CARS: A Containerized Amazon Recommender System

Adam Cassell, Andrew Muñoz, Brianna Blain-Castelli, Nikkolas Irwin
Feng Yan, Sergiu M. Dascalu, Frederick C Harris, Jr.

Computer Science and Engineering
University of Nevada, Reno
Reno, Nevada 89557, USA
acassell@nevada.unr.edu amunoz24@nevada.unr.edu bblaincastelli@unr.edu nikkolasjirwin@nevada.unr.edu
fyan@unr.edu dascalus@cse.unr.edu fred.harris@cse.unr.edu

Abstract—**With the big data boom, recommender systems that make intelligent recommendations for users have been playing an important role in today's industry. However, existing recommender systems often overlook scalability, flexibility, and portability. They also commonly lack in-situ visualizations. To solve these problems, we present CARS: A Containerized Amazon Recommender System. CARS processes large Amazon data sets for analysis and makes product recommendations. However, its utility is not restricted to only prominent organizations like Amazon. CARS achieves scalability by taking advantage of industry-grade recommendation tools irrespective of available hardware resources. CARS runs in a completely isolated environment to promote flexibility and remote collaboration. The demonstrated implementation generates shopping recommendations from user ratings within product review data sets. CARS processes this review data using Apache Spark, a unified analytics engine for big data. The system complements recommendations with data-driven insights and interactive visualizations. In addition to these features, CARS contains a robust set of command line options to customize the results shown to the end-user, perform logging of processed data, and provide performance monitoring through Spark's built-in web-interface. Highly portable and automated analysis of purchase data helps organizations understand the habits of their customers. CARS demonstrates the feasibility of such a system for a wide variety of users.**

Keywords: Big Data, Scalable Systems, Containerization, Recommender System, Data Analysis, Data Visualization, Collaborative Filtering, Alternating Least Squares, Apache Spark, Performance Analysis.

I. Introduction

As companies continue to accumulate big data, it is important to use the collected data to enhance customer experiences and help make data-driven decisions. Recommendation systems are commonly found on e-commerce and entertainment websites. These systems collect individual behavioral data including purchases, ratings, and views. This data, once gathered, is then processed and used to provide future recommendations to users with similar interests.

For large organizations like Amazon and Netflix, the collection of user data, processing of user data, and use of robust recommender systems can be easily accomplished due to large development teams and support infrastructure. However, smaller organizations do not have such resources even though they also want to benefit from these systems. Additionally, the dynamic nature of web traffic makes it difficult for organizations to manage data at scale without building applications that are designed for cloud environments.

CARS uses a containerized environment to provide an accessible, light-weight, portable and scalable recommendation system. We prototype CARS on top of Conda, Docker, Python, and Spark. Our extensive evaluation of CARS demonstrates promising results. CARS allows review data to be processed in a wide range of environments using minimal hardware that can be scaled up to meet the needs of the end-user. The two fixed constraints for CARS are the use of Docker and the physical resources allocated to the container. Item recommendation results and insightful aggregated statistics are provided to users alongside interactive visualizations which provide fine-grained analysis.

The rest of this paper is organized as follows: Section II discusses the project background and provides examples and information on related work. Section III describes the implementation of CARS. Section IV evaluates the individual visualizations and analysis. Section V lists concluding thoughts and future work.

II. Background And Related Work

According to *Spark the Definitive Guide*, recommender systems are "one of the best use cases for big data" [1]. Such a system can analyze users' explicit preferences or implicit preferences to make predictions on a user's future behavior. There are many use cases for recommender systems across a variety of domains. Entertainment platforms often use such systems to help users discover new content that they may like. For example, Netflix leverages Apache Spark to implement the movie recommendation system that many consumers are familiar with. Amazon, likewise, uses a similar implementation on its item catalog to drive shopping recommendations based on user preferences and interests. A previous user study shows the significance of these recommender systems on user shopping trends at Amazon when "[a]ll participants used one or more recommender feature[s]" [2]. User-data-driven recommendations have become core components of many large-scale products. It follows that making these processes accessible to a wider variety of systems will bring more effective experiences to even more domains.

