# QUIZ 2 : GAME ANALYTICS

## BADS7205 DATA STREAMING AND REAL TIME ANAYTICS


Team SKY

นาย กชกร เลื่อนสุขสันต์ รหัส 6310412001

นาย สลิลวสุ เที่ยงธรรม รหัส 6310412017

นางสาว สุพัตรา ต้งัสกุลระหง รหัส 6310412028

# publish_online_score

{ ‘user’ : ‘Waew’,
‘score’ : 54,
‘feature’: [431.2692307692308,241.0,31,23,28,5,4,94,72,43,51] }

NETPIE

```
XX = [aa0, aa1, aa2, aa3, aa4, aa5, aa6, aa7, aa8, aa9, aa10]
# print(XX)
```

```
def publish_online_score(score, name, game_over,XX):
    global publish_time, microgear
    sending_interval = 1
    now = datetime.now()
    if (now - publish_time).seconds >= sending_interval:
        data = dict(user=name, score=score, feature=XX)
        microgear.publish(user_score_topic, json.dumps(data))
        publish_time = now
```

# Subscription

```
score_list = []
Feature_Total = []
publish_time = datetime.now()
```

```
def subscription(topic, message):
    # print(message)
    import ast
    global score_list,cc


    try:
        if topic == f"/{appid}{user_score_topic}" and message:
            score = json.loads(ast.literal_eval(message).decode('utf-8'))
            # print(score)
            # print(type(score))
            score_list.append(score)
            # print(f'score list : {score_list}')
            n = score['user']
            # print(n)
            l = score['feature']
            l.insert(0,n)
            # print(l)
            cc+=1
            # print(cc)
            if cc%1 == 0:
                Feature_Total.append(l)

            # print(f'Feature_Total : {Feature_Total}')
    except Exception:
        pass
    logging.info(topic + " " + message)
```

NETPIE

{ ‘user’ : ‘Waew’,

‘score’ : 54,

‘feature’: [431.2692307692308,241

,31,23,28,5,4,94,72,43,51] }

## Feature_Total

[['Waew', 365.27777777777777, 230.30555555555554, 8, 9, 9, 0, 2, 26, 18, 19, 18],
['Waew', 368.0, 500, 0, 0, 0, 0, 1, 1, 0, 5, 5],
['Waew', 345.25, 484.25, 0, 0, 0, 0, 1, 2, 1, 5, 5],
['Moss', 368.0, 500, 0, 0, 1, 1, 1, 1, 0, 5, 5],
['Waew', 368.0, 409.0, 0, 0, 0, 0, 1, 3, 1, 5, 5],
['Moss', 345.25, 500, 0, 0, 2, 2, 1, 2, 0, 5, 5],
['Waew', 347.875, 371.375, 0, 1, 1, 0, 1, 4, 1, 6, 5],
['Moss', 326.0, 500, 0, 0, 3, 3, 1, 4, 0, 5, 5],
['Waew', 384.8, 348.8, 0, 2, 2, 0, 1, 5, 2, 6, 5],
['Moss', 321.625, 500, 0, 0, 3, 3, 1, 4, 0, 5, 5],
['Waew', 403.0, 304.0, 2, 2, 2, 0, 1, 6, 4, 7, 7]]

# Select Feature

- A0) Position in X axis => position X [1, 2, 3, 2, 1] / 5
- A1) Position in Y axis => position Y [200, 150, 130, 170] / 4
- A2) Number of coins collected => Total
- A3) Number of destroyed enemies => Total
- A4) Number of shots => Total
- A9) Number of enemy created => Total
- A10) Number of coin created => Total

