Alex Churchill, Chang He, Emma Oriol
June 2023

Data-Driven Kart Racing: Utilizing Machine Learning to Dominate the Tracks in Mario Kart 8 Deluxe

Abstract:

Mario Kart is a wildly popular video game designed by Nintendo where users customize their characters and vehicles and race around tracks. This paper utilizes unsupervised learning techniques, specifically K-means clustering and hierarchical clustering, to identify distinct vehicle classes with shared characteristics. The results of the clustering techniques showed that vehicles are best grouped by steering and speed abilities, but there are no distinct classes of vehicles in the data. Additionally, a neural network was fit to predict the character drivers of each kart based on the features of the kart, which performed with 100% validation accuracy. The results of this paper enhance user gameplay by providing insight into vehicle and character selection.

Introduction:

Welcome to our report on “Data-Driven Kart Racing: Utilizing Machine Learning to Dominate the Tracks in Mario Kart 8 Deluxe.” Within this report, we will explore the application of unsupervised learning techniques to identify distinct vehicle classes with the popular video game, Mario 8 Deluxe. Additionally, we will use a neural network to predict the character drivers of each kart based on the features of the kart, enhancing the user gameplay by providing insights into vehicle and character selection.

Developed by Nintendo, Mario Kart is a widely beloved game where players customize their characters and vehicles to compete in thrilling races on various tracks. This unique gameplay paired with a variety of characters to choose from has captivated many people of all ages worldwide. Nintendo’s characters have gained huge popularity, extending beyond the gaming realm to even feature in a motion picture.

Our project focuses on character drivers and the characteristics of their karts. We aim to determine the uniqueness of each character by evaluating the error rate and types of errors made when classifying the character drivers based on the kart’s characteristics. If a neural network cannot accurately determine the drivers of a kart, it suggests that those character combinations have gameplay styles which are too similar to offer players unique experiences. Consequently, the kart features should be adjusted to offer a wider diversity of gameplay.

In order to do this, we will use dimensionality reduction techniques such as Principal Component Analysis (PCA) that allow us to transform the high-dimensional dataset into a lower-dimensional space, while still keeping the essential information while simplifying visualization and interpretation.

Additionally, we will also explore the application of K-means clustering, an unsupervised learning algorithm, to partition the data points into distinct clusters based on shared characteristics. Hierarchical clustering, another unsupervised learning algorithm, will be employed to create a hierarchy of nested clusters, enabling us to explore different levels of granularity without the need to specify how many clusters we would want to use.

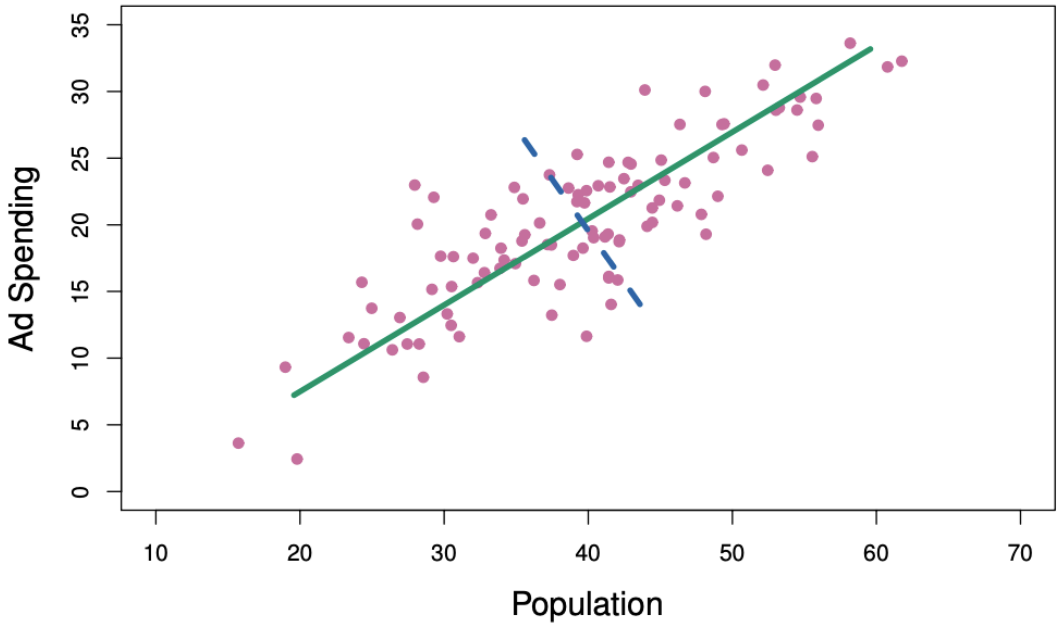
These analyses will be performed on the “Mario Kart 8 Deluxe Kart Stats” Dataset from data.world. Mario Kar8 Deluxe allows players to choose different combinations of characters, karts, and gliders and a kart with unique characteristics is generated for each combination. This dataset contains every possible combination and the statistics that result from each.

Theoretical Background:

Principal Component Analysis (PCA):

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique in machine learning and data analysis. This method aims to transform a high-dimensional dataset into a lower-dimensional space while retaining the essential information. It achieves this by identifying the directions, known as principal components, along which the data exhibits the most significant variation.

Principal Component Analysis uses mathematical equations to derive the orthogonal axes (principal components) that capture the maximum variance in a dataset. The first principal component represents the direction of maximum variation, and each subsequent principal component is orthogonal to the previous ones and captures the remaining variance in a dataset. By projecting the original dataset onto these principal components, we can effectively reduce the dimensionality of the dataset.



Here is an example of PCA from the textbook *An Intro to Statistical Machine Learning.* Using the Advertising dataset and applying PCA, this graph analyzes the population size (in tens of thousands of people) and ad spending (in thousands of dollars) for 100 different cities (purple dots). The green solid line indicates the first principal component, which has the greatest variability in the data. The blue dashed line indicates the second principal component, which has the second greatest variability in the data. Together, these two variables explain the most variability of the Advertising dataset.

