City, University of London

M.Sc. in Software Engineering with Cloud Computing

Project Report

2023

Prediction of Product Prices and estimated Sale time based on marketplace data

Student:

Riddhi More

Supervised by:

Dr. Konstantin Pozdniakov

2nd October 2023

Declaration

By submitting this work, I declare that it is entirely my own except for those parts duly identified and referenced in my submission. It complies with any specified word limits, the requirements and regulations detailed in the assessment instructions, and any other relevant program and module documentation. In submitting this work, I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Program Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct. In this modern era, there is a relentless pursuit of innovation. Machine learning and artificial intelligence have become formidable tools that are driving vast sections of various industries. Among these, the prediction of the price and sale time of the e-commerce products presents an intriguing challenge. I imagined myself as a novice in the field of machine learning plunging into the world of Artificial Neural Networks (ANNs), and the goal of predicting e-commerce product pricing and sale time became an exhilarating journey in the field of machine learning. In this chapter, I have started my adventure as an absolute beginner in this intriguing subject. A trip that I know will be difficult but also incredibly gratifying, teaching me a vital lesson about learning something new that I will apply to my professional life.

Signed: **Riddhi More**

Abstract

Online marketplaces are growing rapidly, with increasing demand for e-commerce products swiftly changing the needs and competitive value of marketplaces. It is a necessity for these online platforms to redefine the way products are sold on them. The rise of e-commerce has not only democratized global markets but has also empowered businesses to explore new horizons with unprecedented ease.

Based on the above idea, this paper explores the fundamental topic for most researchers in the financial and technical domains: predicting price and sale time for marketplace e-commerce products. It implements an artificial neural network (ANN) machine learning model that offers to make predictions for the sale time and price of the product, just like humans, but with more precision. This goal is achieved based on complex, irrelevant, and/or partial information using historical purchasing data and demonstrates impressive performance in various domains. Furthermore, this paper investigates the dimensionality retainability feature of the auto-encoder. Thus, contributing to the success of the price and sale time prediction machine learning model.

Finally, the model evaluates the performance of the proposed models and tunes the model to fit in as a predictive model for any product published on the Marketplace.

Keywords: marketplace, e-commerce, price prediction, sale time prediction, ANN model, autoencoder, machine learning.

Acknowledgements

Primarily, I want to express my appreciation to Dr. Konstantin Pozdniakov for his time and help throughout the entire development of the project. His unwavering guidance and regular review meetings have been crucial in helping me achieve my goal of advancing the project in machine learning. Under his mentorship, I have been motivated to learn concepts and strive for excellence in my work.

would also like to extend my thanks to the author of the book "Projects in Computing and Information Systems: A Students Guide" as well as the author of "Researching Information Systems and Computing" (Oates, 2006). These books have played a role in introducing the framework for computing projects and developing essential skills needed for their successful execution. They have not only helped me understand practices and critical components of projects within an academic context but have also empowered me to explore and develop my own ideas over time. They have emphasized the importance of delivering projects that can be useful and beneficial to others.

Furthermore, I would like to thank all the outstanding lectures in the M.Sc. Computer Science Program. Modules like Introduction to Big Data, Project Management, Cloud Computing, RMPI, and a few more built the foundations for this work and sparked my interest in varied other aspects of the computer science field.

Finally, I would like to thank the teaching assistants, technical staff, and administration of City University of London for their patience, support, and readiness to help me in any conceivable way.

Table of Contents

Table of Figures

Chapter 1: Introduction and Objectives

The rise of marketplace digital commerce has ushered in an era of unrivalled growth in online retail thanks to the broad availability of internet connections and digital payment infrastructure, which have allowed individuals all over the world to shop from the comfort of their homes. In 2022, worldwide retail ecommerce sales will reach $5 trillion for the first time, making up more than a quarter of all retail sales, predicts eMarketer. And despite slowing growth, overall spending will surpass $7 trillion by 2025 (BigCommerce Pvt. Ltd, 2023).

Marketplace E-commerce platforms' cheap barrier to entry and global accessibility have, however, also sparked fierce competition between sellers striving for market supremacy and gaining an advantage over competitors. Utilizing advanced analytics to optimize retail operations has become crucial in this chaotic economic environment. In particular, accurate demand forecasting makes it possible to implement quick pricing decisions, streamline supply chains, and meet customers' demands for quick fulfilment. Thankfully, recent advances in machine learning offer sophisticated techniques to glean useful insights from the massive datasets produced by contemporary e-commerce.

Among the most crucial business metrics to predict is the velocity of sales for online catalogue items. Accurately modelling the time to sell across products facilitates just-in-time inventory acquisition, organization planning, and dynamic markdowns to accelerate the turnover of merchandise. While simpler time series techniques have traditionally been employed ('Prediction of Product Prices and estimate sale time,' no date; Ensafi et al., 2022), they fail to capture the multivariate complexities of real-world transactional data. Modern deep neural networks now offer the requisite representational power to account for intricate dependencies between sales rate, pricing, product attributes, promotions, seasonality, and other factors. Though effective at forecasting the speed of sales, optimizing price remains an equally critical driver of revenues. Jointly modelling both objectives is crucial but lacks robust solutions. (Polamuri, Srinivas and Krishna Mohan, 2019) (Vijh *et al.,* 2020)

This thesis offers solutions for the research challenge pertaining to the creation of a machine learning framework tailored to optimize sales time forecast and pricing predictions, with empirical validation carried out on substantial datasets obtained from the Mercari marketplace.(Mercari, 2020) While previous works have focused separately on forecasting velocity or price optimization, modelling these interconnected objectives holistically across massive product catalogues is novel. The proposed approach leverages multi-task deep networks to share latent representations between the two tasks while capturing complex multivariate relationships. Benchmarking on live e-commerce data will quantify improvements in both objectives compared to single-task models. Given the vital importance of sales and pricing to online retail profitability, this work will provide an invaluable production-ready solution.

Furthermore, the model for price and sale time prediction is built on an open-source e-commerce marketplace called Mercari, the data for which has been published and referenced on the Kaggle platform(Mercari, 2009). This dataset has a wide variety of records ranging from electronics, products for men, women, and kids, to vintage and collectibles, all under one dataset. This series of datasets is aimed to improve the robustness of the model to predict the price and sale time of e-commerce products.

1.1 Aim and Objectives

The research of this project aims to answer the question: "To create architecture for price prediction and sale time forecast using neural network machine learning model based on massive open-source e-commerce marketplaces data.

This study endeavours to investigate machine learning techniques for the intertwined objectives of sales time forecasting and price prediction in an expansive e-commerce marketplace. The primary objectives are as follows:

- Conduct an exhaustive survey, analysis, and critique of the research related to the price and sale time prediction for various ecommerce products.
- Scrutinize varied methodologies for feature extraction and selection employing artificial intelligence, including principal component analysis, autoencoders, and sparsity-inducing regularization, to mitigate the curse of dimensionality and distil the essence of predictive signal from massive, high-dimensional data.
- Construct and validate a working prototype leveraging multi-task deep neural network architectures to capture intricate multivariate relationships and interdependencies between sales time and optimal price points, as demonstrated through rigorous training and testing on live e-commerce transactional data at scale.
- Employ rigorous quantitative performance metrics such as Mean Squared Logarithmic Error (MSLE), Mean Squared Error (MSE) and comparative benchmarks to existing models to comprehensively validate the efficacy and generalizability of the proposed approach on real-world data.

To conclude, while prior works have primarily concentrated on isolated modelling of either forecasting velocity or price optimization, this thesis aims to address the open research gap of unified learning across both objectives systematically, synthesizing ideas from literature surveys, feature engineering, model development, and evaluation across massive marketplace data to create a demonstrably improved solution for maximizing both the rapidity of sales and revenue generation for e-commerce entities.

1.2 Forecasting Sales Velocity and Optimal Pricing

In today's digital era, the marketplace has grown exponentially, with online sales becoming a major revenue source for diverse businesses and entrepreneurs. However, in such a competitive landscape, understanding how quickly inventory sells is critical for retailers' survival and profitability. Accurate demand forecasting enables better decisions around pricing, marketing spending, and inventory planning and drives higher earnings. For perishable goods, sales velocity predictions can help move stock before expiration.

With advances in machine learning, sales time predictions are now possible to optimize delivery times and boost customer satisfaction. Retailers can leverage ML algorithms to forecast customer demand and plan the most efficient delivery routes and warehouses. This boosts efficiency and reduces costs for both buyers and sellers. ML models can uncover patterns in historical data to predict future sales, estimate optimal price points, and account for market trends, seasonality, promotions, and product attributes.

Across industries like retail, e-commerce, and marketing, machine learning has been widely applied for sales forecasting tasks. In retail, ML improves demand planning, pricing strategies, and inventory optimization. In e-commerce, ML enhances search, recommendations, and customization. In marketing, ML segments consumers and targets ads.

Specific research has focused on predicting e-commerce sales velocity usin(Wang *et al.,* 2022)g machine learning. used deep learning to estimate the probability that a product will sell within a time limit. Raza et al. combined ML with time series modelling for sales rate predictions. Ensafi et al. found that artificial neural networks, especially ARIMA, improve forecast accuracy. However, unlike previous work, this thesis uniquely addresses joint sales time and pricing predictions for Marketplace data.

1.3 Beneficiaries

By harnessing the power of ANNs for precise price and sale time predictions, coupled with innovative autoencoder-based feature reduction techniques, this project aims to revolutionize decision-making processes for e-commerce companies, retailers, and financial analysts, fostering profitability, competitive advantage, and data-driven insights."

This project aims to revolutionize decision-making processes for retailers, financial analysts, and e-commerce companies to gain a competitive advantage by providing data-driven insights on precise price and sale time predictions and fostering profitability in businesses.

Retailers: The outcomes of this project will help retailers gain insights into market demand, enhancing their pricing strategy and inventory management based on sale time prediction, improving their competitiveness and customer satisfaction.

Supply chain managers in various industries can benefit from knowing more accurate demand forecasting, especially critical for the product's expiration date and sale target date.

This project will be a great asset for small and medium-sized businesses (SMBs) to improve pricing strategies and forecast their sales, helping them compete with the larger competitors in the market and gain profits with small capital investments.

Finally, are the data scientists and researchers in the fields of ML and AI. This project will be used as a benchmark for similar predictive modelling. This hybrid technique and performance improvement strategy will serve as valuable starting points for others.

Market analysts can use project insights to gain a better understanding of pricing trends, customer demands, and sales strategies for various products.

This research adheres the below mentioned work plan divided into the four key milestones.

1. Data Collection and Pre-processing

Perform a rigorous survey of academic literature to identify appropriate datasets for the task. Define objective success metrics and assemble training and testing data, ensuring proper pre-processing, and formatting. The first step in the machine learning is to collect data for training the predictive model. ML systems prediction depends on the quality of the training data. Raw data about products from the real world is frequently incomplete, inconsistent, and deficient in specific behaviours or trends. They are also likely to have an unformatted, more textual presentation. As a result, once collected, the data is pre-processed into a format that the machine learning algorithm can utilize to build the model. Hold meetings with research supervisors to evaluate progress and align on data quality standards. Refine and enrich datasets through cleaning, normalization, imputation, and feature engineering.

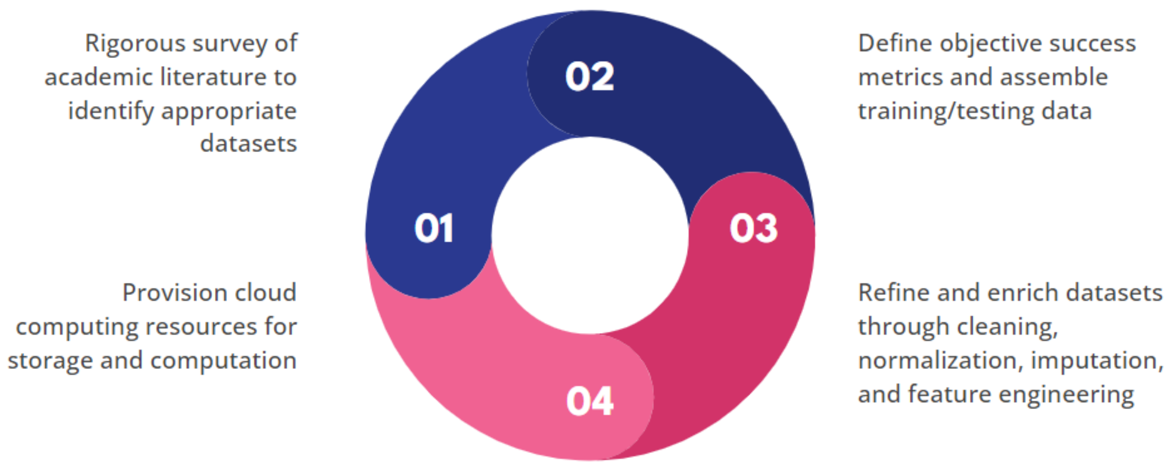

Rigorous survey of academic literature to identify appropriate datasets
02
Define objective success metrics and assemble training/testing data
01
03
Provision cloud computing resources for storage and computation
04
Refine and enrich datasets through cleaning, normalization, imputation, and feature engineering

Figure 1: Workplan of Data Collection and Pre-processing

2. Model Development: Finalize a neural architecture after deliberating the optimal components and hyperparameters. Procure and allocate the necessary cloud-based compute resource; preprocess raw features by executing techniques like encoding, transformation, and dimensionality reduction. Implement and train deep neural networks on transformed data to learn complex relationships. Iteratively evaluate models on validation data and refine as needed to improve generalization. Extend multi-tasking and transfer learning approaches to enhance predictive capabilities.

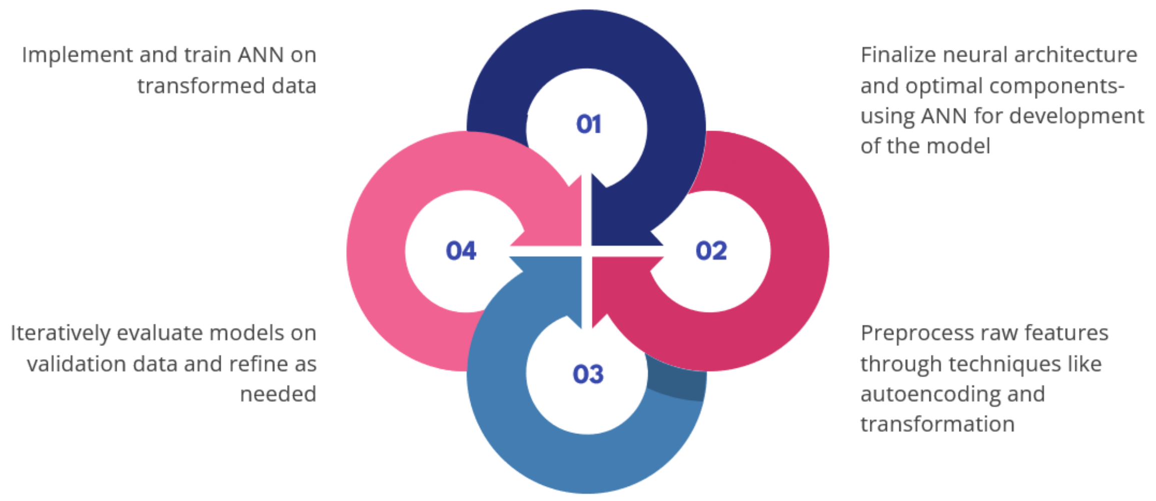

Implement and train ANN on transformed data
01
Finalize neural architecture and optimal components- using ANN for development of the model
04
02
Iteratively evaluate models on validation data and refine as needed
03
Preprocess raw features through techniques like autoencoding and transformation

Figure 2: Workplan of Model Development

3. Model Evaluation: Strategically split pre-processed data into training and evaluation subsets. Feed validation data into trained models to quantify predictive accuracy through technical metrics and visualizations. Perform error analysis by inspecting mispredictions to diagnose limitations. Tune neural architectures and training methodologies based on the resultant insights. Thoroughly assess real-world efficacy using live marketplace data.

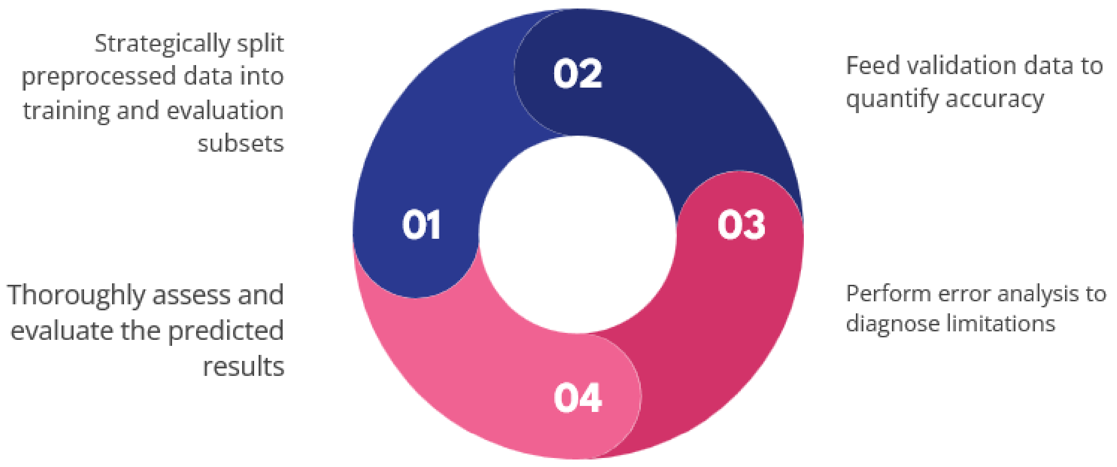

Strategically split preprocessed data into training and evaluation subsets
02
Feed validation data to quantify accuracy
01
03
Thoroughly assess and evaluate the predicted results
04
Perform error analysis to diagnose limitations

Figure 3: Workplan for Model Evaluation

4. Final Evaluation: Rigorously audit end-to-end model behaviour using test data. Measure against quality metrics and baseline methods. Review the results of meetings with supervisors and incorporate suggestions. Synthesize the process and outcomes into a final project report and presentation.

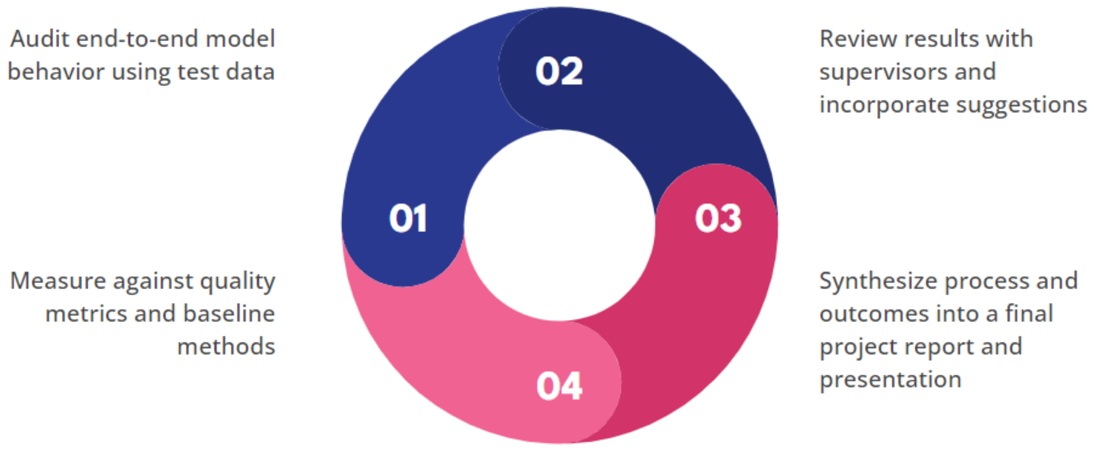

Audit end-to-end model behavior using test data
02
Review results with supervisors and incorporate suggestions
01
03
Measure against quality metrics and baseline methods
04
Synthesize process and outcomes into a final project report and presentation

Figure 4: Workplan for Final Evaluation

The detailed representation of the project plan (i.e., Gantt chart) followed while developing the project, including the documentation timeline for the project report is presented in Appendix C –The Work Plan (Gantt Chart).

1.4 Major changes outlined during the development of the project

This section highlights the changes introduced in the development of the project's price and sale time prediction model

- The autoencoder for feature reduction implements additional functionality called EarlyStopping to improve the performance of this model. In machine learning, this is one of the most widely used techniques for regularization that helps overcome the problem of overfitting. Geoffrey Hinton called it a "beautiful free lunch.", because of its simplicity and efficient regularization technique (B. Chen, 2020). The below diagram shows the process of translating the model.

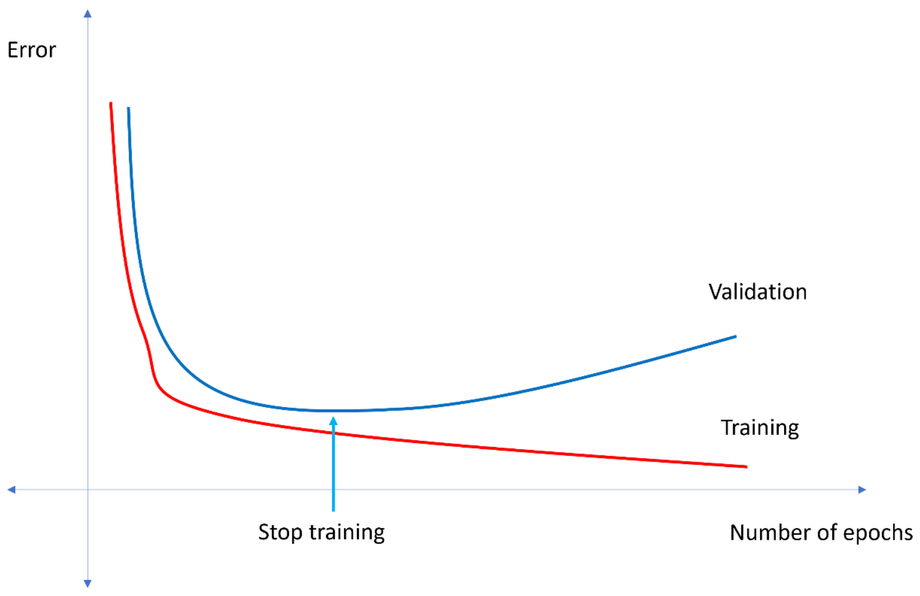


Figure 5: The process of translating the model

While the model is trained, as soon as it encounters the lowest validation error, or the validation error is not decreasing after its patience ratio it will stop training the model. As seen above the validation loss stops decreasing and starts to increase. This problem of overfitting is avoided in the price and sale time prediction model. This feature is explained in chapter 3.2.4Autoencoder for feature reduction

- Whilst attempts are made to hyper-tune the model, the model will not be implemented utilizing the user interface mentioned in the project proposal. The goal of this advancement is to construct a more precise model for predicting prices and estimating sale times. Implementing more powerful machine learning capabilities gives this project a more realistic scope, which improves profitability and sustainability for both B2B and C2C scenarios.
- Hyperparameter of the model is processed through tuning and optimization. These hyperparameter impact the performance and generalization ability of the model. More detailed explanation is covered in chapter 3.4 belowHyperparameter tuning for Optimization.

The section below outlines the structure of the report for price and sale time prediction. Here, a combination of the design, creation, and experiments (Briony J. Oates, 2006) research processes is used to meet the objectives.

Chapter 1: Introduction: This chapter covered the objectives, beneficiaries, and work plan for research that will be performed are composed according to the prototyping (Dawson, 2005) software development process.

Chapter 2: Critical Context

This chapter covers findings from existing literature review, overview of various machine learning models developed to predict the numbers that vary from stock price prediction to sales forecast. Exploring number of well-known methodologies and introducing tools that support the ANN methodology.

Chapter 3: Methods

This chapter provides thorough insights into the research strategy followed and discusses the ANN model implementation with the feature reduction technique and other techniques that helped achieve the goal of this project.

Chapter 4: Results

This section presents the experimental result and explain the significant changes, introduced to support runtime exceptions and conditions necessary to improve or achieve the desired result.

Chapter 5: Discussion

In this chapter, the report compares the objectives set in the first chapter with the outcome of the research and the way they were accomplished.

Chapter 6: Evaluation, Reflection, and Conclusion

The concluding chapter evaluates the success of the project considering the above chapters, highlighting the achievements of the research, and discussing how it has helped in making the project successful, supported by a brief personal reflection.

1.5 Conclusion

This chapter establish the research question the project aims to address to create a model to predict product prices and estimated sale times on marketplace data based on the AI (artificial intelligence) approach. This topic also justified the relevance of the topic with the recent increase in interest in the processing of high volumes of unstructured and semi-structured data, where ANN machine learning can bring value to applications for predicting the price and sale time of an e-commerce product by filtering the features using feature extraction techniques and reducing the dimensionality dynamically.

Chapter 2: Critical Context

The research based on forecast has always been important to many researchers, an approach to the problem in general, people try to explore marketing areas relating behaviour of new customer, price, and sale of the product through methods based on judgment and based on statistical Sources. (Armstrong and Brodie, 1999) documents few approaches based on these categories for forecasting various marketing concepts. In an era of data-driven decision-making, this project addresses the pressing need to harness artificial intelligence for precise predictions of product prices and sale times in the dynamic landscape of online marketplaces. By systematically evaluating existing methodologies and exploring advanced feature extraction techniques, this research contributes vital insights to enhance the accuracy and efficiency of AI-driven predictions. also offering valuable implications for industries reliant on e-commerce data analytics.

Further section depicts detailed research conducted for prediction using machine learning approach.

2.1 The Machine Learning model for predictions

During this study, relevant papers were analysed to understand the scope of the study on this topic of price predictions. The previous experience gained from the development of other machine learning models in similar contexts lends credibility to this piece of literature covered in this chapter.

Scholarly studies have investigated the domain of forecasting sales for e-commerce products, employing a wide range of methodologies, most notably time series analysis, causal regression, and the spectrum of machine learning paradigms. The meticulous construction of a structured **time-series forecast using a linear regression model** intertwined with Taobao search data by (Li, Ji, and Liu, 2018) is an illustrative case in point within the former approach. This intricate combination is planned with the intention of forecasting the amount of apparel sold on the Taobao platform. The objective of this empirical project was to construct a real-time forecasting mechanism that used online search data in conjunction with a structured time series model to forecast the number of sales of clothes. The equipoise of linear regression, a venerable analytical framework, is used in the pursuit of this prediction quest. This approach, well-known for its capacity to forecast continuous target variables supported by one or more numeric or non-numeric input features, lays out the effects of features on the target variable in plain terms. It is nonetheless important to note its limits as a tool best suited for basic model baselining that relies on the implicit assumption that predictors and the target variable have a linear relationship. Any departure from this premise not only signals a potential decline in model performance but also jeopardizes the model's ability to identify complex, nonlinear dynamics hidden in the data.

The complex world of e-commerce, especially in pricing and sale time prediction, reveals a labyrinthine tapestry made up of thousands of varied data streams that are drawn from various sources inside the market. This scenario emphasizes the need to resolve any aberrations, notably the presence of outliers, as it necessitates an unadulterated data strategy to preserve the critical details necessary for detailed product identification. If left unchecked, the negative impact of these outliers extends to prediction accuracy, casting a troubling shadow on the trustworthiness and fidelity of the resulting models (Livingstone, Manallack and Tetko, 1997). To cross the perilous terrain of model sensitivity to these data abnormalities, a prudent recalibration of methodology is required in the crucible of e-commerce predictive analytics, including strong regression approaches or savvy data preparation tactics.

Recent research in price prediction using machine learning for a C2C e-commerce company in Asia has been conducted by Chada (Chada, 2019), Multiple machine learning models were employed in this work to forecast the prices of used products based on different sets of variables. To manage unstructured parameters, feature extraction techniques were used, changing the data into formats acceptable for model processing. The findings emphasized the importance of product attributes in forecasting prices. Modern machine learning methods such as **the gradient boosting framework, the Light GBM, and ridge regression** have demonstrated significant advantages in managing complicated, large datasets while maintaining prominent levels of accuracy. Notably, the incorporation of photographs was noted as a potential component for improving accuracy in the future. (Chada, 2019) study closely parallels the issues highlighted in this research, particularly in dealing with unstructured data and transforming it into a legible format for models.

According to (Zheng, Cai, and Zhang, 2022) one major advantage of the **LightGBM** model is its capacity to efficiently minimize memory utilization and improve model training speed. Hence, it is widely used in various machine learning tasks, such as sorting of the data and classification, but on the other side, it is crucial to fine-tune the model properly, otherwise it can lead to overfitting. Another LightGBM-based research study demonstrates this point by implementing a similar notion in a hybrid model known as the **ARIMA-LightGBM Hybrid Model**. The authors (Zheng, Cai, and Zhang, 2022) underline the need to balance accuracy with the ARIMA model while also giving a faster and more memory-efficient technique suitable for bigger datasets.

Today, deciding about buying or selling any assets or gifts is an exceedingly arduous process. Varied factors and complexities can influence your decision, such as the best time for buying or selling the products, goods, or seasonal gifts. One such article discusses this problem using the following algorithm: **Linear regression, ridge regression, support vector regression (SVR), random forest regression (RF), and ARIMA algorithms.** Followed by the ARIMA model, the prediction based on the random forest regression model achieved the best performance. The ridge regression model had the lowest evaluation metric scores. While the random forest model produces the best results, the ARIMA model was the second-best model with a small performance gap (Mohamed, El-Henawy and Salah, 2022). It identifies the future scope for building hybrid models for the sake of improving prediction quality.

Below are a few statistics on the accuracy of this model. The random forest model, the one with the best evaluation metric scores, has an average training time of 1 s.

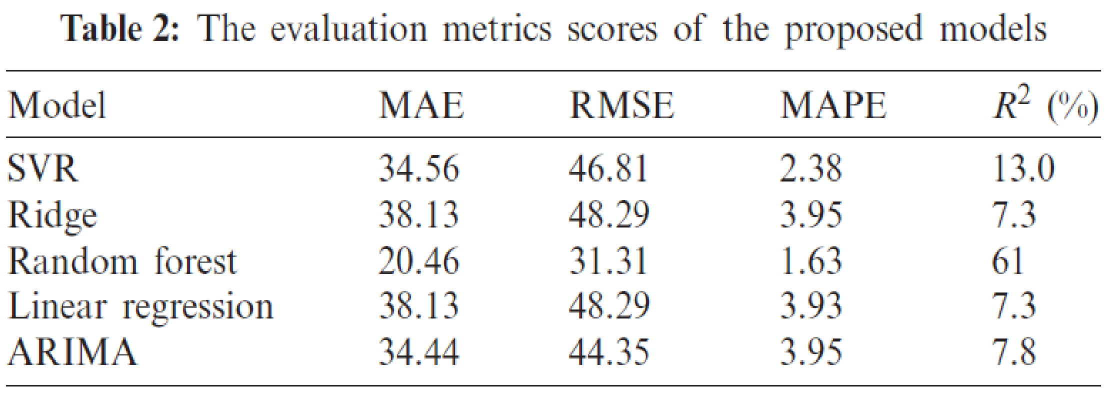


Table 2: The evaluation metrics scores of the proposed models

Model	MAE	RMSE	MAPE	R^2 (%)
SVR	34.56	46.81	2.38	13.0
Ridge	38.13	48.29	3.95	7.3
Random forest	20.46	31.31	1.63	61
Linear regression	38.13	48.29	3.93	7.3
ARIMA	34.44	44.35	3.95	7.8

Figure 6: The evaluation metrics scores of the proposed model for the literature review

The scores of the proposed five models state that there is a high MAE (Mean Absolute Error) against the model, which is not suitable for price- and time-critical scenarios. (Mohamed, El-Henawy and Salah, 2022)

However, there are several items and elements influencing their selling and other characteristics in the e-commerce sector. In addition, several variables, including pricing, promotion, rating, etc., have an impact on how much an e-commerce product sells, and these variables fluctuate quickly depending on the importance of a certain feature.

Survey of Stock Market Prediction Using Machine Learning Approach (RVS Technical Campus, IEEE Electron Devices Society and Institute of Electrical and Electronics Engineers, no date) to forecast stock prices, **including polynomial, radial basis function, sigmoid, and linear regressions** derived from popular regression techniques, The technical analysis method makes use of historical stock prices, including close and open prices, trading volume, and modified close prices. The second type of analysis was quantitative, which was done based on outside factors including the firm profile, the state of the market, and other political and economic factors. When compared to earlier methods, machine learning techniques in this field have been demonstrated to improve performances by 60% to 86%. New variables were created by merging existing variables to increase the accuracy of the expected cost value. The stock's closing price the following day is predicted using **ANN model**, and contrast analysis is performed using RF. The MAPE, MBE, and root mean square error values are used in comparative analysis. (Vijh et al., 2020)

The authors of (Polamuri, Srinivas and Krishna Mohan, 2019) employed a Naive Bayes classifier to calculate sentiment scores, followed by a neural network application of that classifier to both sentiment scores and historical stock market data. Furthermore, the authors concentrated on the use of the Hive ecosystem for data cleaning. On top of this pre-processing environment, this neural network was created based on prices forecasted using sentiment analysis and historical data. According to studies, the model obtains an accuracy level of more than 90% under ideal conditions. It is also discovered that if the model is trained using current data, it may have a stable foundation. While neural networks are used to develop a statistical link with a stock's historical numerical data records and other factors that affect stock prices, they are also used to construct the relationship.

