การศึกษาความสัมพันธ์ของเกมประเภทต่างๆจากเว็บไซต์ Steam

สมาชิก

1)นายมีชัย หนูพิศ	613020594-0
2)นายอรรถพงษ์ หลักคำ	613020605-1
3)นายชาคริต น้อยดวงศรี	613020986-3
4)นางสาวศศิธร วงษานุทัศน์	613021006-8
5)นายอรรถพล วงศ์สสะอาด	613021013-1

บทที่ 1 บทนำ

STEAM®

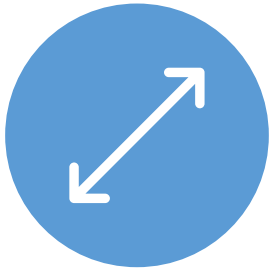
1.ที่มาและความสำคัญ

เว็บไซต์ steam เป็นเว็บไซต์ซื้อขายเกมมี่ได้รับความนิยมอย่างแพร่หลาย ทั้งในประเทศไทยและในต่างประเทศ ซึ่งมีจำนวนผู้เฉลี่ยใช้งานมากกว่า 10 ล้านคนต่อวัน เนื่องจากมีประเภทของเกมและราคาที่หลากหลาย ให้ผู้ใช้งานได้เลือกซื้อเลือกเล่น ทางคณะผู้จัดทำโครงงานจึงต้องการศึกษาถึงเกมแต่ละประเภทว่ามีปัจจัยใดบ้างที่ส่งผลต่อจำนวนผู้เล่นและทำให้เกมนั้น ๆ ได้รับความนิยม

2.วัตถุประสงค์

2.1 เพื่อรวบรวมข้อมูลประเภทของเกมและศึกษาความสัมพันธ์ของตัวแปรต่าง ๆ จากเว็บไซต์ STEAM โดยศึกษาทั้งหมด 11 ประเภท

2.2 เพื่อทำการวิเคราะห์ว่าเกมประเภทใดได้รับความนิยมจากผู้เล่นและสร้างรายได้มากที่สุด



2.3 เพื่อศึกษาว่าเกมที่ได้รับความนิยมและสร้างรายได้มากที่สุด มีปัจจัยอะไรบ้างที่ส่งผลต่อจำนวนผู้เล่น และทำการทำนายผลจากปัจจัยนั้น ๆ

3.ขอบเขตของโครงงาน

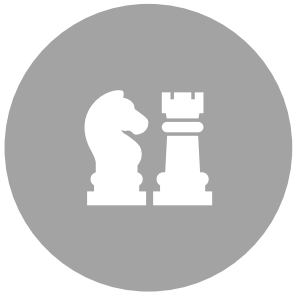
- ผู้ศึกษาได้กำหนดกลุ่มข้อมูลเป้าหมายสำหรับการดำเนินการดังนี้
- 3.1 ประเภทของเกม 11 ประเภท ได้แก่

Action, Adventure, Early Access, Ex Early Access, Free, Massively, RPG, Simulation, Sport, Strategy, Indie

3.2 ข้อมูลสำคัญ (Features) ของเกมแต่ละประเภท ได้แก่

- 1. App Id : หมายเลข ID ของเกม
- 2. Game : ชื่อเกม
- 3. Developer : ผู้พัฒนาเกม
- 4. Publisher : บริษัทผู้จัดจำหน่ายเกม
- 5. Year : ปีที่วางจำหน่ายเกม
- 6. Price : ราคาของเกม
- 7. Meta score : คะแนนความนิยมของเกม
- 8. Playtime : เวลามที่ผู้เล่นใช้เล่นเกม
- 9. Owner : จำนวนผู้ที่เล่นเกม
- 10. value : มูลค่าของเกม (Price x Owner)

4.ผลที่คาดว่าจะได้รับ



4.1 ได้ทราบถึงความสัมพันธ์และปัจจัยของเกมประเภทต่างๆ ที่ส่งผลต่อความนิยมและรายได้ของเกม

4.2 ได้ทำการเรียนรู้และเข้าใจ Data Science Process ทั้ง 5 ขั้นตอน

4.3 การประยุกต์ใช้ Machine Learning กับโครงงาน

บทที่ 2 เอกสารและงานวิจัยที่เกี่ยวข้อง

STEAM®

1.ขั้นตอนในการทำ Data Science (OSEMN)

1.OBTAIN

คือการเก็บข้อมูล

2.SCRUB

คือการทำความสะอาดข้อมูล
CLEANING DATA
หรือ DATA
WRANGLING

3.EXPLORE

คือการค้นหาสิ่งที่น่าสนใจ
และทำการ
VISUALIZE ข้อมูล

4.MODEL

คือการสร้างโมเดล เช่น โมเดล
ทำนายผลหรือ
PREDICTIVE
MODEL

5.INTERPRET

คือการนำข้อมูลมาเล่าให้เป็น
เรื่องราวและให้น่าสนใจ

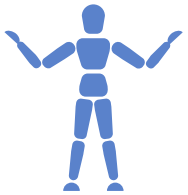
2.Random Forest

2.1ความหมายของ Random Forest

หลักการของ Random Forest คือ สร้าง model จาก Decision Tree หลายๆ model ย่อยๆ (ตั้งแต่ 10 model ถึง มากกว่า 1000 model) โดยแต่ละ model จะได้รับ data set ไม่เหมือนกัน ซึ่งเป็น subset ของ data set ทั้งหมด ตอนทำ prediction ก็ให้แต่ละ Decision Tree ทำ prediction ของใครของมัน และคำนวณผล prediction ด้วยการ vote output ที่ ถูกเลือกโดย Decision Tree มากที่สุด (กรณี classification) หรือ หาค่า mean จาก output ของแต่ละ Decision Tree (กรณี regression) Decision Tree แต่ละ model ใน Random Forest ถือว่าเป็น weak learner — ประมาณว่าเป็น model ที่ไม่เก่งเท่าไหร่ แต่พอนำเอาแต่ละ Decision Tree มาทำ prediction ร่วมกัน ก็จะได้ model รวมที่มีความเก่ง และแม่นยำมากกว่า Decision Tree ที่ทำ prediction แบบเดี่ยวๆ

2.2หลักการทำงานของ Random Forest

1.sample ข้อมูล (bootstrapping) จาก data set ทั้งหมด ให้ได้ข้อมูลออกมา n ชุด ที่ไม่เหมือนกัน ตามจำนวน Decision Tree ใน Random Forest เช่น data set ตั้งต้นมีอยู่ 10 feature (X1,X2,…,X10) แต่ละ Decision Tree จะได้ feature ไปไม่เหมือนกัน และ จะได้ข้อมูลไม่ครบทุก row ด้วยจาก data set ทั้งหมดด้วย (X1 -> X1',X2->X2',…)

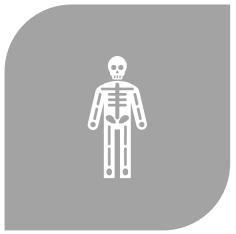


2.สร้าง model Decision Tree สำหรับแต่ละชุดข้อมูล

3.ทำ aggregation ผลลัพธ์จากแต่ละ model (bagging) เช่น voting ในกรณี classification หรือ หาค่า mean ในกรณี regression

2.3ข้อดีของ Random Forest

1.RANDOM FOREST ใช้ได้ทั้งกับปัญหา CLASSIFICATION และ REGRESSION



2.RANDOM FOREST ใช้ได้ทั้งกับข้อมูล STRUCTURED (ข้อมูลลักษณะเป็น COLUMN/ TABLE) และ UNSTRUCTURED (เช่น รูปภาพ, TEXT)