โดยใช้ค่าแตกต่างระหว่าง 1 วินาที

**วินาทีแรก** Waew,430.1796875,242.421875,30,23,28,5,4,92,71,43,50

**วินาทีถัดมา** Waew,431.2692307692308,241.0, 31,23,28,5,4,94,72,43,51

1.09, 0, 1, 0, 0 ,0.53, 0.6078

```python
def create_x_name(Feature_Total):
    print(Feature_Total)
    emtry_list_name = []
    emtry_list = []
    check_name =[]
    nname= []

    indices = [1,2,3, 4,5,10,11]
    for i,j in enumerate(Feature_Total):
        # print(j)

        if len(j) > 1 and j[0] not in check_name:
            nname.append(j[0])
            check_name.append(j[0])
            selected_elements = [j[index] for index in indices]

        elif len(j) > 1 and j[0] in check_name:

            emtry_list_name.append(j[0])
            selected_elements = [j[index] for index in indices]
            index_n =max([c for c, n in enumerate(nname) if n ==j[0] ] )
            prev_Feature = [Feature_Total[index_n][index] for index in indices]
            difference = []
            zip_object = zip(selected_elements, prev_Feature)
            for a, b in zip_object:
                if a-b < 0:
                    difference.append(0)
                else:
                    difference.append(a-b)

            difference[5] = selected_elements[3]/selected_elements[5]
            difference[6] = selected_elements[2]/selected_elements[6]

            emtry_list.append(difference)

            nname.append(j[0])
        else:
            nname.append(j[0])

    return emtry_list_name, emtry_list
```

# Predict_user_type_realtime (1)

Use K-mean for 4 group Clustering

```
def prediction_user_type_realtime(Feature_Total,PLAYER_NAME):


    all_Name, X = create_x_name(Feature_Total)
    k_means = cluster.KMeans(n_clusters=4, halflife=0.4, sigma=3, seed=0)


    for i, (x, _) in enumerate(stream.iter_array(X)):
        k_means = k_means.learn_one(x)

    collect_somthing = []
    for i,j in enumerate(X):

        y = k_means.predict_one ({0:X[i][0], 1:X[i][1], 2:X[i][2] , 3:X[i][3] ,4:X[i][4], 5:X[i][5] ,6:X[i][6]})
        # y = clustream.predict_one({ 0:X[i][0], 1:X[i][1], 2:X[i][2], 3:X[i][3] ,4:X[i][4], 5:X[i][5], 6:X[i][6],
        j.insert(0,all_Name[i])
        j.insert(len(j),y)
        collect_somthing.append(j)
    return Fine_most_user_type(collect_somthing,PLAYER_NAME)
```

## List Collect_somthing

| | Name | A0 | A1 | A2 | A3 | A4 | A5 | A6 | Y_predict |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Ohm | 6.4750 | 0.000000 | 0 | 0 | 0 | 0.000000 | 0.166667 | 1 |
| 1 | Waew | 29.4000 | 0.000000 | 0 | 1 | 1 | 0.166667 | 0.166667 | 0 |
| 2 | Waew | 0.0000 | 0.000000 | 2 | 0 | 0 | 0.166667 | 0.375000 | 1 |
| 3 | Moss | 0.0000 | 0.000000 | 0 | 0 | 0 | 0.166667 | 0.375000 | 1 |
| 4 | Waew | 3.9375 | 0.000000 | 2 | 1 | 1 | 0.285714 | 0.500000 | 1 |
| 5 | Ohm | 0.0000 | 0.000000 | 0 | 1 | 1 | 0.375000 | 0.500000 | 1 |
| 6 | Waew | 0.0000 | 9.163636 | 0 | 1 | 1 | 0.444444 | 0.500000 | 3 |
| 7 | Moss | 0.0000 | 9.386364 | 0 | 0 | 0 | 0.444444 | 0.500000 | 3 |

# Predict_user_type_realtime (2)

Find Most User Type

Rank by mean A2 and A3

| y | Name | A0 | A1 | A2 | A3 | A4 | A5 | A6 |
|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 2 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 |
| 3 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |

{2: 'Hardcore Achiever’,
3: 'Hardcore Killer’,
1: 'Casual Achiever’,
0: 'Casual Killer'}

