Option #2 - Video Game Genres with K-Nearest Neighbors

A Machine Learning Project

# Introduction

Video games are a popular form of entertainment that appeal to a wide range of audiences. However, not all gamers have the same preferences when it comes to video game genres, such as action, adventure, role-playing, simulation, strategy, or sports. Therefore, it would be useful to have a system that can recommend video games to gamers based on their favorite genres. One possible way to achieve this is to use machine learning techniques to classify video games into different genres based on their features, such as graphics, gameplay, story, or sound.

In this project, we will use the K-nearest neighbors (KNN) algorithm to predict the favorite video game genre for a random test row. The KNN algorithm is a supervised machine learning method that classifies an unknown instance based on the majority vote of its k closest neighbors in the feature space. We will assume that we have a dataset of video games with their features and genres, and that we have preprocessed and split the data into training and test sets. We will also assume that we have scaled the features to have a mean of zero and a standard deviation of one, which is a common practice to improve the performance of the KNN algorithm.

# Code Explanation

The code snippet below shows how to use the KNN algorithm to predict the favorite video game genre for a random test row. The code is written in Python and uses the scikit-learn library, which is a popular tool for machine learning in Python. The code consists of four main steps:

1. Creating and fitting a KNN object to the training data.
2. Selecting and scaling a random row from the test set.
3. Predicting the genre for the test row using the KNN object.
4. Printing the predicted genre to the console.

# Import the KNeighborsClassifier class from scikit-learn

from sklearn.neighbors import KNeighborsClassifier

# Create a KNN object with a predefined number of neighbors

knn = KNeighborsClassifier(n\_neighbors = 5)

# Fit the KNN object to the training data

knn.fit(X\_train\_scaled, y\_train)

# Select a random row from the test set

random\_test\_row = X\_test.iloc[0]

# Scale the test row using the same scaler that was used for the training data

random\_test\_row\_scaled = scaler.transform(random\_test\_row.values.reshape(1, -1))

# Predict the favorite video game genre for the test row

predicted\_genre = knn.predict(random\_test\_row\_scaled)

# Print the predicted genre to the console

print("Predicted favorite video game genre:", predicted\_genre[0])

# Screenshot

A screenshot of a computer

Description automatically generated

# Conclusion

The code demonstrates how to apply the KNN algorithm to a multiclass classification problem, where the goal is to predict one of several possible categories for a given instance. The KNN algorithm is simple and intuitive, but it can also be computationally expensive and sensitive to noise and outliers. Therefore, it is important to choose an appropriate value for k, the number of neighbors, and a suitable distance metric, such as Euclidean or Manhattan, to measure the similarity between instances. The code can be modified to use different values of k, different distance metrics, or different features to see how they affect the accuracy of the prediction. The code can also be extended to evaluate the performance of the KNN algorithm on the entire test set, or to compare it with other machine learning algorithms, such as decision trees, logistic regression, or neural networks.

Ref - [What is the k-nearest neighbors algorithm? | IBM](https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20(KNN,of%20an%20individual%20data%20point.)

[K-Nearest Neighbor(KNN) Algorithm - GeeksforGeeks](https://www.geeksforgeeks.org/k-nearest-neighbours/)