Team ID: CS_43

GAME SUCCESS PREDICTION

TEAM MEMBERS

NAME	ID	DEPARTMENT
عبدالرحمن سيد جابر أحمد	20201701089	CS
رقيه محمد ابراهيم مصطفي عبده	20201701253	CS
نورهان ايمن محمد عبدالرحمن	20201700939	CS
حنين ابراهيم امام عكاشه	20201700230	CS
مريم احمد اسماعيل محمود	20201700800	CS
هبه طارق كمال عبدالمطلب	20201700959	CS

TABLE OF CONTENTS

1- EXPLORING DATASET

1.1 - Dataset Shape

After reading the dataset into a pandas data frame, we start exploring by printing it's shape which is **(5214, 18)**. This step will help us later to decide which features could be used to train our models.

1.2 - Dataset Distribution

This step is important to determine if the dataset is imbalanced which might lead to overfitting or underfitting.

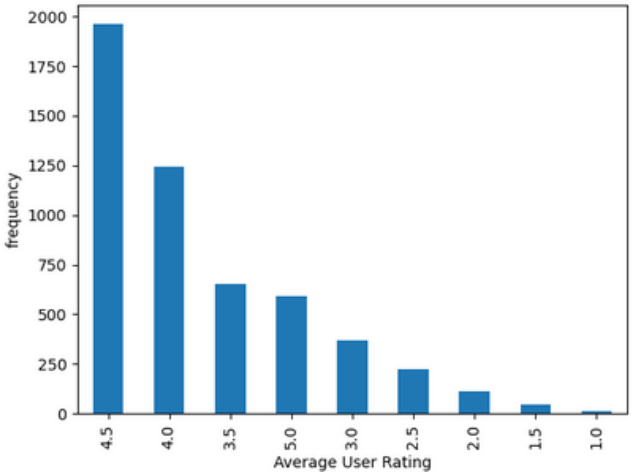


fig 1 - Regression Dataset Distribution

Looking at fig 1, we notice that the majority of the regression dataset is 4.5 and there is a clear skewness. therefore, we consider oversampling and under sampling to solve the issue.

Meanwhile, fig 2 does not show such skewness and the distribution should not be a problem if we used stratified splitting.

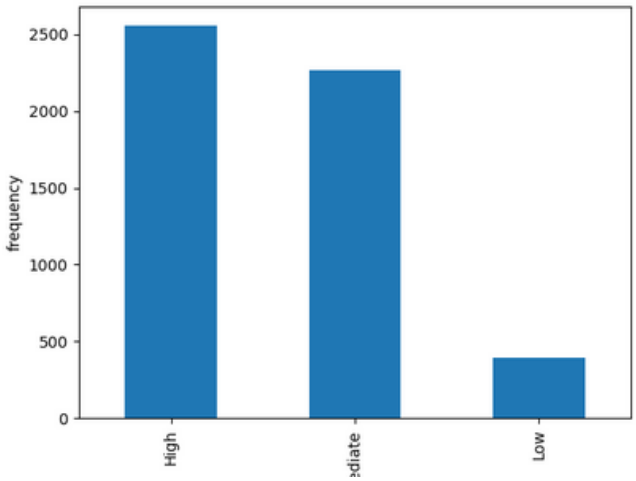


fig 2 - Classification Dataset Distribution

1- EXPLORING DATASET

1.3 - Features Types

URL	object
ID	int64
Name	object
Subtitle	object
Icon URL	object
User Rating Count	int64
Price	float64
In-app Purchases	object
Description	object
Developer	object
Age Rating	object
Languages	object
Size	int64
Primary Genre	object
Genres	object
Original Release Date	object
Current Version Release Date	object
Average User Rating	float64

fig 3 - Regression Dataset Features Types

URL	object
ID	int64
Name	object
Subtitle	object
Icon URL	object
User Rating Count	int64
Price	float64
In-app Purchases	object
Description	object
Developer	object
Age Rating	object
Languages	object
Size	int64
Primary Genre	object
Genres	object
Original Release Date	object
Current Version Release Date	object
Rate	object

fig 4 - Classification Dataset Features Types

1.4 - Outliers Detection

Using IQR, we test some features and plot the results. this will be handeld on train later.

