Predicting Mobile Game Success

Milestone 1

By: Team ID: CS 29

AGENDA

01 **Objective**

02 **Team**

03 **Preprocessing**

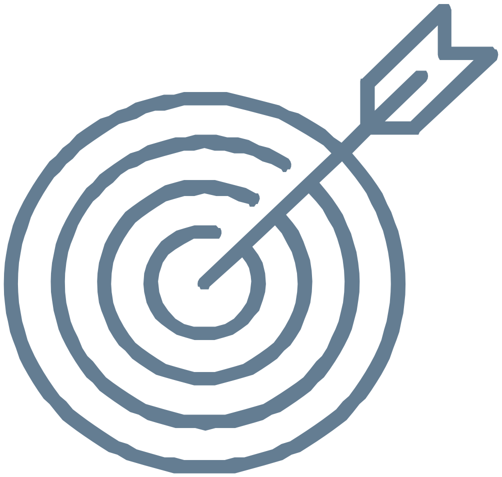
04 **Regression**

05 **Classification**

Objective

The objective of the Milestone 1 is to clean the data, applying pre-processing, feature selection, feature scaling, feature engineering, regression.

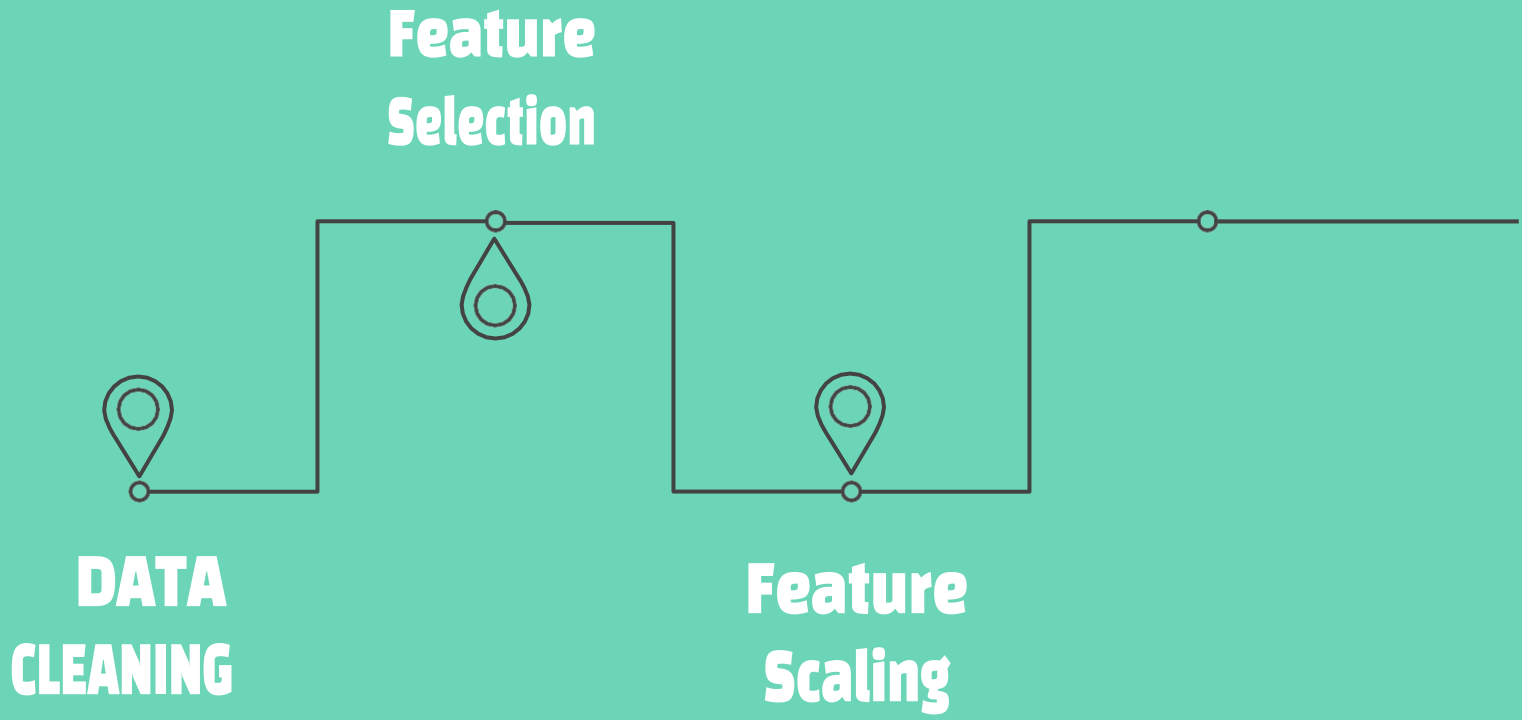
The objective of the Milestone 2 is to Classify games according to its Rate.



TEAM

Nour Adel	20201700931
Ahmed Osama	20201700022
Ahmed Rabie	20201700039
Ahmed Tarek	20201700055
Sondos Mohamed	20201700372
Radwa Mansour	20201700261

Preprocessing



DATA CLEANING

- Dataset Size = 5214
- Number of duplicates = 43
- Number of Nan values = 5799
- URL, ID, Name, Icon URL, Description
  (All these Columns are Unique)
- Subtitle Column Contain almost Nan Values “3749.”
- In-app Purchases Contains 2039 Nan Values.
- Languages Column Contains 11 Nan Values.

