, 0.0)
team_attributes = team_attributes.dropna()

print("\n---MATCH---\n")
print(match)
print("\n---TEAM_ATTRIBUTES---   \n")
print(team_attributes)
```

Εκτελώντας τον παραπάνω κώδικα τυπώνονται στο τερματικό οι πίνακες Match και Team_Attributes, όπως φαίνεται στην παρακάτω εικόνα, όντας πλέον επεξεργασμένοι ώστε να ικανοποιούν τις απατήσεις για να μπορέσουμε να προχωρήσουμε στο training.

```
---MATCH---

       year  home_team_api_id  away_team_api_id  home_team_goal  away_team_goal  B365H  B365D  ...   BWA   IWH  IWD  IWA   LBH  LBD   LB
A
0      2008              9987              9993               1               1   1.73   3.40  ...  4.20  1.85  3.2  3.5  1.80  3.3  3.7
5
1      2008             10000              9994               0               0   1.95   3.20  ...  3.95  1.90  3.2  3.5  1.90  3.2  3.5
0
2      2008              9984              8635               0               3   2.38   3.30  ...  2.55  2.60  3.1  2.3  2.50  3.2  2.50
3      2008              9991              9998               5               0   1.44   3.75  ...  6.80  1.40  3.9  6.0  1.44  3.6  6.50
4      2008              7947              9985               1               3   5.00   3.50  ...  1.60  4.00  3.3  1.7  4.00  3.4  1.72
...     ...               ...               ...             ...             ...    ...    ...  ...   ...   ...  ...  ...   ...  ...   ...
25974  2015             10190             10191               1               0   0.00   0.00  ...  0.00  0.00  0.0  0.0  0.00  0.0  0.00
25975  2015              9824             10199               1               2   0.00   0.00  ...  0.00  0.00  0.0  0.0  0.00  0.0  0.00
25976  2015              9956             10179               2               0   0.00   0.00  ...  0.00  0.00  0.0  0.0  0.00  0.0  0.00
25977  2015              7896             10243               0               0   0.00   0.00  ...  0.00  0.00  0.0  0.0  0.00  0.0  0.00
25978  2015             10192              9931               4               3   0.00   0.00  ...  0.00  0.00  0.0  0.0  0.00  0.0  0.00

[25979 rows x 17 columns]

---TEAM_ATTRIBUTES---

      year  team_api_id  buildUpPlaySpeed  ...  defencePressure  defenceAggression  defenceTeamWidth
0     2010         9930                60  ...               50                 55                45
1     2014         9930                52  ...               47                 44                54
2     2015         9930                47  ...               47                 44                54
3     2010         8485                70  ...               60                 70                70
4     2011         8485                47  ...               47                 47                52
...    ...          ...               ...  ...              ...                ...               ...
1453  2011        10000                52  ...               46                 48                53
1454  2012        10000                54  ...               44                 55                53
1455  2013        10000                54  ...               44                 58                37
1456  2014        10000                54  ...               44                 58                37
1457  2015        10000                54  ...               44                 58                37
```

Έχοντας φορτώσει τους πίνακες Match και Team_Attributes μπορούμε πλέον να προχωρήσουμε στη διαδικασία δημιουργίας των διανυσμάτων εκπαίδευσης. Η δημιουργία του training set, λοιπόν, γίνεται μέσω της συνάρτησης train(matches, teams) η οποία παίρνοντας ως όρισμα τους δύο πίνακες που χρησιμοποιούμε, κατασκευάζει το training set το οποίο αποτελείται από τις στήλες με τα αντίστοιχα στατιστικά των στοιχηματικών εταιρειών και τα χαρακτηριστικά της κάθε ομάδας.

Η διαδικασία δημιουργίας του training set μέσω της train(matches, teams) φαίνεται στην παρακάτω εικόνα:

```
# create training vectors
def train(matches, teams):
    t_set = []
    for i, row in matches.iterrows():
        home_team_id = row['home_team_api_id']
        away_team_id = row['away_team_api_id']

        home_team_data = teams[(teams.team_api_id == home_team_id)]
        home_team_data = home_team_data.drop(columns = 'team_api_id')
        home_team_data.columns = 'HOME' + home_team_data.columns
        away_team_data = teams[(teams.team_api_id == away_team_id)]
        away_team_data = away_team_data.drop(columns = 'team_api_id')
        away_team_data.columns = 'AWAY' + away_team_data.columns

        try:
            join = pd.concat([row, home_team_data.iloc[1], away_team_data.iloc[1]], axis = 0, join = "inner")
        except:
            continue
        t_set.append(join)

    return t_set

# create training set using the above function
train_x = train(match, team_attributes)
train_x = pd.DataFrame(train_x)
train_x = train_x.drop(columns = ['year', 'home_team_api_id', 'away_team_api_id', 'AWAYyear', 'HOMEyear'])
# find the result of the game and store it in a seperate column
train_x['r'] = train_x['home_team_goal'] - train_x['away_team_goal']

print(train_x.columns)
```

Εκτελώντας το παραπάνω κομμάτι κώδικα τυπώνονται στην οθόνη οι στήλες που τελικά θα περιέχονται στο σύνολο εκπαίδευσης.

```
PS C:\Users\golem\Desktop\Pattern Recognition> python -u "c:\Users\golem\Desktop\Pattern Recognition\ergasia.py"
2021-09-19 14:26:44.372895: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic lib
rary 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2021-09-19 14:26:44.380714: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do
 not have a GPU set up on your machine.
Index(['home_team_goal', 'away_team_goal', 'B365H', 'B365D', 'B365A', 'BWH',
       'BWD', 'BWA', 'IWH', 'IWD', 'IWA', 'LBH', 'LBD', 'LBA',
       'HOMEbuildUpPlaySpeed', 'HOMEbuildUpPlayPassing',
       'HOMEchanceCreationPassing', 'HOMEchanceCreationCrossing',
       'HOMEchanceCreationShooting', 'HOMEdefencePressure',
       'HOMEdefenceAggression', 'HOMEdefenceTeamWidth', 'AWAYbuildUpPlaySpeed',
       'AWAYbuildUpPlayPassing', 'AWAYchanceCreationPassing',
       'AWAYchanceCreationCrossing', 'AWAYchanceCreationShooting',
       'AWAYdefencePressure', 'AWAYdefenceAggression', 'AWAYdefenceTeamWidth',
       'r'],
      dtype='object')
```

Τα τελευταία βήματα πρωτού προχωρήσουμε στην απάντηση των ερωτημάτων που τίθονται είναι τα παρακάτω:

Δημιουργία μιας δομής λεξικού που θα περιέχει τις στοιχηματικές εταιρείες σε συνδυασμό με το αποτέλεσμα του αγώνα, ορισμός ενός scaler που θα οριοθετεί τα δεδομένα σε ένα συγκεκριμένο εύρος τιμών και τέλος ορισμός των συναρτήσεων find_best(), find_result(). Η συναρτήσεις αυτές υπολογίζουν αντίστοιχα την καλύτερη στοιχηματική εταιρεία με βάση τα προγνωστικά της καθώς και το αποτέλεσμα ενός αγώνα κάνοντας μια απλή πράξη αφαίρεσης των τερμάτων κάθε ομάδα. Όσα προαναφέραμε φαίνονται στην παρακάτω εικόνα:

```
# create a dictionary for training based on the betting company
companies = {
    'B364': ['B365H', 'B365D', 'B365A' ,'r'],
    'BW':   ['BWH', 'BWD', 'BWA', 'r'],
    'LW':   ['IWH', 'IWD', 'IWA', 'r'],
    'LB':   ['LBH', 'LBD', 'LBA', 'r']
}

# scale the data
scaler = MinMaxScaler()
kfold = KFold(n_splits = 10)

# find the betting company with the best predictions
def find_best(accuracies):
    bookers = ['B364', 'BW', 'LW', 'LB']
    i, value = max(enumerate(accuracies), key = operator.itemgetter(1))

    return bookers[i]