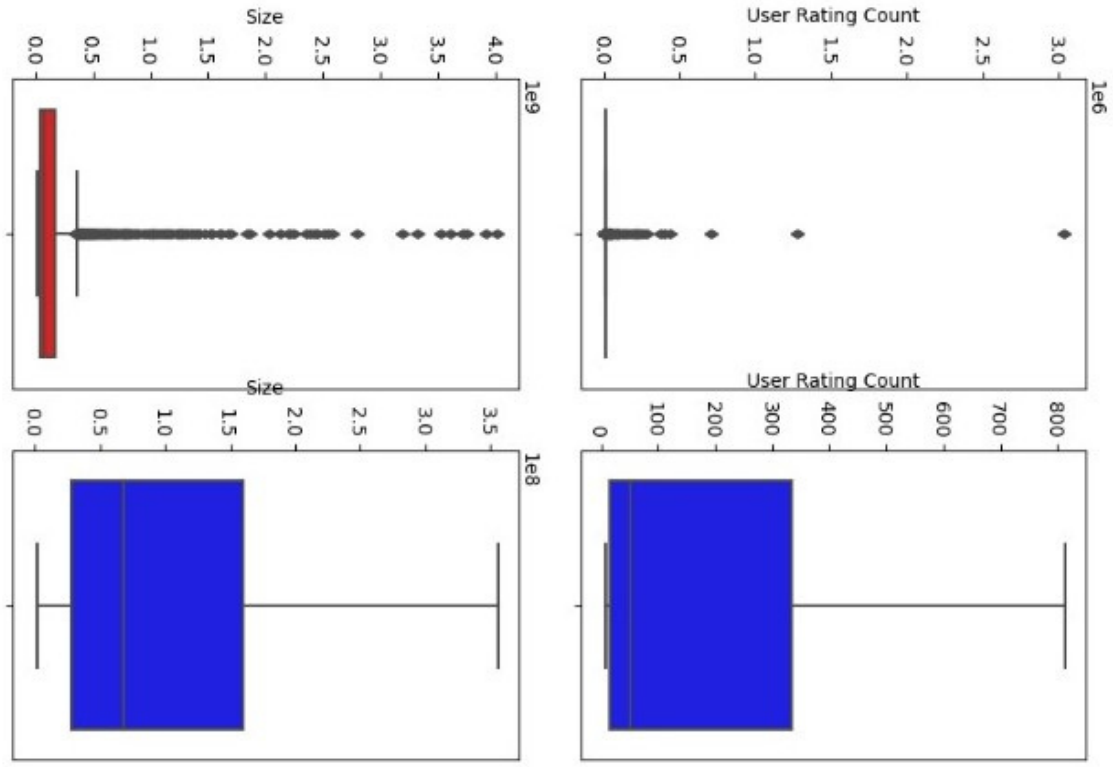


fig 5 - Outliers Detection

2- PREPROCESSING

2.1 - General Preprocessing

- Drop null & duplicate rows: keep only the first of the duplicates

- Train Test Split**:** 80% train, 20% test.
  - In Classification, Setting the Stratify parameter improved the accuracy from 67% to 69% using the RBF model in testing.

- Drop ID, URL, Icon URL, and Name column: These columns have unique value equal to number of rows in the data set. Therefore, we can't connect them to the target column.

- Drop Subtitle column: the column has more than 70% null values.

- Drop Primary Genre column: the column values are repeated in Genres column.

- Fill null Values:
  - filled nulls in Languages with EN because it is the most common languages.
  - filled nulls in Price with 0 considering it as a free game.
  - filled nulls in In-app purchases with 0.
  - filled nulls in User Rating Count with 0.
  - filled nulls in Size with 0.
  - filled nulls in Developer, Age Rating with the train dataset mode.
  - filled nulls in dates, with the train dataset mean.

- Convert date columns to integers.

- Encode age rating: age_rating_map = {4: 1, 9: 2, 12: 3, 17: 4}

- Extract game difficulty from description column: three categories: Hard, Intermediate, Easy.

