Capstone Project 2 - Final Report

Introduction:

As technology advances and many more people are leaning towards online retail services for their products, being able to personalize product recommendations to unique consumers is something any company with an online presence should want to achieve. It would not make sense to randomly recommend a plunger to someone who just bought a high end graphics card or vice versa. It would make much more sense to recommend other computer parts, or other related items, to the consumer who just bought the graphics card. However, it may not necessarily be most apparent as to what other items a consumer may also be interested in. How can a product be determined as relevant to a consumer? Can products of entirely different categories be properly recommended to people? How do other consumers help identify similar items to others? With the enormous amounts of data generated by users of a website, a recommender system can be created to solve these problems.

Data Cleaning:

The goal of this capstone project is to create a recommender system based off of amazon product reviews. A Professor at UCSD named Julian McAuley provides a very large dataset of amazon product reviews ranging from 1996 - 2014. The citation and link to the data is as follows:

> **Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering**
> R. He, J. McAuley
> WWW, 2016
> http://jmcauley.ucsd.edu/data/amazon/

The professor provides the complete review datasets as well as smaller datasets that are split up by categories of products. The complete dataset is a 20gb file, making it impractical to work with in this personal project. Therefore, the smaller dataset of amazon reviews for the product category of electronics, which is ~1.4gb, was used. This smaller dataset only contained data for both users and products who had given and received at least 5 reviews, respectively. Given the size of the dataset, it cannot be uploaded to a GitHub repository. A local copy was accessed for the data cleaning and wrangling.

In order to access the folder in which the data sits, locally, the python module ‘os’ had to be imported. With this module, the os.environ[‘HOMEPATH’] provided the home path where the folder named ‘data’ contained the dataset. Pandas provides a method for importing json data, which was the form of the amazon dataset. The initial attempt at importing the dataset threw a ValueError. This error was resolved by providing the parameter of ‘lines=True’ for the pandas read_json method. This read the file as a json object for each line.

Observing the first 5 rows of the DataFrame showed that the file was imported properly, with the correct columns. Looking at the .info() method for the DataFrame revealed that there were nearly 1.7 million records with the reviewerName column being the only one that had missing values. Instead of using the reviewerName, it was dropped from the DataFrame and the reviewerID was used to identify users. It was also seen that there were two different columns for recording the time of the review, the unixReviewTime and the reviewTime. The unixReviewTime

column was dropped and the reviewTime was kept and converted to a datetime data type. The helpful column provided a list of values for the amount of people who found the review helpful and the total amount of people who gave it a helpful or not rating. This column was split into two separate columns, foundHelpful and totalHelpful. The original helpful column was also dropped after splitting it up. The cleaned dataset was saved to a csv file, locally, allowing for exploratory data analysis to be performed on it. See figure 1 below for an example of what the cleaned dataset looked like.

	itemID	rating	reviewText	reviewTime	reviewerID	summary	foundHelpful	totalHelpful
0	0528881469	5	We got this GPS for my husband who is an (OTR)...	2013-06-02	AO94DHGC771SJ	Gotta have GPS!	0	0
1	0528881469	1	I'm a professional OTR truck driver, and I bou...	2010-11-25	AMO214LNFCEI4	Very Disappointed	12	15
2	0528881469	3	Well, what can I say. I've had this unit in m...	2010-09-09	A3N7T0DY83Y4IG	1st impression	43	45
3	0528881469	2	Not going to write a long review, even thought...	2010-11-24	A1H8PY3QHMQQA0	Great grafics, POOR GPS	9	10
4	0528881469	1	I've had mine for a year and here's what we go...	2011-09-29	A24EV6RXELQZ63	Major issues, only excuses for support	0	0

Figure 1: The first 5 rows of the cleaned amazon dataset.

Exploratory Data Analysis:

With a clean dataset, exploratory data analysis was performed and visualizations were made in order to see trends and relationships within the data. The following are the features of the dataset that were explored: itemID, rating, reviewerID, and reviewTime.

