What Makes a Video Game Successful?

Group 6

Analysis by Jackson Windhorst, Nicholas Clewley, Munchootsorn Wangsriviroj, Sean Murray, William Rocha

Why This Info Matters

Game Studios can align development with proven consumer preferences.

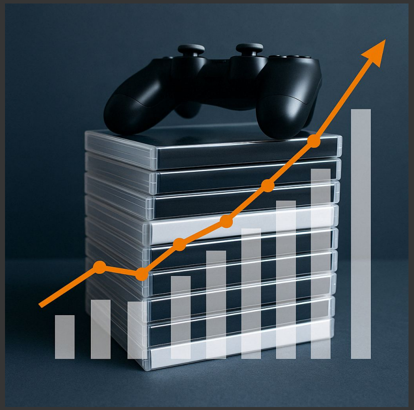
Indie developers boost their chances of breakout success

Marketers can tailor campaigns to regions with the highest sales potential

Publishers can refine release timing and platform strategy for maximum impact

Investors gain a predictive edge in a competitive market

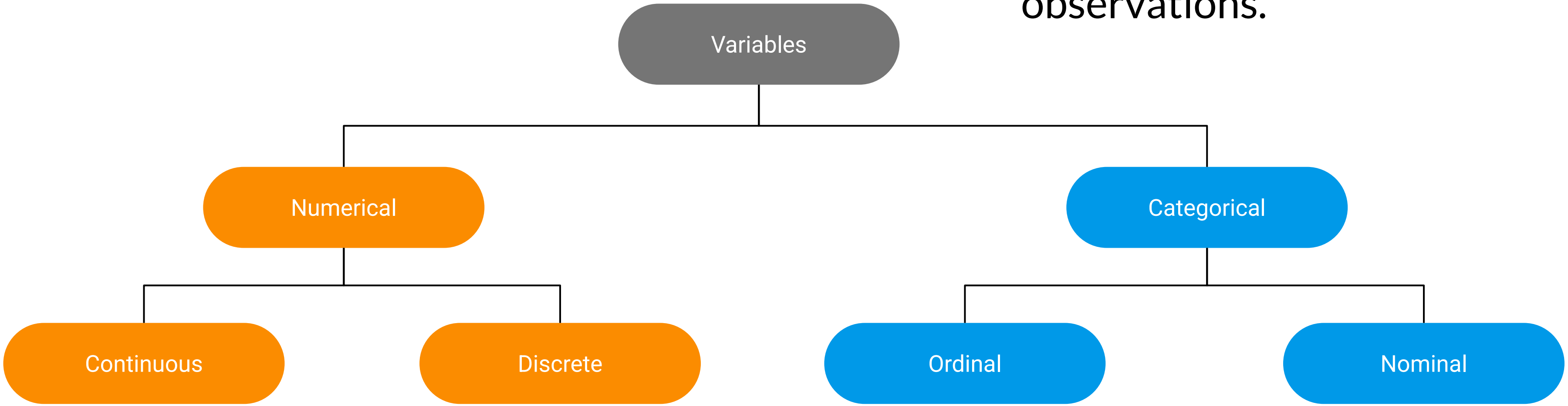
Model Building Process



Define Problem	Data Collection	EDA	Evaluation	Validation	Maintenance
What makes a video game successful?	Online dataset	Visuals, summary statistics, and feature engineering	Train/test set split Performance metrics Cross Validation	Determining if the model will perform well in its intended environment	Pipelines and monitoring tools to keep peak model performance

Our Data

All the data was sourced from Kaggle. It contains 16 variables and 16,719 observations.



Continuous

- North American Sales
- European Sales
- Japan Sales
- Sales in other regions
- Global Sales