This can be helpful in various scenarios. It helps to simplify complex datasets with a large number of variables, allowing for easier visualization and interpretation, it identifies the most informative features by ranking them based on their contribution to the total variance of the dataset. It can be employed as a preprocessing step to decorrelate variables and remove redundancy, and it can help filter out the noise and focus on the dominant underlying patterns in the data.

Singular Value Decomposition (SVD):

Singular Value Decomposition (SVD) is a matrix factorization technique used to decompose a matrix into multiple matrices. Given a matrix, SVD separates it into three smaller matrices, which can provide valuable insights into the data.

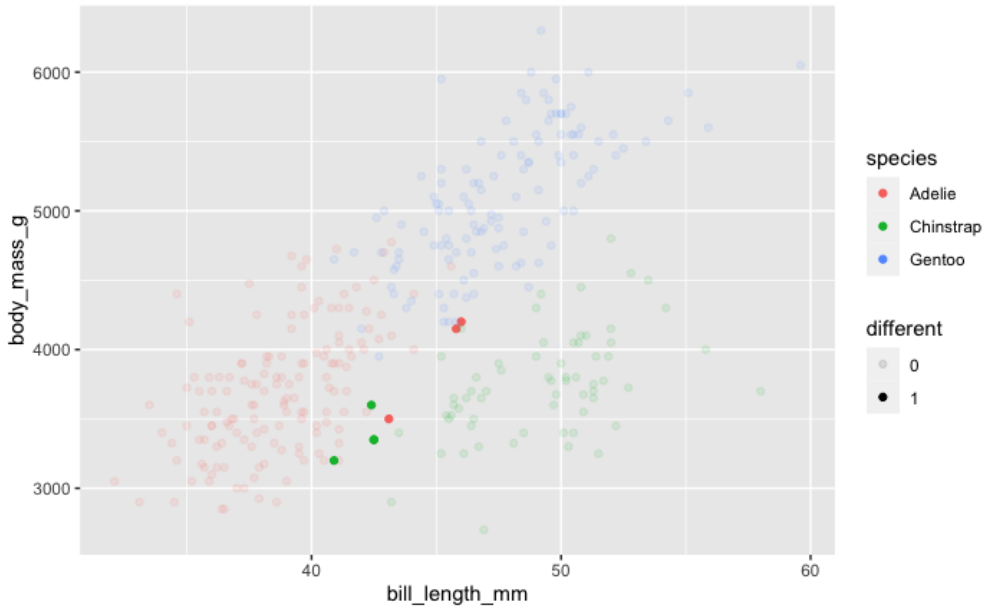
Similar to PCA, SVD can help reduce the number of variables in a dataset. By selecting only the most important variables, we can simplify the data without losing too much information. SVD can also be used to estimate or approximate a matrix using fewer values. This helps when

compressing the data and reduces computational cost/space. The main difference is the way they process the dataset. PCA focuses on finding the most important patterns by looking at how the measurements relate to each other, while SVD directly breaks down the data into simpler patterns and values without considering the relationships between the measurements. Also, PCA assumes all variables contribute equal importance to the variance of the data. Conversely, SVA can be useful for correcting confounding factors and addressing differences in variable scale.

K-means Clustering:

K-means clustering is another popular unsupervised learning algorithm used for partitioning data points into distinct groups of clusters. The goal is to assign data points to clusters in such a way that the data points within the same cluster are more similar to each other than those in other clusters.

The K refers to the number of clusters that need to be identified in the dataset. The algorithm works by iteratively optimizing the clustering assignment based on minimizing the sum of squared distances between data points and their corresponding cluster centroids.



Here is a graph that analyzes the relationship between a penguin's body mass (measured in grams) and its bill length (measured in millimeters) from the Penguins dataset in An Introduction to Statistical Machine Learning. Each color corresponds to a penguin species and the solid colors represent the false predictions made from the K-means clustering model. Here, the model chose 3 clusters to analyze and as a result, a very small error is made when predicting penguin species.

K-means clustering is applicable in many scenarios like identifying groups of customers with similar behavior or preferences, dividing an image into regions or segments with similar pixel characteristics, identifying unusual or outlier data points that do not conform to the normal clusters, and grouping documents based on their content for document organization. Keep in mind that this method requires us to specify the number of clusters we want to use as a parameter when running the model. Determining the optimal value of K can be a challenge and requires further knowledge of the dataset.

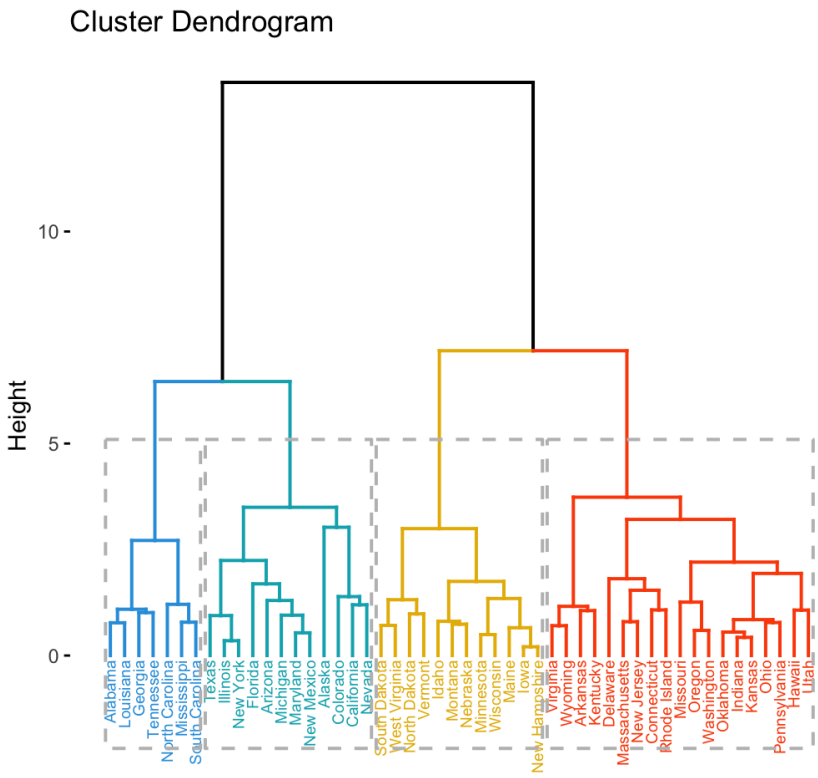
When performing K-means clustering, the “start” parameter determines the number of times the K-means algorithm will run. The model with the lowest within-cluster variance is then chosen. Each time the model runs, it randomly initializes the centroids (cluster centers) to the data points. Clustering results can be highly sensitive to the initialized centroid, so it is important to run enough models to compare results and select the strongest one.