The first area of interest that was explored was the average ratings that items received. Creating a subset of that data that only included itemID and rating, the data was grouped by each item's average ratings. This was done so that a histogram could be created to visualize the distributions of average ratings for items within the dataset (figure 2).

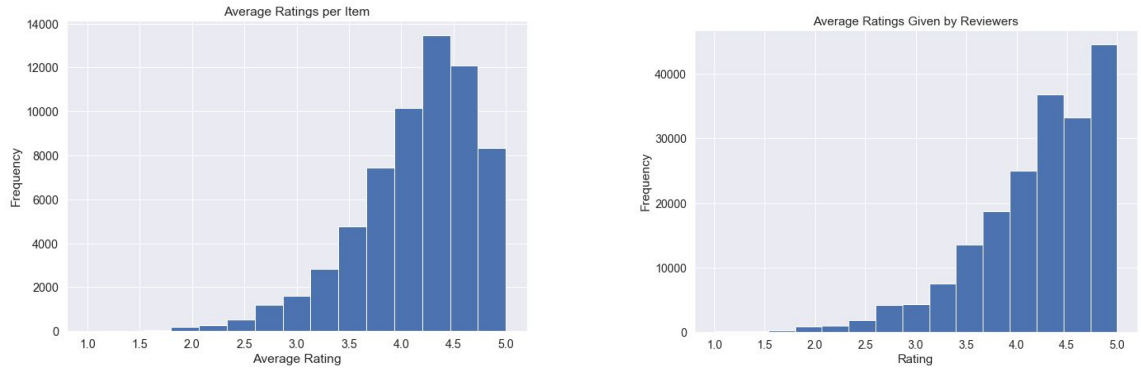

Figure 2:: Histograms of the average ratings that items received (left) and reviewers gave (right).

The visualization showed that the distribution of average ratings had a left-skewed distribution. This meant that a majority of items tend to receive higher average scores of 4-5. The same steps were repeated for reviewerIDs and average ratings to see what type of ratings were given to items on average. The same type of left-skewed distribution was seen, but it was also seen that a large amount of reviewers gave ratings above 4.5.

Next, the count of reviews items received and count of reviews that reviewers gave were looked at. Grouping the respective columns of interest by the count of ratings provided data that showed this. In both cases, most of the dataset contained low amount of reviews received or given. Looking at the pandas describe method showed the distribution of the data. For items, 75% of the data ranged from 5-22 reviews received with a max of 4915. For reviewers, 75% of the data ranged from 5-9 reviews given with a max of 431. The reason why 5 is minimum for both items and reviewers was because that dataset was originally prepared so that only items and reviewers with at least 5 reviews received or given were considered.

A couple of time series plots were created and looked at. The first was a plot of average overall ratings given or received per year. A subset of the data that only included rating and reviewTime was created and the reviewTime column was set as the index. Doing so reverted the reviewTime back to an object data type. In the data wrangling portion of the project the reviewTime was converted to a datetime data type. This was needed to be done again with the reviewTime index. Once again, the pandas to_datetime method was performed on the index and converted it from an object data type to a datetime data type. The datetime dataset was also filtered by years before 2014. This is because the dataset did not contain data past July 2014, making the years between 2014-2015 incomplete. Similar to the pandas groupby method, pandas has a method called resample for datetime data that groups the data by time periods, such as 1 day or 1 month. The rest of the data is aggregated by a given function. This was performed on the subset data of reviewTime and rating in order to group the data by 1 year periods with average ratings. The plot of this data shown in figure 3 displays how the average ratings changed over the years.

