# Game Application Success Prediction project.

# Team ID: 25

# Team Information

Member 1: Khaled Galal Yehia

ID 1: 20191700223

Member 2: Nahed Mohamed Mostafa

ID 2: 20191700916

Member 3: Hagar Ahmed Mahmoud

ID 3: 20191700723

Member 4: Mohamed Essam Ali

ID 4: 20201700711

Member 5: Hazem Abd El Hamid Malek

ID 5: 20191700020

Member 6: Omar Ahmed Farouk

ID 6: 20191700400

# Classification Accuracy

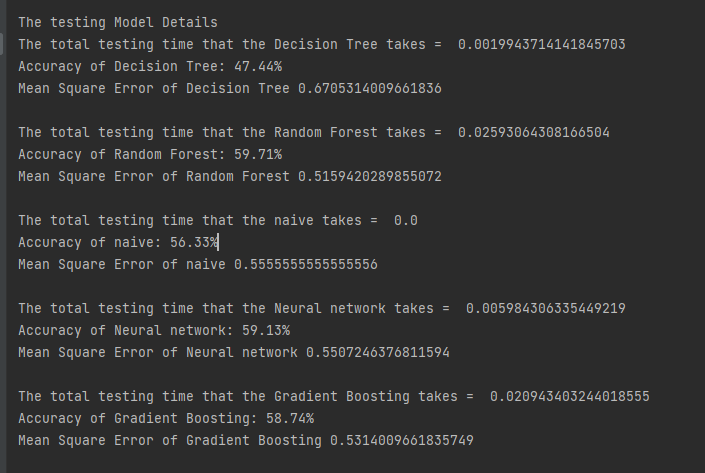
The models include Decision Tree, Random Forest, Naive Bayes, Neural Network, and Gradient Boosting.

The Decision Tree model has the lowest accuracy of 47.44%, This low accuracy may be due to overfitting or underfitting of the model.

The Random Forest model has an accuracy of 59.71%, which is the highest accuracy. Random Forest is an ensemble of decision trees, which can reduce overfitting and improve accuracy by combining the predictions of multiple trees.

The Naive Bayes model has an accuracy of 56.33%, Naive Bayes is a simple probabilistic model that assumes independence between features.

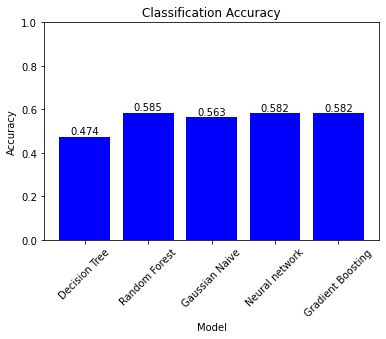
The Neural Network model has an accuracy of 59.13%, which is close to the Random Forest model. Neural Networks are a type of machine learning model that can learn complex relationships between features.

The Gradient Boosting model has an accuracy of 58.74%, Gradient Boosting is another ensemble method that combines weak learners (decision trees) and can improve accuracy by iteratively adjusting weights for misclassified samples.

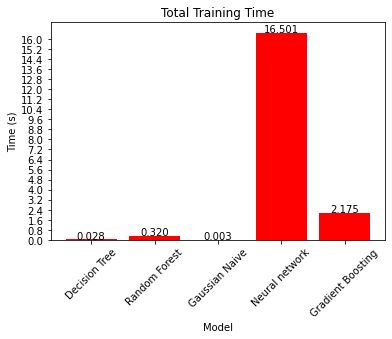
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| classifier | Decision tree | Random forest | Naïve bayes | Neural network | Gradient boosting |
| accuracy | 47.4% | 59.7% | 56.3% | 59.1% | 58.7% |

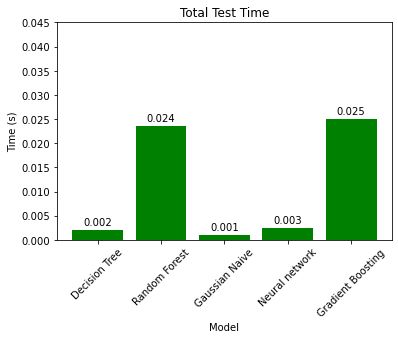
# Visualization

* Classification accuracy



* Training time



* Training time

# Feature selection

- drop column if all its values in column is 1

- the difference between feature selection in regression and classification that:-

Y is categorical data in classification so

1-apply **Anova** test on all numerical x columns ,categorical y column

if p value of it > 0.01 🡺 drop this column

2-Performing **Chi-sq test** on all categorical x columns, categorical y column

If ChiSqResult > 0.01🡺this means this column has no effect on result so drop this column

And non drop columns used to apply test on the same selected features in train

# Hyperparameters

* **Decision tree**
* We used the hyperparameters: criterion, max\_depth, min\_samples\_split, min\_samples\_leaf, max\_features, min\_impurity\_decrease, random\_state.
* We used 3 choices of max\_depth and max\_features.
* Each choices the other parameters were fixed.
* max\_depth: This hyperparameter sets the maximum depth of the decision tree. A higher value allows the tree to grow deeper and capture more complex relationships in the data. However, setting it to None means there is no maximum depth, and the tree will continue to grow until all leaves are pure or until all leaves contain the minimum number of samples required. And that was the highest accuracy possible when changing the hyperparameter.
* max\_features: This hyperparameter determines the maximum number of features to consider when looking for the best split. Setting it to None means that all features will be considered. And that was the highest accuracy possible when changing the hyperparameter.