# find the result of the game
def find_result(data):
    result = []
    for i, j in enumerate(data):
        if j > 0:
            result.append('H')
        elif j == 0:
            result.append('D')
        else:
            result.append('A')

    return result
```

I. Να υλοποιήσετε τον Αλγόριθμο Ελάχιστου Μέσου Τετραγωνικού Σφάλματος (Least Mean Squares), ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί την συνάρτηση διάκρισης της μορφής $g_k(\psi_k(m))$: $\mathbb{R}^3 \rightarrow \{H, D, A\}$ για κάθε στοιχηματική εταιρεία. Να αναγνωρίσετε την στοιχηματική εταιρεία τα προγνωστικά της οποίας οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης.

Για την υλοποίηση του LMS algorithm και την εύρεση της εταιρείας της οποίας τα προγνωστικά οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης εκτελείται το παρακάτω κομμάτι κώδικα, το οποίο εν συνεχεία θα αναλύσουμε.

```
# QUESTION 1
acc = []
for cmp in companies:
    train = train_x[companies[cmp]].reset_index(drop = True)
    y = train['r']
    x = scaler.fit_transform(train.drop(columns = ['r']))

    print(f'BETTING COMPANY: {cmp}')
    kfold.get_n_splits(x)
    k = 1
    accuracy = 0
    for train_i, test_i in kfold.split(x):
        x_train, x_test = x[train_i], x[test_i]
        y_train, y_test = y[train_i], y[test_i]
        y_train = np.array(y_train)

        model = Sequential()
        model.add(Dense(1, activation = "relu"))
        model.compile(
            loss = 'mean_squared_error',
            optimizer = 'adam',
            metrics = ['accuracy']
            )
        model.fit(x_train, y_train, epochs = 3)

        prd = scaler.transform(x_test)
        prd = model.predict(prd)
        prd = find_result(prd)
        y_test = find_result(y_test)
        print(f'Fold Number: {k}, {accuracy_score(y_test, prd)}')
        accuracy = accuracy + accuracy_score(y_test, prd)
        k = k + 1
    k = 0
    accuracy = accuracy / 10
    acc.append(accuracy)

print(f'The best betting company is: {find_best(acc)}')
```

Για την υλοποίση του LMS, όπως φαίνεται από τον κώδικα, δημιουργήσαμε ένα νευρωνικό δίκτυο perceptron, που χρησιμοποιεί τον αλγόριθμο gradient descent για τη βελτιστοποίηση των βαρών και τη συνάρτηση Mean Squared Error ως συνάρτηση κόστους. Επιπλέον, ως συνάρτηση ενεργοποίησης του νευρωνικού, χρησιμοποιήσαμε την relu η οποία είναι ιδανική για προβλήματα ταξινόμησης και για κάθε περίπτωση χρησιμοποιούνται 3 epochs για την εκπαίδευση του δικτύου.Μέσω του sequential neural network που κατασκευάσαμε σε συνδυασμό με τις συναρτήσεις και τα δεδομένα που ορίσαμε προηγουμένως, είμαστε σε θέση να βρούμε τη στοιχηματική εταιρεία με τα καλύτερα προγνωστικά.

Παρακάτω παρατίθεται εικόνα από την εκτέλεση του κώδικα για την απάντηση του ερωτήματος (μέρος του αποτελέσματος με έμφαση στην τελική απάντηση για την καλύτερη εταιρεία). Όπως φαίνεται, καλύτερη στοιχηματική εταιρεία με βάση τα σκορ και την ακρίβεια του νευρωνικού είναι η BW.