3.ทำ HYPER-PARAMETER TUNING ให้ RANDOM FOREST ไม่ OVERFIT ไม่ยาก

4.RANDOM FOREST ไม่ตั้ง ASSUMPTION กับ FEATURE ว่าจะต้องกระจายข้อมูลแบบ NORMAL DISTRIBUTION, หรือสัมพันธ์กับ TARGET แบบ LINEAR, และ ไม่ต้องสร้างความสัมพันธ์ระหว่าง FEATURE เพิ่มเติม (เรียกว่า INTERACTION — เช่น สร้าง FEATURE X_1*X_2 จาก X_1 และ X_2)

5.จากข้อ 4 ประหยัดแรงทำ FEATURE ENGINEERING เช่น ไม่จำเป็นต้องทำ LOG TRANSFORM, หรือสร้าง INTERACTION จาก FEATURE

บทที่ 3วิธีการดำเนินงาน

1.แหล่งข้อมูล

ข้อมูลของเกมทั้ง 11 ประเภทจากเว็บไซต์
https://steamspy.com/

2.ขั้นตอนและวิธีการดำเนินโครงงาน

2.1 กำหนดขั้วข้อโครงงาน วัตถุประสงค์ และขอบเขตการทำโครงงาน

2.2 สืบค้นข้อมูล และเตรียมข้อมูลสำหรับนำเสนอโครงงาน

2.3 ดำเนินโครงงาน โดยผู้จัดทำได้เริ่มดำเนินโครงงานในวันที่ 25 กันยายน พ.ศ.2562

3.ระยะเวลาในการดำเนินโครงงาน

การดำเนินงาน	สัปดาห์ที่								
	กันยายน	ตุลาคม				พฤศจิกายน			
	4	1	2	3	4	1	2	3	4
1.กำหนดหัวข้อโครงงาน									
2.ส่งโครงงร่างโครงงาน (Project Proposal)									
3.รวบรวมข้อมูลที่จะใช้ในการทำโครงงาน									
4.ทำโครงงานตามขั้นตอนในการทำ Data Science (OSEMN)									
5.จัดทำรายงานและสรุปผลของการทำโครงงาน									
6.นำเสนอโครงงาน									

4.งบประมาณ

จำนวนเงิน 0 บาท

5.เครื่องมือ

5.1 Google Colab

5.2 Jupyter Notebook

บทที่ 4 ผลการดำเนินงาน

1.Obtain หรือการเก็บข้อมูล

เราได้ทำการนำข้อมูลมาจากเว็บไซต์ https://steamspy.com/ โดยทำการนำข้อมูลประเภทของเกม 11 ประเภท ได้แก่

1.Action games , 2.Adventure games, 3.Early Access games, 4.Ex Early Access games

5.Free games, 6.Massively games, 7.RPG games, 8.Simulation games

9.Sports games, 10.Strategy games, 11.Indie games

ข้อมูลเกมแต่ละประเภท เราดาวน์โหลดมาเป็นไฟล์ . xlsx (excel) แล้วนำมาเปลี่ยนเป็นไฟล์ . csv เพราะสามารถเรียกใช้ฟังก์ชันต่างๆได้ดีกว่า ไฟล์ .xlsx และเราได้นำไฟล์ข้อมูลบันทึกลงใน google drive เพื่อง่ายต่อการเรียกใช้บน colab ดังนี้

```
[ ]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns ; sns.set()
```

ทำการ import เครื่องมือที่สำคัญต่างๆ

```
[ ]  from google.colab import drive
     drive.mount('drive/')
```

ทำการเชื่อต่อ Google colab เข้ากับข้อมูลต่างๆใน Google drive

2.Scrub การทำความสะอาดข้อมูล (Data Wrangling / Cleaning Data)

เราได้นำข้อมูลของเกมทั้ง 11 ประเภทมาทำความสะอาด ดังนี้

2.1ตัวอย่างข้อมูลที่ยังไม่ได้มีการทำความสะอาด

```
[ ] # อ่านข้อมูลจากไฟล์ csv
    df1 = pd.read_csv('/content/drive/My Drive/Info/Actiongames.csv')
    df1.head()
```

	#	Game	Release date	Price	Score rank(Userscore / Metascore)	Owners	Playtime (Median)	Developer(s)	Publisher(s)
0	35	Dota 2	Jul 9, 2013	Free	N/A (N/A/90%)	100,000,000 .. 200,000,000	22:54 (11:38)	Valve	Valve
1	248	Counter-Strike: Global Offensive	Aug 21, 2012	Free	N/A (N/A/83%)	100,000,000 .. 200,000,000	16:08 (05:33)	Valve, Hidden Path Entertainment	Valve
2	14	Team Fortress 2	Oct 10, 2007	Free	N/A (N/A/92%)	50,000,000 .. 100,000,000	22:34 (10:04)	Valve	Valve
3	119	PLAYERUNKNOWN'S BATTLEGROUNDS	Dec 21, 2017	14.99 $	N/A (N/A/86%)	50,000,000 .. 100,000,000	12:54 (04:03)	PUBG Corporation	PUBG Corporation
4	63	Counter-Strike	Nov 1, 2000	9.99 $	N/A (N/A/88%)	20,000,000 .. 50,000,000	02:41 (00:59)	Valve	Valve

ข้อมูลที่ยังไม่ได้มีการทำความสะอาด จะประกอบไปด้วย columns ดังนี้

1. # คือ หมายเลขไอดีของเกม
2. Game คือ ชื่อเกม
3. Release date คือ วันดือนปี ที่วางจำหน่ายเกม
4. Price คือ ราคาของเกม
5. Score rank(Userscore / Metascore) คือ เก็บข้อมูล Score rank , User score คะแนนผู้เล่น, Meta score คะแนนความนิยมของเกม
6. Owners คือ จำนวนผู้ที่เล่นเกม
7. Playtime(Median) คือ เวลาในการเล่นเกม
8. Developer(s) คือ ผู้สร้างและพัฒนาเกม
9. Publisher(s) คือ บริษัทที่วางจำหน่ายเกม