2- PREPROCESSING

2.2 - Train Dataset

- Encode Languages and Genres: using One Hot Encoding

- Scaling: used MinMaxScaler on User Rating Count, In-app Purchases, Size, Original Release Date, and Current Version Release Date features.

- Outliers removal: reassigned the outliers to the min/max values of the distribution.

- Count Develpor games: count and replace the names with the count.

2.3 - Features Selection

- K-best (SelectKBest) is a technique that involves selecting of the top k features based on a scoring function. The scoring function measures the amount of information that one variable (feature) provides about another variable (target) in a dataset.
- We set k=100 in the regression phase and k=64 in the classification phase, as shown in the graphs below.
- use same technique as regression data set almost categorical

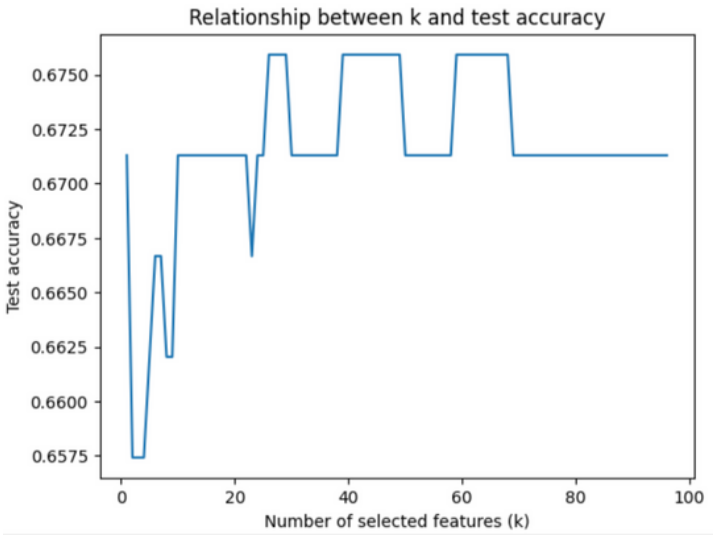


fig 6 - Classification feature selection

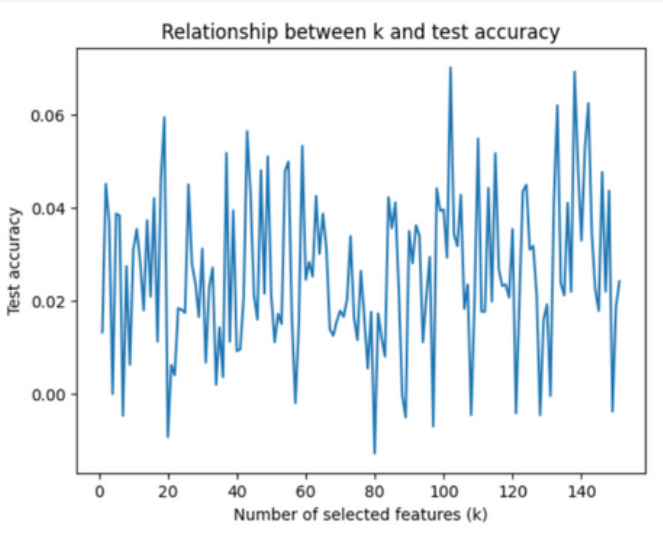


fig 7 - Regression feature selection

2.4 - Test Dataset

- Encode Languages and Genres: making sure to include features seen at training and ignore new categories.

- Scaling: used the saved scaler from the training.

- Count Develpor games: used counts from training and replaced the names with the count. new names will be replaced with zero.

3- REGRESSION MODELS

Model	Train	Test
Random Forest	• R2 score: 0.88 • MSE: 0.037	• R2 score: 0.08 • MSE: 0.32
Decision Tree Regressor	• R2 score: 0.16 • MSE: 0.25	• R2 score: 0.003 • MSE: 0.35
Gradient Boosting Regressor	• R2 score: 0.57 • MSE: 0.13	• R2 score: 0.11 • MSE: 0.31