Hierarchical Clustering:

Hierarchical Clustering is another unsupervised machine learning algorithm used to cluster data points into a hierarchy of nested clusters. Hierarchical clustering does not specify the number of clusters beforehand like k-means clustering. Instead, this method builds a tree-like structure of clusters, referred to as a dendrogram, where each data point starts as an individual cluster and gradually merges with other clusters based on their similarity.

The similarity or dissimilarity between clusters is typically measured using distance metrics such as Euclidean distance or correlation coefficients. Different linkage methods, such as single, complete, or average linkage determine how the similarity between clusters is calculated during the merging process:

- In a single linkage, the similarity between two clusters is defined as the minimum distance between two data points. one point from each cluster. It focuses on the closest pair of data points between two clusters, considering it as a representative of the similarity between the clusters. This approach is best used when dealing with elongated or chain-like clusters in the data.
- In complete linkage, the similarity between two clusters is defined as the maximum distance between any two data points, one from each cluster. This focuses on the most dissimilar pair of data points between two clusters that represent the similarity between the clusters. This is useful when we deal with unevenly sized clusters or when we want to identify distinct and well-separated clusters.
- In average linkage, the similarity between two clusters is defined as the average distance between all pairs of data points in each cluster and uses that to represent the similarity between them. This is a balance between the compactness of complete linkage and the robustness of a single linkage. It helps to handle clusters of different sizes in a dataset.



This Dendrogram from Datanovia uses the USArrests dataset which contains information about the arrests per 100,000 residents for various crimes in each of the 50 US states in 1973. From the bottom, each leaf corresponds to one object. As we move up the tree, objects that are similar to each other are combined into branches, which are themselves fused as we move up the tree.

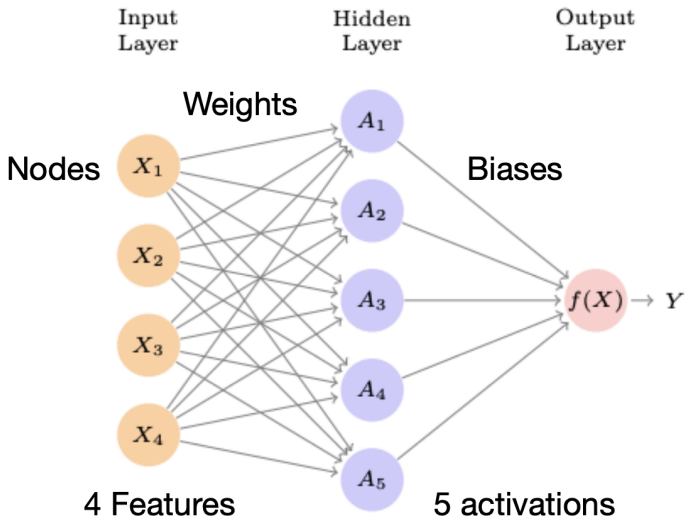
Hierarchical clustering provides a hierarchical structure of clusters, which allows us to explore different levels of granularity. This provides us with a comprehensive view of the data, especially when the number of clusters is unknown. It is important, however, to keep in mind that this method is very computationally expensive when dealing with large datasets.

Neural Networks

Neural networks are highly predictive and flexible models that are widely used to model relationships between non-linear and high-dimensional variables. However, high accuracy comes at the cost of interpretability. Neural networks are often referred to as “black boxes”, since the details of the model that produces the output are unknown.

The characteristics of neural networks are a result of the way they are designed. They are built in layers, where each layer processes the output of the previous layer. Each layer also has a defined number of nodes. Nodes process outputs from the previous layer using some predefined activation function. The first layer is known as the input layer, which has one node for

every feature in the dataset. The input layer is followed by hidden layers, which have a user-specified number of nodes, and then the output layer, whose nodes change depending on the problem. The below image shows the layers of a neural network and how they interact. It depicts a model with four features, a hidden layer with five nodes, and an output layer with one node.



The input layer transforms the features of a dataset using a linear activation function. The equation is shown below, where a weight constant is added to the sum of the weight for every observation multiplied by the observation.

$$w_{k0} + \sum_{j=1}^{p} w_{kj} X_j$$

This equation generates the inputs for the next set of layers, known as hidden layers. Hidden layers apply a nonlinear, user-specified activation function to the outputs from the previous layer. There are many possible activation functions, but the models in this paper use rectifier functions, commonly known as relu functions. Relu functions take in a value from the previous layer and return that value if it is over zero. If it is below zero, it returns zero. There can be one to many hidden layers in a neural network.

Output layer activation functions are specified by the user depending on the type of model. For example, a binary classification model may use a sigmoid function, which has two possible outputs, while a regression function may use an identity function, which outputs the value from the hidden layers as they are. The model in this paper uses a softmax function for multinomial classification. Softmax functions create a probability distribution for a range of possible

categories and uses this distribution to classify observations. The formula for a softmax function is below.

$$g(z) = \frac{e^{zj}}{\sum e^{zj}}$$

Inputs are processed by the model in chunks known as batches. Batch size can be tuned by the user and can affect run time. Additionally, all input data is passed through the layers of a network a predetermined number of times, known as epochs. After an epoch finishes, the weights of the linear input function are updated to improve the model based on a loss function. During the first epoch, the weights of the linear input function are randomized. A loss function is a metric used by the neural network to evaluate its own performance. Loss functions are also chosen based on the type of problem. For example, mean squared error can be used for regression problems. The models in this paper use a sparse categorical cross entropy loss function for multinomial classification. The model aims to adjust the weights of the model to minimize sparse categorical cross entropy loss.