2.2วิธีในการทำความสะอาดข้อมูล

```
[ ]  # ทำการสร้าง new column 'Year' เก็บข้อมูลปีที่เกมวางขาย
     action['Year'] = action['Release date'].str.split(',',n=1,expand = True)[1] # ทำการ split แล้วเลือกเอาปี
     action['Year'] = action['Year'].replace(np.nan,0) # ทำการทแทนที่ ที่เป็นเป็นค่า NaN ด้วย 0
     action['Year'] = action['Year'].astype(int) # cast type int
     action.drop(action[(action['Year'] < 1999) | (action['Year'] > 2018)  ].index,inplace=True)  # เราจะเอาข้อมูลตั้งแต่ปี 1999 - 2018

     # ทำการสร้าง new column 'Metascore' เก็บค่า metascore ของแต่ละเกม
     action['Metascore'] = action['Score rank(Userscore / Metascore)'].map(lambda x: x.lstrip('N/A (N/A/)').rstrip('%)'))

     # ทำการลบข้อมูล ในบางกรณีที่ข้อมูลในของ column Metascore  มีอักขระพิเศษ เช่น '100% (95%/84'  เพราะไม่สามารถ clean data ตาม condition ของโค้ดบรรทัดด้านบนได้
     action.drop(action[action['Metascore'].map(len)>3].index,inplace=True)  # เมื่อลบแล้วจะสามารถ cast type ได้สะดวกขึ้น

     # ทำการสร้าง new column 'Playtime' เก็บเวลาเฉลี่ยที่แต่ละเกมถูกเล่น หน่วย : ชั่วโมง
     action['Playtime'] = action['Playtime (Median)'].map(lambda x : x.split('(')[0].replace(':','.'))

     # ทำการสร้าง new column 'Owners_median' เก็บจำนวนผู้ที่ซื้อเกม เช่น ข้อมูลเป็น 10-20  เราโดยจะเก็บเป็น (10+20)/2
     action['Owners_median'] = action['Owners'].map(lambda x : ( int(x.split('..')[0].replace(',','')) + int(x.split('..')[1].replace(',',''))))/2 )
     action['Owners_median'] = action['Owners_median']/1000000 # ทำการลดหน่วย

     # ทำการแปลงและตัดค่า empty string , string , $
     action['Price'] = action['Price'].replace('Free',np.nan)
     action['Price'] = action['Price'].str.replace('$','')
     action['Metascore'] = action['Metascore'].replace('',np.nan)

     # ทำการ cast type ในแต่ละ column เพื่อป้องกัน Error เมื่อจะนำข้อมูลไปใช้
     action['Price'] = action['Price'].astype(np.float)
     action['Playtime'] = action['Playtime'].astype(float)
     action['Metascore'] = action['Metascore'].astype(float)
     action['Owners_median'] = action['Owners_median'].astype(float)

     # ทำการ delete column ที่ไม่ได้ใช้งาน ได้แก่ [ Release date , Score rank(Userscore / Metascore) , Owners ,  Playtime (Median) ]
     action_games = action.drop(columns=['Release date', 'Score rank(Userscore / Metascore)',  'Owners', 'Playtime (Median)'],axis=1)

     # ลบข้อมูล ในทุกๆ row ที่มีค่า 0
     action_games = action_games.fillna(0)
     action_games = action_games[(action_games != 0).all(1)]
```

ในเกมทั้ง 11 ประเภทเราได้ทำการทำความสะอาดข้อมูลด้วยวิธีการที่คล้ายคลึงกัน

2.3ตัวอย่างข้อมูลที่ทำความสะอาดแล้ว

```
[ ] action_games.head()
```

AppId	Game	Price	Developer(s)	Publisher(s)	Year	Metascore	Playtime	Owners_median
119	PLAYERUNKNOWN'S BATTLEGROUNDS	14.99	PUBG Corporation	PUBG Corporation	2017	86.0	12.54	75.0
63	Counter-Strike	9.99	Valve	Valve	2000	88.0	2.41	35.0
2	Grand Theft Auto V	14.99	Rockstar North	Rockstar Games	2015	96.0	11.16	15.0
4	Half-Life 2	0.99	Valve	Valve	2004	96.0	0.43	15.0
5	Portal 2	9.99	Valve	Valve	2011	95.0	4.28	15.0

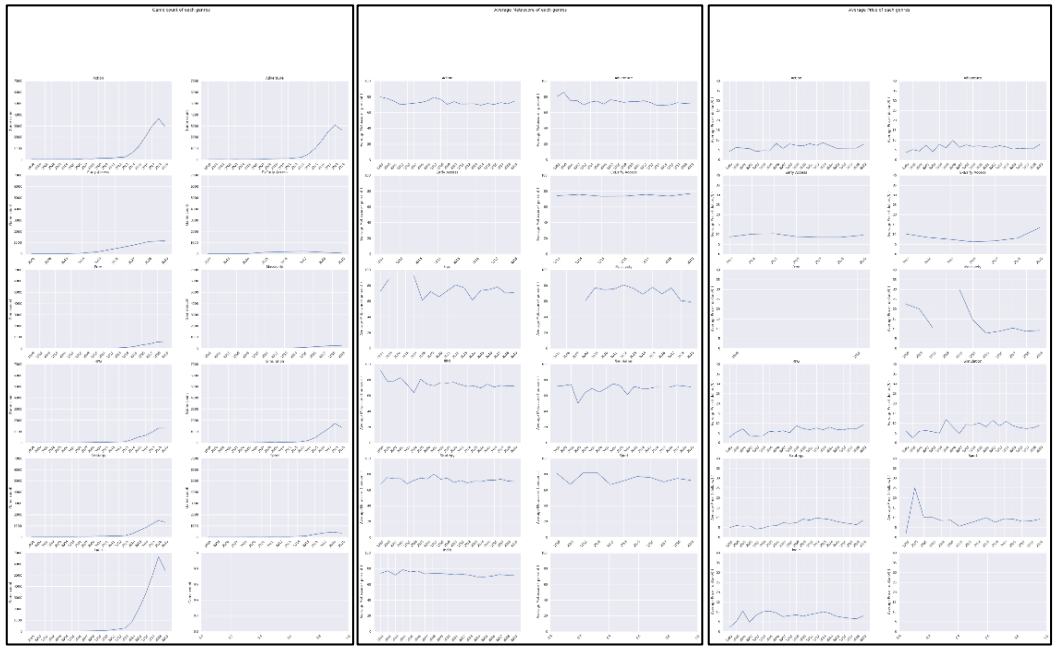
ข้อมูลที่ได้มีการทำความสะอาดแล้ว จะประกอบไปด้วย columns ดังนี้

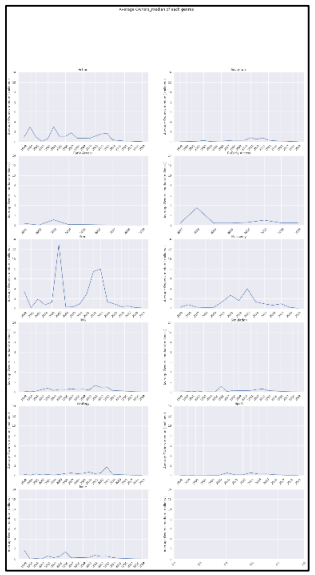
1. AppId คือ หมายเลขไอดีของเกม
2. Game คือ ชื่อเกม
3. Price คือ ราคาเกม
4. Developer(s) คือ ผู้สร้างและพัฒนาเกม
5. Publisher(s) คือ บริษัทที่วางจำหน่ายเกม
6. Year คือ ปีที่วางขขายเกม
7. Metascore คือ คะแนนความนิยมของเกม
8. Playtime คือ เวลาที่ใช้ในการเล่นนเกม
9. Owners_median คือ จำนวนผู้เล่นเกม

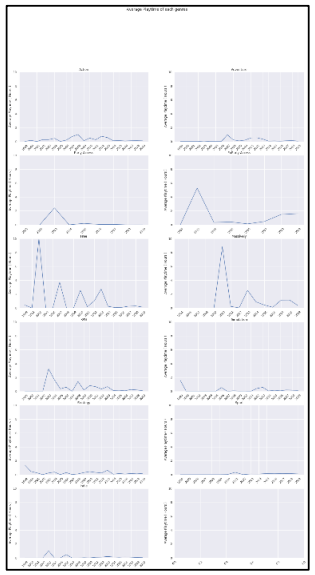
3.Explore การค้าหาข้อมูลทที่น่าวนใจ

3.1 แสดงข้อมูลของเกมแต่ละประเภทในแต่ละปี โดยแบ่งเป็นแต่ละ Feature ของ Data

เราได้ทำการสร้างฟังชก์ชัน plot_feature เพื่อสามารถเรียกดู Feature ต่างๆของ Data โดยแสดงข้อมูล แบ่งเป็นแต่ละประเภทตั้งแต่ปี 1999 – 2018 แต่เกมบางประเภท ในบางปี ไม่มีข้อมูล เราจึงใช้การplot graph แยกแต่ละประเภท ดังนี้







