THOMPSON RIVERS UNIVERSITY

COMP 3710 – Applied Artificial Intelligence

Project Report

ZiCheng Ren
T00548429
04/08/2020

Introduction

As the technology advances, videos games has become a basic need for people’s daily life. More and more people would like to choose to play video games as amusement. Thus, The gaming company is encouraging creativity by continuously expanding the limits of what is possible, motivating businesses like Sony and Microsoft to develop new technologies to satisfy the world’s billions of gamers.

Gaming is becoming more than ever important to the entertainment industry. It's constantly evolving how customers interact with and play games. Not only does this result in greater customer participation but it also contributes to completely new gaming fan groups. Today, there are more than 2.5 billion gamers worldwide. Combined, in 2019 they will spend $152.1 billion on sports, which marks an annual rise of 9.6 percent.

By 2019, the US will outperform China by sales as the world's biggest gaming industry. Overall, this year the U.S. gaming industry will produce $36.9 billion, powered primarily by a + 13.9 percent rise in video game revenues. At $18.5 billion, console accounts for more than 50 percent of the US overall gaming market. Hence, We can analyze data and predicting data by Machine Learning through video game sale dataset.

- Purpose of Study
  1. The purpose of this project is to learn about the Video Game Sale dataset.
  2. Analyzing the data from Video Game Sale dataset.
  3. Learning how to use Scikit Learn to training data and predict.
  4. Learning how to use Pandas to process data
  5. Learning normalize data such as StandardScaler.
  6. Developing a classifier that can predict whether a video game will be successful or not.
  7. Prediction of sale in terms of dollar amount with platform, year, publisher, etc.
- Research Questions
  1. How to load the Video Game sale dataset into the Python program?
  2. How to analyze the dataset?
  3. How to solve the N/A data?
  4. How to extract your features?
  5. Which features should be training dataset and which data should be the target dataset?
  1. How to use the classification algorithm to predict whether a video game will be successful or not.
  6. How to use the regression algorithm to predict sales.

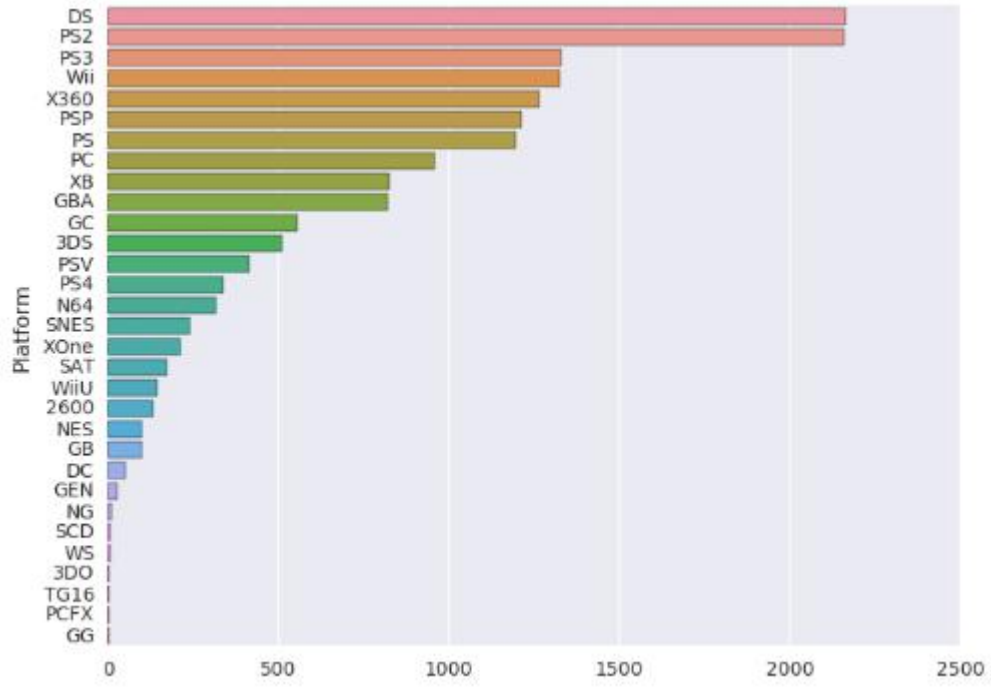
7. How to evaluate the accuracy of the model?

