International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211
Volume 9, Issue 4, April -2021, Impact Factor: 7.429, Available online at: www.ijaresm.com

Optimization of Recommendation System Design Using Data Pipelining

CH Raja Shaker[1], B Hrithikender Reddy[2]

[1]Department of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu - 600119
[2]Department of Computer Science and Engineering, B.V. Raju Institute of Technology, Narsapur, Medak Dist., Telangana - 502313

--*****************--

ABSTRACT

Users can get suggestions based on their tastes thanks to recommender programs. With the ever-increasing amount of information available on the internet, recommender programs have proven to be an effective method for overcoming information overload. The importance of using recommender schemes cannot be overstated, given their ability to alleviate certain over-choice issues. There are a variety of recommendation frameworks available, each with its own set of methodologies and principles. Various technologies, such as e-commerce, healthcare, transportation, agriculture, and media, have implemented advisory systems.Our study focuses on performance optimization of recommendation systems through the use of customer transformers to build an efficient data pipeline. A recommendation system pipeline is created for property pricing and suggestions with multiple pipeline processing units to produce a computationally faster approach without compromising on the accuracy.

Keywords: recommendation system, data mining, prediction, pipelining, machine learning

INTRODUCTION

Shopping items, books, news stories, music, movies, academic papers, and other essential items have filled many data-warehouses and libraries in both volume and variety [1-2]. Intelligent recommendation systems and efficient search engines will support consumers in this regard. The potential of such schemes to manifest valuable knowledge from a nearly limitless storehouse is the reason for their success and utility [3]. As a result, recommendation services like Amazon, Netflix, and others take the opportunity to learn about their customers' interests and educate them about the things that they are interested in.Despite the fact that these programs vary in terms of the applications they serve, the central method for locating objects of user interest is user interest to object matching [4].

User habits, item attributes, user-item purchases, and other environmental variables such as time, season, and position can all be used to produce recommendations. In the research on recommendation, these are divided into three categories: mutual filtering (recommendation focused solely on user-item interaction information), content-based (recommendation based on user expectations, item preferences, or both), and mixed suggestion models (recommendation based on both interaction information and user and item metadata) [5]. Each of these types has its own set of constraints, such as data sparsity, consumer cold start, and objects [6].

Machine learning has been applied to the realm of information processing and prediction systems as a result of recent developments in the field of deep learning in various technology areas such as machine vision and speech recognition [7]. The general consensus on the effect of incorporating deep learning into recommendation systems is that it improves on traditional models significantly.

LITERATURE REVIEW

Recommendation systems

A recommendation framework typically has a broader application use case, but we've concluded the validation dataset was used as the predictor of application domain for the purposes of categorization depending on how the study was performed. Most current recommendation systems have been validated on statistics from the Media domain. The use of publicly available datasets from Amazon, Netflix, and MovieLens can be credited with this. After that, e-commerce is the next area of focus for deep learning advanced recommendation systems.Researchers have used product analysis datasets from Yelp, Amazon, Slashdot, and other sources to validate their findings. Posts such as reporting, quotes, and scholarly review articles fall under the third group. We categorize studies performed on databases such as Epinions, social marking, and so on under the social group.

Content Based Systems

When no use data is available, deep convolution neural networks are used to produce latent factors for songs from their audio. Using the Million song dataset, the method outperforms simplistic baselines such as linear regression and multi-layer perceptron trained on bag-of-words representation of audio signals (consisting of Last.fm dataset and Echo Nest Taste Profile Subset) [8].

[9] suggested a hybrid model that uses deep belief networks and probabilistic graphical models to extract feature from audio content and render custom suggestions at the same time. Using the Echo Nest Taste Profile Subset, the model was compared to only content-based models as well as a mixed model without deep learning (a music recommendation dataset). On the validation dataset, the model outperforms both the content-based and shared filtering-based baselines.