Discrete

- Year of Release
- Critic Count
- User Count

Ordinal

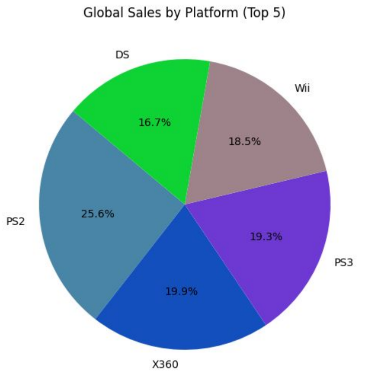
- Rating
- Critic Score
- User Score

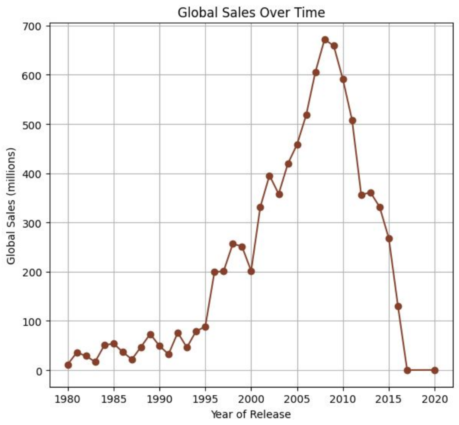
Nominal

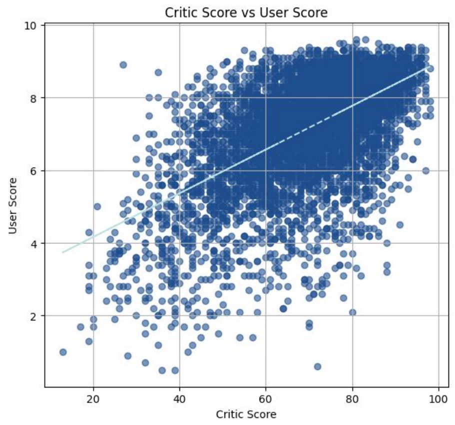
- Genre
- Publisher
- Gaming Platform
- Game Name
- Developer