Finally, there are many types of neural networks available. One type of network is a recurrent neural network. Recurrent neural networks are used to model sequential input data. As a result, hidden nodes take inputs from one feature at a time. Once a hidden layer processes an input, it passes the output as a final output for that feature and as an input to the next activation node, where it is combined with the next feature as an input. Convolutional neural networks are networks designed to mimic the way human vision works to classify images and are also widely used. Image data is read into the network in a two-dimensional array. A subset of the array is selected and passed over the image many times. Initially, the subset filters the image to select low-level features such as edges and color. As passes continue, the subset adds more and more detail, until it recreates an approximation of the original image. These subsets are known as convolution layers. Once the layers are finished, they are combined in a process known as pooling. The combined convolutions create a slightly smaller approximation of the original image. This process can be repeated many times. Once a model is fit to an image, it can be shown new images and will classify them based on what it learned from the images it trained on.

Methodology:

Clustering

Before beginning any analyses, the data was loaded into R Studio as a data frame. Data cleaning and preprocessing included confirming there was no missing, null, or invalid data. Additionally, a column was added for each entry indicating “bike” or “car” as a vehicle class.

After the initial data preprocessing, the data was prepared for unsupervised learning analysis. First, dimension reduction was performed using principal component analysis (PCA). This helped simplify the high dimensional data, increase computational efficiency, and understand the most relevant features of the data that contribute to the most variation. All factor variables were removed, and then PCA was performed on the remaining numerical variables. The 'scale' parameter was used to scale/standardize the data. The "total" variable was several factors higher than all other variables, so scaling the data helped prevent the total variable from having higher importance than all other variables.

Next, K-Means clustering was performed on the PCA-processed data to classify the data into two, three, and four groups. Each time, the parameter 'nstart' was set to twenty, so that the data was initialized with a randomly selected starting centroid twenty times, and the cluster with the lowest within-cluster variance was selected. The results were then plotted and the total within the group sum of squares was measured to assess variance within each cluster.

Lastly, hierarchical clustering was performed. The data was randomly subset to a sample size of 1,000 because hierarchical clustering is highly computationally expensive and does not run on large data sets. Next, complete, average, and single-linkage hierarchical clustering was done and the resulting dendrograms were printed. Finally, the trees were pruned at four branches to compare clusters with the K = 4 K-means clustering results.

Neural Networks

The data for the neural network was loaded into an r dataframe for cleaning. The "total" column was removed to avoid redundancy with other metrics. In order to make a neural network computationally feasible for a single laptop, 10% of the observations were sampled. This sample was then divided into training and test sets with an 80/20 split. There were four categorical variables that were encoded. A custom encoding function was built to produce matrices for each categorical variable. These matrices were then concatenated along with the numerical variables into an input matrix. The final matrix had dimensions of 18,816x81. The first axis represents one kart and the second axis represents the features for each kart. The response variable, characters, was stored in a vector. This vector had sixteen categories, one for each possible character combination.

A softmax output function was used to predict which of the sixteen character combinations each kart was driven by. Since the input function is predetermined, the model tuning was focused on the hidden layer nodes. Models were tuned in order to achieve the highest validation accuracy in the shortest amount of time.

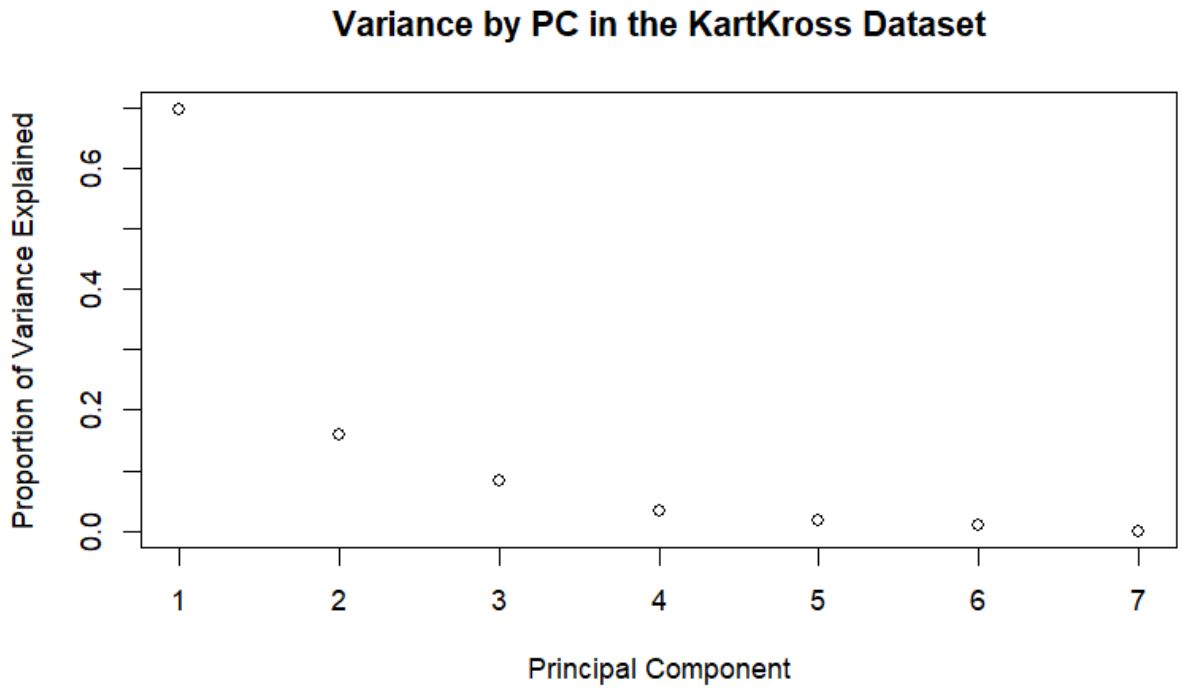
A single hidden layer was used with node values of 40, 81, and 162. These values were chosen since they are multiples of the number of features in the data (with the exception of 40, which was used to represent one-half the number of features). Each model was timed using the tictoc library and was run using a batch size of 20. In addition, one model with two layers was tested. The first layer used 162 nodes, since that number of nodes yielded the best results among the

