# Project: Game Application Success Prediction.

# Team ID: 25

# Team Information

Member 1: Khaled Galal Yehia

ID 1: 20191700223

Member 2: Nahed Mohamed Mostafa

ID 2: 20191700916

Member 3: Hager Ahmed Mahmoud

ID 3: 20191700723

Member 4: Mohamed Essam Ali

ID 4: 20201700711

Member 5: Hazem Abd Elhamid Malek

ID 5: 20191700020

Member 6: Omar Ahmed Farouk

ID 6: 20191700400

# Preprocessing

1. **Remove duplicate rows and unique columns:**

First, we figured out that the columns (URL, ID, NAME, Subtitle, ICON URL) has duplicate values and that’s wrong because its guarantee that those features are unique features because each game is unique and must have one record, so we dropped the duplicates then we dropped those columns because the unique features wouldn’t help the model to learn.

The number of duplicates = 43

Subtitle had nulls values = 1465

Before removing duplicate rows, the score (For the best model) was 23.60% and the mean square error was 0.45,

After the score = 23.55% and the mean square error =0.46, but if we doesn’t remove the duplicates the data will be unclean so we removed them.

1. **In-app Purchases**:

Nulls = 2039

Filled nulls with zeros.

Get two new features from this feature which are Average In-app purchases “Mean”, and Length of In-app purchases “Count”.

1. **Age Rating**:

The unique classes for this feature were four so, we applied the factorize on this feature, so each age has its own class.

1. **Developer**:

Get new feature from this feature which is the count of developers.

1. **Original Release Date, Current Version Release Date:**

Get the year feature (original year – current year) from each date of those columns.

Get new feature from Original and Current year “diff years” the difference between Original and current year.

Last we dropped the Original Release Date and Current Version Release Date features.

1. **Language**

- Nulls =11

- filled nulls with ‘EN’.

- applying one hot encoding from scratch of this column because it is a dictionary and can’t use get-dummies.

- get new another new feature which is the count of languages.

1. **Primary Genre**

- applying one hot encoding using get-dummies and concatenating the new feature that we get from get-dummies with the original data.

1. **Genres**

- applying one hot encoding again to ensure that all categories have columns because not all the categories in these genres was in the primary genres and vice versa.

- get new another new feature also which is the count of genres.

- after the preprocessing on the primary genre and genres features, we dropped them.

# Feature selection

- drop column if all its values in column is 1 because its correlation is null

- apply anova test on all categorical columns if p value of it > 0.05 🡺 drop this column

* Correlation

- apply correlation on all numerical values after analysis its clear that any correlation < 0.03 not affect y

So, drop these columns.

# Scaling

* Use Min-Max scaler for x values.

# Splits sizes

* Test = 20%
* Train = 80%
* Hyper parameters: shuffle = True, random state = 30

# Features we used:

* [‘User Rating Count’, ‘In-app Purchases’, ‘Age Rating’, ‘Size’, ‘Average purchases’, ‘Original year’, ‘Current year’, ‘lang count’, ‘FR’, ‘DE’, ‘JA’, ‘KO’, ‘ZH’, ‘ES’, ‘TH’, ‘RU’, ‘ID’, ‘PT’, ‘AR’, ‘Entertainment’, ‘Games’, ‘Productivity’, ‘Reference’, ‘Simulation’, ‘Casual’, ‘Puzzle’, ‘Action’, ‘Adventure’, ‘Board’, Family’, ‘Kids & Family’]