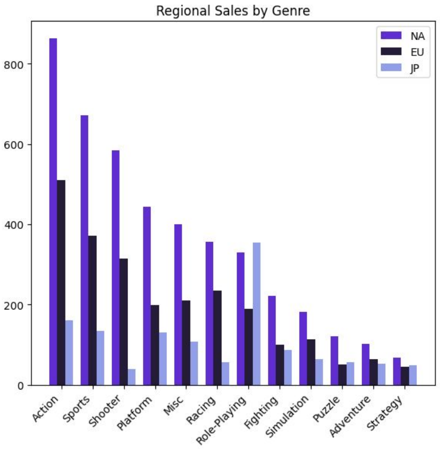
Exploratory Data Analysis

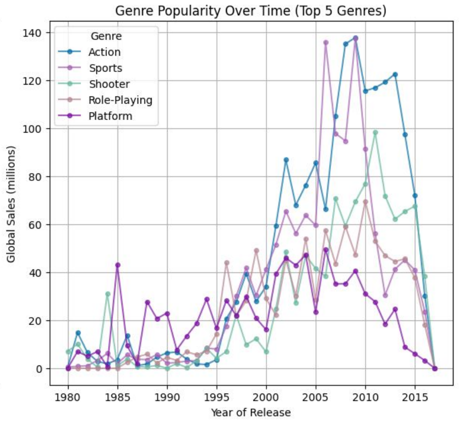
Visualizations

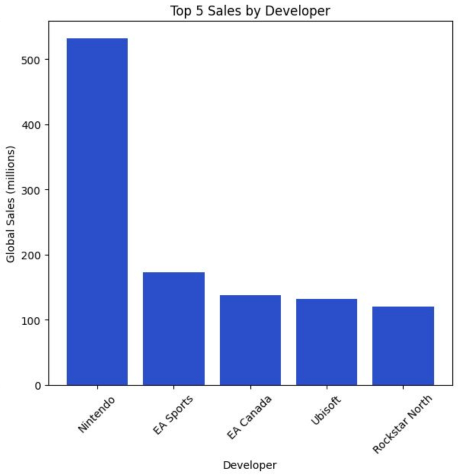






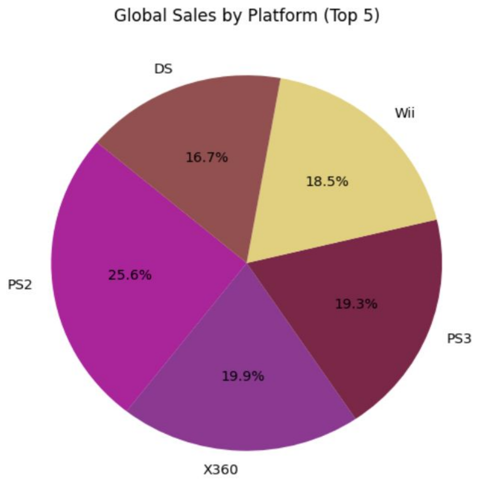






EDA – Sales by Platform and Over Time

Sales Trends: Platform & Year



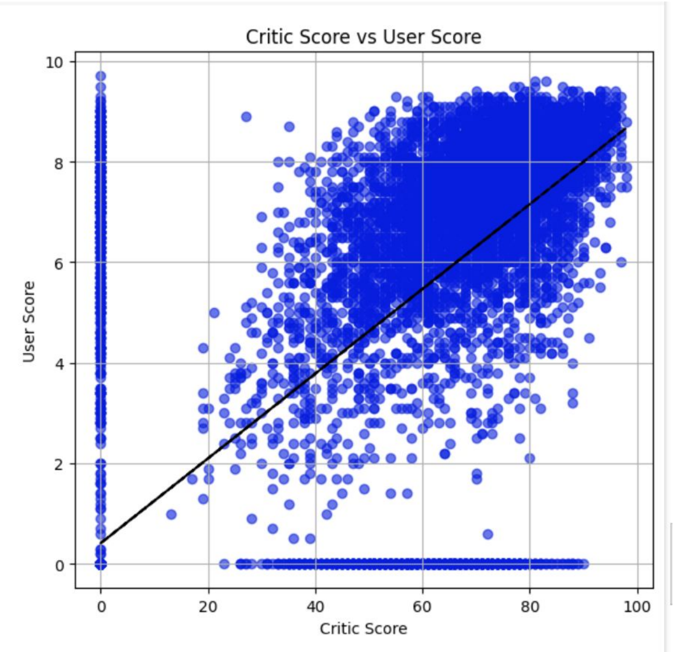
Global Sales by Platform (pie chart)

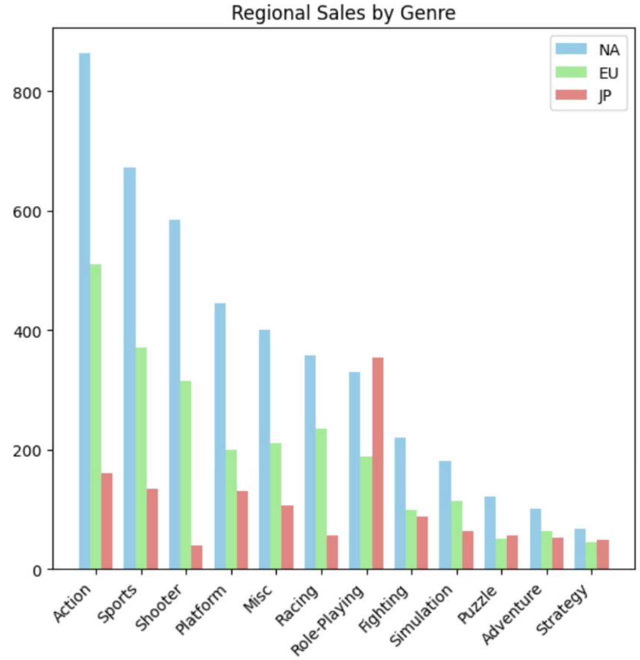
Global Sales Over Time (line chart)

This shows which platforms sold the most — PS2 leads with over 25%.
We also see sales peaked around 2008, then drop after, which might impact our model.

EDA - Critic vs User Score + Regional Sales

Scores and Regional Preferences





- Critic Score vs User Score (scatter plot)

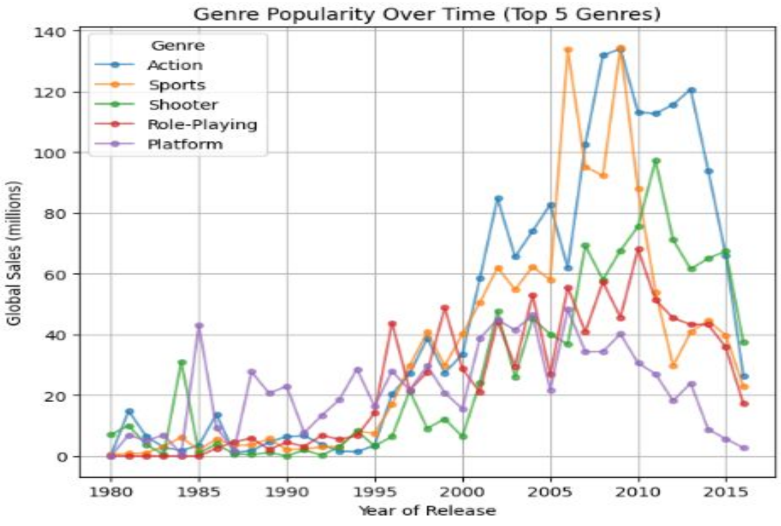
  Critic and user scores show a positive trend but not perfect agreement.

