Capstone Project 1: Path of Exile Economic Analysis

Overview:

Path of Exile is a free-to-play Action RPG (i.e. Diablo 2/3 etc.) that utilizes player-to-player trading within their economy. However, as there is a distinct lack of any single monetary currency or a trading method (such as an auction house) within the game, associated worth of individual items is instead tied to a set of currency items that have rapid fluctuations throughout the league that depend on a multitude of factors. For this capstone project, I will analyze the time-series data sets aggregated from previous 3-month leagues (Incursion, Bestiary, Abyss, Harbinger, Legacy, Breach) that include both currency item values and unique item values to build a model that can be used in future leagues for calculated investments within the game economy.

Target Audience:

With recent league launches of over 100,000 unique players, there are many people interested in growing their personal wealth within the game. However, due to the complex nature of this game, an ongoing issue for many players is that they shy away from making investments due to number of variables needed to consider to accurately price their individual loot found. Along with these players, there are also many players who use the game as a way to flip currency and high-priced unique items for the sole purpose of Real Money Trading (RMT) aka the act of selling in game currency for real world money. (Note: This is illegal within the game's terms of service, but realistically has not been stopped). My goal is to provide an easy-to-understand model that all players can use to overcome the steep learning curve of the game in hopes of maximizing their personal profits.

Data Acquisition:

The data collected for this project was taken from Poe.Ninja's datadump section of their website (https://poe.ninja/data). The following challenge leagues' currency and item csv files were downloaded:

- Incursion (June 1st, 2018 through August 28th, 2018)
- Bestiary (March 2nd, 2018 through May 28th, 2018)
- Abyss (December 8th 2017 through March 1st, 2018)
- Harbinger (August 4th, 2017 through December 4th, 2017)
- Legacy (March 3rd, 2017 through July 31st, 2017)
- Breach (December 2nd, 2016 through February 28th, 2017)

I focused only on the different temporary league's economies (vs the permanent leagues) for two major reasons:

1. The lifespan of a temporary league is about 3 months from start to finish. This fact means that I see much more volatile (and therefore interesting) changes in the economy. This also allows us to compare and contrast the different leagues against each other to draw conclusions I might not have seen otherwise.
2. Challenge leagues are always accompanied by the newest mechanics implemented in the game. This provides us with a fresh set of items/currencies that I can conduct our analysis on that makes for a more interesting project. Sometimes these changes are brought into the base version of the game, sometimes they are not. I will keep this in mind as I move forward.

Now that the files have been collected, I noted that I have two distinct types of files (currencies vs unique items), and so I started my data wrangling with the more complicated item files. (Note: For the first half of this project, I'm also choosing to only utilize the earliest four leagues' worth of data when conducting our wrangling, EDA and inferential statistical analysis. This is because I would like to ultimately use the last two sets of data to test against the model built using the first four as a way to check for accuracy.)

Data Wrangling - Item Files:

For both the currency and item files, I started by pulling the data from the earliest league, Breach. All files have also been renamed to the form "x_currency.csv", "x_items.csv" where x = 1, 2, 3 etc. starting from the earliest league. This is done specifically so that a generalized line of code to read data into my dataframes can be used later on if I wish to expand my project. Once my breach item files are pulled into a dataframe, I noticed that a header value was missing, causing my first four columns to be shifted. I fixed this by resetting the index and creating a dictionary that maps the headers to the correct column values.

Applying the .info() method to my dataframe, I noticed that only 4 columns contain missing values: "BaseType", "Variant", and "Links". From sampled information of the non-missing values in "Variant" and "Links", I noticed that "Variant" contains information regarding unique items that can take on variations of a mod and "Links" provides the number of linked sockets available on that item. Utilizing my domain knowledge of the game, I know that both of these columns do not contain any significant information that could change the value of the item in question and so both of these columns are dropped.

To decide what I needed to do with the missing values in the "BaseType" column, I looked more closely at what items are affected by this. As it were, the datadumps downloaded only provided the type of two sets of items, Divination Cards and Prophecies. To fix this issue, I filled in the missing values in the "BaseType" column with the values in the "Type" column.

