Exploration of Predictive Machine Learning Models on Video Game Dataset

MLVU information sheet

29th March 2024

Group number
34

Authors

name	student number
Jonas Hubbers	2739388
Selma Kocabıyık	2759524
Céleste Mordenti	2773931
Jacquiline Rose Roney	2761539
Kaira Steeman	2777482

Software used

- Kaggle, an online source of data science in the form of models, datasets, and in progress projects.
- Vscode, an interactive code editor for most digital languages.
- Python, a programming language with various applications.
- Matplotlib, a comprehensive python library for modeling and data visualization in python.
- Pandas, a data analysis and manipulation tool in python.
- Seaborn, a library for python that uses matplotlib to plot graphs.
- Sci-kit learn, a machine learning library for python.

Use of AI tools AI was used to clarify some concepts that were more difficult to understand such as the exact functioning of algorithms implemented in the program.

1 Abstract

This project aims to take a comprehensive dataset on video games and its logistical features and run it through different machine learning algorithms, namely k-nearest neighbour, decision trees and logistic regression. The aim is to generate predictive models and compare the errors and biases between the algorithms to find the best fit for the data when predicting the platform the video game is found on. These predictions are based on other features of the game such as developer, genre and ratings.

2 Introduction

Video games have gained massive popularity in society over the last few decades. With technological and artistic advancement came demand for the various types of games known today and normalized in every imaginable form. Originally reserved for the rich with the means, games have become much more accessible and startlingly varied, their uses expanded to its own culture, careers, and professional sports community (Southern, 2017). Today, video games are accessible to almost everyone by simply downloading the software on a mobile phone or personal computer.

While more common as a pastime, the professional video games industry have made significant steps and professional gaming events have gained popularity with its versatility and lucrative opportunities (Boyle, 2019). For this reason, the dataset that was chosen for this machine learning project is about video games, for reference see: Kaggle Dataset. The analysis of this project will dig deeper into the topic by manipulating the dataset in different ways with the help of various machine learning algorithms.

2.1 Machine Learning

With the fact that both are backed heavily by technological advancements, it comes as no surprise that the popularity of video games also meant that machine learning in the field of artificial intelligence also got more widespread(Cohen, 2021). With these innovations, enormous amounts of data can be used to train predictive algorithms and programs. For this purpose, three different machine learning algorithms are implemented in relation to the previously mentioned video games dataset in order to predict a target class. More specifically: k-nearest neighbour, decision tree, and logistic regression algorithms. From the video games dataset, 6 features are used to train the machine learning models. From this, the class 'platforms' is predicted, to which belong Android, Linux, Nintendo Switch, PC, SNES, iOS, and macOS.

2.2 Research Question

By implementing the three algorithms, k-nearest neighbour, decision tree, and logistic regression, with regard to the chosen dataset, the goal is to find the

best fitting model. More concretely, the following research question is asked: **To what extent can Logistic Regression, Decision tree and K-Nearest Neighbour models accurately classify the "platform" label of a video game based on several game features?**

3 Dataset and Preparation

The contribution of video games to the entertainment software industry and economy is significant, with direct and complementary effects totalling up to 16.4 billion in 2004, and growing today (Crandall and Sidak, 2006). In the video game industry, selling various games and reaching the right audience depends on different factors such as its developers, publishers, genres and platforms as the leading factors. In this project, game platforms are taken as the predicted label. The focus is on examining the extent to which players play games across various platforms, which is influenced by other features. The diversity of platforms can depend on factors such as genre, developer, publisher, website, gameplay time, and ratings. For instance, some individuals rate a game higher on the iOS platform compared to when they play it on Linux. Moreover, certain publishers and developers might produce games exclusively playable on PC. Taking all these factors into account, a dataset has been created with 14 features, containing 6 different instances under the target 'platform' label.

The dataset has been prepared to increase the performance of the models. The dataset initially contained over 100,000 rows and 27 columns, along with categorical and numerical data. In preparation for the data for models, all the empty rows were cleaned, and the spelling errors of the categorical values were removed and corrected into new columns. Additionally, categorical columns were organized by removing outliers. At this point, the leading classes in various columns are selected, such as developer and publisher, and the rest are removed.

