KEATECH ENTERPRISES

C964 CAPSTONE PROJECT — MDP1

VIDEO GAME SALES: TRENDS AND PREDICTIONS

A1. LETTER OF TRANSMITTAL

March 30, 2020

Paul Chapman, Project Manager
American Video Game Company
4001 700 E #700
Salt Lake City, UT 84107

Dear Mr. Chapman,

Video games have been a popular form of interactive media entertainment since the early 1980s, and many technology companies have now become multifaceted organizations with divisions solely dedicated to the publishing of video games. So too is the popularity of games highlighted by the fact that entire events are held around the world for developers to create new games, usually in the span of a few days. However, with this rise in popularity and demand comes one major downfall — where does the market's demand lie? There are twelve major genres of games recognized today, so the focus of game publishers is in the next big thing; the most cost effective solution is to produce games that there is a demonstrated need for. The only problem is that predicting the best genres to produce a game for can be time-consuming due to the vast amounts of data available.

With this, KEATech Enterprises is proposing a web application that takes video game sales data as input, and produces two primary forms of useful output: historic trends in game sales from the past through the present, and a tool that allows for sales predictions based on the desired game genre and year of release. This will benefit the American Video Game Company by allowing their sales estimations to be automated. Utilizing historic sales data, estimated future sales performance can be determined by simply choosing the genre and year instead of manually referencing past projects, either internal or those of rival publishers.

It is understandable that, as we have only worked together on a past Quality Assurance and testing project, you may be wary of the reason for this contact. Put simply, KEATech Enterprises is trying to enter the data science field and we thought it best to reach out and proffer our services for a mutually beneficial application: your team gets an analytics tool, and our team gains necessary experience in the data analytics field. Since we have very little experience in the field, you can consider our development team to have very little expertise — they have a basic understanding of the language they will use to code the application, but the additional functionalities will be new. We have two phases

of development: first, produce a prototype application for $10,000 (100 development hours); second, the maintenance and fine tuning of the application, should the first phase meet established acceptance. We estimate that for both phases, the total budget will be $32,000 (around $40,000 including reserves).

We look forward to hearing from you.

Sincerely,

CEO, KEATech Enterprises

A2. PROJECT RECOMMENDATION

Problem Summary

The current problem is that, with at least twelve major recognized genres of video games, deciding the next type of game to make can be difficult. For many companies, this likely means combing through many sales reports covering a large range of years to manually compute predictions of future game sales performance. In short, there are too many records from over thirty years of selling video games (with over one billion total sales across the world market), and computers can take this time, effort, and cost away from video game publishers and developers to make their decision easier.

This application will provide several data analysis visualizations and a prediction tool using historic video game sales data. The code will be hosted in a notebook that includes a link to a dashboard to view the code's output. Since it is hosted in a notebook, simply navigating to the link for said notebook is enough to access it in any web browser.

It will not include a way to upload a file of new or updated records — this must be performed by the development team to ensure compatibility, formatting, and proper sourcing of the data. As the application can be accessed by a web browser, there will be no support for individual stand alone desktop applications and other mobile apps; the web browser is all that is necessary.

Application Benefits

The application will take a video game sales record file and present two things. First, several graphs depicting historic trends in past sales of video games, based on genre overall (total sales), platform popularity, and genre popularity based on each year's sales. Second, a prediction tool that estimates future game sales given a genre and year of release. This will allow for quick, simplified analysis of sales trends with little human intervention and the ability to predict future performance based on those past trends (again with little human intervention).

Application Description

The application will be written in the Python programming language, which is well-known for its data analytics and computation abilities. It is this recognition that has made it a major resource in the

scientific community, especially in laboratories to process their experiment results. The Python code will be written in a Jupyter Notebook, a form of a locally hosted server that contains its own Python kernel to allow for dynamic running of code in cells, which are individual blocks that can be run separately or all at once. The kernel serves as the global memory of the notebook server, such that variables can be modified and hold their value if any cell is run. This does introduce problems, however, if data in earlier cells depends on data in later cells when the kernel must be restarted and the cells run sequentially like a traditional IDE. However, overcoming this obstacle is a simple matter of keeping the code properly organized, and the overwhelming benefit of the notebook is that, being locally hosted, it is accessed through the web browser. Further functionality comes through using the Voila library, which turns the code's output into a dashboard by hiding the code and only showing any markup and the actual code output. The web browser use and dashboard functionality allow the client to access the application from any web browser, regardless of operating system.