3.1.4 3.1.5

3.1.1 แสดงจำนวน Game ในแต่ละปี ของแต่ละประเภท

3.1.2 แสดงค่าเฉลี่ยของ Meta score ในแต่ละปี ของแต่ละประเภท

3.1.3 แสดงค่าเฉลี่ยของ Price (ราคา) ในแต่ละปี ของแต่ละประเภท

3.1.4 แสดงค่าเฉลี่ยของ Onwers (จำนวนผู้เล่น) ในแต่ละปี ของแต่ละประเภท

3.1.5 แสดงค่าเฉลี่ยของ Play time (เวลาในการเล่น) ในแต่ละปี ของแต่ละประเภท

3.2 ตารางแสดงค่าเฉลี่ยทุกๆปีของ features ต่างๆของเกมแต่ละประเภท

	Price	Playtime	Metascore	Owners_median	value
exEarlyAccess	13.371875	5.423125	79.968750	5.118750	71.799891
rpg	14.511186	8.280339	79.644068	3.372881	47.290805
strategy	16.394605	5.923026	78.644737	1.993553	42.109143
action	14.574356	4.204257	78.034653	3.127896	39.771393
indie	17.437674	3.443256	76.457364	1.974729	36.333102
sports	30.790000	5.610556	81.111111	1.300000	35.582833
adventure	12.120536	5.100179	77.133929	2.870223	34.922137
simulation	17.202308	6.420923	76.569231	1.757077	27.394718
massively	9.597324	0.898596	73.369231	0.785702	7.529532
earlyAccess	8.708846	0.086656	74.200000	0.058004	0.482585
free	NaN	0.322216	73.559524	0.797327	NaN

จากตารางนี้เราจะเห็นได้ว่า เมื่อลองพิจารณาไปที่อัตราส่วนของค่าต่างๆในแต่ละเกม เกมที่สร้างรายได้ ได้มากที่สุดคือเกมประเภท exEarly Access แต่เราต้องการข้อมูลที่ค่อนข้างเป็นปัจจุบันเพื่อพิจารณาว่าเกมประเภทใดสร้างรายได้ได้มากที่สุด

3.3 แสดงข้อมูลของเกมทุกประเภท โดยแบ่งเป็นแต่ละปี(ปี 1999-2018)

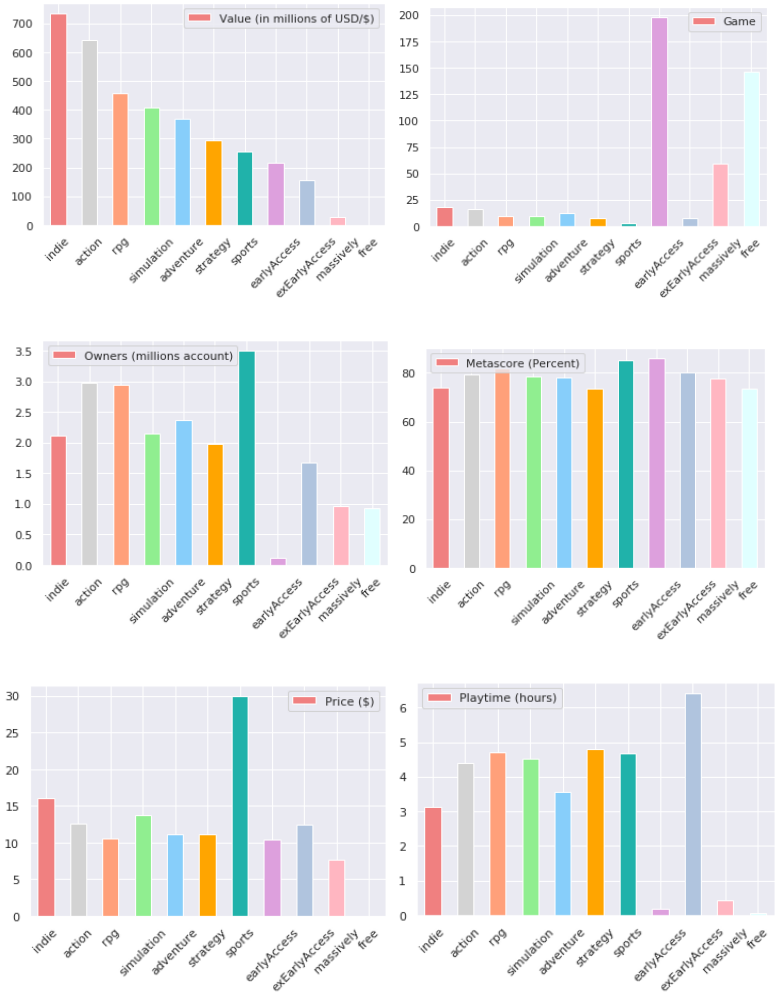
โดยมีการ สร้างฟังก์ชันแสดงข้อมูลของเกมทุกประเภท โดยสามารถเลือกดูได้ในแต่ละปี (ปี 1999-2018) และ สร้างฟังก์ชันแสดงข้อมูลของเกมทุกประเภทและแสดงกราฟข้อมูล โดยสามารถเลือกดูได้ในแต่ละปี (ปี 1999-2018)

ทดสอบดูข้อมูลในปี 2015-2018 ดังนี้

3.3.1 ข้อมูลในปี 2015

```
[ ] plot_tableOfYear(tableOf_year(2015))
```

	Value (in millions of USD/$)	Game	Owners (millions account)	Metascore (Percent)	Price ($)	Playtime (hours)
indie	735.02405	18	2.116389	74.111111	16.101111	3.117778
action	642.69500	16	2.968750	79.187500	12.565000	4.412500
rpg	456.50100	10	2.940000	81.900000	10.580000	4.696000
simulation	407.16100	10	2.140000	78.400000	13.810000	4.523000
adventure	370.01625	12	2.364583	78.250000	11.215000	3.569167
strategy	292.89150	8	1.981250	73.375000	11.177500	4.786250
sports	254.89500	3	3.500000	85.000000	29.990000	4.683333
earlyAccess	215.03050	198	0.123889	86.000000	10.362515	0.194091
exEarlyAccess	154.86650	8	1.668750	80.250000	12.427500	6.412500
massively	30.06650	59	0.967288	77.600000	7.613529	0.436441
free	0.00000	146	0.931610	73.444444	NaN	0.073425