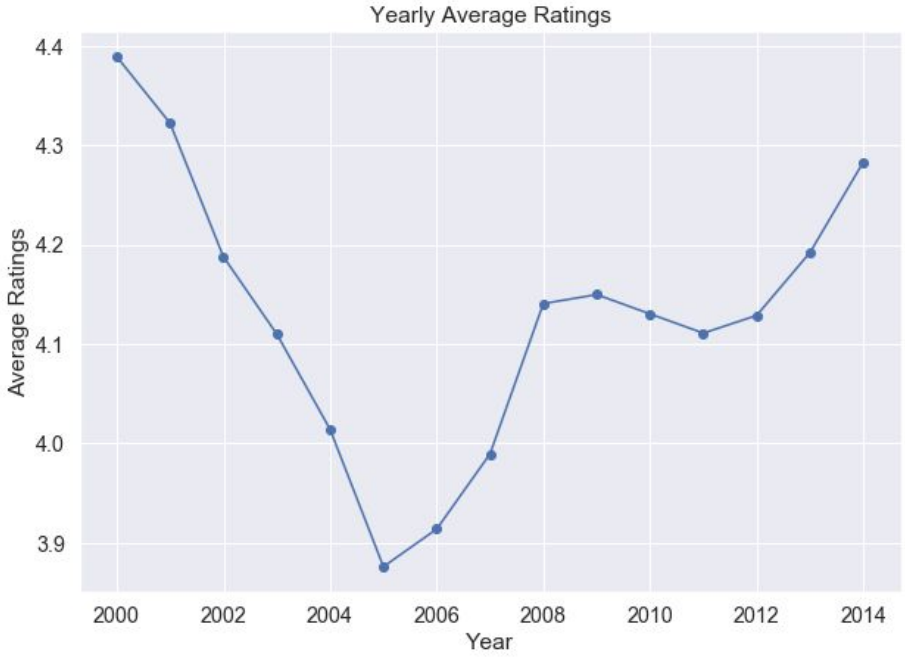

Figure 3: A plot of the yearly average ratings. It is seen that the average ratings tend to be dynamic over the years.

It was seen that the average ratings started at the highest value of about 4.4 in 2000 and then took a sharp dip until 2005 with a low of about 3.8. The ratings then sharply rose until 2008 where they remained relatively stable at values between 4.1-4.2 for about 4 years. The average ratings then started to steadily rise again from 2012 until the end of the dataset.

The next time series plots that were observed were the overall amount of ratings yearly (figure 4), the amount of unique items yearly (figure 5), and the amount of unique reviewers yearly (figure 6).

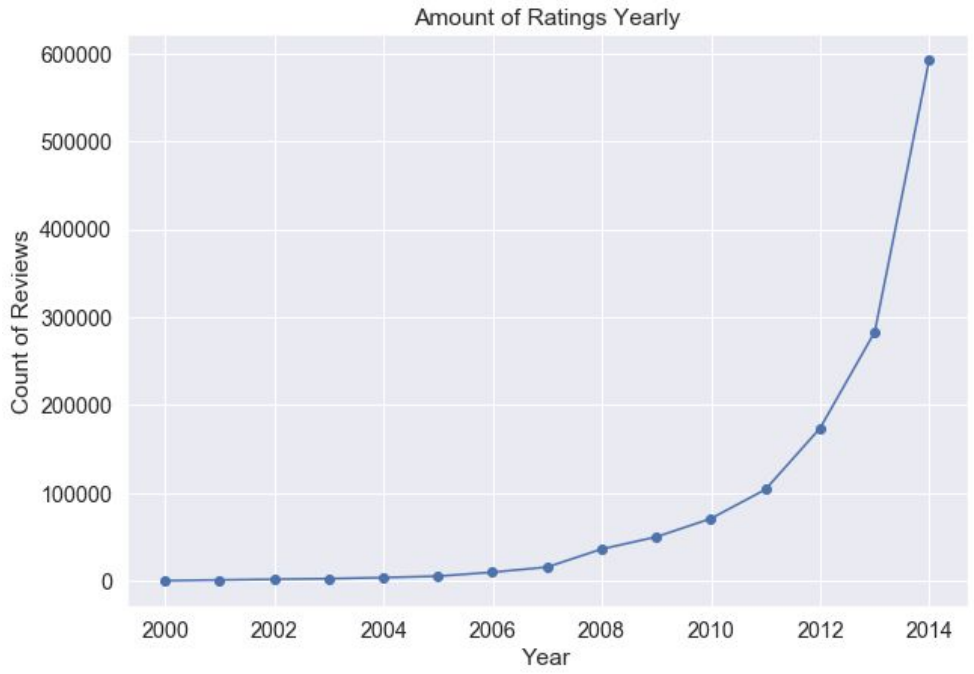

Figure 4: A plot of the total amount of reviews yearly. It follows an exponential curve.