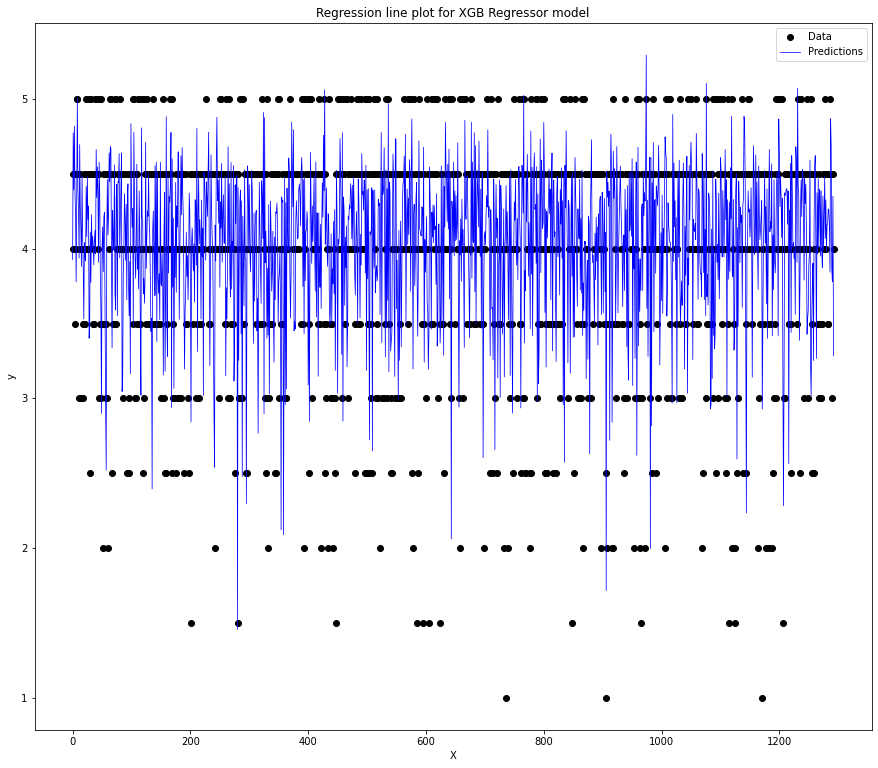
# Models

# We have used 4 regression techniques:

1. **XGB Regressor:** **is a type of gradient boosting algorithm used for regression tasks in machine learning. It works by combining multiple weak decision tree models into a single strong model that can make more accurate predictions. The algorithm iteratively adds trees to the model, focusing on the areas where the previous trees made mistakes. XGB Regressor is known for its high accuracy and speed, as well as its ability to handle large datasets and complex features.**