A similar approach, covered in the article Engineering Applications of Artificial Intelligence (Keynia, 2012) discovers a new two-staged forecast strategy, a **Composite Neural Network (CNN)** and a few axillary predictors. This two-stage model filters out any irrelevant and repetitive information

This model depends heavily on products information to predict the sale and price of the product. Unfortunately, this information might not always be available in its required format for worldwide data be it stock price prediction or for any other products. Although this two-stage model offers more efficiency than single forecast. However, performance and accuracy are highly dependent on the selection of input filters, dataflow, and its building blocks. Optimizing these aspects is indeed a challenging task and requires lots of effort and research. Thus, inspired with these thoughts of two stage model while considering the benefits offered by ANN model, the paper conducts research on the Artificial Network with the well-known techniques for dimensionality reduction called 'Autoencoder.'

ANN is one of the most widely used mathematical and computational models, composed of a larger set of neurons that can vary based on the prediction model developed. These neurons can simulate the structure and functional characteristics of the neural network (Wang, Yao, and Zhao, 2016). It is now used for financial and time series forecasting and is most viable to model the complex relationship between input and output.

The ANN model is a self-adaptive system and can change its internal structure based on input parameters dynamically, making it suitable to capture intricate patterns between complex datasets. Once the model is trained, it can learn from data and improve its performance on given tasks (Vijh *et al.,* 2020). It can manage noisy and incomplete data, generalize to an optimum level from limited data, and make accurate predictions (Wang, Yao, and Zhao, 2016).

ANN model is used widely for solving many forecasting and decision modelling scenarios. It offers to easily transform any type of parameters and optimizes the input data. This theory is studied in the research conducted as ('Artificial neural network models for forecasting and decision,' 1994) an attempt to share review literature comparing ANN with other statistical techniques.

This study also highlights contributors in the using approach (Lin Zhao, 2009)found ANN to be inferior to Holt's, Brown's, and the least squares statistical models for yearly data. Sharda and Patil and (Tang, de Almeida and Fishwick, 1991) found that ANN for long memory produced comparable results whilst ANN for short memory, Tang found ANN to be superior to Box-Jenkins and to perform better even when predicting beyond the given input. The model covered under this research covers this feature of

ANN to strength the model to predict sale time and price for any complex and less formal dataset input and still predict highly accurate results. This feature will be supported by the best-known feature reduction technique called Autoencoder.

To Reduce the dimension of the complex dataset and get precision in the price and sale time prediction models, this section identifies different techniques studied earlier in a similar field of study. One effective approach is to use methods like **Principal Component Analysis (PCA)**. PCA helps identify the factors responsible for the variations in the data and uses them to represent the information using fewer bits. This method is known as "Dimensionality Reduction.” However, it is important to note that PCA can only provide an encoding with features that are linearly uncorrelated (Baeldung, 2023).

The alternative solution to the above problem is **An Autoencoder**. Autoencoders, also known as auto-associations, can learn faster using non-linear encoding features. These functions can represent complex data in a latent space, which has been demonstrated to achieve higher performance in various application domains. (Wang, Yao, and Zhao, 2016). The data in the latent space is also called encoded layer of the autoencoder. This encoded layer is used for feature reduction, thus mapping larger feature set with smaller feature set. Building an ANN model and representation of an autoencoder for dimensionality reduction regularizes the model, keeping it from overfitting the data. Based on this idea, this report explores a new evidential prediction model for price and sale time prediction for a complex marketplace dataset.

2.2 Conclusion

The above studies encourage to dive deep into the field of machine learning and explore the best possible way to develop the price and sale time prediction model. In the analysis conducted above, it can be observed that compared to other models like linear regression, ridge regression, support vector regression (SVR), random forest regression (RF), and ARIMA algorithms, ANN can be particularly valuable when there are a larger number of non-linear elements, with another advantage being that it eases model building and has great capacity for building predictive models. An ANN model with the combination of an autoencoder model will help turn a complex dataset into a more manageable one, improving the performance and accuracy of the model, which is significant in assessing the sale and price of a wide variety of products on the marketplace.

Chapter 3: Research Methods

This chapter discussed the fundamental components of this research journey, starting by thoroughly examining the development methodology and research methods. This chapter reveals the underlying motivation that has influenced the choice of essential elements within the project. Below is the diagram offering an overview of the development process and the toolkit meticulously used throughout the project.

Further sub-chapters here aim to provide a clear and comprehensive understanding of the fundamental principles that underpin this research and development efforts by exploring these primary components.

3.1 System Overview

The proposed methodology is broken down into five stages, including 1) Data Collection, 2) Model design, 3) model training, 4) hyperparameter tuning and 5) model testing and evaluation. The proposed system's framework is depicted in Fig. 3. Shows the architecture of the proposed system:

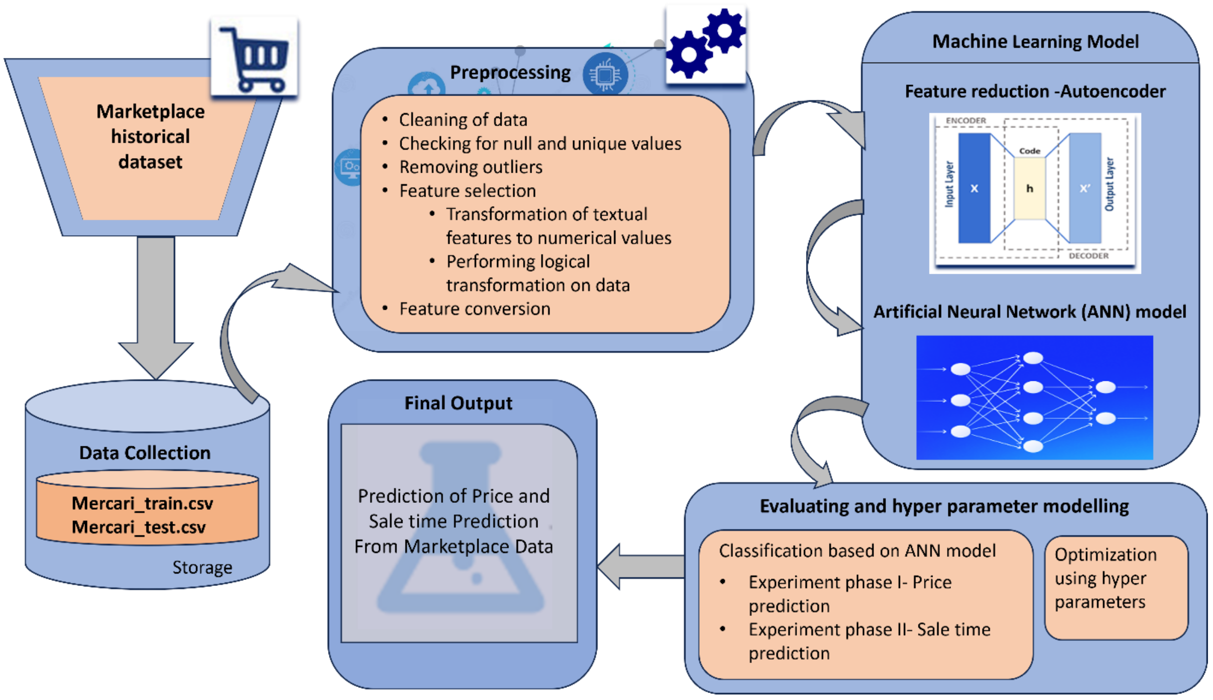

Marketplace historical dataset
Preprocessing
Cleaning of data
Checking for null and unique values
Removing outliers
Feature selection
Transformation of textual features to numerical values
Performing logical transformation on data
Feature conversion
Machine Learning Model
Feature reduction -Autoencoder
ENCODER
Code
DECODER
Input Layer
Output Layer
x
h
x'
Artificial Neural Network (ANN) model
Data Collection
Mercari_train.csv
Mercari_test.csv
Storage
Final Output
Prediction of Price and Sale time Prediction From Marketplace Data
Evaluating and hyper parameter modelling
Classification based on ANN model
Experiment phase I- Price prediction
Experiment phase II- Sale time prediction
Optimization using hyper parameters

Figure 7: The architecture of proposed model for price and sale time prediction

The initial phase starts by carefully identifying the data sources and matching them with the goals of the current project. In this situation, the dataset comes from the well-known Kaggle Marketplace. The quality and volume of this gathered data are key factors in how effective our results are. Our prediction abilities become more sophisticated as data availability and complexity increase. This unprocessed data is subsequently transformed into a format that can be used. The dataset must be distilled into more useful data values during the data pre-processing stage. The resulting dataset is then split into training and testing subsets at random.

The proposed model was created, its hyperparameters were revealed via experiments (for more information please refer Chapter 3.4 Hyperparameter tuning for Optimization and output of which is explained in Chapter 4: Results, and it was fit for analysis using the ANN model in the second step. Building machine learning models to forecast product prices and sale times is the goal of this phase.

To capture the diverse patterns of the data, relationships, and crucial features to create the learning model, the proposed model is trained on a subset of the datasets (i.e., 80%) during the model training phase.

The final testing phase involves evaluating the machine learning models to gauge their performance after training them on a subset of the dataset. By feeding the models a test set (20%), the models' accuracy rate is verified at this stage. Next, the loss functions compute the percentage of accuracy connected to the price prediction problem and sale time prediction separately.

3.2 Data Collection

One of the main goals of the online retailer is the increase the demand and supplies for the products according to the demand. Retail stores often offer promotions and special prices in an effective method for driving ancillary traffic to the site.

The data collected here is real data from an online retailer- (Mercari, 2020) for Mercari Price Suggestion Challenge launched by The Mercari marketplace app (Mercari, 2020) on Kaggle platform the company's primary product, was released to the Japanese market in July of 2013, after it was founded in 2013. Since then, it has developed into one of Japan's leading community-driven marketplaces, with over 10 JPY billion worth of transactions occurring on its platform each month.

The e-commerce Mercari dataset used for building the proposed model is described in the following table.

Table 1: Description of the dataset features

No.	Feature	Type	Description
1	Id	Numeric	The ID of the listing
2	name	Categorical	The name of the product
3	item_condition_id	Numeric	The condition of the items provided by the seller
4	category_name	Categorical	Category of the product
5	brand_name	Categorical	The brand name for the product
7	price	Numeric	The price of the product in USD

8	shipping	Categorical	The indicator of shipping paid by seller or buyer
9	item description	Categorical	The full description of the product
10	date_first_available	DateTime	The date products uploaded on the online store
11	sell_date	DateTime	The date product sold through the online store
12	sale_time	Days	dynamically created

Data comprehension

Every row in the training dataset corresponds to a product that is listed and contains specific information about it. The product's exact details are shown in each column. There are seven columns in total, and they are as follows: “Women/Athletic Apparel/Pants, Tights, Leggings," for example.4. brand_name is the brand name is provided by this feature.

There are eleven features and two output features. The response feature is the price and the dynamically created feature sale_time which is the number of days required to sell the product calculated from two existing features- Date first available, Sell Date.

The dataset consists of two partitions. The first part of the dataset comprises ~700K records for the training model, and the second portion includes ~3.5M records as the test data.

3.3 Model Design

3.2.1. Implementation details: Installation and setup

The proposed model is implemented in the Python programming language and Google Colab notebook for running the proposed ANN Price and sale prediction model. The model references eight important library which is used widely across the document:

- Imported panda's library, a widely used data manipulation and analysis library. [https://numpy.org/doc/stable/]
- Imports NumPy library used to work with arrays and for numerical computations [https://numpy.org/doc/stable/]
- Scikit-learn machine learning library, which is an extension of NumPy for additional scientific computing functionalities [https://docs.scipy.org/doc/scipy/reference/]
- Imports the Seaborn library as sns, a library designed for creating informative and attractive Statistical graphs and charts, build on the top of Matplotlib which is discussed in the next point [https://seaborn.pydata.org/]
- Matplotlib imported to create Static or interactive visuals in python. [https://matplotlib.org/]
- Multiprocessing-for multi-core computing support.

  [https://docs.python.org/3/library/multiprocessing.html]
- % matplotlib inline - I Python Magic Commands to allow plots to render in the notebook. [https://ipython.readthedocs.io/en/stable/interactive/magics.html#magic-matplotlib]
- These libraries and configurations setup the required environment for data analysis, manipulation, and visualization, as well as for monitoring the execution of the code.

3.2.2. Data exploration and preparation

For the analysis of the dataset, the data is sorted ascending order to get the glance of the dataset for further pre-processing in clear and systematic manner.

3.2.3. Pre-processing of the dataset

The third stage in the pre-processing phase is to analyse the dataset.

- **Eliminating outliers**: Outliers are crucial in the dataset pre-pre-processing and their alteration or exclusion can influence the accuracy and performance of the proposed model. Therefore, here the model gets rid of features record with price values bigger than two hundred. This documentation will help for reproducibility in model's learning workflow.

- **Correlating the feature set**: This analysis starts with a look at the "item description" column, specifically its length in both the training and test datasets. Following that, the focus shifts to other columns, such as price and category length. The goal is to compute correlation coefficients to help understand the strength and direction of the linear correlations between these four descriptive variables or columns.

  The resulting correlation matrix has four elements that indicate the correlation matrix with the other three elements and itself. The correlations between different pairs of variables are represented by the off-diagonal elements. The correlation between 'no_of_words' and 'price', for example, is 0.123456, showing a weak positive linear association. It assists in understanding how strongly or weakly the selected variables are related to one another. Positive values indicate a positive connection, negative values indicate a negative correlation with respect to the feature on the other axis, and values close to zero indicate a weak or no linear association.

  Based on the correlation matrix, which is visualized in results the model will determine the features with highest correlations. This will be further decomposed in data preprocessing.

- Modifying features set: Category feature is passed through a lambda function to make it easier to work with the individual categories for analysis, calculating value counts for item_condition_id and encoding it to binary item_condition_id values. Merging columns for analyses, converting all columns to the lower-case fields, calculating unique values, and checking null values.

3.2.4. Autoencoder for feature reduction

Considering the scenario where there is need to predict a large volume of dataset with nonlinear features it is extremely hard to develop a model predict the price and sale time of wide range of datasets which product in this case, accurately. This paper comes from this idea that the implementation of autoencoder will have significant property for reducing the complex nature of these features and thus contribute to the success of this machine learning model.

Auto-encoder is one of the deep learning architectures used for representation of data, typically for the purpose of dimensionality reduction. It is an unsupervised Artificial Neural Network that compresses the complex data into lower dimensionality and then decodes the data later reconstructing the original input. This model implements one such autoencoder with a multilayer architecture called deep generalized autoencoder to manage overly complex dataset. Below is structural representation of the autoencoder

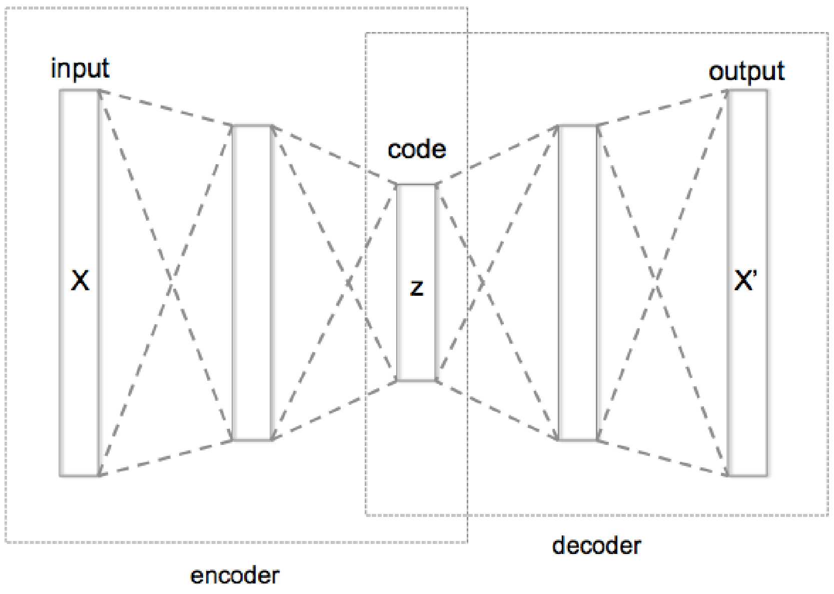


Figure 8: Structure with visual description of auto-encoder(Chervinskii, 2015)

The above diagram is the more detailed visualization of an autoencoder. As the autoencoder broadly divided into two divisions' an encoder and a decoder, both are used to perform the representation learning which a type of learning to find the representation necessary for feature detection or classification of the raw data using a class of machine learning (The Python Code, 2023). This technique allows the encoder to acknowledge the feature, identify it uniquely and reduce the feature without the need for manual intervention. Thus, called as autoencoder feature engineering.

The code part in the diagram, is the latent space which is the compressed feature space also called as hidden layers which saves as compressed and reduced feature set- learning the features of data and simplifying data representations for the purpose of finding patterns. (Ekin Tiu, 2020).

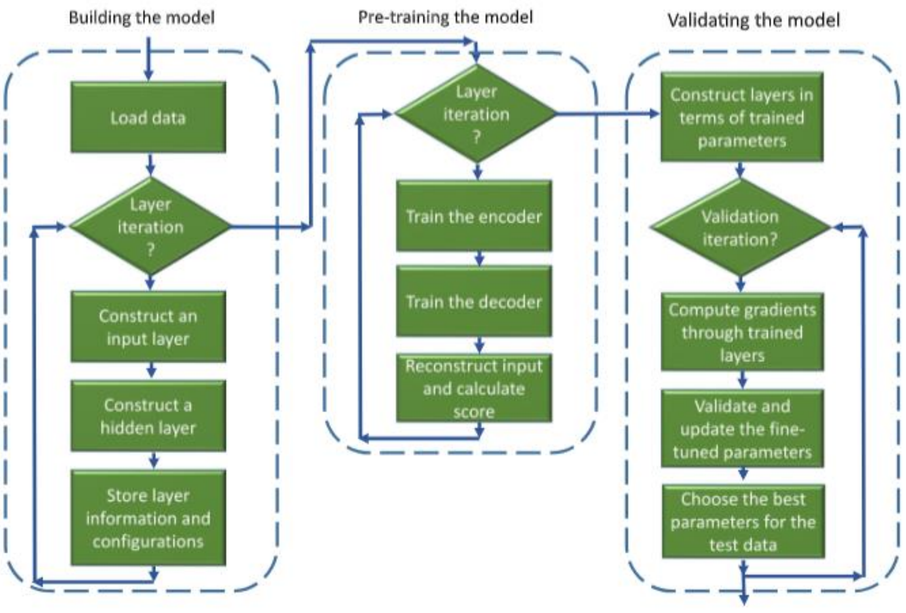

Building the model
Pre-training the model
Validating the model
Load data
Layer iteration ?
Construct an input layer
Construct a hidden layer
Store layer information and configurations
Layer iteration ?
Train the encoder
Train the decoder
Reconstruct input and calculate score
Construct layers in terms of trained parameters
Validation iteration?
Compute gradients through trained layers
Validate and update the fine-tuned parameters
Choose the best parameters for the test data

*Figure 9: The flow chart of autoencoder algorithm (*Yu, Yu, and Pan, 2017)

The implementation of the autoencoder follows with the rule that both input and output need to be of same size. To make it more efficient the model has Early Stopping Keras functionality which check the training loop at end of every epoch, whether the loss is not decreasing considering the min_delta and patience. Once it is found no longer decreasing it will terminate the training process for autoencoding. (Keras.io, 2023)

While developing the autoencoder algorithm, the model also considers the following hyper parameters for the architecture of the feature reduction. The brief description of below mentioned hyper parameters is covered in section-3.4. Hyperparameter tuning for Optimization

- The number of the layer:

  This includes the input, output, and number of hidden layers, for the prediction model only one hidden layer will be implemented in autoencoder as well as in the ANN model, if the precise result is achieved with it, or it might be considered for hyper parameter tuning.

- Encoding layer: Similarly, the experiment shall evaluate for optimized number of features to be encoded, which will serve as input to the ANN model as reduced feature set. This number will be considering the scenario that there is no major loss of feature set and salient feature are considered for prediction while training the ANN model. These conditions will provide number of layers in the encoder and decoder, and number of neurons or nodes per layer.

- Function of activation: Activation functions in ANN plays a vital role in neural network to decide the whether the neuron should be activated or not and to decide the pattern of activation. The model implements Rectified Linear Unit (ReLU) as the activation function. ReLU offers simplicity while offering the advantage that it does not activate all the neurons at the same time (Dishashree26 Gupta, 2020).

- Function of optimization: Also called optimization algorithm offers model to minimize the loss function by updating the model's parameters. The developed model implements Adaptive Moment Estimation (ADAM) optimizer. ADAM optimizer computes adaptive learning rates for each parameter on basis the past ascent information(Jeremy Zhang, 2021)

- Loss Function: The model implements Mean Squared Logarithmic Error (MSLE) for calculating the loss while compiling the model to quantify the discrepancy between actual values and predicted values.

The above architect serves as the crucial part of the price and sale time prediction model and demonstrates a promising improvement in the performance and accuracy of the experiment in phase I of price prediction as well in experiment for phase II for sale time prediction.

The model then applies One-hot encoding, a common technique used in data pre-processing, to the brand_name feature, bucketing the category features and passing them through 'pd.get_dummies' (One-hot encoding), then margining the columns and deleting the unnecessary features to get the resultant reduced categories feature.

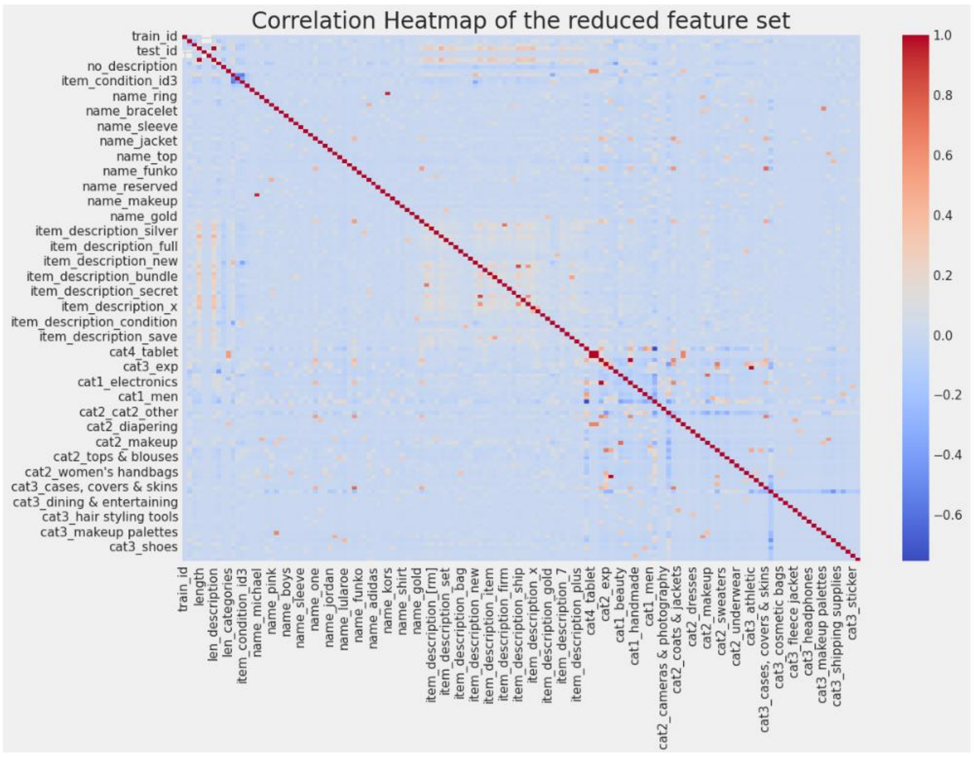


Figure 10: The heatmap visualizing features correlation

The above graph is a visual representation of the pairwise relationship between the feature of the dataset. It shows how unique features are corelated to each other. The colours in the above graphs represents this correlation; The cool colours for example the blue colour represents negative correlations while the warms colour for example the red colour represents the positive correlation. The gradient colour combination on the right side of the graph helps in identifying the highly correlated features that can be either reduced or removed completely to improve model efficiency and interpretability of the model.

3.2.5. Building the ANN model

There are varied types of NNs model, out of which ANN is more popular than others. When neural networks used for data analysis, it is essential to distinguish between ANN models (the network's arrangement) and ANN algorithms (computations that eventually produce the network outputs). The most often used ANN is a fully connected, supervised network with backpropagation learning rule (Fig. 5). This type of ANN is excellent at prediction and classification tasks

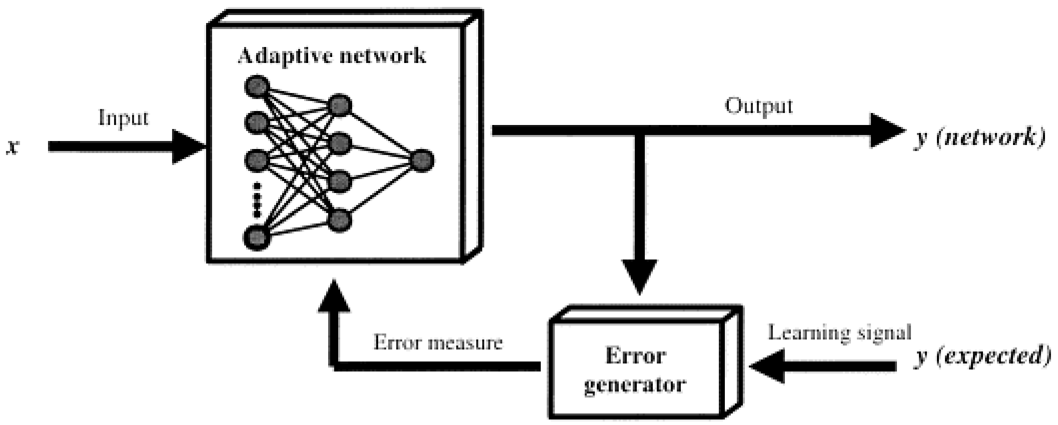


Figure 11: Supervised network with backpropagation learning rule

The following equation depicts the mathematical description of ANN (Puri et al., 2016)

$$Y(t) = F(\sum_{(i=1)}^{n} \left[(Xi(t)Wi(t) + c)) \right]$$

Where:

$Xi(t)$ *is the input value at time t,*

$Wi(t)$ *is the weight of neural input at time t,*

c *is the bias,*

F *is a transfer function,*

$Y(t)$ *is the output value at time t.*

Here the model defines input with 128 neurons to our Keras neural network and 1043742 records divided into 70 % of the data for training and other 30% of the dataset for evaluating and evaluating the model for price and sale time estimates.

While choosing the number of neurons as the reduced feature set, the autoencoder algorithm will undergo the performance and loss evaluation during training.

The modelling of the system starts with configuring the architecture of an artificial neural network (ANN) using Keras, a high-level neural network.

Adding the Input Layer and one hidden layer with standard configuration for deep learning. The model also implements EarlyStopping functionality while training the model. This is helps to overcome and avoid overfitting problem.

The code creates an output layer with a single neuron, 'uniform' weight initialization (weights are initialised uniformly), and with ‘Adam ‘optimizer. Later compiling the entire ANN setup which will be used for further evaluation and for calculating the loss while training the model.

Overall, through above defined steps the method for price and sale time prediction using ANN model aim at achieving highest precision with the best performing and extremely low error propagation near to none.

3.4 Hyperparameter tuning for Optimization

Hyperparameters are the parameters that control both the network's structure (such as the number of hidden units) and its training (such as the learning rate). Hyperparameters are crucial because they affect how well an artificial neural network performs (Gaspar *et al.*, 2021).

Hyperparameter tuning and optimization are two of the critical steps in building machine learning models, including ANNs for tasks like e-commerce product predictions. These hyperparameters, once tuned, impact the performance and generalization ability of the model. This is one of the major milestones achieved in this project.

Lot of researchers have conducted research on the methodology of hyperparameter tuning and optimization to find optimal hyperparameters (Bergstra *et al.*, 2011), (Bergstra, Yamins and Cox, 2013)

The aim of this stage is to build the machine learning model to predict precise prices and sale times. This stage starts with the determination of problems and proceeds with the selection of the hyperparameter-tuning machine learning techniques that are most suitable for the problem. (Yoo, 2019)

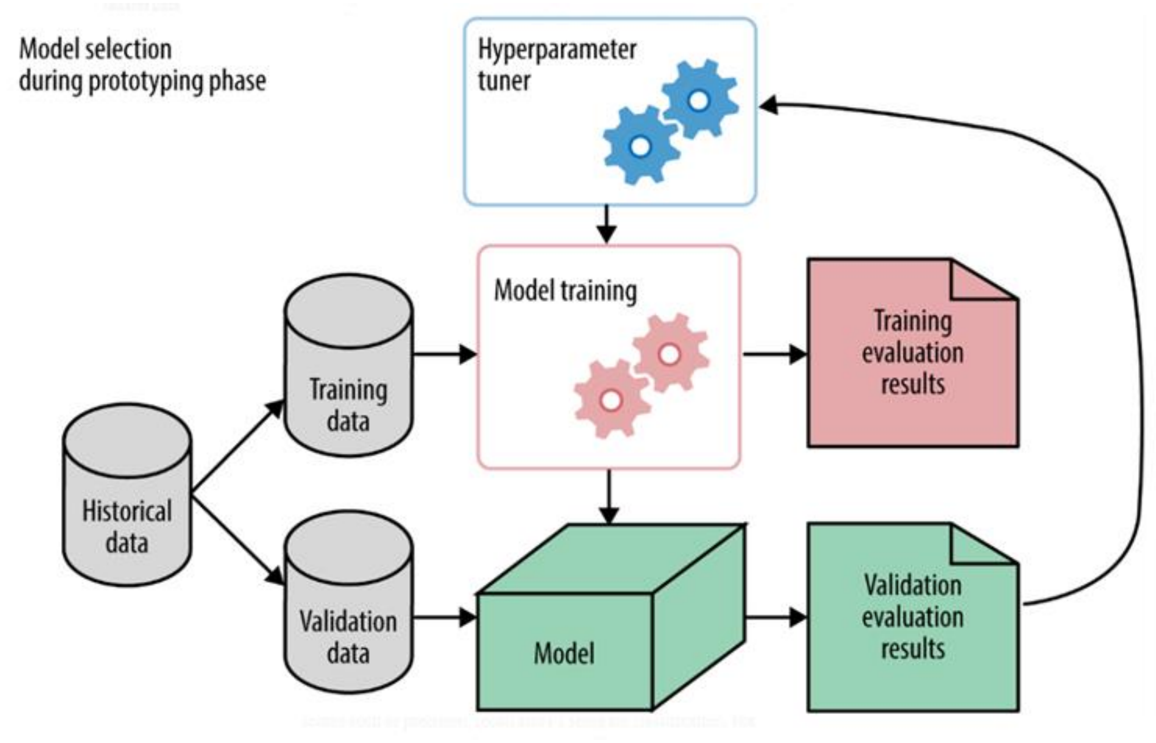

Model selection during prototyping phase
Hyperparameter tuner
Model training
Training evaluation results
Training data
Historical data
Validation data
Model
Validation evaluation results

Figure 12: General flow diagram for hyperparameters tuning (Sreenath S, 2020)

As can be seen in the above figure, it is unambiguous that the "hyperparameter tuner" is separate from the model and that tuning takes place before training the model. The ideal hyperparameter values that come from the tuning procedure are subsequently used in the model training phase.

Here is a more detailed explanation of hyperparameters adjusted and fine-tuned for the price and sale time prediction models.

1. Number of hidden layers

The depth of the ANN model depends on the number of hidden lines allowed in the model to capture the complex relationships. It is necessary to tune the neuron present in the hidden layer, as it can be one of the prominent reasons for overfitting or underfitting. More neurons than expected can lead to underfitting, and vice versa. Here the model is tuned by passing the combination of neurons with a few hidden layers and increasing the complexity if necessary.

2. Number of neurons

Experimenting with the number of neurons (also called units or nodes) controls the ability of the network to learn and reduces the error or loss while training the model. In fine-tuning this parameter, the hidden layer was passed a number near the input neurons; this resulted in exceptionally low error, but this had high chances of overfitting the model. Thus, after experimenting with different numbers of neurons, the ideal number of 100 neurons was decided for the network with 179 neuron inputs.