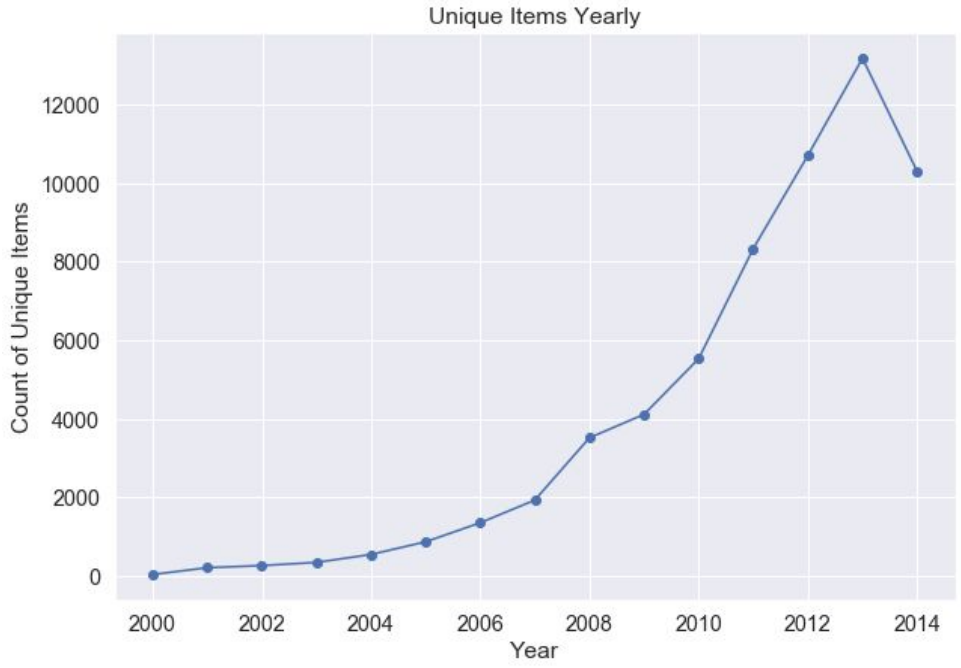

Figure 5: A plot of the count of unique items yearly. It follows an exponential curve and looks to drop from 2013 to 2014.