models with one hidden layer. The second layer used 81 nodes, the number of nodes that yielded the second-best results amongst networks with one hidden layer. Additionally, every model used a batch size of 200, which was chosen due to the smooth accuracy and loss curves generated by this batch size. Epochs were added until 100% validation accuracy was reached.

Computational Results:

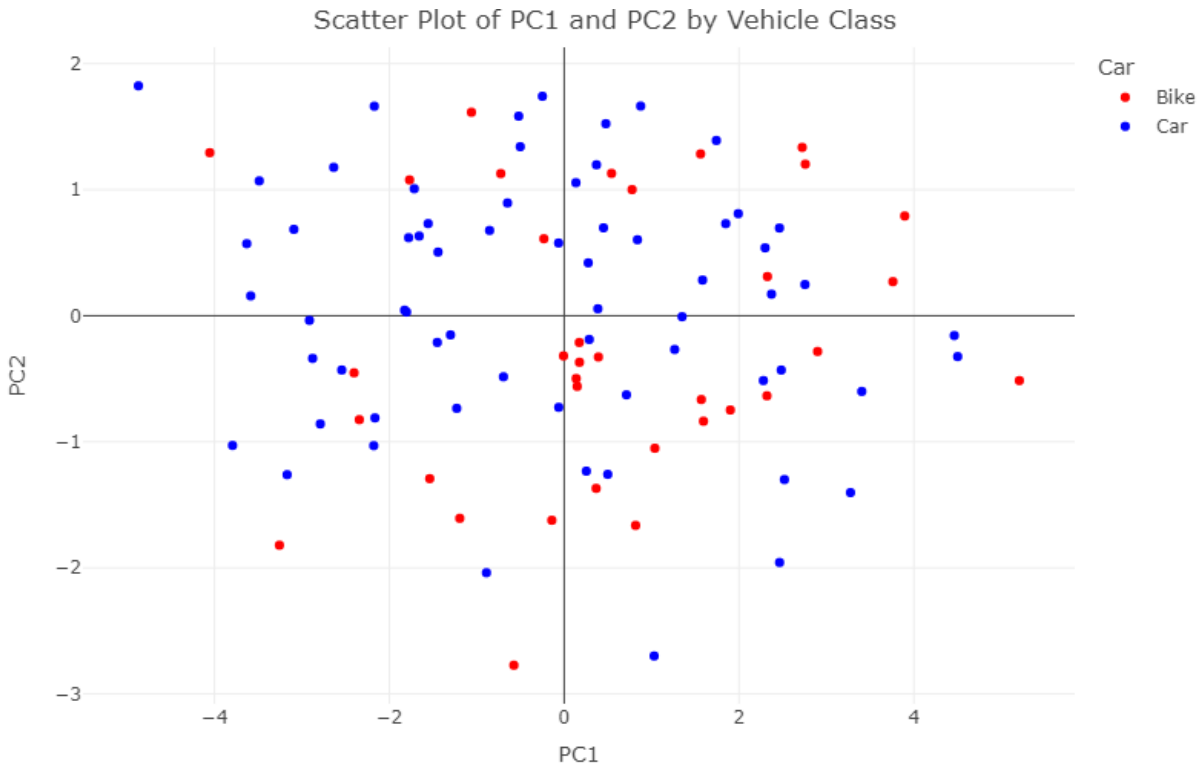
Clustering

The results of the principal component analysis show that the first two principal components account for over 90% of the variance in the data set. This is demonstrated by the figure below, which plots the variance in the KartKross dataset accounted for by each principal component.



Principal component one is composed of mostly speed, acceleration, and turbo. Principal component two is composed mostly of traction, total, and handling. Therefore, principal component one seems to describe how fast a vehicle is while principal component two describes how well it steers.

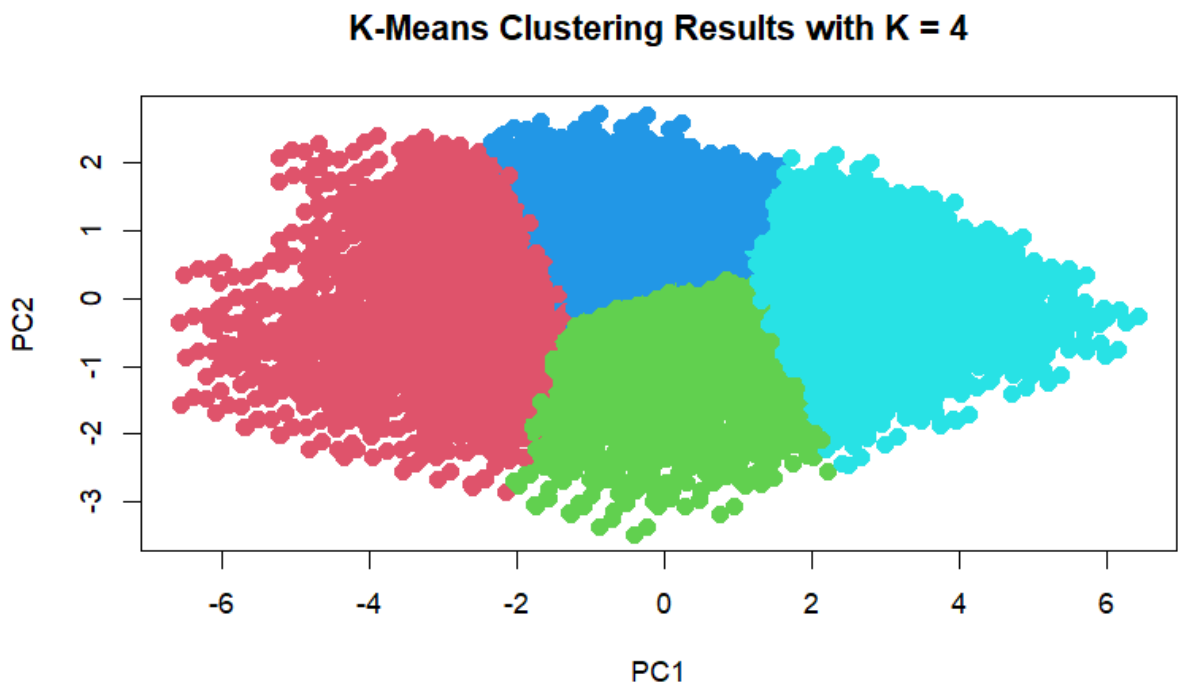
A randomly selected sample of the data was plotted along the first two principal components, with each point labeled as “bike” or “car”. The plot, which is shown below, did not show any apparent grouping of bikes or cars, suggesting that these two classes of vehicles don’t necessarily share characteristics.