One way to implement a recommender system is by using a content-based approach, which uses detailed information about specific items or user attributes to drive predictions. This includes features such as textual content, genre, item category, and other relevant metadata [3]. Conversely, collaborative filtering relies on the historical preferences of users (i.e. the actions they take). This approach is especially useful for "complex and hard to represent concepts, such as taste and quality" [4].

There are two types of user preferences: explicit rating and implicit rating. The former describes direct ratings given by users, such as a five-star rating or numeric scale. The latter is characterized by indirect indications, such as page clicks, image views, purchase records, and other passively traceable statistics. CARS uses explicit rating for user preferences.

One of the most widely-used algorithms for collaborative filtering is Alternating Least Squares (ALS), which supports explicit or implicit user feedback. Spark natively supports ALS and includes multiple variants of scalable ALS implementations in its MLlib library. MLlib was used for the CARS recommender system due to its convenience and performance benefits. ALS finds a k-dimensional feature vector for every user and item such that the dot product of each item's feature vector with each user's feature vector estimates the user's rating for that item [1]. This resulting feature vector then drives the recommendations.

A set of Amazon's review data is made freely available by researchers at UCSD [5]. This data set is divided into separate subsets of data which include the five-core data sets, raw review data, user review data, and ratings only data. As stated previously, the CARS project utilized the five-core data sets for its implementation. This Amazon data set has also been the subject of multiple related works. In the paper *Estimating Reactions And Recommending Products With Generative Models Of Reviews*, Ni *et al.* use the Amazon data set to generate predicted review text using natural language inference techniques [6].

Another paper, *Justifying Recommendations using Distantly-Labeled Reviews and Fined-grained Aspects* by Ni et al., uses Amazon clothing data to extract meaningful justifications that are pertinent to customers' decision-making processes [7]. An additional work, titled *Large Scale Parallel Collaborative Filtering for the Netflix Prize* by Zhou et al. [8], uses Netflix data to demonstrate an ALS implementation with weighted regularization for ratings prediction. It is also worth noting that both papers, [6] and [7], were written by the researchers who made the Amazon review data set freely available.

III. Approach

The implementation of CARS starts with Conda. First, dependencies are installed and then the application is isolated into a Docker container equipped with Jupyter Notebook. Then, the recommender system and interactive visualizations are added. After completing the features above, a command line parser is integrated to provide the end-user with additional options when executing CARS with a given data set.

CARS uses Conda, "an open source package management system and environment management system that runs on Windows, macOS and Linux" [9], to facilitate cross-platform package management, dependency management, and isolation through a virtual environment. By using Conda, CARS automates the process of installing and running the application in a consistent and reproducible manner. Through Conda, CARS users can also extend the existing application by installing new packages without dependency conflicts.

Conda, and the application code for CARS are contained within a Docker container. This container uses a custom image that builds upon Project Jupyter's minimal-notebook image. Using this approach, CARS was guaranteed to have stable Jupyter Notebook support along with custom configurations. One of the configurations required by CARS was the ability to store results regardless of the container's lifecycle/state.

Relying on the container's persistence layer would not be sufficient for working with the Amazon review data sets since any work performed while running CARS would be lost if the CARS container's lifecycle was interrupted. The solution to this issue was to add a mount point to the custom Docker image file and then configure the volume to store our files as well as the data sets used for running our recommender system.

While adding a Docker volume may not be a necessity for end-users who have experience with Docker, CARS utilizes this mechanism to bundle the data sets directly into CARS so that the end-user can immediately begin working with the data sets and further customizing the program for their specific needs. To ensure that the Docker image size is not too large, only a subset of the five-core data sets is bundled by default, but more can be added to the Docker volume as needed.