Data Description

The data is a comma-separated list (.csv file) of video game sales data for games that have sold over 100,000 copies total. Specific columns of the data include rank, name, platform, year, genre, publisher, and sales in millions(in North America, Europe, Japan, the rest of the world, and globally). For this purpose, rank (a number), name (the game title, a string), and sales data from all other regions aside from North America have been discarded because the American Video Game Company seeks to serve a primary audience in the North American market. Any additional sales in other regions are unintended benefits. For the major function of the application — analysis of past genre sales, and prediction of future genre sales — the independent variables are the genre and year of release, and the dependent variable is the amount of North American sales (in millions).

There are significant points in the North American sales data set. One is that around 2014, sales of all games appears to go down rapidly. We believe this to be due to the data set not being absolutely current (absolutely being used because there are data points past the year 2014, but by no means complete). Two, while video games in this data set appear to have become available starting in 1980, not all genres were represented.

For both of these reasons, we have selected to represent the specific visualization of genre performance over years using a subset of the data for the years 2000-2015. This means all genres will be included as they have been recorded during this range. Our prediction computations will also use this range so that there is agreement between the historic sales performance and predicted sales performance.

It is of note that using a slightly smaller data range of years, or employing better detection and compensation of outlier data, may be necessary to produce more accurate predictions. The above range will be an initial attempt at doing so.

Objective and Hypotheses

While the application has two goals, the primary objective will be to provide a prediction of a game genre's sales given a year of release. The hypothesis is that for a chosen video game genre, if it is released in year X, then the genre will have approximately Y millions of sales. While this can't specifically predict an actual amount of sales since our data set is based on a genre's total sales for a year (which includes a large number of individual games), it does illustrate where market demand lies. In any case, producing a game in a popular genre will yield higher sales than a less popular genre. This

application's purpose is to just analyze trends and explicitly identify the performance of a specific genre with numbers, instead of guessing.

Methodology

KEATech Enterprises is a devout supporter of the Agile development methodology, when applicable. Continuous testing of program outputs results in rapid creation of a demonstrable product that can also integrate customer feedback to shape the product into the customer's desired requirements. The shorter development periods (sprints) in the Agile methodology also serve to more concretely decompose stories into smaller, easily tracked tasks.

Using Jupyter Notebook's cell functionality allows for Agile's continuous testing, because the developer can run single or multiple cells at any time and, depending on cells run previously, the kernel stores and updates variables. Just as well, each cell can be treated like individual programs with their own import statements, or just used to represent individual tasks to be completed. While the individual cell functionality can be useful here, it does need to be reconciled by either ordering the cells properly (so that imports come first, a variable is defined before it is accessed), or by eventually moving code to earlier cells because at some point, the kernel will have to be reset. Once it is, all data saved from previous runs is cleared and the entire program must be run sequentially, and earlier or later cells can then throw errors if they were using information defined somewhere else.

Though Agile has six phases, they can all blend together, as is often the case on many projects. From planning to analysis, design to implementation, then testing and integration to maintenance, it is often more natural to follow this process (or at least the first five) for each task, and continue to follow the cycle as the tasks join together into larger components and eventually a whole software product.

Funding Requirements

All necessary software tools (Python, Jupyter Notebook, and Voila) are free and/or open source. As stated in our letter of transmittal, the total final budget required depends on our two phases of development. The first phase will be $10,000 for the prototype application (100 hours of development time). Should the project proceed, we estimate the total final cost (the above cost already included) of the software product to be $32,000 (around $40,000 including reserves). For complete transparency, this cost is based on our estimate of 320 developer hours (four developers working a total of eighty hours each) at a rate of $100 per hour of development time. This includes all later maintenance costs to get the application up to specification. If development does not proceed, the total final cost will be $10,000.

Stakeholders Impact

Our stakeholders fully support the success of this project as they see the data science field as lucrative and desire KEATech to enter the space.

The American Video Game Company's stakeholders should support this project as success means the company will have more guidance to video game popularity trends. That is, the company will get automated guidance as to which video game genres are the most popular and therefore the most

likely to yield the greatest sales. More sales for the company typically mean a higher return on stakeholder investments (or at least on stockholder and employee investments).

Data Precautions

Fortunately, none of the data in our data set should be sensitive as it is publicly available sales data — it doesn't involve any human element (per HIPAA and FERPA regulations), has no financial transaction connections (per PCI DSS regulations), and though publisher and platform names may be sensitive they aren't copyrighted/trademarked against usage for statistical purposes (versus using those trademarks in, say, another video game without permission). The other point of data potentially sensitive would be the title of the games, but that will removed from the used data set as it doesn't complete any of our stated goals.