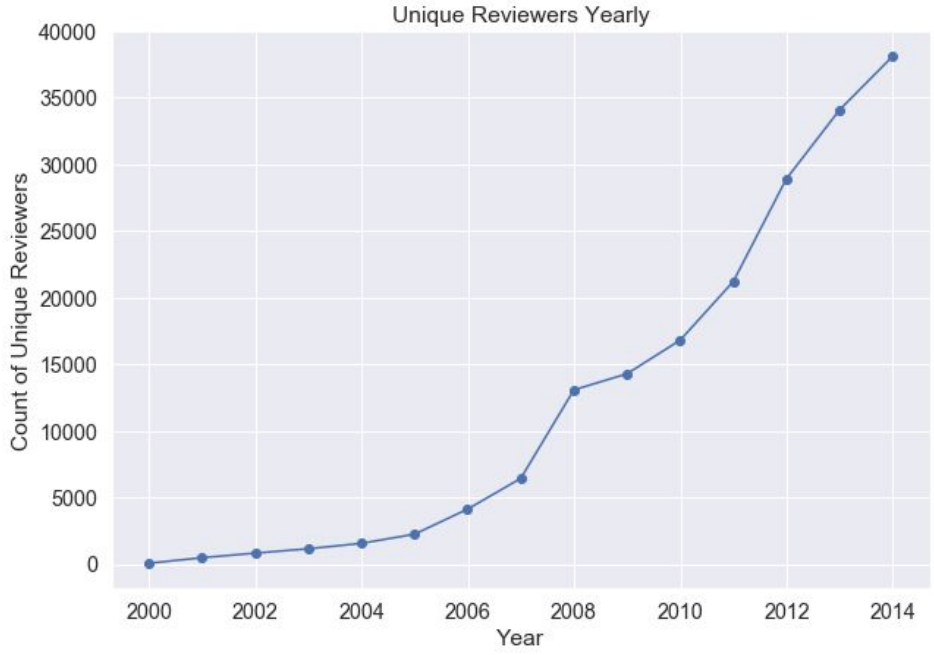

Figure 6: A plot of the count of unique reviewers yearly. It follows an exponential curve and looks to be constantly increasing from 2011 onwards.

For the unique items and the unique reviewers yearly, the dataset was first grouped by the respective columns on the minimum year value. The grouped data was then resampled yearly for counts of unique items or reviewers. All three of these plots showed an exponential increase for the count values every year. All three also showed a spike from year 2007 to 2008. It was also seen that for the plot of unique items yearly, there was a drop from 2013 to 2014, which was not seen in the plot of unique reviewers yearly. This could signify that the amount of unique reviewers may continuously grow while the amount of unique items may start to level out.

There were many interesting observations made in this initial exploratory data analysis of the dataset. It was seen that both items and users tend to receive and give high ratings for reviews. A majority of the dataset contained relatively low amounts of reviews given and received. A time series plot of average ratings yearly showed some interesting changes. The ratings started high, dropped sharply until 2005, sharply rose until 2008, remained stable for 4 years, and then steadily rose until the end of the dataset. Three additional time series plots of amounts of ratings yearly, amount of unique items yearly, and amount of unique reviewers yearly all showed exponential increases. They also all showed a spike from 2007 to 2008. There were also signs of the amount of unique items yearly starting to level out while the amount of unique reviewers yearly continuously growing.

Deeper Analysis:

One of the interesting observations from the average yearly ratings plot was the drop in 2005. In order to draw some insight from this plot, the proportions of ratings per year were looked at. This was done by dividing the count of the rating by the total count of ratings every year. This was repeated for each rating from 1-5, then all were plotted on the same graph (figure 7).

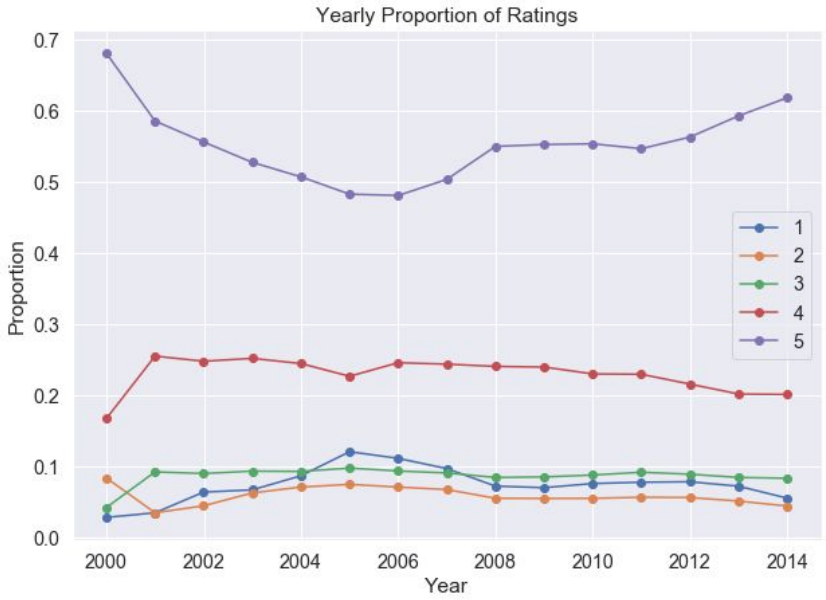

Figure 7: A plot of the yearly proportions of ratings from 1 to 5. The ratings of 1 and 5 seem to be the most dynamic.