- Grouped bar chart: Regional Sales by Genre

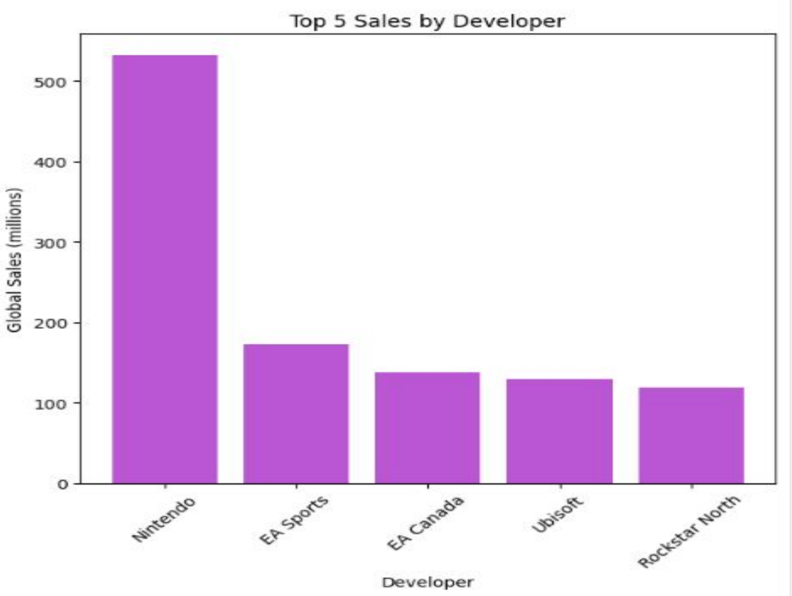
  Different regions prefer different game genres.

EDA – Top 5 Genre Trends & Developers

Genre Trends



Top Developers



Action and Sports games were top-selling genres over time.
Nintendo leads the developer rankings by far in global sales.

Summary Statistics

	Year_of_Release	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Score	Critic_Count	User_Count
count	16448.0	16448.0	16448.0	16448.0	16448.0	16448.0	7983.0	7983.0	7463.0
mean	2006.489	0.264	0.146	0.078	0.048	0.536	68.994	26.441	163.015
std	5.877	0.818	0.507	0.311	0.188	1.558	13.92	19.008	563.863
min	1980.0	0.0	0.0	0.0	0.0	0.01	13.0	3.0	4.0
25%	2003.0	0.0	0.0	0.0	0.0	0.06	60.0	12.0	10.0
50%	2007.0	0.08	0.02	0.0	0.01	0.17	71.0	22.0	24.0
75%	2010.0	0.24	0.11	0.04	0.03	0.47	79.0	36.0	81.0
max	2020.0	41.36	28.96	10.22	10.57	82.53	98.0	113.0	10665.0

Data Cleaning

```python
# Show number of rows before cleaning
rows_before = df.shape[0]

# cleaned version
rows_after = df_cleaned.shape[0]

# Print the result
print(f"Rows before cleaning: {rows_before}")
print(f"Rows after cleaning:  {rows_after}")
print(f"Rows removed:         {rows_before - rows_after}")
```

```
Rows before cleaning: 16719
Rows after cleaning:  16448
Rows removed:         271
```

- Dropped rows missing game name or release year
- Replaced 'tbd' in User Score and converted to numeric
- Filled missing Publisher and Developer with “Unknown”
- Converted Year of Release to integer
- Filled missing Rating with the most common value (mode)

Data Preprocessing

Using the label encoder library we can assign a unique integer to each value in the categorical columns. The missing values across the dataset are replaced with the mean (numerical) and mode(mode).

```python
from sklearn.preprocessing import LabelEncoder # assigns unqiue integer to each categorical column

# Separate numerical and categorical columns
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
categorical_cols = df.select_dtypes(include=['object']).columns

# 1. Replace missing values with the median
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].median())

# 2. Replace missing values with the mode
for col in categorical_cols:
    df[col] = df[col].fillna(df[col].mode()[0])

# 3. Encode categorical variables with LabelEncoder
le = LabelEncoder()
for col in categorical_cols:
    df[col] = le.fit_transform(df[col].astype(str))
df.head()
```

Models

Making the models

Linear Regression: To predict game sales based on various features such as genre, platform, and critic/user ratings.