- Summary of Work

The dataset that I used from Kaggle. First, I will read the CSV file into my python file, and to check whether there are some N/A value. If there is, I will fill up the N/A value by Pandas. Then, process data such as drop useless, change string word to digit such as 1,2,3. Separate data into training dataset and testing dataset. And then, I will do feature engineering by StandardScaler. After the previous step, we can train the data and get predict by supervisor learning's Random forest classifier model, Logical Regression model, Linear Regression, K-NeighborsRegressor. In the last, compare each model's accuracy.

Background

Video game sale is a high votes dataset in Kaggle. And, some people already did very perfect visualization, for example, genera data analyze and arrange.



This graph was made by GarfiledLiang from Kaggle. The total amount of games in each different platform can be view such as DS and PS2 have the most games on their platform.

For the machine learning part, I could not find too many examples in Kaggle. However, there is one example for predict sales. That example shows they use a Decision Tree Regression model and they got an R squared score of 0.65 - 0.7. This is a good example for some people who do not know much about machine learning.

- Main techniques

1. Scikit-Learn

Scikit-Learn is a machine learning library for the Python language. It has various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, and K-Means, etc. In this project, I can use the Scikit-Learn library to solve the classification problem and the regression problem.

2. Dataset

The dataset that I choose is called Video Game Sales. This dataset contains a list of video games with sales greater than100,000 copies. There are 11 columns in this dataset. Each different games's total sales in different area and platforms that are clearly showed in the table: Rank - Ranking of overall sales by integer.

- Name - The games name, object.
- Platform - Platform of the games release for example, PC, PS4 etc. Object.
- Year -Year if the game's release, float.
- Genre - Genre of the game, object.
- Publisher - Publisher of the game.
- NA_Sales - Sales in North America (in millions).
- EU_Sales - Sales in Europe(in millions).
- JP_Sales - Sales in Japan(in millions).
- Other_Sales - Sales in the rest of the world(in millions).
- Global_Sales - Sales in the global(in millions).

3. LinearRegression(Supervised learning)

LinearRegression fits a linear model with coefficients w = (w1, ..., wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. In my program. LinearRegression can get about 97% accuracy on the test dataset.

Equation:

$$h(w) = w_1x_1 + w_2x_2 + w_3x_3 \ldots + \mathrm{b} = w^Tx + b$$

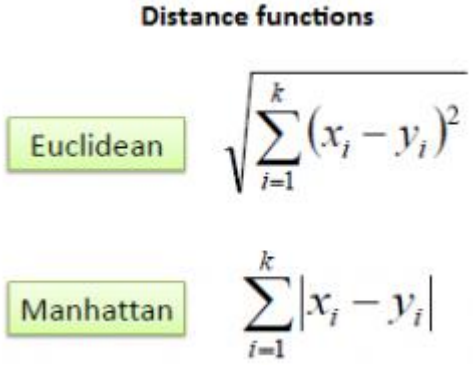
Where w,x are metric:

$$\mathrm{w} = \begin{pmatrix} b \\ w_1 \\ w_2 \end{pmatrix}, \mathrm{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

4. K-NeighborsRegressor(Supervised learning)

Regression-based on k-nearest neighbors. The target is predicted by local interpolation of the targets associated with the nearest neighbors in the training set. In my program. LinearRegression can get about 81% accuracy on the test dataset.

There are 3 distance: Euclidean Distance, Manhattan Distance, Hamming



Distance.

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

5. LogisticRegression(Supervised learning)

LogisticRegression is a classification model in machine learning. Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). like all regression analyses, logistic regression is a predictive analysis.