3. Activation Functions

The activation function parameter enables non-linearity in the model, which extends the model's learning capability to predict non-linear boundaries.. Activation functions in ANN plays a vital role in neural network to decide the whether the neuron should be activated or not and to decide the pattern of activation. The model implements Rectified Linear Unit (ReLU) as the activation function. ReLU offers simplicity while offering the advantage that it does not activate all the neurons at the same time (Dishashree26 Gupta, 2020). This results in deactivation of the neuron only when the output of the linear transformation is less than zero, thus is more computational than other activation functions like sigmoid and tanh function. Gaspar et al., 2021)

4. Number of epochs

The number of epochs determines how many times the entire training dataset is processed by the model. The model defines a maximum of 50 epochs for the autoencoder while using regularization techniques of early stopping; this will take care that the model does not overfit or underfit by stopping training of the model at the optimal number of epochs. Keeping it simple with 5 epochs for the ANN model, the model achieves the expected precision of prediction, which will be discussed in **Experiment phase I-Price Predictions.**

5. Optimizer choice

There are different optimizers that can be used for prediction, such as Adam, SGD, and RMSprop. The implemented neural network uses the Adam algorithm optimizer.

According to Kingma, the Adam method is computationally efficient, with little memory requirement, and well suited for large data or parameters problems. (Kingma and Ba, 2017)

6. Batch size

When training neural networks, batch size determines how well the error gradient estimate is calculated. The model keeps batch sizes smaller than 32 throughout the network model. A smaller batch size brings more noise but can help the model converge faster than a larger batch size and thus require less memory. (Jason Brownlee, 2019a)

3.5 Training and evaluating the ANN model for making prediction

In the models training stage, the proposed models are trained on a portion of the datasets (i.e., 80%) so that the models can capture the various patterns, relationships, and prominent features.

The compiled model is processed through the training data using predict function, output of which will be discussed in the chapter 4: Results.

Below are few metrics which will be used to evaluate the performance of the output predicted values:

- **Mean Squared Logarithmic Error (MSLE)**

The model implements MSLE for calculating the loss while compiling the model to quantify the discrepancy between actual values and predicted values. The aims towards using the loss function are to gradually minimize the loss and achieve lowest value which states highest precision for the prediction.

The Root Mean Square Logarithmic Error (MSLE) is calculated as the difference between actual and the predicted values. Adding the Logarithmic value reduces the punishing effect of significant difference in large, predicted values. Thus, more appropriate to be implemented as loss function for forecasting values (Jadon, Patil, and Jadon, 2022), Below is the loss calculation for MSLE

loss = square(log(y_true + 1.) - log(y_pred + 1.))

where, y_true is the actual value to be predicted

y_pred being the value to be predicted for y_true(TensorFlow, no date)

Below mentioned in the mathematical formula for calculating MSLE

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} (\log(y_i + 1) - \log(\hat{y}_i + 1))^2$$

Where, $\hat{y}$ is the predicted value.

- **Mean Squared Error (MSE)**

The model used Mean Squared Error (MSE) to evaluate the performance of the ANN model while training. MSE is a model assessment statistic that is often used in conjunction with regression models. The mean squared error of a model in relation to a test set is equal to the average of the squared prediction errors over all occurrences in the test set. The prediction error is defined as the difference between the actual and predicted values (Jadon, Patil, and Jadon, 2022), formula for MSE is as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

One of the advantages of using MSE as metrics while training the model is, its value increases exponentially if the error increases while for the high-performance model MSE value is close to zero.

- **Actual VS predicted price and sale time prediction**

The model will further evaluate the model's prediction by aggregating the values predicted comparing with the actual price. This information shall be presented graphically to visual overall performance and accuracy of the price and sale time prediction. This will be represented as a line plot with actual predicted prices, a scattered plot presenting trends of the error, a histogram showing calculate differences between actual and predicted prices.

- **Learning curve**

learning curve will help assess the changes in the model's performance with different training dataset sizes, allowing to make informed decisions about model complexity and data requirements and expedite the decision-making power for improvement in the model effectiveness. A learning curve will help diagnose underfitting and overfitting and give insights into the model's ability to generalize to new data, increasing power of the model to deal with complex dataset.

3.6 Conclusion

In summary, our research set out to tackle the challenging task of anticipating product pricing and expected sale periods within the dynamic environment of online marketplaces. It sought to offer insightful perspectives into the complex environment of marketplace data analysis through a comprehensive procedure that included data sourcing, pre-processing, model creation, and stringent evaluation.

The study began with a thorough understanding of the data sources, highlighting the crucial role that data quality and quantity play in determining the precision of forecasts. Important turning points in our project were the formatting of raw data for usability and the trimming of features for better meaning.

The developed experimented with designing machine learning models specifically suited for the task at hand, drawing on the broad capabilities of Artificial Neural Networks (ANN). To achieve our goal of forecasting product prices and sale dates, this phase represented a significant step.

The following phases, which included model training and thorough testing, were crucial in confirming the effectiveness of our models. It was able to thoroughly examine accuracy and loss functions because to the precise splitting of the data into training and testing subsets.

By these efforts, the goal is to provide a thorough framework for the analysis of marketplace data, considering the significance of both computing efficiency and forecast accuracy.

The research recognizes the complexity of data analysis in marketplace dynamics as it wraps up this endeavour. To further enhance prediction quality, the model implements innovative methods like hybrid models that use ensemble learning. Even more accurate predictions might be obtained by considering the temporal part of the data by rephrasing the issue as a time-series problem and adding date and time input features. To achieve data-driven excellence, marketplace data analysis is a field that will require constant investigation and innovation, which is the primary accomplishment if this project.

To evaluate the results of our research discussed in chapter 4 Results, the evaluation is divided into two unique phases: **Experiment Phase I**, which focuses on price prediction for the marketplace data, and **Experiment Phase II**, which focused on sale time prediction.

Both trials use the same dataset, but with phase-specific pre-processing customised to improve the applicability and predicted accuracy of the results. The feature dataset used as input in both experiments, with a consistent dataset count maintained throughout. For evaluation purposes, null values were deleted, and distinct values were carefully analysed.

In all trials, the data is separated into a portion of test and training sets. The implementation approach remained consistent across both models, comprising the removal of irrelevant features particular to each experiment. These attributes followed a standardised format in both the test and training datasets, ensuring data consistency.

The key difference between the trials is the underlying logic used, particularly the specialised computations introduced for sale time prediction is discussed in Experiment Phase 2.

Chapter 4: Results

This chapter summarises the results of the approaches discussed in Chapter 3. These outcomes include price prediction and sale time prediction, which were computed using test data obtained from the Kaggle platform. This platform provided a wide range of e-commerce products accessible from the Online Mercari retail platform. The chapter also introduces the developed tool and the output files. The output is painstakingly clarified, following a step-by-step approach, and emphasising the graphical representations as indicated in Chapter 3: Research Methods. Furthermore, demographic data is shown to demonstrate the accuracy and precision of sale timing and price projections for various e-commerce products.

4.1 Experiment phase I- Price Predictions

This section represents the preliminary phase (Phase 1) of the research, and it is concerned with generating price prediction for the marketplace ecommerce products processing the test data. The following part digs deeper into improving the precision of these price projections, spanning a broader range of findings.

The following were the **primary goals** of this experimental phase:

1. To evaluate the effectiveness of the established Artificial Neural Network (ANN) model in price forecasting, with the goal of assuring the least amount of variation from accurate results.

2. To maintain data integrity by minimising data loss within a loss rate range of 0.00 to 0.99.

3. To optimise the prediction model's speed in accordance with the previously stated goals.

The approach used in this step is graphically illustrated, effectively illustrating the model's efficiency, accuracy, and performance.

There is approximately 1043742 of the data records including various ecommerce product with their detailed description, category, price and with few more features. For the test results, the data is divided into 70% for training the model and remaining 30% which is around 342575 records for training and evaluation of the results. Below are the stepwise results obtained by applying the methods discussed in Chapter 3. Research methods

Step 1: Output of data collection and data interpretation:

To start the analysis of the data structure the model needs to get insights of the number of rows and column to check the content and the format of the dataset to be build the neural network model.

This is achieved by reading the data from the excel sheet by uploading it into the project local folder

```
price_train = pd.read_csv("/content/drive/My
Drive/Project/Data/ANN/Meraci_train.csv", encoding='utf-8-sig')
```

```
price_test = pd.read_csv("/content/drive/My
Drive/Project/Data/ANN/Meraci_test.csv", encoding='utf-8-sig')

print(price_train.shape)
print(price_test.shape)
```

This print gives the number of row and column present in the test and train dataset, respectively.

This experiment considers 85000 and 25000 records for training and testing for prediction model respectively, this is to limit to limit records for preprocessing to avoid systems to crash out of resource.

```
price_train=price_train[:85000]
price_test=price_test[:25000]
```

Above lines of code can be comments to test model with entire dataset which is as follows in the format of number of rows, columns (feature set).

```
(706000, 10)
(342575, 10)
```

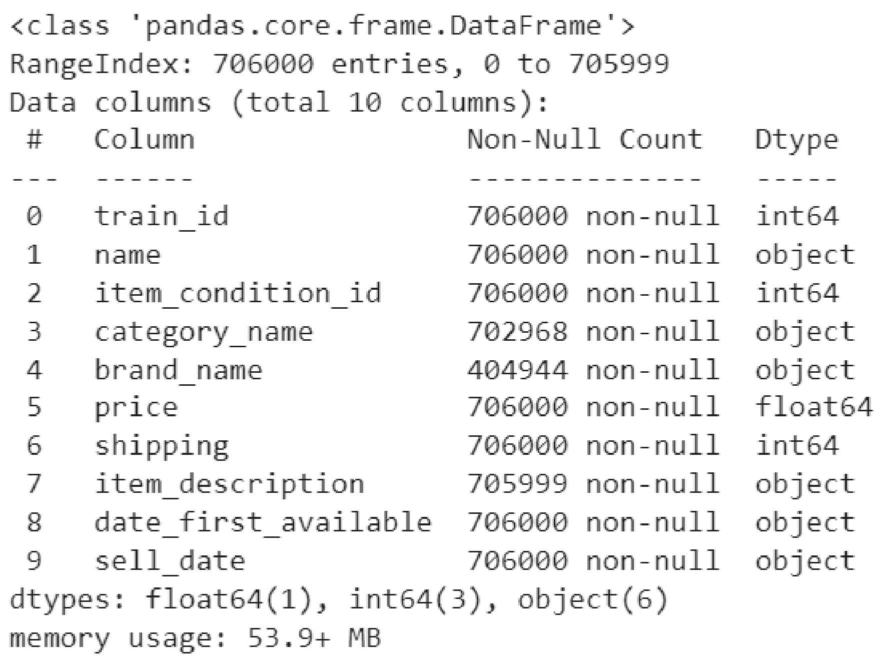
Output to this is as follows:

```
(706000, 10)
(342575, 10)
```

Upon checking the format of 10 columns also referred to as features in this report

```
#checking content and format of the train dataset
saleTime_train.info ()
```

Below is output feature set,



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 706000 entries, 0 to 705999
Data columns (total 10 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   train_id             706000 non-null  int64  
 1   name                 706000 non-null  object 
 2   item_condition_id    706000 non-null  int64  
 3   category_name        702968 non-null  object 
 4   brand_name           404944 non-null  object 
 5   price                706000 non-null  float64
 6   shipping             706000 non-null  int64  
 7   item_description     705999 non-null  object 
 8   date_first_available 706000 non-null  object 
 9   sell_date            706000 non-null  object 
dtypes: float64(1), int64(3), object(6)
memory usage: 53.9+ MB
```

The output feature shows column name with the non-null values with the total count of records and datatypes of each feature. As depicted above brand_name, category_name and item_description has some null values like ‘301056’, ‘3,032’ and ‘1’ count, respectively.

Below is the graphical representation of the above missing values in the train dataset, here x axis represents the column, or the feature names and Y-axis represents the input records in the range of 0.0 to 1.0 scale with the record counts for non-null values on top of each bar.

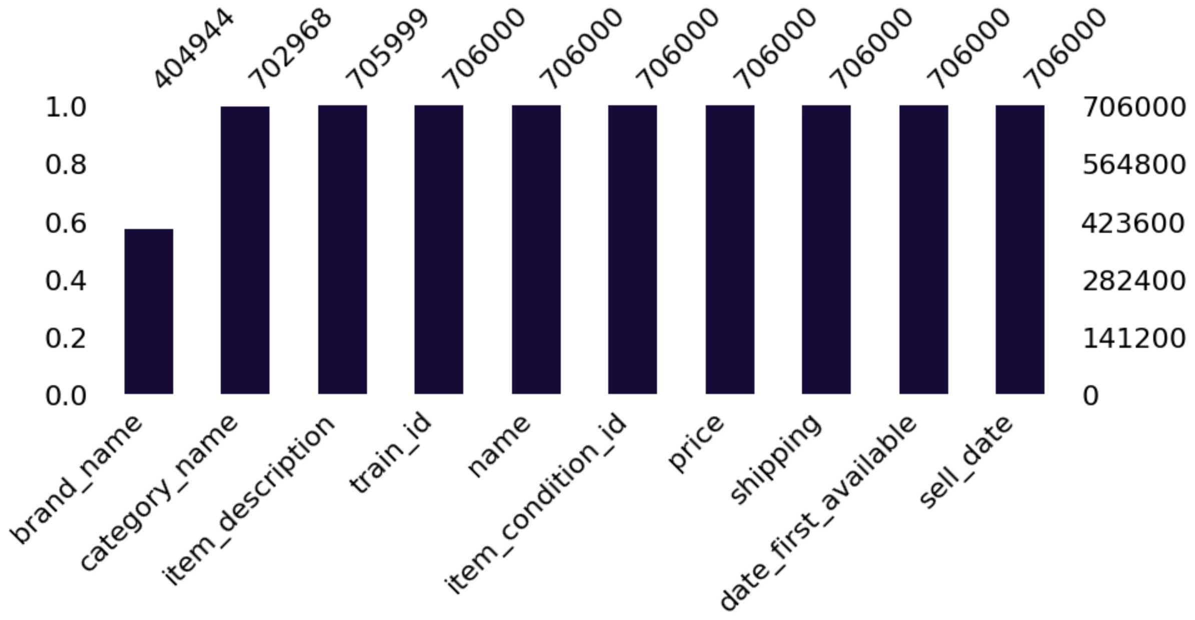
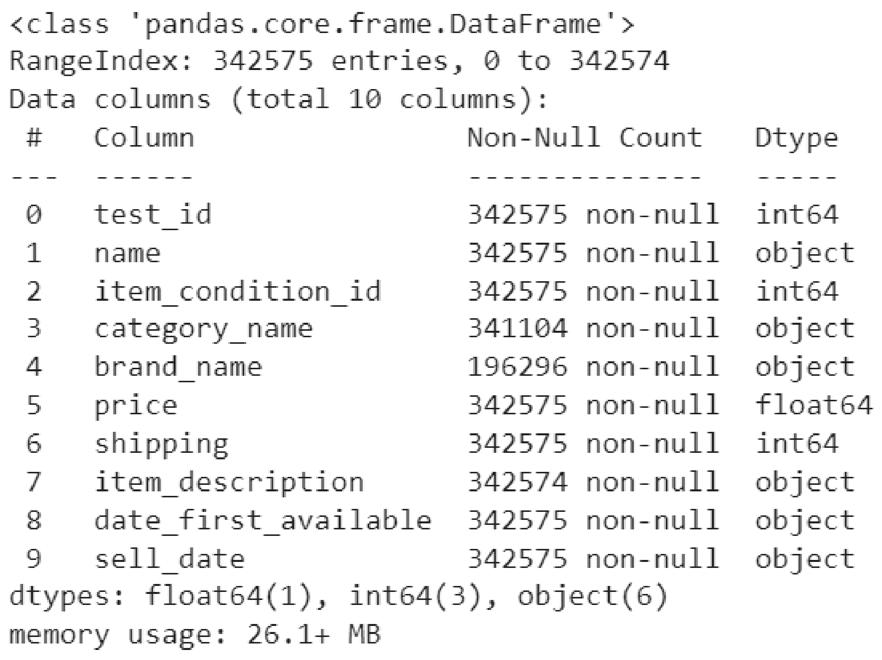


Figure 13 Count of records with presentation of missing values in the Train dataset

Similarly, checking the format of test dataset with 342575 records through command “saleTime_test.info ()” presents output as brand_name, category_name and item_description has some null values like ‘1,46,279, ‘1,471’ and ‘1’ count, respectively.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 342575 entries, 0 to 342574
Data columns (total 10 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   test_id              342575 non-null  int64  
 1   name                 342575 non-null  object 
 2   item_condition_id    342575 non-null  int64  
 3   category_name        341104 non-null  object 
 4   brand_name           196296 non-null  object 
 5   price                342575 non-null  float64
 6   shipping             342575 non-null  int64  
 7   item_description     342574 non-null  object 
 8   date_first_available 342575 non-null  object 
 9   sell_date            342575 non-null  object 
dtypes: float64(1), int64(3), object(6)
memory usage: 26.1+ MB
```

Below is the graphical representation of the above missing values in the test dataset, like the train dataset graph.

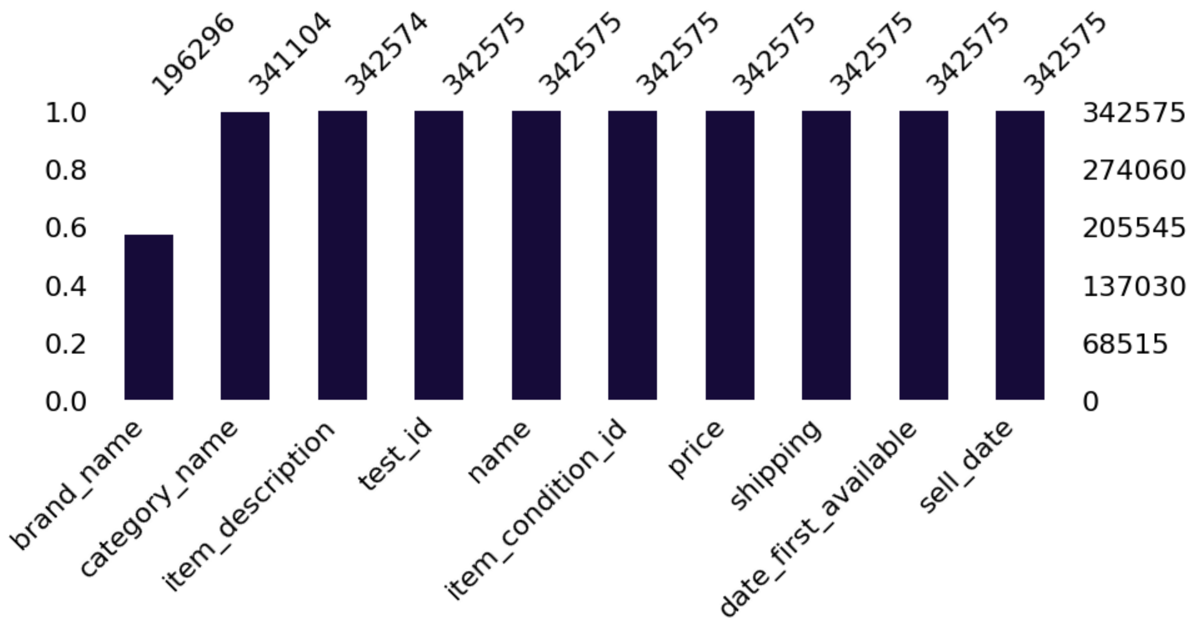


Figure 14: Count of records with presentation of missing values in Test dataset

Step 2: Results of pre-processed dataset:

After thoroughly understanding the pattern of the dataset features, the model proceeds with preprocessing this dataset, the prominent goal for this step is conversion of the quantitative (Involving numerical data) as well qualitative (Focusing on non-numerical) data features into a range of binary values and common numerical format, respectively.

Starting with getting rid of outlier in the model, the developed model removes all records having prices feature greater than 200. Below is the histogram representing the price feature in the training dataset in the range 0 to 200 after removing outliers.

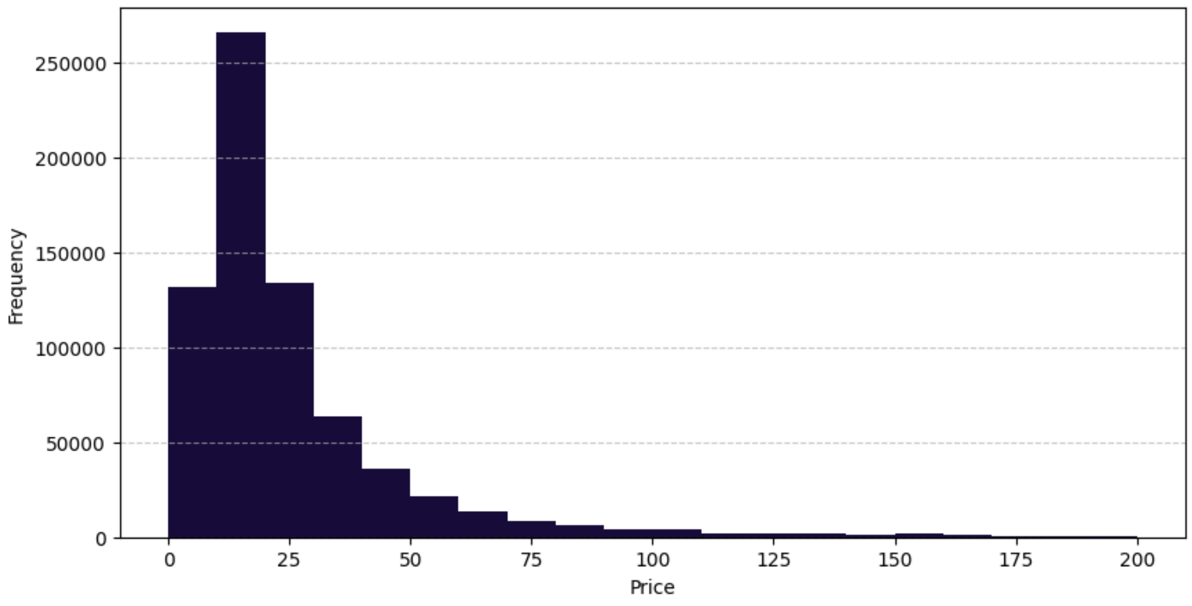


Figure 15: Histogram of Price in Train Data

This figure shows price of each product on x-axis and the frequency, which is the number of records on y-axis, representing most of the products in the dataset has price between 10 to 20, the unit is USD. This is column presenting price the item is sold and the target variable to predict in the test dataset for in this experiment.

Checking the range of the Category ID feature to determine appropriate conversion techniques

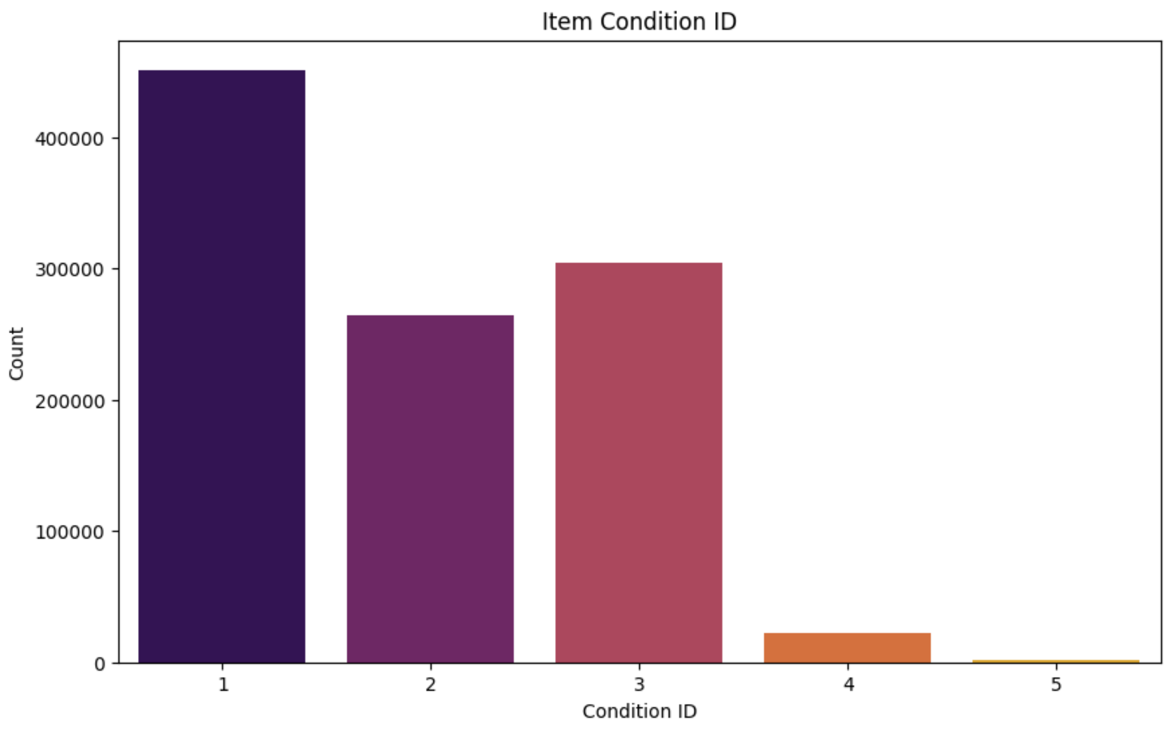


Figure 16: Range of records for Item condition ID Feature

Above diagram shows the graphical representation of dataset being spread out in the range of 0 to 5 unevenly, thus making it suitable for binary conversion

Below is the correlation between len_description and price, shipping, len_categories, length, feature records

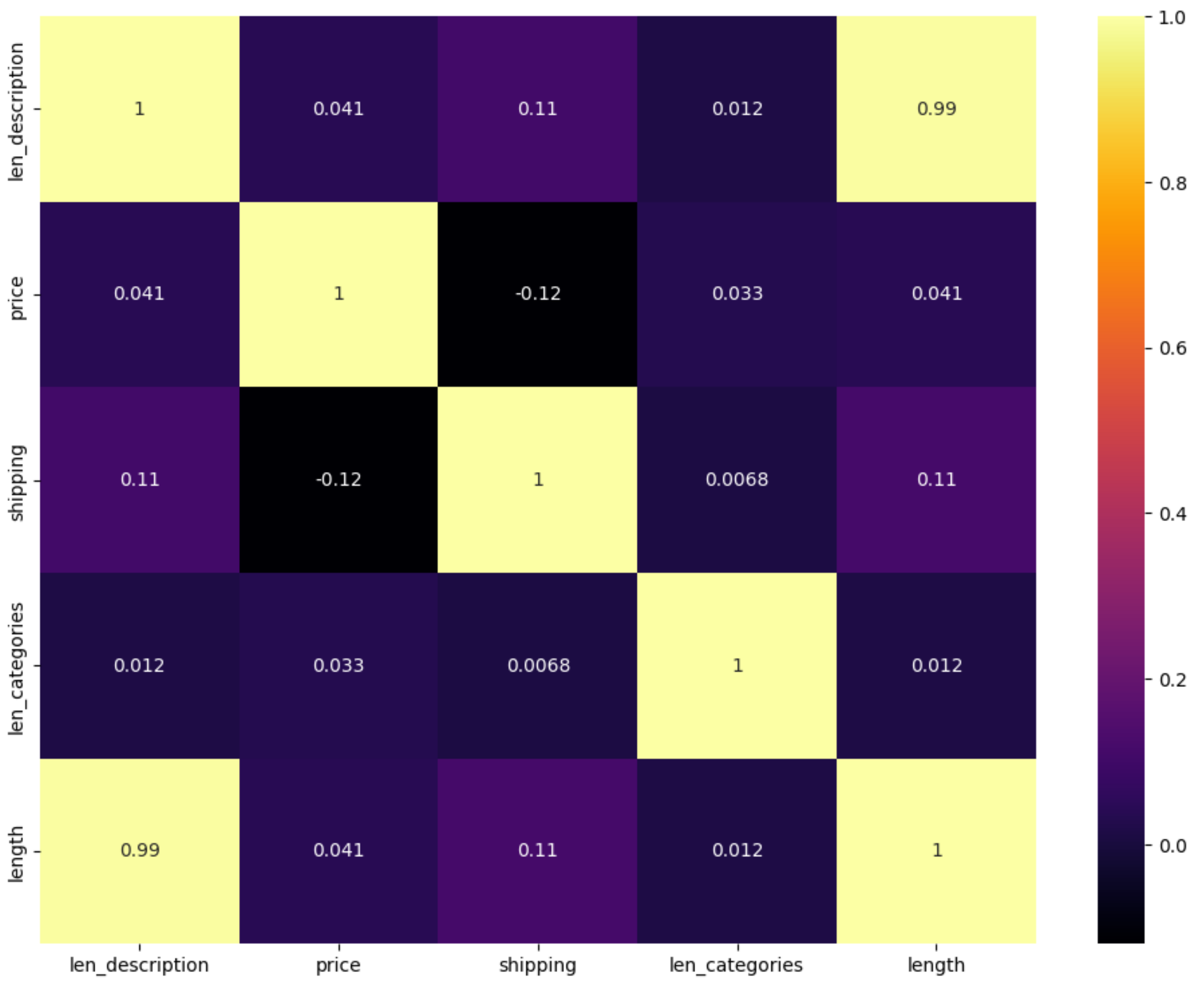


Figure 17: Correlation matrix with selected feature set

The above figure provides the outstanding features representing the highest correlation with feature dataset (1 being the highest i.e., tile with lighter colour and 0 being the lowest i.e., darker colour on the scale for the above graph).

Based on the above correlation matrix, the model determines notable features i.e., with the length of 158 features, each divided as follows

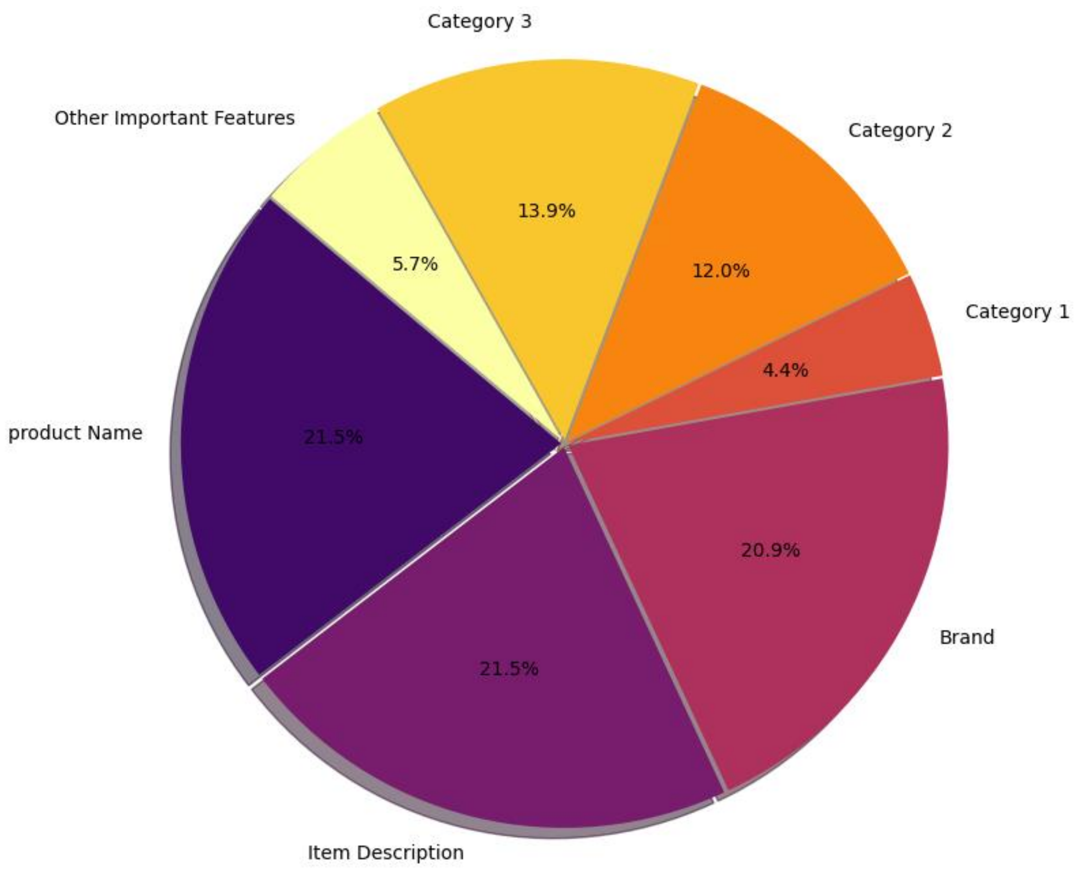


Figure 18: Distribution of extracted features and % weightage

Length of name:	34
Length of item description:	34
Length of brand:	33
Length of category_1:	7
Length of category_2:	19
Length of category_3:	22
Length of other key features:	9
Total length of pre-processed features:	158

Table name: Length of new features in relation to the parent column

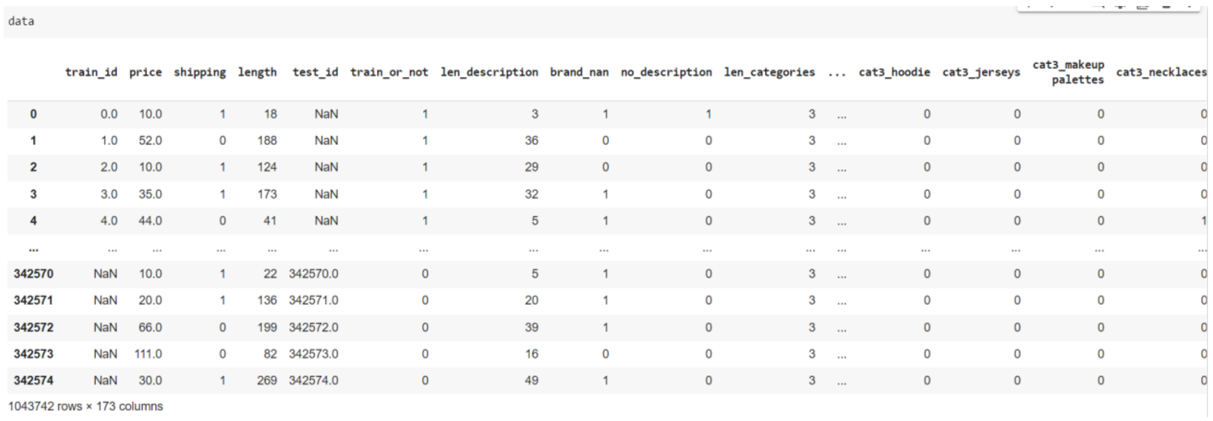
The output new features are used to create columns for each parent features discussed above.