3- REGRESSION MODELS

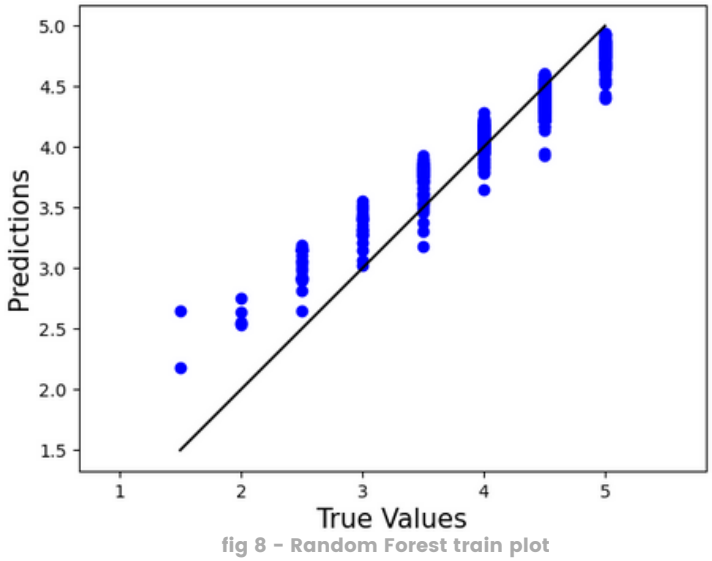


fig 8 - Random Forest train plot

After training, we plot the actual Average User Rating vs the predicted values for each model using the train dataset.

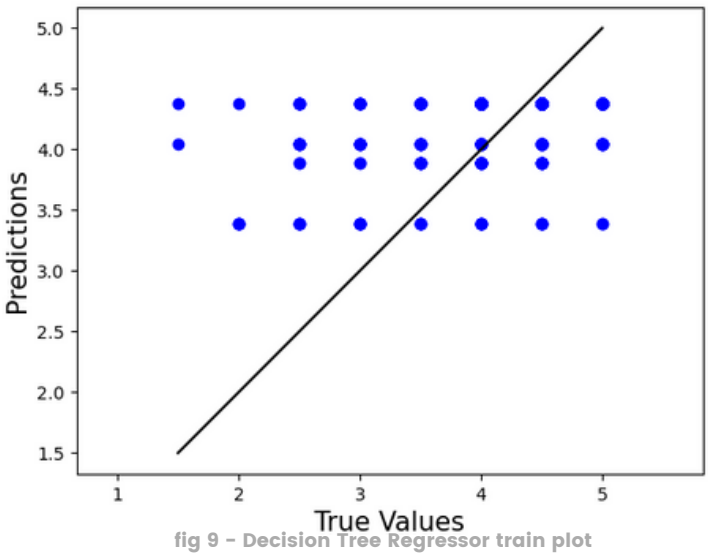


fig 9 - Decision Tree Regressor train plot

The scattered blue dots represent the predicted values against the actual, and the black line represents the distribution.

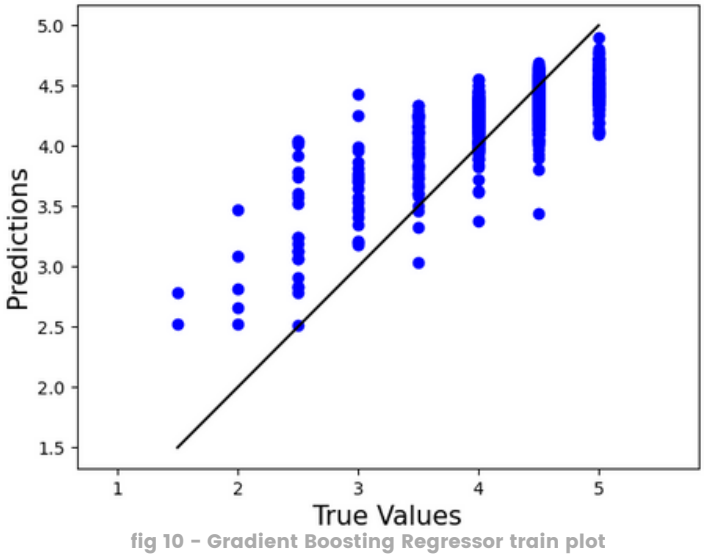


fig 10 - Gradient Boosting Regressor train plot

all three models seem to overlook values lower than or equal 2. this could lead to less accurecy in testing.