	Name	dtypes	Missing	Uniques
0	URL	object	0	5171
1	ID	int64	0	5171
2	Name	object	0	5171
3	Subtitle	object	3749	1399
4	Icon URL	object	0	5171
5	User Rating Count	int64	0	1410
6	Price	float64	0	17
7	In-app Purchases	object	2039	2052
8	Description	object	0	5099
9	Developer	object	0	3084
10	Age Rating	object	0	4
11	Languages	object	11	580
12	Size	int64	0	5081
13	Primary Genre	object	0	19
14	Genres	object	0	507
15	Original Release Date	object	0	2281
16	Current Version Release Date	object	0	1976
17	Average User Rating	float64	0	9

DATA CLEANING

- Check duplicates, then remove them.

```
dataset.duplicated().sum() # Duplicates =  43
# Drop duplicate rows
dataset = dataset.drop_duplicates() # Duplicates =  0
```

- Check missing Values, then replace them with Mean or Mode or Zero.

```
URL                              0
ID                               0
Name                             0
Subtitle                      3717
Icon URL                         0
User Rating Count                0
Price                            0
In-app Purchases              2025
Description                      0
Developer                        0
Age Rating                       0
Languages                       11
Size                             0
Primary Genre                    0
Genres                           0
Original Release Date            0
Current Version Release Date     0
Average User Rating              0
```

DATA CLEANING

each cell in the **'In-app Purchases'** column contains prices of available in-app purchases, First, we sum the prices listed in each cell of the "In-app Purchases" column and calculate the mean value of the column. Then, we replace any missing values in the column with this mean value.

```
dataset['In-app Purchases'].isnull().sum()
missing values = 2025
```

Before

In-app Purchases
29.99, 19.99, 9.99, 29.99, 29.99, 8.99, 4.99, ...
NaN
NaN
NaN
NaN
...
1.99, 1.99, 1.99, 1.99, 1.99, 1.99, 2.99, 7.99...
2.99, 4.99, 1.99, 2.99, 5.99, 9.99, 11.99, 19.99

After

In-app Purchases
307.840000
61.674269
61.674269
61.674269
61.674269
...
53.860000
60.920000
61.674269
121.900000
88.930000

DATA CLEANING

each cell in the ‘**Languages**’ column contains a lot of languages that available in the game First, we separate each language in each cell and calculate the mode value of all languages in the column. Then, we replace any missing values in the column with this mode value which it is English

```
dataset['Languages'].isnull().sum()
missing values = 11
```

Before	After
Languages	**Languages**
EN, FR, DE, JA, KO, ZH, ES, TH, ZH, VI	EN, FR, DE, JA, KO, ZH, ES, TH, ZH, VI
EN	EN
EN, ZH	EN, ZH
NaN	EN
NaN	EN

DATA CLEANING

each cell in the **'Age Rating'** column contains either 4+, 9+, 12+ or 17+.

We perform the following steps to make to usable:

- we remove the positive sign(+) next to each age.
- we change data type of column from 'object' to 'int'.
- we redefined column name from age rating to Min Age for better usages

Before	After
Age Rating	Min_Age
12+	12
12+	12
4+	4
9+	9
12+	12

DATA CLEANING

Before

	URL	Name	Subtitle	Icon URL	Description
0	https://apps.apple.com/us/app/heir-of-light/id...	HEIR OF LIGHT	Dark Fantasy RPG	https://is3-ssl.mzstatic.com/image/thumb/Purpl...	A Dark Fantasy, Collectible RPG\n\nDarkness ha...
1	https://apps.apple.com/us/app/endgame-eurasia/...	Endgame:Eurasia	NaN	https://is4-ssl.mzstatic.com/image/thumb/Purpl...	"This interactive experience is an exploration...
2	https://apps.apple.com/us/app/free-solitaire/i...	Free Solitaire+	NaN	https://is5-ssl.mzstatic.com/image/thumb/Purpl...	Same Solitaire game with classic Solitaire run...
3	https://apps.apple.com/us/app/draft-trainer/id...	Draft Trainer	NaN	https://is1-ssl.mzstatic.com/image/thumb/Purpl...	** Discounted for a limited time **\n\nEver wo...
4	https://apps.apple.com/us/app/rogue-knight-inf...	Rogue Knight: Infested Lands	Tactical roguelike w/ stealth	https://is2-ssl.mzstatic.com/image/thumb/Purpl...	Fight or sneak your way through hordes of mons...

We convert the categorical columns, namely, " **URL** ", " **Name** ", " **Subtitle** ", " **Icon URL** ", and " **Description** " from strings to numerical values and this by using the Label Encoder function from the Sklearn Package.

After

	URL	Name	Subtitle	Icon URL	Description
0	2336	2379	451	2526	2131
1	1710	1841	1394	3359	1581
2	2010	2123	1394	4863	4082
3	1582	1712	1394	491	1941
4	3839	3882	1154	1604	3053
...	...	...	...	...	...
5209	3568	3618	316	1288	378
5210	2610	85	14	348	935
5211	4713	4716	1394	433	1636
5212	2549	2652	1394	4252	584
5213	3401	3468	1394	768	4912

DATA CLEANING

Before

	Developer	Languages	Primary Genre	Genres
0	GAMEVIL Inc.	EN, FR, DE, JA, KO, ZH, ES, TH, ZH, VI	Games	Games, Role Playing, Strategy
1	Auroch Digital Ltd	EN	Games	Games, Simulation, Strategy, News
2	Chen Zhong Yuan	EN, ZH	Games	Games, Strategy, Entertainment, Card
3	GG Wizards, LLC	EN	Games	Games, Utilities, Card, Strategy
4	Luis Regueira	EN	Games	Games, Role Playing, Strategy
...	...	...	...	...
5209	Ndemic Creations	EN, FR, DE, IT, JA, KO, NB, PL, PT, RU, ZH, ES...	Games	Games, Strategy, Simulation
5210	AFEEL, Inc.	EN, JA, KO	Games	Games, Simulation, Strategy, Entertainment
5211	Stasis Software LLC	EN	Utilities	Utilities, Games, Board, Strategy
5212	ZEN Studios Ltd.	EN	Games	Games, Strategy, Role Playing
5213	Supervillain Studios	EN	Games	Games, Card, Entertainment, Strategy

Convert Categorical Data to numeric based on Label (Average User Rating)
" **Developer** ", " **Languages** ", " **Primary Genre** " and “**Genres** ",

We Give weights to categorial strings based on label column.