The data visualizations were designed to serve as key insight into the Amazon review data and user preferences. The visualizations that were created include:

- Items Over Time
- Summary Statistics
- Helpful Reviews
- Prediction Performance

These data visualizations provide insight into some of the basic statistics of the review data, trends over time, and several interesting relationships. The prediction performance plots explore how well the ALS algorithm handles the data and produces its recommendations. Each of the data visualizations mentioned will be explained in further detail in Section IV.

IV. Evaluation

We evaluate CARS in this section, including review data analysis and recommendation performance. Corresponding visualizations serve to highlight different aspects of the data, such as relationships or changes over a period of time. All

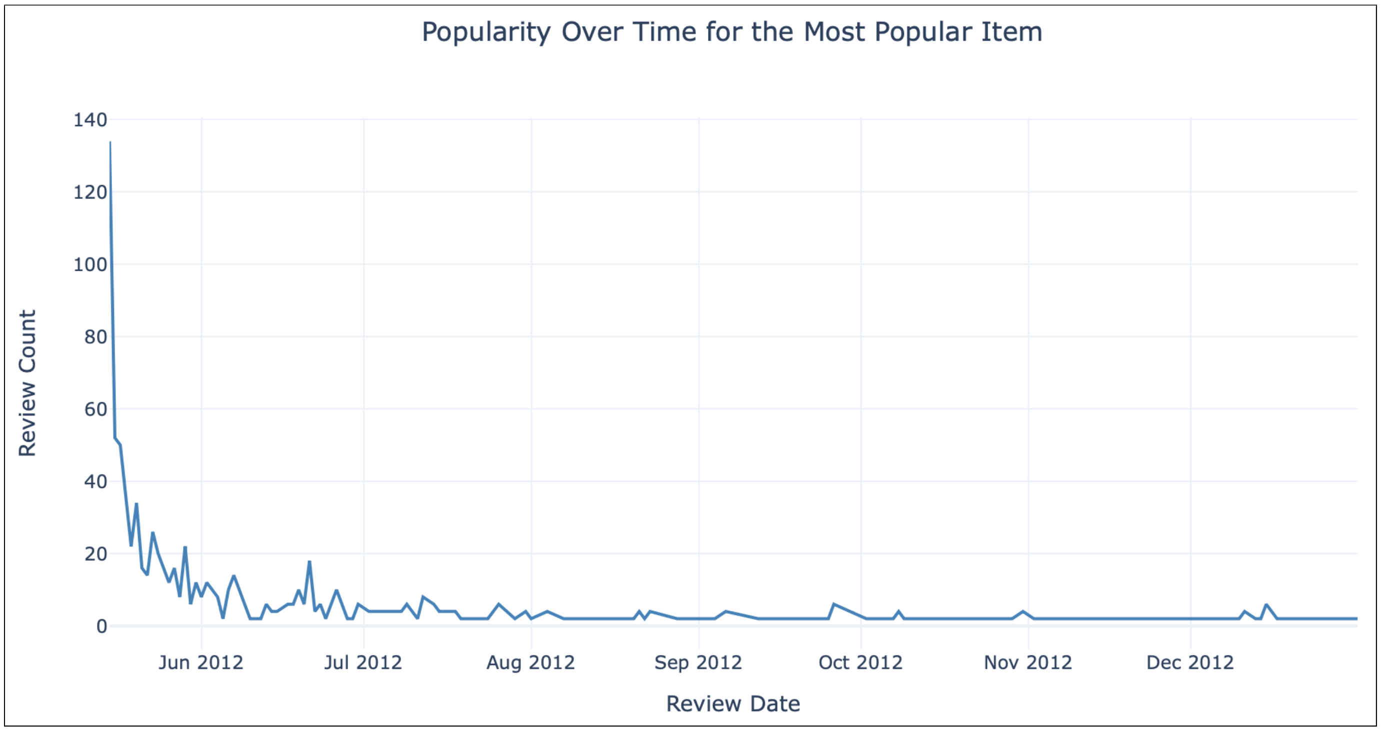


Fig. 1. Time series plot showing review count over time for the most popular item.

visualizations depicted and shown throughout this section utilize the five-core Video Games data set as the primary basis. The five-core data sets is simply a collection of products and reviewers that each have a minimum of at least five reviews. This helped with processing and running the data sets locally.