Developer's Expertise

The developers that will be assigned to this project are recent college graduates that have some experience in the Python language, but none in many of the other tools and libraries they will be using. They have also done some development in the Java language. Their educational performance shows quick turnaround times that have equated to devotion and attention to detail; projects are often completed faster than quoted as they dedicate themselves to solving problems. There is no question that, despite the new skills required to complete this project, they will meet the challenge.

Again, this project is seen as a learning experience for both them and KEATech, and we all have the utmost confidence that, even in the case the developers need assistance, we will be able to find the answers necessary to produce a viable software product.

B. PROJECT PROPOSAL

Problem Statement

All video game companies (developers and publishers) face the same problem: they want to be successful. The best way to do that is to meet consumer demands (trends), but right now, most companies use manual analysis of past sales or rough estimates of what is popular at the time. While that can be effective, manual analysis takes time and large amounts of information; add to that the cost of a trained and experienced data scientist that can manipulate the data properly in order to show trends. Just as well, video games take time to develop, so basing decisions off of what is currently popular only works within a very short range of time (think fads).

Our proposed application will perform both the analysis of past trends and the prediction of future performance in one package — without the recurring time and cost requirements or the dangers of releasing a product too late. Through Python's Pandas, Numpy, and Scikit libraries, KEATech will be able to provide a web-based dashboard that can parse video game sales data starting in 1980, and later include data beyond 2015, to present visualizations of past sales trends to the user. Further, by performing linear regression calculations on that past data, future predictions can be made about a genre's potential total sales. Forget the SQL database and spreadsheets, everything will be managed in one convenient place. Even the code should be able to be quickly changed to: point to the newer data files, set new date ranges for calculations, include new worldwide region sales data, and add/remove/modify graphs.

Customer Summary

Ideally, the application will be used by the management teams that green light projects. The prediction tool will give them just a little more insight into what the next project ought to be. On the other hand, all the employees of the company may find it enlightening to examine genre sales trends throughout the years. Some data may even correlate to particular events and releases in the gaming industry — the historical charts can reflect that data.

Since all the calculations will done in the software, no special skills will be necessary. It can be as simple as clicking a couple of buttons to get to the dashboard, and all the information is displayed. Additionally, for any user input, dropdown boxes with a predefined set of inputs will be used so as to minimize any chance of invalid inputs that cause program errors.

Existing System Analysis

Using the Jupyter Notebooks, Binder hosting, and Voila dashboard systems, there is no overhead and nothing required of the client. Underlying code is hosted on GitHub, and can be easily modified by the developer to make client-requested changes, if necessary. The client need only computers with current web browsers, nothing proprietary or unique, just standard equipment.

Data

The data is a comma-separated values text file (.csv file) of video game sales data taken from the website VGchartz.com (more specifically, this scraped dataset was made available from the following link: https://www.kaggle.com/gregorut/videogamesales/version/2). It includes the following columns of data for video games that have sold over 100,000 copies total (worldwide) between the years of 1980 and 2016: rank, name, platform, year, genre, publisher, and sales in millions(in North America, Europe, Japan, the rest of the world, and globally).

To limit the scope of the data, rank (a number), name (the game title, a string), and sales data from all other regions aside from North America will be discarded because the American Video Game Company seeks to serve a primary audience in the North American market. Any additional sales in other regions are unintended benefits. For the major function of the application — analysis of past genre sales, and prediction of future genre sales — the independent variables will be the genre and year of release, and the dependent variable will be the amount of North American sales (in millions).

Despite this data cleaning there may still be discrepancies. Any entries missing the year of release or amount of sales will be discarded. Further, there are two anomalies that we notice in the data. One is that around 2014, sales of all genres of games appear to go down rapidly. We believe this to be due to the data set not being absolutely current (absolutely being used because there are data points past the year 2014, but the set is by no means complete). Two, while video games in this data set appear to have become available starting in 1980, not all genres were represented. For both of these reasons, will represent the specific visualization of genre performance over the years by using a subset of the data from the years 2000-2015. This means all genres will be included as they have been recorded during this range. Our prediction computations will also use this range so that there is agreement between the historic sales performance and predicted sales performance.

It should be noted that by using a slightly smaller data range of years (where the range of years ends before 2015), or employing better detection and compensation of outlier data, may be necessary to produce more accurate predictions. The above range will be an initial attempt at doing so. It is our initial assessment that sales values in 2015 are still within acceptable limits, and so that will be our first ending range.

Project Methodology

The project will be developed in accordance with the Agile development methodology. We feel that the continuous testing and feedback best serves us and the customer in order to ensure value co-creation. The following is a more detailed overview of our workflow process.

Requirements:

- We will work with the client to define the software requirements as clearly and concisely as possible. The less open to interpretation a requirement is, the greater chance it will be properly met.
- Requirements should be gathered from not just the client, but also from all relevant stakeholders.