Equation:

Input:

$$h(w) = w_1x_1 + w_2x_2 + w_3x_3 \ldots + \mathrm{b}$$

Sigmoid:

$$h(w) = w_1x_1 + w_2x_2 + w_3x_3 \ldots + \mathrm{b}$$

6. RandomForestClassifier(Supervised learning)

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

7. Cross-validation

If test sets can provide unstable results because of sampling in data science, the solution is to systematically sample a certain number of test sets and then average the results. It is a statistical approach (to observe many results and take an average of them).

Project Details

- Step of Work

1. Import pandas and numpy packages in to my Python File.

```
import numpy as np
import pandas as pd
```

2. Import data Into python

```
data = pd.read_csv("vgsales.csv")
```

```
data.head()
```

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37

By using the Pandas package DataFrame can directly load the dataset from the current folder. Calling the head() function from pandas which can display the first 5 rows of my dataset.

3. Check Data information

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16598 entries, 0 to 16597
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Rank          16598 non-null  int64
 1   Name          16598 non-null  object
 2   Platform      16598 non-null  object
 3   Year          16327 non-null  object
 4   Genre         16598 non-null  object
 5   Publisher     16540 non-null  object
 6   NA_Sales      16598 non-null  float64
 7   EU_Sales      16598 non-null  float64
 8   JP_Sales      16598 non-null  float64
 9   Other_Sales   16598 non-null  float64
 10  Global_Sales  16596 non-null  float64
dtypes: float64(5), int64(1), object(5)
memory usage: 1.4+ MB
```

Using Pandas dataframe.info function to get a summary of the data. As you can see, there are 16598 row that contains data, the first row contains attributes. But, this dataset has the NA value on year and publisher columns.

4. Checking NA value

```
data.isnull().any()
```

```
Rank            False
Name            False
Platform        False
Year             True
Genre           False
Publisher        True
NA_Sales        False
EU_Sales        False
JP_Sales        False
Other_Sales     False
Global_Sales     True
dtype: bool
```

```
data.isnull().sum()
```

```
Rank              0
Name              0
Platform          0
Year            271
Genre             0
Publisher        58
NA_Sales          0
EU_Sales          0
JP_Sales          0
Other_Sales       0
Global_Sales      2
dtype: int64
```

Calling isnull.any() from pandas, it will generate the Boolean metric. If there is a NA value in a column, it should be False ,otherwise True.
Also, isnull.sum() will check each NA whetherit is null, if not it will count the number of 0 in that column.

5. Drop the NA value row

```
data.dropna(inplace=True)
```

I decided to drop the NA value row and it will not influence our model training. By using dropna() function from pandas DataFrame I set the inplace value to True, thus, it will do the operation inplace.

```
data.isnull().any()
```

```
Rank            False
Name            False
Platform        False
Year            False
Genre           False
Publisher       False
NA_Sales        False
EU_Sales        False
JP_Sales        False
Other_Sales     False
Global_Sales    False
dtype: bool
```

Check again, there is no NA value.

6. Instead of '2600' to 'PC'

```
np.unique(data["Platform"])
```

```
array(['2600', '3DO', '3DS', 'DC', 'DS', 'GB', 'GBA', 'GC', 'GEN', 'GG',
       'N64', 'NES', 'NG', 'PC', 'PCFX', 'PS', 'PS2', 'PS3', 'PS4', 'PSP',
       'PSV', 'SAT', 'SCD', 'SNES', 'TG16', 'WS', 'Wii', 'WiiU', 'X360',
       'XB', 'XOne'], dtype=object)
```

First, checking the ''Platform'' colum, I used the numpy's unique it can distinguish the duplicate value.