Similarly, the classes in the platform column that were barely mentioned and could potentially lead to overfitting during the model training processes were removed. To address instance imbalances in the platform column of the dataset, oversampling and undersampling methods are used. To illustrate that, noticing that iOS was represented more than other classes, the count of iOS and appropriately increased the counts of other classes were reduced, thereby making the data more organized and balanced.

The entropy and information-gain methods were used to understand how suitable each column was for the predicted label (see figure 2). Entropy measures the disorder and uncertainty in a dataset; a high entropy value indicates a high diversity within the dataset. Conditional entropy refers to how disordered the label is based on the value of a feature. Therefore, a high information-gain value signifies that the feature (column) aligns well with the predicted label (Brownlee, 2020).

For the predicted label 'Platform', feature selection was conducted by choosing the appropriate columns using the information gain formula. Information gain was calculated iteratively for all columns using Python codes. At this

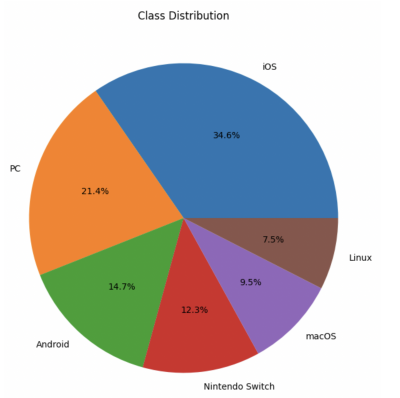

Figure 1: Class distribution for Platform Column

Entropy Formula: $H(Y) = -\sum_{i=1}^{n} P(y_i)\log_2 P(y_i)$

Conditional entropy formula: $H(Y|X) = -\sum_{x \in X} P(x) \sum_{y \in Y} P(y|x)\log_2 P(y|x)$

Information-Gain Formula: IG(Y/X) = H(Y) - H(Y/X)

Figure 2: Entropy Formulas

stage, by treating each column as X, it was attempted to determine how suitable the column is for predicting the platform (e.g. IG(Platform—Column) = H(Platform) - H(Platform—Column)). Ultimately, only 14 out of the 27 features and a target were retained (Appendix A).

3.1 Data Splitting

The main focus of the models is to have a high accuracy score and to ensure it generalizes well to unseen data. To increase the performance, different hyperparameters were selected. For the optimization of the hyperparameters, the grid search technique was used, which evaluates the combination of parameters through cross-validation. Additionally, in the process of cross-validation training-data is split into subsets to validate the performance of the model. Therefore, the dataset was split into training (80 %) to train the model on the data and select the parameters, and testing (20 %) to test the performance of the models. See appendix C for class distributions. In addition to that, for some models the optimisation of the hyperparameters is processed on validation set. In this case, data is split into training (60%), validation (20%) and test set(20%).

4 Methods

4.1 k-nearest Neighbour

The k-Nearest Neighbors algorithm is a powerful technique in supervised machine learning used for classification and regression tasks. In the case of this project, it is used for classification. The algorithm operates by finding the k nearest data points to a new input, based on a chosen distance metric, and then

making predictions based on those neighbours (Zhang and Zhou, 2007). During training, kNN memorizes the entire training dataset without constructing an explicit model. While kNN's can be easy to implement and understand, its prediction phase can be computationally intensive, especially for large datasets, since it requires calculating distances to all training points (Abu Alfeilat et al., 2019). Moreover, hyperparameters need to be set to optimize the performance of this algorithm, systematic trial and error was used by increasing the neighbors in increments of 3 until 21 neighbors. The metric is kept constant and the weight is only tested for the first 5 trials.

- **Neighbours**: This hyperparameter sets the value of k; the number of nearest neighbours that are taken into consideration. The choice of k is critical as it can possibly make the model overfit, or on the opposite, making it not accurate enough. In general, smaller k values tend to produce more flexible, noise-sensitive models, while larger k values yield smoother decision boundaries. The best fitting value for k turned out to be 15 in the case of the video games dataset.