**Accuracy of XGB Regressor: 15.46%**

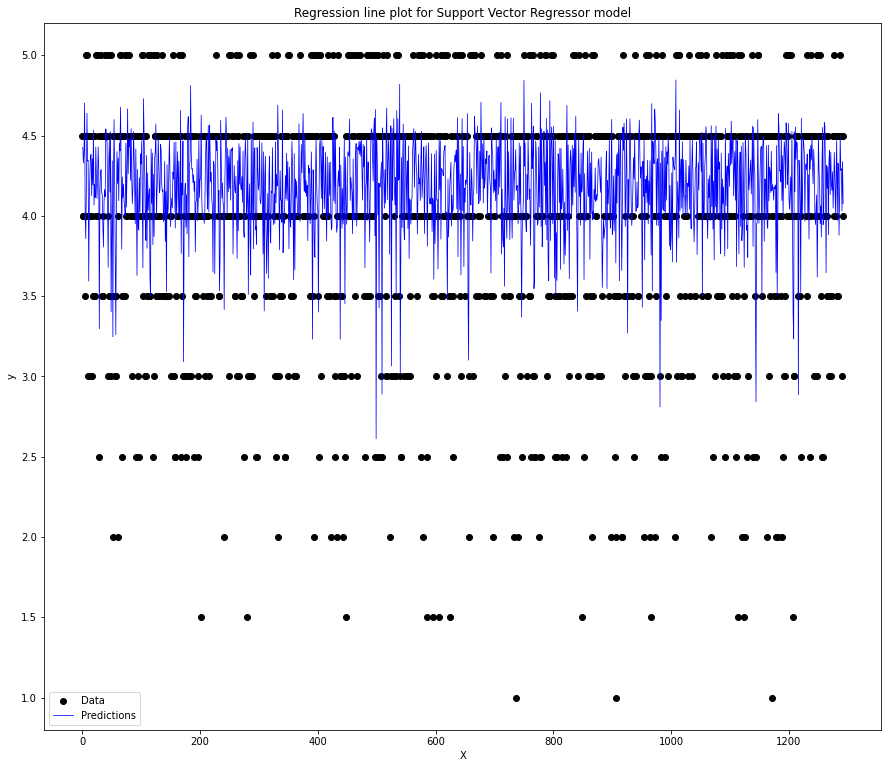
**MSE: 0.500948452265277**



1. **Support Vector Regressor: is a machine learning technique for regression tasks. It finds a hyperplane that maximizes the margin between predicted and actual values using a kernel function. The resulting model makes accurate predictions on unseen data.**

**Accuracy of Support Vector Regressor: 8.95%**

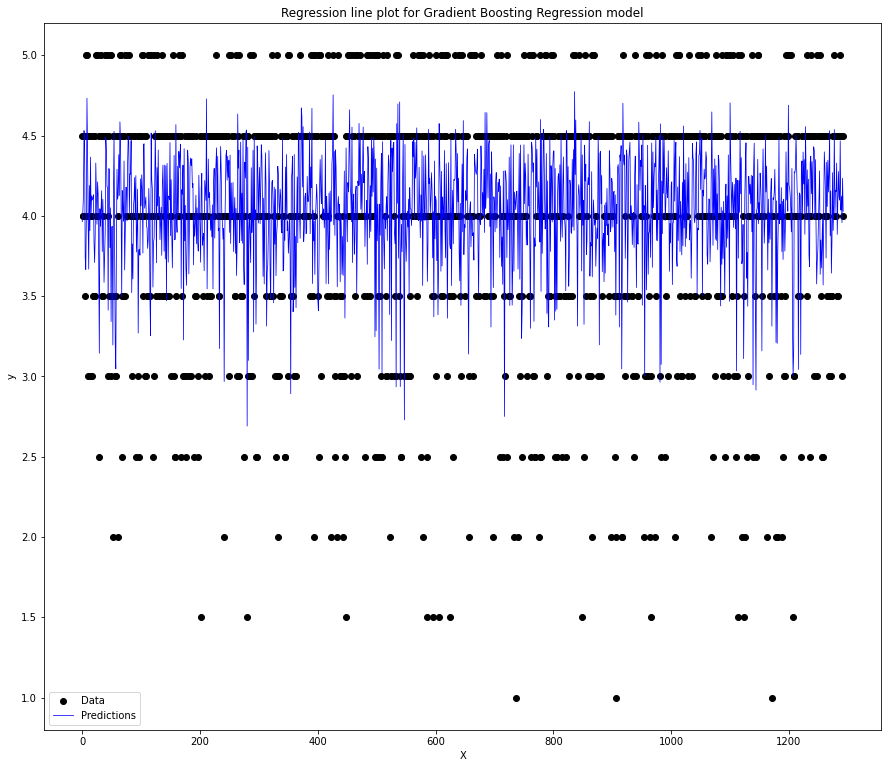
**MSE: 0.539543358826231**



1. **Gradient Boosting Regression: is a machine learning technique for regression tasks that uses an ensemble of decision trees. It works by sequentially adding trees to the ensemble, with each tree trained to correct the errors of the previous tree. The algorithm uses a loss function to measure the difference between predicted and actual values, and gradient descent to optimize the model parameters. The resulting model is able to make accurate predictions on unseen data and is robust to overfitting.**