Development:

- Using the requirements, we will create plans (backlogs) to follow during sprints, including the stories and further decomposed tasks.
- Actual implementation of the project will begin.

Testing:

- During development, we will be performing continuous unit testing of the code, both black box (from the user's point of view) and white box (from the coder's point of view), to ensure the project meets its functionality requirements.

Delivery:

- The project will be delivered to the client for acceptance testing and their team will be instructed how to use the application.

Feedback:

- During delivery and user testing, additional feedback will be gathered for any future changes to make to the application.
- The cycle will repeat as necessary until the application meets all necessary requirements such that it will be considered complete by the client.

Project Outcomes

Project deliverables include a schedule of both phases of development, showing how the development hours are allocated, as well as the dataset itself should the client wish to personally examine it with other tools, and rough sketches of the charts we are proposing to display on the product dashboard. Budget allocation was highlighted in previous sections, and is only based on development hours, therefore the development schedule should be sufficient.

Product deliverables include the Jupyter Notebook containing the Python code necessary to create the dashboard with all the required information and any other created files that will be stored in our company's repository until officially deployed, after which the client may leave it under our responsibility or can take ownership of it.

Implementation Plan

The plan for implementation is very simple and occurs very similarly for the two previously defined phases (phase one: prototyped working application, phase two: refined application). Upon reaching the end of either phase, where the application is considered complete, the application and its associated deliverables (the project deliverables of schedule, dataset, and chart sketches; the product deliverables of all files created or used for the application, including the code itself) will be made available to the client. If they so desire, ownership of the project repository will also be transferred after each phase (phase one only if work is to be halted, phase two if the client wants sole ownership).

Upon delivery, all relevant users, including hallway testers, may have access to the application to execute acceptance and usability testing, and for KEATech Enterprises to gather feedback for further improvements, if necessary. As the application is web-based, there should be very few compatibility issues as the systems analysis already returned that the only required systems were computers and an up-to-date web browser. There should not be any need for stages of rollout, as the intention is for a small number of users to access the dashboard for analysis and prediction (this isn't a software solution that an entire company must use to complete their tasks). Fortunately, as users will not be making changes to the code and the dataset is static, the application will allow multiple users access to the same data on individual dashboards simultaneously.

Any further updates can be push or pull deployment based on the client's discretion. Each of the two phases of the application is considered working and usable, therefore the client can decide whether they wish for the application to update immediately or if they must initiate the update themselves. One

thing of note is that by using Binder, if the developer updates the repository containing the notebook files, the server hosting the application will automatically update when navigating to the Binder link.

Evaluation Plan

It is our thought that by examining the subsets of data we will be using to generate visualizations, we can verify whether the data has been properly processed, or at least coded to display the correct variables in the proper locations. With the application being simple, we feel that the aforementioned unit testing during development, and usability and acceptance testing by the client at delivery (with feedback gathering), will be sufficient.

The project will make no use of sensitive data following established regulatory policies.

The dynamically generated aspects (graphs, predictions) should have a response time less than three seconds, on average, after changing the input data.

Any organizational policies or accessibility policies will be applied as necessary during the second phase of the development cycle.

The initial goal of testing will be to uncover as many errors as possible, and will be deemed complete after executing all possible ranges of input. The first development phase may contain errors that will likely be dependent upon the range of data used. Formal error analysis will be provided with the predictions. Initially, we will provide three types of calculated error values: x, y, and z. These can be later hidden or removed for official deployment after phase two.

Resources and Costs

- Programming Environment: Python, additional libraries, Jupyter Notebook, Binder, and Voila. All will run on a Windows or Mac computer that can install them, and requires a web browser. KEATech already has computers with updated browsers, and has installed and tested the necessary software.
  - Cost: $0
- Environment Costs: all hardware was already in ownership. Additional costs include utilities and real estate, which are negligible to the client.
  - Cost: $0
- Human Resource Requirements: we are basing our cost estimate entirely on development hours.
  - Cost: either $10,000 or $32,000 total, depending on client acceptance.
    - Phase One (working prototype, 100 development hours): $10,000
    - Phase Two (finished application, 320 total hours): $32,000 ($40,000 with reserves)

Timeline and Milestones

Event	Start Date	End Date	Duration (business days)	Dependencies	Resources Assigned
1. Project Start	3-23-2020	3-23-2020	0	NA	Stakeholders
2. Phase One	3-23-2020	3-30-2020	5	Task 1	Developers