- **Metric**: This hyperparameter is used to determine how similar the data point that needs to be classified is to its nearest neighbours. The metric that proved to be most suitable is the euclidean distance, which is given by the following formula:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

4.2 Decision Tree

The Decision Tree algorithm is an algorithm that consists of root nodes and leaf nodes and can be used for both classification and regression tasks (Navada et al., 2011). The root nodes represent features, and the leaf nodes represent the labels to be predicted, which in this case is 'Platform'. The decision tree model is based on the scikit-learn decision tree package which uses an optimized version of the CART algorithm. This algorithm uses the Gini coefficient to determine on which features the root should be split and further the corresponding leaves.

$$G = 1 - \sum_{i=1}^{N} (p_i)^2$$

The Gini-coefficient is measuring purity across the dataset, a Gini-coefficient of 0 means all instances are part of one class while a value of 1 means there is an equal distribution of instances between the classes. Furthermore, to make the model fit a dataset better it can be tuned using hyperparameters. For decision trees, the max depth, minimum sample split and the minimum sample leaf is what was aimed to be optimized through the grid search algorithm (See Appendix E).

- **Max depth**: This hyperparameter adjusts the depth of the search tree. The higher the depth of the model there are more complex solutions it can

find, however, this often leads to overfitting. Having a depth that is too low is also not beneficial as the model might not be able to fit the data well. The values tuned for the parameter will therefore be between 5 and 15.

- **Minimum sample leaf**: This hyperparameter determines how many instances are necessary for a leaf to split into sub-leaves again. Setting this value very low means that it always splits thus risking overfitting. Having it too high would mean the leaves barely split so the model can't learn as best as possible. The values for this hyperparameter will be between 2 and 10.
- **Minimum sample split**: Similar to minimum sample leaf, the sample split splits an internal node based on the value of the hyperparameters. It follows similar principles as the sample leaf, however, it is applied at a different time in the construction of the tree. The values for these parameters are set between 1 and 3.

4.3 Logistic Regression

Logistic regression is a discriminative classifier model used for binary classification tasks (Peng et al., 2002a). It creates a linear combination of the input features onto which the following logistic sigmoid function is applied; to generate probabilities between 0 and 1 (Zaidi, 2022).

$$f(x) = \frac{1}{1 + e^{-x}}$$

The probability generated can then be interpreted as the probability that an instance belongs to a corresponding pre-defined class (Peng et al., 2002b). In this report, the model will be used to predict the target label 'Platform' as previously mentioned. The model is partially implemented using the sklearn learn library whose Logistic regression's hyperparameters include:

- **Solver**: Logistic regression models require optimization to find the coefficients to the model that minimize the loss the most. Solver is the choice of optimization algorithm that is used to perform this task. Options include: liblinear, newton-cg, lbfgs etc.
- **Penalty**: Specifies the exact type of regularization penalty applied. Regularization is required to ensure that the model performs well without overfitting, which includes a penalty term which can be of various types such as: L1 Regularization, L2 Regularization and Elastic Net Regularization.

A pragmatic approach of 5-fold cross validation strategy followed by systematic trial and error approach with the optimal hyperparameters was performed on the validation set like in the K-nn model. The combination of hyperparameters with the highest accuracy without overfitting was chosen; Which are **penalty = '12' and solver = 'liblinear'.**

5 Results

5.1 Baseline

The baseline is an important factor when writing out the results as it allows comparison. The baseline that is used with regard to the video games dataset is to assign the majority class to the target label, which results in approximately 0.38 accuracy for classification. From this, clear statements can be made when analyzing outcomes for the three algorithms that were used.