Now that the data has been cleaned and all missing values accounted for (choosing to wait until after the EDA has been completed to investigate outliers as that will provide the necessary information for me to make the appropriate decision), I needed to pick out which unique items are worth looking into. I grouped the entire dataframe by the “Name” column and conducted an aggregated calculation of the median on the “Value” column to figure out what the top 15 most valuable items are. I then sliced my dataframe to only include these items.

To make it easier to compare the data from different leagues against each other, I converted the “Date” column into a datetime object and add a “RelativeDate” column that measures the total number of days since the league start and filled in the values by subtracting the start date from the corresponding “Date” column value. Since I have a normalized timescale, I drop the “Date” column entirely. Lastly, I added an extra column indicating the league’s lifespan by splitting up the league into Early, Mid and End (First 0 - 14 days, 15 - 60 days, 60 days - end date respectively). For each step of this process after completing it by hand once, I generalized the steps needed for any item file into a set of functions. This will be important later on.

Data Wrangling - Currency Files:

The wrangling for the currency files were much simpler. Since a preliminary analysis did not show any missing values and the header was already set up correctly, I just created a list of currency items combining a mix of domain knowledge and low median values that I choose to ignore. The dataframe was then spliced to only include the other values. The function for setting up relative date and lifespan columns was also applied, and a currency specific cleaning function was written.

Generalized File Import and Concatenation:

Since I would like to have this code be scalable to any number of sets of data, using our file naming methodology I used the glob library to write a for loop that will import any number of currency/item data sets. Within the for loop, I also applied the respective cleaning function to the dataframe created. Lastly, I concatenated our data into one dataframe for items and one dataframe for currency. I did this by initializing two empty lists for the currency datasets and the item datasets and applying the import loop and appending the imported dataframes to these two lists. The lists were then concatenated using pandas to a new dataframe and deleted.

Data Visualizations (EDA)

I conducted my Data Visualizations in two separate segments: Item Data Visualization and Currency Data Visualization.

- Item Data Visualization
    - League separated - I started by taking a look at the items and the changes in their values over time league by league. I did this by splitting up my list of items into groups that make sense to have possible dependencies on each other and then making separate graphs that only include those items per line graph.

    - Item group separated - Similar to above, I instead separated the items by their groups and then graphed each item across the different leagues in one line graph to make it easier to see if there's any common changes in the values of these items.
    - Individual item boxplots vs lineplots - To have a more detailed look at the individual item fluctuations, I graphed each item in my item list in a separate graph containing the data for the unique item over the four leagues. This graph is a combined boxplot and lineplot graph, with the lineplots showing each league's dataset and the boxplots plotting the data for each item's value at the same relative date.
- Currency Data Visualization
    - League separated - Exactly as done with the item datasets, I split up the currency items into separate groups and plotted lineplots for each group of item per league. I also only include rows where the payment is done using the chaos orb (explained down below).
    - Item separated boxplots - Next, I made individual graphs for each currency item as boxplot graphs, showing the ranges of the values from the 4 leagues for each unique currency item.
    - Exalt & Chaos plots
        - In PoE, the majority of item and currency trades are initiated using two currency orbs: chaos orbs and exalt orbs. On top of this, trades done using these two currency values typically are done with more chaos orbs than exalts due to the relative abundance of chaos orbs within the game compared to exalts. Knowing this, I've decided to take a closer look at the relationship between exalts and chaos across the length of a league.
        - I started by plotting for each league, the value of exalts in chaos and the value of chaos in exalts. Next, I plotted these values against each other in one graph per league (as it does not make sense to compare say, the value of chaos to exalt in the Abyss league to the exalt to chaos value in Breach). Finally, I plotted the ratio of these two values (essentially, what I'm looking at is the value you'll receive by taking an exalt, converting it into chaos orbs then reconverting back into an exalt). Seeing how jagged our data looks when inspecting the ratios per day, I finally decided to take a rolling mean average across a period of 4 days to create a smoother curve that shows a better picture of the gradual changes in the economy over time.
        - Combining all of the above, I finally ended my EDA by creating a boxplot of these rolling mean averages per league across all the leagues.