```
data['Platform'].replace('2006', 'PC', inplace=True)
```

```
np.unique(data['Platform'])
```

```
array(['3DO', '3DS', 'DC', 'DS', 'GB', 'GBA', 'GC', 'GEN', 'GG', 'N64',
       'NES', 'NG', 'PC', 'PCFX', 'PS', 'PS2', 'PS3', 'PS4', 'PSP', 'PSV',
       'SAT', 'SCD', 'SNES', 'TG16', 'WS', 'Wii', 'WiiU', 'X360', 'XB',
       'XOne'], dtype=object)
```

Then I called replace function to insead of previous value.

7. String value conver to numeric

Increaseing the accuracy, better to convert string value to numeric.

```
label3 = data['Publisher'].unique().tolist()
data['Publisher'] = data['Publisher'].apply(lambda n: label3.index(n))
np.unique(data['Publisher'])
```

```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
       247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259,
       260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272,
       273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285,
       286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298,
       299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311,
       312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324,
       325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337,
       338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350,
       351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363,
       364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376,
       377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389,
       390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402,
       403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415,
       416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428,
       429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441,
       442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454,
       455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467,
       468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480,
       481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493,
       494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506,
       507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519,
       520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532,
       533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545,
       546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558,
       559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571,
       572, 573, 574, 575], dtype=int64)
```

to_list() function can convert the ‘‘publisher’’ column to array and it is also drop unique. Data.publisher.apply() is to apply the index of publisher to current publisher.

```
label1 = data['Platform'].unique().tolist()
data['Platform'] = data['Platform'].apply(lambda n: label1.index(n))
np.unique(data['Platform'])
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29], dtype=int64)
```

```
label2 = data['Genre'].unique().tolist()
data['Genre'] = data['Genre'].apply(lambda n: label2.index(n))
np.unique(data['Genre'])
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11], dtype=int64)
```

```
data.head()
```

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	0	2006	0	0	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	1	1985	1	0	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	0	2008	2	0	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	0	2009	0	0	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	2	1996	3	0	11.27	8.89	10.22	1.00	31.37

8. Divide the dataset

```
x = data[['Platform','Genre','Publisher','NA_Sales','EU_Sales']]
y = data["Global_Sales"]
```

```
y.head()
```

```
0    82.74
1    40.24
2    35.82
3    33.00
4    31.37
Name: Global_Sales, dtype: float64
```

```
x.head()
```

	Platform	Genre	Publisher	NA_Sales	EU_Sales
0	0	0	0	41.49	29.02
1	1	1	0	29.08	3.58
2	0	2	0	15.85	12.88
3	0	0	0	15.75	11.01
4	2	3	0	11.27	8.89

X is features = 'Platform','Genre','Publisher','NA_Sales','EU_Sales'.
y is target = "Global_Sales"

9. Spliting data into test datset and training dataset

```
#train_test_split
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3)
```

```
from sklearn.preprocessing import StandardScaler
transfer = StandardScaler()
x_train = transfer.fit_transform(x_train)
x_test = transfer.transform(x_test)
```

```
x_train
```

```
array([[-1.49336177, -1.53339744, -0.55738442, -0.13754217, -0.27335692],
       [-0.6063009 , -1.53339744, -0.57686731, -0.14950765, -0.27335692],
       [-0.25147655,  0.80240826, -0.13850242, -0.31702439, -0.27335692],
       ...,
       [ 1.70005738,  0.80240826,  0.60184716, -0.28112795, -0.23656666],
       [ 0.63558433,  0.51043254, -0.56712587, -0.12557669, -0.07101049],
       [-0.6063009 ,  0.80240826, -0.13850242, -0.31702439, -0.27335692]])
```

I used the sklearn.model_select's train_test_split which can separete data by size. I set 70% of data for training 0.3 for testing and return x train data,x test data,y train data,y test data.
Also, I user StandarScaler from sklearn.preprocessing to nomalize the x_train and x_test.

10. Using LinearRegression model