In the area of quote suggestions in Writings and Dialogues, [10][11] suggest deep learning methods for improving content-based recommendations. Tan et al use an LSTM model to describe the meanings and quotes' scattered value. To learn semantic representation of quotes in the dialogue line, Lee et al merge recurrent neural networks and convolutional neural networks. For the twitter dialogue thread, quotations were taken from Wikiquotes and the Oxford Concise Dictionary of Proverbs.

[12] use GRU-based recurrent neural networks to translate item text into latent features in order to boost collaborative filtering efficiency, especially for cold start. The model is related to a simplified version of a shared subject modeling model on a citation suggestion scheme using two real-world datasets (dense and sparse versions) from CiteULike. The proposed model improved results statistically significantly on both datasets.

[13] use deep neural networks called Deep Cooperative Neural Networks to jointly learn object properties and consumer interactions using review knowledge. A shared layer is often used in the model to connect item functions to context awareness. Using three real-world datasets: yelp ratings, Amazon reviews, and beer reviews, the model is linked to five dependent lines: matrix factorization, probabilistic matrix factorization, LDA, collaborative topic regression, hidden factor as topic, and collaborative deep learning. On all benchmarking datasets, the model outperformed all baselines.

[14] created a news story recommendation algorithm based on a hierarchical focus deep model to solve the problem of editors choosing news articles for the coverage pool for end users of non-explicit selection criteria. This stage of news recommendation comes before the end user's final news recommendation. The editors choose a subset of news articles from a constantly shifting pool of articles sourced from different news feed outlets at this time. There are no hard and fast guidelines for editors when it comes to choosing or dismissing posts. This study employs deep learning to learn the requirements for an editor's creative article selection style.Traditional bag of words methods cannot explicitly solve such a problem. As a result, deep learning attention models are used to understand unique attributes to reflect article style and then determine if the editor likes the article or not.

Collaborative Filtering Based Systems

[15] deal with the sparsity issue in CF approaches as well as the scarcity of auxiliary data in collective subject regression-based approaches. Generalized Bayesian Stacked Denoising Autoencoders were used in the proposed model. On two CiteULike datasets and one Netflix dataset, the model outperforms Matrix factorization and Collaborative topic regression approaches.

[16] is the first paper to provide a method for incorporating deep learning functions such as matrix factorization into CF models. On four real-world datasets, the model is compared to other collective filtering methods that use matrix factorization and shows some efficiency improvement: Book-Crossing, Advertising dataset, MovieLens-100k, MovieLeans-1M, MovieLens-100k, MovieLens-1M, MovieLens-100k, MovieLens-1M, Movie

[17] created a probabilistic rating auto-encoder to conduct unsupervised feature learning and create user profiles from user-item rating data in order to improve collaborative filtering methods. Using the yelp.com dataset, the effects of combining deep learning with traditional collaborative filtering methods including matrix factorization indicate a statistically important increase in rating prediction.

[18] demonstrate that collaborative filtering can be transformed into a sequence prediction challenge, making recurrent neural networks extremely useful. On two recommender system datasets, MovieLens and Netflix, the LSTMs are implemented for CF problems and experimentally compared with k-nearest neighbor and matrix factorization approaches. The efficiency analysis proves how effective machine learning algorithms are for collective filtering as compared to other state-of-the-art models.

[19] used deep learning to isolate the group impact in user's trusted friendships and to initialize user and object latent attribute vectors for trust conscious social recommendations. On two real-world datasets, Epinions and Flixster, the method outperforms most variations of matrix factorization approaches.

In social media network datasets from Epinion and Slashdot, [20] suggested a Bayesian Personalized Ranking Deep Neural Network model for friendship recommendation. The model uses Convolution neural networks to derive latent structural patterns from the input network, then uses Bayesian ranking to make recommendations. Matrix factorization algorithms, Katz similarity, Adamic/Adar similarity, and basic pairwise input neural network are all outperformed by this deep learning-based recommendation method.