<u>Inferential Statistics</u>

Before being able to apply any stats functions to my data, I needed to once again wrangle it into a format that makes more sense with regard to processing. Under the assumption that each dataset is an independent observation of the expected trend in time, I decided to treat each day as an observation and each unique item or currency as a feature in this dataset. This lead me to the following list of features that I decided using domain knowledge combined with the EDA from the previous section:

- The Fiend
- The Doctor
- Headhunter
- United in Dream
- The Blue Nightmare
- The Green Nightmare
- The Red Nightmare
- Emperor's Mastery
- Skyforth
- Atziri's Acuity
- Atziri's Disfavor
- The Retch
- House of Mirrors
- Trash to Treasure
- Fated Connections
- Mortal Ignorance
- Mortal Hope
- Blessing of Chayula
- Chayula's Breachstone
- Splinter of Chayula
- Exalted Orb
- Orb of Fusing

With this list, I reduced the dataset to only include these specific items, then aggregated the data by the item's name and it's value at each relative date for each data set using the median value. Pivoting the table from here, I got each unique item/currency in it's own column with the values on each date as observations. I noticed then that I have missing values in our table due to the fact that certain item values aren't available until later in the league (in terms of relative days passed since the beginning of the league). This makes sense, as a couple of these features are so rare that the minimum required amount of time to pass before even one occurrence of said unique item/currency is available for trade is long enough that the rest of the features have already been posted and available for trade. To solve this problem, I backfilled items that did not have data until later in the league and forward filled items that stopped being sold late in the league.

From here, our pandas table is now complete and ready for inferential statistical tests. I defined a heatmap function to show correlations between the features with a mask applied that only shows the unique values for each row and column. I started by doing a pearson correlation test. I then record the corresponding p-values in a pandas dataframe. I then repeated the process except with a spearman correlation test. As I do not expect a linear relationship between the features, the spearman correlation test should provide more accurate results. Finally, I do a standardized cluster map across the entire dataframe.

Machine Learning

Data Wrangling:

From our inferential statistics set of data, I needed to set up my data to make sense for a machine learning algorithm. Based off of my domain knowledge of the game, I'll assume that the price of the Exalted Orb would be the target of our regression calculations. Therefore, from our inferential data section's manipulated data, I pulled out the column for the Exalted Orb as our "y" (target feature) and used the rest of the columns as our learning features (X). From there, I used the sklearn library's model selection feature to split it into a train and test set.. Based off of our data set, our train set has 71 rows and our test set has 18 rows.

Random Forest Regression

From here, we chose to do a random forest regression method to calculate the value of the exalt based off of the other features (listed in X). Before any parameter tuning, I make sure to enable the oob_score = True parameter in our RF class to ensure that we can get a scoring factor that doesn't involve the testing set. To start this, I have to tune our parameters: our n_estimators, max_depth, and max_features parameters. I do this ran a for loop using a list of possible variations of each parameter and appending each oob score to a list corresponding to the variation involved. I then chose the best possible option before moving onto the next parameter. Doing this, I get an n_estimators of 100, max_depth of 16 and max_features of "sqrt".

From here, I made a model including all the features in X and calculated the R^2 and RMSE (Root Mean Squared Error) using the test data. The R^2 value turned out to be .897 and the RMSE turned out to be 4.25. This means we have an 89.7% of being within the mean of the regression with a standard deviation of 4.25 chaos off of the actual value of exalts.