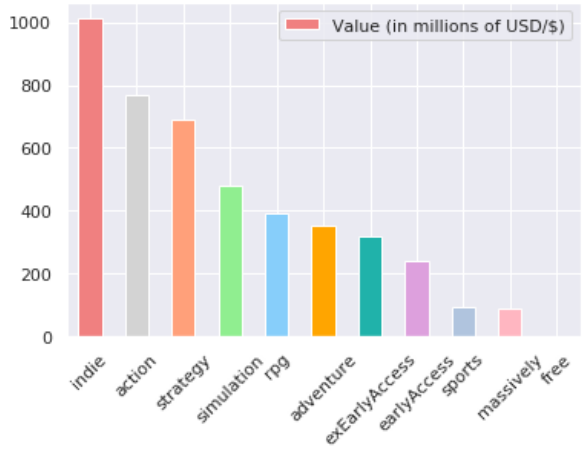


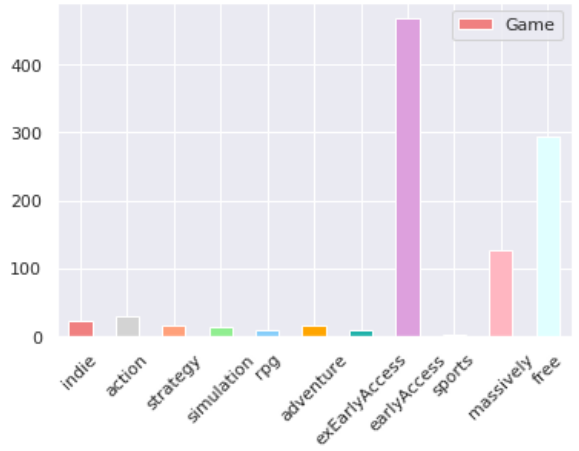
กราฟและตารางแสดงข้อมูลต่างของเกมแต่ละประเภทในปี 2015

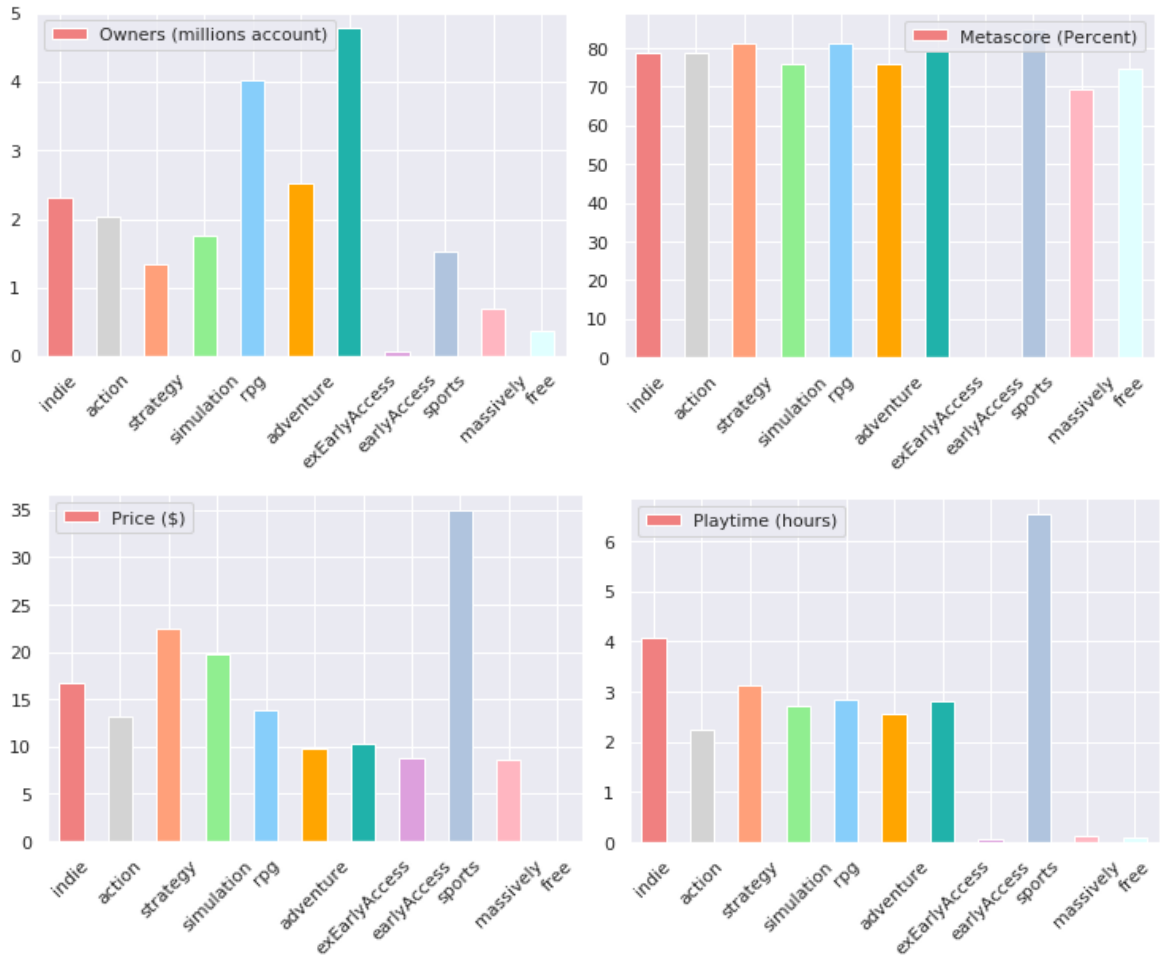
3.3.1 ข้อมูลในปี 2016

```
[ ]  plot_tableOfYear(tableOf_year(2016))
```

	Value (in millions of USD/$)	Game	Owners (millions account)	Metascore (Percent)	Price ($)	Playtime (hours)
indie	1013.97775	23	2.325000	78.608696	16.761739	4.080870
action	770.76180	30	2.044000	78.700000	13.200000	2.254333
strategy	688.33415	16	1.333438	81.250000	22.393125	3.114375
simulation	480.26900	14	1.757143	75.785714	19.804286	2.717143
rpg	389.87850	8	4.018750	81.250000	13.908750	2.852500
adventure	352.21365	16	2.524062	75.750000	9.849375	2.551875
exEarlyAccess	320.76650	8	4.793750	79.125000	10.258750	2.800000
earlyAccess	239.58705	468	0.070182	NaN	8.814439	0.060064
sports	91.85400	3	1.533333	85.000000	34.956667	6.540000
massively	87.14625	126	0.685317	69.250000	8.688182	0.111984
free	0.00000	293	0.377918	74.714286	NaN	0.104676