```
#LinearRegression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
import matplotlib.pyplot as plt
# 4)estimator
estimator = LinearRegression()
estimator.fit(x_train,y_train)
# 5)get model
print("coef_:\n",estimator.coef_)
print("intercept_:\n",estimator.intercept_)
# 6)evaluate
y_predict = estimator.predict(x_test)
squared_error = mean_squared_error(y_test,y_predict)
LR_r2 = r2_score(y_test,y_predict)
print("mean_squared_error:\n",squared_error)
print("R2:\n",LR_r2)
LR_accuracy_on_test = estimator.score(x_test, y_test)
print("Accuracy on test: ", LR_accuracy_on_test* 100,"%")
fig = plt.figure(figsize=(10,6))
```

```
coef_:
 [-0.00508335 -0.00905167 -0.00767198  0.91791135  0.77237196]
intercept_:
 0.5405727065427117
mean_squared_error:
 0.08775677607013016
R2:
 0.9558594660301751
Accuracy on test:  95.5859466030175 %

<Figure size 720x432 with 0 Axes>
```

First, I chose the LinearRegression model from sklearn linear_model to predict global sale. Then, using r2 score, mean_squared_error to evaluate result.

11. LinearRegression with cross validation

```
#LinearRegression CV
from sklearn.model_selection import cross_val_predict
estimator2 = LinearRegression()
cv_value = cross_val_predict(estimator2, x, y, cv=10)
LRCV_r2 = r2_score(y,cv_value)
print("R2:\n",LRCV_r2)
```

```
R2:
 0.9368714685208545
```

Using cross_val_predict set the cv=10, estimator is LinearRegression from sklearn.model_selection which can get predictions in Ndarray. Then, we use the r2_score() function to get r2 value.

12. Using KneighborsRegressor model

```
#KNN
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error,r2_score
# 4)estimator
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(x_train,y_train)
# 6)evaluate
y_predict = knn.predict(x_test)
squared_error = mean_squared_error(y_test,y_predict)
KNN_r2 = r2_score(y_test,y_predict)
print("mean_squared_error:\n",squared_error)
print("R2:\n",KNN_r2)
KNN_accuracy_on_test = knn.score(x_test, y_test)
print("Accuracy on test: ", KNN_accuracy_on_test)
fig = plt.figure(figsize=(10,6))
```

```
mean_squared_error:
 0.1299805868631062
R2:
 0.9346214301985786
Accuracy on test:  0.9346214301985786

<Figure size 720x432 with 0 Axes>
```

13. Second, I choose the KNeighborsRegressor model to predict global sales. Then, using r2 score, mean_squared_error to evaluate result.

```
#KNN_CV
KNN_CV = KNeighborsRegressor(n_neighbors=5)
KNN_CV_r2 = -cross_val_score(KNN_CV, x, y, cv=5,scoring='r2')
KNN_CV_r2 = (KNN_CV_r2.mean())/100
print(KNN_CV_r2)
```

```
0.8061024596180095
```

I set K = 5, and I use the cross_val_score again with scoring='r2', we can get the mean of R2 of KNeighborsRegressor with cross-validation.

14. DataSet Classification problem(Prediction whether a game is success). The data Processing is same as previous regression problem.

```
data["SuccOrNot"] = (data["Global_Sales"]>0.22) #1->success 0->unseccess
```

```
data['SuccOrNot']
```

```
0         True
1         True
2         True
3         True
4         True
         ...
16593    False
16594    False
16595    False
16596    False
16597    False
Name: SuccOrNot, Length: 16289, dtype: bool
```

Setting a threshold, if global sale greater than theshold is success otherwise, not. Hence, global greater than 0.22 ($ in million) is success.

15. Feature dataset and target dataset

```
x = data[['Platform','Genre','Publisher','NA_Sales','Other_Sales']]
y = data["SuccOrNot"]
```

16. Spliting data into test datset and training dataset and also data standardization by StandardScaler.

```
#train_test_split
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
from sklearn.preprocessing import StandardScaler
transfer = StandardScaler()
x_train = transfer.fit_transform(x_train)
x_test = transfer.transform(x_test)
```

17.