Event	Start Date	End Date	Duration (business days)	Dependencies	Resources Assigned
3. Phase One Complete	3-30-2020	3-30-2020	0	Task 2	Stakeholders
4. Phase One Deployment/Feedback Gathering	3-31-2020	4-3-2020	4	Task 3	Developers, Client, Users
5. Phase Two	4-7-2020	4-21-2020	10	Task 4	Developers
6. Phase Two Complete	4-21-2020	4-21-2020	0	Task 5	Stakeholders
7. Phase Two Deployment/Feedback Gathering	4-22-2020	4-28-2020	4	Task 6	Developers, Client, Users
8. Final Testing and Modification	4-29-2020	5-4-2020	4	Task 7	Developers, Client, Users
9. Project Finish	5-1-2020	5-1-2020	0	Task 8	Stakeholders

Note: Phase Complete means ready to deploy for user testing and feedback gathering.

D. PHASE ONE POST-IMPLEMENTATION REPORT

Project Purpose

The problem we wanted to address was that, with at least twelve major recognized genres of video games, game developers and publishers require an investment of effort to decide the next type of game to make. That investment usually means combing through many sales reports covering a large range of years to manually compute predictions of future game sales performance. In short, there are too many records from over thirty years of selling video games (with over one billion total sales across the world market), and computers can take this time, effort, and cost away from video game publishers and developers to make their decision easier.

The provided application at deployment Phase One provided four data analysis visualizations and a prediction tool that use historic video game sales data. It performed both the analysis of past trends and the prediction of future performance in one package — without the recurring time and cost requirements or the potential error that comes with those manual calculations, which would resulting in incorrect estimates and eventually releasing a game too late. The prediction was calculated by performing linear regression calculations on that past data, yielding a line on a pretend graph that extends over many years. By simply choosing a genre and a year, the prediction returned the amount of sales by following that genre's invisible line. By integrating these functions into one convenient package, there was no longer a need for the video game companies to use SQL databases and spreadsheets for their data analysis.

The code was hosted in an interactive Python web-server based development environment called a notebook, and the notebook included a link to a dashboard to view the code's output. Since it was hosted in a notebook, the only thing necessary to access the code and dashboard was an up-to-date web browser. Additionally, the code could be quickly changed to: point to the newer data files, set new date ranges for calculations, include new worldwide region sales data, and add/remove/modify graphs by updating the code files or data file in the repository (the place hosting the files) that contained the project.

Datasets

The data used is a comma-separated values text file (.csv file) of video game sales data taken from the website VGchartz.com (the raw dataset can be viewed from its source: https://www.kaggle.com/gregorut/videogamesales/version/2, or it can be viewed at the repository for the project, provided in the below section Application Files or through the link to the repository: https://github.com/techguy682/vgsalesprojectv2). The original dataset included the following columns of data for video games that have sold over 100,000 copies total (worldwide) between the years of 1980 and 2016: rank, name, platform, year, genre, publisher, and sales in millions(in North America, Europe, Japan, the rest of the world, and globally).

To clean the data, rank (a number), name (the game title, a string), and sales data from regions other than North America were discarded because the American Video Game Company sought to serve the North American gamer market.

There were some significant points in the North American sales data set. One was that around 2014, sales of all games appeared to go down rapidly. We believed this to be due to the data set not being absolutely current (absolutely being used because there were data points past the year 2014, but

that more recent data was not accurately reflective of the proper amount of sales). Two, while video games in this data set became available starting in 1980, not all genres were represented.

For both of these reasons, we represented the specific visualization of genre performance over years using a subset of the data for the years 2000-2015. This meant all genres were included as they had a recorded amount of sales within that range. Our prediction computations also used this range to keep results between historic and predicted sales performance consistent.

We also thought it important that by using a slightly smaller data range of years, or employing better detection and compensation of outlier data, we could produce more accurate predictions. The 2000-2015 date range was our initial thought, and until the data is updated, that range was not very effective as some sales predictions returns values nearing zero far earlier than they should have.

The following figures provide examples of the dataset and how it was cleaned for more simple and efficient usage.