DATA CLEANING

```
dataset['Developer'] = dataset.groupby('Developer')['Average User Rating'].transform(lambda x: x.mean())
dataset['Languages'] = dataset.groupby('Languages')['Average User Rating'].transform(lambda x: x.mean())
dataset['Primary Genre'] = dataset.groupby('Primary Genre')['Average User Rating'].transform(lambda x: x.mean())
dataset['Genres'] = dataset.groupby('Genres')['Average User Rating'].transform(lambda x: x.mean())
```

After

	Developer	Languages	Primary Genre	Genres
0	4.250	4.000000	4.038315	4.116667
1	3.250	4.014370	4.038315	4.000000
2	4.000	4.056122	4.038315	4.096774
3	3.500	4.014370	4.038315	3.250000
4	4.500	4.014370	4.038315	4.116667
...	...	...	...	...
5209	4.500	4.500000	4.038315	3.947802
5210	4.125	4.195652	4.038315	3.853448
5211	5.000	4.014370	4.016129	5.000000
5212	4.250	4.014370	4.038315	4.182203
5213	3.500	4.014370	4.038315	3.865385

DATA CLEANING

Before

Original Release Date	Current Version Release Date
2018-03-06	2019-07-31
2013-03-21	2017-06-28
2013-04-04	2015-04-21
2011-05-26	2019-07-23
2017-05-19	2019-02-06
...	...
2012-05-26	2019-02-08
2015-01-11	2018-04-16
2012-08-16	2017-02-21
2016-06-08	2017-01-30
2012-09-10	2015-03-14

After

Days Since Release
512
1560
747
2980
628
...
2449
1191
1650
236
915

We calculate the duration between the **"Original Release Date"** and the **"Current Version Release Date"**, and this duration is added in a new Column called **"Days Since Release"**, and the **"Original Release Date", "Current Version Release Date"** columns are removed.

Feature Selection

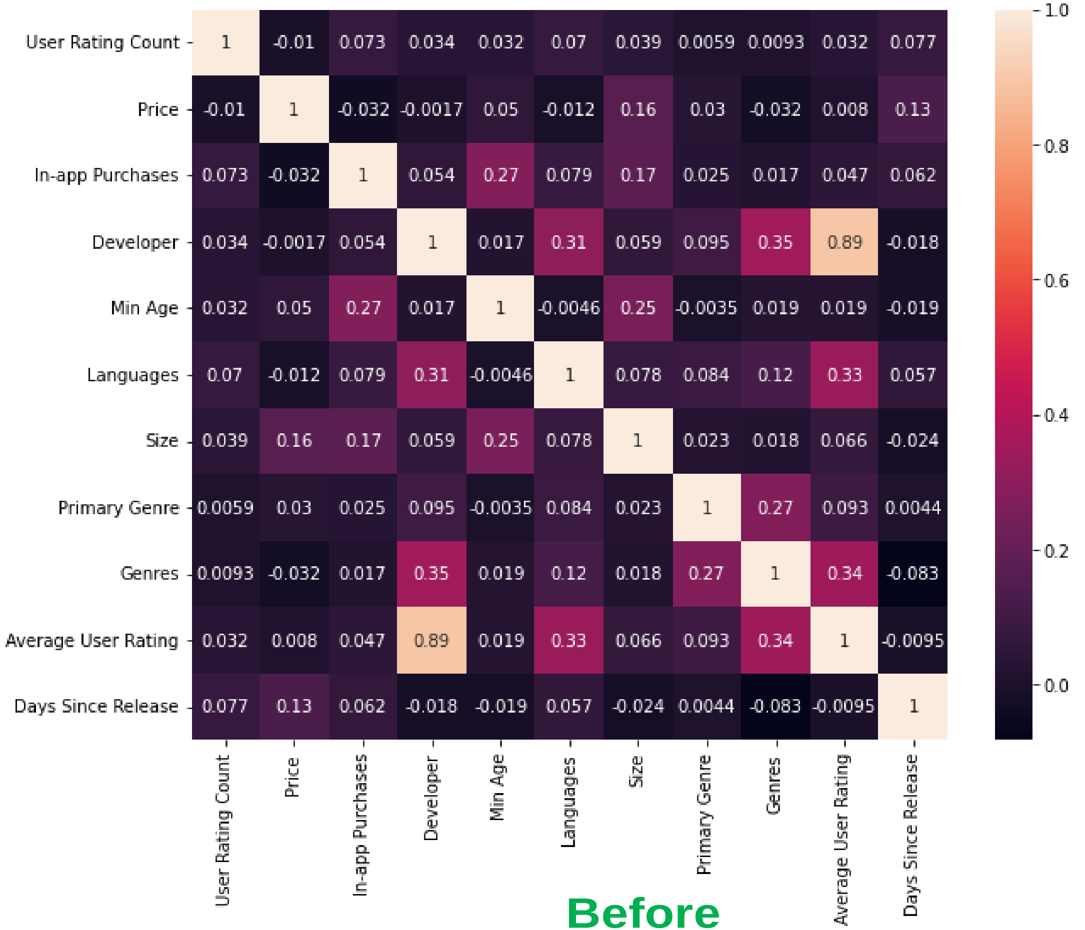
```python
dataset[['URL','ID','Subtitle','Icon URL','Name','Description']].nunique()
```

```
URL            5171
ID             5171
Subtitle       1400
Icon URL       5171
Name           5171
Description    5099
```

```python
#We deleted all theses columns because they are Unique (Non-Meaningful)
#Subtitle Column almost all cells are Nan values
#So we will drop this Columns

Columns_To_Deleted = ['URL','ID','Subtitle','Icon URL','Name','Description']
dataset = dataset.drop(columns = Columns_To_Deleted)
```