Methodology:
Preprocessing is the method of transforming data into the desired format. It consumes a significant portion of a machine learning practitioner's time. Preprocessing and fitting are two separate methods for engineers, but in a manufacturing setting, when we serve the model, there is no difference. It's all about data in and estimation out. That's what pipelines are about. They combine the preprocessing and fitting or forecasting processes into a single procedure. They bring a great deal of reproducibility to the experimental process, in addition to assisting in the development of the model.

The main objectives are to build a system that:

- Reduces latency;
- Is tightly combined with the other components of the device, such as data warehouses, monitoring, and the graphical user interface.;
- Can scale both horizontally and vertically;
- the system communicates via asynchronous, non-blocking message passing.
- Provides efficient computation with regards to workload management;
- Is fault-tolerant and self-healing i.e., breakdown management;
- Supports batch and real-time processing.

Dataset
For our research study, we take into consideration a real-world dataset based on property prices. The dataset keeps records pertaining to property characteristics and constraints such as number of bedrooms, beds, bathrooms, latitude, longitude, customer reviews, prices etc. The dataset contains a total of 26,931 entries with 20 features.

Pipeline Architecture
Many steps in the machine learning workflow start with data preparation (for example, dealing with missing values, scaling/encoding, and feature extraction). We must complete the data preparation process one step at a time. Since we need to prepare both the training and testing results, this can be time intensive. Pipelines help us to speed up this process by compiling the planning steps and making model tuning and tracking easier. The Pipeline class in Scikit-Learn offers a framework for performing a sequence of data transformations accompanied by an estimator.

The Scikit-Learn pipeline class has multiple advantages:

- We call fit and predict only once on the data to fit an entire sequence of estimators.
- We can perform a grid search over parameters of all estimators in the pipeline.

- Pipelines help to avoid data leakage from the testing data into the trained model during cross-validation. This is achieved by ensuring that the same samples are used to train the transformers and predictors.

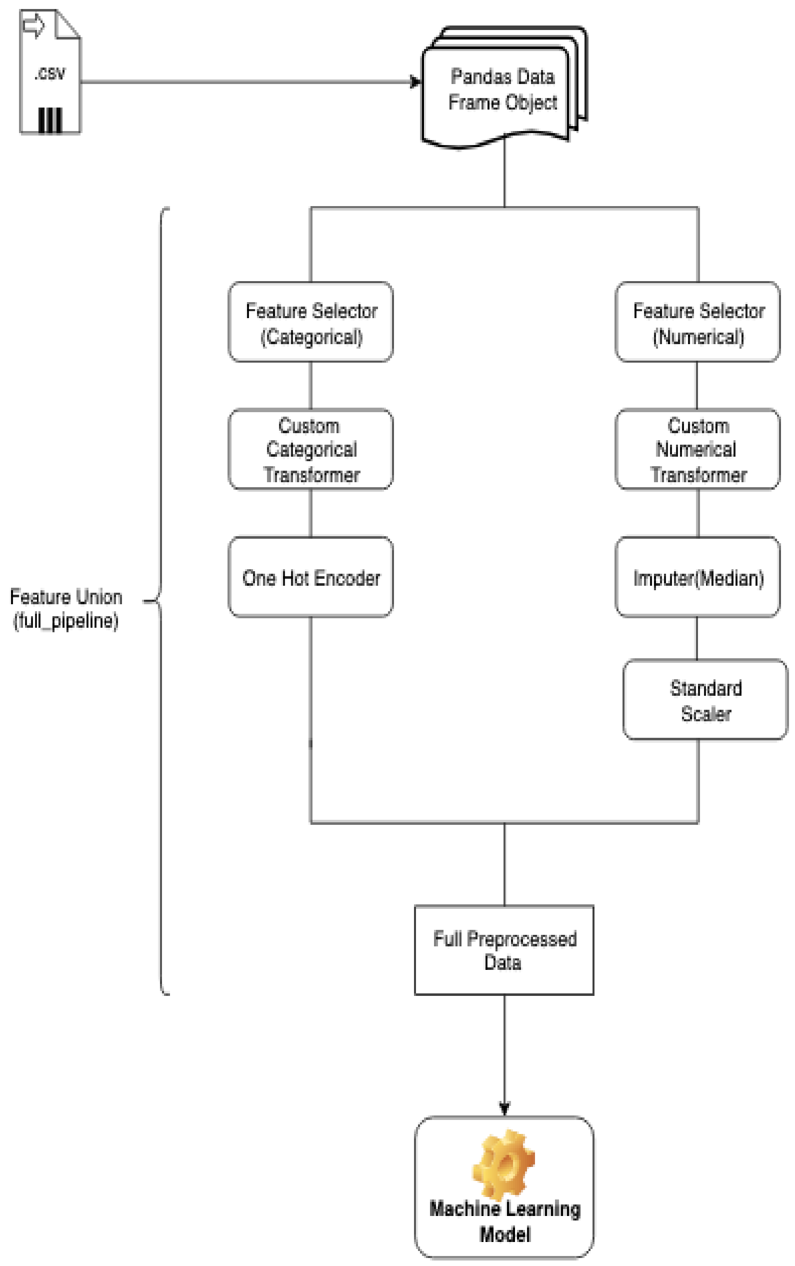
.csv
Pandas Data Frame Object
Feature Selector (Categorical)
Feature Selector (Numerical)
Custom Categorical Transformer
Custom Numerical Transformer
One Hot Encoder
Imputer(Median)
Standard Scaler
Feature Union (full_pipeline)
Full Preprocessed Data
Machine Learning Model