3- REGRESSION MODELS

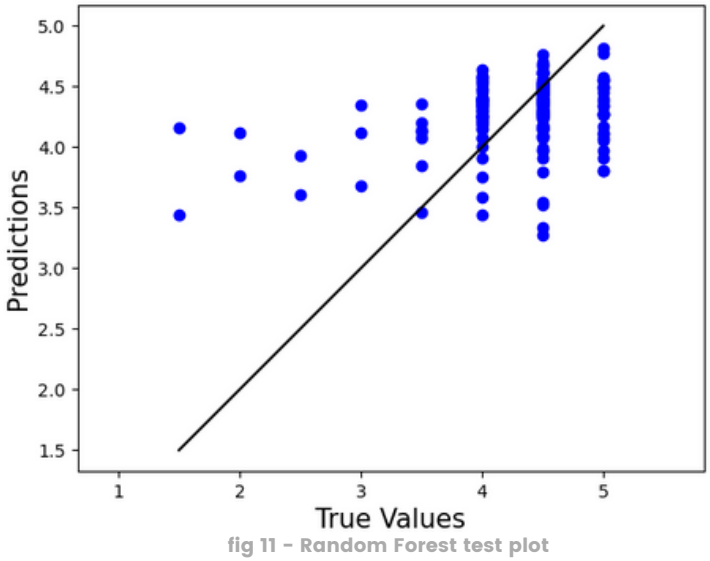


fig 11 - Random Forest test plot

For testing, we do the same but using the test dataset.

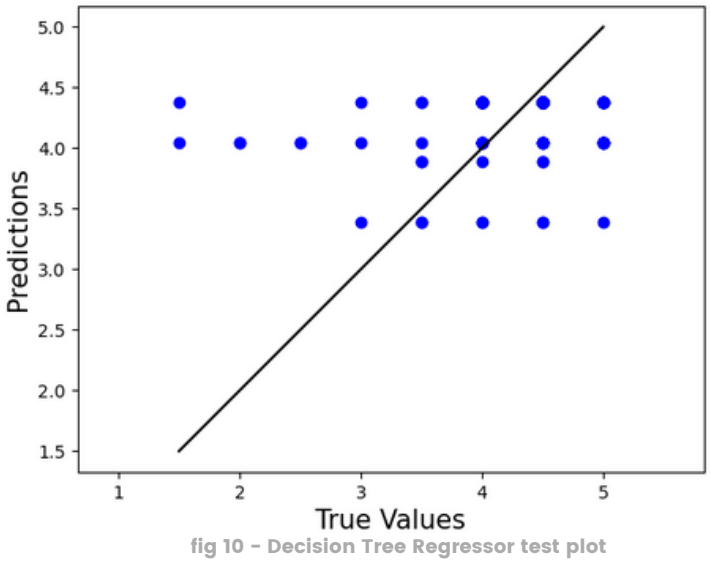


fig 10 - Decision Tree Regressor test plot

Obviously, all three models are not accurately predicting the data.

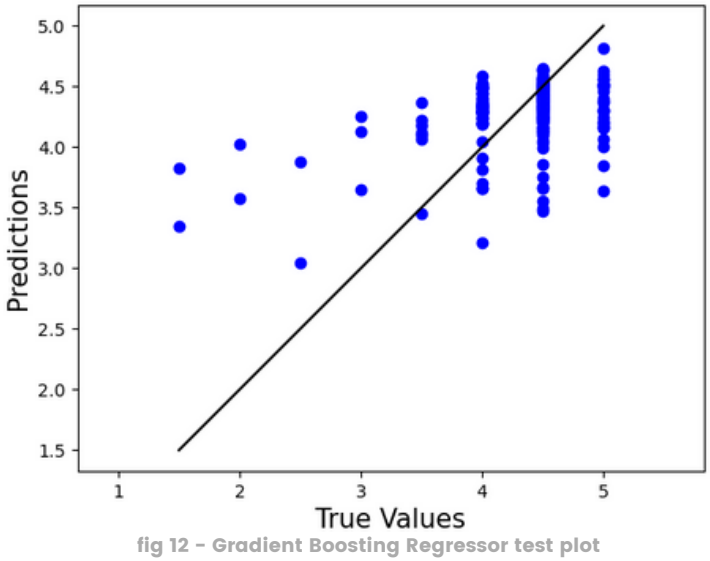


fig 12 - Gradient Boosting Regressor test plot

We can consider it as overfitting cases.