Figure D1: an example of the raw .csv file.

```
Rank,Name,Platform,Year,Genre,Publisher,NA_Sales,EU_Sales,JP_Sales,Other_Sales,Global_Sales
1,Wii Sports,Wii,2006,Sports,Nintendo,41.49,29.02,3.77,8.46,82.74
2,Super Mario Bros.,NES,1985,Platform,Nintendo,29.08,3.58,6.81,0.77,40.24
3,Mario Kart Wii,Wii,2008,Racing,Nintendo,15.85,12.88,3.79,3.31,35.82
4,Wii Sports Resort,Wii,2009,Sports,Nintendo,15.75,11.01,3.28,2.96,33
5,Pokemon Red/Pokemon Blue,GB,1996,Role-Playing,Nintendo,11.27,8.89,10.22,1,31.37
6,Tetris,GB,1989,Puzzle,Nintendo,23.2,2.26,4.22,0.58,30.26
7,New Super Mario Bros.,DS,2006,Platform,Nintendo,11.38,9.23,6.5,2.9,30.01
8,Wii Play,Wii,2006,Misc,Nintendo,14.03,9.2,2.93,2.85,29.02
9,New Super Mario Bros. Wii,Wii,2009,Platform,Nintendo,14.59,7.06,4.7,2.26,28.62
10,Duck Hunt,NES,1984,Shooter,Nintendo,26.93,0.63,0.28,0.47,28.31
```

Figure D2: an example of the code used to clean the dataset by removing unnecessary rows, as well as limiting the data by the date range of 2000-2015.

```
# Creates a dataframe specifically for NA sales. Automatically converts year back to Int64.
# This should make it much easier to create charts as there is only one type of sales
# to be dependent on any other variables.
df_NAsales = pd.read_csv(pathname,
                         header=0,
                         names=['Rank', 'Name', 'Platform', 'Year', 'Genre', 'Publisher',
                        'NA_Sales','EU_Sales','JP_Sales','Other_Sales','Global_Sales'],
                        usecols=['Platform','Year','Genre','Publisher','NA_Sales'],
                        dtype={'Year':'Int64'})

# Data sets for each genre, using years from 2000-2015.
# First, clean the data and select rows with non-null values inclusively between 2000 and 2015.
df_2000_2015_NAsales=df_NAsales[(df_NAsales['Year'] >= 2000) & (df_NAsales['Year'] <= 2015) & (df_
NAsales['Year'].notnull())]
```

Data Product Code

In order to yield the data required for historical sales trend analysis, the cleaned North American sales data was further split into three main groups of data. First, the complete set of North American

sales from the earliest year (around 1980) to the latest year (around 2016). Second, the subset of North American sales that were recorded between the years of 2000-2015. Third, we created twelve subsets of the 2000-2015 data, one for each genre. From this point, in order to get each genre subset into the proper form for plotting and creating predictions, each of those subsets were grouped by their year value. This allowed for the creation of sums of a genre's sales for each year recorded.

By creating those specific subsets of data, we were able to create graphs tailored specifically to the trends we sought to convey. All of the descriptive functions (represented through the visualizations) use simple math functions based on a specific subset of data. See the below section Effective Visualizations and Reporting for more information. In short, each plot used data grouped by a particular column, and the sales were calculated by summing up the values in the sales for the grouped column (the Sales vs Platforms, Sales vs Genres, and Sales vs Year for each genre).

The third subset that grouped genres, and allowed for the creation of the Sales vs Year for each genre visualization, was also used in the linear regression equation for the genre and year sales prediction tool. Each individual genre's data points were provided to an equation that calculated the slope coefficient and y-intercept for a line of best fit. That is, this equation created a flat, invisible line that best represented the average of that genre's plotted line (it followed the positive or negative trend that the data established). Additionally, some error analysis equations were provided to show the user the margin of error in the calculations.

The source code is available in the submission and at the repository indicated below in the section Application Files, the specific file C964_Project.ipynb.

Hypothesis Verification

Our initial hypothesis stated that for a chosen video game genre, if it is released in year X, then the genre will have approximately Y millions of sales. While it isn't possible to verify the actual performance of our prediction (especially considering all of the unknown variables), we believed our hypothesis to be true. We set out to build an application that could give a user a predicted amount of game sales for a chosen genre for a chosen year, and it does just that.

Some unfortunate side effects came with the data not being high quality and the initial date range potentially including outlier data. In either case, the resulting predictions appeared to skew more negative than they should have, where some predictions are showing zero or even negative amounts of sales for some genre and year combinations. This is not an error per se, but an artifact of the linear regression equation that simply creates a line to return a y-value (sales) given an x-value (year).

Effective Visualizations and Reporting

The three visualizations we chose to present were Sales vs Platforms (all years), Sales vs Genres (all years), and Sales vs Year (between 2000-2015) for each genre. We believed each graph told a compelling story.

The total platform sales showed the amazing difference between the amount of video games sold on each of those platforms. Some particular points to note were that despite some platforms existing

longer than others, the best performers were released within the last twenty years (around 2000-2020). It also showed that some platforms may attribute their success to platform-exclusive games, such as on Sony's PlayStation 2 (PS2) and PlayStation 3 (PS3), Microsoft's Xbox 360 (X360), and Nintendo's DS (DS) and Wii (Wii). Each of those platforms alone sold around 400 million or more games.