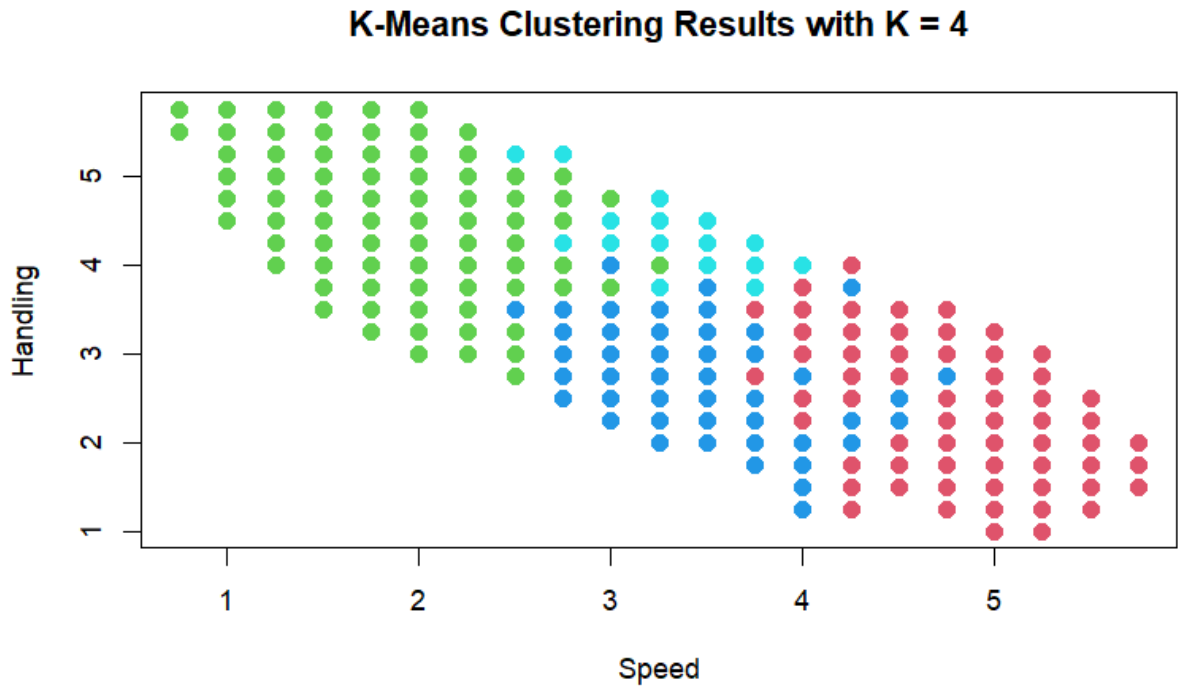
K-means clustering was performed using two, three, and four clusters. For two and three groups, the data was divided entirely along principal component one. In four groups, the data was segmented along principal component two as well. The total within-group sum of squares decreased as group size increased, which is to be expected and does not necessarily indicate which group size is optimal. Ultimately, the clusterings do not necessarily reflect any natural segmentation in the data. The algorithm is merely finding the most similar points, based on their principal components, and clustering from there. The plots below show the clusters for two, three, and four groupings.