5.2 Logistic Regression

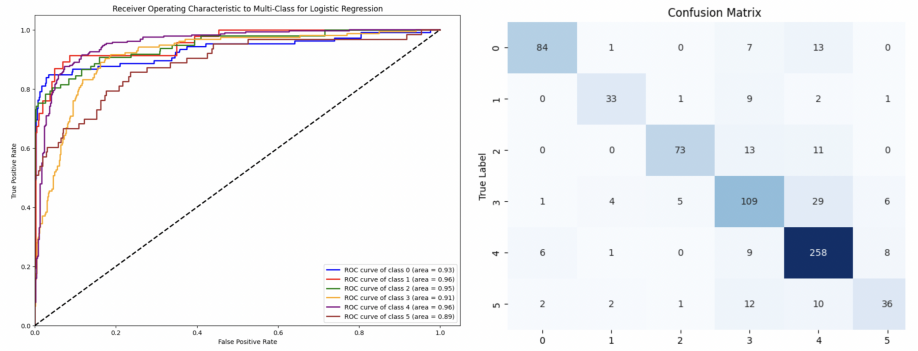

Figure 3: ROC Curve and Confusion Matrix for Logistic Regression

The confusion matrix, a vital tool for performance evaluation, depicts the relationship between predicted values (x axis) and true label (y axis). There are 5 distinct classes under consideration with the following values corresponding to class names: 0: Android, 1:Linux, 2: Nintendo, 3: PC, 4: iOS and 5: macOS. The values on the diagonal of the confusion matix in figure 3, namely 84, 33, 73, 109, 256 and 3 represent the true positives or the correctly classified values. Conversely, off- diagonal values represent misclassifications. The notable predominance of correct classification outweighs the misclassification indicating the model's high efficacy. The highest number of misclassifications (29) is for class 3 (PC) for which model has erroneously predicted class 4 (iOS).

Meanwhile, each line on the ROC curve illustrates each class one through five. The y-axis demonstrates the True positive rate while the x-axis demonstrates the false positive rate. It is explicitly visible that the AUC is relatively high for all curves ranging from 0.89 to 0.96. This indicates generally good performance in classification. There is no class with AUC below 0.89 suggesting there is no class that is particularly challenging to distinguish.

Further analysis along with the classification report reveals that classes such as Android, Nintendo Switch, and iOS exhibit strong performance with high

precision and recall while classes like PC and macOS show slightly lower scores suggesting room for improvement here, which could be a possible future improvement (refer appendix B).

Perhaps, the larger number of positive true classifications contribute to a higher accuracy rate of 79 %. Compared to the baseline accuracy of 38 % the logistic regression model performs much better with the accuracy of 79 %. This, aligns with the positive classification report, ROC curve and confusion matrix suggesting that the logistic regression model is truly a better fit in general and in comparison to the baseline.

5.3 k-Nearest Neighbour

Predicting the 'Platform' from the video games dataset with the help of the k-Nearest Neighbour algorithm resulted in a 0.78 accuracy, which is considerably higher than the baseline. Therefore, the kNN positively impacts the classification of the target label. This can also be seen by comparing the weights average precision and recall values of 0.77 and 0.78 with the ones from the baseline, which were 0.14 and 0.38 respectively.

The confusion matrix for kNN has the lowest amount of correct classified instances along the diagonal. It has the highest misclassification of the class 4 (iOS) with 24 instances that incorrectly classified. For all other classes the misclassifications do not surpass 20 instances. The ROC curve has the classes range between 0.87-0.96. The variance in between the curves is relatively low which suggests the model isn't a significant difference when the model is classifying one class with another.

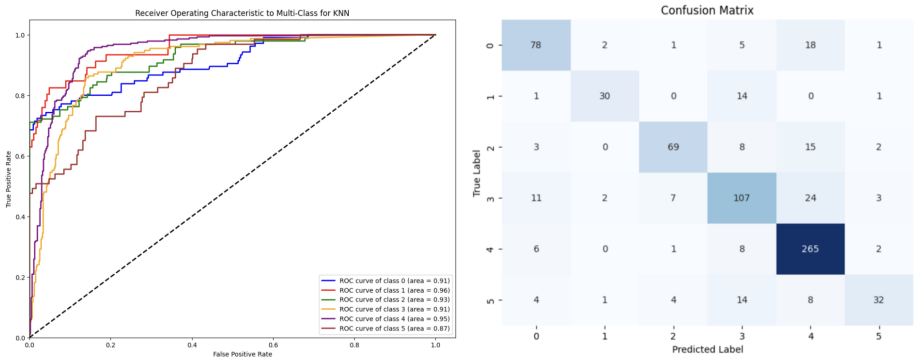

Figure 4: ROC Curve and Confusion Matrix for k-Nearest Neighbour

5.4 Decision Tree

The confusion matrix for the decision tree has the values 84, 32, 75, 101, 256, 32 when comparing the true label against the predicted label for the features. These values are very similar to values of the confusion matrix of logistic regression.