What we did next was use the initial model and pick out the important features. From our code, we see that the most important features are:

- Mortal Hope
- Atziri's Acuity
- Headhunter

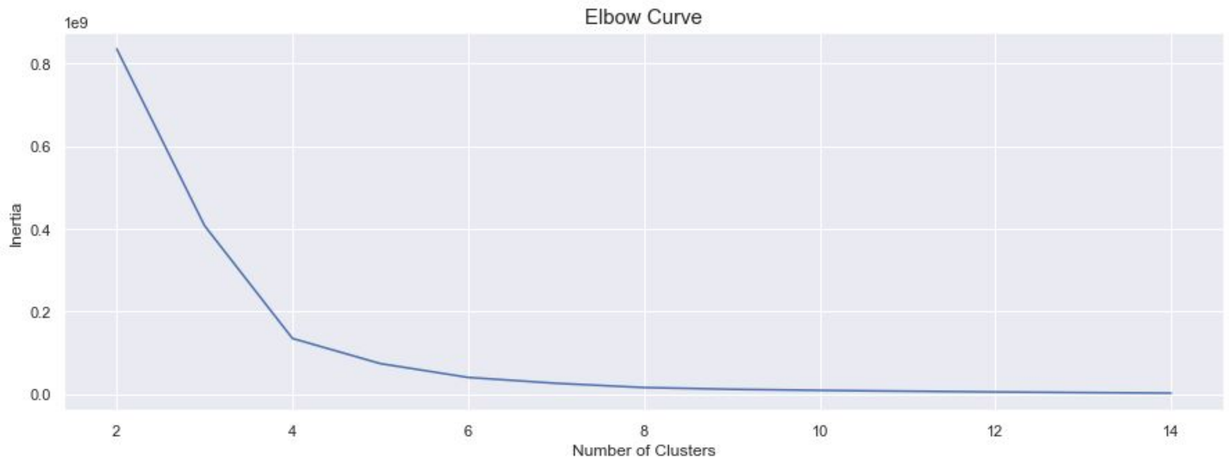
- Hortal Ignorance
- The Doctor
- The Fiend
- Emperor's Mastery
- The Retch

From here, we remake a RF regression class only including these features to test against the test sets. Testing against our two test sets: Beastiary and Incursion, we get these values:

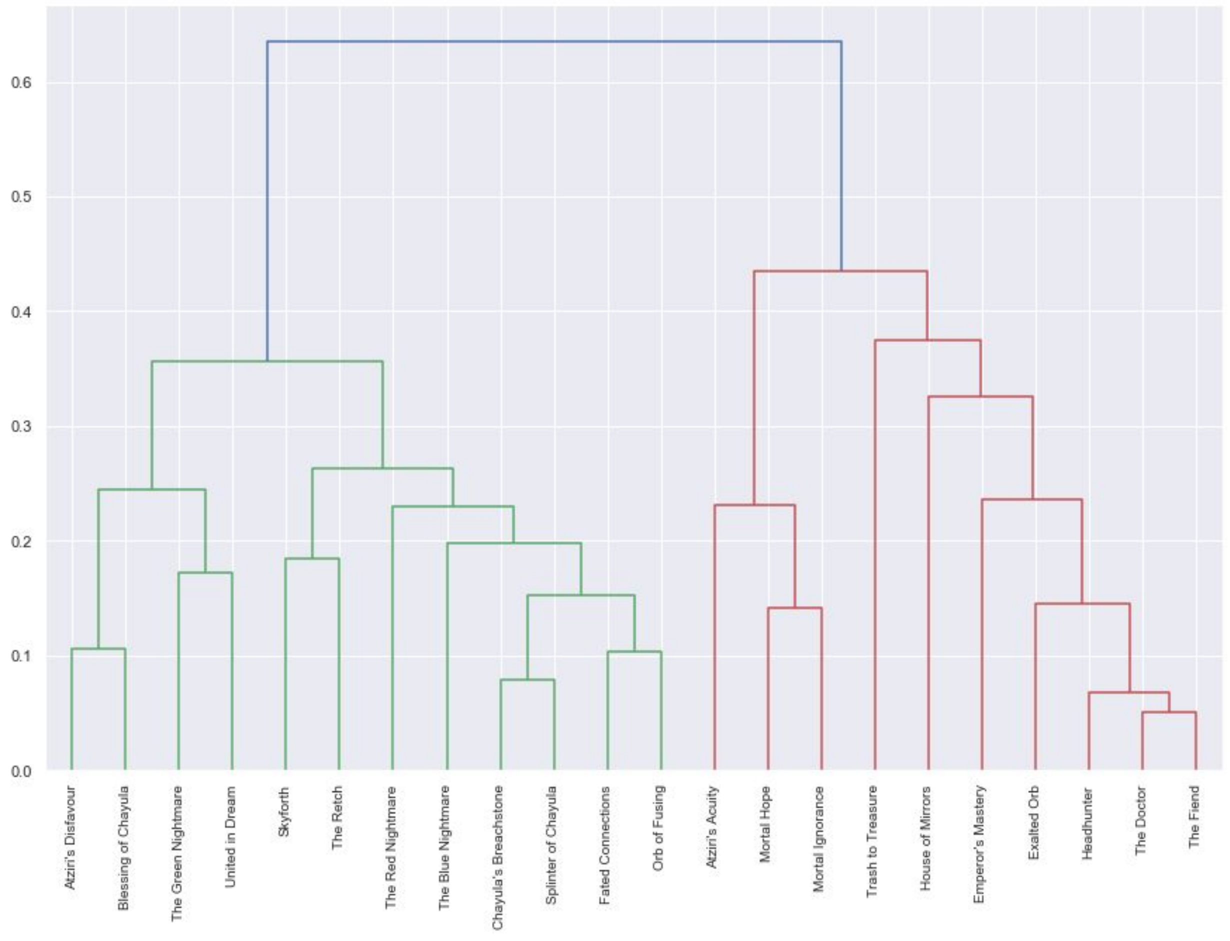
- Beastiary
    - Testing without feature selection:
        - R^2 : -0.692
        - RMSE: 37.48
    - Testing with selection:
        - R^2 : -.0.654
        - RMSE: 37.05
- Incursion
    - Testing without feature selection:
        - R^2 : -0.578
        - RMSE: 33.45
    - Testing with selection:
        - R^2 : -.0.5801
        - RMSE: 33.47