The final output after preprocessing all new features into a binary and numeric datatype is as follows:

Output 1: information of the final pre-processed DataFrame:

Command	result
data.info ()	<class 'pandas.core.frame.DataFrame'> Int64Index: 1043742 entries, 0 to 342574 Columns: 173 entries, train_id to cat3_tracksuits & sweats dtypes: float64(3), int64(87), uint8(83) memory usage: 807.3 MB None
data.shape	(1043742, 173)

Printing the information of the final pre-processed Data Frame ready for modelling and data interpretation:



data

	train_id	price	shipping	length	test_id	train_or_not	len_description	brand_nan	no_description	len_categories	...	cat3_hoodie	cat3_jerseys	cat3_makeup palettes	cat3_necklaces
0	0.0	10.0	1	18	NaN	1	3	1	1	3	...	0	0	0	0
1	1.0	52.0	0	188	NaN	1	36	0	0	3	...	0	0	0	0
2	2.0	10.0	1	124	NaN	1	29	0	0	3	...	0	0	0	0
3	3.0	35.0	1	173	NaN	1	32	1	0	3	...	0	0	0	0
4	4.0	44.0	0	41	NaN	1	5	1	0	3	...	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
342570	NaN	10.0	1	22	342570.0	0	5	1	0	3	...	0	0	0	0
342571	NaN	20.0	1	136	342571.0	0	20	1	0	3	...	0	0	0	0
342572	NaN	66.0	0	199	342572.0	0	39	1	0	3	...	0	0	0	0
342573	NaN	111.0	0	82	342573.0	0	16	0	0	3	...	0	0	0	0
342574	NaN	30.0	1	269	342574.0	0	49	1	0	3	...	0	0	0	0

1043742 rows × 173 columns

The model then processes the pre-processed data frame (i.e., illustrated above) by splitting into the train and test dataset and prepares the model for ANN model research design.

After splitting, the data frame is converted to numpy.ndarray and the model obtains four sets of output:

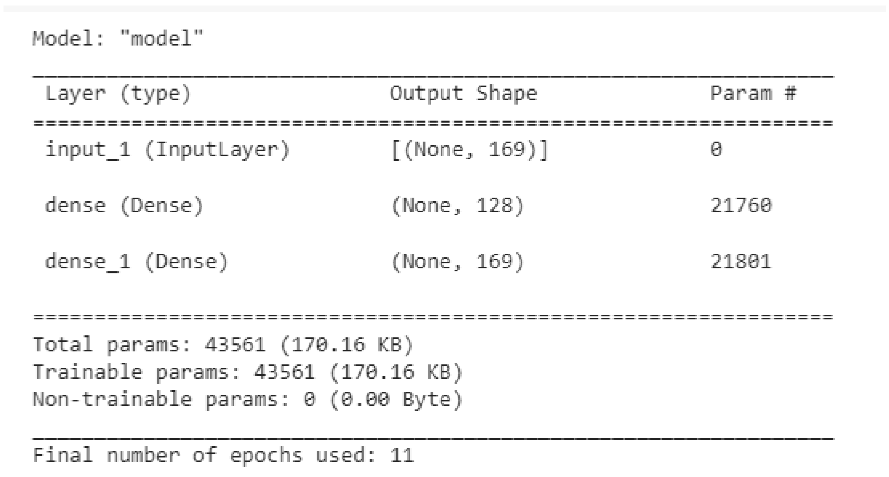
Shape of new dataset:	
Training dataset	(701167, 169)
Testing dataset	(342575, 169)

This represents the new dataset has 701167 rows for training and 342575 rows for testing the trained model, and 169 column or features. These 169 output features of pre-processed data will be the input nodes for the first layer of the ANN model framework.

Step 3: Results of reduction with autoencoder technique

Below is the summary of the model build for autoencoder

The autoencoder model includes one input, one batch- normalization, one activation and one dense layer. The activation function chosen for autoencoder is ‘The Rectified Linear Unit (ReLU)’ function, it computes F(x) = max (0; x). The last layer uses the linear activation the model needs the outputs to be in a linear sequential which has further been used to predict the model. The similar activation function is used to produce the output more suitable for the model, the output is a weighted sum of the input variables. (Saturn Cloud, 2023). Below is the summary of the autoencoder build for feature reduction



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 169)]             0

 dense (Dense)               (None, 128)               21760

 dense_1 (Dense)             (None, 169)               21801

=================================================================
Total params: 43561 (170.16 KB)
Trainable params: 43561 (170.16 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Final number of epochs used: 11
```

The autoencoder model is built with 169 input neurons as the first layer, one hidden layer with 128 reduced features. The number of encoded neurons is decided based on the training loss of ANN model and to achieving precision in prediction model. Less the number of reduction dimensionality the faster will be performance of the model, but this was guarded by check that feature for the ANN model performs well without losing any important feature set and fitting the prediction. This was achieved well with 128 encoded features. This encoded feature dataset will be used as an input for the ANN model discussed further in this section. The output of the autoencoder is the decoded layer of 169 output neurons.

Below is the detailed visualization of the model architecture, structure of layer and their names.

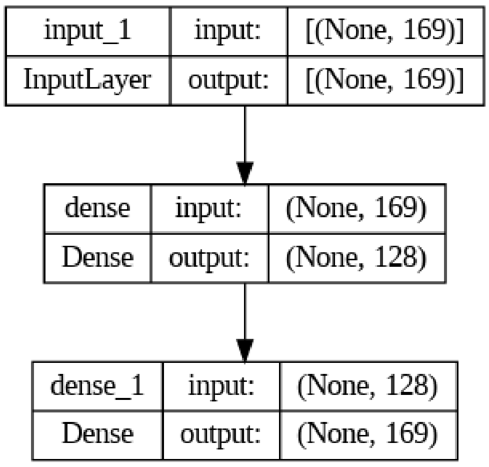

input_1 | input: | [(None, 169)]
InputLayer | output: | [(None, 169)]
dense | input: | (None, 169)
Dense | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 169)

Figure 19: Autoencoder model architecture

The model is implemented with EarlyStopping functionality discussed in Research methods chapter, as seen in the below snapshot this early stopping helps optimize model to compile the model until loss value reflects to remain constant while observing it for consequently for 2 epochs. This improves the models performance and prevents overfitting or underfitting problems in the prediction model.

```
Epoch 1/50
2639/2639 [==============================] - 10s 3ms/step - loss:
0.4903 - val_loss: 0.4427
Epoch 2/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.4398
- val_loss: 0.4406
Epoch 3/50
2639/2639 [==============================] - 9s 3ms/step - loss: 0.4388
- val_loss: 0.4401
Epoch 4/50
2639/2639 [==============================] - 8s 3ms/step - loss: 0.4383
- val_loss: 0.4396
Epoch 5/50
2639/2639 [==============================] - 8s 3ms/step - loss: 0.4380
- val_loss: 0.4394
Epoch 6/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.4378
- val_loss: 0.4393
Epoch 7/50
2639/2639 [==============================] - 8s 3ms/step - loss: 0.4375
- val_loss: 0.4391
Epoch 8/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.4374
- val_loss: 0.4389
Epoch 9/50
2639/2639 [==============================] - 9s 3ms/step - loss: 0.4372
- val_loss: 0.4390
Epoch 10/50
2639/2639 [==============================] - 8s 3ms/step - loss: 0.4372
- val_loss: 0.4386
Epoch 11/50
2639/2639 [==============================] - 8s 3ms/step - loss: 0.4371
- val_loss: 0.4386
Epoch 12/50
2639/2639 [==============================] - 9s 3ms/step - loss: 0.4370
- val_loss: 0.4383
Epoch 13/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.4369
- val_loss: 0.4384
Epoch 14/50
2639/2639 [==============================] - 9s 3ms/step - loss: 0.4369
- val_loss: 0.4384
2639/2639 [==============================] - 4s 1ms/step
782/782 [==============================] - 2s 2ms/step
```

As seen above, the autoencoder algorithm stops training after 14 epochs, which is the effect of the early stopping callback technique. This has a direct impact on the model being optimized and improved performance by training it until loss is decreased to a minimal value.

Step 4: Output of building an ANN model for price prediction

The model employed for price prediction starts with importing the required libraries to build an ANN architecture framework. Taking as input of 84422 records and 128 input nodes, the model trains the model with kernel_initializer ='he_normal'. This kernel initializer prevents the issue of vanishing gradients while training the model making it well-suited for activation functions like ReLU (Rectified Linear Unit). ￼ (Jason Brownlee, 2019). Below is the snapshot of training the model with the execution time required to complete the training process.

Here the output unambiguously states that the model is trained with loss less than 0.02 with mean squared logarithmic error also being equivalent to less than 0.30.. This prediction output creates a base for high performance while testing the model.

This approximator then will be used to predict the output value i.e., price for the data values in the test sets (i.e., for product's output values it has never seen before). The result of which will be presented be presented and analysed further.

The total execution time for the model took around an hour while processing 1000000 of data records inclusive of test and train dataset. This processing time also includes loading of such a large dataset file, analysing the dataset, pre-processing the columns for accurate data interpretation, and then reduction of the feature for precise price prediction.

```
Epoch 1/50
2639/2639 [==============================] - 7s 2ms/step - loss: 0.0287
- mse: 0.4306
Epoch 2/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0237
- mse: 0.3574
Epoch 3/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0229
- mse: 0.3455
Epoch 4/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0224
- mse: 0.3379
Epoch 5/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0220
- mse: 0.3311
```

Continuing till 50 epochs

```
Epoch 45/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0170
- mse: 0.2578
Epoch 46/50
2639/2639 [==============================] - 8s 3ms/step - loss: 0.0170
- mse: 0.2570
Epoch 47/50
2639/2639 [==============================] - 7s 2ms/step - loss: 0.0170
- mse: 0.2566
Epoch 48/50
2639/2639 [==============================] - 8s 3ms/step - loss: 0.0169
- mse: 0.2552
```

```
Epoch 49/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0169
- mse: 0.2546
Epoch 50/50
2639/2639 [==============================] - 8s 3ms/step - loss: 0.0168
- mse: 0.2534
0:06:23.168420
```

The above output shows information of model in the training phase. Each epochs includes the loss and MSE values with total time taken for training the price prediction model which is 6 min 23 seconds

The model shows that the MSE and loss gradually decreases typically meaning that the model can achieve the proposed goal for predicting the prices more precisely and achieve high performance. These metrics are further used to evaluate the performance of the model.

Step 5: Evaluation Metrics of the Proposed System and visualization

The proposed model is evaluated based on the four different methods discussed in Chapter 3.

- **Mean Squared Logarithmic Error (MSLE) and Mean Squared Error (MSE)**

On Compilation of the model over ‘Adam’ optimizer, loss as 'mean_squared_logarithmic_error' and metrics = ['mse'], the model produces output as below snapshot

```
Epoch 1/50
2639/2639 [==============================] - 7s 2ms/step - loss: 0.0287
- mse: 0.4306
Epoch 2/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0237
- mse: 0.3574
Epoch 3/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0229
- mse: 0.3455
Epoch 4/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0224
- mse: 0.3379
Epoch 5/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0220
- mse: 0.3311
```

Continuing till 50 epochs

```
Epoch 45/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0170
- mse: 0.2578
Epoch 46/50
2639/2639 [==============================] - 8s 3ms/step - loss: 0.0170
- mse: 0.2570
Epoch 47/50
2639/2639 [==============================] - 7s 2ms/step - loss: 0.0170
- mse: 0.2566
Epoch 48/50
2639/2639 [==============================] - 8s 3ms/step - loss: 0.0169
- mse: 0.2552
```

```
Epoch 49/50
2639/2639 [==============================] - 7s 3ms/step - loss: 0.0169
- mse: 0.2546
Epoch 50/50
2639/2639 [==============================] - 8s 3ms/step - loss: 0.0168
- mse: 0.2534
0:06:23.168420
```

The model here shows the modified version of the ANN model covering 50 epochs for training the model. This is a part of hyperparameter tuning which helps accomplish highest level of accuracy for prediction model. The code for ANN model implementation also includes callback for early stopping, which prevents model from overfitting and underfitting problem and improve regularization.

As depicted above MSE and MSLE for the prediction model improves drastically showing effect on the price prediction as below:

ANN model before hyperparameter tuning	**ANN model after hyperparameter tuning**
Number of epochs: 5	Number of epochs: 50 with EarlyStopping callback function
Loss function: Started at Loss: 0.03 - mse: 0.46	Loss function: Started at loss: 0.0620 - mse: 0.58
Loss function at final epoch loss: 0.02 - mse: 0.30	Loss function at final epoch loss: 0.01 - mse: 0.25

Plotting training loss as shown below:

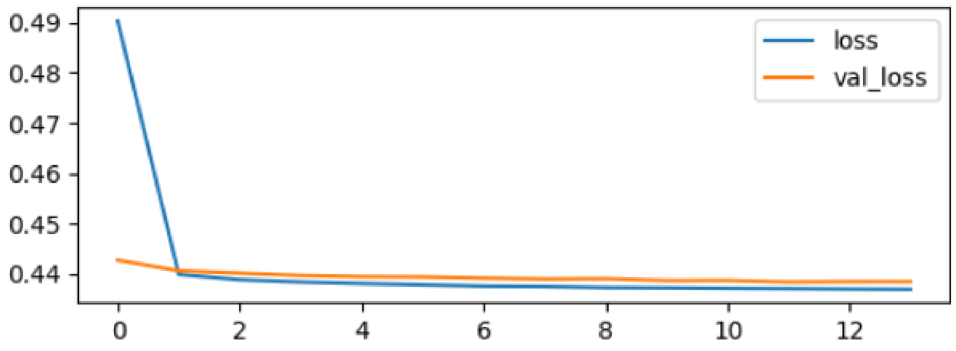


Figure 20 Loss during training the ANN model

- **Actual Vs predicted price prediction**

Evaluating the model on test data achieves the output as

Test Loss: 0.0239 and Test Accuracy: 0.3793

The results for mean of absolute errors between labels and predictions are:

Mean Absolute Error (MAE): [0.5979437 0.67535126 0.5838018 ... 0.58861053 0.6056334
0.59718776]

With the scalar value of Mean Absolute Error (MAE): 0.7201 , over entire test dataset.

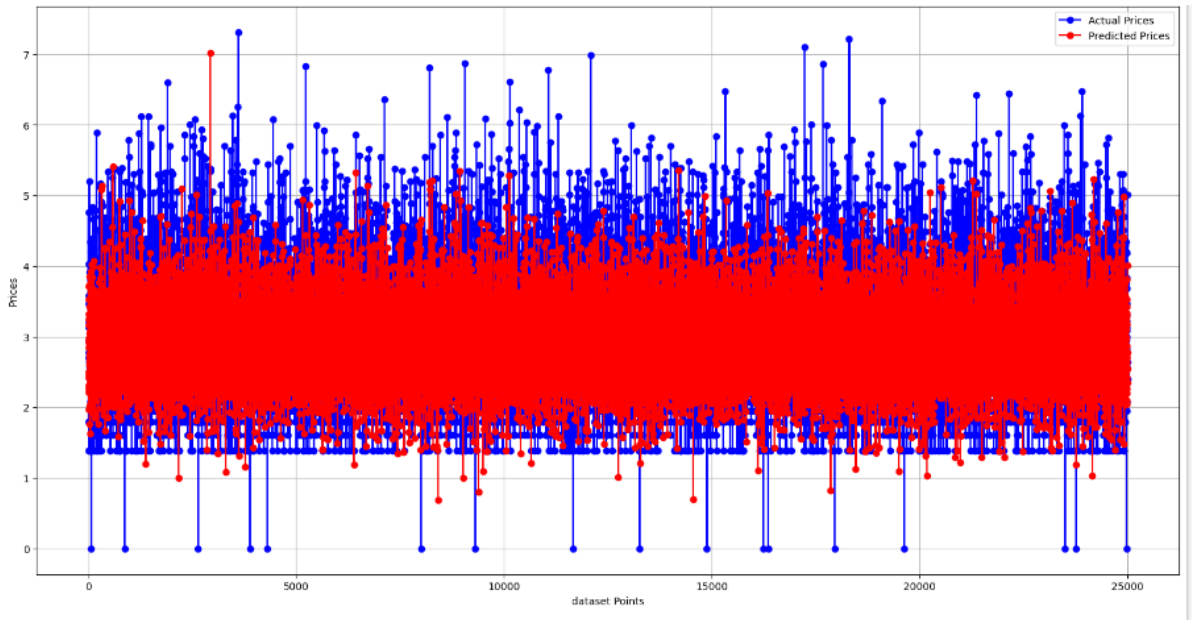
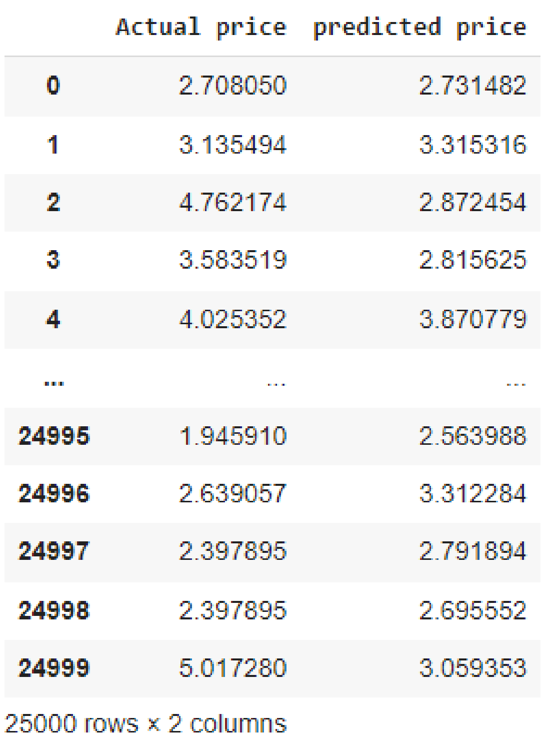


Figure 21: Actual prices vs predicted prices (Line Plot)

As the line plot graph depicts the two-line in the graph is intersecting and overlapping each other suggest excellent accuracy between the actual price and the price predicted by the developed model. The prediction model predicts price with loss of 0.023, which can be seen whilst there were high actual prices depicted with the blue peaks visible on the graph. The predicted value slightly peaks just below the actual price.

Here is the tabular representation of the price prediction after hyper tuning the parameters of the ANN model and autoencoder algorithm.



	Actual price	predicted price
0	2.708050	2.731482
1	3.135494	3.315316
2	4.762174	2.872454
3	3.583519	2.815625
4	4.025352	3.870779
...	...	...
24995	1.945910	2.563988
24996	2.639057	3.312284
24997	2.397895	2.791894
24998	2.397895	2.695552
24999	5.017280	3.059353

25000 rows × 2 columns

- **Learning curve**

Learning curve for the price predicted plots the model's performance on the training showing the training error and cross-validation error as a function of the number of training examples. The learning curve is plotted and initially there is loss with around than 0.30 MSE which gradually reduces as the model is trained with more epochs to below 0.20 and thus suggesting that the model generalizes to the unseen data.

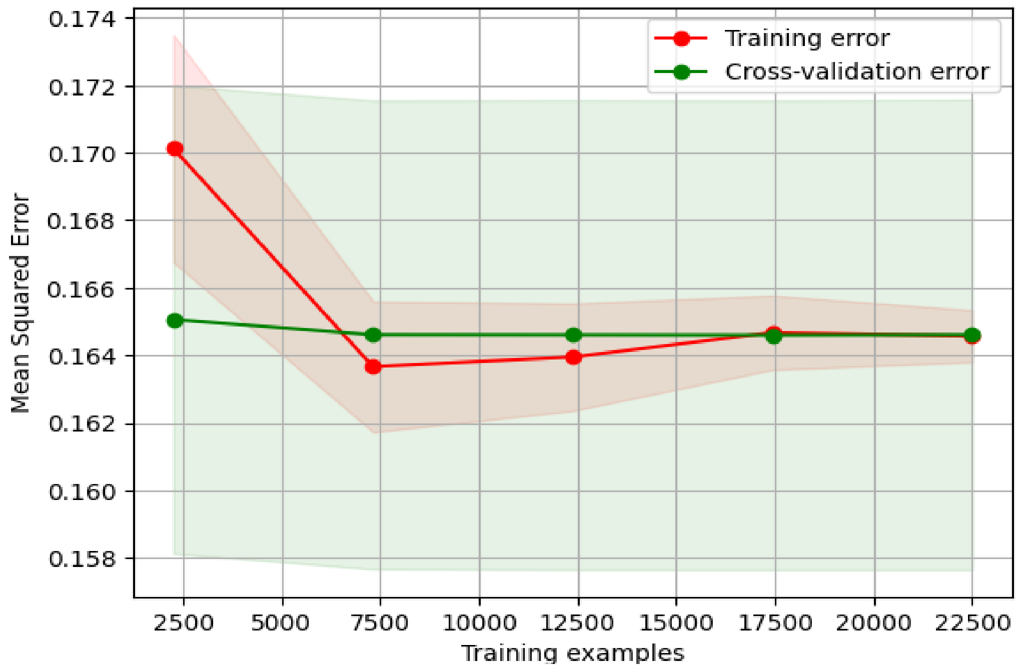


Figure 22: Learning curve for price prediction

This resultant graph depicts converge at the end, thus achieving excellent accuracy.

All this result reflects the potential of the ANN model build with autoencoder feature reduction technique can be suggested as a good method for price prediction.

- **Scatter plot**

The below scattered plot graph illustrates, on x axis is the actual price and on y-axis is the predicted price with regression line overlaying visualizing the actual price prediction is scatter around the regression line.

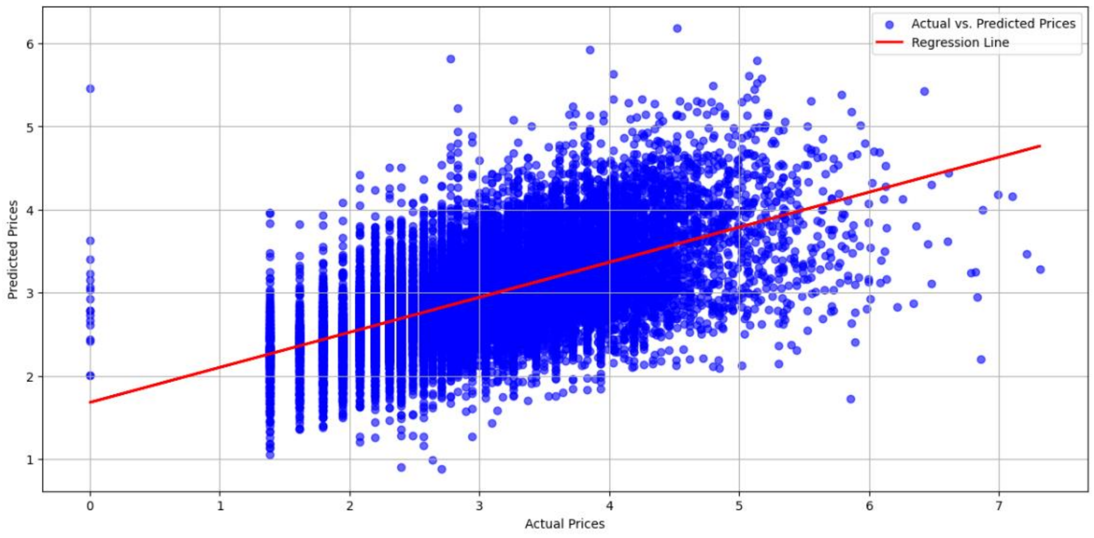


Figure 23: Actual vs predicted prices (over Regression Line)

- **Evaluation histogram:**

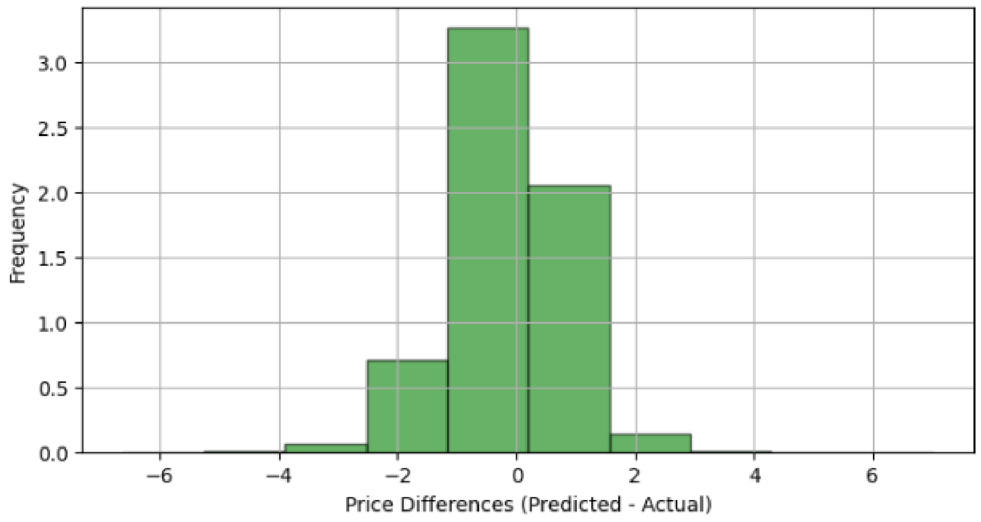


Figure 24: Distribution of price prediction difference with the actual price

The above diagram evaluation of the performance of the ANN model. Here the model calculates the difference between the actual price and predicted price. Here, the model takes into consideration the first 100 records for evaluation. Here, as seen in the above figure the price distribution difference ranges from -2 to 2, making prediction with difference of 2 points in the price.

Overall, the model performance with highest level of accuracy with maximum prediction are made reflected in the tallest bar near to the 0 difference and the model does not overestimate or underestimate the prices for the give marketplace data.

4.2 Experiment phase II- Sale time Predictions

This section represents the preliminary phase of the research, and it is concerned with forecasting sale time for various marketplace products through the processing of test data. The following part remarks deeper into improving the precision of these sale time projections, spanning a broader range of findings.

The following were the primary goals of this experimental phase II:

1. To evaluate the efficiency of the established Artificial Neural Network (ANN) model in sale time forecasting, with the goal of assuring the least amount of variation from accurate results.

2. To maintain data integrity by minimising data loss within a loss rate range of 0.00 to 0.99.

3. To optimise the prediction model's speed in accordance with the previously stated goals in chapter 1: 1.1. Objectives

The approach used in this step is graphically illustrated, effectively illustrating the model's efficiency, accuracy, and performance.

There is approximately 1043742 of the data records including various ecommerce product with their detailed description, category, price and with few more features. For the test results, the data is divided into 70% for training the model and remaining 30% which is around 342575 records for training and evaluation of the results. Below are the stepwise results obtained by applying the methods discussed in Chapter 3. Research methods.

This experiment for phase II model develops the data for the ANN model by making use of the first and second step discussed in the experiment phase I, with few additional codes that produces output elaborated below:

Step 1 and 2: Output of data collection , data interpretation and results of pre-processed dataset

For experiment phase II, step 1 and step 2 covers similar process towards data collection and pre-preprocessing dataset, as discussed in experiment phase I,

This experiment considers 80000 and 20000 records for training and testing for prediction model respectively, this is to limit to limit records for preprocessing to avoid systems to crash out of resource.

```
saleTime_train=saleTime_train[:80000]
saleTime_test=saleTime_test[:20000]
```

Above lines of code can be comments to test model with entire dataset which is as follows in the format of number of rows, columns (feature set).