Just as interesting is the genre sales for all time. It showed that by far, action, sports, shooter, and platform genre games were the most popular, with each selling over 400 million games. This is a trend that certainly appears to hold true.

Finally, the genre sales for each year, which provided accurate historical analysis, but due to our data and data ranges affected some of the predictions for future sales. Besides the predictions, this graph, and its custom genre sales chart that allowed a user to see a single genre plotted on a dynamically generated graph, provided insight into the video game market. As games take years to develop, some of the more successful games (particularly series) are released about once every two years. This is reflected in the graph as there are multiple peaks and valleys (high and low spots) on each genre's plot; for instance, it could be inferred that a major (and successful) action game was released in 2002, another one between 2004-2005, certainly again in 2008 (with moderately successful releases on either side of that year), and another peak in 2013. There are others way to interpret this — the market may have been saturated with more action games than normal in a particular year, or games may have been released but they were so poorly received that sales during that year plummeted; world events can also explain certain trends, such as sales of all video games almost doubling after the year 2000, when the Internet started becoming ubiquitous and speeds starting to increase (allowing for downloading of games and online play). In any case, such speculations are out of the scope of this document, but do serve to highlight what we tried to capture through the data visualizations.

Accuracy Analysis

As discussed above regarding the data visualization and in our previous reported correspondence with the client, the focus of this application was very limited and was never meant to serve in any highly professional scenario. KEATech Enterprises sought to gain a foothold in the data science field, and we feel that we did just that. Our work here is still not done — there is still Phase Two and the official deployment and release for the American Video Game Company. Improvements are planned, and better data will be used that includes a wider range of data, be it global sales or more current sales data.

The descriptive graphs definitely served their purpose of telling the story of platform and genre popularity in the video game industry. However, it's our same starting data that also skewed the predictive function's results and gave pretty fair percentages of error. After Phase One was deployed, we stood by both data methods, acknowledging the necessary areas of improvement. In the end, the application visualized trends in platforms and genres for all time (which we felt to be extremely representative), and an informative graph of a genre's trend over a range of years. The genre sales prediction, while it needed work to utilize a more accurate regression equation (line of best fit, which we could get by disregarding outlier data), it still served its purpose of providing a prediction (and a proof of concept for the prototype).

Application Testing

Phase One required very little testing. During development, constant unit, usability, and acceptance testing was performed. The Python language in Jupyter Notebook made it easy to constantly

make changes to the code and retest if the code returned an error or the output was not as expected. An example where this was helpful occurred when creating the genre sales over years graphs. The first implementation had sale counts that were inaccurate. The general graph shape was still the same, however, the sales values were completely wrong. After verifying the data by checking the values we wanted to plot, the problem was found to be the use of an incorrect math function — the values were being counted instead of summed. This meant that if there were three values, they were being counted as three, instead of taking the desired sum of each of those three values. Through the quick testing, the function was changed from count to sum, and the graph produced the correct values.

There were no explicit tests written as there were very few types of input or variables that could go wrong once passed into the user's hands. Brute-force testing was sufficient to test the customized graph and the future sales predictions. Errors we noted and intended to fix at this point were the presence of zero or negative sales numbers for some genre and year combinations. As discussed, this is an artifact of the dataset we used and the idiosyncrasies further introduced by performing linear regression on that data.

Further testing included usability and acceptance testing by the client and any other users (including hallway testers) to ensure ease of use and understanding of the data being presented. The American Video Game Company found the prototype to be extremely user-friendly in both usability and understanding. Acceptance included the presence of descriptive analysis, predictive statistics, and a dashboard to access the analysis and statistics. Both were successful.

Application Files

All of the necessary files for the Phase One deployment can be found at the following GitHub repository link: https://github.com/techguy682/vgsalesprojectv2. These files were also provided in the submission. However, the project is intended to be deployed through Binder, which is linked in the GitHub repository readme (the Binder link is automatically displayed in the readme on the GitHub repository page).

This repository includes:

- C964_Project.ipynb: A copy of the Jupyter Notebook Python file
- README.md: A readme containing instructions for how to access the notebook (with the Python source code) and to bring up the dashboard with the data visualizations and prediction
- environment.yml: A text file containing information necessary for Binder to properly build and host the notebook
- vgsales.csv: A copy of the original dataset containing the video game sales data.

For further assistance, see the section User's Guide below.