Feature Selection

Correlation Figure

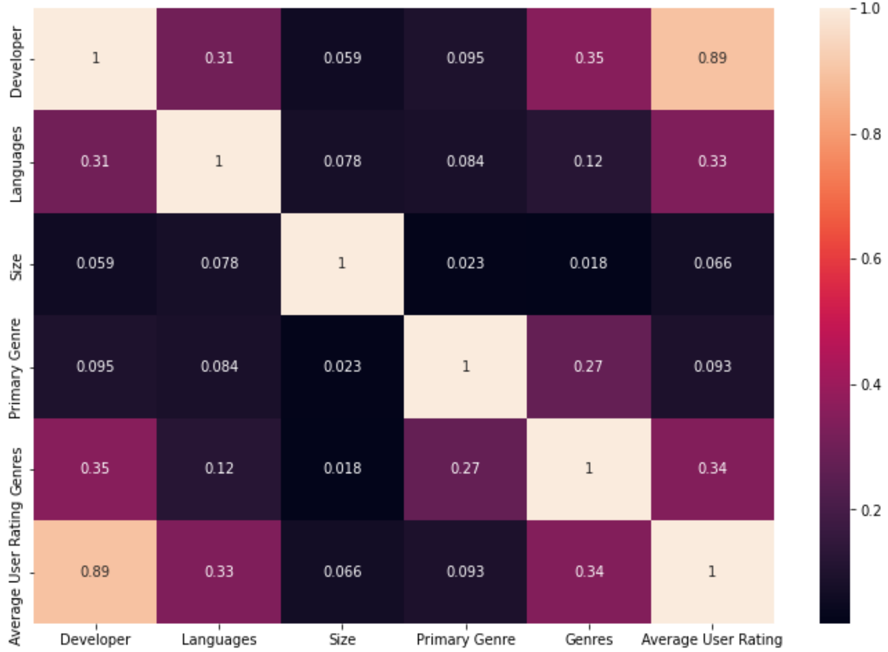


Before

Feature Selection

```
Game_corr = dataset.corr()
# Top 5% Correlation training features with the Value
Common_features = Game_corr.index[abs(Game_corr['Average User Rating']) > 0.05]
# Correlation plot
plt.subplots(figsize=(12, 8))
top_corr = dataset[Common_features].corr()
sns.heatmap(top_corr, annot=True)
plt.show()
Top = Common_features.delete(-1)
data_input = data_input[Top]
```

After



Feature Scaling

Feature Scaling using (StandardScaler)

We Scale all Features between range [-1, 1]

```
# Initialize the scaler
scaler = StandardScaler()

# Fit the scaler to the training data
scaler.fit(X_train)

# Scale the training, validation, and test data
X_train_scaled = scaler.transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

# Convert the scaled  data to a dataframe
X_train = pd.DataFrame(data=X_train_scaled, columns=X_train.columns)
X_val = pd.DataFrame(data=X_val_scaled, columns=X_val.columns)
X_test = pd.DataFrame(data=X_test_scaled, columns=X_test.columns)
```

After

Developer	Languages	Size	Primary Genre	Genres
-0.049922	-0.157089	-0.455459	0.058217	0.769608
0.689308	-0.100639	-0.046105	0.058217	-0.075257
0.935718	1.610713	-0.221579	0.058217	0.467521
0.319693	-0.100639	-0.306659	0.058217	-0.169423
1.294132	2.929549	-0.468293	0.058217	-0.169423
...	...	...	...	...
0.393616	-0.100639	-0.314261	0.058217	0.687418
1.428538	-0.100639	-0.373891	0.058217	1.163609
0.689308	1.807135	-0.108935	0.058217	-0.446282
1.058923	-0.100639	-0.503625	0.058217	-1.170334
0.689308	2.003558	-0.465055	0.058217	1.409287

Before

Developer	Languages	Size	Primary Genre	Genres
4.000000	4.000000	19414016	4.038315	4.223776
4.500000	4.014370	131017728	4.038315	4.017442
4.666667	4.450000	83177472	4.038315	4.150000
4.250000	4.014370	59981824	4.038315	3.994444
4.909091	4.785714	15915008	4.038315	3.994444
...	...	...	...	...
4.300000	4.014370	57909248	4.038315	4.203704
5.000000	4.014370	41652224	4.038315	4.320000
4.500000	4.500000	113888256	4.038315	3.926829
4.750000	4.014370	6282240	4.038315	3.750000
4.500000	4.550000	16797696	4.038315	4.380000

Regression

1-Linear Regression Model

Linear regression is a statistical method that is used to model the relationship between a dependent variable (also called the response variable or outcome variable) and one or more independent variables (also called predictor variables or explanatory variables). The goal of linear regression is to find the best linear relationship between the variables.

In simple linear regression, there is only one independent variable and one dependent variable, and the relationship between them is modeled as a straight line. The equation for a simple linear regression model is: y(pred) = $theta_0$ + $theta_1$*x + c
where y is the dependent variable, x is the independent variable, $theta_0$ is the y-intercept (the value of y when x is zero), $theta_1$ is the slope (the change in y for a one-unit increase in x), and c is the error term (which represents the part of the variation in y that cannot be explained by the model).

Features which used in ALL Regression Models are (Developer, Languages, Size, Primary Genre, Genres)

The following picture show the MSE and R2_Score for Test sets Using linear Regression.

```
Linear Regression:
Mean Square Error on test set: 0.0996
R2 Score on test set: 0.8100
```

2-Random Forest Model

Random Forest is a machine learning algorithm that is used for classification, regression, and other tasks that involve predicting a target variable based on one or more input variables. It is an ensemble learning method that combines multiple decision trees to produce a more accurate and robust model.