A. Items Over Time

It is possible to infer an item's popularity over time using review data. For any data set provided, CARS selects the most popular item of that category as a case study for such analysis. The result is a time-series line plot, which can be seen in Figure 1, displaying the number of daily reviews for that particular item.

It is worthwhile to observe how an item's popularity changes over time as potential indicators of product success. In this example, this particular video game received a large amount of reviews (over 100) on the first day of release. After that, however, popularity rapidly dropped after those initial few days and weeks (down to more modest single-digit daily numbers). This visual suggests that the item had high public interest before and at the time of release. This pattern intuitively makes sense for something like a video game or movie. There is built-up demand followed by the item quickly becoming outdated compared to other new releases. Contrast this with an item of another category, such as medical. Popularity for a medical item, such as a box of bandages, is likely to have much more steady popularity as it is a common good. Marketers could use these insights to best plan their advertising strategies. Advertising campaigns could either take advantage of the existing popularity patterns, or attempt to change them to better reach certain goals.

B. Summary Statistics

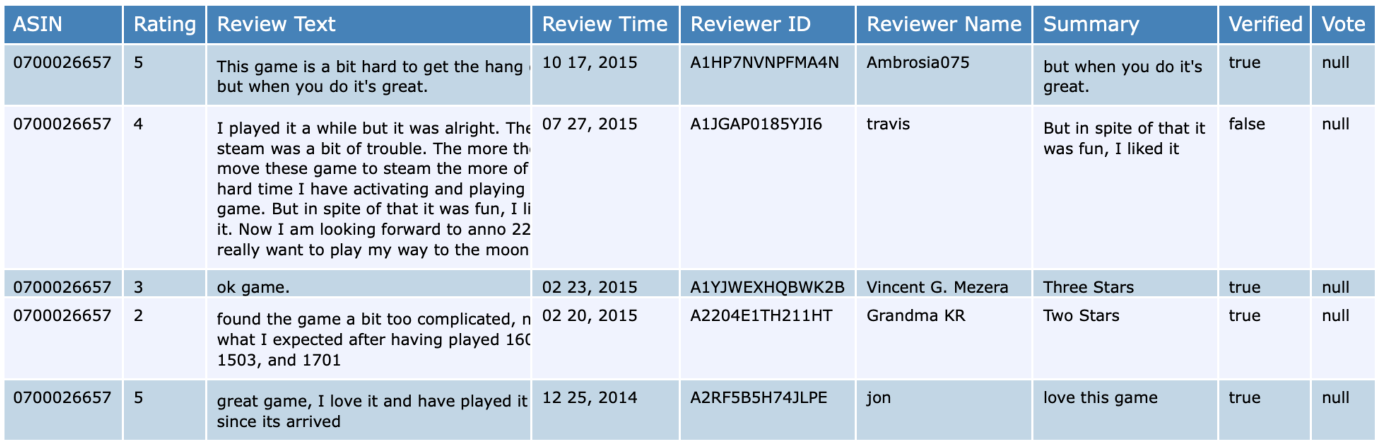
It is important to evaluate the distribution of the data set when considering any downstream analysis. Table I shows the Amazon data set schema and a subset of the video games data. Summary statistics are provided in the form of a summary table as well as a ratings distribution histogram. The summary table, as seen in Table II, communicates the mean, standard deviation, count, minimum/maximum values, and quartile ranges of all rating values in the data set. The ratings distribution displayed in Figure 2 shows how many occurrences of each rating (discrete values between one and five) exist in the data set.

The histogram is useful for understanding how the ratings are distributed for each data set. In this case, it is evident that video game ratings overwhelmingly trend positive, with the highest occurring ratings being five stars, followed by four star ratings.