User's Guide

For evaluators and users of the application:

1. Ensure you have a computer and an up-to-date browser.
2. **(Note: this step can be skipped if you wish to go straight to the application. This link takes you to the repository containing all the project files and also provides a link to the application.)** Navigate to the following webpage by either clicking the link or copying

and pasting it into the address bar of your web browser: https://github.com/techguy682/vgsalesprojectv2

3. Scroll down on the page to view the readme. On it is another link to click, or a button below it that says 'launch binder.' There are also instructions on what to do after clicking that button. The link will launch the Jupyter Notebook that contains the application. Here is the link to click or enter into the address bar of your web browser: https://mybinder.org/v2/gh/techguy682/vgsalesprojectv2/master?filepath=C964_Project.ipynb
4. Allow a moment for the notebook to load. This screen is the source code for the application.
5. When loaded, there is a button on the toolbar labeled 'Voila.' Click that to open the dashboard. **Having trouble? See below.**
6. When the dashboard opens, explore the visualizations, and pay special attention to the heading 'Custom Genre Sales Chart,' which has a dropdown box below it to dynamically generate a chosen genre's sales performance between 2000-2015. At the bottom of the page is the predictive function under the heading 'Potential (Predictive) Sales Trends,' where the genre and year can be selected using dropdown boxes, and the estimated sales will be presented, as well as an error analysis of that prediction. Each error analysis is specific to the genre (it only changes with the genre, and not the year).

If Voila is not loading, there are a few things to try after the notebook loads:

1. On the toolbar, click Kernel > Restart and Run All. Wait for the process to finish. Try to click Voila again. Continue with step 6.
2. If the above doesn't work, check your browser settings to allow pop-ups on the website (or first check if your browser blocked a pop-up, and allow it to open). In the browser settings, the website's title should look something like 'hub.gke.mybinder.org.' Allow pop-ups on this site, accept the changes, then reload the webpage and try to click Voila again.

Advanced: If you wish to run the application the same way it was developed, on your local machine:

1. Download Anaconda (conda) by going to this page and following the directions for your operating system (I used MacOS and installed Anaconda, not Miniconda, though either should supposedly work): https://docs.conda.io/projects/conda/en/latest/user-guide/install/
2. To launch Jupyter Notebook (which comes with Anaconda), open a terminal or command line and type 'jupyter notebook' which will start the local web server and open a webpage with your computer's file directory. You can open my project file by locating C964_Project.ipynb and clicking on it.
3. Note: The instructions I followed were on the Jupyter Documentation page here ('Installing Jupyter using Anaconda and conda'): https://jupyter.readthedocs.io/en/latest/install.html
4. Lastly, to install Voila, start a fresh terminal and enter 'conda install -c condo-forge voila'.

Summation of Learning Experience

If there is one thing I have learned in my lifelong learning experience so far, and especially true regarding higher education, is that school, and more generally learning, never truly ends. A degree doesn't only represent the graduate's knowledge in a particular subject, but it shows that they know how to adapt to challenges and seek resources when necessary.

This was especially true for this project. The Computer Science Capstone is intended to synthesize all of the disciplines students have gained experience with through the degree program. Before starting, I had an understanding of the fundamentals of programming (which are often much more important than knowledge of a specific language because they usually apply to all languages) and mathematical concepts (which helped with figuring out the descriptive and predictive elements to include in this project). I had also done a couple projects in Java and one in Python, where I created a much more traditional stand-alone Python application using the PyCharm IDE (traditional because I wrote it as separate modules, as I would for object-oriented programming in Java) because it was what I had experience with. Thankfully that project introduced me to the syntax of Python else the capstone may have taken even longer than it did. That being said, this project was so open to interpretation that I found myself floundering to first find a project idea, then how I was going to accomplish that proposed project in software. Enter my skills in seeking knowledge. The Internet, today's version of the library, was serving up official documentation to guide my usage of these new libraries (previously not allowed on other projects) and other user answers to the same problems I was having. Just as well, some ideas for how to deploy the project from past students served me well — I had no idea Jupyter Notebooks, Binder, and Voila even existed before this. Even with my basic knowledge of Python, to me it was still the language that research and data scientists use to analyze data. In a way, truer words have never been spoken and this project has reaffirmed that definition, with the caveat that it is also able to perform simple tasks, too. Then again, nearly all languages can have similar capabilities given the proper implementation and libraries.

In conclusion, this project and eventual completion of the degree program only signify the end of learning if I make it so. So many people treat the end of their school time as the end of learning, but it doesn't need to be so. I will treat my degree as a sign that I not only have experience in the Computer Science field, but that I know how to seek information when I need it, and will continue to learn new and necessary skills in my future endeavors.

E. SOURCES

References

Wesner, J., 2016. MAE And RMSE — Which Metric Is Better?. [online] Medium. Available at: <https:// medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d> [Accessed 4 April 2020].