Video Game Sales Prediction

# Contents

[Contents 2](#_Toc108790308)

[Overview 3](#_Toc108790309)

[About Data 3](#_Toc108790310)

[Architecture 4](#_Toc108790311)

[Data Cleaning 4](#_Toc108790312)

[Data Pre-processing 5](#_Toc108790313)

[Modelling 5](#_Toc108790314)

[Evaluation 7](#_Toc108790315)

[Development 8](#_Toc108790316)

# Overview

A video game is any electronic game that may be played on a computer, gaming console, or mobile device. Depending on the platform, computer games can be divided into PC games and control centre games. However, new categories of games, such as portable and social games, have recently been established due to the development of informal groups, cell phones, and tablets. Computer games have essentially advanced since the 1970s, when they first appeared.

It is frequently perplexing how much reality is recreated in today's computer games, which also have photorealistic graphics. Video Game developers, many outside parties, competitors and Video Game users would like to know what kinds of Video games would succeed in the future and what kinds of criteria would affect them. Video games sales prediction project is initiated to answer those needs in the video game industry.

Since a long time ago, video games have been a billion-dollar industry. By handling and researching a lot of data on computer gaming transactions in this module, we will lay the foundation for our evaluation. Video Game Sales with Ratings data set is used in this process. Using the data set, an Effective data analysis is done and via the observations of it a classification problem is identified.

With the assumption of a successful video game is a video game that has over 1 million sales, model is built to predict whether a given data point will succeed or not. To increase profits, users of the application can select which platform do they need to select, which developer is best suited to their need etc.

# About Data

This informational index was inspired by Gregory Smith's online scrape of the VGChartz Video Games Sales, and it simply increases the number of variables by adding another web scrape from Metacritic. Sadly, because Metacritic only covers a portion of the stages, there are perceptions that are missing. Similarly, a game might not have all of the views of the additional components discussed below. There are nearly 6,900 complete cases. Qualities of the data set is discussed below.

|  |  |
| --- | --- |
| Dataset Name | Video Game Sales with Ratings |
| Dataset Size | 1581KB |
| Number of Attributes | 16 |
| Number of Data Records | 16719 |
| Data Source Provider | Kaggle |
| Data Privacy | Public |
| Notes | DOI |
| Prepared by | Rush Kirubi (Owner) |

|  |  |  |
| --- | --- | --- |
| **Variable** | **Description** | **Data Type** |
| Name | Name of the Video Game | object |
| Platform | Platform of the games release (i.e. PC,PS4, etc.) | object |
| Year\_of\_Release | Year of the game's release | float64 |
| Genre | Genre of the game | object |
| Publisher | Publisher of the game | object |
| NA\_Sales | Sales in North America (in millions) | float64 |
| EU\_Sales | Sales in Europe (in millions) | float64 |
| JP\_Sales | Sales in Japan (in millions) | float64 |
| Other\_Sales | Sales in the rest of the world (in millions) | float64 |
| Global\_Sales | Total worldwide sales | float64 |
| Critic\_Score | Aggregate score compiled by Metacritic staff | float64 |
| Critic\_Count | The number of critics used in coming up with the Criticscore | float64 |
| User\_Score | Score by Metacritic's subscribers | object |
| User\_Count | Number of users who gave the userscore | float64 |
| Developer | Party responsible for creating the game | object |
| Rating | The ESRB ratings | object |

Data set contains total of 16 fields and 16719 data records. Platform, Genre, Publisher, Developer and Rating are all categorical fields and NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales, Global\_Scales, Critic\_Score, Critic\_Count, User\_Score and User\_Count are all numeric fields.

# Architecture

In general, a framework is desired to function as a web application that can be used with any device, regardless of the operating system. To understand the traits and behaviors of the elements with the objective component, a exploratory data analysis is first completed to the information edge. Following the completion of data cleaning and feature engineering to the data set to create the accuracy of the final model, information outline conduct is understood.