K-Means Clustering Results with K = 2
PC2
PC1
-6
-4
-2
0
2
4
6
-3
-2
-1
0
1
2

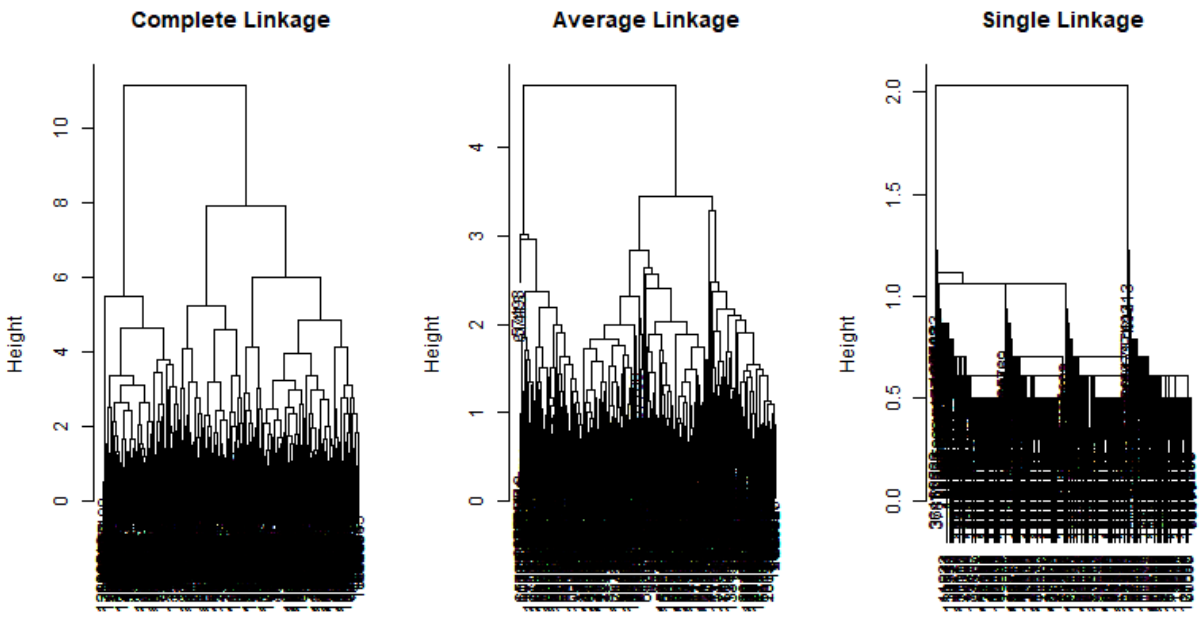

K-Means Clustering Results with K = 3
PC2
PC1
-6
-4
-2
0
2
4
6
-3
-2
-1
0
1
2



Plotting the K-Means results with K=4 against speed and handling gives a little more insight into characteristics of the clusters. The green cluster has strong handling and low speed, the light blue has medium speed and medium-high handling, the dark blue cluster has medium speed and medium-low handling, and the red cluster has high speed and low handling.



Hierarchical clustering using complete linkage, average linkage, and single linkage methods produced the following dendrograms. The Average and Complete Linkage hierarchical clustering show similarly balanced dendrograms.



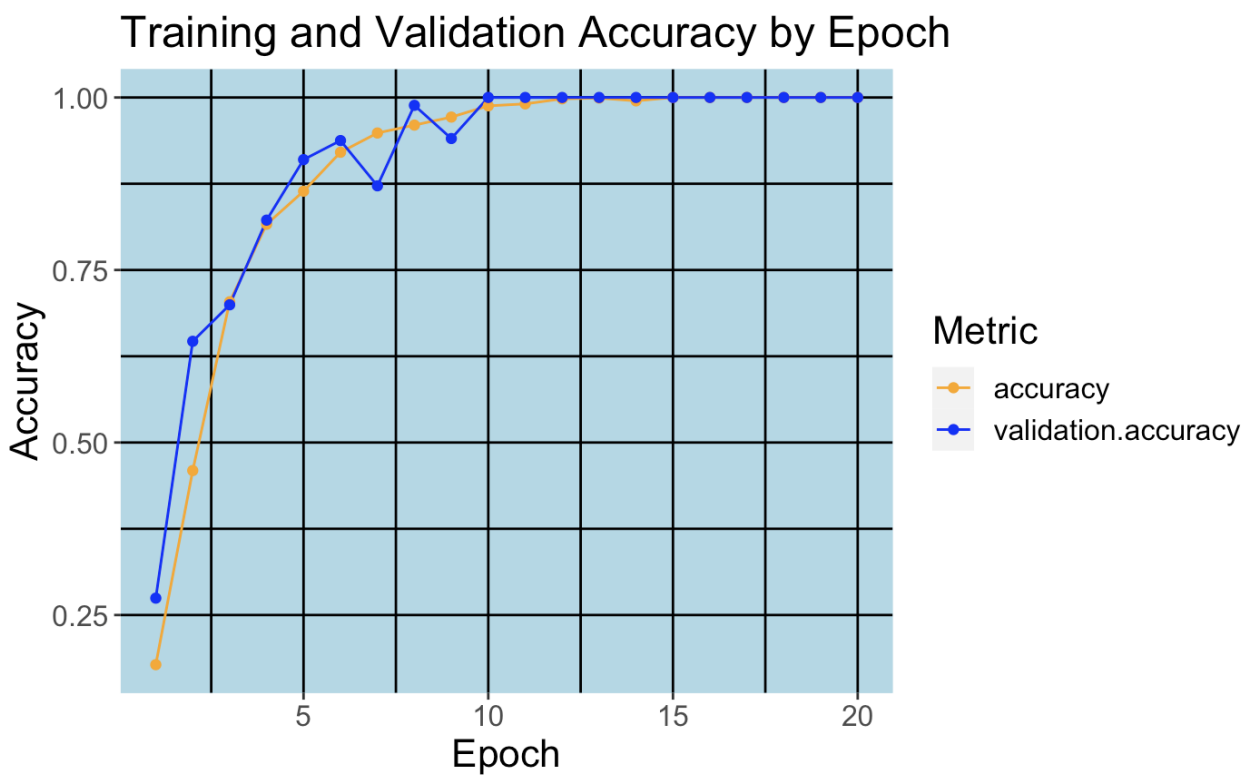
Using the complete linkage clustering, the dendrogram was “cut” so there are a total of four branches that categorize the data into four clusters. Comparing the complete linkage clustering to the K-means clustering, 79.9% of the data is clustered identically.

Neural Networks

Every neural network was able to achieve 100% validation accuracy, but there were slight differences in the time taken by each model. Additionally, all models used a batch size of 200. The below chart summarizes the results of the models.

Hidden Layers	Nodes	Epochs	Time
1	40	39	7.523
1	81	26	5.39
1	162	18	4.137
2	162, 81	10	2.841

The model with two hidden layers performed best, with a run time of just 3.581. Generally, as the number of nodes and layers increased the time taken to reach 100% validation accuracy decreased. The below graph shows the accuracy curves for the neural network with two hidden layers.



The accuracy increases logarithmically until it reaches 10 epochs, at which point the model reaches convergence and can't improve anymore.

In addition, predictions were obtained for the test using the fitted two-layer model and the predict function. This method yielded an accuracy of 99.8%. The misclassifications are shown below.