4- CLASSIFICATION MODELS

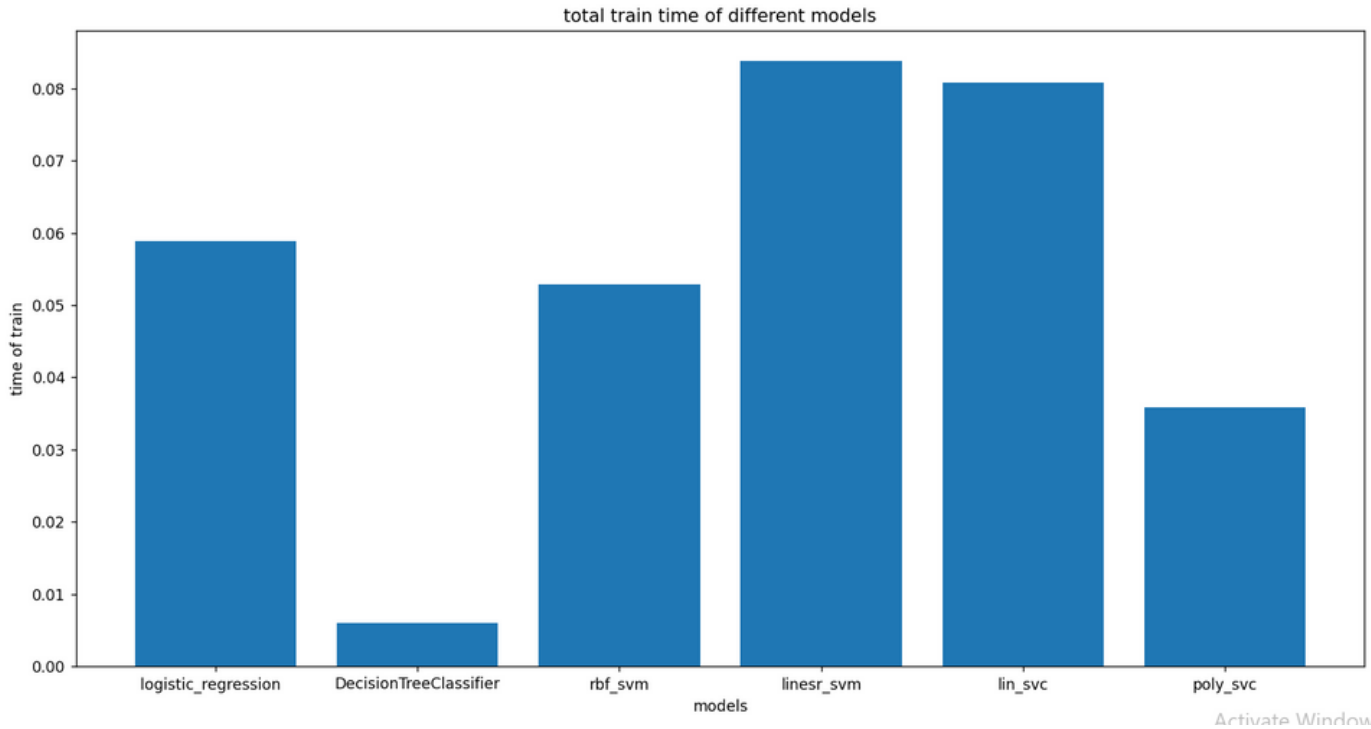
Model	Train	Test
SVC (RBF Kernal)	• Accuracy: 0.81	• Accuracy: 0.69
SVC (Linear Kernal)	• Accuracy: 0.70	• Accuracy: 0.63
Linear SVC	• Accuracy: 0.72	• Accuracy: 0.61
SVC with polynomial (degree 3) kernel	• Accuracy: 0.69	• Accuracy: 0.67
logistic_regression	• Accuracy: 0.71	• Accuracy: 0.68
DecisionTreeClassifier	• Accuracy: 0.99	• Accuracy: 0.62