Decision Trees: To classify games into different sales performance categories based on key attributes.

K-Means Clustering: To group similar games based on sales trends, ratings, and other characteristics, which may help identify market patterns.

—

Linear Regression

Predicting (y): **Global Sales**

Features: (x): **name, platform, year of release, genre, critic score, critic count, user score, user count, developer**, and **rating**

We used K Fold cross-validation to improve model performance, with recursive elimination feature selection.

```
MAE:  0.003
RMSE: 0.006
R²:   1.000
```

```python
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFECV
from sklearn.model_selection import KFold
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error


# Create base model
lr = LinearRegression()

# Define cross-validation strategy
cv = KFold(n_splits=10, shuffle=True, random_state=42)

# Use RFECV for automatic feature selection
rfecv = RFECV(estimator=lr, step=1, cv=cv, scoring='r2')
rfecv.fit(X_train, y_train)

# Get the best feature subset
X_train_selected = rfecv.transform(X_train)
X_test_selected = rfecv.transform(X_test)

# Train final model on selected features
lr.fit(X_train_selected, y_train)
y_pred_lr = lr.predict(X_test_selected)

# Evaluate
print("Linear Regression Results with RFECV:")
print(f"Selected {rfecv.n_features_} features: {list(X_train.columns[rfecv.support_])}")
print(f"MAE:  {mean_absolute_error(y_test, y_pred_lr):.3f}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_pred_lr)):.3f}")
print(f"R²:   {r2_score(y_test, y_pred_lr):.3f}")
```

```
Linear Regression Results with RFECV:
Selected 4 features: ['NA Sales', 'EU Sales', 'JP Sales', 'Other Sales']
```

Linear Regression w/ Selected Features (Based on Lasso)

Predicting (y): **Global Sales**

Features: (x): **na_sales, eu_sales, critic score, critic count, years since release, user count**

```
MAE:  0.120

RMSE: 0.311

R²:   0.977
```

	Feature	Importance
1	NA_Sales	1.213866
2	EU_Sales	0.743820
5	Critic_Score	0.003199
6	Critic_Count	0.001161
9	Years_Since_Release	0.000585
8	User_Count	0.000113
4	Other_Sales	0.000000
3	JP_Sales	0.000000
7	User_Score	0.000000
10	Genre_Adventure	-0.000000
18	Genre_Simulation	0.000000
11	Genre_Fighting	0.000000
12	Genre_Misc	-0.000000
13	Genre_Platform	0.000000
14	Genre_Puzzle	0.000000
15	Genre_Racing	-0.000000
16	Genre_Role-Playing	0.000000
17	Genre_Shooter	-0.000000
22	Platform_3DS	0.000000
19	Genre_Sports	-0.000000
20	Genre_Strategy	-0.000000
21	Platform_3DO	-0.000000
24	Platform_DS	0.000000
23	Platform_DC	0.000000
25	Platform_GB	0.000000
26	Platform_GBA	-0.000000
42	Platform_SCD	-0.000000
27	Platform_GC	-0.000000
28	Platform_GEN	-0.000000
29	Platform_GG	0.000000
30	Platform_N64	-0.000000
31	Platform_NES	0.000000