Fig. 1: Machine Learning Pipeline Architecture

Data planning is the first step in the deep learning workflow (for example, dealing with missed values, scaling/encoding, and feature extraction). One phase at a time, we must finish the data planning process. This will take a long time because we need to schedule both the preparation and the research reports. Pipelines aid in the speeding up of this project by collecting the preparation steps and facilitating model tuning and recording. Scikit-Pipeline Learn's class provides a basis for executing a series of data transformations using an estimator.

Stateless transformers and stateful transformers are the two types of transformers. Stateless transformers consider each sample separately, while stateful transformations focus on previous results. If a stateful transformer is needed, use the save state on fit () process. Self can be returned by both stateless and tasteful transformers.

In our experiment, we construct a transformer class that inherits from the Base Estimator class, which gives us free access to the getparameters() and setparameters() methods, enabling us to use the new transformer in the quest for the best parameter values.

We may construct different pipeline processing units for separated calculation of categorical and numerical data using these classes. If our custom transformer has handled all of these functions, they will be transformed to a Numpy list and pushed to the categorical pipeline's next and final transformer. A simple one-hot encoder in Scikit-learn that returns a dense description of our pre-processed files.

Similarly, after our custom numerical transformer in the numerical pipeline has handled all of these functions, the data will be translated to a Numpy array and forwarded to the next stage in the numerical pipeline, a scikit-learn imputer. The imputer will measure the column-wise median and substitute the necessary median values for any Nan values. The data will then be sent to the numerical pipeline's final converter, a basic scikit-learn Standard Scaler.

We need a way to merge our numerical and categorical transformers horizontally now that we've written them and identified our pipelines. We can do this with scikit-Feature Union learn's class. Giving it two or more pipeline objects with transformers creates a function union class object. When you use the fit transform method on a function union object, it moves data down the pipelines separately, then combines and returns the output. Since the first step in each of our pipelines is to remove the appropriate columns for each pipeline, merging them using feature union, and fitting the feature union object to the whole dataset, the appropriate set of columns will be moved down the appropriate set of pipelines and merged after they are transformed.

The computations are parallelized, i.e., the transition happens in parallel, making it computationally faster and more effective than linear processing cycles. The Feature Union object accepts only transformer-only pipeline elements. We generate another pipeline object, transfer the preprocessing pipeline object as the first step, and incorporate a machine learning model as the final step, since a machine learning model is an estimator. The entire preprocessed dataset, which will be the product of the first stage, will simply be transferred down to my model, allowing it to work like every other scikit-learn pipeline where machine learning techniques will be implemented as discussed in next section.