```
(706000, 10)
(342575, 10)
```

Furthermore, in this step 2 of pre-processing of the dataset the sale time prediction model has an additional logical transformation process for calculating sale time for marketplace products. The computational transformation defines a dynamic column "sale_time." This is the actual output column to predict by the model in the experiment phase II. Code for this section define under the "Logic for sale time calculation- output feature" section for sale time calculation- output feature."

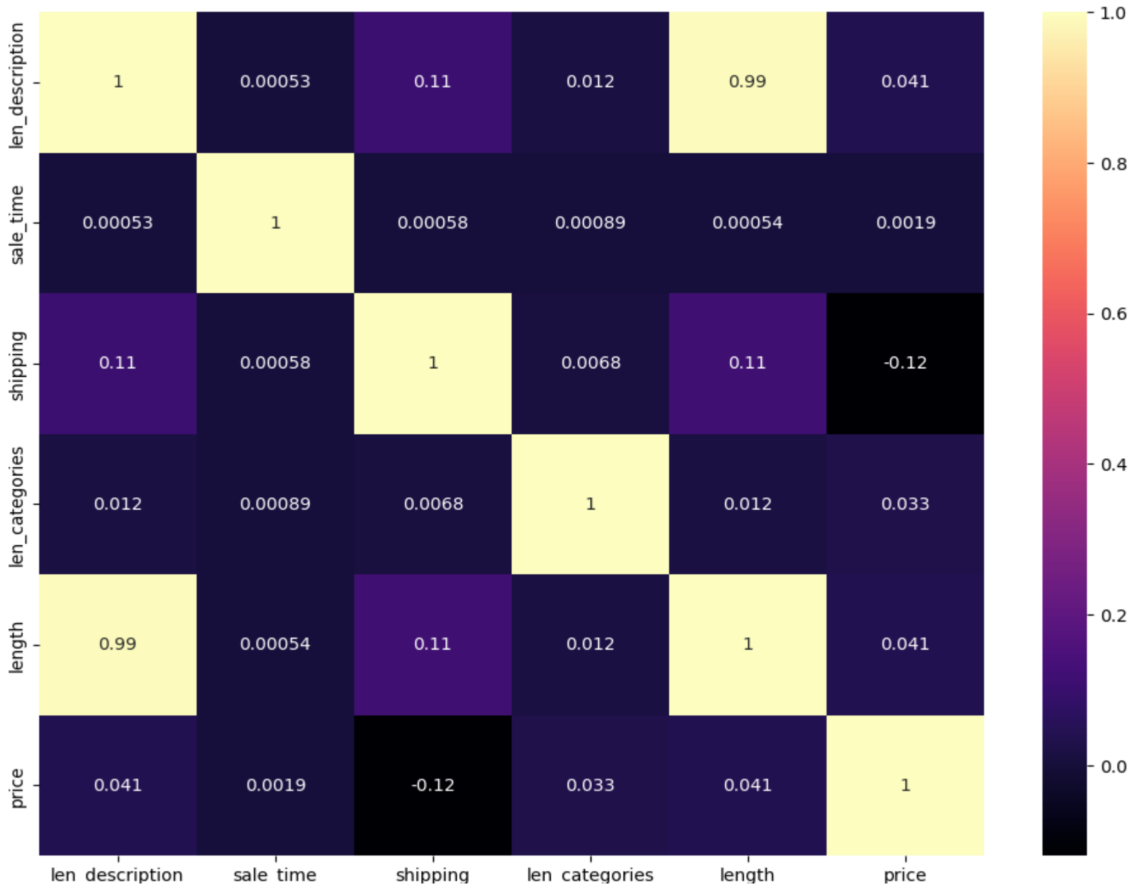


Figure 25: The correlation matrix for important features

The above figure provides the outstanding features representing the highest correlation with feature dataset (1 being the highest i.e., tile with lighter colour and 0 being the lowest i.e., darker colour on the scale for the above graph).

Based on the above correlation matrix, the model determines prominent features i.e., with the length of 158 features with price being the independent column which will be added to the final pre-processed dataset.

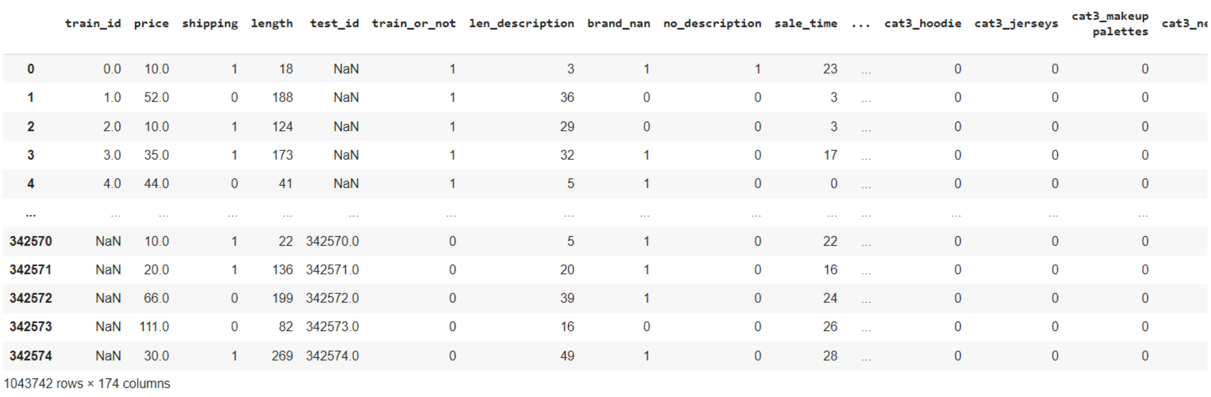
The output new features used to create columns for each parent features are discussed in the phase I of the experiment.

The final output after preprocessing all new features into a binary and numeric datatype is as follows:

Output 1: information of the final pre-processed DataFrame:

Command	result
data.info ()	<class 'pandas.core.frame.DataFrame'> Int64Index: 1043742 entries, 0 to 342574 Columns: 174 entries, train_id to cat3_tracksuits & sweats dtypes: float64(3), int64(88), uint8(83) memory usage: 815.2 MB None
data.shape	(1043742, 174)

Printing the information of the final pre-processed Data Frame ready for modelling and data interpretation:



	train_id	price	shipping	length	test_id	train_or_not	len_description	brand_nan	no_description	sale_time	...	cat3_hoodie	cat3_jerseys	cat3_makeup palettes	cat3_ne
0	0.0	10.0	1	18	NaN	1	3	1	1	23	...	0	0	0	
1	1.0	52.0	0	188	NaN	1	36	0	0	3	...	0	0	0	
2	2.0	10.0	1	124	NaN	1	29	0	0	3	...	0	0	0	
3	3.0	35.0	1	173	NaN	1	32	1	0	17	...	0	0	0	
4	4.0	44.0	0	41	NaN	1	5	1	0	0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
342570	NaN	10.0	1	22	342570.0	0	5	1	0	22	...	0	0	0	
342571	NaN	20.0	1	136	342571.0	0	20	1	0	16	...	0	0	0	
342572	NaN	66.0	0	199	342572.0	0	39	1	0	24	...	0	0	0	
342573	NaN	111.0	0	82	342573.0	0	16	0	0	26	...	0	0	0	
342574	NaN	30.0	1	269	342574.0	0	49	1	0	28	...	0	0	0	

1043742 rows × 174 columns

The model then processes the pre-processed data frame (i.e., illustrated above) by splitting into the train and test dataset and prepares the model for ANN model design.

After splitting, the data frame is converted to numpy.ndarray and the model obtains four sets of output:

Shape of new dataset:	
Training dataset	(701167, 170)
Testing dataset	(342575, 170)

This represents the new dataset has 701167 rows for training and 342575 rows for testing the trained model, and 170 column or features. These 170 output features of pre-processed data will be the input nodes for the first layer of the ANN model framework.

Step 3: Results of reduction with autoencoder technique,

Like the experiment in the phase I, the autoencoder runs with 32 batch size, 50 epochs and with the early stopping regularization feature. This means the model stops training if there is no improvement in loss for three consecutive epochs, as depicted in the below mentioned code

```
 tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
```

The first epochs for autoencoder starts with loss of 0.49 as follows

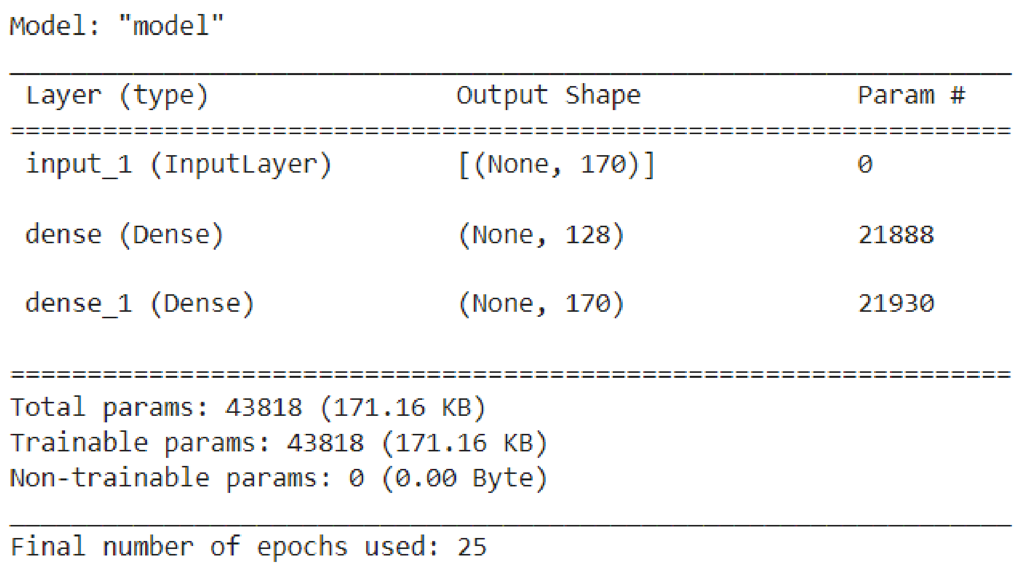
```
Epoch 1/50
2483/2483 [==============================] - 4s 2ms/step - loss: 0.5366
- val_loss: 0.4963
Epoch 2/50
2483/2483 [==============================] - 5s 2ms/step - loss: 0.4930
- val_loss: 0.4946
Epoch 3/50
2483/2483 [==============================] - 4s 2ms/step - loss: 0.4920
- val_loss: 0.4943
Epoch 4/50
2483/2483 [==============================] - 4s 2ms/step - loss: 0.4916
- val_loss: 0.4934
Epoch 5/50
2483/2483 [==============================] - 5s 2ms/step - loss: 0.4912
- val_loss: 0.4935
```

While gradually decreasing the loss to 0.48 within 28/50 epochs

```
Epoch 24/50
2483/2483 [==============================] - 4s 2ms/step - loss: 0.4850
- val_loss: 0.4865
Epoch 25/50
2483/2483 [==============================] - 4s 1ms/step - loss: 0.4849
- val_loss: 0.4863
Epoch 26/50
2483/2483 [==============================] - 4s 2ms/step - loss: 0.4849
- val_loss: 0.4867
Epoch 27/50
2483/2483 [==============================] - 4s 2ms/step - loss: 0.4849
- val_loss: 0.4865
Epoch 28/50
2483/2483 [==============================] - 4s 2ms/step - loss: 0.4848
- val_loss: 0.4865
2483/2483 [==============================] - 3s 1ms/step
625/625 [==============================] - 1s 890us/step
```

As discussed in the research methodology, the model implements MSE metrics for calculating the loss and ADAM optimizer, which minimize the loss function by updating the model's parameters.

Step 4: Output of building an ANN model for sale time prediction



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 170)]             0

 dense (Dense)               (None, 128)               21888

 dense_1 (Dense)             (None, 170)               21930

=================================================================
Total params: 43818 (171.16 KB)
Trainable params: 43818 (171.16 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Final number of epochs used: 25
```

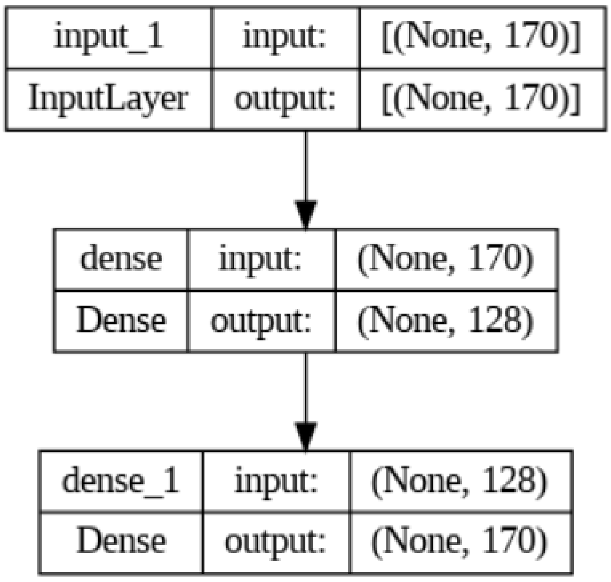


Figure 26: Autoencoder model architecture for sale time prediction

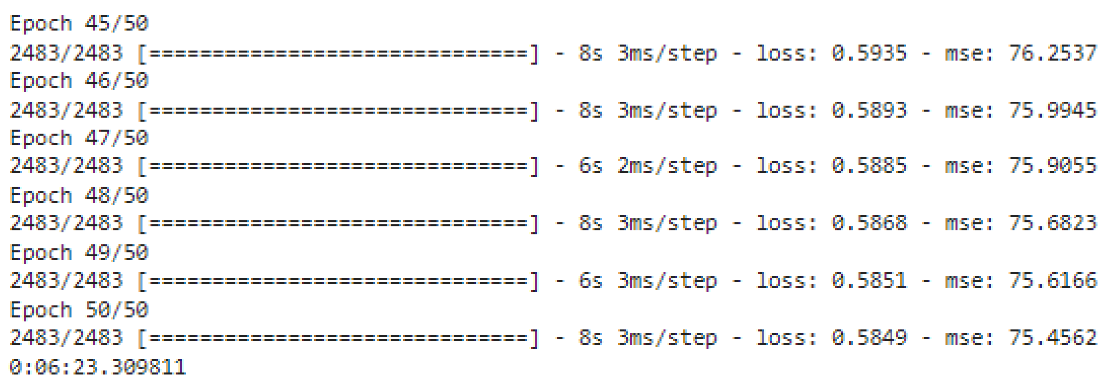
Step 5: Evaluation Metrics of the Proposed System and visualization

- **Mean Squared Logarithmic Error (MSLE) and Mean Squared Error (MSE)**

On Compilation of the model over ‘Adam’ optimizer, loss as 'mean_squared_logarithmic_error' and metrics = ['mse'], the model produces output as below snapshot

```
Epoch 1/50
2483/2483 [==============================] - 11s 3ms/step - loss:
0.9484 - mse: 91.8145
Epoch 2/50
2483/2483 [==============================] - 8s 2ms/step - loss: 0.9873
- mse: 91.8039
Epoch 3/50
2483/2483 [==============================] - 9s 3ms/step - loss: 0.9877
- mse: 91.8112
```

```
Epoch 4/50
2483/2483 [==============================] - 8s 2ms/step - loss: 0.9877
- mse: 91.8088
```
Continued until 50 epochs.



```
Epoch 45/50
2483/2483 [==============================] - 8s 3ms/step - loss: 0.5935 - mse: 76.2537
Epoch 46/50
2483/2483 [==============================] - 8s 3ms/step - loss: 0.5893 - mse: 75.9945
Epoch 47/50
2483/2483 [==============================] - 6s 2ms/step - loss: 0.5885 - mse: 75.9055
Epoch 48/50
2483/2483 [==============================] - 8s 3ms/step - loss: 0.5868 - mse: 75.6823
Epoch 49/50
2483/2483 [==============================] - 6s 3ms/step - loss: 0.5851 - mse: 75.6166
Epoch 50/50
2483/2483 [==============================] - 8s 3ms/step - loss: 0.5849 - mse: 75.4562
0:06:23.309811
```

The model here shows the modified version of the ANN model covering 50 epochs for training the model.

The training output above showed difference of 4-5 days in the sale time prediction and took longer time for processing the given test and train dataset. This prediction was improved by hyper parameter tunning which includes change in the epochs, batch size, including EarlyStopping as callback function while modelling ANN model for sale time prediction model. Below is the output of training the ANN model for starting 5 and last 50 epochs

```
Epoch 1/50
2483/2483 [==============================] - 4s 1ms/step - loss: 0.7177
- mse: 88.5288
Epoch 2/50
2483/2483 [==============================] - 3s 1ms/step - loss: 0.7043
- mse: 87.5365
Epoch 3/50
2483/2483 [==============================] - 4s 2ms/step - loss: 0.7030
- mse: 87.4157
Epoch 4/50
2483/2483 [==============================] - 3s 1ms/step - loss: 0.7010
- mse: 87.2407
Epoch 5/50
2483/2483 [==============================] - 3s 1ms/step - loss: 0.6989
- mse: 86.9476
```

Continued till 50 epochs

```
poch 45/50
2483/2483 [==============================] - 3s 1ms/step - loss: 0.5875
- mse: 75.8694
Epoch 46/50
2483/2483 [==============================] - 3s 1ms/step - loss: 0.5866
- mse: 75.9182
Epoch 47/50
2483/2483 [==============================] - 3s 1ms/step - loss: 0.5848
- mse: 75.7165
```

```
Epoch 48/50
2483/2483 [==============================] - 4s 2ms/step - loss: 0.5812
- mse: 75.5247
Epoch 49/50
2483/2483 [==============================] - 3s 1ms/step - loss: 0.5799
- mse: 75.3123
Epoch 50/50
2483/2483 [==============================] - 3s 1ms/step - loss: 0.5783
- mse: 75.2813
0:03:22.517025
```

There has slight decrease in the loss for 0.5849, mse 75.4562 to 0.5783, mse 75.2813. This is the effect of hyperparameter tuning which showed higher impact on the sale time prediction (sale time compared below) helped accomplish highest level of accuracy for prediction model.

The code for ANN model implementation also includes callback for early stopping, which prevents model from overfitting and underfitting problem and improve regularization.

- **Plotting training loss as shown below:**

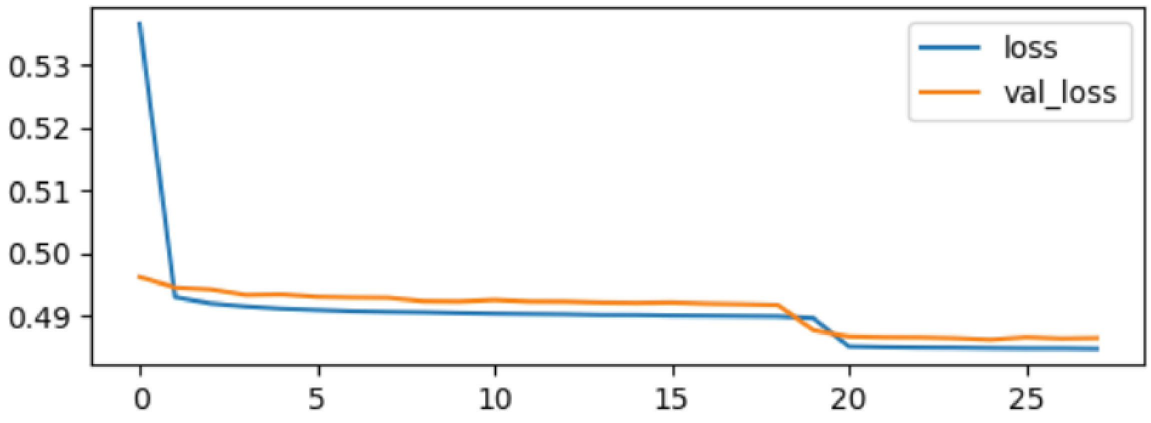


Figure 27: Loss during training the ANN model

- **Actual Vs predicted sale time prediction**

Evaluating the model on test data achieves the output as

```
625/625 [==============================] - 1s 1ms/step - loss: 0.8381 -
mse: 100.6790
Test Loss: 0.8381, Test Accuracy: 100.6790
```

The results for mean of absolute errors between labels and predictions are:

```
Mean Absolute Error (MAE): [ 7.688086   7.858128  19.706907  ...
10.623822   7.538662   7.5208225]
```

With the scalar value of Mean Absolute Error (MAE): `8.2855`, calculated over the test dataset.

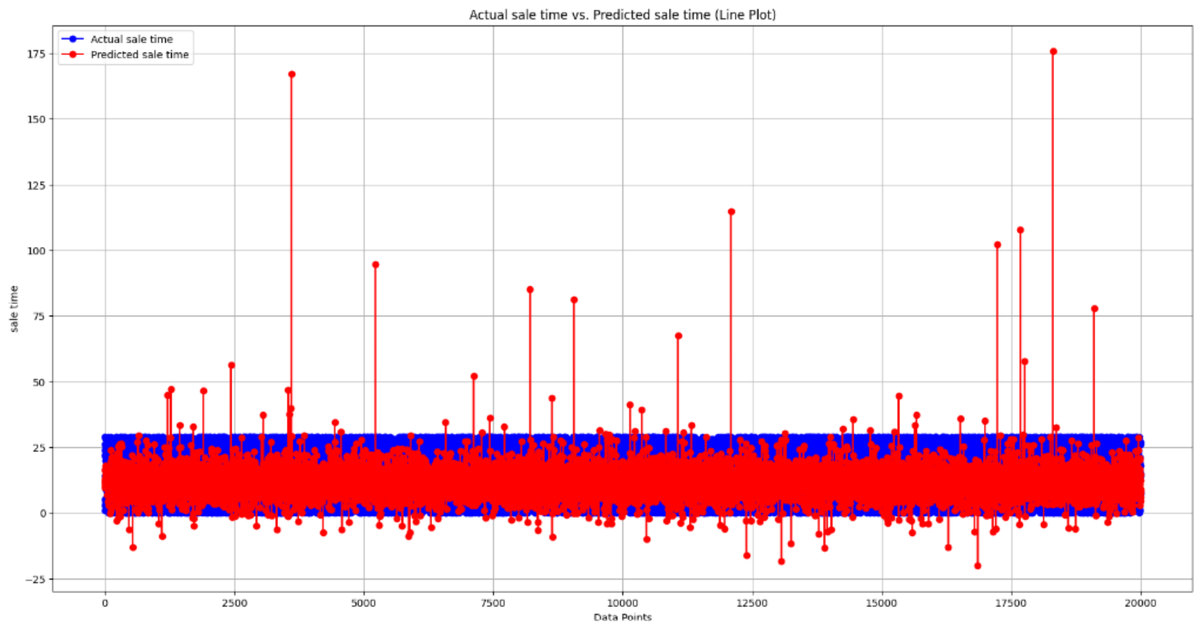
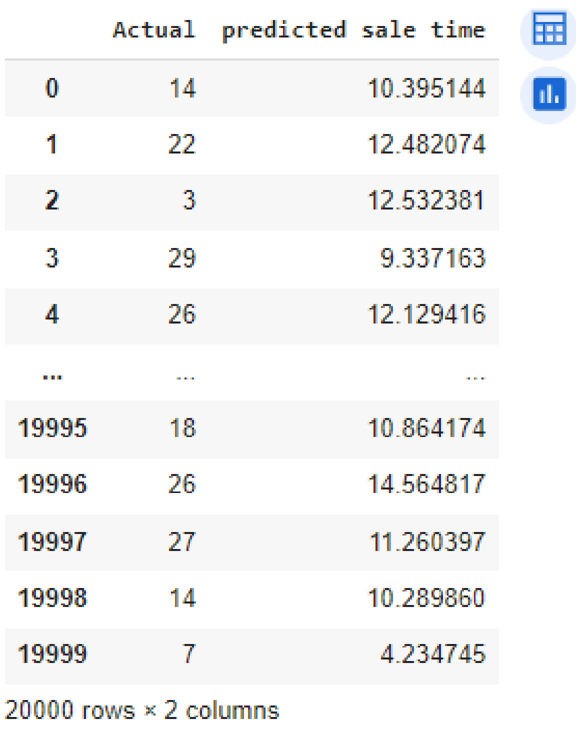


Figure 28: Actual sale time vs predicted sale time (Line Plot)

As the line plot graph depicts the two-line in the graph is intersecting and overlapping each other suggest excellent accuracy between the actual sale time and the sale time predicted by the developed model.

The prediction model predicts price with loss of 0.8182, which can be seen whilst there was high predicted sale time (depicted with the red peaks visible on the graph) compared to the actual sale time. Below mentioned is the tabular representation of the sale time prediction comparison with actual sale time for the products.

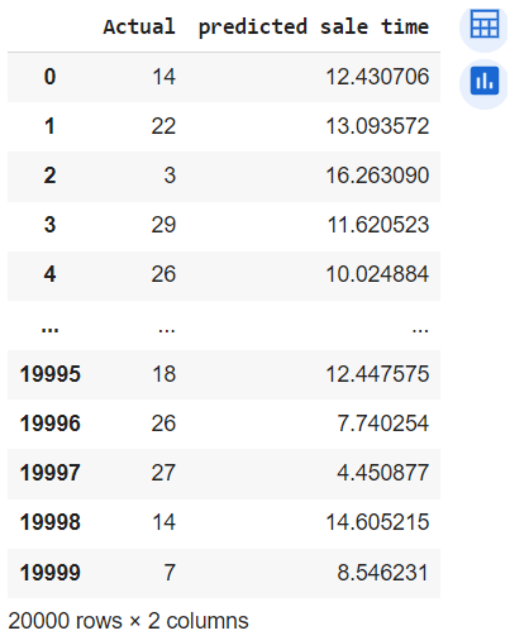
Sale time prediction accuracy before hyper parameter tuning



	Actual	predicted sale time
0	14	10.395144
1	22	12.482074
2	3	12.532381
3	29	9.337163
4	26	12.129416
...	...	...
19995	18	10.864174
19996	26	14.564817
19997	27	11.260397
19998	14	10.289860
19999	7	4.234745

20000 rows × 2 columns

Sale time prediction accuracy after hyper parameter tuning



	Actual	predicted sale time
0	14	12.430706
1	22	13.093572
2	3	16.263090
3	29	11.620523
4	26	10.024884
...	...	...
19995	18	12.447575
19996	26	7.740254
19997	27	4.450877
19998	14	14.605215
19999	7	8.546231

20000 rows × 2 columns

- **Learning Curve**

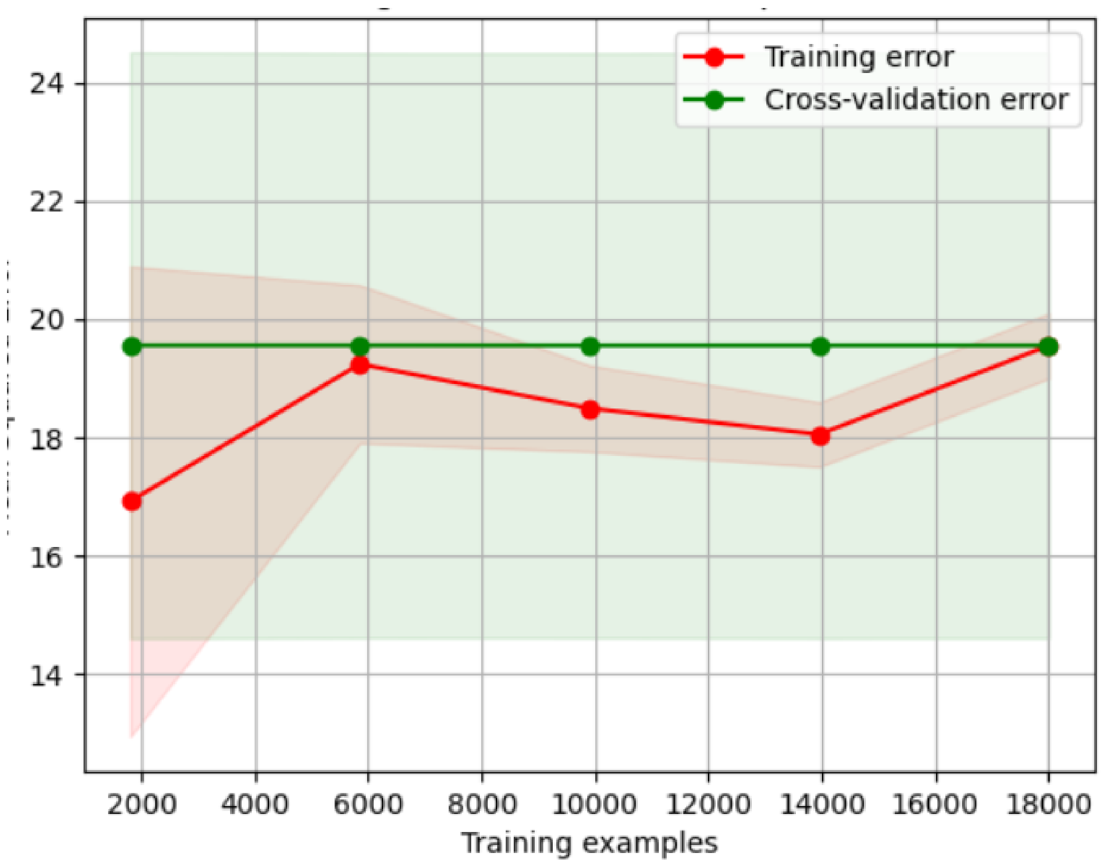


Figure 29: Learning curves for sale time prediction

- **Scatter plot**

The below scattered plot graph illustrates, on x axis is the actual sale time and on y-axis is the predicted sale time with regression line overlaying visualizing the actual sale time prediction is scatter around the regression line.

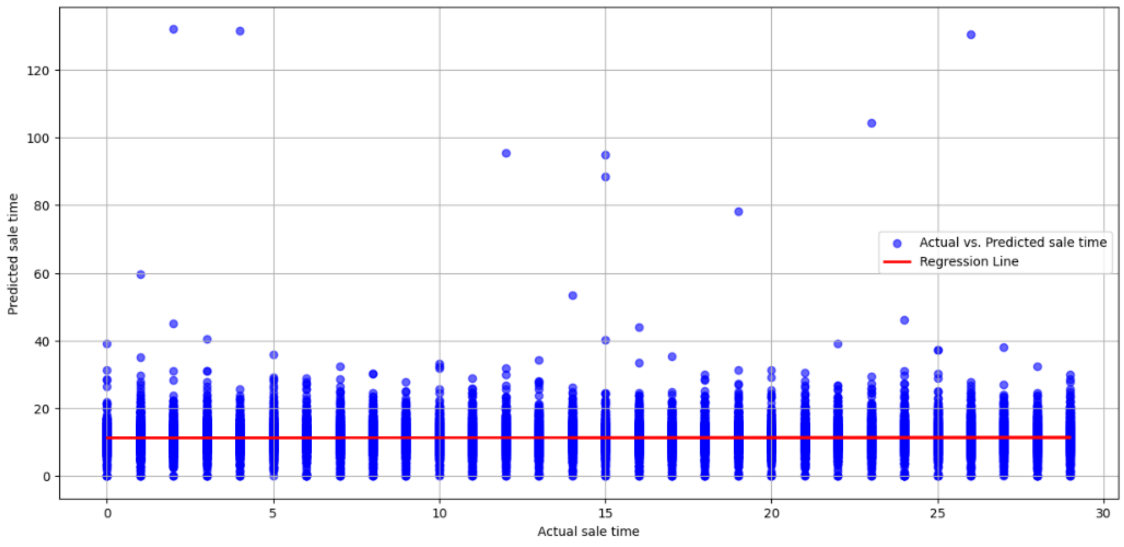


Figure 30: Actual vs predicted sale time (with regression line)

- **Evaluation histogram**

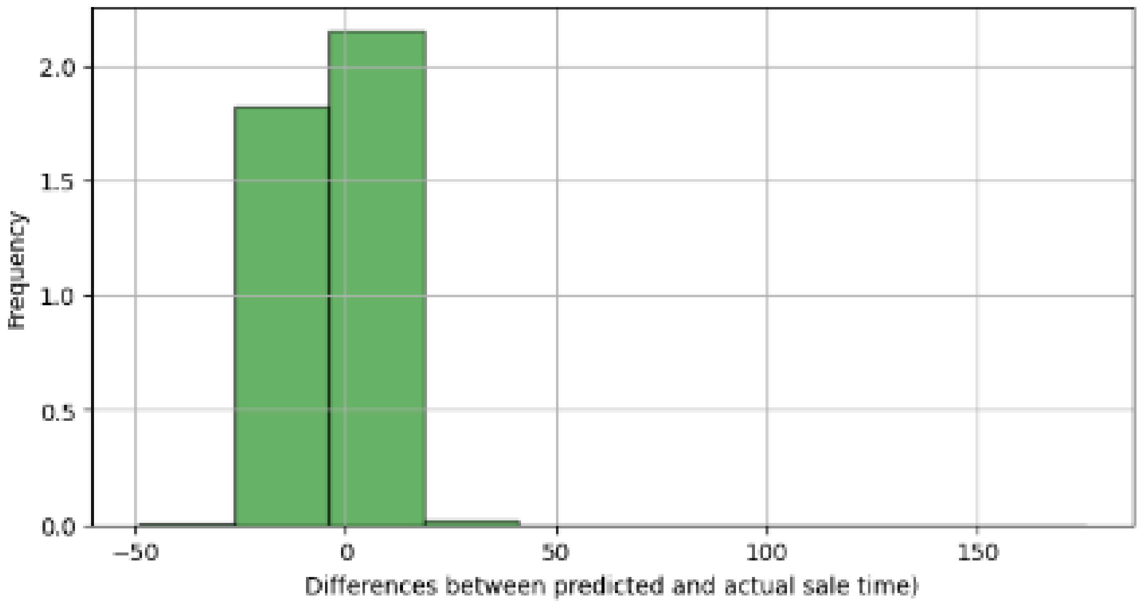


Figure 31: Distribution for predicted sale time difference with the actual sale time

The above diagram evaluation of the performance of the ANN model. Here the model calculates the difference between the actual price and predicted price. Here, as seen in the above figure the price distribution difference ranges from -20 to 20, making prediction with difference of 20 points in the sale time prediction.

4.3 Conclusion

In conclusion, after performing the experiment in phases I and II, the model successfully performed price and sale time predictions for the Mercari Marketplace dataset. With minor preprocessing of the feature set, the ANN model provided results with a minimal loss of 0.5801 for sale time prediction and even less for price prediction, i.e., a loss of 0.0168. Further, the model implements an autoencoder technique, which resulted in a major performance change in the ANN model, increasing the overall result of the model, with the capacity to process a dataset as large as one lakh records, presented in unformatted textual and descriptive format. On the other hand, initially, as the model was implemented, there were major performance issues and differences in the predicted values for both experiments. These issues were overcome through hyperparameter tuning of the model. This indeed achieves the aim of predicting the sale time and price for the marketplace data. The next chapters 5 and 6 present these outcomes through detailed discussion and evaluation comments.

Chapter 5: Discussion

This segment thoroughly examines the outcomes presented in Chapter 4 in the context of the project's aims. The findings are also placed within a larger context of recent theoretical and practical research, as discussed in Chapter 2. This section also explores the research's ramifications with the study's validity, applicability, generalizability, conclusions, and suggestions. Moreover, the main goal of this project, which is price and sale time prediction from the application of an Artificial Neural Network (ANN) model in conjunction with Autoencoder for feature reduction implementation, will be validated according to these results.

5.1 Alignment with Objectives

This research aims to create a predictive model that forecasts the price and sale time in the e-commerce industry. These objectives have been accomplished using an Artificial Neural Network (ANN) model with autoencoder for feature reduction. The model performs noticeably better than a conventional method and has high accuracy. This alignment between the objectives and outcomes underscores the efficacy of the approach.