# **Gradient Boosting:**

## n\_estimator:

This parameter controls the number of boosting stages or weak learners used in the model. Increasing the number of estimators can improve the model's accuracy, but it also increases the computational complexity and training time. The value of 150 in the code indicates that 150 weak learners will be sequentially added to the ensemble. We tried multiple values but found 150 gives the highest accuracy.

## learning \_rate:

It determines the contribution of each weak learner to the final ensemble. A lower learning rate requires more weak learners to achieve the same overall effect, but it can lead to better generalization. A learning rate of 0.01 in the code indicates a small contribution of each weak learner. We tried multiple values but found that 0.01 gives the highest accuracy.

## subsample:

It controls the fraction of data used for training each model. Using a smaller fraction introduces randomness, which can prevent overfitting and improve accuracy.

## max\_depth:

This limits the depth or complexity of each individual model. A deeper model can capture more complex patterns but may overfit. Finding the right balance is crucial for accuracy.

## min\_samples\_split:

This sets the minimum number of samples required to split a node in a model. Increasing this value can make the model more general and less prone to overfitting.

## loss:

It specifies the loss function optimized during training. For binary classification, the 'deviance' loss is commonly used.

## warm\_start:

This determines whether to train the model from scratch or continue training from the previous state. Starting from scratch may lead to better accuracy.

## validation\_fraction:

It determines the portion of the training data used for validation. Using a validation set helps monitor the model's performance and prevent overfitting.

# Decision Tree:

## max\_depth:

This hyperparameter sets the maximum depth of the decision tree. A higher value allows the tree to grow deeper and capture more complex relationships in the data. However, setting it to None means there is no maximum depth, and the tree will continue to grow until all leaves are pure or until all leaves contain the minimum number of samples required. And that was the highest accuracy possible when changing the hyperparameter.

## max\_features:

This hyperparameter determines the maximum number of features to consider when looking for the best split. Setting it to None means that all features will be considered. And that was the highest accuracy possible when changing the hyperparameter.

## min\_impurity\_decrease:

This hyperparameter sets the minimum decrease in impurity required for a split to be considered. If a split results in an impurity decrease less than this value, the split will be canceled.

## random\_state:

This hyperparameter sets the random seed to ensure reproducibility of the results. It determines the random number generation for randomness in the decision tree algorithm.

## criterion:

This hyperparameter determines the measure used to evaluate the quality of a split in the decision tree. In this case, "entropy" is chosen as the criterion, which indicates that the decision tree will be built based on the information gain using entropy.

## min\_samples\_split:

This hyperparameter specifies the minimum number of samples required to split an internal node. If the number of samples at a node is below this threshold, the node will not be split further. Setting it to 2 means that a split will be considered even if there are only two samples at a node.

## min\_samples\_leaf:

This hyperparameter sets the minimum number of samples required to be at a leaf node. If a split results in a leaf node with fewer samples than this threshold, the split will be canceled. A value of 1 means that each leaf node must contain at least one sample.

# Random Forest:

## n\_estimators:

This hyperparameter specifies the number of decision trees in the random forest. Increasing the number of estimators generally leads to better performance, up to a certain point. However, it also increases computational complexity. We found that 100 is the best number to set the hyperparameter.

## max\_depth:

This hyperparameter determines the maximum depth of each decision tree in the random forest. A deeper tree can learn more complex patterns in the data, but it can also lead to overfitting. Setting an appropriate value helps control the trade-off between model complexity and generalization.

## min\_samples\_split:

This hyperparameter specifies the minimum number of samples required to split an internal node during the construction of each decision tree. Increasing this value can prevent the tree from splitting too early and result in more robust models.

## min\_samples\_leaf:

This hyperparameter determines the minimum number of samples required to be at a leaf node. Setting a higher value can prevent overfitting by ensuring that each leaf contains a minimum amount of information.

## max\_features:

This hyperparameter controls the number of features to consider when looking for the best split at each node. 'sqrt' indicates that the square root of the total number of features will be used. Limiting the number of features considered can reduce the potential for overfitting.

## random\_state:

This hyperparameter sets the seed for the random number generator used by the random forest. It ensures that the model produces reproducible results.

# Neural Network:

## hidden\_layer\_sizes:

The MLP has two hidden layers with 100 neurons in the first hidden layer and 10 neurons in the second hidden layer. The number of neurons in the hidden layers affects the model's capacity to learn complex patterns and representations from the data. Increasing the number of neurons can potentially allow the model to capture more intricate relationships in the data, but it also increases the computational complexity of the model. We tested more than 1 combination but this one gives us the highest accuracy.

## learning\_rate:

The learning rate is set to "adaptive", which means it adjusts dynamically during training based on the performance of the model. The adaptive learning rate allows the model to make larger updates initially and gradually decrease the learning rate as it gets closer to the optimal solution. This adaptive learning rate can help the model converge faster and improve accuracy.

## batch\_size:

The batch size is set to 32, which means that the model updates its weights after processing 32 samples at a time during training. The choice of batch size affects the speed and stability of the training process. Smaller batch sizes introduce more noise into the weight updates but allow for more frequent updates, while larger batch sizes provide more stable updates but may slow down the training process.

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