```
from sklearn.linear_model import LogisticRegression
```

```
LR = LogisticRegression()
```

```
LR.fit(x_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

18. By LogisticRegression model to training data

```
y_predict = LR.predict(x_test)
LR.coef_
```

```
array([[ 0.10280632, -0.04599002, -0.47002436,  9.80433222, 11.44739293]])
```

```
LR.intercept_
```

```
array([3.32239371])
```

```
#Evaluation
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(confusion_matrix(y_test, y_predict))
print(classification_report(y_test,y_predict,labels=[1,0],target_names=["Success","Unsucc
```

```
[[2754   88]
 [ 307 1738]]
              precision    recall  f1-score   support

     Success       0.95      0.85      0.90      2045
   Unsuccess       0.90      0.97      0.93      2842

    accuracy                           0.92      4887
   macro avg       0.93      0.91      0.92      4887
weighted avg       0.92      0.92      0.92      4887
```

For evaluation, I used the confusion_matrix and clssification_repot both of them from sklearn.metics. Classification report will show the presicion on success and unsecess and recall f1-score and support.

19. Logistic regression with cross-validation

```
from sklearn.model_selection import cross_val_score,GridSearchCV
```

```
LR_CV = cross_val_score(LR, x, y, cv=5, scoring='accuracy')
```

```
print("Accuracy:\n",LR_CV.mean()*100,"%")
```

```
Accuracy:
 85.32065138824571 %
```

Forthe cross_value_score's scoring now is accuracy and cv is 5.

20. Using RandomForest get the cccuracy and other information

```
#RandomForest
from sklearn.ensemble import RandomForestClassifier
```

```
RFC = RandomForestClassifier()
```

```
RFC.fit(x_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

21. Get result of confusion matrix and classification report

```
y_predict2 = RFC.predict(x_test)
```

```
print(confusion_matrix(y_test, y_predict2))
print(classification_report(y_test,y_predict2,labels=[1,0],target_names=["
```

```
[[2718  124]
 [ 161 1884]]
              precision    recall  f1-score   support

     Success       0.94      0.92      0.93      2045
   Unsuccess       0.94      0.96      0.95      2842

    accuracy                           0.94      4887
   macro avg       0.94      0.94      0.94      4887
weighted avg       0.94      0.94      0.94      4887
```

22. RandomForest with Cross-validation

```
#RandomForest cv
RFC_CV = RandomForestClassifier()
param_dict = {"n_estimators": [120, 200, 300, 500, 800],
              "max_depth":[5,8,10]}
RFC_CV = GridSearchCV(RFC_CV, param_grid=param_dict, cv=3)
```

```
RFC_CV.fit(x_train,y_train)
```

```
GridSearchCV(cv=3, error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              max_samples=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators=100, n_jobs=None,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'max_depth': [5, 8, 10],
                         'n_estimators': [120, 200, 300, 500, 800]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

Here, I set the domain of n_estimators is [120,200,300,500,800], domain of max_depth is [5,8,10].

23. Estimator calling the best_params_, best_score_,best_estimator to get best parameters, best score(accuracy)

```
#Evaluate
#Best parameters
print("Best parameters:\n",RFC_CV.best_params_)
##Best score
print("Best score:\n", RFC_CV.best_score_)
#Best_estimator_
print("Best estimator:\n", RFC_CV.best_estimator_)
```

```
Best parameters:
 {'max_depth': 10, 'n_estimators': 800}
Best score:
 0.9390461882145512
Best estimator:
 RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                        criterion='gini', max_depth=10, max_features='auto',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=800,
                        n_jobs=None, oob_score=False, random_state=None,
                        verbose=0, warm_start=False)
```

Results

- Regression problem

Linear Regression:

```
coef_:
 [-0.00508335 -0.00905167 -0.00767198  0.91791135  0.77237196]
intercept_:
 0.5405727065427117
mean_squared_error:
 0.08775677607013016
R2:
 0.9558594660301751
Accuracy on test:  95.5859466030175 %

<Figure size 720x432 with 0 Axes>
```