5.2 Comparative Analysis

In contrast to the other theoretical and applied studies described in Chapter 2, this study presents a combination of machine learning techniques of ANN and autoencoder. As discussed in Chapter 2, previous research has explored the ANN model for predictive tasks and Autoencoder for feature reductions. The fusion of these represents a unique contribution. The main advantage of using the ANN model is that it is easy to use and understand compared to other statistical methods. This research implements the ANN model with Backpropagation, used widely in solving various classifications, making the model more precise for prediction while still being simple for implementation. However, the ANN alone cannot deal with uncertainties.

Although the above observation has reflected in various research, we used the simple ANN model for our prediction experiment. This shows the implementation of ANN with Autoencoder feature that was discussed in the implementation model for price and sale time prediction in the project proposal. The which showed converging learning curve stating that the model is not learning from the data well, even with more data, because it is too simplistic to capture the underlying patterns in the data. additional data has no effect on the model, and adding additional training instances has no effect on validation error. Also, addition of autoencoder layer or increasing the number of hidden layers did not improve the Mean squared error loss. (Wikipedia contributors, 2023). This imply that the model has hit its limit in terms of learning from the given data, and that additional data would be useless.

Compared with the methods discussed in the literature review and the model developed above, the final developed model has centred on the development of a predictive model that enhances the accuracy of price and sale time predictions in the e-commerce sector. Through integrating the Artificial Neural Network (ANN) model with autoencoder for feature reduction.

It has succeeded in achieving the desired output. The model thus demonstrates an elevated level of accuracy while performing efficiently. From a practical user perspective, our research was able to address the fundamental user requirements in the e-commerce domain.

This precision in the price predictions will help businesses optimize their marketing and sales strategy, and precise sale time predictions will facilitate supply and inventory management in all types of retail industries. Furthermore, users will benefit from the competitive prices and prompt availability of product that suits their need more easily.

The choice to employ autoencoder for feature reduction has proven to be a highly effective decision, it has not only simplified the model implementation but also enhanced the power of the prediction model to manage a large volume of data without any distribution to the model's learning pattern. This makes it more powerful and can be implemented in all other areas of business extending its scope easily.

Our research has far-reaching ramifications. The current combination of ANN with autoencoder can promise diverse applications ranging from retail to healthcare and finance domains. To improve the predictive model, future work could focus on the generalization of the model across different datasets and industries to ascertain the model's versatility. The future scope has been broadly discussed in Chapter 6.4.

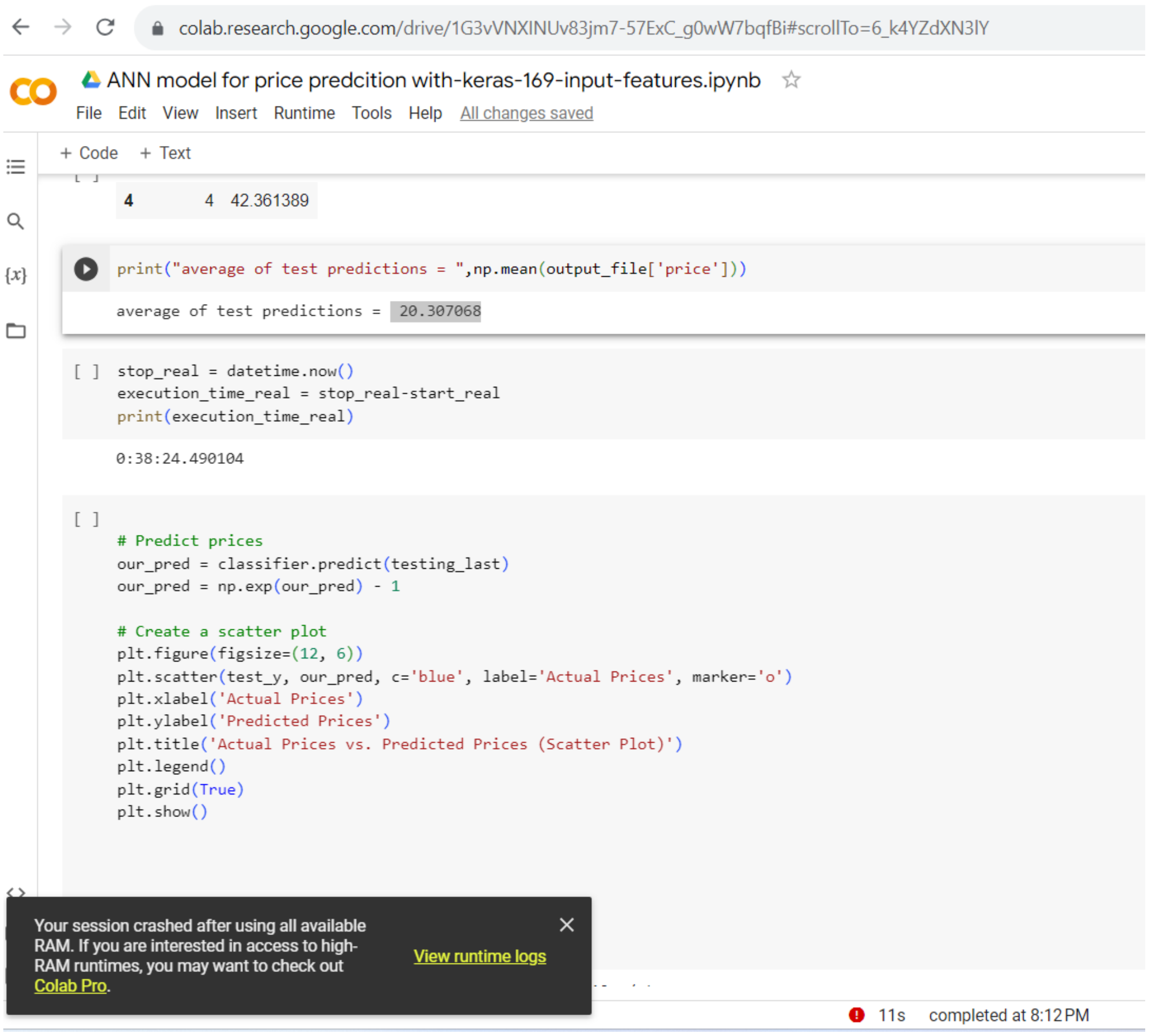
5.3 Implications in the project

ML modelling has challenges, messy real-world data, complex data pipelines, choosing the right algorithms, tuning hyperparameters, and avoiding overfitting. But advances in deep learning and neural networks enable more powerful forecasting models. Recurrent neural networks model time-series dynamics. Convolutional networks extract visual patterns from product images. Ensemble methods combine multiple models. With growing computing power, larger ML models are becoming feasible for big e-commerce data. Though it requires expertise, properly implemented ML can significantly improve demand planning and profits.

While machine learning shows promise for improving sales predictions in e-commerce, most current research has focused on using ML for either sales time forecasts or optimal price modelling separately. However, optimally pricing products and predicting how quickly they will sell are inherently connected problems. Pricing an item too high means it may sit unsold for longer. Pricing it too low leaves profits on the table. To maximize both velocities of sales and revenue, joint modelling of sales time and pricing is crucial and challenging as well.

Therefore, this thesis aims to address the key research gap of jointly modelling sales time and optimal pricing on large, real-world marketplace data. The complex dependencies between price, demand, seasonality, inventory, and other factors require a flexible yet robust machine-learning approach. By leveraging deep neural networks, this work will explore multi-task and graph-based models to share representations and capture relationships in the data. Improving joint sales and price forecasts would enable marketplaces to maximize revenues while maintaining availability and fast delivery times for customers. This work will benchmark predictive accuracy on live marketplace data compared to current single-task models.

Secondly, the crucial accomplishment of this model was dealing with large datasets demanding resource-intensive tasks. As system demanded more resource it was challenging to make model work for complex , informal dataset.


colab.research.google.com/drive/1G3vVNXINUv83jm7-57ExC_g0wW7bqfBi#scrollTo=6_k4YZdXN3lY

ANN model for price predcition with-keras-169-input-features.ipynb

```
4        4   42.361389
```

```
print("average of test predictions = ",np.mean(output_file['price']))
```

```
average of test predictions =  20.307068
```

```
stop_real = datetime.now()
execution_time_real = stop_real-start_real
print(execution_time_real)
```

```
0:38:24.490104
```

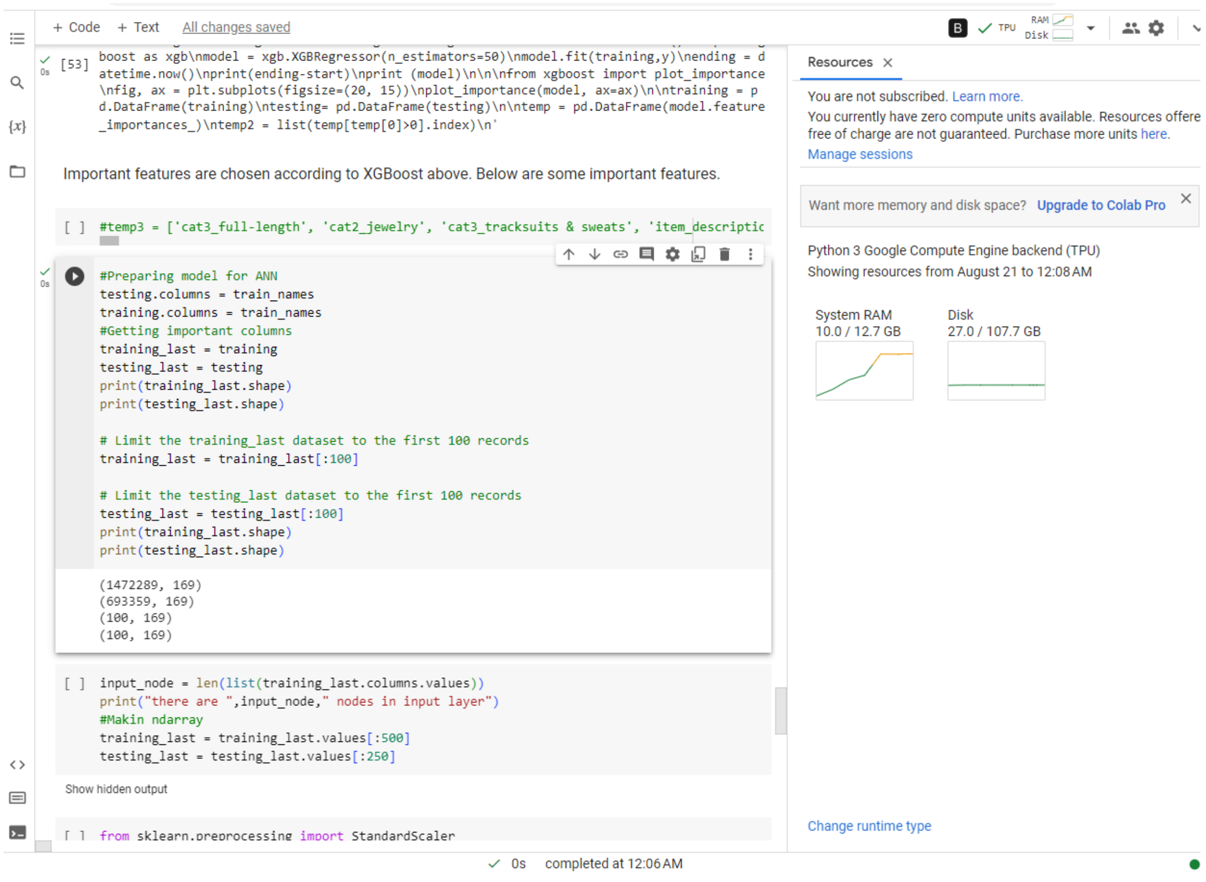
```
# Predict prices
our_pred = classifier.predict(testing_last)
our_pred = np.exp(our_pred) - 1

# Create a scatter plot
plt.figure(figsize=(12, 6))
plt.scatter(test_y, our_pred, c='blue', label='Actual Prices', marker='o')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual Prices vs. Predicted Prices (Scatter Plot)')
plt.legend()
plt.grid(True)
plt.show()
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

Figure 32: Screenshot of the model demanding high compute resources

As seen in the above Figure 32 it can been illustrated that the model demanded high resource to process the dataset. The system crashed while using all available resource which was around 12 GB while the prediction average still being over 20, calculated as mean over the predicted price.



```
[53] boost as xgb\nmodel = xgb.XGBRegressor(n_estimators=50)\nmodel.fit(training,y)\nending = d
     atetime.now()\nprint(ending-start)\nprint (model)\n\n\nfrom xgboost import plot_importance
     \nfig, ax = plt.subplots(figsize=(20, 15))\nplot_importance(model, ax=ax)\n\ntraining = p
     d.DataFrame(training)\ntesting= pd.DataFrame(testing)\n\ntemp = pd.DataFrame(model.feature
     _importances_)\ntemp2 = list(temp[temp[0]>0].index)\n'
```

Important features are chosen according to XGBoost above. Below are some important features.

```
[ ] #temp3 = ['cat3_full-length', 'cat2_jewelry', 'cat3_tracksuits & sweats', 'item_descriptic
```

```
#Preparing model for ANN
testing.columns = train_names
training.columns = train_names
#Getting important columns
training_last = training
testing_last = testing
print(training_last.shape)
print(testing_last.shape)

# Limit the training_last dataset to the first 100 records
training_last = training_last[:100]

# Limit the testing_last dataset to the first 100 records
testing_last = testing_last[:100]
print(training_last.shape)
print(testing_last.shape)
```

```
(1472289, 169)
(693359, 169)
(100, 169)
(100, 169)
```

```
[ ] input_node = len(list(training_last.columns.values))
    print("there are ",input_node," nodes in input layer")
    #Makin ndarray
    training_last = training_last.values[:500]
    testing_last = testing_last.values[:250]
```

Show hidden output