The information outline was then familiar to grouping models, who calculated its exactness, accuracy, review, and f1 scores. In the final iteration of the program, the best model is used. The Streamlit library is used to run the application. The program is finally uploaded to the web using features provided by GitHub and Streamlit.

## Data Cleaning

First, all the null values in the dataset are observed using isnull() function. There were 2 null values in the Name field, 269 null values in the Year\_of\_Release field, 2 in Genre, 54 in Publisher. Critic\_Score, Critic\_Count, User\_Score, User\_Count, Developer and Rating fields have larger missing values. To be exact 8582, 8582, 6704, 9129, 6623 and 6769 in the given order.

In the process of handling missing values, fields with small portion of missing values were replaced with the field’s median and the mode value according to their data type. If the data is categorical, mode is used and if it is numerical, median is used. Then all the missing values which had the missing value count more than 5000 were dropped from the data frame.

obj\_features =['Name', 'Platform', 'Genre' , 'Publisher', 'Developer', 'Rating', 'User\_Score']

num\_features = list(set(df.columns) - set(obj\_features))

for f in num\_features:

  df[f] = np.where(df[f] < 0, np.NaN , df[f])

for f in num\_features:

  nulls = df[f].isnull().sum()

  if (nulls < 5000):

    median=df[f].median()

    df[f].fillna(value=median, inplace=True)

for f in obj\_features:

  nulls = df[f].isnull().sum()

  if (nulls < 5000):

    mode=df[f].mode()[0]

    df[f].fillna(value=mode, inplace=True)

## Data Pre-processing

In order to create the predicting classifier field, a new field is introduced with the name Successful. All the data points that their Global\_Sales are higher than 1 million are considered successful. They were given the Boolean value 1 and all others who do not exceed 1 million sales globally were given the Boolean value 0. These were added under the Successful field. Project’s prime objective is to predict the successfulness of the video game by checking if it exceeds 1 million sales globally.

y = df[['Successful']].values

X = df.drop(columns=['Successful', 'Name', 'NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Other\_Sales', 'Global\_Sales', 'Year\_of\_Release'])

All the unnecessary features that are not needed to do the prediction were dropped from the database as well such as Name, NA\_Sales, EU\_Sales, JP\_Sales, Other\_Sales, Global\_Sales etc.

Remaining categircal fields were Platfofrm, Genre, Publisher, Developer and Rating. Label encoding is used to encode each field to a number.

from sklearn.preprocessing import LabelEncoder

def label\_encode\_columns(df, columns):

    encoders = {}

    for col in columns:

        le = LabelEncoder().fit(df[col])

        df[col] = le.transform(df[col])

    return df

X = label\_encode\_columns(X,  ['Platform', 'Genre', 'Publisher', 'Developer', 'Rating'])

# Modelling

Three classifier modelling models were used in the project and from those 3, the best method is evaluated and selected to use in the final application. Three techniques were Decision Tree, Random Forest and Naïve Bayes.

As for the first step, decision tree approach is used. Decision Tree is a supervised learning method that can be applied to classification and regression issues, but it is most frequently used to address classification issues. It is a tree-structured classifier, where internal nodes stand in for the dataset's features, branches for the rules of classification, and each leaf node for the result.

from sklearn import tree

dtc = tree.DecisionTreeClassifier(random\_state=0)

dtc.fit(X\_train, y\_train)

dtc\_eval = evaluate\_model(dtc, X\_test, y\_test)

Decision Tree Results:

Accuracy: 0.8226827864133563

Precision: 0.5341246290801187

Recall: 0.5438066465256798

F1 Score: 0.5389221556886227

Cohens Kappa Score: 0.4291683293819394

Area Under Curve: 0.7160711753254287

Then, RandomForest approach used to model the data set. This approach, as implied by its name, consists of a huge number of separate decision trees that work together. Each each tree in the random forest emits a class forecast, and the class that receives the most votes becomes the expectation of our model. The low correlation between models is crucial. Similar to how assets with low correlations combine to build a portfolio that is larger than the sum of its parts, uncorrelated models have the capacity to provide ensemble estimates that are more accurate than any of the individual projections. The attractive result is due to the fact that the trees protect one other from their individual flaws as long as they do not routinely all err in the same direction. The group of trees will move in the right direction since many of them will be correct but others may be mistaken.