**Accuracy of Gradient Boosting Regression: 23.55%**

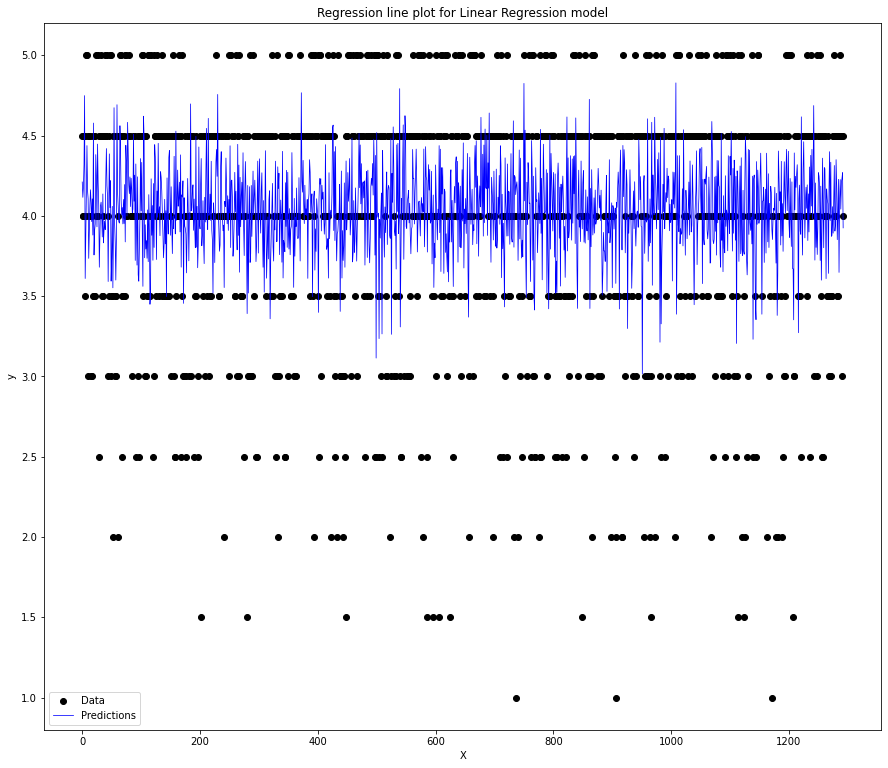
**MSE: 0.45302855666455555**



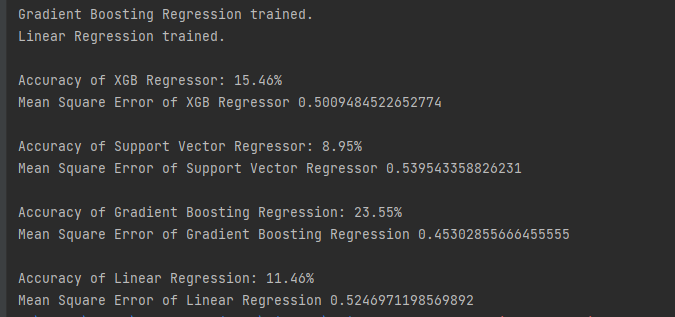
1. **Linear Regression: is used for predicting a continuous numerical output based on one or more input variables. It works by fitting a linear equation to the input data, where the output is a linear combination of the input variables with some constant offset. The algorithm uses a cost function to minimize the distance between the predicted values and the actual values. The resulting model can be used to make predictions on new data and to understand the relationship between the input and output variables.**

**Accuracy of Linear Regression: 11.46%**

**MSE: 0.5246971198569892**



# Models:

* We tried a lot of models and got the highest accuracies with those models.

1. Accuracy of XGB Regressor: 15.46%

Mean Square Error of XGB Regressor 0.50

1. Accuracy of Support Vector Regressor: 8.95%

Mean Square Error of Support Vector Regressor 0.53

1. Accuracy of Gradient Boosting Regression: 23.55%

Mean Square Error of Gradient Boosting Regression 0.45

1. Accuracy of Linear Regression: 11.46%

Mean Square Error of Linear Regression 0.52

Bonus

Apply NLP technique on column description

1-Clean text

- Removing punctuation  
- Removing whitespace and newlines

2-Remove stop words

3-TF-IDF Vectorizer

4-Return top k keywords from a doc using TF-IDF method

-generate tf-idf for document  
 - sort the tf-idf vectors by descending order of scores  
 - extract only TOP\_K\_KEYWORDS that is 1 word in a list

5-convert to string , apply encoder-decoder add this column to original data then drop description column