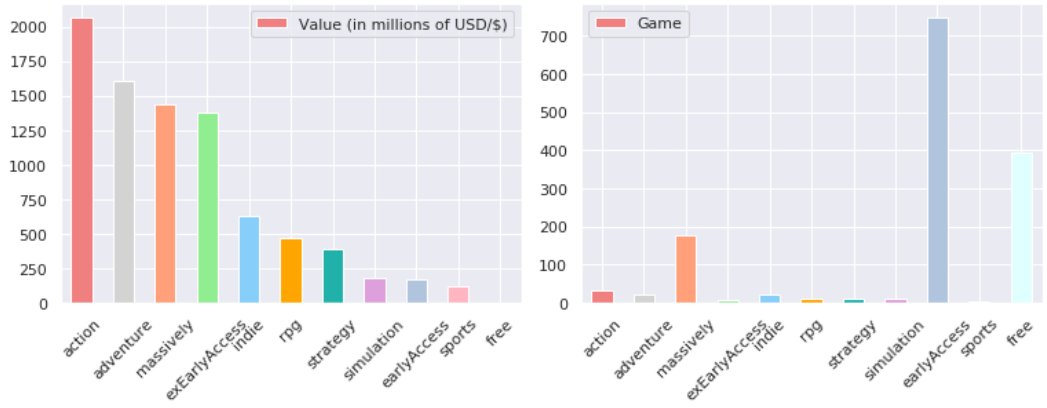


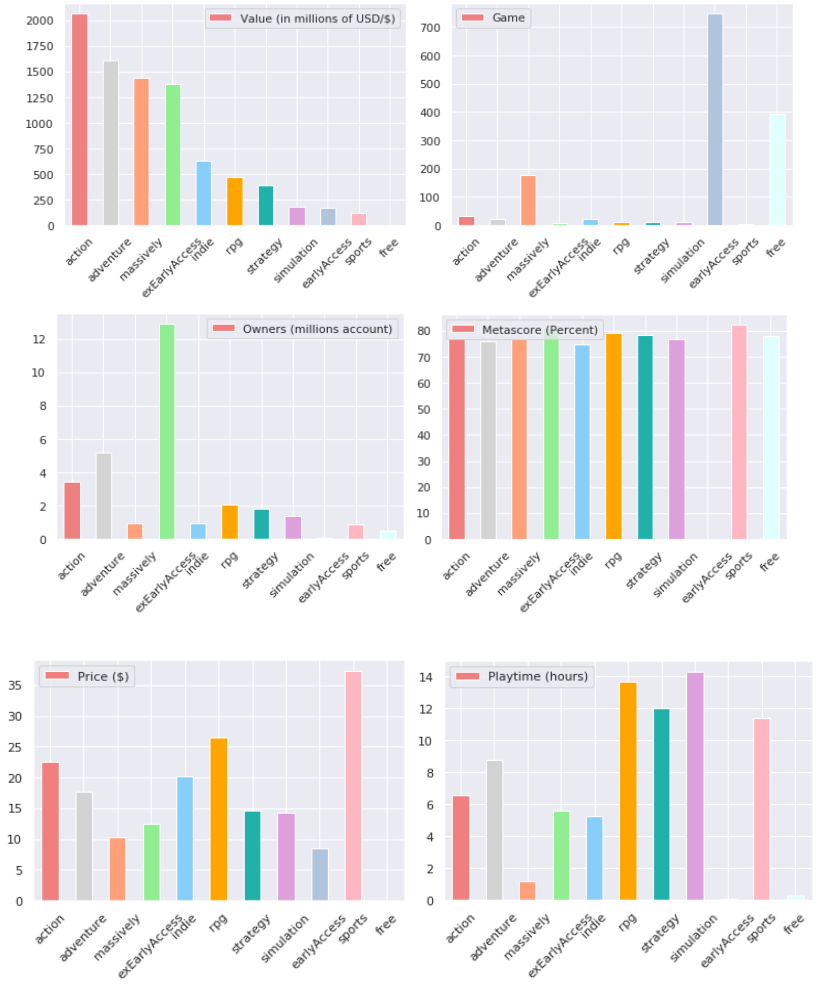
กราฟและตารางแสดงข้อมูลต่างของเกมแต่ละประเภทในปี 2016

3.3.1 ข้อมูลในปี 2017

```
[ ]  plot_tableOfYear(tableOf_year(2017))
```

	Value (in millions of USD/$)	Game	Owners (millions account)	Metascore (Percent)	Price ($)	Playtime (hours)
action	2068.64090	34	3.453235	77.294118	22.485588	6.590294
adventure	1604.74765	20	5.211750	76.050000	17.620000	8.778500
massively	1435.32540	178	0.968315	76.777778	10.319067	1.149831
exEarlyAccess	1381.34500	7	12.928571	81.000000	12.525714	5.608571
indie	630.22225	21	0.965476	74.571429	20.228095	5.221905
rpg	466.80600	12	2.033333	79.250000	26.448333	13.669167
strategy	386.75350	11	1.786364	78.181818	14.649091	12.032727
simulation	178.38750	10	1.375000	76.800000	14.240000	14.310000
earlyAccess	167.90335	749	0.044933	NaN	8.444344	0.040921
sports	124.17400	3	0.866667	82.333333	37.323333	11.393333
free	0.00000	393	0.494987	77.888889	NaN	0.263053



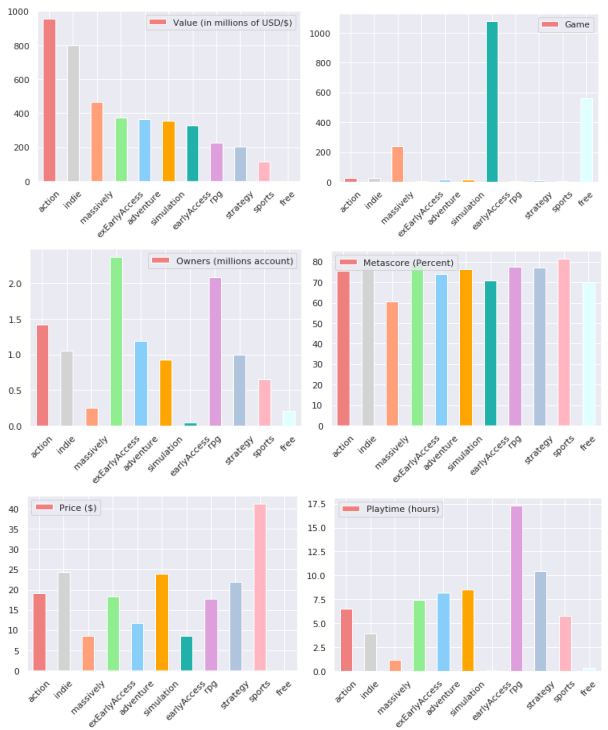


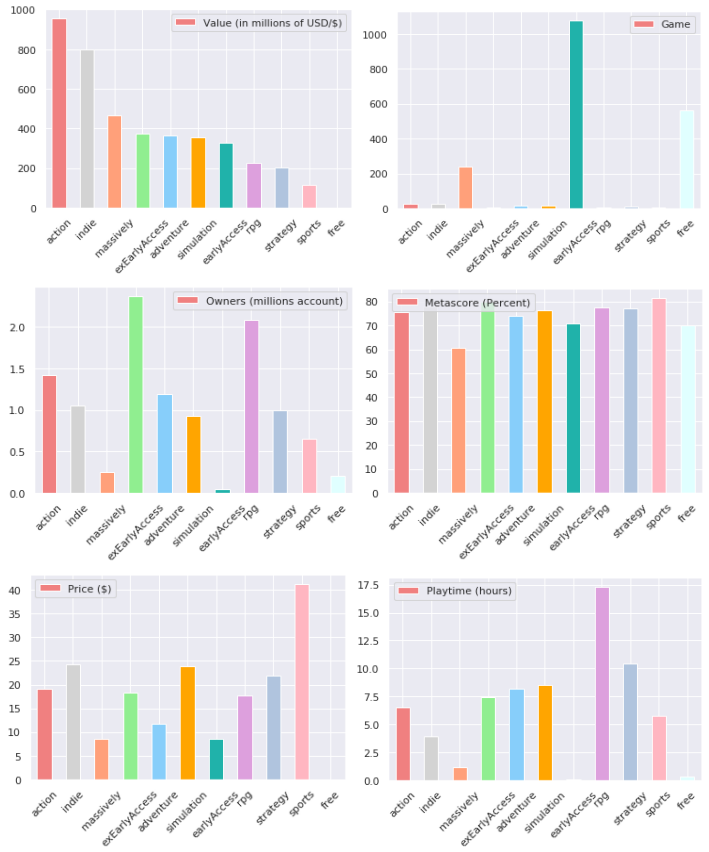
กราฟและตารางแสดงข้อมูลต่างของเกมแต่ละประเภทในปี 2017