```
700/700 [==============================] - 0s 427us/step - loss: 3.0418 - accuracy: 0.2638
Fold Number: 5, 0.47023330651649237
Epoch 1/3
700/700 [==============================] - 1s 479us/step - loss: 3.0111 - accuracy: 0.2584
Epoch 2/3
700/700 [==============================] - 0s 465us/step - loss: 2.9292 - accuracy: 0.2655
Epoch 3/3
700/700 [==============================] - 0s 455us/step - loss: 2.8833 - accuracy: 0.2678
Fold Number: 6, 0.4754625905068383
Epoch 1/3
700/700 [==============================] - 1s 446us/step - loss: 3.1426 - accuracy: 0.2588
Epoch 2/3
700/700 [==============================] - 0s 455us/step - loss: 3.0672 - accuracy: 0.2630
Epoch 3/3
700/700 [==============================] - 0s 490us/step - loss: 3.0027 - accuracy: 0.2641
Fold Number: 7, 0.45132743362831856
Epoch 1/3
700/700 [==============================] - 1s 462us/step - loss: 3.3300 - accuracy: 0.2535
Epoch 2/3
700/700 [==============================] - 0s 442us/step - loss: 3.3300 - accuracy: 0.2535
Epoch 3/3
700/700 [==============================] - 0s 435us/step - loss: 3.3300 - accuracy: 0.2535
Fold Number: 8, 0.2578439259855189
Epoch 1/3
700/700 [==============================] - 1s 506us/step - loss: 3.2860 - accuracy: 0.2569
Epoch 2/3
700/700 [==============================] - 0s 452us/step - loss: 3.2858 - accuracy: 0.2569
Epoch 3/3
700/700 [==============================] - 0s 443us/step - loss: 3.2858 - accuracy: 0.2569
Fold Number: 9, 0.22727272727272727
Epoch 1/3
700/700 [==============================] - 1s 446us/step - loss: 3.2622 - accuracy: 0.2558
Epoch 2/3
700/700 [==============================] - 0s 424us/step - loss: 3.2622 - accuracy: 0.2558
Epoch 3/3
700/700 [==============================] - 0s 456us/step - loss: 3.2622 - accuracy: 0.2558
Fold Number: 10, 0.2370221327967807
The best betting company is: BW
```

II. Να υλοποιήσετε τον Αλγόριθμο Ελάχιστου Τετραγωνικού Σφάλματος (**Least Squares**), ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί την συνάρτηση διάκρισης της μορφής $g_k(\psi_\kappa(m))$: $\mathbb{R}^3 \rightarrow \{H, D, A\}$ για κάθε στοιχηματική εταιρεία. Να αναγνωρίσετε την στοιχηματική εταιρεία τα προγνωστικά της οποίας οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης.

Για την υλοποίηση του Least Squares algorithm και την εύρεση της εταιρείας της οποίας τα προγνωστικά οδηγούν σε μεγαλύτερη ακρίβεια ταξινόμησης εκτελείται το παρακάτω κομμάτι κώδικα, το οποίο εν συνεχεία θα αναλύσουμε.

```
# QUESTION 2
regression = linear_model.LinearRegression()
acc = []
for cmp in companies:
    train = train_x[companies[cmp]].reset_index(drop = True)
    y = train['r']
    x = scaler.fit_transform(train.drop(columns = ['r']))

    print("BETTING COMPANY: " + cmp)
    kfold.get_n_splits(x)
    k = 1
    accuracy = 0
    for train_index, test_index in kfold.split(x):
        x_train, x_test = x[train_index], x[test_index]
        y_train, y_test = y[train_index], y[test_index]
        regression.fit(x_train, y_train)

        prd = scaler.transform(x_test)
        prd = regression.predict(prd)
        prd = find_result(prd)
        y_test = find_result(y_test)
        print(f'Fold Number: {k}, {accuracy_score(y_test, prd)}')
        accuracy = accuracy + accuracy_score(y_test, prd)
        k = k + 1
    k = 0
    accuracy = accuracy / 10
    acc.append(accuracy)