4- CLASSIFICATION MODELS

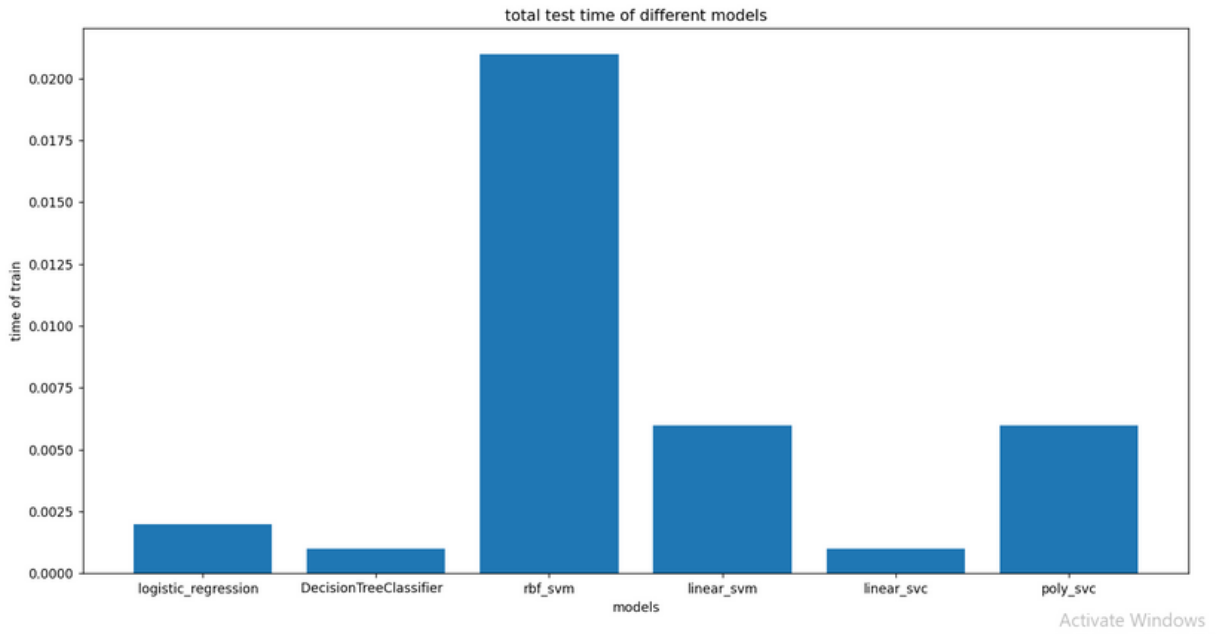
	C = 100	C = 1	C = 0.01
SVC (RBF Kernal)	Train: 0.96 Test: 0.62	Train: 0.83 Test: 0.64	Train: 0.67 Test: 0.67
SVC (Linear Kernal)	Train: 0.70 Test: 0.66	Train: 0.68 Test: 0.68	Train: 0.67 Test: 0.67
Linear SVC	Train: 0.68 Test: 0.63	Train: 0.70 Test: 0.65	Train: 0.69 Test: 0.68
SVC with polynomial (degree 3) kernel	Train: 0.68 Test: 0.65	Train: 0.69 Test: 0.66	Train: 0.67 Test: 0.67
	Degree = 10	**Degree = 3**	**Degree = 1**
SVC with polynomial kernel	Train: 0.70 Test: 0.67	Train: 0.69 Test: 0.66	Train: 0.68 Test: 0.67
	Gamma = 1	**Gamma = 0.8**	**Gamma = 0.1**
SVC (RBF Kernal)	Train: 0.83 Test: 0.69	Train: 0.83 Test: 0.64	Train: 0.69 Test: 0.68