3.3.1 ข้อมูลในปี 2018

```
plot_tableOfYear(tableOf_year(2018))
```

	Value (in millions of USD/$)	Game	Owners (millions account)	Metascore (Percent)	Price ($)	Playtime (hours)
action	956.41895	25	1.423200	75.760000	19.037200	6.561200
indie	797.76440	27	1.057778	77.962963	24.249259	3.909259
massively	465.61295	239	0.255188	60.500000	8.636719	1.155732
exEarlyAccess	374.96850	7	2.371429	79.714286	18.364286	7.427143
adventure	364.85580	16	1.191875	74.187500	11.718750	8.191875
simulation	353.90165	17	0.924118	76.176471	23.945882	8.551176
earlyAccess	328.11895	1078	0.040867	70.666667	8.467561	0.104583
rpg	224.72050	5	2.090000	77.600000	17.680000	17.308000
strategy	201.27525	10	0.997500	77.300000	21.990000	10.449000
sports	117.72400	4	0.650000	81.500000	41.240000	5.772500
free	0.00000	561	0.211399	70.111111	NaN	0.294742

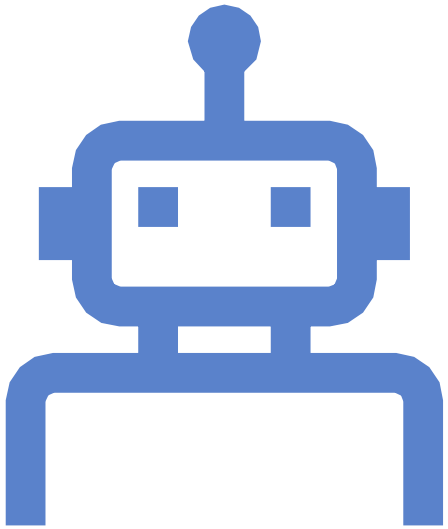




กราฟและตารางแสดงข้อมูลต่างของเกมแต่ละประเภทในปี 2018

จากการสังเกตค่าใน column value ในปี 2016 - 2018 จะเห็นได้ว่าเกมประเภท Action สร้างรายได้มากที่สุด เราจึงสามารถอนุมานได้ว่าในปัจจุบันเกมประเภท Action เป็นที่นิยม มีจำนวนเกมมากและมีผู้เล่นจำนวนมาก

4.Model การสร้างโมเดลหรือการทำนายผล Predictive Model



ทำการสร้าง model เพื่อวิเคราะห์ว่าปัจจัยใดที่ส่งผลต่อจำนวนผู้เล่นเกม ประเภท Action และลองทำการทำนายผล โดยใช้ Random forests เนื่องจาก features ของข้อมูลค่อนข้างน้อย เราจึงนำทุก features ที่มีมาใช้ในการทำนายผล

4.1 เริ่มทำการสร้าง Random Forests Model

4.1.1 เนื่องจาก action_data เรามีข้อมูลค่อนข้างน้อย ซึ่งเราได้ลองทำการแบ่งข้อมูล ออกเป็นสามส่วน คือตัว train, test, predict แล้วทำการรัน Model ผลปรากฏว่า Accuracy score น้อยมาก เพราะข้อมูลในการ train, test มีน้อย

4.1.2 ดังนั้นเราจึงแบ่งสัดส่วนตัว train และ test ออกเป็น 80% และ 20% ตามลำดับ จะได้ Accuracy score สูงขึ้น

4.1.3เมื่อเราได้ Accuracy score ที่เหมาะสมแล้ว เราจึงลองนำข้อมูลบางส่วนของ Action_data มาทำการ prediction Model

4.1.4 จากขั้นตอนที่แล้ว การนำข้อมูลเดิมที่ผ่านการ train ของ Model มาแล้ว ย่อมมีความแม่นยำในการ prediction มากอยู่แล้ว เราจึงจะลองหาข้อมูลจากเกมประเภทอื่นที่ได้รับความนิยมหรือมีมูลค่าทางการตลาดใกล้เคียงกับ Action games อาทิเช่น indie games , adventure games มาใช้เป็นตัว prediction Model

```
# เริ่มสร้าง Model

#Import scikit-learn dataset library
from sklearn import datasets

# Import train_test_split function
from sklearn.model_selection import train_test_split

X = action_data[['Price','Playtime','value','Metascore']] # Features ที่ใช้ในการ train model
y = action_data['Owners_median'] # lebel

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2) # 80% training and 20% test
```

ทำการสร้าง Features และ Lebel และแบ่งข้อมูลสำหรับการ test ,train Model

```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100) # สร้าง model

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train) # fit model โดยใส่ค่า X_train, y_trian

y_pred=clf.predict(X_test) # ทำการการ test model
```

ทำการสร้าง Model และทำการ training , test Model

```
[ ] #Import scikit-learn metrics module for accuracy calculation
    from sklearn import metrics

    # ตรวจสอบความแม่นยำของ model
    # Model Accuracy, how often is the classifier correct?
    print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

    Accuracy: 0.6829268292682927
```

ตรวจสอบความแม่นยำของ Model ซึ่ง Accuracy score = 0.6829 คิดเป็นประมาณ 68.29%

4.2 นำข้อมูลมาPredict Model

โดยการลองนำข้อมูลจากเกมประเภท indie และ adventure มาใช้ในการ prediction Model ดังนี้

AppId	Game	Developer(s)	Publisher(s)	Year	Price	Metascore	Playtime	value	Owners_median	Predicted
59	Rocket League	Psyonix, Inc.	Psyonix, Inc.	2015	19.99	86.0	6.43	299.850	15000000.0	15000000.0
145	Grim Dawn	Crate Entertainment	Crate Entertainment	2016	24.99	83.0	42.17	374.850	15000000.0	15000000.0
174	Terraria	Re-Logic	Re-Logic	2011	9.99	83.0	9.53	149.850	15000000.0	15000000.0
1326	The Tiny Bang Story	Colibri Games	Colibri Games	2011	4.99	63.0	0.05	74.850	15000000.0	3500000.0
132	Don't Starve Together	Klei Entertainment	Klei Entertainment	2016	14.99	83.0	2.44	112.425	7500000.0	15000000.0

ตารางแสดงการเปรียบเทียบระหว่างค่าจริงกับค่าจากการ Prediction Model ของข้อมูลจากเกมประเภท indie

AppId	Game	Developer(s)	Publisher(s)	Year	Price	Metascore	Playtime	value	Owners_median	Predicted
59	Rocket League	Psyonix, Inc.	Psyonix, Inc.	2015	19.99	86.0	6.43	299.850	15000000.0	15000000.0
145	Grim Dawn	Crate Entertainment	Crate Entertainment	2016	24.99	83.0	42.17	374.850	15000000.0	15000000.0
174	Terraria	Re-Logic	Re-Logic	2011	9.99	83.0	9.53	149.850	15000000.0	15000000.0
1326	The Tiny Bang Story	Colibri Games	Colibri Games	2011	4.99	63.0	0.05	74.850	15000000.0	3500000.0
132	Don't Starve Together	Klei Entertainment	Klei Entertainment	2016	14.99	83.0	2.44	112.425	7500000.0	15000000.0

ตารางแสดงการเปรียบเทียบระหว่างค่าจริงกับค่าจากการ Prediction Model ของข้อมูลจากเกมประเภท Adventure

5.Interpret การนำเสนอข้อมูล

1.เราสามารถหาประเภทเกมที่ได้รับความนิยมและสร้างรายได้มากที่สุดนั่นคือ เกมประเภท Action โดยอ้างอิงข้อมูลจาก average value ในปี2015-2018

```
[ ]  plot_tableOfYear(tableOf_year(2016))
```