Techniques:

Linear Regression:

In statistics, Linear Regression (LR) is a straightforward way to show the relation between a dependent variable and at least one independent variable. LR was the principal method of regression analysis to be extensively developed and commonly used in practical applications (Yan and Su 2009). LR demonstrates the relation between two variables by applying a straight state to the knowledge centered on them. One variable is considered as autonomous, and the other as dependent. An LR1 line has a condition of the structure:

$$Y = bX + a \tag{1}$$

here X is the independent and Y is the dependent variable. The slope of the line is b and a is the intercept (the value of y when x = 0).

Support Vector Regression:

For continuous variables, Support Vector Regression works well. The SVR matches the best line under a threshold or predefined error size, unlike other linear models that minimize the error between the expected and real value. By treating the line that passes through the error boundary as a possible support vector, the SVR aims to estimate the uncertain value. The disparity between the expected and real values of lines that do not travel through the error boundary are not taken into account unless they are greater than the error threshold. SVR is a useful model for dealing with continuous valued functions and compensating for error and complexity in prediction.

Random Forest:

Random Forest is an ensemble of learning algorithms based on methods. RF consists of a series of classifiers for tree. Every tree is composed of nodes and edges. The received group classifies new data points through a majority within each classification model 's predictions, as shown in Fig. 2. This approach incorporates a bagging cycle (bootstrap aggregation) and a set of random splits. Each tree is extracted from the data set from a separate bootstrap sample, and each tree categorizes the data. The final outcome is a majority vote between the trees.

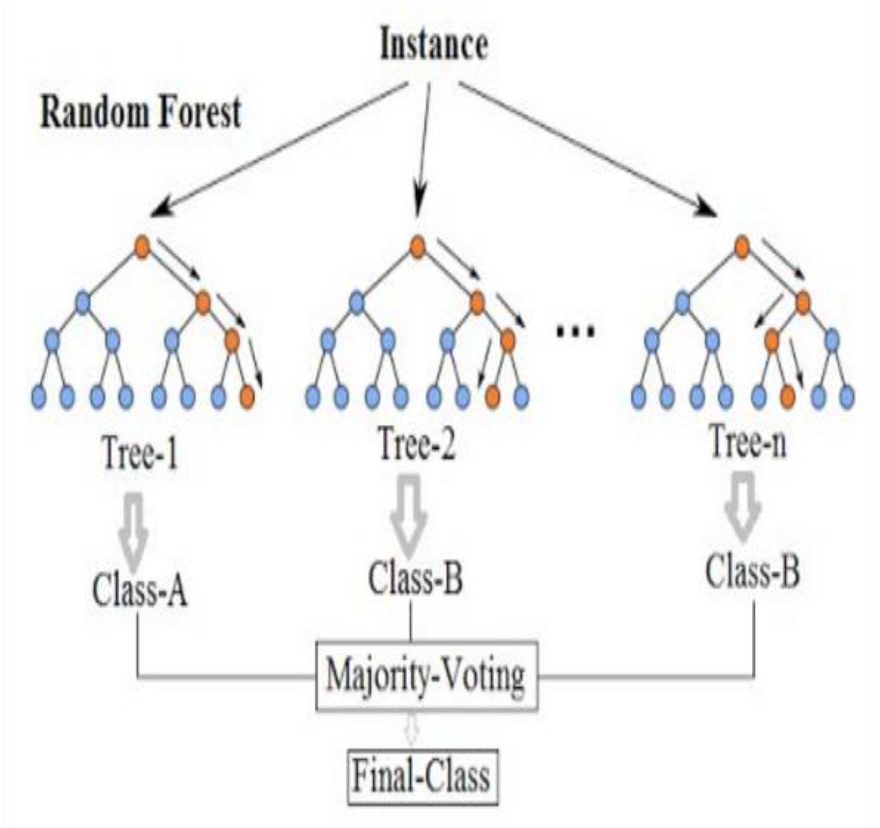

Fig. 2. Random Forest Algorithm

The random forest algorithm is defined by the following steps:

- Construct samples of the data from k trees bootstrap.
- For each of the bootstrap samples grow an unpruned tree.
- Randomly sample n-try of the predictors at each node, and pick the best split among those factors.
- Predict new data through a combination of the k tree predictions

It is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. As mentioned above, the performance of a model significantly depends on the value of hyper parameters. Note that there is no way to know in advance the best values for hyper parameters so ideally, we need to try all possible values to know the optimal values. Doing this manually could take a considerable amount of time and resources and thus we use Grid Search CV to automate the tuning of hyper parameters.

Grid Search CV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method. Hence after using this function, we get accuracy/loss for every combination of hyper parameters and we can choose the one with the best performance.

RESULTS

Before designing the pipeline, we start off with visualization to gain a better understanding of the patterns and distribution. This also helps in detecting outliers from the data space. As shown in Fig. 2, scatter plot is used to display the review allocation with respect to the location (latitude, longitude) and prices.

Further in Fig. 3, the correlation matrix has been visualized to demonstrate correlation scores between the parameters considered for prediction and pipeline designing. To improve the insight generation at hand, we using manual feature engineering to create some additional parameters that could be useful towards better relation patterns such as *bedroom per person*, *bathroom per person*, and *days on Airbnb.*

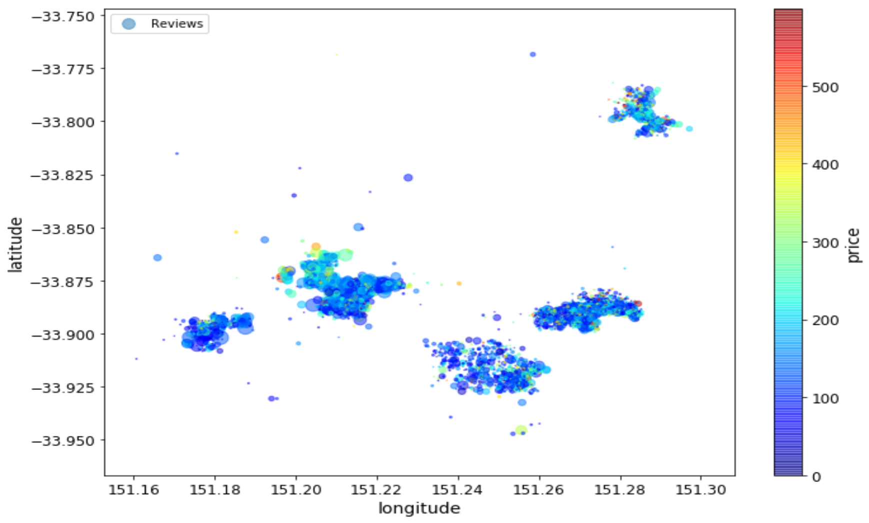

Fig. 3:Scatter plot analysis based on dataset distribution

Next, we use customer transformer classes *Combined Attributes Adder* and *To Pandas DF* to implement a pipeline processing unit for reading data array and converting it to dataframe for feature analysis followed by preprocessing. The processing units are integrated into the main pipeline which is sent for data mining.