User Type

'Hardcore Achiever’

```python
def Fine_most_user_type(collect_somthing,PLAYER_NAME):
    import pandas as pd
    import numpy as np

    Total_df = pd.DataFrame (collect_somthing, columns = ['Name','A0','A1','A2','A3','A4','A5','A6','y'])

    predict_user_type = Total_df.loc[Total_df['Name'] == PLAYER_NAME ].groupby('y').count()
    # print(predict_user_type)
    index_group = predict_user_type.index.values.tolist()


    mean_coin = list(Total_df.groupby(['y']).mean()['A2'])
    mean_enemies = list(Total_df.groupby(['y']).mean()['A3'])
    sorted_index_coin = np.argsort(mean_coin).tolist()[::-1]
    sorted_index_enemies = np.argsort(mean_enemies).tolist()[::-1]

    new_label_coin = []
    for i in sorted_index_coin:
        new_label_coin.append(index_group[i])

    new_label_ememies = []
    for i in sorted_index_enemies:
        new_label_ememies.append(index_group[i])
    Full_label = [0,1,2,3]
    labels = []
    label_dummy = []
    if sorted_index_coin != sorted_index_enemies:
        if sorted_index_coin[0] != sorted_index_enemies[0]:
            labels.append(new_label_coin[0])
            labels.append(new_label_ememies[0])
            label_dummy.append(sorted_index_coin[0])
            label_dummy.append(sorted_index_enemies[0])
            M_list = list(set(sorted_index_coin) - set(label_dummy))

            d_label = []
            for i in M_list:
                if mean_coin[i] >mean_enemies[i] and mean_coin[i] != mean_enemies[i]:
                    label_dummy.append(sorted_index_coin[i])

                    labels.append(index_group[i])
                else:
                    d_label =labels+[index_group[i]]
                    m_index = list(set(Full_label)-set(d_label))
                    labels = labels+m_index
                    pass
            list_difference = list(set(Full_label)-set(labels))
            labels = labels+list_difference
        else:
            if mean_coin[sorted_index_coin[0]] >mean_enemies[sorted_index_enemies[0]]:
                labels.append(new_label_coin[0])
                labels.append(new_label_ememies[1])
                M_list = list(set(sorted_index_coin) - set(labels))
                d_label = []
                for i in M_list:
                    if mean_coin[i] >mean_enemies[i] and mean_coin[i] != mean_enemies[i]:
                        label_dummy.append(sorted_index_coin[i])

                        labels.append(index_group[i])
                    else:
                        d_label =labels+[index_group[i]]
                        m_index = list(set(Full_label)-set(d_label))
                        labels = labels+m_index

                list_difference = list(set(Full_label)-set(labels))
                labels = labels+list_difference
    else:
        if len(sorted_index_coin) == len(Full_label):
            labels = sorted_index_coin
        else:
            new_label_coin = []
            for i in sorted_index_coin:
                new_label_coin.append(index_group[i])
            M_list = list(set(sorted_index_coin) - set(new_label_coin))
            labels = new_label_coin+M_list
    LABELS = {
    labels[0]: 'Hardcore Achiever',
    labels[1]: 'Hardcore Killer',
    labels[2]: 'Casual Achiever',
    labels[3]: 'Casual Killer',
    }
    # print(LABELS)
    column = predict_user_type["A0"]
    max_index = column.idxmax()
    return LABELS.get(max_index)
```

# Game_Over

```python
def show_game_over(screen_sizeX, screen_sizeY, score, high_score, coin_count, user_type,user_type_realtime):

    # Move enemies below screen (is there a better way?)
    for i in range(num_of_enemies):
        enemy[i].posY = screen_sizeY + 100

    # Display text and score
    message_display_center('GAME OVER', font_large, yellow, int(screen_sizeX/2), int(screen_sizeY * 3/20))
    message_display_center('You\'re a', font_medium, yellow, int(screen_sizeX/2), int(screen_sizeY * 4/15))

    message_display_center(user_type, font_medium, green, int(screen_sizeX/2), int(screen_sizeY * 5/15))
    message_display_center('For use K-mean +Randomforest', font_medium, green, int(screen_sizeX/2), int(scree

    message_display_center(user_type_realtime, font_medium, green, int(screen_sizeX/2), int(screen_sizeY * 8/
    message_display_center('For use K-means River ', font_medium, green, int(screen_sizeX/2), int(screen_size

    message_display_center('Score: ' + str(score), font_medium, yellow, int(screen_sizeX/2), int(screen_sizeY
    message_display_center('Coins: ' + str(coin_count), font_medium, yellow, int(screen_sizeX/2), int(screen_
    message_display_center('Highscore: ' + str(high_score), font_medium, yellow, int(screen_sizeX/2), int(scr
    message_display_center('Press Home to continue', font_medium, yellow, int(screen_sizeX/2), int(screen_siz
```