C. Helpful Review Data

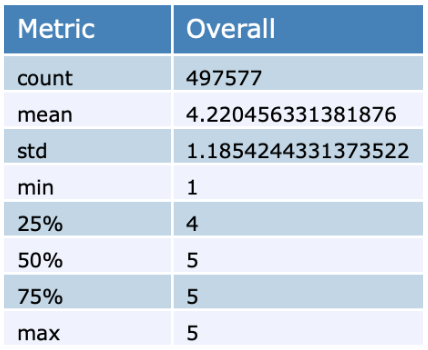
A valuable metric for analyzing user reviews is review 'up-votes'. Users can express how 'helpful' they find a particular review by voting on it. This in turn helps shoppers decide which reviews to give more credence to, thus revealing the most influential reviews. CARS visualizes this metric for the most-reviewed item of the data set, as well as for the data

TABLE I
Structure and example data of the video game review data set.



ASIN	Rating	Review Text	Review Time	Reviewer ID	Reviewer Name	Summary	Verified	Vote
0700026657	5	This game is a bit hard to get the hang but when you do it's great.	10 17, 2015	A1HP7NVNPFMA4N	Ambrosia075	but when you do it's great.	true	null
0700026657	4	I played it a while but it was alright. The steam was a bit of trouble. The more th move these game to steam the more of hard time I have activating and playing game. But in spite of that it was fun, I li it. Now I am looking forward to anno 22 really want to play my way to the moon	07 27, 2015	A1JGAP0185YJI6	travis	But in spite of that it was fun, I liked it	false	null
0700026657	3	ok game.	02 23, 2015	A1YJWEXHQBWK2B	Vincent G. Mezera	Three Stars	true	null
0700026657	2	found the game a bit too complicated, n what I expected after having played 160 1503, and 1701	02 20, 2015	A2204E1TH211HT	Grandma KR	Two Stars	true	null
0700026657	5	great game, I love it and have played it since its arrived	12 25, 2014	A2RF5B5H74JLPE	jon	love this game	true	null

TABLE II
Summary statistics of the video game review data set.



Metric	Overall
count	497577
mean	4.220456331381876
std	1.1854244331373522
min	1
25%	4
50%	5
75%	5
max	5

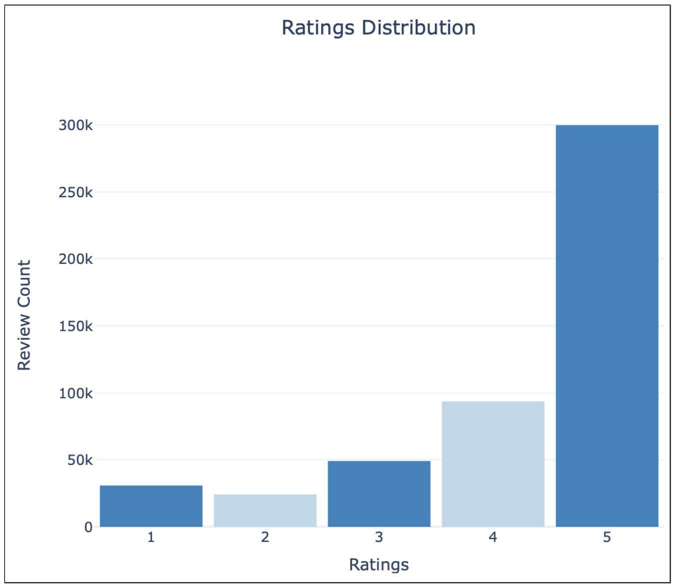


Fig. 2. Histogram visualizing the ratings distribution of the video game review data set.

set as a whole. This evaluation section focuses on the former visualization for the most popular item.

To explore the ratings and votes associated with each of the item's reviews, a scatter plot is provided in Figure 3. The purpose of this plot is mostly investigative. The x-axis represents every single reviewer for the item, and the dual y-axes represent ratings and votes, respectively. The intended usage is for the to pan and zoom around this plot to analyze items of interest. This plot, like many others generated by CARS, includes Plotly [10] interactivity for these purposes. This particular visualization implements WebGL to more efficiently render hundreds of thousands of interactive points if necessitated by the data set.

