ETL Report:

Regarding Marketing Analytics for Video Game Sales

Compiled by Group 6
Members:
Hugh, Jason, Ryan, and Todd

[1.1] Introduction

The goal of this project was to determine what factors were the main drivers behind video game sales on a global and regional basis. Since marketing data is usually private, our analysis was based on trends observed across publicly available sales data sourced from a web-scraped Kaggle-dataset [1]. The dataset spans over 20 years of sales records from 1995 to 2018 and features over 1,000 rows of critical reviews and categorical data (developer, publisher, ESRB-rating, and genre) with regards to the video games featured.
To get around the roadblock induced by the private nature of streamed marketing data, we simulated a data stream using Azure Databricks and Kafka. We automated the process using an Azure Datafactory, triggered at 15-minute intervals, and saved the data consumed to a SQL database. The retrieved data was then used to train a machine-learning model for predictive analysis w.r.t global sales. The flowchart below summarizes our setup:

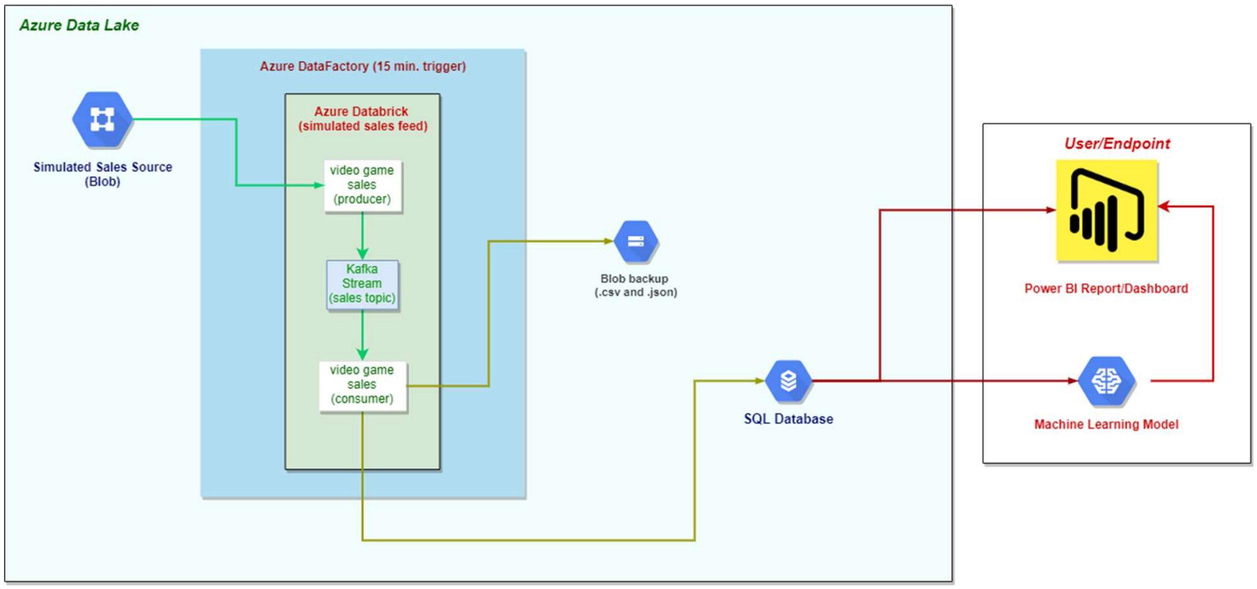

Azure Data Lake
Azure DataFactory (15 min. trigger)
Azure Databrick (simulated sales feed)
Simulated Sales Source (Blob)
video game sales (producer)
Kafka Stream (sales topic)
video game sales (consumer)
Blob backup (.csv and .json)
SQL Database
User/Endpoint
Power BI Report/Dashboard
Machine Learning Model

Figure 1: Data-Flow mediated by Kafka, from source to endpoint

[1.2] Regarding the Data:

Using the data retrieved and with some machine-learning refinement, we will attempt to answer the following questions with regards to our marketing analytics, viz.:

1. How much does a game’s review (critical/user) affect its sales?
2. Do different consoles sell more games than others?
3. How do different regions affect game sales?
4. Do certain genres sell more than others?
5. How are global sales trending over time?
6. What factors have the most influence on games sales?