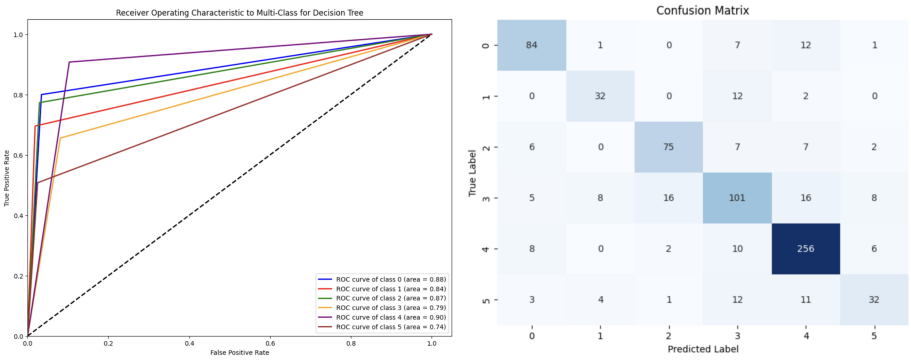

Figure 5: ROC Curve and Confusion Matrix for Decision Tree

This suggests that both models are similarly good at predicting the correct class from the dataset. The confusion matrix for the decision tree has better prediction for the correct labels compared to the KNN model. The highest amount of misclassifications is for the class Nintendo which has 16 instances mislabelled as Linux it also has 16 instances mislabelled as PC.

The ROC curve lines range between 0.74-0.9. This is a relatively wide range with some classes such as PC class being well predicted by the model while the model is not so good at classifying instances from iOS. The ROC curve suggests that the model is more inconsistent and not as reliable for each class.

The decision tree models an accuracy of 77.6 %, a precision score of 77 % and a recall score of 78 %. Based solely on these metrics this model is the worst performing, however, the difference between the measures of this model and the KNN model is extremely minimal. It does perform worse than the logistic regression model. Looking at the individual class metrics, the inconsistency shown in the ROC curve is also true for the metrics. Looking at the precision and recall scores, the class (iOS) with the highest precision and recall is 0.84 and 0.91 respectively. The worst class (maxOS) has a precision 0.65 and a recall of 0.51. The data for this model shows that the model has high correct predictions for certain classes but much lower for others. These fluctuations are not good for the model as a stronger model should have more uniform classification scores.

6 Evaluation and Conclusion

Notably, all the models have compelling results. To dive deeper, in terms of accuracy, the logistic regression model emerges first as the most accurate with an accuracy rate of 79 % followed by k-nearest neighbors model at 78 % and decision tree with 77.6 %. Furthermore, each model outperforms the baseline model's 38 % accuracy substantially. Hence, from accuracy the logistic regression models perform the best.

To analyse precision, a boxplot comparing the precision scores of the tree

different models were created(refer appendix D). The plot illustrates logistic regression with the highest median precision score of 0.8 along with the larger IQR indicating a general reliable prediction. This is followed by decision tree with a precision score of 0.75 and second better spread (IQR) and lastly the K-nn model with a precison of 0.79 and comparitievly less spread (IQR). Hence, it can be said that in terms of precision logistic regression is most precise followed by decision tree and logistic regression. Importantly, the box plot shows the spread of a classifier's classification skill. The logistic regression has the highest precision but has the largest spread meaning it is good at predicting certain classes and significantly worse than others. In comparison to the KNN model which performs slightly worse but has a lower variance in the precision this in general being more evenly precise in it's predictions.