print(f'The best betting company is: {find_best(acc)}')
```

Ο παραπάνω αλγόριθμος αφορά συνήθως προβλήματα ελαχιστοποίησης, έχοντας ως στόχο να βρεθεί η καλύτερη ευθεία από την οποία η απόσταση από όλα τα σημεία του χώρου ελαχιστοποιείται. Για τον σκοπό αυτό και την απάντηση του ερωτήματος ΙΙ, χρησιμοποιήθηκε η συνάρτηση LinearRegression(), σε συνδυασμό με τις συναρτήσεις και τις μεταβλητές που έχουμε ήδη ορίσει στον κώδικα μας. Με αυτόν τον τρόπο μπορούμε να υλοποιήσουμε τη συναρτήση διάκρισης που ζητείται και να βρούμε την ιδανική εταιρεία προγνωστικών με βάση τα σκορ ακρίβειας της.

Από την εκτέλεση του παραπάνω κώδικα, παίρνουμε το εξής αποτέλεσμα που υποδεικνύει πώς η καλύτερη στοιχηματική εταιρεία είναι η B364.

```
BETTING COMPANY: B364
Fold Number: 1, 0.4754625905068383
Fold Number: 2, 0.4440868865647627
Fold Number: 3, 0.4481094127111826
Fold Number: 4, 0.4557522123893805
Fold Number: 5, 0.47023330651649237
Fold Number: 6, 0.4754625905068383
Fold Number: 7, 0.45132743362831856
Fold Number: 8, 0.42920353982300885
Fold Number: 9, 0.48069187449718426
Fold Number: 10, 0.4659959758551308
BETTING COMPANY: BW
Fold Number: 1, 0.4754625905068383
Fold Number: 2, 0.4440868865647627
Fold Number: 3, 0.4481094127111826
Fold Number: 4, 0.4557522123893805
Fold Number: 5, 0.47023330651649237
Fold Number: 6, 0.4754625905068383
Fold Number: 7, 0.45132743362831856
Fold Number: 8, 0.42920353982300885
Fold Number: 9, 0.48069187449718426
Fold Number: 10, 0.4659959758551308
BETTING COMPANY: LW
Fold Number: 1, 0.4754625905068383
Fold Number: 2, 0.4440868865647627
Fold Number: 3, 0.4481094127111826
Fold Number: 4, 0.4557522123893805
Fold Number: 5, 0.47023330651649237
Fold Number: 6, 0.4754625905068383
Fold Number: 7, 0.45132743362831856
Fold Number: 8, 0.42920353982300885
Fold Number: 9, 0.48069187449718426
Fold Number: 10, 0.4659959758551308
BETTING COMPANY: LB
Fold Number: 1, 0.4754625905068383
Fold Number: 2, 0.4440868865647627
Fold Number: 3, 0.4481094127111826
Fold Number: 4, 0.4557522123893805
Fold Number: 5, 0.47023330651649237
Fold Number: 6, 0.4754625905068383
Fold Number: 7, 0.45132743362831856
Fold Number: 8, 0.42920353982300885
Fold Number: 9, 0.48069187449718426
Fold Number: 10, 0.4659959758551308
The best betting company is: B364
```

III. Να υλοποιήσετε ένα πολυστρωματικό νευρωνικό δίκτυο, ώστε ο εκπαιδευμένος ταξινομητής να υλοποιεί μια συνάρτηση διάκρισης της μορφής $g(\boldsymbol{\Phi}(\boldsymbol{m}))$: $\mathbb{R}^{28} \rightarrow \{\boldsymbol{H}, \boldsymbol{D}, \boldsymbol{A}\}$, όπου το $\boldsymbol{\Phi}(\boldsymbol{m}) \in \mathbb{R}^{28}$ αντιστοιχεί στο πλήρες διάνυσμα χαρακτηριστικών του κάθε αγώνα που δίνεται από την σχέση:

$$\boldsymbol{\Phi}(\boldsymbol{m}) = [\boldsymbol{\varphi}(\boldsymbol{h}), \boldsymbol{\varphi}(\boldsymbol{\alpha}), \boldsymbol{\psi}_{B365}(\boldsymbol{m}), \boldsymbol{\psi}_{BW}(\boldsymbol{m}), \boldsymbol{\psi}_{IW}(\boldsymbol{m}), \boldsymbol{\psi}_{LW}(\boldsymbol{m})]$$

Για την απάντηση του παραπάνω ερωτήματος έχει αναπτυχθεί το κομμάτι κώδικα που φαίνεται στην εικόνα που ακολουθεί:

```
# QUESTION 3
y = train_x['r']
x = scaler.fit_transform(train_x.drop(columns = ['home_team_goal', 'away_team_goal', 'r']))

kfold.get_n_splits(x)
k = 1
accuracy = 0
for train_index, test_index in kfold.split(x):
    x_train, x_test = x[train_index], x[test_index]
    y_train, y_test = y[train_index], y[test_index]
    y_train = np.array(y_train)

    model = Sequential()
    model.add(Dense(14, activation = "sigmoid"))
    model.add(Dense(14, activation = "sigmoid"))
    model.add(Dense(1, activation = "sigmoid"))
    model.compile(
        loss = 'mean_squared_error',
        optimizer = 'adam',
        metrics = ['accuracy']
        )
    model.fit(x_train, y_train, epochs = 3)

    prd = scaler.transform(x_test)
    prd = model.predict(prd)
    prd = find_result(prd)
    y_test = find_result(y_test)
    print(f'Fold Number: {k}, {accuracy_score(y_test, prd)}')
    accuracy = accuracy + accuracy_score(y_test, prd)
    k += 1
```

Για τη δημιουργία του νευρωνικού δικτύου χρησιμοποιείται η βιβλιοθήκη Tensorflow της Python, όπως έγινε και στα προηγούμενα ερωτήματα, σε συνδυασμό με τις συναρτήσεις και όσα έχουμε ορίσει νωρίτερα στον κώδικα.

Εκτελώντας το παραπάνω κομμάτι τυπώνεται στο τερματικό για κάθε φάση εκπαίδευσης το αντίστοιχο σκορ ακριβείας και πληροφορίες σχετικά με τη διαδικασία training. Παρακάτω παρατίθεται κομμάτι του αποτελέσματος από την εκτέλεση:

```
Fold Number: 5, 0.47023330651649237
Epoch 1/3
700/700 [==============================] - 1s 1ms/step - loss: 3.0897 - accuracy: 0.2541
Epoch 2/3
700/700 [==============================] - 1s 1ms/step - loss: 2.9245 - accuracy: 0.2661
Epoch 3/3
700/700 [==============================] - 1s 1ms/step - loss: 2.8193 - accuracy: 0.2632
Fold Number: 6, 0.4754625905068383
Epoch 1/3
700/700 [==============================] - 2s 1ms/step - loss: 3.1663 - accuracy: 0.2559
Epoch 2/3
700/700 [==============================] - 1s 1ms/step - loss: 2.9733 - accuracy: 0.2620
Epoch 3/3
700/700 [==============================] - 1s 1ms/step - loss: 2.8801 - accuracy: 0.2610
Fold Number: 7, 0.45132743362831856
Epoch 1/3
700/700 [==============================] - 1s 1ms/step - loss: 3.1455 - accuracy: 0.2546
Epoch 2/3
700/700 [==============================] - 1s 1ms/step - loss: 3.0048 - accuracy: 0.2630
Epoch 3/3
700/700 [==============================] - 1s 1ms/step - loss: 2.8985 - accuracy: 0.2597
Fold Number: 8, 0.42920353982300885
Epoch 1/3
700/700 [==============================] - 1s 1ms/step - loss: 3.1242 - accuracy: 0.2540
Epoch 2/3
700/700 [==============================] - 1s 1ms/step - loss: 2.9664 - accuracy: 0.2651
Epoch 3/3
700/700 [==============================] - 1s 1ms/step - loss: 2.8513 - accuracy: 0.2642
Fold Number: 9, 0.48069187449718426
Epoch 1/3
700/700 [==============================] - 1s 1ms/step - loss: 3.0894 - accuracy: 0.2565
Epoch 2/3
700/700 [==============================] - 1s 1ms/step - loss: 2.9060 - accuracy: 0.2651
Epoch 3/3
700/700 [==============================] - 1s 1ms/step - loss: 2.8064 - accuracy: 0.2631
Fold Number: 10, 0.4659959758551308
```