A random forest, which functions as a meta estimator, uses averaging to improve prediction accuracy and decrease overfitting by integrating several decision tree classifiers on various subsamples of the dataset. Using a random state, the train data was initially divided into X and Y train test data. From the sklearn.ensemble dataset, RandomForestClassifier was chosen to create a model that forecasts churning.

rfc = RandomForestClassifier(n\_estimators=100)

rfc.fit(X\_train, y\_train)

rfc\_eval = evaluate\_model(rfc, X\_test, y\_test)

RandomForest Results:

Accuracy: 0.871042026482441

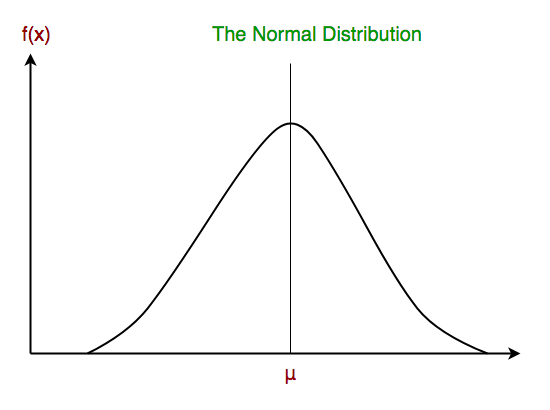
Precision: 0.7635467980295566

Recall: 0.46827794561933533

F1 Score: 0.5805243445692885

Cohens Kappa Score: 0.5094531829161897

Area Under Curve: 0.8709189790840293

Finally, Naïve Bayes approach used to model the data set. A group of classification algorithms built on the Bayes' Theorem are known as naive Bayes classifiers. It is a family of algorithms rather than a single method, and they are all based on the idea that every pair of features being classified is independent of the other. The core tenet of Naive Bayes is that each characteristic contributes equally and independently to the result. In practical applications, Naive Bayes' presumptions are frequently incorrect. Although the independence assumption is false in theory, it frequently holds true in practice.

Continuous values connected to each feature in Gaussian Naive Bayes are presumptively distributed in a Gaussian manner. Normal distribution is another name for a Gaussian distribution.

naive\_bayes = GaussianNB()

naive\_bayes.fit(X\_train , y\_train)

naive\_bayes\_eval = evaluate\_model(naive\_bayes, X\_test, y\_test)

Naive Bayes Results:

Accuracy: 0.8186528497409327

Precision: 0.5439560439560439

Recall: 0.2990936555891239

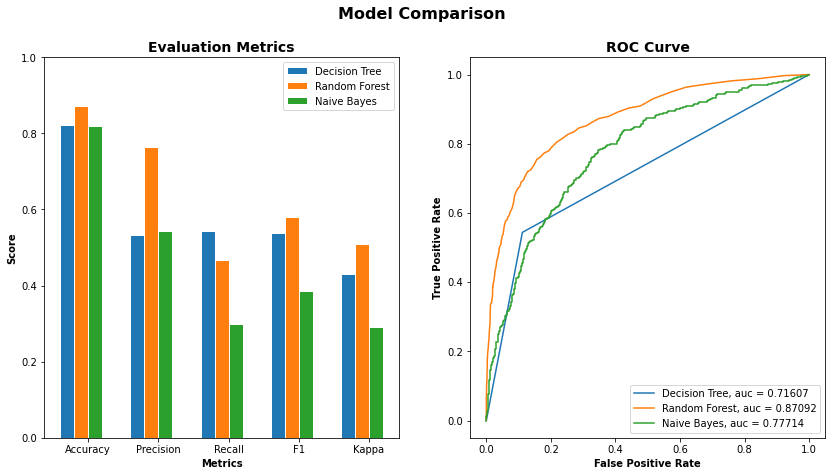
F1 Score: 0.3859649122807018

Cohens Kappa Score: 0.28995960274955657

Area Under Curve: 0.7771355391008754

# Evaluation

By comparing all 3 Models an evaluation is done.