```
[ ] from sklearn.preprocessing import StandardScaler
```

You are not subscribed. Learn more.
You currently have zero compute units available. Resources offere free of charge are not guaranteed. Purchase more units here.
Manage sessions

Want more memory and disk space? Upgrade to Colab Pro

Python 3 Google Compute Engine backend (TPU)
Showing resources from August 21 to 12:08 AM

System RAM
10.0 / 12.7 GB

Disk
27.0 / 107.7 GB

Change runtime type

✓ 0s completed at 12:06 AM

Figure 33: screenshot highlighting the high usage of available RAM

This issue was overcome by hyper parameter tuning discussed in the previous chapters, and slightly limiting the records for feature dataset making it robust to processed model for accuracy and performance.

5.4 Conclusion

In Conclusion, the price and sale time prediction model marks improvement in the prediction and with minor loss rate while dealing with the nonlinear dataset. Unlike other models discussed in Chapter 2: Critical Context- The Machine Learning model for predictions this model seems to work best for nonlinear dataset. On the other hand, it was easy to optimize the ANN model while achieving the desired precision. The potential of ANN with autoencoder proved to work in tackling real-world by attaining the objective of predicting price and sale time in experiment phase I and phase II. The exploration of the hybrid model has not only opened the door to improvise the machine learning models but will also flourish its application in various industries. Thus, this model explores the larger landscape of data driven decision making across wide variety businesses.

Overall, the project seemed to be a successful one, where the objectives to create architecture for price prediction and sale time forecast using neural network machine learning model based on marketplace data is achieved. Detailed evaluation and final thoughts can be found in the succeeding chapter.

Chapter 6: Evaluation, Reflections, and Conclusions

This chapter does a thorough examination of the entire project, including the initial objectives, literature study, methodology used, project planning, and more. The main goal is to analyse how well the project's goals were realized and to draw inferences from the work done. Also covered are the consequences of these conclusions, as well as observations on the project process and recommendations for future work.

The literature review served as the project's skeleton, directing the selection of approaches, and giving the study background. The evaluation reveals that the project's direction was impacted by the review of the body of literature. The use of autoencoder for feature reduction was encouraged by the availability of pertinent literature that demonstrated how helpful the algorithm is in improving feature selection accuracy.

The project's execution and planning were key factors in its success. The robustness of the results was guaranteed by the meticulousness of the data collection, pre-processing methods, model training, and evaluation approaches. The evaluation shows the benefit of the careful thought and execution that went into integrating the ANN model with autoencoder.

The project's work has validated the value of the hybrid ANN, autoencoder model for price and sale time prediction. The contribution of this project lies in the successful combination of two powerful machine learning techniques, resulting in a model with enhanced accuracy and computational efficiency. The implications of these conclusions extend to new domain application and industries reliant on accurate predictions, such as e-commerce and retail, where optimized pricing and sale timing strategies can lead to increased revenue.

6.1 Achievements of the project

The successes of this study in creating a pricing and sale time prediction model employing an integrated strategy of Artificial Neural Networks (ANN) with Autoencoder for feature reduction are noteworthy and may be summed up as follows:

Improved predicted Accuracy: This project's most significant accomplishment is the marked increase in predicted accuracy. This has been achieved with the combination of an ANN model, Autoencoder for efficient feature reduction and most importantly because of hyper parameter tuning. Tuning the hyper parameter played a crucial role om achieving the high-quality precision level. In practical applications, where accurate price and sale time predictions are essential for enterprises to maximize revenue and inventory management, this advancement is vital.

Feature Reduction Method: The choice to employ Autoencoder for feature reduction rather than customary autoencoders is a creative strategy. This decision was supported by the model's performance, which revealed that Autoencoder was effective at selecting features because it not only decreased computational complexity but also preserved relevant characteristics. This accomplishment casts doubt on the widespread belief that using complex procedures would always produce the best results and emphasizes the value of investigating easier-to-understand feature reduction techniques.

Useful Insights for Businesses: The project's successes offer practical commercial insights. It provides a powerful forecasting tool that may help businesses make data-driven decisions about pricing tactics and timing of sales. Businesses may improve revenue generation and keep a competitive edge in the market by correctly forecasting when to alter prices and when to launch sales promotions.

Participation to the Field of Predictive Analytics: By highlighting the possibilities of hybrid modelling, this study makes a substantial contribution to the field of predictive analytics. It demonstrates how combining various machine learning algorithms can provide effective results. This contribution is especially helpful for sectors and areas like e-commerce, banking, and supply chain management where precise predictions are crucial. The effectiveness of the model emphasizes how crucial carefully thought-out feature engineering and feature selection are in predictive modelling.

Additionally, it promotes the investigation of strategies to improve model interpretability and the inclusion of extraneous variables to further increase the contextual relevance of predictions. The implications of this endeavour go beyond its initial focus thanks to these new research directions.

As a result, the project's successes are noteworthy not only for the model created but also for their wider implications for the fields of machine learning and predictive analytics. They emphasize the relevance of predictive modelling, the significance of feature engineering, and the possibility of novel methodologies in enabling data-driven decision-making for enterprises and organizations.

6.2 Future work

Price prediction has been a challenge for the online marketplaces. Sellers often are optimistic and label products with selling price way higher than what it is meant to be. The approach is to automatically generate a price and sale time for the product based on the product features which helps as a baseline for buyer and seller to negotiate on. With the exponential growth of big data and machine learning, everyone in the market wants to leverage the potential of it and built a model which predicts the price accurately.

Through this project, I successfully built a machine learning model using ANN model with Autoencoder based algorithm. This model performs unsupervised learning on the testing data and automatically predicts the price on a given product. I have used the modern data pre-processing techniques using Python the cleanse and standardize the data. The feature extraction approach has helped me to indulge unstructured features into the model. The major takeaway from this project is the valuable usage of the boosting algorithm Autoencoder in light with ANN model implementation. It is fast and consumes very less memory making it an efficient machine learning model on big data platform.

For this project, I have used just the textual features and the structured data for the analysis. The future scope for price prediction would be inclined towards using visual features for the prediction. For every product, in majority of the marketplaces, the seller must post at least three pictures of the product. These pictures can be analysed, and the visual parameters can be used in building the model. Right now, it is difficult to perform this analysis at academic level considering the amount of size each image takes. Also, an operational machine learning model requires large amount of data for prediction. But considering the boon of big data in coming future visual parameters along with the textual features can be used for price prediction.

Secondly, generalizability across different datasets and industries will be the secondary area for improvement in this field. While this research has shown outstanding results in predicting prices and sale times, the true value of the model resides in its capacity to adapt and perform consistently across multiple datasets and industries. To accomplish so, the model needs to be carefully evaluated on the datasets from the perspective of being able engineer wide variety feature datasets from sectors such as finance, healthcare, and real estate. Examining its performance in various scenarios will indicate its generalizability, confirming that its predictive potential is not limited to a single dataset or industry.

6.3 Personal reflection

In this modern era, there is an ongoing pursuit of innovation, machine learning and artificial intelligence have become formidable instruments that are driving broad sectors of businesses. Among these, predicting the price and sale time of e-commerce products presents an exciting problem. As a newcomer in the field of machine learning, I saw myself diving headfirst into the world of Artificial Neural Networks (ANNs), and the goal of predicting e-commerce product price and selling time became a wonderful adventure in the realm of machine learning. As a beginner, I embarked my journey in the field of machine learning from this project, a journey that promises to be both tough yet highly gratifying, delivering a valuable lesson in learning something new, which I wish to take ahead in my professional career.

At the start of my research started with the finding the suitable dataset from Marketplace that will suit the requirements of my study, here I encountered the intricacies of data pre-processing. While it was invigorating to forego precise data sets, I remained steadfast in my pursuit of accurate datasets that could offer valuable insights into price and sale time forecasting. My aim was to locate real-world datasets that could offer an accurate perspective on price and sale time forecasting. After rigorous searching, I discovered a comprehensive dataset on Kaggle marketplace (Mercari, 2019), with a diverse array of datasets conducive to the analysis and the development of practical machine learning models. This dataset has plethora of complex textual and numeric features, presenting an array of data pre-processing challenges from null values to complex unformatted textual description. After thoroughly examining the dataset, I found it crucial to pre-process the data and bring it to the granular form to achieve a higher accuracy. While in code there are normal pre-processing checks, checking null value, formatting the string dataset, removing outliers, and dealing with the descriptive features of the dataset. As I navigated these obstacles, I gained a deeper appreciation for the importance of data quality - a fundamental principle that underpins the entire machine-learning process.

Predicting the price and sale time estimation are the core objectives of my ANN model. Predicting the price of e-commerce product demanded a deep understanding of data pre-processing and feature engineering. Sale time estimation, on the other hand, involved temporal analysis of the two featured specifically to calculate the sell time for the product. This was accomplished with the ANN model with autoencoder feature reduction technique. The heart of this thesis revolved around constructing an Artificial Neural Network. From defining this architecture to implementing the layers for the model was indeed akin to solving a complex puzzle. It builds willingness to adapt to unforeseen circumstances and growing through experiments.

Building the autoencoder an neural network within neural network, held key intricate patterns and features from the dataset. Building this architecture was like crafting the blueprint of the high accuracy model. Through this feature I was able to achieve high performance and the outcomes of the project have been successful.

Throughout this journey, the pursuit of excellence was a guiding principle. For me excellence was only not about perfection but also involved continuous execution of planned work and improvement to create better version of model. I followed my project plan and strictly adhered to it. However, it took longer than I expected for design and implementation of experiments and evaluation. I encountered issues running the experiments, these are described in section 9.4. Investigating and rectifying these issues took a month but it highly contributed towards my knowledge and my model grew stronger.

6.4 Conclusion

As e-commerce has become an emerging industry, predicting the price and sale time will have an important place in the present as well as in the future. Complementing this area of study with machine learning to forecast price and sale time from marketplaces will drive success in the field and open a larger scope for researchers in this area.

This research explores a suitable e-commerce site dataset and achieves a prediction model. The model has emphasized the importance of meticulous data pre-processing, which is a learning curve for the study, showing how the model's precision is affected by properly overseeing it. Including the process to eliminate the outliers, illustrating corelation matrix and adherence to it indeed had an impact on regularizing the feature set.

Furthermore, the model recognizes the need for a balanced feature selection strategy, and the impact of assembling approaches used in machine learning has added to the research’s depth. As a result, the novel goal of the project of predicting price and sale time from Marketplace data is satisfied by combining ANN and an autoencoder.

The success of the experiments for prediction of sale and price reflected the collaboration of the ANN and autoencoder architecture within the model. This was supported by having a well-structured project plan exhibiting the constructive efforts between careful planning, method selection, and implementation.

The pathway to achieve goal exhibited range of challenges from the system crash to large variation in the prediction values. To overcome this, significant research was conducted to look over past studies and learn core concepts and techniques to attain a strong base for a prediction model. It took almost 3 weeks to reach this stage and develop a model that will yield a minor loss for prediction while also dealing with a larger set of features.

Finally, the results and insights from the study offer significance scope for predictive modelling in a variety of industries exploring price and sale time prediction with the combination of the ANN and autoencoder algorithm. This study also offers recommendations for potential improvements that can open a broader scope for the price and sale time prediction model in the future. And capacity of the ANN model with autoencoder to explore the predictive model for varied industries.

List of abbreviations

AI	. . . .	Artificial Intelligence
ML	. . . .	Machine Learning
ANN	. . . .	Artificial Neural Network
SMBs	. . . .	Small and Medium-sized Businesses
SGD	. . . .	Stochastic Gradient Descent
CSV	. . . .	Comma-separated values
ReLU	. . . .	Rectifies Linear Unit
RMSprop	. . . .	Root Mean Squared Propagation
EC2	. . . .	Elastic Cloud 2
GPU	. . . .	Graphical Processing Unit
EDA	. . . .	Exploratory data analysis
LightGBM	. . . .	Light Gradient Boosting Machine
B2C	. . . .	Business to Consumer
C2C	. . . .	Consumer to Consumer
MSLE	. . . .	Mean Square Logarithmic Error
AWS	. . . .	Amazon Web Services
AMI	. . . .	Amazon Machine Image
MSE	. . . .	Mean Squared Error
ADAM	. . . .	Adaptive Moment Estimation

References

Armstrong, J.S. and Brodie, R.J. (1999) *Forecasting for Marketing*. International Thompson Business Press. Available at: http://ssrn.com/abstract=662622.

'Artificial neural network models for forecasting and decision' (no date).

B. Chen (2020) *Early Stopping in Practice: an example with Keras and TensorFlow 2.0*, *Towards Data Science- https://towardsdatascience.com/a-practical-introduction-to-early-stopping-in-machine-learning-550ac88bc8fd*.

Baeldung (2023) *Autoencoders Explained*, *https://www.baeldung.com/cs/autoencoders-explained*.

Bergstra, J. *et al.* (2011) 'Algorithms for Hyper-Parameter Optimization', in J. Shawe-Taylor et al. (eds) *Advances in Neural Information Processing Systems*. Curran Associates, Inc. Available at: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

Bergstra, J., Yamins, D. and Cox, D. (2013) 'Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures', in S. Dasgupta and D. McAllester (eds) *Proceedings of the 30th International Conference on Machine Learning*. Atlanta, Georgia, USA: PMLR (Proceedings of Machine Learning Research), pp. 115–123. Available at: https://proceedings.mlr.press/v28/bergstra13.html.

BigCommerce Pvt. Ltd (2023) *Ecommerce: The History and Future of Online Shopping*, *https://www.bigcommerce.co.uk/articles/ecommerce/*.

Chada, A.R. (2019) *Strategic Pricing of Used Products for e-Commerce Sites*. Available at: https://towardsdatascience.com/machine-.

Chervinskii (2015) *Autoencoder structure.png*, *Cross-wiki upload from en.wikipedia.org*.

Dishashree26 Gupta (2020) *https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/#:~:text=The%20ReLU%20function%20is%20another,neurons%20at%20the%20same%20time.*, *https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/#:~:text=The%20ReLU%20function%20is%20another,neurons%20at%20the%20same%20time.*

Ekin Tiu (2020) *Understanding Latent Space in Machine Learning*, *towardsdatascience.com*.

Ensafi, Y. *et al.* (2022) 'Time-series forecasting of seasonal items sales using machine learning – A comparative analysis', *International Journal of Information Management Data Insights*, 2(1). Available at: https://doi.org/10.1016/j.jjimei.2022.100058.

Gaspar, A. *et al.* (2021) 'Hyperparameter Optimization in a Convolutional Neural Network Using Metaheuristic Algorithms', in, pp. 37–59. Available at: https://doi.org/10.1007/978-3-030-70542-8_2.

Jadon, A., Patil, A. and Jadon, S. (2022) 'A Comprehensive Survey of Regression Based Loss Functions for Time Series Forecasting'. Available at: http://arxiv.org/abs/2211.02989.

Jason Brownlee (2019a) *How to Control the Stability of Training Neural Networks With the Batch Size*, *Deep Learning Performance*.

Jason Brownlee (2019b) *How to Fix the Vanishing Gradients Problem Using the ReLU*, *https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/*.

Jeremy Zhang (2021) *Optimisation Algorithm — Adaptive Moment Estimation(Adam)*.

Keras.io (2023) *EarlyStopping*, *https://keras.io/api/callbacks/early_stopping/*.

Kingma, D.P. and Ba, J. (2017) 'Adam: A Method for Stochastic Optimization'.

Li, M., Ji, S. and Liu, G. (2018) 'Forecasting of Chinese E-Commerce Sales: An Empirical Comparison of ARIMA, Nonlinear Autoregressive Neural Network, and a Combined ARIMA-NARNN Model', *Mathematical Problems in Engineering*. Hindawi Limited. Available at: https://doi.org/10.1155/2018/6924960.

Lin Zhao, Ph.D., P.U.C.U. (2009) 'Neural Networks In Business TimeSeries Forecasting: Benefits And Problems', *Review of Business Information Systems –Third Quarter 2009*, 13(3), pp. 57–62.

Livingstone, D.J., Manallack, D.T. and Tetko, I. V (1997) *Data modelling with neural networks: Advantages and limitations**, *Journal of Computer-Aided Molecular Design*.

Mercari (2020) *Mercari*, *https://www.mercari.com/*.

Mercari (no date) *mercari price prediction train and test data*, *Kaggle*.

Mohamed, M.A., El-Henawy, I.M. and Salah, A. (2022) 'Price prediction of seasonal items using machine learning and statistical methods', *Computers, Materials and Continua*, 70(2), pp. 3473–3489. Available at: https://doi.org/10.32604/cmc.2022.020782.

Oates, B.J. (2006) *Researching information systems and computing*. London: SAGE.

Polamuri, S.R., Srinivas, K. and Krishna Mohan, A. (2019) 'Stock market prices prediction using random forest and extra tree regression', *International Journal of Recent Technology and Engineering*, 8(3), pp. 1224–1228. Available at: https://doi.org/10.35940/ijrte.C4314.098319.

'Prediction of Product Prices and estimate sale time' (no date).

Puri, M. *et al.* (2016) 'Introduction to Artificial Neural Network (ANN) as a Predictive Tool for Drug Design, Discovery, Delivery, and Disposition: Basic Concepts and Modeling. Basic Concepts and Modeling', in *Artificial Neural Network for Drug Design, Delivery and Disposition*. Elsevier Inc., pp. 3–13. Available at: https://doi.org/10.1016/B978-0-12-801559-9.00001-6.

Saturn Cloud (2023) *How to Use a Linear Activation Function in TensorFlow*, *https://saturncloud.io/blog/how-to-use-a-linear-activation-function-in-tensorflow/*.

Sreenath S (2020) *Hyperparameter Tuning using Optuna*, *https://www.analyticsvidhya.com/blog/2020/11/hyperparameter-tuning-using-optuna/*.

Tang, Z., de Almeida, C. and Fishwick, P.A. (1991) ‘Time series forecasting using neural networks vs. Box- Jenkins methodology’, *Simulation*, 57, pp. 303–310. Available at: https://api.semanticscholar.org/CorpusID:46942327.

TensorFlow (no date) *tf.keras.losses.MeanSquaredLogarithmicError*, *https://www.tensorflow.org/api_docs/python/tf/keras/losses/MeanSquaredLogarithmicError*.

The Python Code (2023) *Autoencoders for Dimensionality Reduction using TensorFlow in Python*, *https://thepythoncode.com/article/feature-extraction-dimensionality-reduction-autoencoders-python-keras*.

Vijh, M. *et al.* (2020) ‘Stock Closing Price Prediction using Machine Learning Techniques’, in *Procedia Computer Science*. Elsevier B.V., pp. 599–606. Available at: https://doi.org/10.1016/j.procs.2020.03.326.

Wang, M. xian *et al.* (2022) ‘Stock price prediction for new energy vehicle enterprises: An integrated method based on time series and cloud models’, *Expert Systems with Applications*, 208. Available at: https://doi.org/10.1016/j.eswa.2022.118125.

Wang, Y., Yao, H. and Zhao, S. (2016) ‘Auto-encoder based dimensionality reduction’, *Neurocomputing*, 184, pp. 232–242. Available at: https://doi.org/10.1016/j.neucom.2015.08.104.

Wikipedia contributors (2023) ‘Mean squared error — Wikipedia, The Free Encyclopedia’. Available at: https://en.wikipedia.org/w/index.php?title=Mean_squared_error&oldid=1170510533.

Yoo, Y. (2019) ‘Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches ☆’, 178, pp. 74–83. Available at: https://doi.org/10.1016/j.knosys.

Yu, N., Yu, Z. and Pan, Y. (2017) ‘A deep learning method for lincRNA detection using auto-encoder algorithm’, *BMC bioinformatics*, 18, p. 511. Available at: https://doi.org/10.1186/s12859-017-1922-3.

Zheng, X., Cai, J. and Zhang, G. (2022) ‘Stock Trend Prediction Based on ARIMA-LightGBM Hybrid Model’, in *2022 3rd Information Communication Technologies Conference, ICTC 2022*. Institute of Electrical and Electronics Engineers Inc., pp. 227–231. Available at: https://doi.org/10.1109/ICTC55111.2022.9778304.

Appendix A – Project Proposal

1. Introduction

In the contemporary technological era, the sector of e-commerce has rapidly expanded, and internet sales have developed into a sizable source of revenue for many people and businesses. However, in such a competitive industry, it is crucial for sellers to comprehend how swiftly they can supply their items. They may make better decisions on pricing, marketing, and inventory management with this information, leading to more earnings in the end. Also, for the product critical with the expiration date this information can help speed up the sale of the product.

With technological development in Machine Learning using potent techniques, one can now predict how soon a product will sell. Machine learning algorithms can identify patterns and trends in previous data that can be used to predict future sales, approximate price of the product for the best sale deals. These forecasts might consider a range of variables, including consumer and market trends as well as product characteristics.

Machine learning techniques have been widely employed for predicting sales in numerous fields, including marketing, e-commerce, and retail. Machine learning has been applied in the retail industry, for instance, to forecast sales, improve pricing plans, and determine product demand. Machine learning has been applied to e-commerce to enhance search results and personalise product suggestions, highly in demand in the marketing sector to segment consumers and focus on advertisements.

Numerous studies have specifically looked at forecasting how quickly sales will grow in the e-commerce sector. For instance, Wang et al. (2018) (Ensafi *et al.*, 2022a) employed deep learning to forecast the likelihood that a product will be sold within a specific time limit (Hosseinnia Shavaki and Ebrahimi Ghahnavieh, 2022). A hybrid method was used in different research by Raza et al. (2020), which combined machine learning with traditional time-series analysis to predict sales velocity. Artificial neural network (ANN) models have been popular for sales forecasting especially with ARIMA to improve accuracy of the prediction(Ensafi *et al.*, 2022a). Results have confirmed the efficacy of the model. But unlike all these previous studies the focus of this thesis is to resolve the problem of sale time prediction and price estimation together in a more efficient way saving time and serving even in larger non-linear dataset from Marketplace.

The rest of this paper is organised as follows Chapter1 – defining Aim and Objective of the research Chapter 2 elaborates the above discussed Literature Review for related papers, Chapter 3 explaining methods and tools used for design, analysis and evaluation, Chapter 4. Work Plan showing processes, dependencies, and milestones and Chapter 5 discusses risks and ways to mitigate it.

1.1. Aim and Objectives

Aim of the project is to create a model to predict Product Prices and estimated Sale time on marketplace data based on the AI (Artificial Intelligence) approach. This AI architecture will use historical purchasing data for various products available on the Marketplace.

The primary objective of this study is to explore in a systematic way the below:

- Research, review, and critique existing approaches towards estimating techniques for price and sale time forecasting.
- Analyse different techniques for Feature Extraction using AI approach and do a prediction based on reduced features set.
- To build a working prototype predicting sale time and price of the product based on the extracted (reduced feature set) data.
- To evaluate the performance of the model.

To accomplish the above objectives, the next step involves research and overviewing the existing approaches into similar areas. Even though there are other reviews on Feature extraction and using ordinary neural networks or machine learning, some areas remain unexploited. There are limited reviews especially focused on sale time and price prediction model together.

2. Critical context

The essential part of the product price and sale time prediction deals with the accuracy of the forecasting. This chapter reviews some of the similar papers and approaches discussing the related work of price prediction using machine learning and deep learning methods.

A support system for predicting eBay end prices, one of the previous studies of final sale price predictions have used a substantial number of features for classification focusing on finding factors determining the auction end price, with no information on the sale time of the product. (Van Heijst, Potharst and van Wezel, 2008). It uses the statistical and Machine Learning model for forecasting. The primary strength of this predicting model is it uses the textual information contained in the item description. It is comparatively easy to evaluate and master this data because they have all information contained in the single field. But this limits the scope of feature availability, sometimes constraining it to the single field for study. Also, it focuses more on the data mining method which updates prices of on-going auction based on newly arrived information neglecting past trends making it less suitable for using it in the extended research theme for sale time and price prediction. Clearly calculating this risk, this experiment i.e., The prediction model states to be capable of predicting 20-40% of the sale price within the 5% range of the actual selling price. The limitation with the above study shows that the statistical and ML (Machine Learning) model use only the structured data while there is possibility that there can be unstructured data in the getting feed into the embedded vector.

The most Classic Time-Series approach used for time series models introduces deterministic time series models. This example predicts the stock price for a new energy vehicle enterprise. The model applies the part of time series data incorporated into the cloud model with some applied variance. This variance here describes the scope for fluctuation of time series data(Wang *et al.*, 2022). In today's era, there exists diverse data with various complexity which might be difficult to model in this approach. There is a great challenge to handle a large amount of data and needs manual intervention to eliminate irrelevant data to improve the efficiency of subsequent data processing and analysis and ensure the precision of results. On the other hand, applying this technique for forecasting for two attributes that is Price and sale time of the product will turn out to be more time consuming and error prone.

Another famous technique for prediction widely used is Recurrent Neural Network. This is a class of neural network that is used to model the data in sequence of time series(Tensorflow, 2022). This model enables storing events from the past and building knowledge based on their memory. Therefore, becoming more prominent in time forecasting and estimating. But despite these benefits it takes long training times, also a fixed number of hidden neurons must be selected in advance, and in practical applications the context length that is considered is small(Ensafi *et al.*, 2022b) making it less favourable for larger sets of non-linear data.

In this paper the main goal is to analyse a larger set of ecommerce data for forecasting both price and sale time of the product. It is identified that there is a need for new orientation of data manipulation overcoming the above deficiencies and keeping it simpler unlike RNN (Recurrent Neural Networks). The best approach to suit these needs to learn through unsupervised learning and efficiently represent data is modelling the system by applying an auto-encoder algorithm (Wu, Wang, and Wu, 2022). Deep Neural Architecture with application of Machine learning using autoencoder and ordinary approximator making it a more flexible approach for large amounts of nonlinear data, with most accurate results without any need for manual pre-processing saving time. This model will significantly reduce the computational resources and enhance forecasting accuracy predicting two features of any ecommerce product namely Price forecasting and sale time of the product.

3. Approaches

Methods and tools are critical components of the design, analysis, and evaluation process. After carefully understanding strength and weakness of various methods and tools in contrast with the stated aim, the comprehensive approach has been identified broken down into following:

i. **Collection of the Dataset from Marketplace:** To train and test the proposed system, data has been collected from 25 best retail, sales, and e-commerce datasets for Machine Learning(iMerit, 2021). that is between 2021 and 2023 where each days' data includes variables such as categories, prices, description, brand, is Sale and offer. The system heavily depends on the features like sale price and sale time data from these resources. The input must be formalised to include details from past few years and should help sufficiently evaluate results as expected. currently, there is 38432 records of data collected from these resources from all types of the product which are in demand currently and help predict the model suitable for present market trends. Out of 38432 records samples 80% will be given as training data, 10% Validation and the remaining 10% were used as test data.

   To automate the collection of the data, we will use the API provided by the Marketplace like eBay, Amazon, Zalando to access the data through authentication, collecting features set like sale price and sale time data, product features, seller information

ii. **Feature reduction**- The data collected from these Marketplace API will be reduced into a set of key features required for approximating the client/users' purchasing patterns. To reduce the given set of features which can be larger data sets, the system will implement the autoencoder algorithm. This Feature reduction algorithm is an automated algorithm which gives benefit over other types of neural network, keeping it simple and easy to develop.

iii. **Autoencoder** is a kind of signal compression model based on a neural network. It is a non-supervised learning network (Lange and Riedmiller, 2010), that maps the input data to a lower-dimensional representation, The autoencoder is broadly divided into two sections namely encoder and decoder. In the encoder section the input feature is converted relating it to the code i.e., less dimensional than the input layer. Further, in the decoding layer the code i.e., also referred to as hidden layer code is decoded to the different mapping output (Kingma and Ba, 2015a). It presents strong nonlinear characteristics as a result, deep autoencoder is a promising approach comparatively. The main feature offered by this neural network learning pattern is that the quality of input matches the quality of output samples.

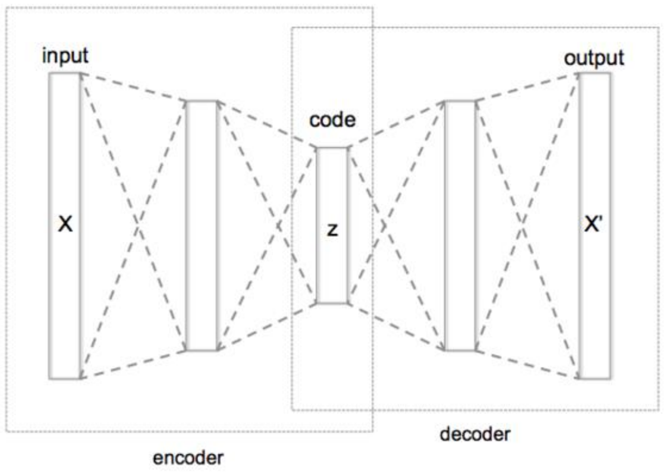


Fig.3.1 Simplified version of autoencoder (https://en.wikipedia.org/, 2023)

The above diagram shows the graphical representation of the Simplified network structure of deep autoencoder, receiving several input features i.e., x1, x2, x3.. xn, transforming data into the hidden layer represented as z1, z2.. . Output feature set represented as x^1, x^2, x^3… This is the same as an ordinary neural network as it checks the error level between input and output and corrects the error if they are not the same. With every cycle of training on this model, it will try to create output with highest levels of accuracy, as depicted in the diagram below as viewed from the autoencoder.

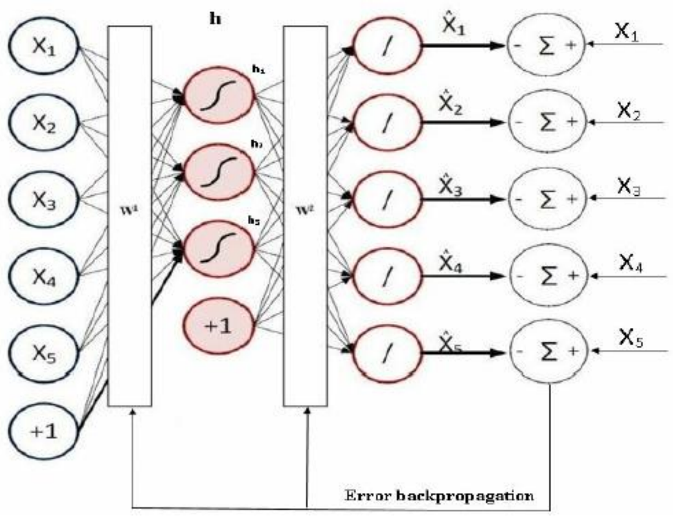


Fig.3.2 View of an autoencoder with error backpropagation(Kingma and Ba, 2015b)

After minimising the encoding/decoding error, the hidden layer will present the output features accurately, thus evaluating the performance of the autoencoder on the testing set. These data in the hidden layer will represent the important sample set that will be used for decision making further in the model. (Kingma and Ba, 2015b)

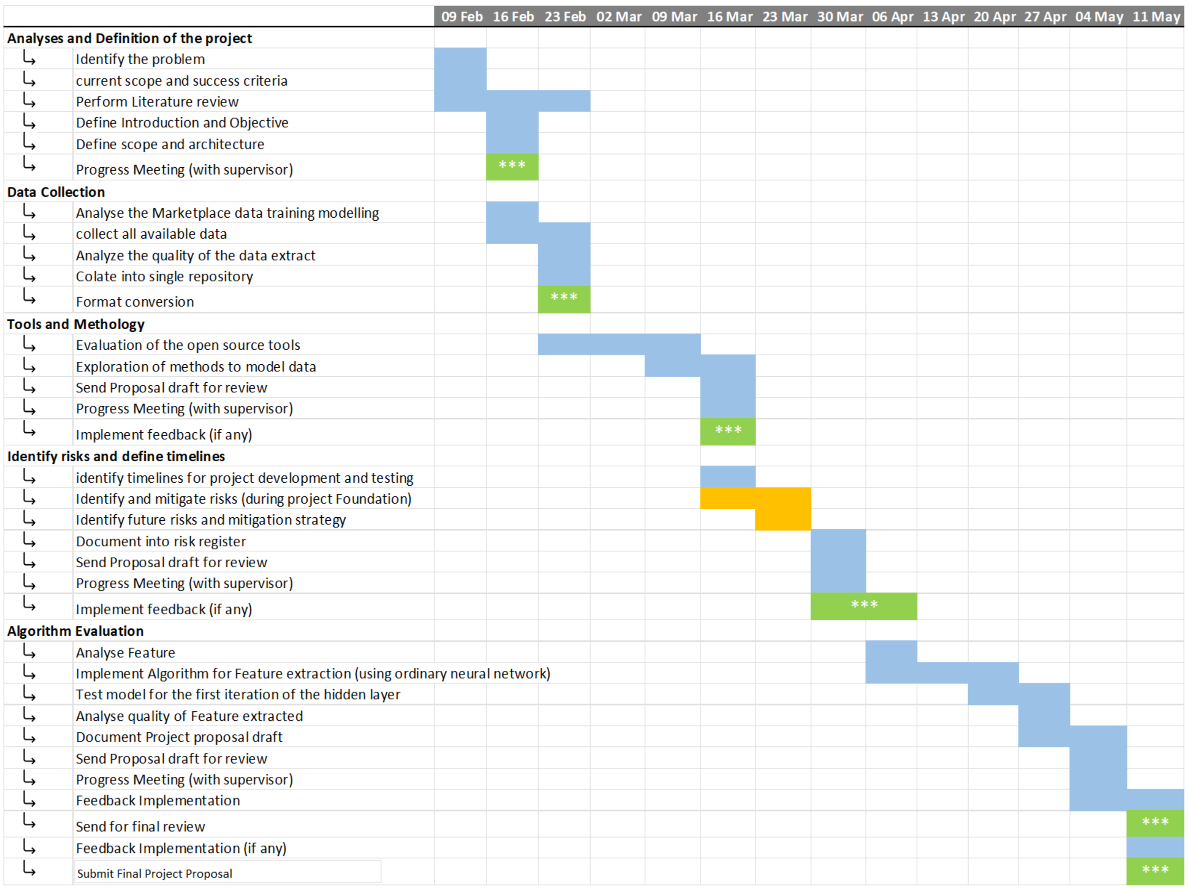
In the next step the reduced feature set is used as an input for the neural network to do a prediction/to learn through the available reduced feature set from the hidden layer of the autoencoder comes prediction of the price and the sale time in the following way.

iv. **Ordinary approximator**- is the ordinary neural network that takes the input parameters and gives the parameter value sale time prediction as the function value of inputs.

Artificial Neural Networks (ANNs) architecture: It is the general architecture to classify and perform regression tests on patterns of non-linear data which can define complex nonlinear decision boundaries. The primary purpose of the ANNs here is to learn mapping between the given feature set typically accomplished through a training phase. During this training phase the network will learn internal parameters of the unit including weight and biases to produce the output layer as the forecasted value of the price and estimate sale time of the product.(Hosseinnia Shavaki and Ebrahimi Ghahnavieh, 2022),

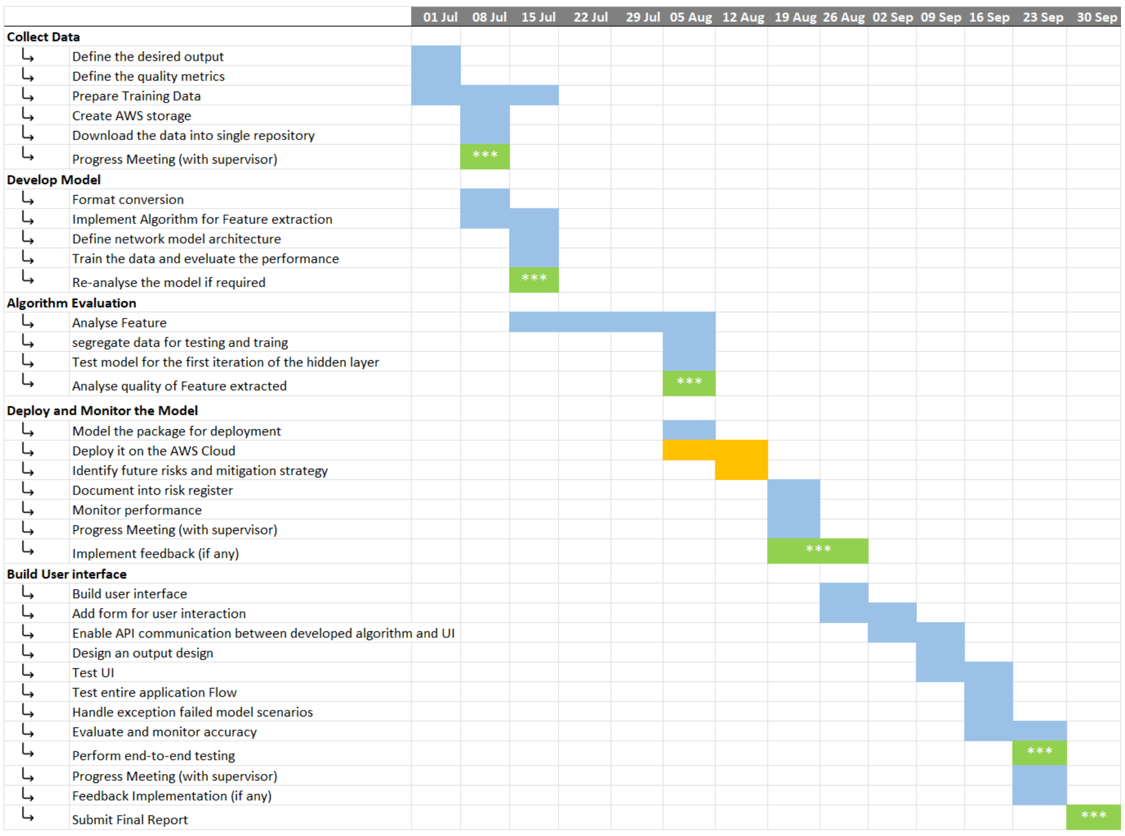
To summarise, the model will use an autoencoder with ANNs for time series forecasting which starts with pre-processing the data, training with the autoencoder algorithm to extract meaningful features set, and then use this encoded data value as input to train the ANN for forecasting. Further to improve the performance of the model we will evaluate the error propagation between input and output of the hidden layer and to achieve efficient results

4. Work plan



		09 Feb	16 Feb	23 Feb	02 Mar	09 Mar	16 Mar	23 Mar	30 Mar	06 Apr	13 Apr	20 Apr	27 Apr	04 May	11 May
Analyses and Definition of the project															
↳	Identify the problem	🟦													
↳	current scope and success criteria	🟦													
↳	Perform Literature review	🟦	🟦	🟦											
↳	Define Introduction and Objective		🟦												
↳	Define scope and architecture		🟦												
↳	Progress Meeting (with supervisor)		🟩 ***												
Data Collection															
↳	Analyse the Marketplace data training modelling		🟦												
↳	collect all available data		🟦	🟦											
↳	Analyze the quality of the data extract			🟦											
↳	Colate into single repository			🟦											
↳	Format conversion			🟩 ***											
Tools and Methology															
↳	Evaluation of the open source tools			🟦	🟦	🟦									
↳	Exploration of methods to model data					🟦	🟦								
↳	Send Proposal draft for review						🟦								
↳	Progress Meeting (with supervisor)						🟦								
↳	Implement feedback (if any)						🟩 ***								
Identify risks and define timelines															
↳	identify timelines for project development and testing						🟦								
↳	Identify and mitigate risks (during project Foundation)						🟧	🟧							
↳	Identify future risks and mitigation strategy							🟧							
↳	Document into risk register								🟦						
↳	Send Proposal draft for review								🟦						
↳	Progress Meeting (with supervisor)								🟦						
↳	Implement feedback (if any)								🟩 ***	🟩					
Algorithm Evaluation															
↳	Analyse Feature									🟦					
↳	Implement Algorithm for Feature extraction (using ordinary neural network)									🟦	🟦	🟦			
↳	Test model for the first iteration of the hidden layer											🟦	🟦		
↳	Analyse quality of Feature extracted												🟦		
↳	Document Project proposal draft												🟦	🟦	
↳	Send Proposal draft for review													🟦	
↳	Progress Meeting (with supervisor)													🟦	
↳	Feedback Implementation													🟦	🟦
↳	Send for final review														🟩 ***
↳	Feedback Implementation (if any)														🟦
↳	Submit Final Project Proposal														🟩 ***

The above Gantt Chart maps the Project Foundation plan- presenting dependencies of each step, further broken down into achievable milestones (represented in green with Start symbol). In amber colour is the phase where the risks were identified and mitigated for smooth delivery of the project proposal.



The above Gantt Chart presents the Development Plan- overall plan for achieving project goals and objectives within the given time constraints. It also identifies potential challenges, monitoring plans before achieving each milestone that plans for contingencies of any risk, and documents and reports on progress and outcomes.

5. Risks

The Table below shows the risk register depicting major risk encountered with its likelihood, impact, and mitigation steps:

Risk	Likelihood (1-3)	Consequence (1-5)	Impact (L x C)	Mitigation
Unable to produce deliverables or achieve key milestones.	1	5	5	Divide project into achievable milestone, meet supervisor to discuss and scrutinise results achieved

Loss of data due to system crash	1	3	3	Data will be backed up daily on OneDrive to avoid any loss due unexpected system failure
Speed of the training model can be slow for a long set of data.	3	3	9	Use only a ratio of random sample data to train the model. Use NN networks with GPUs.
Cloud storage and computation for modelling data might incur higher cost	2	4	8	Use the free tier feature of Cloud provider for the development of the project valid for Around 90 days (about 3 months)
Unexpected technical issues with hardware or software	1	4	4	Maintain a backup of system, mostly in cloud Storage
Unable to meet supervisor for discussion	1	3	3	Arrange Teams meeting in advance, for discussion on each milestone achieved, to get prior agreement for the meeting

Risk	Consequence Score
Extremely low	1
Low	2

Key:

Risk	Likelihood Score
Low	1
Medium	2
High	3

Medium	3
High	4
Extremely high	5

6. References

Ensafi, Y. *et al.* (2022a) ‘Time-series forecasting of seasonal items sales using machine learning – A comparative analysis’, *International Journal of Information Management Data Insights*, 2(1). Available at: https://doi.org/10.1016/j.jjimei.2022.100058.

Ensafi, Y. *et al.* (2022b) ‘Time-series forecasting of seasonal items sales using machine learning – A comparative analysis’, *International Journal of Information Management Data Insights*, 2(1). Available at: https://doi.org/10.1016/j.jjimei.2022.100058.

van Heijst, D., Potharst, R. and van Wezel, M. (2008) ‘A support system for predicting eBay end prices’, *Decision Support Systems*, 44(4), pp. 970–982. Available at: https://doi.org/10.1016/j.dss.2007.11.004.

Hosseinnia Shavaki, F. and Ebrahimi Ghahnavieh, A. (2022) ‘Applications of deep learning into supply chain management: a systematic literature review and a framework for future research’, *Artificial Intelligence Review* [Preprint]. Available at: https://doi.org/10.1007/s10462-022-10289-z.

https://en.wikipedia.org/ (2023) *Schematic structure of an autoencoder with 3 fully connected hidden layers. The code (z, or h for reference in the text) is the most internal layer., wikipedia.org*.

iMerit (2021) *25 Best Retail, Sales, and Ecommerce Datasets for Machine Learning*, *iMerit.net*.

Kingma, D.P. and Ba, J.L. (2015a) ‘Adam: A method for stochastic optimization’, in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.

Kingma, D.P. and Ba, J.L. (2015b) ‘Adam: A method for stochastic optimization’, in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.

Lange, S. and Riedmiller, M. (2010) ‘Deep auto-encoder neural networks in reinforcement learning’, in *the 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. Available at: https://doi.org/10.1109/IJCNN.2010.5596468.

Tensorflow (2022) *Recurrent Neural Networks (RNN) with Keras*, *https://www.tensorflow.org/guide/keras/rnn*.

Wang, M. xian *et al.* (2022) ‘Stock price prediction for new energy vehicle enterprises: An integrated method based on time series and cloud models’, *Expert Systems with Applications*, 208. Available at: https://doi.org/10.1016/j.eswa.2022.118125.

Wu, D., Wang, X. and Wu, S. (2022) ‘A hybrid framework based on extreme learning machine, discrete wavelet transform, and autoencoder with feature penalty for stock prediction’, *Expert Systems with Applications*, 207. Available at: https://doi.org/10.1016/j.eswa.2022.118006.

7. Ethics review form

Research Ethics Review Form: BSc, MSc, and MA Projects

Computer Science Research Ethics Committee (CSREC)-

http://www.city.ac.uk/department-computer-science/research-ethics

Undergraduate and postgraduate students undertaking their final project in the Department of Computer Science are required to consider the ethics of their project work and to ensure that it complies with research ethics guidelines. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people (“participants”) in the project.

To ensure that appropriate consideration is given to ethical issues, all students must complete this form and attach it to their project proposal document. There are two parts:

PART A: Ethics Checklist. All students must complete this part.

The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

PART B: **Ethics Proportionate Review Form**. Students who have answered "no" to all questions in A1, A2 and A3 and "yes" to question 4 in A4 in the ethics checklist must complete this part. The project supervisor has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk. The approval may be **provisional** – identifying the planned research as likely to involve MINIMAL RISK. In such cases you must additionally seek **full approval** from the supervisor as the project progresses and details

A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
1	Does your research require approval from the National Research Ethics Service (NRES)? e.g., because you are recruiting current NHS patients or staff? If you are unsure try - **https://www.hra.nhs.uk/approvals-amendments/what-approvals-do-i-need/**	NO
2	Will you recruit participants who fall under the auspices of the Mental Capacity Act? Such research needs to be approved by an external ethics committee such as NRES or the Social Care Research Ethics Committee - **http://www.scie.org.uk/research/ethics-committee/**	NO
3	Will you recruit any participants who are currently under the auspices of the Criminal Justice System, for example, but not limited to, people on remand, prisoners, and those on probation? Such research needs to be authorised by the ethics approval system of	NO

A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
	the National Offender Management Service.	

A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
1	Does your research involve participants who are unable to give informed consent? For example, but not limited to, people who may have a degree of learning disability or mental health problem, which means they are unable to make an informed decision on their own behalf.	NO
2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	NO
3	Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)?	NO
4	Does your project involve participants disclosing information about special category or sensitive subjects? For example, but not limited to racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and proceedings	NO

A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study? Please check the latest guidance from the FCO - **http://www.fco.gov.uk/en/**	NO
6	Does your research involve invasive or intrusive procedures? These may include, but are not limited to, electrical stimulation, heat, cold or bruising.	NO
7	Does your research involve animals?	NO
8	Does your research involve the administration of drugs, placebos, or other substances to study participants?	NO

A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/ **Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.**		*Delete as appropriate*
1	Does your research involve participants who are under the age of 18?	NO
2	Does your research involve adults who are vulnerable because of their	NO

A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/ **Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.**		*Delete as appropriate*
	social, psychological, or medical circumstances (vulnerable adults)? *This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.*	
3	Are participants recruited because they are staff or students of City, University of London? *For example, students studying on a particular course or module.* *If yes, then approval is also required from the Head of Department or Programme Director.*	NO
4	Does your research involve intentional deception of participants?	NO
5	Does your research involve participants taking part without their informed consent?	NO
6	Is the risk posed to participants greater than that in normal working life?	NO
7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	NO

A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK. **If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.** **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.**		*Delete as appropriate*
4	Does your project involve human participants or their identifiable personal data? For example, as interviewees, respondents to a survey or participants in testing	NO

PART B: Ethics Proportionate Review Form

If you answered YES to question 4 and NO to all other questions in sections A1, A2 and A3 in PART A of this form, then you may use PART B of this form to apply for a proportionate ethics review of your project. Your project supervisor has delegated authority to review and approve this application under proportionate review. You must receive final approval from your supervisor in writing before beginning the planned research.

However, if you cannot provide all the required attachments (see B.3) with your project proposal (e.g., because you have not yet written the consent forms, interview schedules etc), the approval from your supervisor will be **provisional**. You **must** submit the missing items to your supervisor for approval prior to commencing these parts of your project. Once again, you must receive written confirmation from your supervisor that any provisional approval has been superseded by with full approval of the planned activity as detailed in the full documents. **Failure to follow this procedure and demonstrate that final approval has been achieved may result in you failing the project module.**

Your supervisor may ask you to submit a full ethics application through Research Ethics Online, for instance if they are unable to approve your application, if the level of risks associated with your project change, or if you need an approval letter from the CSREC for an external organisation.

B.1 The following questions must be answered fully. **All grey instructions must be removed.**		*Delete as appropriate*
1	Will you ensure that participants taking part in your project are fully informed about the purpose of the research?	Not Applicable
2	Will you ensure that participants taking part in your project are fully informed about the procedures affecting them or affecting any information collected about them, including information about how the data will be used, to whom it will be disclosed, and how long it will be kept?	Not Applicable
3	When people agree to participate in your project, will it be made clear to them that they may withdraw (i.e., not participate) at any time without any penalty?	Not Applicable
4	Will consent be obtained from the participants in your project?	Not Applicable
	Consent from participants will be necessary if you plan to involve them in your project or if you plan to use identifiable personal data from existing records. "Identifiable personal data" means data relating to a living person who might be identifiable if the record includes their name, username, student id, DNA, fingerprint, address, etc. If YES, you must attach drafts of the participant information sheet(s) and consent form(s) that you will use in section B.3 or, in the case of an existing dataset, provide details of how consent has been obtained. You must also retain the completed forms for subsequent inspection. Failure to provide the completed consent request forms	Not Applicable

B.1 The following questions must be answered fully. **All grey instructions must be removed.**		*Delete as appropriate*
	will result in withdrawal of any earlier ethical approval of your project.	
5	Have you decided to ensure that material and/or private information obtained from or about the participating individuals will remain confidential?	Not Applicable

B.2 If the answer to the following question (B2) is YES, you must provide details		*Delete as appropriate*
2	Will the research be conducted in the participant's home or other non-University location? If YES, you must provide details of how your safety will be ensured.	No

B.3 Attachments **ALL the following documents MUST be provided to supervisors if applicable.** **All must be considered prior to final approval by supervisors.** **A written record of final approval must be provided and retained.**		***YES***	***NO***	***Not Applicable***
1	Details on how safety will be assured in any non-University location, including risk assessment if required (see B2)			Not Applicable

B.3 Attachments **ALL the following documents MUST be provided to supervisors if applicable.** **All must be considered prior to final approval by supervisors.** **A written record of final approval must be provided and retained.**		***YES***	***NO***	***Not Applicable***
2	Details of arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential (see B1.5) Any personal data must be acquired, stored, and made accessible in ways that are GDPR compliant.			Not Applicable
3	Full protocol for any workshops or interviews**			Not Applicable
4	Participant information sheet(s)**			Not Applicable
5	Consent form(s)**			Not Applicable
6	Questionnaire(s)** sharing a Qualtrics survey with your supervisor is recommended.			Not Applicable
7	Topic guide(s) for interviews and focus groups**			Not Applicable
8	Permission from external organisations or Head of Department** e.g., for recruitment of participants			Not Applicable

Appendix B – Important Links to Additional files

1. **Link to the code for ANN model implementation with output results (.iypnb)**

- **Experiment phase I- Price Predictions**

https://gitlab.city.ac.uk/adcy342/inm363-individual-project-adcy342/-/blob/main/ANN_model_for_price_prediction_with_autoencoder_feature.ipynb

- **Experiment phase II- Sale time Predictions**

https://gitlab.city.ac.uk/adcy342/inm363-individual-project-adcy342/-/blob/main/ANN_model_for_sale_time_prediction_with_autoencoder_feature.ipynb

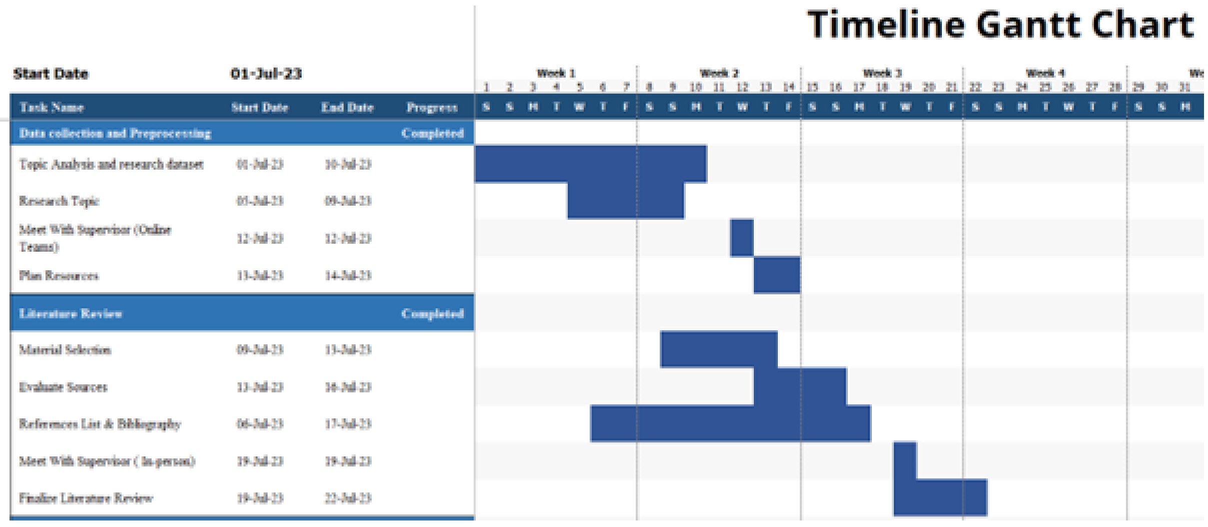
2. **City GitHub Link for train and test dataset for price and sale time prediction (Mercari ecommerce Marketplace data- from Kaggle** (Mercari, no date)**)**

Mercari_test.csv and Mercari_train.csv at -

https://gitlab.city.ac.uk/adcy342/inm363-individual-project-adcy342.git

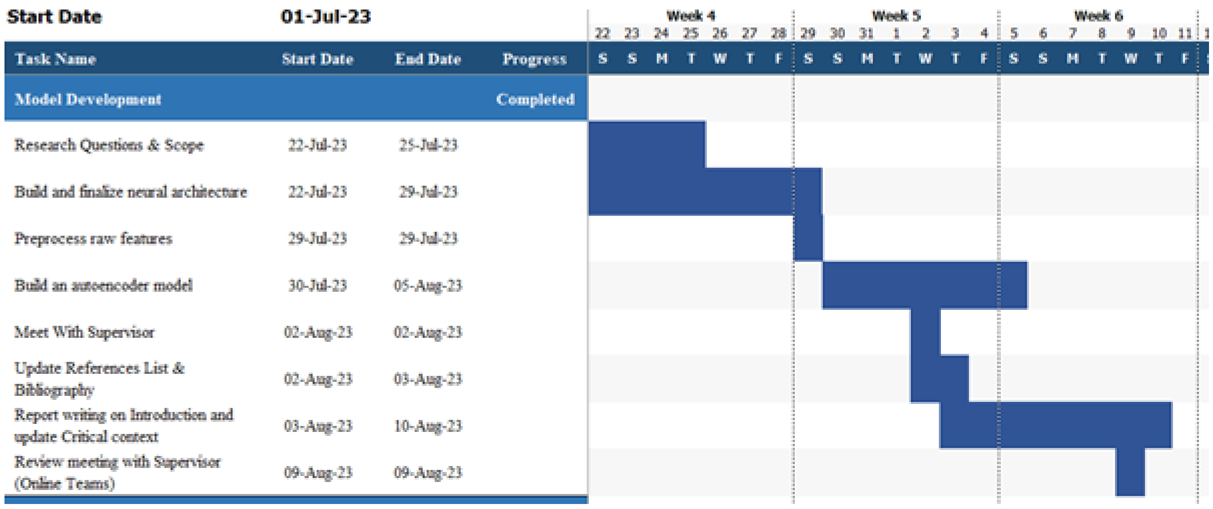
Appendix C –The Work Plan (Gantt Chart)

The detailed representation of the project plan (i.e., Gantt chart) followed while developing the project, including the documentation timeline for the project report is presented below



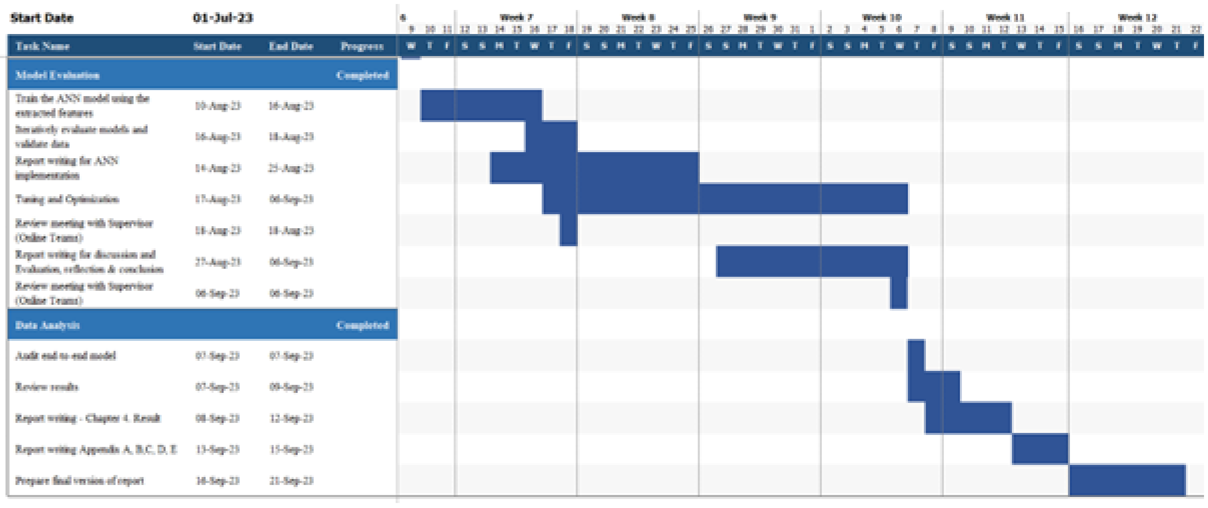
Start Date 01-Jul-23

Task Name	Start Date	End Date	Progress
Data collection and Preprocessing			Completed
Topic Analysis and research dataset	01-Jul-23	10-Jul-23	
Research Topic	05-Jul-23	09-Jul-23	
Meet With Supervisor (Online Teams)	12-Jul-23	12-Jul-23	
Plan Resources	13-Jul-23	14-Jul-23	
Literature Review			Completed
Material Selection	09-Jul-23	13-Jul-23	
Evaluate Sources	13-Jul-23	16-Jul-23	
References List & Bibliography	06-Jul-23	17-Jul-23	
Meet With Supervisor (In-person)	19-Jul-23	19-Jul-23	
Finalize Literature Review	19-Jul-23	22-Jul-23	



Start Date 01-Jul-23

Task Name	Start Date	End Date	Progress
Model Development			Completed
Research Questions & Scope	22-Jul-23	25-Jul-23	
Build and finalize neural architecture	22-Jul-23	29-Jul-23	
Preprocess raw features	29-Jul-23	29-Jul-23	
Build an autoencoder model	30-Jul-23	05-Aug-23	
Meet With Supervisor	02-Aug-23	02-Aug-23	
Update References List & Bibliography	02-Aug-23	03-Aug-23	
Report writing on Introduction and update Critical context	03-Aug-23	10-Aug-23	
Review meeting with Supervisor (Online Teams)	09-Aug-23	09-Aug-23	



Start Date 01-Jul-23

Task Name	Start Date	End Date	Progress
Model Evaluation			Completed
Train the ANN model using the extracted features	10-Aug-23	16-Aug-23	
Iteratively evaluate models and validate data	16-Aug-23	18-Aug-23	
Report writing for ANN implementation	14-Aug-23	25-Aug-23	
Tuning and Optimization	17-Aug-23	06-Sep-23	
Review meeting with Supervisor (Online Teams)	18-Aug-23	18-Aug-23	
Report writing for discussion and Evaluation, reflection & conclusion	27-Aug-23	06-Sep-23	
Review meeting with Supervisor (Online Teams)	06-Sep-23	06-Sep-23	
Data Analysis			Completed
Audit end to end model	07-Sep-23	07-Sep-23	
Review results	07-Sep-23	09-Sep-23	
Report writing - Chapter 4. Result	08-Sep-23	12-Sep-23	
Report writing Appendix A, B,C, D, E	13-Sep-23	15-Sep-23	
Prepare final version of report	16-Sep-23	21-Sep-23	

Appendix D – ANN implementation in Python (.iypnb)

Experiment phase I- Price Predictions

# Step 1: Installation and Setup

Import Section