Random Forest is based on the idea of bagging, which stands for Bootstrap Aggregating. In bagging, multiple models are trained on different bootstrap samples of the training data, and the predictions from these models are aggregated to produce a final prediction. This helps to reduce the variance of the model and improve its generalization performance.

Random Forest builds on the idea of bagging by adding randomness to the model building process. Instead of using all the features to split each node of the decision tree, it selects a random subset of the features for each split. This helps to decorrelate the trees and reduce overfitting. In addition, Random Forest also uses random sampling of the training data to create each bootstrap sample, further increasing the diversity of the trees.

The following picture show the MSE and R2_Score for Test sets Using Random Forest

```
Random Forest Regression:
Mean Square Error on test set: 0.1024
R2 Score on test set: 0.8046
```

Regression

3-Polynomial Regression Model

Polynomial regression is a type of regression analysis where the relationship between the independent variable (x) and dependent variable (y) is modeled as an nth degree polynomial. It is used when the linear relationship between the variables does not fit the data well and a more complex relationship is required.

To perform polynomial regression, we first select the degree of the polynomial that best fits the data. We can then use the least squares method to estimate the coefficients of the polynomial equation that best fits the data. The coefficients are estimated by minimizing the sum of the squared errors between the predicted values of y and the actual values of y.

The equation for a polynomial regression model is: $y = b0 + b1x + b2x^2 + b3x^3 + \dots + bnx^n + \varepsilon$

where y is the dependent variable, x is the independent variable, b0, b1, b2, b3, ..., bn are the coefficients, n is the degree of the polynomial, and ε is the error term.

The following picture show the MSE and R2_score for Test sets Using Polynomial Regression

```
Polynomial Regression:
MSE on test set: 0.0996
R2 Score on test set: 0.8099
```

Classification

1- Logistic Regression

```python
log_C_values = [0.1, 1, 10]
solvers = ['sag', 'liblinear', 'lbfgs']
logreg_models = {}
print("-------------------Logistic Regression-------------------")

for sol in solvers:
    for C in log_C_values:
        start = timeit.default_timer()
        logreg = LogisticRegression(solver=sol, C=C)
        logreg.fit(X_train, y_train)
        end = timeit.default_timer()
        train_Time.append(end-start)
        y_pred = logreg.predict(X_test)

        logaccuracy = accuracy_score(y_test, y_pred)
        train_accurices.append(logaccuracy)
        logmodel_name = f'{sol}_{C}'
        logreg_models[logmodel_name] = (logreg, logaccuracy)

        print(f"Solver: {sol}, C: {C}")
        print("----------------------------")
        print("Accuracy score:", logaccuracy)
```

```
-------------------Logistic Regression-------------------
Solver: sag, C: 0.1
----------------------------
Accuracy score: 0.8830917874396135
Solver: sag, C: 1
----------------------------
Accuracy score: 0.8917874396135266
Solver: sag, C: 10
----------------------------
Accuracy score: 0.893719806763285
Solver: liblinear, C: 0.1
----------------------------
Accuracy score: 0.8338164251207729
Solver: liblinear, C: 1
----------------------------
Accuracy score: 0.8396135265700483
Solver: liblinear, C: 10
----------------------------
Accuracy score: 0.8917874396135266
Solver: lbfgs, C: 0.1
----------------------------
Accuracy score: 0.8830917874396135
Solver: lbfgs, C: 1
----------------------------
Accuracy score: 0.8917874396135266
Solver: lbfgs, C: 10
----------------------------
Accuracy score: 0.893719806763285
```

Classification

3-SVM

```python
kernel_values = ['linear', 'rbf', 'sigmoid']
SVM_C_values = [0.1, 1, 10]
svm_models = {}

print("------------------SVM------------------")

for kernel in kernel_values:
    for C in SVM_C_values:
        start = timeit.default_timer()
        svm = SVC(kernel=kernel, C=C)
        svm.fit(X_train, y_train)
        end = timeit.default_timer()
        train_Time.append(end-start)
        y_pred = svm.predict(X_test)

        svmaccuracy = accuracy_score(y_test, y_pred)
        train_accurices.append(svmaccuracy)

        svmmodel_name = f'{kernel}_{C}'
        svm_models[svmmodel_name] = (svm, svmaccuracy)

        print(f"Kernel: {kernel}, C: {C}")
        print("----------------------------")

        print("Accuracy score:", svmaccuracy)
```

```
------------------SVM------------------
Kernel: linear, C: 0.1
----------------------------
Accuracy score: 0.8888888888888888
Kernel: linear, C: 1
----------------------------
Accuracy score: 0.8966183574879227
Kernel: linear, C: 10
----------------------------
Accuracy score: 0.8966183574879227
Kernel: rbf, C: 0.1
----------------------------
Accuracy score: 0.8946859903381642
Kernel: rbf, C: 1
----------------------------
Accuracy score: 0.8966183574879227
Kernel: rbf, C: 10
----------------------------
Accuracy score: 0.8917874396135266
Kernel: sigmoid, C: 0.1
----------------------------
Accuracy score: 0.49468599033816424
Kernel: sigmoid, C: 1
----------------------------
Accuracy score: 0.41449275362318834
Kernel: sigmoid, C: 10
----------------------------
Accuracy score: 0.08888888888888889
```