Upon closer analysis of the model through confusion matrices and ROC plots, it is evident that all three models exhibit a higher frequency of misclassifying PCs compared to other classes, while iOS is inconsistently correctly classified the most. The decision tree model's highest misclassification is at 16 instances which have true label PCs but are misclassified as iOS and Nintendo. Similarly, logistic regression and k-nn demonstrate the same tendency by their highest number of misclassifications 29 and 24 respectively being misclassifying PC as iOS. Conversely, the most accurately predicted across all models iOS with K-nn achieving 265 correct predictions, followed by the decision tree with 256 and logistic regression with 258 correct predictions. Perhaps, the misclassification of PC into iOS could be due to iOS being the majority or dominant class.

Based on the results and metrics it is hard to conclude which model is best as one model has higher score for the metrics while having a larger variance in prediction. Overall, these models could all be beneficial for specific tasks regarding the dataset.

So, to conclude and answer the research question, the decision tree, K-nn, and logistic regression model perform decently well in predicting the target label "Platform" of the video games instances based on its features. All the models demonstrate significantly higher accuracy than the baseline model's accuracy of 38%. Perhaps, within the scope of this study, these models effectively address the goal of with adequate creativity and thoroughness. Nevertheless, future training and tuning the hyperparameters of the model to make them capable to handle unique datasets and possibly datasets with more instances of PC could test the true potential of the models and extending the findings of this report.

References

Haneen Arafat Abu Alfeilat, Ahmad BA Hassanat, Omar Lasassmeh, Ahmad S Tarawneh, Mahmoud Bashir Alhasanat, Hamzeh S Eyal Salman, and VB Surya Prasath. Effects of distance measure choice on k-nearest neighbor classifier performance: a review. *Big data*, 7(4):221–248, 2019.

Robert Boyle. The history of video games and the rise of esports. 2019.

Jason Brownlee. Information gain and mutual information for machine learning, Dec 2020. URL `https://machinelearningmastery.com/information-gain-and-mutual-information/`.

Stanley Cohen. The evolution of machine learning: Past, present, and future. In *Artificial intelligence and deep learning in pathology*, pages 1–12. Elsevier, 2021.

Robert W Crandall and J Gregory Sidak. Video games: Serious business for america's economy. *Entertainment software association report*, 2006.

Arundhati Navada, Aamir Nizam Ansari, Siddharth Patil, and Balwant A Sonkamble. Overview of use of decision tree algorithms in machine learning. In *2011 IEEE control and system graduate research colloquium*, pages 37–42. IEEE, 2011.

Chao-Ying Joanne Peng, Kuk Lida Lee, and Gary M Ingersoll. An introduction to logistic regression analysis and reporting. *The journal of educational research*, 96(1):3–14, 2002a.

Chao-Ying Joanne Peng, Tak-Shing Harry So, Frances K Stage, and Edward P St. John. The use and interpretation of logistic regression in higher education journals: 1988–1999. *Research in higher education*, 43:259–293, 2002b.

Neal Southern. The rise of esports: A new audience model and a new medium. *BA Camdidate, Department of Mathematics, California State University Staniaslaus*, 1:65–68, 2017.

Abdelhamid Zaidi. Mathematical justification on the origin of the sigmoid in logistic regression. *Central European Management Journal*, 30(4):1327–1337, 2022.

Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

7 Appendices

Appendix A

Features	Feature Description
Game Name	Name of the game
Website	Game Website
Rating	Rating rated by user
Playtime	Hours needed to complete the game
Suggestion Counts	Users who suggested the game
Reviews Counts	Users who reviewed the game
Developers	Game developers.
Publishers	Game publishers.
esrb_rating	ESRB ratings
added_status_yet	User that "not played" the game
added_status_beaten	User that "completed" the game
added_status_toplay	User that "to play" the game
added_status_playing	User that "playing" the game
Main Genre	Genre of the games

Figure 6: Remaining Features

Appendix B

Classification report		
	precision	recall
Android	0.9	0.8
Linux	0.8	0.72
Ninentendo Switch	0.91	0.75
PC	0.69	0.71
iOS	0.8	0.91
macOS	0.71	0.57
accuracy		
macro avg	0.8	0.74
weighted avg	0.8	0.79