Using this visual, it is evident that most users up-voted the reviews that gave one-star ratings. This implies a generally negative shopper consensus. Very few shoppers up-voted the five-star reviews, which is notable. There is a stark contrast between this item's vote distribution and the general ratings distribution for the video game category as a whole (See Section IV-B). A logical conclusion from this discrepancy follows: This item seems to be far more poorly received than most of its competitors. Yet, it was still far more popular than the other games sold on Amazon. It is up to the seller to determine whether this constitutes a success.

D. Prediction Performance

The previous visualizations pertained to general analysis of the data set. The following outputs instead demonstrate the performance of the ALS algorithm used for the recommender system in CARS. Before ALS can generate it's list of item recommendations per user, rating predictions must first be computed. These are the rating values that the algorithm predicts each user would assign to each item. These predicted ratings form the basis of the eventual recommendation groupings presented as final output. Thus, the accuracy of these predictions is critical and can be visualized to assess ALS performance.

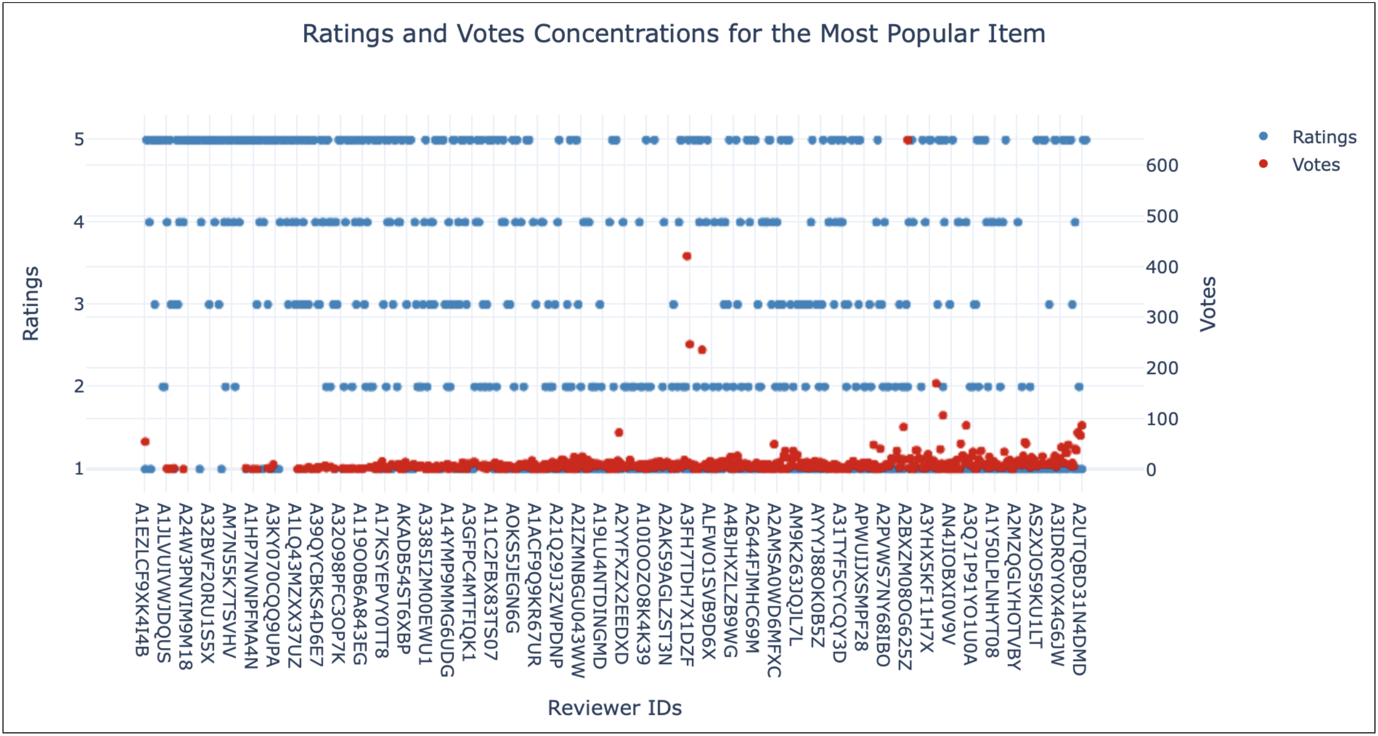
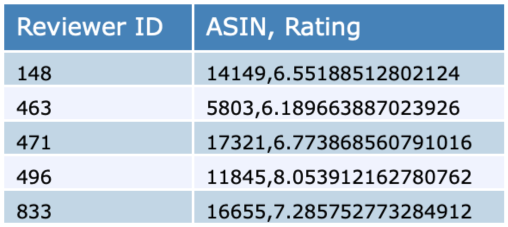