Classification

3-Naive Bayes

```python
var_smoothing_values = [1e-9, 1e-7, 1e-5]
priors = [None, [0.3, 0.3, 0.4], [0.3, 0.3, 0.4]]
NB_models = {}

print("------------------Naive Bayes------------------")

for prior in priors:
    for var_smoothing in var_smoothing_values:
        nb = GaussianNB(var_smoothing=var_smoothing, priors=prior)

        # Fit the model on the training data
        nb.fit(X_train, y_train)

        # Predict on the test data
        y_pred = nb.predict(X_test)

        # Calculate accuracy score and mean squared error
        NBaccuracy = accuracy_score(y_test, y_pred)

        NBmodel_name = f'{prior}_{var_smoothing}'
        NB_models[NBmodel_name] = (nb, NBaccuracy)

        print(f"Priors: {prior}, Var smoothing: {var_smoothing}")
        print("---------------------------")
        print("Accuracy score:", NBaccuracy)
```

```
------------------Naive Bayes------------------
Priors: None, Var smoothing: 1e-09
---------------------------
Accuracy score: 0.8714975845410629
Priors: None, Var smoothing: 1e-07
---------------------------
Accuracy score: 0.8714975845410629
Priors: None, Var smoothing: 1e-05
---------------------------
Accuracy score: 0.8714975845410629
Priors: [0.3, 0.3, 0.4], Var smoothing: 1e-09
---------------------------
Accuracy score: 0.8705314009661835
Priors: [0.3, 0.3, 0.4], Var smoothing: 1e-07
---------------------------
Accuracy score: 0.8705314009661835
Priors: [0.3, 0.3, 0.4], Var smoothing: 1e-05
---------------------------
Accuracy score: 0.8705314009661835
Priors: [0.3, 0.3, 0.4], Var smoothing: 1e-09
---------------------------
Accuracy score: 0.8705314009661835
Priors: [0.3, 0.3, 0.4], Var smoothing: 1e-07
---------------------------
Accuracy score: 0.8705314009661835
Priors: [0.3, 0.3, 0.4], Var smoothing: 1e-05
---------------------------
Accuracy score: 0.8705314009661835
```

Classification

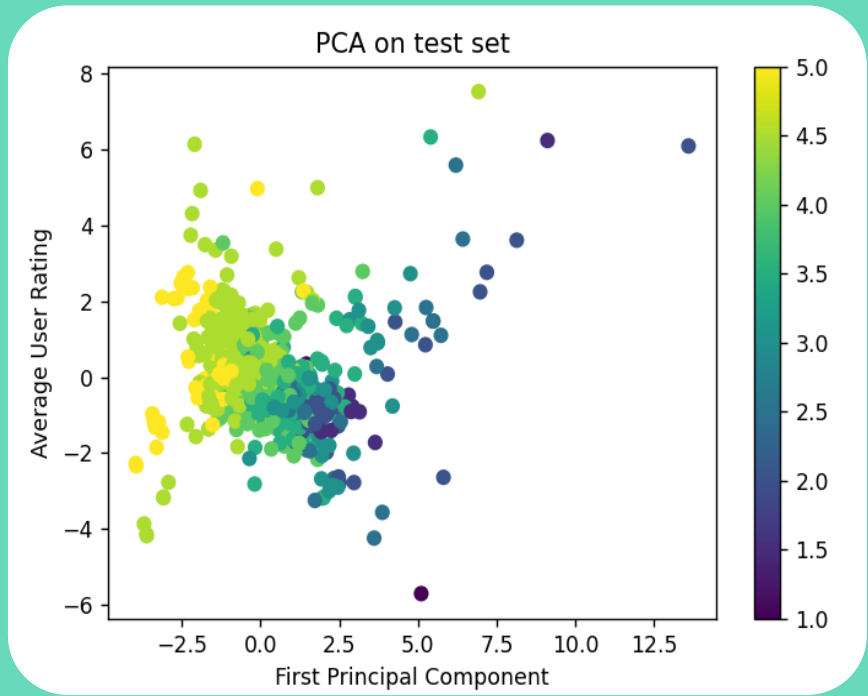
4- Decision Tree

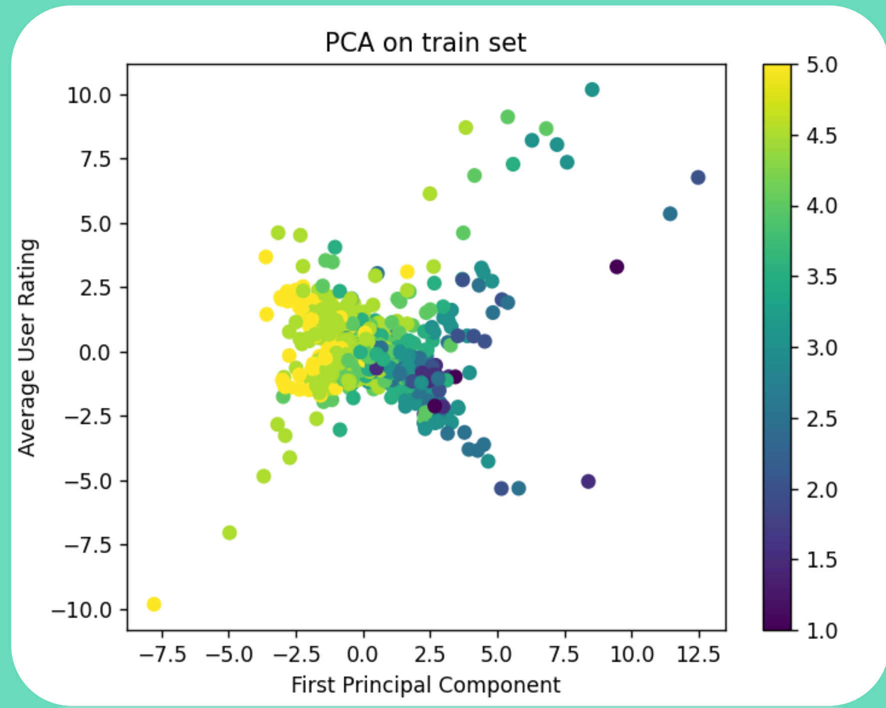
```python
################## Decision Tree #####################
max_depth_values = [2, 5, 10]
min_samples_split_values = [2, 5, 10]
criterion_values = ['gini', 'entropy']
dt_models = {}

print("------------------Decision Tree------------------")

for max_depth in max_depth_values:
    for min_samples_split in min_samples_split_values:
        for criterion in criterion_values:
            dt = DecisionTreeClassifier(max_depth=max_depth, min_samples_split=min_samples_split, criterion=criterion)

            dt.fit(X_train, y_train)
            y_pred = dt.predict(X_test)

            dt_accuracy = accuracy_score(y_test, y_pred)
            dt_model_name = f'{max_depth}_{min_samples_split}_{criterion}'
            dt_models[dt_model_name] = (dt, dt_accuracy)

            print(f"Max depth: {max_depth}, Min samples split: {min_samples_split}, Criterion: {criterion}")
            print("---------------------------")
            print("Accuracy score:", dt_accuracy)
```

```
------------------Decision Tree------------------
Max depth: 2, Min samples split: 2, Criterion: gini
---------------------------
Accuracy score: 0.8792270531400966
Max depth: 2, Min samples split: 2, Criterion: entropy
---------------------------
Accuracy score: 0.8743961352657005
Max depth: 2, Min samples split: 5, Criterion: gini
---------------------------
Accuracy score: 0.8792270531400966
Max depth: 2, Min samples split: 5, Criterion: entropy
---------------------------
Accuracy score: 0.8743961352657005
Max depth: 2, Min samples split: 10, Criterion: gini
---------------------------
Accuracy score: 0.8792270531400966
Max depth: 2, Min samples split: 10, Criterion: entropy
---------------------------
Accuracy score: 0.8743961352657005
Max depth: 5, Min samples split: 2, Criterion: gini
---------------------------
Accuracy score: 0.8792270531400966
Max depth: 5, Min samples split: 2, Criterion: entropy
---------------------------
Accuracy score: 0.8724637681159421
Max depth: 5, Min samples split: 5, Criterion: gini
---------------------------
Accuracy score: 0.8792270531400966
Max depth: 5, Min samples split: 5, Criterion: entropy
---------------------------
Accuracy score: 0.8724637681159421
Max depth: 5, Min samples split: 10, Criterion: gini
---------------------------
Accuracy score: 0.8792270531400966
Max depth: 5, Min samples split: 10, Criterion: entropy
```