This plot showed that the ratings that had the most dynamic movements were the ratings of 1 and 5. The valley for average ratings in 2005 can be represented by looking at the proportion of ratings that were 1 and 5 in 2005. For that year, the proportion of ratings that were 5 were at the lowest and the proportion of ratings that were 1 were at the highest. From observing the proportions it looked like a majority of reviews received ratings of 4-5.

The next plot that was looked at was the count of ratings yearly. This plot showed that the count of reviews increased exponentially over the years. In order to perform a linear regression, the curve would need to be transformed. This was done by taking the logarithm of the count of reviews. Doing so created a linear curve for the log of count of reviews for each year. The python module that was used for the linear regression was scipy stats. The linear regression function from this module provided the coefficients for the variables and the coefficients, the p-value, and the R-squared value. The coefficients were used to plot the linear regression line, p-value was used for the hypothesis test, and the R-squared value was used to measure the goodness of fit for the line. The following figure is a linear regression plot of the transformed count of reviews over the years (figure 8).

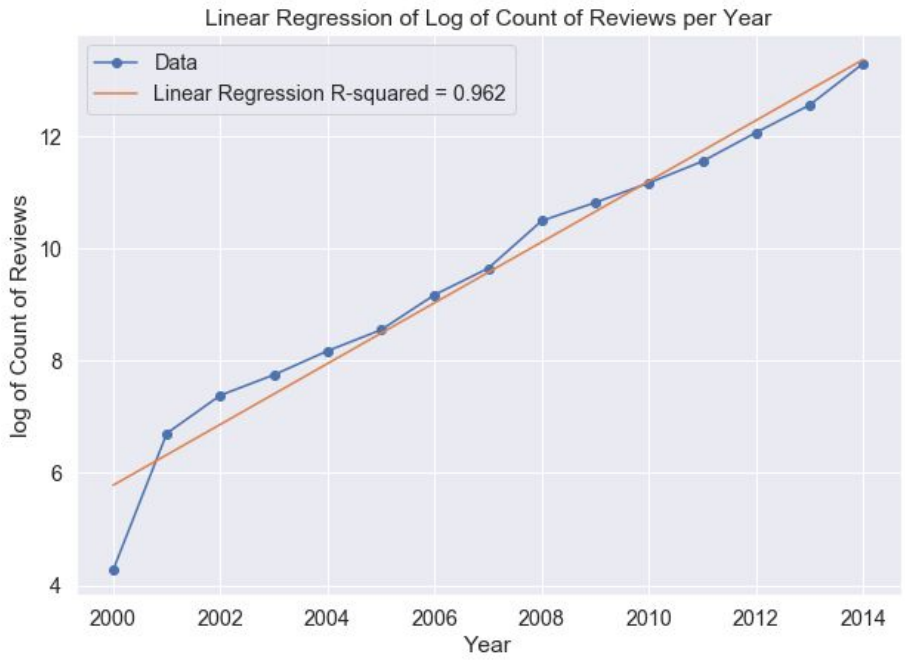

Figure 8: A plot of the log transformed count of reviews over the years. A linear regression was performed and an R-squared value of 0.962 was calculated.

The hypothesis test was performed in order to determine whether or not there was a significant relationship between the log of count of reviews and years. The null hypothesis was that there is no relationship between the two variables. The alternative hypothesis was that there was a relationship between the two variables. With a significance level of 0.05 and a

p-value of 1.20e-10, the null hypothesis was rejected and the alternative hypothesis was accepted. The R-squared value represents the proportion of the variance that can be explained by the explanatory variable. The R-squared value was calculated to be 0.962. This meant that about 96% of the variability in the log of count of reviews could be explained by the year.