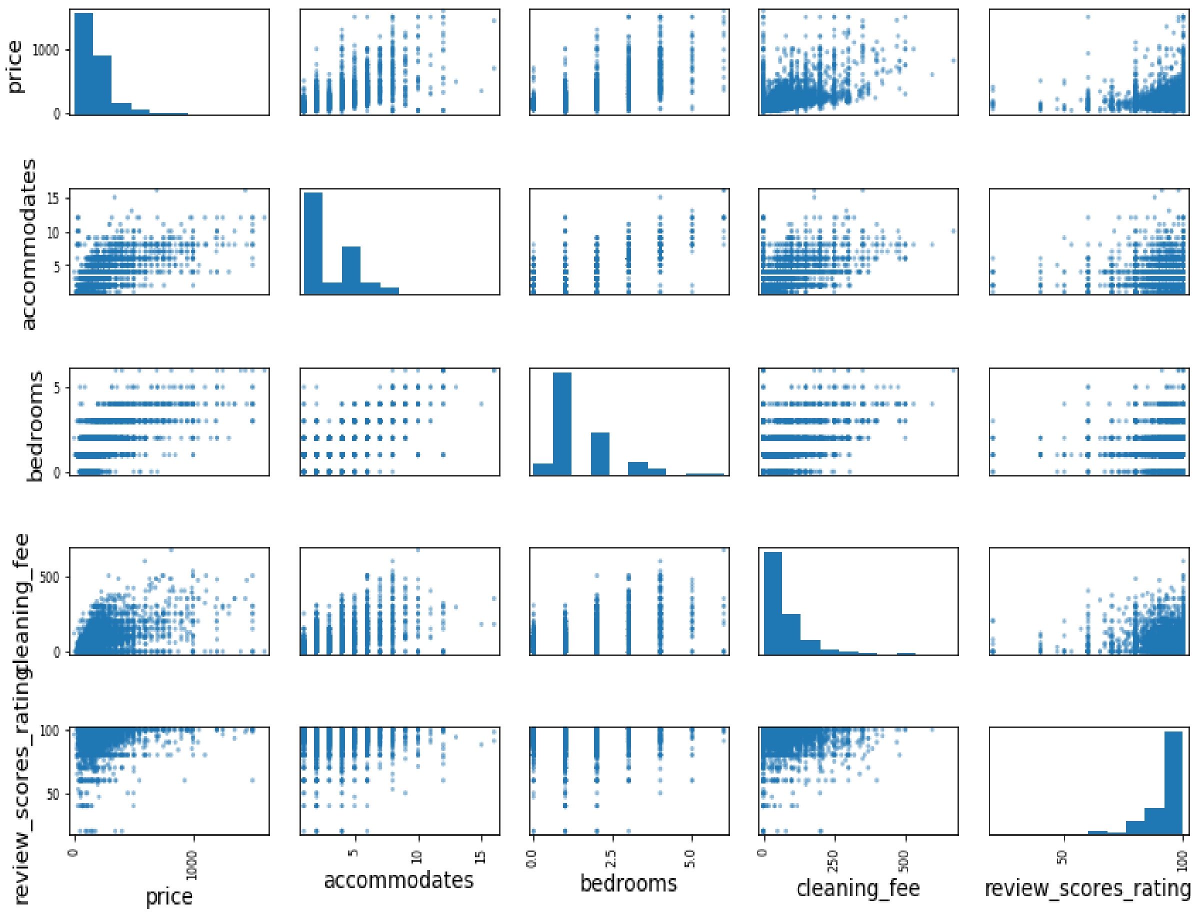

Fig. 4: Correlation matrix displayed using plots

For data mining, we start off with linear regression, random forest and linear support vector regression. The results are displayed in Table. I which are achieved from the initial modelling into the pipeline design.

Table I: Machine learning algorithms performance

	Mean	Standard Deviation	RMSE
Linear Regression	8.23	0.16	127.60
Linear SVR	23.26	1.14	115.17
Random Forest	102.19	6.57	43.16

For hyperparameter tuning, we implement two techniques which is grid search and randomized search for picking out the best combinations of hyperparameters leading to improved performance. We further test random forest algorithm due to its comparatively better performance on sample set which is shown in Table. II. The results shown are the best four combinations generated; both search algorithms for fine tuning give close results.