Figure 7: Classification Report - logistic regression

Appendix C

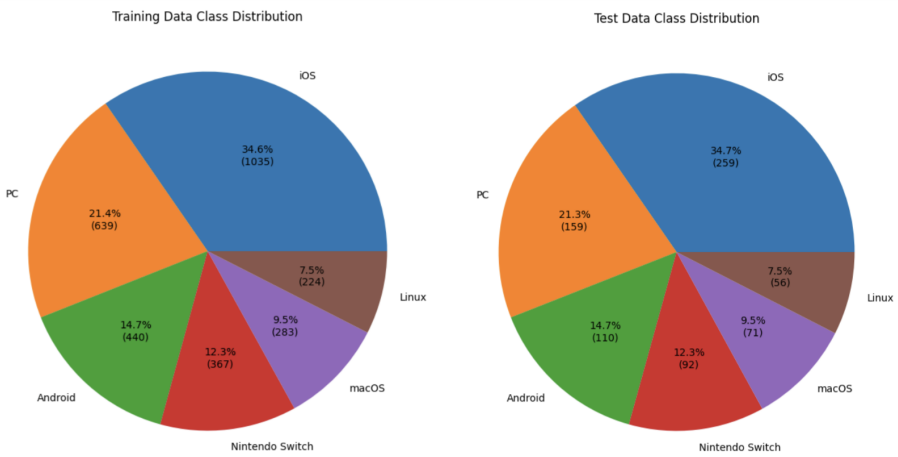

Figure 8: Training and Test Data Class Distribution

Appendix D

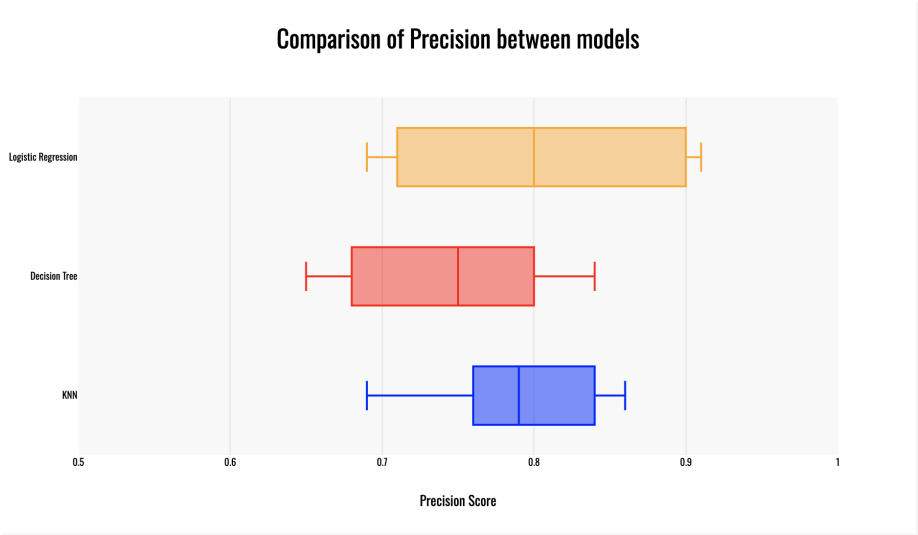

Figure 9: Comparing the precision values between models

Appendix E

Gridsearch: Decision Tree

Max depth	Min sample split	Min sample leaf	Accuracy score
5	2	1	0.68
5	2	2	0.65
5	2	3	0.67
5	5	1	0.70
5	5	2	0.71
5	5	3	0.69
5	10	1	0.73
5	10	2	0.68
5	10	3	0.66
10	2	1	0.77
10	2	2	0.76
10	2	3	0.74
10	5	1	0.72
10	5	2	0.70
10	5	3	0.69
10	10	1	0.73
10	10	2	0.72
10	10	3	0.67
15	2	1	0.78
15	2	2	0.76
15	2	3	0.71
15	5	1	0.74
15	5	2	0.75
15	5	3	0.71
15	10	1	0.79
15	10	2	0.76
15	10	3	0.71

Note: The highest accuracy was not chosen because the model would have a higher chance of overfitting. Instead, the value was chosen with the highest accuracy while taking into account which hyperparameters would lead to the least chance of overfitting.

Figure 10: Gridsearch: Decision Trees