After performing a deeper analysis of the dataset, insights were discovered and a linear regression was performed. The average ratings per year could mostly be explained by proportions of ratings that were 1 or 5. It was also seen that a majority of ratings were 4 or 5. The linear regression showed that there was a linear relationship between the log of count of reviews and the year. The R-squared value was able to show that 96% of the variability could be explained from the linear regression.

Recommender System:

There are many different approaches to implementing a recommender system. For this project, matrix factorization will be performed. Matrix factorization can be applied to a large matrix of user/item pairs of ratings. The rows will be each user and the columns will be each item. This type of matrix is commonly sparse because not every user is expected to have ratings for every item and not every item is expected to have ratings from every user.

Factorization can be applied to this matrix in order to produce two smaller, separate matrices for users and items. One matrix will contain latent features for each user and the other matrix will contain latent features for each item. The dot product can be performed on these two separate matrices in order to calculate the ratings for each user/item pair. Based on the actual ratings for each user/item pair from the original matrix, the values of the latent features can be updated in order to minimize the loss function. With the best estimate for the latent features, user/item pairs that did not exist in the original matrix can be estimated. For a specific user, the top 5 highest rated items that had not been rated can be recommended. Figure X shows an example of matrix factorization.

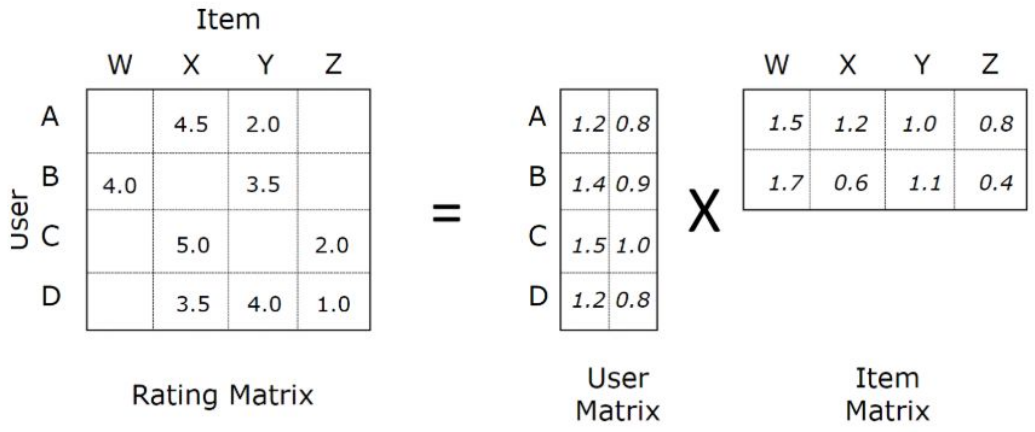

Figure 10: An example of matrix factorization from a medium article. This shows how a larger User x Item matrix of ratings can be factorized into two separate matrices for Users and Items. https://medium.com/@connectwithghosh/simple-matrix-factorization-example-on-the-movielens-dataset-using-pyspark-9b7e3f567536

For this capstone project, the scikit surprise package was used in order to implement the recommender system on the Amazon dataset. This package contains models such as SVD which was popularized by Simon Funk from the Netflix Challenge. SVD closely resembles matrix factorization. Surprise takes data in the form of 3 columns: users, items, and ratings. The amazon dataset had the columns of reviewerID and itemID which were in the datatype of object because the IDs were strings. In order to more efficiently and easily work with the IDs, they needed to be encoded. This was done by creating Categorical datatypes for reviewers and items based on the unique values of each. The index, or codes, for each of the categories were used as the ID value for reviewerID and itemID. Figure 10 shows the preprocessed dataset to be fed into the surprise model.

	reviewerID	itemID	rating
0	176008	0	5
1	173739	0	1
2	134504	0	3
3	24476	0	2
4	57419	0	1

Figure 10: The preprocessed dataset for building a recommender system.