The coef mean estimated coefficients for the linear regression problem. If multiple targets are passed during the fit (y 2D), this is a 2D array of shape (n_targets, n_features), while if only one target is passed, this is a 1D array of length n_features.
If you have 5 feature then you must have 5 coef_. Here is[-0.00508335 -0.00905167 -0.00767198 -0.91791135]

The intercept(bias) added to the decision function here is about 0.54.
Mean_squared_error is a measure of how close a fitted line is to data points. The smaller value is better. Here is 0.0877 that is good.

R^2 ranges from 0 to 1 which is a prediction score. In our case, we got the 0.95 which is good. Last, accuracy on test is about 95% which is got by estimator.score().

Linear Regression with Cross-validation:

```
R2:
 0.9368714685208545
```

R2 on Linear Regression with Cross-validation is 0.93 = 93%

KNeighborsRegressor:

```
mean_squared_error:
 0.1299805868631062
R2:
 0.9346214301985786
Accuracy on test:  0.9346214301985786
```

Mean_squared_error on Kneighbors Regressor is 0.129981
R2 on KNeighborsRegressor with Cross is 0.93 = 93%

KNeighborsRegressor with Cross-validation:

```
KNN_R2:
 0.8061024596180095
```

R2 from KNeighborsRegressor with Cross-validation is 0.8061=80.61%

- Classification problem

LogisticRegression:

```
 [[2754   88]
  [ 307 1738]]
              precision    recall  f1-score   support

     Success       0.95      0.85      0.90      2045
   Unsuccess       0.90      0.97      0.93      2842

    accuracy                           0.92      4887
   macro avg       0.93      0.91      0.92      4887
weighted avg       0.92      0.92      0.92      4887
```

LogisticRegression with cross-validation:

```
Accuracy:
 85.32065138824571 %
```

RandomForestClassifier:

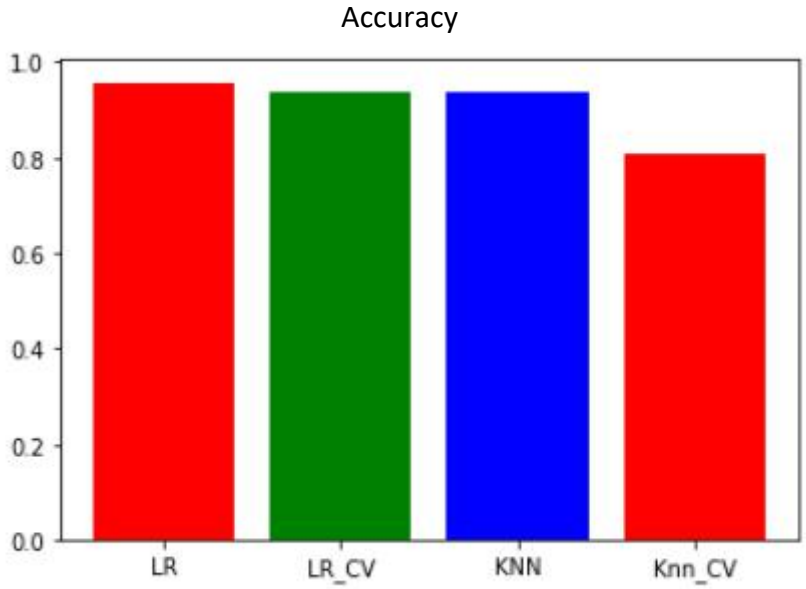
```
[[2718  124]
 [ 161 1884]]
              precision    recall  f1-score   support

     Success       0.94      0.92      0.93      2045
   Unsuccess       0.94      0.96      0.95      2842

    accuracy                           0.94      4887
   macro avg       0.94      0.94      0.94      4887
weighted avg       0.94      0.94      0.94      4887
```