Fig. 3. Ratings and votes concentration for the most popular item in the video game review data set.

TABLE III
RECOMMENDATION RESULTS COMPUTED BY ALS ALGORITHM.



Reviewer ID	ASIN, Rating
148	14149,6.55188512802124
463	5803,6.189663887023926
471	17321,6.773868560791016
496	11845,8.053912162780762
833	16655,7.285752773284912

To best visualize prediction accuracy, Figure 4 shows an error distribution plot. Prediction error is calculated by subtracting the true rating from the predicted rating for each sample. The result is a prediction error sequence that can be organized using a histogram. The resulting distribution is close to Gaussian, which indicates the algorithm is satisfyingly accurate. In Table III, the final recommendation results outputted by CARS are listed.

V. Conclusion and Future Work

Recommender systems enhance user experiences by providing suggestions tailored towards users' interests. Implementing recommender systems and collecting enough user data to generate accurate recommendations can be challenging. CARS is designed to address these challenges and demonstrates the feasibility of such a system for organizations of any size. Built on top of Conda, Docker, Python, and Spark, CARS enables extensibility, dependability, portability, and scalability for systems deployed on varying hardware resources. In addition to these features, CARS supports a variety of visualizations on individual product and aggregated data.

As our future work, CARS will be improved by incorporating Ansible into the design so that our containerized application can be further automated to simplify the executions. In addition to automated Docker, CARS will continue to gain new data visualization features. The visualization wish list includes sales rank, item popularity based on categories, and comparative analysis based on different parameters.

Acknowledgements

This material is based in part upon work supported by the National Science Foundation under grant numbers IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] B. Chambers and M. Zaharia, *Spark: The Definitive Guide : Big Data Processing Made Simple*, pp. 1–11, 468–475. O'Reilly Media, 2018.
[2] J. Leino and K.-J. Räihä, "Case amazon: Ratings and reviews as part of recommendations," in *Proceedings of the 2007 ACM Conference on Recommender Systems*, RecSys '07, (New York, NY, USA), p. 137–140, Association for Computing Machinery, 2007.
[3] Shuyu Luo, "Introduction to Recommender System." https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26, December 2018. Last Accessed (2020-03-24).