Surprise is able to load the dataset from a pandas dataframe and then parse it with its reader. With the dataset loaded, the surprise GridSearchCV method was used in order to find the optimal hyperparameters for the SVD model. This GridSearchCV is very similar to the one in sklearn. It takes a list of hyperparameter values to try out and calculates the RMSE values for each combination of them. The combination of hyperparameters that produced the lowest RMSE score was used. For the SVD model on this dataset, the following “best” hyperparameters resulted from the grid search: lr_all = 0.01, reg_all = 0.5. With these hyperparameters, the entire dataset was fit to the model and then was 5-fold cross validated. Figure 11 shows the results of the cross validation.

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.0898  1.0883  1.0886  1.0917  1.0932  1.0903  0.0019
MAE (testset)     0.8190  0.8183  0.8180  0.8204  0.8206  0.8193  0.0011
Fit time          117.27  124.32  120.35  123.40  111.09  119.29  4.79
Test time         5.05    4.44    3.82    4.70    4.10    4.42    0.43
```

Figure 11: The results of the cross validation on the Amazon dataset for the SVD model. This shows that the model performed equally as well on different subsets of the data.

The results show that the model performed similarly on the 5 different subsets of the data. The RMSE values for each of the folds were all very similar and this can also be seen by the low standard deviation value.

Now that the model has been fit to the dataset, predictions on the ratings can be made for any given user/item pair. A list of all the items that a specific user had not rated yet was created. The predicted ratings were put into a list and sorted by values. The top 5 highest predicted ratings of items were recommended. Figure 12 shows the results of the recommender system for an arbitrary reviewer.

```
The top 5 recommendations for user A1ZD690RCXOSB are:
        itemID: 50352   item: B0087RF5RG        rating: 4.943
        itemID: 32961   item: B0041ORN9M        rating: 4.904
        itemID: 9653    item: B000HVHDJ8        rating: 4.898
        itemID: 1376    item: B000068IGO        rating: 4.893
        itemID: 57954   item: B00C10T2EC        rating: 4.891
```

Figure 12: The top 5 recommended items for an arbitrary reviewer. With the recommender system made, top 5 recommendations can be predicted for any given reviewer.

Using the Amazon dataset and the scikit surprise package, a recommender system was created with the SVD algorithm which closely resembles matrix factorization. GridSearchCV was performed in order to find the best hyperparameters for the model, which were used to fit and cross validate the model on the dataset. The algorithm was able to predict unrated user/item pairs based on the latent features of each separate item and user. For any specific user, the ratings of items that have not been rated yet can be predicted. From these predictions, the top 5 highest values can be recommended as items that the particular user may also rate highly.

Summary:

The Amazon dataset was used in this project in order to create a recommender system. Before implementing the recommender system, the dataset had to first be cleaned and filtered. After the data wrangling step, the cleaned dataset was ready to be explored. Some of the features of the amazon data, such as itemID, reviewerID, ratings, and reviewTime were explored. Exploratory data analysis was performed and visualizations were made in order to

better understand the dataset. It was seen that the average ratings every year followed an interesting pattern. Upon deeper analysis, it was seen that the erratic movements of that plot was largely due to the change in the proportions of ratings that were 1 and 5. Looking at the time-series plots of the count of overall reviews, the count of unique reviewers, and the count of unique items over the years revealed that they all followed an exponential curve. Performing a log transformation on the count of reviews allowed the plot to follow a linear curve and a linear regression line to be fit to it. The R-squared value of the fitted line was calculated to be 0.962.

After cleaning and exploring the dataset, a recommender system was created. The Surprise scikit package was used to fit an SVD model to the dataset using the optimal hyperparameters found from the GridSearchCV. The SVD model is similar to matrix factorization. The fitted model allowed predictions for any user/item pair of ratings. For any arbitrary reviewer, the predicted ratings for all items could be estimated. By providing a list of all the unrated items for a reviewer, recommendations can be made based on the highest estimated ratings for the items.