Visualization

The Scatter For (Test set, Train set)





- Split The data into (0.20) For Test sets (0.20) For Validate sets and (0.60) For Training set.
- Apply (PCA) Technique on both (Test set, Train set) on features (Developer, Languages, Size, Primary genre, Genres)

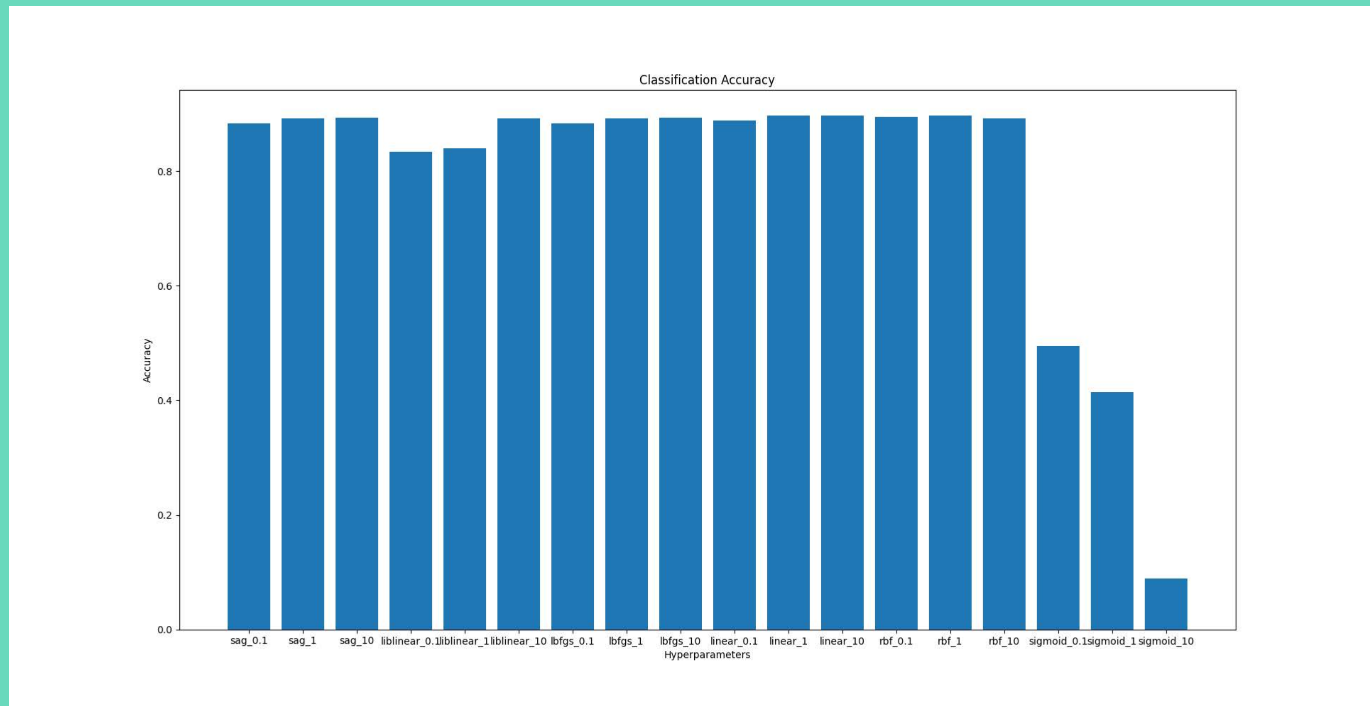
Which is used for analyzing datasets containing a high number of dimensions (Features) per observation,

Increasing the interpretability of data while preserving the maximum amount of information, and enabling