Table II Hyperparameter tuning using grid search and randomized search

Grid Search			Randomized Search		
RMSE	**Max Features**	**Estimators**	**RMSE**	**Max Features**	**Estimators**
99.00	8	30	98.00	7	180
99.29	6	30	98.22	7	122
102.38	8	10	100.66	5	21
102.40	6	10	101.22	3	72

CONCLUSION

The paper successfully implements a data pipelining architecture on a recommendation system framework. The model is production ready and can be used for creating automated workflows which can be very beneficial to businesses. As data pipelines are a standard in large software organizations, the implementation of automated models and pipelines can be extremely useful to small scale organizations due to less workforce and low maintenance times. The machine learning prediction models implemented provided decent performance with fine tuning giving expected performance upgrade which tend to get better with input data.

REFERENCES

[1]. A. Singhal and J. Srivastava, “Research dataset discovery from research publications using web context,” Web Intell., vol. 15, no. 2, pp. 81–99, 2017.
[2]. A. Singhal, R. Kasturi, and J. Srivastava, “DataGopher: Context-based search for research datasets,” in Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration, IEEE IRI 2014, 2014, pp. 749–756.
[3]. A. Singhal, “Leveraging open-source web resources to improve retrieval of low text content items,” ProQuest Diss. Theses, p. 161, 2014.
[4]. A. Singhal, R. Kasturi, V. Sivakumar, and J. Srivastava, “Leveraging Web intelligence for finding interesting research datasets,” in Proceedings - 2013 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2013, 2013, vol. 1, pp. 321–328.
[5]. J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, “Recommender system application developments: A survey,” Decis. Support Syst., vol. 74, pp. 12–32, 2015.
[6]. S. Lakshmi and T. Lakshmi, “Recommendation Systems: Issues and challenges,” Int. J. Comput. Sci. Inf. Technol., vol. 5, no. 4, pp. 5771–5772, 2014.
[7]. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[8]. A. van den Oord, S. Dieleman, and B. Schrauwen, “Deep content-based music recommendation,” Electron. Inf. Syst. Dep., p. 9, 2013.
[9]. X. Wang and Y. Wang, “Improving Content-based and Hybrid Music Recommendation using Deep Learning,” MM, pp. 627–636, 2014.
[10]. J. Tan, X. Wan, and J. Xiao, “A Neural Network Approach to Quote Recommendation in Writings,” Proc. 25th ACM Int. Conf. Inf. Knowl. Manag. - CIKM ’16, pp. 65–74, 2016.

[11]. H. Lee, Y. Ahn, H. Lee, S. Ha, and S. Lee, "Quote Recommendation in Dialogue using Deep Neural Network," in Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval - SIGIR '16, 2016, pp. 957–960.

[12]. T. Bansal, D. Belanger, and A. McCallum, "Ask the GRU," in Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16, 2016, pp. 107– 114.

[13]. L. Zheng, V. Noroozi, and P. S. Yu, "Joint Deep Modeling of Users and Items Using Reviews for Recommendation," 2017.

[14]. X. Wang et al., "Dynamic Attention Deep Model for Article Recommendation by Learning Human Editors' Demonstration," in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17, 2017, pp. 2051– 2059.

[15]. H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative Deep Learning for Recommender Systems," in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1235–1244.

[16]. S. Li, J. Kawale, and Y. Fu, "Deep Collaborative Filtering via Marginalized Denoising Auto-encoder," in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management - CIKM '15, 2015, pp. 811–820.

[17]. R. Devooght and H. Bersini, "Collaborative Filtering with Recurrent Neural Networks," Aug. 2016.

[18]. H. Liang and T. Baldwin, "A Probabilistic Rating Auto-encoder for Personalized Recommender Systems," in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management - CIKM '15, 2015, pp. 1863–1866.

[19]. S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, "On Deep Learning for Trust-Aware Recommendations in Social Networks," IEEE Trans. Neural Networks Learn. Syst., vol. 28, no. 5, pp. 1164–1177, 2017.

[20]. D. Ding, M. Zhang, S.-Y. Li, J. Tang, X. Chen, and Z.-H. Zhou, "BayDNN: Friend Recommendation with Bayesian Personalized Ranking Deep Neural Network," in Conference on Information and Knowledge Management (CIKM), 2017, pp. 1479–1488.