Actual	Predicted
Luigi, Iggy	Baby Mario/Luigi, Dry Bones
Luigi, Iggy	Koopa Troopa, Lakitu, Bowser Jr
Luigi, Iggy	Baby Mario/Luigi, Dry Bones
Luigi, Iggy	Tanooki Mario, Inkling Boy, Villager Boy
Luigi, Iggy	Peach, Daisy, Yoshi

Luigi, Iggy	Link, King Boo, Rosalina
Luigi, Iggy	Baby Mario/Luigi, Dry Bones

Discussion:

Clustering

The K-means clustering and dendrograms performed similarly, with almost 80% of the data grouped identically between the Complete Linkage dendrogram and the K=4 means clustering. Examining data points at random from each cluster showed that the clusters were grouped by varying levels of speed and handling (high speed with low handling, medium-low speed with medium-high handling, medium-high handling with medium-low speed, and low speed with high handling).

The results from the unsupervised analysis did not reveal any distinct separate groupings within the data. Importantly, there was no pattern of difference in characteristics between bikes and cars as users might expect to find. Principal component analysis showed that vehicles are best described by their overall speed (speed, acceleration, turbo) and ability to steer (handling, traction). While the data was easily segmented into groups using these principal components, the groups and the number of groups were relatively arbitrary and merely reflected varying levels of steering and speed and the number of groups defined by the programmer.

This reflects that Mario Kart game designers probably did not intend for there to be any ‘superior’ or ‘inferior’ class of vehicles. Instead, users are challenged to consider the tradeoffs of speed and steering when choosing a vehicle. This can enhance the gameplay experience by adding layers of strategy to vehicle selection, i.e. there is no one ‘right’ or ‘best’ vehicles to choose from. Instead, users must consider what attributes of a vehicle are most important to them in their upcoming races.

Neural Networks

The primary limitation encountered during the analysis was computational power. However, the bottleneck was the custom-made encoding function rather than the neural network. It was written using a nested loop. The first loop iterates over the unique values in a given column and an encoded vector is created for each value by iterating over the entire column. This means that for a column with 10 unique values and 1,000 observations, 10,000 iterations would be needed. This function is not ideal, but it works. Other methods were attempted, including encoding packages, but this was the only method that yielded a matrix which could be fed to a neural network.

The predictions were excellent from all models, yielding 100% validation accuracy from each. This suggests that the character combinations are distinct enough to provide players with a unique gameplay experience regardless of which characters they choose.
The errors generated by the predict function shed some light on the kinds of characters that have more overlap in gameplay style. For example, Luigi and Iggy are both characters who historically have had low rankings in power statistics such as speed and weight but have higher rankings in finesse statistics such as handling. With the exception of the kart which was misclassified as “Link, King Boo, Rosaline”, all the misclassifications of “Luigi, Iggy” are combinations of characters with high finesse statistics. For example, “Peach, Daisy, and Yoshi” are also characters with historically high finesse, so it is reasonable that the model would misclassify a “Luigi, Iggy” Kart as a “Peach, Daisy, Yoshi” Kart.

Interestingly, we did not see karts with traditionally high power stats being misclassified. One way this analysis could be improved would be to further analyze why certain karts or kart groups are misclassified and others are not. For example, it would be interesting to further investigate why only “Luigi, Iggy” karts were classified as other finesse characters, but not the other way around.

Conclusion:

Clustering

When selecting characters and vehicles, users should be mindful that optimizing speed or steering generally comes as a tradeoff. This means that as speed or steering increases, the other decreases. Users should tailor their character and vehicle choices to the tracks they intend to race (i.e. a vehicle with strong steering could be better suited for more complex tracks with more obstacles) or choose a vehicle with middle-of-the-road speed and steering for best overall performance. There is no ‘best’ or ‘ideal’ vehicle, and instead, users are challenged to apply strategy to their vehicle selection with consideration to their upcoming tracks, racing style, and personal preferences.

Neural Networks

The complete accuracy of the neural networks suggests that the character combinations in Mario Kart 8 offer unique gameplay experiences. Therefore, the balancing and game design of the characters is in line with Nintendo’s market position as a brand that offers varied and exciting experiences based on playable characters. Large game developers such as Nintendo may consider including neural networks as one method among others for evaluating game balance and gameplay experience while testing future iterations of Mario Kart. During testing, neural networks can be run and their errors analyzed to find karts that are too similar to have a distinct gameplay experience.

It should be noted, however, that this analysis focuses on gameplay from a strictly statistical point of view. There are many elements that go into a successful game such as art style, map

design, and audio design. Therefore, it is important to see neural networks as one possible tool for evaluating gameplay.

References:

Brownlee, J. (2019, October 22). Loss and loss functions for training Deep Learning Neural Networks. MachineLearningMastery.com. https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

"Datanovia. 'Agglomerative Hierarchical Clustering.'" Datanovia. N.p., n.d. Web. https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/

Doug (https://stats.stackexchange.com/users/438/doug), How to choose the number of hidden layers and nodes in a feedforward neural network?, URL (version: 2022-08-31): https://stats.stackexchange.com/q/1097

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer, 2013.

James, Gareth. Witten, Daniela. Hastie, Trevor. Tibshirani, Rob. An Introduction to Statistical Learning. 2nd Edition. 2021, https://www.statlearning.com/

Ortiz, Jonothan. "Mario Kart 8 Deluxe Kart Stats." Data.world. Web. 8 Sep. 2017. https://data.world/databeats/mario-kart-8-deluxe-kart-stats

“Softmax Function” Wikipedia. Accessed 05/13/2023. https://en.wikipedia.org/wiki/Softmax_function

Wikimedia Foundation. (2023, June 4). The super mario bros movie. Wikipedia. https://en.wikipedia.org/wiki/The_Super_Mario_Bros._Movie#:~:text=Reception-,Box%20office,worldwide%20total%20of%20%241.300%20billion

Yathish, V. (2022, August 4). Loss functions and their use in neural networks. Medium. https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9#:~:text=A%20loss%20function%20is%20a,the%20predicted%20and%20target%20outputs