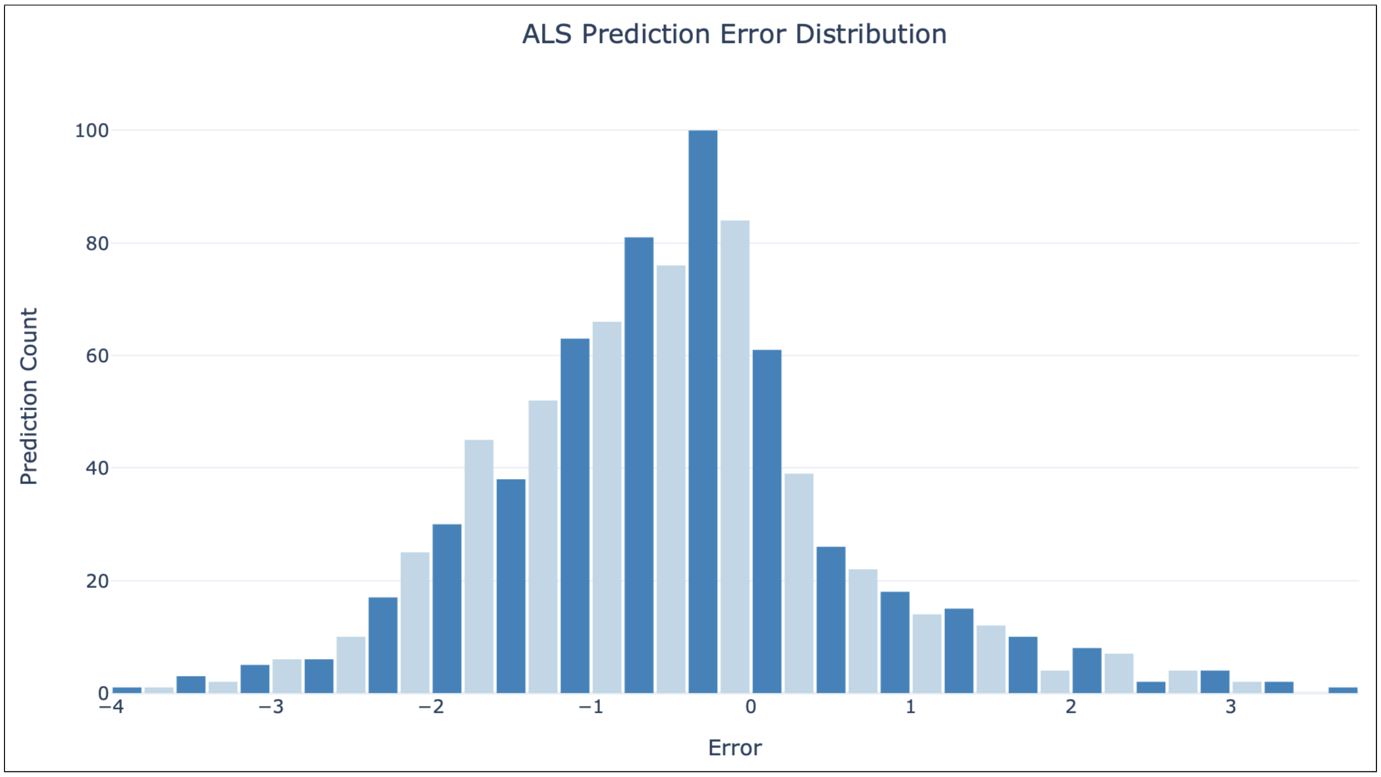


Fig. 4. Error distribution plot demonstrating ALS prediction performance in CARS.

[4] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, (New York, NY, USA), p. 241–250, Association for Computing Machinery, 2000.

[5] Jianmo Ni, "Amazon Review Data." https://nijianmo.github.io/amazon/index.html, 2018. Accessed on 2020-03-24.

[6] J. Ni, Z. C. Lipton, S. Vikram, and J. McAuley, "Estimating reactions and recommending products with generative models of reviews," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (Taipei, Taiwan), pp. 783–791, Asian Federation of Natural Language Processing, Nov. 2017.

[7] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 188–197, Association for Computational Linguistics, Nov. 2019.

[8] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Algorithmic Aspects in Information and Management* (R. Fleischer and J. Xu, eds.), (Berlin, Heidelberg), pp. 337–348, Springer Berlin Heidelberg, 2008.

[9] Anaconda, Inc., "Conda." https://docs.conda.io/en/latest/, 2017. Accessed on 2020-05-14.

[10] Plotly, "Take data science and AI out of the lab. Free the data. Share the knowledge.." https://plotly.com, 2020. Last Accessed (2020-03-24).