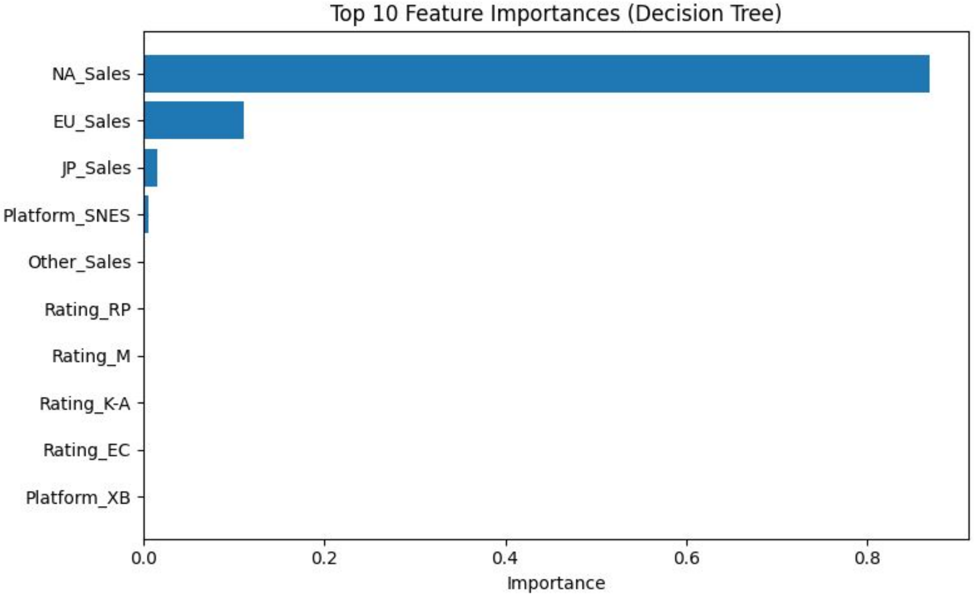
Decision Tree

Goal: Predict global sales using a tree-based model

- **MAE**: 0.161
- **RMSE**: 0.811
- **R^2**: 0.844

The decision tree model worked pretty well with an R^2 of **0.844.**
The most important feature was NA_Sales, then EU_Sales.
Other features didn't affect the results that much.

Feature importance plot

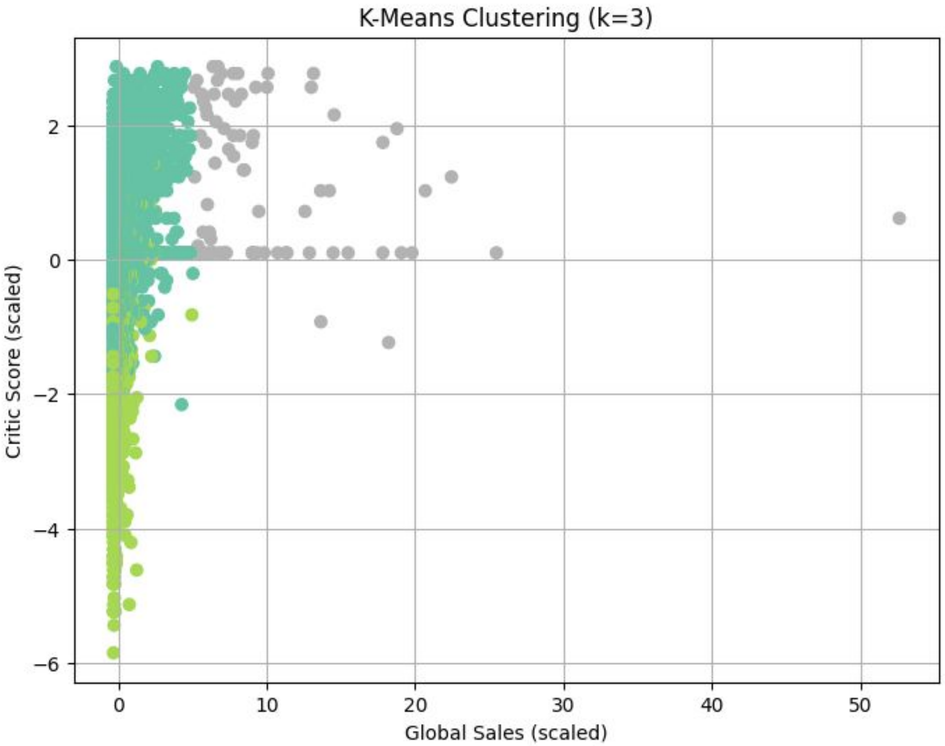


- **Top feature**: NA_Sales had the highest importance

— K Means Clustering (k = 3)

```
Games per cluster:
 Cluster
0     14249
1      2120
2        79
Name: count, dtype: int64
```

- Grouped games into 3 clusters using Global Sales, Critic Score, an User Score
- **Cluster sizes:**
  - **Cluster 0** : 14,249 games
  - **Cluster 1** : 2,120 games
  - **Cluster 2 :** 79 games