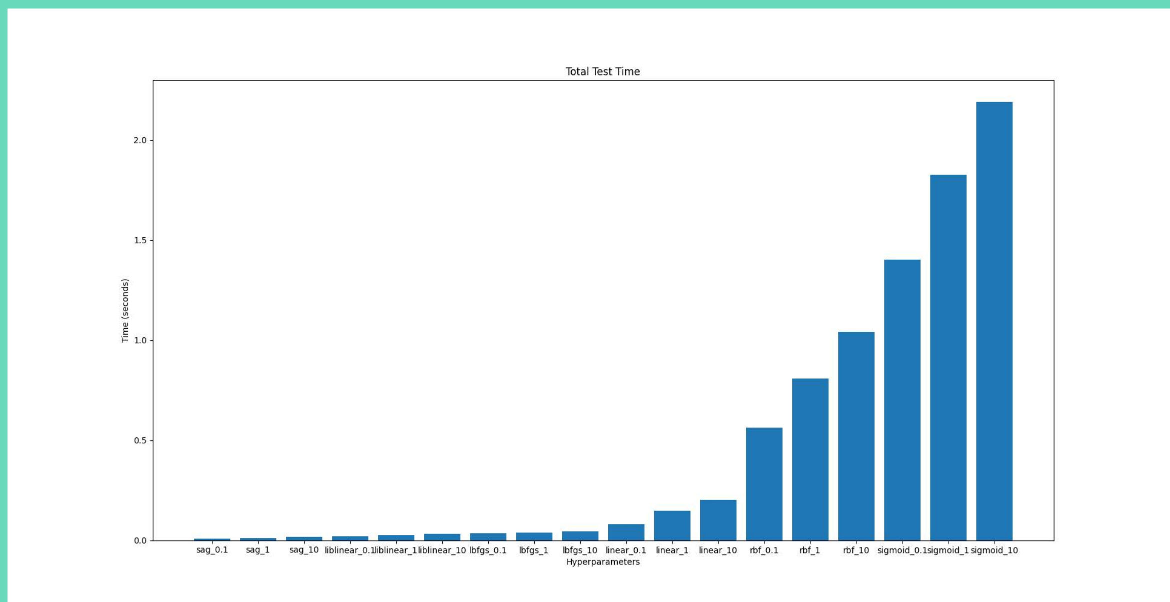
The visualization of multidimensional data.

Visualization

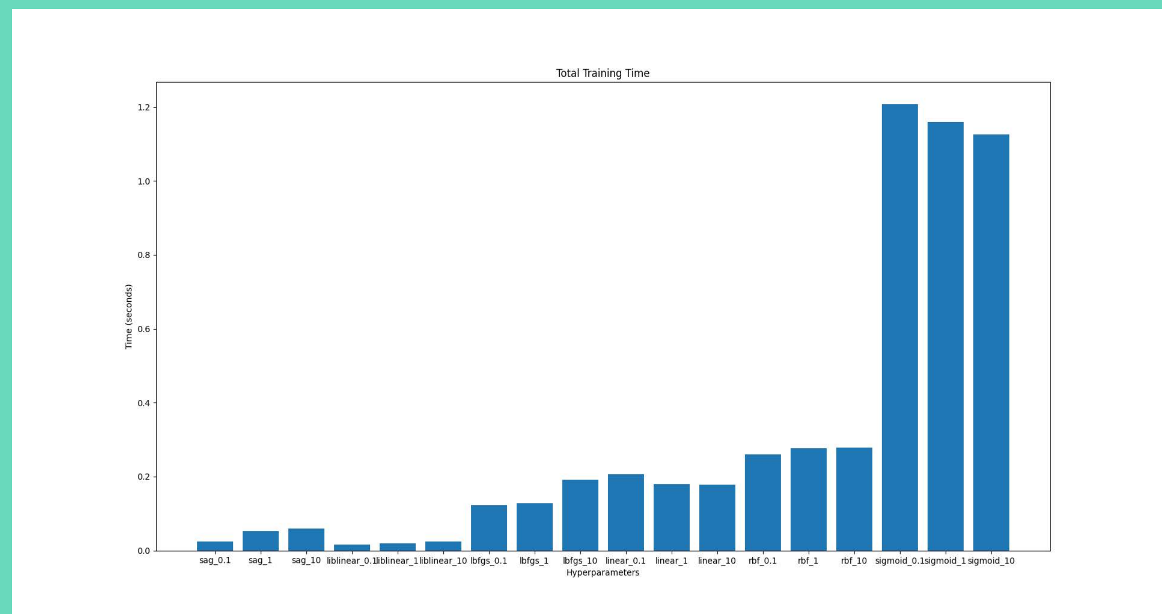
Classification Accuracy



Total Test Time



Total Training Time



Conclusion

- Delete all duplicate rows
- Check missing Values, then replace them with mean or mode or zero
- remove the positive sign (+) next to each age in (Age column) and change its name to (Min Age)
- convert the categorical columns namely (URL, Name, Sub title, Icon URL, Description) from Strings to numerical using the label encoder based on Label (Average User Rating)
- calculate the duration between the “The original Release Date” and the current “Current Version Release Date”, and this duration is added in a new Column called” Days Since Release” and the “Original Release Date “and remove each Columns (Original Release Date, Current Version Release Date)
- Delete ALL Unique columns (URL, ID, Sub title, Icon URL, Name, Description)
- Apply Feature Scaling in All features between Range [-1,1]
- Apply Three Regression Models (Linear Regression, Random Forest Regression, Polynomial Regression)
- Visualizing The Scatter for (Test set, Train set) Using PCA technique

Conclusion

Our intuition was that Polynomial linear regression would be better, but it was disproved by the error of the output.

Multivariable linear regression has proven to be better than Polynomial linear regression according to the features that we have selected.

GAME
OVER