<u>K-Means & Hierarchical Clustering</u>

We first figure out our parameter optimization by using the elbow curve vs inertia as given here:



We see that the optimal n_cluster value is at 6, and by plugging it in, we get a hierarchical clustering map of:

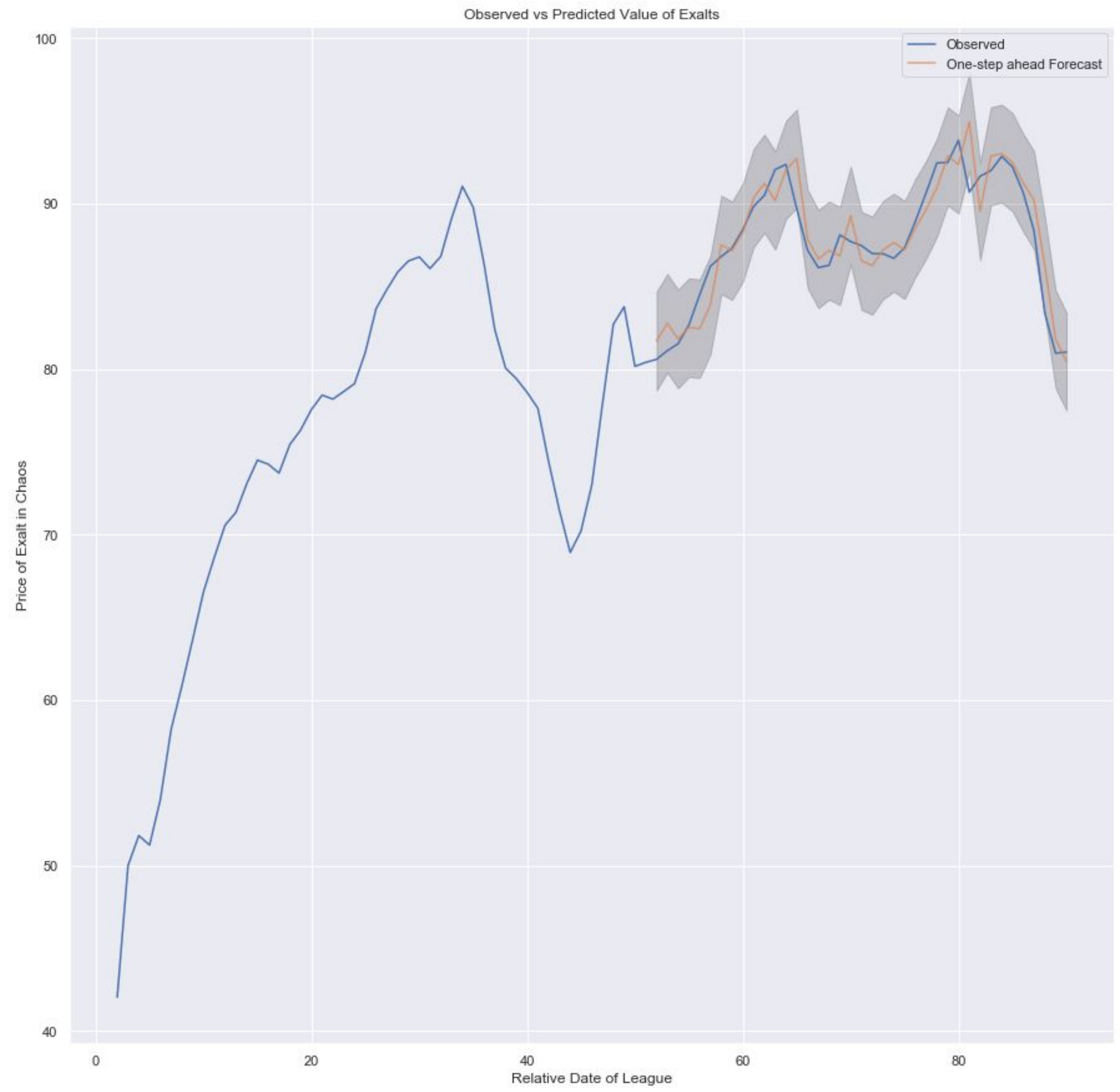


Forecasting with ARIMA

Finally, I attempt to make a forecasted model of exalt prices using the statsmodel library. To do this, I have to figure out the parameters that best fit our model:

- **p:** The lag order
- **d**: The degree of differencing
- **q**: The order of moving average

I do this by initializing my **p** and **d** to test values of 0 and 1, and my value of **q** to test values of 2 and 3. Then, using itertools I can iterate over all parameter iterations and evaluating the AIC scores for each model. Doing this, I see the values that gives me the lowest AIC scores are **p** = 0, **d** = 1, and **q =** 2. Plotting the diagnostics to see if I have any major outliers found in the residual plot, the estimated density plot and the Q-Q plot (I don't), I finally build a predictive one-step ahead forecasting model as shown here:



Conclusion

And that wraps up my preliminary analysis of the economic system of Path of Exile. Some things to note:

- Due to the unique way the data was presented (four leagues of overlapping 90 day observations), our models had to be built using only 90 days worth of data instead of the initial 360 days worth. This led to overfitting and thus when tested against our two test data sets I see poor results.

- We do get valuable insight to exalt prices using our random forest regressor: We managed to pull out the most impactful features (items) and this knowledge can be leveraged in game to make basic predictions on exalt prices.
- K-means clustering gives us additional insight to items that have dependencies on each other. In particular, certain items with high level correlations that were new to me were:
    - Atziri's Disfavor & Blessing of Chayula
    - Chayula's Breachstone/Splinters & Orb of Fusings/Fated Connection
- Our Forecasted ARIMA was not very useful, seeing as the predicted prices lagged behind the observed prices which is useless for a 1-day forecast.

Future possible work include:

- Implementing a model that can include different leagues' datasets as separate observations to expand the number of rows of data that can be used to train the model.
- Find out a better regression model that can fit the data more accurately/Find a better way of conducting forecasting.