Additionally, using census data [2], we will attempt to paint a broader picture w.r.t. sales trends in the US (potentially) – used on the analytics side.

[2.1] Regarding the Cloud Setup:

Our main data source [1] was saved as a .csv to a blob in an Azure Data Lake. An Azure Databrick was then set up to create the Kafka Producer and Consumer for the simulated data stream, using the manually saved CSV from the blob as the data source.
The Kafka Producer was made so that it pulled refined data from the source blob (the data was cleaned and converted to a list of dictionaries, as detailed in the next section), and pushed it to a Kafka topic, one row at a time, at 5-second intervals, simulating a live-market feed.
The Kafka Consumer then pulled from the Kafka topic, the rows that were pushed from the Kafka producer, and accumulated them in a list of dictionaries. After the Consumer had retrieved all of the data from the Producer, it saved it to an SQL Database, where it could be retrieved and used for data analysis and machine-learning (additionally, the consumed Kafka stream was saved to a blob in the data lake, as a backup).
The Databrick was encapsulated in an Azure DataFactory, to act as a pipeline for the flow from the source blob to the SQL Database (and blob backup). The DataFactory was made to trigger flows every 8 hours to simulate a real-world automated process.

[2.2] Data Pre-cleaning Performed:

The data was retrieved as a .csv from a blob in our data lake. It was extracted and cleaned in a databrick-notebook, as follows:

1. After loading the .csv into a spark data frame we noticed several columns contained mostly null-values or were not relevant for our ML predictions, so we chose to drop them, viz.: 'url', 'status', 'img_url', 'Last_Update', 'VGChartz_Score', 'VGChartzScore', 'User_Score' and 'basename'.
2. We also chose to drop rows that contained null values in any of the following columns: 'Global_Sales', 'NA_Sales', 'PAL_Sales', 'JP_Sales' and 'Other_Sales'.
3. After removing the nulls, the data was mostly cleaned and was ready to be pushed to the Kafka Topic, and consumed via the Kafka Consumer, and saved to a SQL Database.

[2.3] SQL Database Setup:

The data retrieved from the Kafka stream was initially stored as one temporary table called 'TotalShipped_Test'; it was then transferred to two tables (to achieve second-normal form) in our SQL database, called 'group6'. The following defines the schema for our 'Sales' table as well as its population from 'TotalShipped_Test':

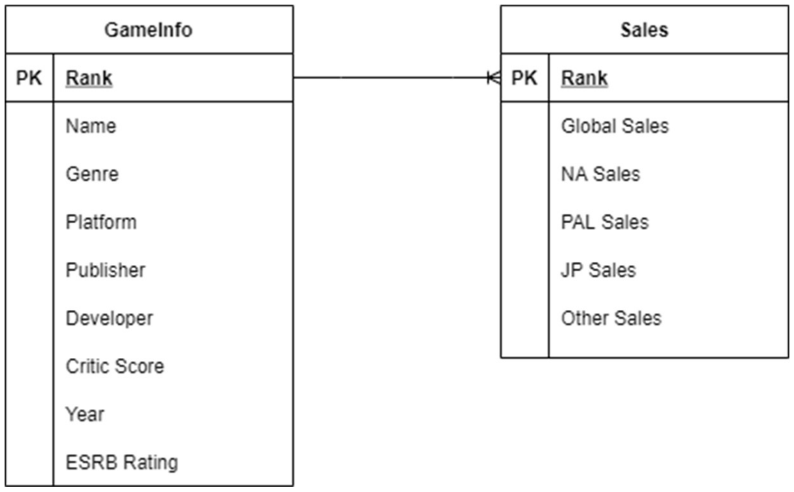
```sql
CREATE TABLE Sales(
    [Rank] int primary key,
    [Total_Shipped] [float] NULL,
    [Global_Sales] [float] NULL,
    [NA_Sales] [float] NULL,
    [PAL_Sales] [float] NULL,
    [JP_Sales] [float] NULL,
    [Other_Sales] [float] NULL
)
INSERT into Sales ([Rank], [Total_Shipped], [Global_Sales], [NA_Sales], [PAL_Sales],
 [JP_Sales], [Other_Sales])
SELECT [Rank], [Total_Shipped], [Global_Sales], [NA_Sales], [PAL_Sales], [JP_Sales],
 [Other_Sales] FROM TotalShipped_Test
```