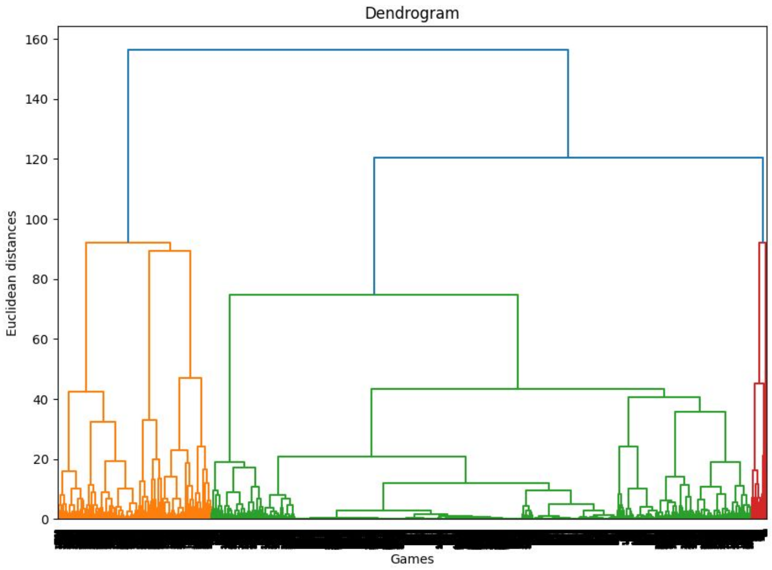


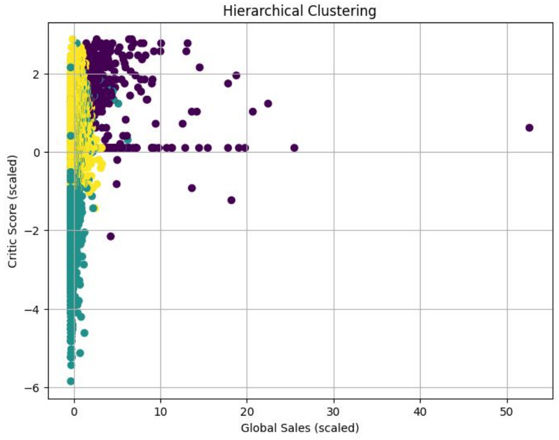
Most games had low sales and were in the same group.
Only a few games had really high sales or high scores and were in smaller groups.

Hierarchical Clustering

```
Games per cluster (Hierarchical Clustering):
 Cluster_HC
2    12535
1     3575
0      338
Name: count, dtype: int64
```

- Grouped games into 3 clusters using Global Sales, Critic Score, and User Score





Final Model & Summary

- We compared Linear Regression, Linear Regression w/ Lasso, and Decision Tree models
- Linear Regression had the best scores (very low error and R^2 near 1)
- But it may be overfitting due to strong predictors
- Linear Regression w/ Lasso seems to be better with less predictors.
- Decision Tree was simpler but still did well.
- K-means & Hierarchical was used to group games but not for prediction

model summary statistics

```python
from IPython.display import display

summary = summary.round(4)
display(summary)
```

	Model	MAE	RMSE	R²
0	Linear Regression (RFECV)	0.002962	0.005219	0.999994
1	Linear Regression (Lasso)	0.120237	0.310686	0.977103
2	Decision Tree	0.160613	0.811391	0.843832

We choose these three models because they're simple and good for understanding

Future Work

- Explore more sophisticated regression models such as polynomial regression.

- Use external data integration such as marketing or social media data to measure game popularity.

- Use real-time prediction system to estimate sales based on early data and trends.

Works Cited

Dataset: https://www.kaggle.com/datasets/xtyscut/video-games-sales-as-at-22-dec-2016csv

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An introduction to statistical learning: With applications in Python*. Springer.

McKinney, W. (2022). *Python for Data Analysis: Data wrangling with pandas, NumPy, and Jupyter*. O'Reilly Media, Inc.

McKinney, W. (2022). *Python for Data Analysis: Data wrangling with pandas, NumPy, and Jupyter*. O'Reilly Media, Inc.

Wang, W. (2025). *Principles of Machine Learning: The Three perspectives*. Springer.