RandomForestClassifier with cross-validation:

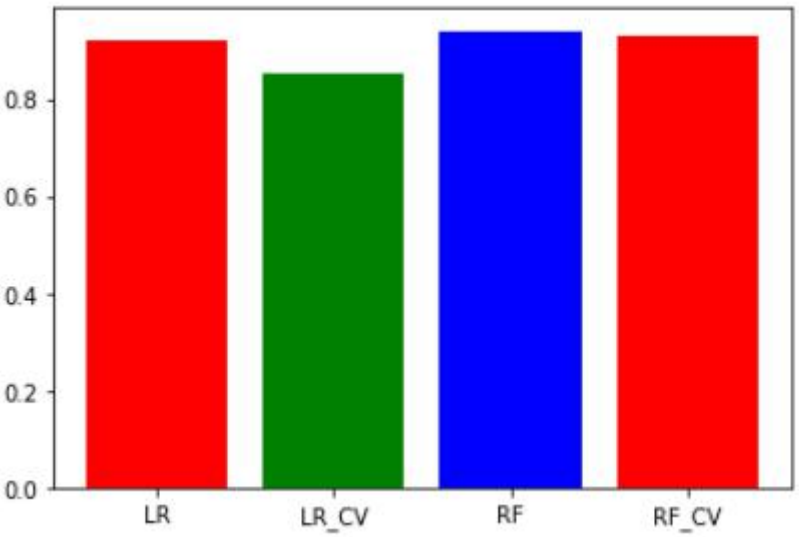
```
Best parameters:
 {'max_depth': 10, 'n_estimators': 800}
Best score:
 0.9390461882145512
Best estimator:
 RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                        criterion='gini', max_depth=10, max_features='auto',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=800,
                        n_jobs=None, oob_score=False, random_state=None,
                        verbose=0, warm_start=False)
```

With the better prediction, RandomForestClassifier model should be the most performance model with 94% accuracy. And, 124 false negative, 161 false positive.

Analysis the result



In the accuracy results, the logistic regression has the biggest accuracy which is 0.95. the accuracy of Linear Regression with Cross-validation and KNeighbors Regressor are consistent 93%. the smallest knn_cv that is KNeighborsRegressor with Cross-validation got the smallest accuracy 80%. In the whole training process, the number of features, whether data has been processed. I tried to drop the all sale feature, only keep the “Platform”,” Genre”,” publisher”, then the accuracy is reduced.



For the Classification, the Random Forest classifier proven to have the highest performance given the data. Accuracy of the logical regression is very close to the Redom_forest with Cross_validtion 92% and 93%. Classsification_report of Random Forest Classifier shows the precision of both success and unsuccess is 0.94, and unsuccess' recall is 0.96 which means 94% of 1, and 0.96% of that will be recalled, it is very large. Also,0.93 f1-score on success and 0.95 f1-score on unsuccess that are quite moderate.

Future Plan

In this project, I used linear regression, KNN, Logical Regression and Random forest . In the future, will try the more different model such as neural network. Also, trying to use the unsupervised learning.

Reference

Anderton, K. (2019, June 26). The Business Of Video Games: Market Share For Gaming Platforms in 2019 [Infographic]. Retrieved from https://www.forbes.com/sites/kevinanderton/2019/06/26/the-business-of-video-games-market-share-for-gaming-platforms-in-2019-infographic/#215213777b25

The Global Games Market Will Generate $152.1 Billion in 2019 as the U.S. Overtakes China as the Biggest Market. (n.d.). Retrieved from https://newzoo.com/insights/articles/the-global-games-market-will-generate-152-1-billion-in-2019-as-the-u-s-overtakes-china-as-the-biggest-market/

Garfieldliang. (2016, November 7). video games analysis. Retrieved from https://www.kaggle.com/garfieldliang/video-games-analysis

What are Mean Squared Error and Root Mean Squared Error? (-1, November 30). Retrieved from https://www.vernier.com/til/1014