	Value (in millions of USD/$)	Game	Owners (millions account)	Metascore (Percent)	Price ($)	Playtime (hours)
indie	1013.97775	23	2.325000	78.608696	16.761739	4.080870
action	770.76180	30	2.044000	78.700000	13.200000	2.254333
strategy	688.33415	16	1.333438	81.250000	22.393125	3.114375
simulation	480.26900	14	1.757143	75.785714	19.804286	2.717143
rpg	389.87850	8	4.018750	81.250000	13.908750	2.852500
adventure	352.21365	16	2.524062	75.750000	9.849375	2.551875
exEarlyAccess	320.76650	8	4.793750	79.125000	10.258750	2.800000
earlyAccess	239.58705	468	0.070182	NaN	8.814439	0.060064
sports	91.85400	3	1.533333	85.000000	34.956667	6.540000
massively	87.14625	126	0.685317	69.250000	8.688182	0.111984
free	0.00000	293	0.377918	74.714286	NaN	0.104676

ตารางแสดงข้อมูลในปี 2016

```
[ ]  plot_tableOfYear(tableOf_year(2017))
```

	Value (in millions of USD/$)	Game	Owners (millions account)	Metascore (Percent)	Price ($)	Playtime (hours)
action	2068.64090	34	3.453235	77.294118	22.485588	6.590294
adventure	1604.74765	20	5.211750	76.050000	17.620000	8.778500
massively	1435.32540	178	0.968315	76.777778	10.319067	1.149831
exEarlyAccess	1381.34500	7	12.928571	81.000000	12.525714	5.608571
indie	630.22225	21	0.965476	74.571429	20.228095	5.221905
rpg	466.80600	12	2.033333	79.250000	26.448333	13.669167
strategy	386.75350	11	1.786364	78.181818	14.649091	12.032727
simulation	178.38750	10	1.375000	76.800000	14.240000	14.310000
earlyAccess	167.90335	749	0.044933	NaN	8.444344	0.040921
sports	124.17400	3	0.866667	82.333333	37.323333	11.393333
free	0.00000	393	0.494987	77.888889	NaN	0.263053

ตารางแสดงข้อมูลในปี 2017

```
plot_tableOfYear(tableOf_year(2018))
```

	Value (in millions of USD/$)	Game	Owners (millions account)	Metascore (Percent)	Price ($)	Playtime (hours)
action	956.41895	25	1.423200	75.760000	19.037200	6.561200
indie	797.76440	27	1.057778	77.962963	24.249259	3.909259
massively	465.61295	239	0.255188	60.500000	8.636719	1.155732
exEarlyAccess	374.96850	7	2.371429	79.714286	18.364286	7.427143
adventure	364.85580	16	1.191875	74.187500	11.718750	8.191875
simulation	353.90165	17	0.924118	76.176471	23.945882	8.551176
earlyAccess	328.11895	1078	0.040867	70.666667	8.467561	0.104583
rpg	224.72050	5	2.090000	77.600000	17.680000	17.308000
strategy	201.27525	10	0.997500	77.300000	21.990000	10.449000
sports	117.72400	4	0.650000	81.500000	41.240000	5.772500
free	0.00000	561	0.211399	70.111111	NaN	0.294742

ตารางแสดงข้อมูลในปี 2018

2.เราสามารถหา features ที่มีความสัมพันธ์หรือส่งผลต่อจำนวนผู้เล่นเกมในประเภทเกม Action นั่นคือ feature : Price, Playtime, value, Metascore

3.เราสามารถทำนายหาจำนวนผู้เล่นในเกมนั้นๆได้ โดยใช้ Feaures ที่มีความสัมพันธ์กัน โดยใช้ Random forests Model

```
[ ]  # ตัวอย่างตารางแสดงค่าจากการ predictionn model
     df_predict2.head()
```

AppId	Game	Developer(s)	Publisher(s)	Year	Price	Metascore	Playtime	value	Owners_median	Predicted
59	Rocket League	Psyonix, Inc.	Psyonix, Inc.	2015	19.99	86.0	6.43	299.850	15000000.0	15000000.0
145	Grim Dawn	Crate Entertainment	Crate Entertainment	2016	24.99	83.0	42.17	374.850	15000000.0	15000000.0
174	Terraria	Re-Logic	Re-Logic	2011	9.99	83.0	9.53	149.850	15000000.0	15000000.0
1326	The Tiny Bang Story	Colibri Games	Colibri Games	2011	4.99	63.0	0.05	74.850	15000000.0	3500000.0
132	Don't Starve Together	Klei Entertainment	Klei Entertainment	2016	14.99	83.0	2.44	112.425	7500000.0	15000000.0

ตารางแสดงการเปรียบเทียบระหว่างค่าจริงกับค่าจากการ Prediction Model

บทที่ 5 สรุปผลการดำเนินงาน และข้อเสนอแนะ

STEAM®

จากการจัดทำโครงงาน เรื่อง การศึกษาความสัมพันธ์ของเกมประเภทต่างๆจากเว็บไซต์ Steam สามารถสรุปผลการดำเนินงานและข้อเสนอแนะได้ดังนี้

5.1 การดำเนินโครงงานนี้ได้บรรลุตามวัตถุประสงค์ที่กำหนดไว้

1.เราสามารถหาประเภทเกมที่ได้รับความนิยมและสร้างรายได้มากที่สุดนั่นคือ เกมประเภท Action โดยอ้างอิงข้อมูลจาก average value ในปี2015-2018

2.เราสามารถหา features ที่มีความสัมพันธ์หรือส่งผลต่อจำนวนผู้เล่นเกมในประเภทเกม Action นั่นคือ feature : Price, Playtime, value, Metascore

3.เราสามารถทำนายหาจำนวนผู้เล่นในเกมนั้นๆได้ โดยใช้ Feaures ที่มีความสัมพันธ์กัน โดยใช้ Random forests Model

5.2 ข้อเสนอแนะ

1. ในการจะทำการทำนายหรือ Predictive Model ควรใช้ Features ที่มีความสัมพันธ์ที่ชัดเจนและมีข้อมูลที่มากพอ

เอกสารอ้างอิง

Data TH.com - Data Science ชิล ชิล.(2561). 5 ขั้นตอนในการทำ Data Science ต้นจนจบ.สืบค้นเมื่อ 12ตุลาคม 2562.จา เว็บไซต์:https://www.facebook.com/datasciencechill/posts/516768302041565/

Sergey Galyonkin.(2558).Steam Spy. สืบค้นเมื่อ 9 ตุลาคม 2562,จากเว็บไซต์ : https://steamspy.com/

Witchapong Daroontham.(2561).เจาะลึก Random Forest !!!— Part 2 of “รู้จัก Decision Tree, Random Forest, และ XGBoost!!!”.สืบค้นเมื่อ 1 พฤศจิกายน 2562.จากเว็บไซต์ : https://medium.com/@witchapongdaroontham/%E0%B9%80%E0%B8%88%E0%B8%B2%E0%B8%B0%E0%B8%A5%E0%B8%B6%E0%B8%81-random-forest-part-2-of-%E0%B8%A3%E0%B8%B9%E0%B9%89%E0%B8%88%E0%B8%B1%E0%B8%81-decision-tree-random-forest-%E0%B9%81%E0%B8%A5%E0%B0%B8%B0-xgboost-79b9f41a1c1c

ขอบคุณครับ