5- BAR PLOTS

5.1- train time

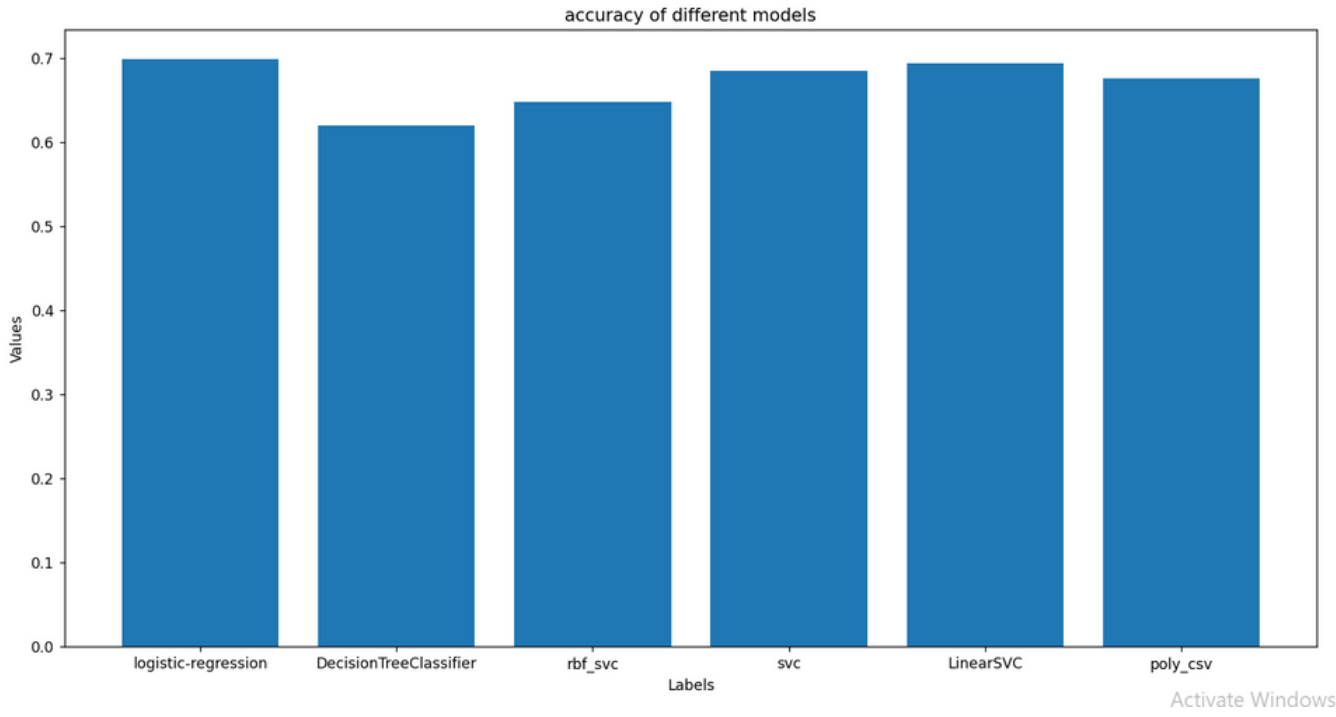


5.2 - test time

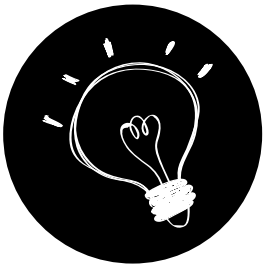


5- BAR PLOTS

5.3 - accuracy

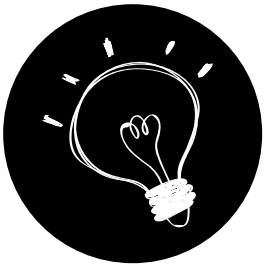


6- CONCLUSION



No. 01 — Dataset

The lack of samples for some ratings/ classes makes it hard to distinguish whether an app will be given high or low ratings since we only have samples for the high ratings. Also, most of the features uniquely identifies the games which we could not use.



No. 02 — Models

- Regression Models: Random Forest Models and Gradient Boosting performed well in training. But Gradient Boosting performed better in testing.
- Classification Models: There is no big difference between the four models, But SVM with RBF kernel showed less overfitting behavior.

No. 03 — Future Work

- More feature engineering to extract meaningful features to improve the models learning.
- More hyperparameters tuning.