As it is clearly visible, all 3 models have Accuracy score higher than 0.8. But the Random forest approach have much higher score than other two. Since the class distribution is considerably fair in this data set, accuracy can be taken as a good evaluation metric.

When it comes to precision, both Decision Tree and Naïve Bayes have much less score than the Random Forest model. Without no hesitation, we can come to the conclusion that the Random Forest is best suited model comparatively.

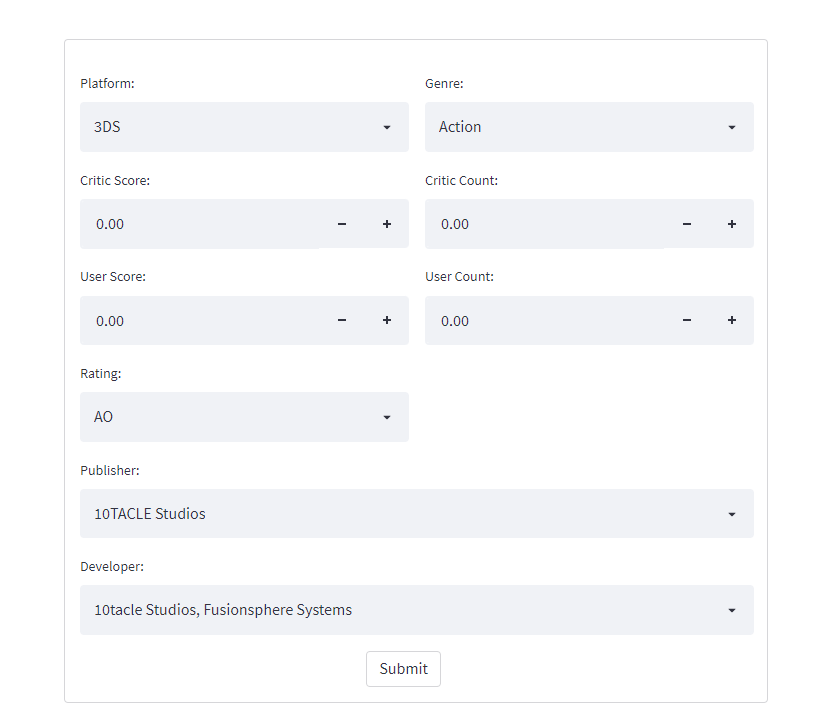
Both in F1 score and in Kappa score, Random Forest has higher score but in Recall, it has a small drawback. Decision tree has the highest score in Recall among these three models.

As we can see, area under the curve is much higher for the random forest approach than other two methods.

Hence, we can come to the conclusion that RandomForest model gives the best overall Score for the dataset. Therefore, it is chosen to use in the application.

# Development

The dashboard will be made involving Streamlit as its improvement apparatus. It is a totally Python structure that can be joined with a similar Python document that is utilized for information investigation with a couple of extra lines of code and requires no skill in web building. By utilizing Streamliy, the development time for dashboards can be sliced, saving additional opportunities for information investigation rather than building a web application. Input gadgets, media parts, and various extra showcase components are all important for the irrefutable Streamlit framework. Charting bundles like 4 matplotlib, Vega-Lite, and Altair are additionally viable with Streamlit. Furthermore, it highlights reserving choices, permitting you to speed up the speed of the program by storing tedious tasks. Streamlit is utilized to develop an intuitive dashboard, which is then distributed to the web utilizing GitHub. The Web application will consequently be refreshed with any update that we submit to the vault. To conjecture in the event that an information point will have 1 million deals or, excessive information values are required.



The code will construct a data frame object for those entered values once the appropriate fields have been filled in and submitted. Then, using the predict() method in the model, it will determine whether or not this client will leave in the near future.

            # Random Forest

            rf\_prediction = model\_rf.predict(dp\_new)

If the outcome is churn, the business can take the required steps to keep the customer by making offers or acting appropriately.