The following defines the schema for our second table, as well as its population:

```
CREATE TABLE GameInfo (
    [Rank] int primary key,
    [Name] [nvarchar](max) NULL,
    [Genre] [nvarchar](max) NULL,
    [ESRB_Rating] [nvarchar](max) NULL,
    [Platform] [nvarchar](max) NULL,
    [Publisher] [nvarchar](max) NULL,
    [Developer] [nvarchar](max) NULL,
    [Critic_Score] [float] NULL,
    [User_Score] [float] NULL,
    [Year] [int] NULL
    CONSTRAINT FK_GameInfo_Sales
        FOREIGN KEY (Rank)
        REFERENCES Sales(Rank)
)

INSERT into GameInfo ([Rank], [Name], [Genre], [ESRB_Rating], [Platform], [Publisher
], [Developer], [Critic_Score], [User_Score], [Year])
SELECT [Rank], [Name], [Genre], [ESRB_Rating], [Platform], [Publisher], [Developer],
[Critic_Score], [User_Score], [Year] FROM TotalShipped_Test
```

The following ERD summarizes the relationship between these two tables:



[3.] Machine Learning Model Transformations:

After consuming the pre-cleaned data from Kafka, and storing it in a SQL Database. We retrieved it and prepped it for our ML model as follows (please reference the jupyter notebook for further detail):

1. We used the pyodbc library (along with our login credentials) to connect to our SQL Server and retrieve our data. We then used the pandas library to convert it to a data frame for use in our ML model.
2. Since we were interested in predicting global sales from our dataset, we removed the 'JP_Sales', 'NA_Sales', 'Other_Sales', 'PAL_Sales' and 'Rank' columns, along with the Kafka 'timestamp'.
3. Our dataset had nulls in the 'Critic_Score' column, so we used pandas dropna function to clear those out so that there weren't any nulls in any columns before we applied our ML model.
4. There were many distinct Developers and Publishers in the dataset which could be problematic for fitting to an ML model, so we used a lambda filter function to group Developers and Publishers by their prominence in the dataset; after grouping them we concatenated them back into one data frame for the ML model.
5. We were interested in seeing if the string length of the name title had any influence on its sales, so we added a column called 'NameLength' for that purpose.
6. We noticed that our global sales data was heavily skewed to the right, so to maximize the usability of our data points, we did a log-normal transformation (using the numpy library), and we created a column called 'log_Global_Sales' for that purpose.
7. Rather than destructively modifying our data frame in the prepping process for our ML model, we created a new data frame to include all the columns necessary for our ML model, viz.: 'Namelength', 'Critic_Score', 'ESRB_Rating', 'Genre', 'Platform', 'Publisher', 'Developer', 'Year', and 'log_Global_Sales'.
8. We then used the pandas '.get_dummies' function to get dummy variables for our categorical data, viz.: 'ESRB_Rating', 'Genre', 'Platform', 'Developer', and 'Publisher'.
9. We separated our target column, 'log_Global_Sales', and stored it as 'y' (storing the remaining data frame as 'x'). Then using 'train_test_split' from the sklearn library, we split the data frame for training and testing as a 70:30 ratio.
10. We then imported the GradientBoostingRegressor from sklearn.ensemble and used it as our ML model (normalization isn't necessary for this model). After performing a grid-search optimization, we used the following parameters for our predictive analysis: 'learning_rate' = 0.001, 'max_depth = 5', 'max_features' = 'sqrt', 'n_estimators' = 15,000, and a 'random_state' = 1.

Do we need a conclusion?

References:

1.) ***Video Games Sales 2019***, *"Sales and Scores for more than 55,000 games"*. Retrieved from Kaggle

2.) US Census Bureau, Retail Trade: Summary Statistics for the U.S., States, and Selected Geographies: 2017. Survey/Program: Economic Census, TableID: EC1744BASIC, Dataset: ECNBASIC2017. (directlink)