```
# Importing necessary libraries

import pandas as pd

import numpy as np

import multiprocessing

%matplotlib inline

# Required to initialize NN

from keras.models import Sequential

# Required to build layers of ANN and autoencoder

import tensorflow as tf

from tensorflow import keras

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.layers import Input, Dense

# Required for visualization

import missingno as msno

import matplotlib.cm as cm

import scipy as sci

import seaborn as sns

import matplotlib.pyplot as plt

# Required for displaying learning_curve

from sklearn.model_selection import learning_curve

from datetime import datetime

start_real = datetime.now()
```

```
# Mounts google drive to load Test and Train dataset

from google.colab import drive

drive.mount('/content/drive')


# Retrieving train and test Mercari dataset records for processing

price_train = pd.read_csv("/content/drive/My Drive/Project/Data/ANN/Mercari_train.csv", encoding='utf-
8-sig')

price_test = pd.read_csv("/content/drive/My Drive/Project/Data/ANN/Mercari_test.csv", encoding='utf-
8-sig')


# Displaying the dimensions of the dataset (rows, column)

print(price_train.shape)

print(price_test.shape)


**Note: please comment below code to run the model on entire datasetset**

price_train=price_train[:85000]

price_test=price_test[:25000]


#checking content and format of the Mercari train dataset

price_train.info()

#checking content and format of the Mercari test dataset

price_test.info()


#Checking for missing values in the train dataset

# Creating a colormap using 'inferno' colors

cmap = plt.get_cmap('inferno')


# Generate the missing data visualization with custom colors
```

```
msno.bar(price_train, sort='ascending', figsize=(10, 3), color=cmap(0.10))
# Display the plot
plt.show()

# Checking for missing values in the test dataset
# Creating colormap using 'inferno' colors
cmap = plt.get_cmap('inferno')
# Generate the missing dataset visualization with custom colors
msno.bar(price_test, sort='ascending', figsize=(10, 3), color=cmap(0.10))
# Display the plot
plt.show()
```

# Step 2: Pre-processing of the dataset

```
#removing unwanted columns
price_train=price_train.drop('date_first_available', axis=1)
price_train=price_train.drop('sell_date', axis=1)
price_test=price_test.drop('date_first_available', axis=1)
price_test=price_test.drop('sell_date', axis=1)

# Removing outlier with price more than 200
price_train['outliers'] = price_train['price'].map(lambda x: 1 if x >200 else 0)
price_train = price_train[price_train['outliers'] ==0]
del price_train['outliers']

#Getting the length of item description
price_train['length'] = price_train['item_description'].apply(lambda x: len(str(x)))
price_test['length'] = price_test['item_description'].apply(lambda x: len(str(x)))
```

```
#Merging test and train dataset for further preprocessing of the textual features

dataset = pd.concat([price_train,price_test])

#Defining a variable to spliot

dataset['train_or_not'] = dataset['train_id'].map(lambda x: 1 if x.is_integer() else 0)


#lowering letters

dataset['brand_name'] = dataset['brand_name'].map(lambda x: str(x).lower())

dataset['category_name'] = dataset['category_name'].map(lambda x: str(x).lower())

dataset['item_description'] = dataset['item_description'].map(lambda x: str(x).lower())

dataset['name'] = dataset['name'].map(lambda x: str(x).lower())


dataset['len_description'] = dataset['item_description'].map(lambda x: len(str(x).split()))


#Nan values in brand

dataset['brand_nan'] = dataset['brand_name'].map(lambda x: 1 if x =="nan" else 0)


#Number of unique brand names

print(len(set(dataset['brand_name'])))

print('brand_name in train',len(set(price_train['brand_name'])))

print('brand_name in test',len(set(price_test['brand_name'])))


#checking the duplicate 'brand_name' records comparing with test and train dataset

train_categories= list(set(price_train['brand_name']))

test_categories= list(set(price_test['brand_name']))


in_test_not_in_train = [x for x in test_categories if x not in train_categories]
```

```
print(len(in_test_not_in_train))


in_train_not_in_test = [x for x in train_categories if x not in test_categories]

print(len(in_train_not_in_test))


#category to make it easier to work with the individual categories for analysis

dataset['categories'] = dataset['category_name'].map(lambda x: list(str(x).split('/')))


# Items with no descriptions

dataset['no_description'] = dataset['item_description'].map(lambda x: 1 if str(x) =='no description yet' else
0)

print(len(dataset[dataset['no_description']==1]))


print('Blank brand_name column & no description',len(dataset[(dataset['brand_name']=='nan') &
(dataset['no_description'] ==1)]))


#No brand name and no desc

no_desc_no_brand = dataset[(dataset['brand_name']=='nan') & (dataset['no_description'] ==1)]

no_desc_no_brand['test'] = no_desc_no_brand['test_id'].apply(lambda x: 1 if x.is_integer() else 0)

no_desc_no_brand = no_desc_no_brand[no_desc_no_brand['test'] ==0]


#No of rows whose price is bigger than 200

print("No of rows whose price is bigger than 200 in above
category",len(no_desc_no_brand[no_desc_no_brand['price'] >200]))

no_desc_no_brand['price'].describe()

del no_desc_no_brand


cmap = plt.get_cmap('inferno')
```

```
# Filter the 'price' column in the 'train' dataset

filtered_prices = price_train[price_train['price'] < 200]['price']


# Create a histogram using Matplotlib with navy blue bars

plt.figure(figsize=(10, 5))

plt.hist(filtered_prices, bins=range(0, 201, 10), color=cmap(0.10))

plt.xlabel('Price')

plt.ylabel('Frequency')

plt.title('Histogram of Price in Train dataset')

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()


dataset['price'].describe().apply(lambda x: format(x, 'f'))


#Length of categories

dataset['len_categories'] = dataset['categories'].map(lambda x: len(x))


# Value counts for item_condition_id

item_condition = dataset['item_condition_id'].value_counts()[:5].to_frame().reset_index()


plt.figure(figsize=(10, 6))

sns.barplot(x='index', y='item_condition_id', data=item_condition, palette='inferno')

plt.title('Item Condition ID')

plt.xlabel('Condition ID')

plt.ylabel('Count')

plt.show()
```

```
#Making binary 'item_condition_id' feature column

ic_list = list(set(dataset['item_condition_id']))


for i in ic_list:

    dataset['item_condition_id'+str(i)] = dataset['item_condition_id'].map(lambda x: 1 if x==i else 0)


del dataset['item_condition_id']


#Correlation between len_description and price, shipping, len_categories, length, feature records

corr = dataset[['len_description','price','shipping','len_categories','length']].corr()


# Set up the matplot figure

f,ax = plt.subplots(figsize=(12,9))


#Draw the heatmap using seaborn

sns.heatmap(corr, cmap='inferno', annot=True)


#Determined important features based on above correlation matrix

most_imp = ['cat3_full-length', 'cat2_jewelry', 'cat3_tracksuits & sweats', 'item_description_case',
'name_michael', 'name_ring', 'name_nike', 'item_description_price', 'name_pink', 'cat3_headphones',
'len_description', 'item_description_[rm]', 'cat1_electronics', 'cat3_sticker', 'length',
'item_description_silver', 'cat3_consoles', 'item_condition_id5', 'item_description_-', 'brand_old navy',
'item_condition_id4', 'brand_forever 21', 'name_palette', 'cat2_cat2_other', 'name_bracelet',
'item_description_set', 'cat3_hoodie', 'name_boys', 'cat3_makeup palettes', 'name_purse',
'name_bundle', 'brand_lululemon', 'cat2_makeup', 'brand_beats', 'cat1_home', 'item_description_high',
'cat2_home appliances', 'item_description_full', 'cat2_dresses', 'brand_apple', 'brand_ugg australia',
'cat1_beauty', 'name_sleeve', 'brand_kate spade', 'brand_gymshark', "cat2_women's accessories",
'cat2_diapering', 'item_description_bag', 'name_air', 'name_eagle', "cat2_men's accessories",
'name_one', 'brand_lularoe', 'cat3_socks', 'name_jacket', 'name_coach', 'item_description_super',
'name_jordan', "brand_victoria's secret", 'brand_air jordan', 'cat1_women', 'item_description_fits',
'name_lace', 'item_description_new', 'name_top', 'brand_louis vuitton', 'brand_chanel', 'cat3_shoes',
'cat2_bags and purses', 'item_description_shipping', 'cat2_underwear', 'item_condition_id1',
'name_lularoe', 'item_description_comes', 'item_description_item', 'item_description_bundle',
'brand_samsung', 'name_disney', 'cat1_handmade', 'brand_supreme', 'brand_lilly pulitzer',
'item_description_see', 'shipping', 'name_case', 'cat2_cell phones & accessories', 'name_funko',
'name_silver', 'item_description_firm', 'cat3_shipping supplies', 'brand_tiffany & co.',
'item_description_8', 'brand_michael kors', 'brand_tory burch', 'cat2_sweaters', 'brand_kendra scott',
'name_vs', 'name_adidas', 'name_reserved', 'cat3_backpack style', 'brand_adidas',
'item_description_secret', 'item_condition_id2', 'item_description_ship', 'item_description_red',
'cat2_coats & jackets', 'cat3_hair styling tools', 'cat3_fleece jacket', 'cat2_shoes', 'item_description_free',
'name_shorts', 'brand_american eagle', 'len_categories', 'name_kors', 'brand_senegence', 'name_girls',
'item_description_x', 'name_makeup', 'name_shirt', 'item_description_box', 'item_description_body',
'cat3_pants, tights, leggings', 'name_set', 'item_description_gold', 'item_description_condition',
'brand_puma', 'brand_birkenstock', 'name_lot', 'item_description_cute', 'cat2_trading cards', 'cat1_men',
'item_description_7', 'item_description_beautiful', 'brand_miss me', 'name_gold', "cat2_women's
handbags", 'cat3_necklaces', 'brand_other_brand', 'cat3_dining & entertaining', 'cat2_tops & blouses',
'brand_dooney & bourke', 'item_description_save', 'item_condition_id3', 'brand_rae dunn', 'brand_pink',
'item_description_plus', 'cat3_jerseys', 'cat3_cosmetic bags', 'cat3_flats', 'cat3_athletic', 'brand_beats by
dr. dre', 'brand_nan', 'cat1_kids', 'brand_free people', 'cat2_computers & tablets', 'cat3_cases, covers &
skins', 'cat2_cameras & photography', 'cat3_boots', 'item_description_original']

print(len(most_imp))


itemdesc_imp = [x[17:] for x in most_imp if 'item_description_' in x]

name_imp = [x[5:] for x in most_imp if 'name_' in x]

brand_imp = [x[6:] for x in most_imp if 'brand_' in x]

cat1_imp= [x[5:] for x in most_imp if 'cat1_' in x]

cat2_imp= [x[5:] for x in most_imp if 'cat2_' in x]

cat3_imp= [x[5:] for x in most_imp if 'cat3_' in x]

other_imp = ['len_description', 'length', 'item_condition_id5', 'item_condition_id4', 'item_condition_id1',
'shipping', 'item_condition_id2', 'len_categories', 'item_condition_id3']


# New Pre-processed feature dataset based on correlation

# presented with relation to the main feature column and length associated with it

print("Length of new features related to each parent feature" )

print("Length of name: ",len(name_imp))

print("Length of item description: ",len(itemdesc_imp))

print("Length of brand: ",len(brand_imp))

print("Length of category_1: ",len(cat1_imp))
print("Length of category_2: ",len(cat2_imp))

print("Length of cat egory_3: ",len(cat3_imp))

print("Length of other important features: ",len(other_imp))

print("Total length of pre-processed features: ", len(most_imp))


new_features = ['product Name', 'Item Description', 'Brand', 'Category 1', 'Category 2', 'Category 3', 'Other
Important Features', ]

new_featurelengths = [len(name_imp), len(itemdesc_imp), len(brand_imp), len(cat1_imp), len(cat2_imp),
len(cat3_imp), len(other_imp)]


#creating a pie chart representation

colors = cm.inferno(np.linspace(0.2, 1, len(new_features)))

plt.figure(figsize=(8, 8))

plt.pie(new_featurelengths, labels=new_features, autopct='%1.1f%%', startangle=140, colors=colors,
shadow=True, explode=np.linspace(0.01,0.01, len(new_features)))

plt.title('Distribution of extracted features related to each parent feature and % weightage \n')

plt.axis('equal')

plt.show()


#Creating columns for parent "name" column

for i in name_imp:

    dataset['name_'+str(i)] = dataset['name'].map(lambda x: 1 if i in x else 0)



print("preprocessing on name feature completed")



#Creating columns for parent "item_description_" column

for i in itemdesc_imp:

    dataset['item_description_'+str(i)] = dataset['item_description'].map(lambda x: 1 if i in x else 0)
```

```
print("preprocessing on description feature completed")


#Creating columns for parent "brands" column

most_common_brands = brand_imp

#If a brand not in common brands, it was labeled as other_brand

other_brand = "other_brand"

dataset['brand_name'] = dataset['brand_name'].map(lambda x: x if x in most_common_brands else
other_brand)


# Creating dummnies for brand_name column

empty_df = pd.get_dummies(dataset['brand_name'])

impFeature_list = list(empty_df.columns.values)

impFeature_list = ['brand_'+str(x) for x in impFeature_list]

empty_df.columns = impFeature_list

print(impFeature_list)


dataset2 = pd.concat([dataset,empty_df],axis=1)

dataset = dataset2

del dataset2,empty_df

print("preprocessing on brand feature completed")


print(list(dataset.columns.values))

print(len(list(dataset.columns.values)))


#Creating separate  olumns for each category

dataset['categories']= dataset['categories'].map(lambda x: list(x)+[0,0,0,0])

dataset['cat1']=dataset['categories'].map(lambda x: x[0])
```

```
dataset['cat2']=dataset['categories'].map(lambda x: x[1])

dataset['cat3']=dataset['categories'].map(lambda x: x[2])

dataset['cat4']=dataset['categories'].map(lambda x: x[3])

dataset['cat5']=dataset['categories'].map(lambda x: x[4])


most_common_cat1=dataset['cat1'].value_counts().sort_values(ascending=False)[:11]

most_common_cat2=dataset['cat2'].value_counts().sort_values(ascending=False)[:70]

most_common_cat3=dataset['cat3'].value_counts().sort_values(ascending=False)[:90]

most_common_cat4=dataset['cat4'].value_counts().sort_values(ascending=False)[:100]

most_common_cat5=dataset['cat5'].value_counts().sort_values(ascending=False)[:100]


#Bucketing features for Category 1

cat1_b1 = ['women','vintage & collectibles','sports & outdoors','nan','home']

cat1_b2 = ['other','beauty','handmade']

cat1_b3 = ['men','electronics','beauty']

dataset['cat1_fe'] = dataset['cat1'].map(lambda x: 1 if x in cat1_b1 else 2 if x in cat1_b2 else 3)


#Bucketing features for Category 4

dataset['cat4_tablet'] = dataset['cat4'].map(lambda x: 1 if x =='tablet' else 0)


#Bucketing features for Category 5

ebook = ['ebook access','ebook readers']

dataset['cat5_ebook'] = dataset['cat5'].map(lambda x: 1 if x in ebook else 0)


#Considering the top 3 categories

most_common_cat1=cat1_imp

most_common_cat2=cat2_imp
```

```
most_common_cat3=cat3_imp

cat1_list = list(most_common_cat1)

cat2_list = list(most_common_cat2)

cat3_list = list(most_common_cat3)


#If a category not in cat1, it was labeled as 'cat1_other'

cat1_other = "cat1_other"

dataset['cat1'] = dataset['cat1'].map(lambda x: x if x in cat1_list else cat1_other)

#If a category not in cat2, it was labeled as 'cat2_other'

cat2_other = "cat2_other"

dataset['cat2'] = dataset['cat2'].map(lambda x: x if x in cat2_list else cat2_other)

#If a category not in cat3, it was labeled as 'cat3_other'

cat3_other = "cat3_other"

dataset['cat3'] = dataset['cat3'].map(lambda x: x if x in cat3_list else cat3_other)


cat1_exp = ['electronics']

dataset['cat1_exp'] = dataset['cat1'].map(lambda x: 1 if x in cat1_exp else 0)


cat2_exp = ["women's handbags","cell phones & accessories","shoes"]

dataset['cat2_exp'] = dataset['cat2'].map(lambda x: 1 if x in cat2_exp else 0)


cat3_exp = ["cell phones & smartphones","shoulder bag","athletic","totes & shoppers","messenger &
crossbody"]

dataset['cat3_exp'] = dataset['cat3'].map(lambda x: 1 if x in cat3_exp else 0)


good_brands = ['forever 21', 'american eagle', 'under armour', 'old navy', 'hollister', "carter's", 'brandy
melville', 'gap', 'charlotte russe', 'ralph lauren', 'converse', 'h&m', 'express', 'abercrombie & fitch', 'nyx',
```

'hot topic', 'calvin klein', "levi's®", 'anastasia beverly hills', 'torrid', 'tommy hilfiger', 'mossimo', 'aeropostale', 'columbia', 'guess', 'urban outfitters', 'target', 'xhilaration', 'maybelline', 'american apparel', 'maurices', 'elmers', 'rue21', "l'oreal", 'smashbox', 'champion', 'fashion nova', 'lucky brand', 'wet n wild', 'banana republic', 'toms', 'popsockets', 'wet seal', 'ann taylor loft', 'colourpop cosmetics', 'hello kitty', 'it cosmetics', 'merona', "osh kosh b'gosh", 'crocs', 'rue', 'e.l.f.', 'avon', 'revlon', "the children's place", 'starbucks', 'stila', 'jessica simpson', 'new york & company', 'lane bryant', 'pacific sunwear', 'skechers', 'motherhood maternity', 'nine west', "children's place", 'no boundaries', 'simply southern', 'athleta', 'roxy', 'fox racing', 'covergirl', 'bareminerals', 'aldo', 'gildan', 'new era', 'bare escentuals', 'silver jeans co.', 'yankee candle', 'bullhead', 'lacoste', 'lc lauren conrad', 'faded glory', 'hollister co.', 'hot wheels', 'billabong', 'laura mercier', 'tupperware', 'white house black market', 'affliction', 'stride rite', 'mac cosmetic', 'crest', 'sally hansen', 'nickelodeon', 'cacique', 'aéropostale', 'bobbi brown', "candie's", 'gillette', 'tobi', 'volcom', 'sperrys', 'mudd', 'gerber', 'leap frog', 'diamond supply co.', 'cato', 'nautica', 'laura geller', 'my little pony', 'disney princess', 'danskin', 'cherokee', 'mossimo supply co.', 'lime crime', 'vtech', 'sperry', 'dc shoes', 'daytrip', 'kenneth cole new york', 'dickies', 'stussy', 'pampered chef', 'cotton on', 'the limited', 'neutrogena', 'inc international concepts', 'ardell', 'hanna anderson', 'liz lange', 'so', 'comfort colors', 'liz claiborne', 'hurley', 'eddie bauer', 'bcbgeneration', "burt's bees", 'ann taylor', "chico's", "dr. brown's", 'nerf', 'thebalm', 'garnier', 'papaya', 'aden & anais', 'bongo', 'melissa & doug', 'fila', 'dove', 'make up for ever', 'american rag', 'ed hardy', 'sonoma', 'beautyblender®', 'aerie', 'petsmart', 'huggies', 'sesame street', 'ikea', 'anne klein', 'febreze', 'origins', 'pier one', 'worthington', 'munchkin', 'ivory ella', 'floam', 'bonne bell', 'ambiance apparel', 'avent', 'converse shoes', 'full tilt', 'dkny', 'vanity', 'shiseido', 'wrangler', 'lokai', 'arizona', 'the body shop', 'spanx', 'apt.', 'jumping beans', 'hourglass cosmetics', 'hard candy', 'a.n.a', 'obey', 'sperry top-sider', 'boppy', 'schick', 'rock & republic', 'simply vera vera wang', 'ben nye', 'almay', 'thrasher magazine', "lands' end", 'jennifer lopez', 'infantino', 'bke', "o'neill", 'rimmel', 'chaps', 'disney pixar cars', 'croft & barrow', 'op', "gilligan & o'malley", 'colgate', 'bdg', 'eos', 'rvca', 'pampers', 'dermablend', 'wilton', 'delia*s', 'modcloth', 'fabletics', 'ymi', 'venus', 'la hearts', 'dressbarn', 'disney pixar', "kiehl's", 'style&co.', 'soffe', 'playtex', 'tommee tippee', 'xoxo', 'vigoss', 'speedo', 'hanes', 'rave', 'paper mate', 'tommy bahama', 'sinful by affliction', 'derek heart', 'refuge', 'sanuk', 'talbots', 'elizabeth arden', 'olay', 'zella', 'lalaloopsy', 'avenue', 'pokemon usa', 'pampers swaddlers', "francesca's collections", 'gilly hicks', 'kendall & kylie', 'zumba', 'la idol', 'bumbo', 'arizona jean company', 'decree', 'huggies snug & dry', 'glade', 'dreamworks', 'franco sarto', "st. john's bay", 'nivea', 'chinese laundry', 'incipio', 'us polo assn', "claire's", 'boohoo', "lulu's", 'kotex', 'cabi', 'obey clothing', 'jones new york', 'crayola', 'disney jr.', 'everlast', 'lee', 'material girl', 'catalina', 'art', 'levi strauss & co.', 'bic', 'nick jr.', 'l.e.i.', "tilly's", 'dockers', 'russell athletic', 'capezio', 'kiplling', 'avia', 'charming charlie', 'air wick', 'nike golf', 'kimchi blue', 'hydraulic', 'dollhouse', 'geneva', 'justfab', 'kardashian kollection', 'pur minerals', 'nollie', 'lucy activewear', 'partylite', 'bobbie brooks', 'the hundreds', "dr. scholl's", 'hamilton beach', 'young & reckless', 'always', 'life is good', 'reef', 'southern marsh', 'brooks brothers', 'unionbay', 'izod', 'elle', 'fila sport', 'playskool', 'lenox', 'aerosoles', 'coty', 'baby phat', 'danskin now', 'moda international', 'bravado', 'sharpie', 'george', 'kodak', 'loft', 'belkin', 'apt. 9', 'red cherry', 'huf', 'a pea in the pod', 'john deere', 'new directions', 'robeez', 'twenty one', 'premier designs', 'ivanka trump', 'accessory workshop', 'ecko unltd.', 'safety st', 'max studio', 'hot kiss', 'jj cole collections', 'baby einstein', 'madden girl', 'tek gear', 'cynthia rowley', 'xersion', 'nostalgia electrics', 'bisou bisou', 'mam baby', 'huggies little snugglers', 'thrasher', 'arden b', 'angie', "summer's eve", 'nuk', 'quiksilver', 'precious moments', 'neff', 'degree', 'luvs', 'keds', 'maidenform', 'm.i.a.', 'pillow pets', 'celebrity pink', 'tahari', 'bass', "a'gaci", 'a. byer', 'one clothing', 'carbon', 'corningware', 'bright starts', 'scott paper', 'daisy fuentes', 'nhl', 'kut from the kloth', 'jaclyn smith', 'white stag', 'bandolino', 'cartoon network', 'silver jeans', 'missguided', 'fashion bug', 'jakks pacific', 'antonio melani', 'stance', 'rainbow shops', 'stüssy', 'qupid', 'belly bandit®', 'brita', 'browning', 'aveeno', 'lrg', "cabela's", 'oster', '% pure', 'huggies pull-ups', 'silence + noise', 'liz lange for target', 'ashley stewart', 'monopoly', 'k-swiss', 'willow', 'nascar', 'southpole', 'nicole miller', 'manic panic', 'mally beauty', 'homedics', 'charter club', 'minnetonka', 'angels', 'okie dokie', 'tonka', '47 brand', 'nuby', 'dress barn', 'gloria vanderbilt', 'love culture', 'aroma', 'rewind', 'realtree', 'machine', 'top paw', 'rampage', 'on the byas', 'rocawear', 'crooks & castles', 'ferasali', 'eyeshadow', 'coldwater creek', 'hasbro games', 'divided', 'jada toys', 'jockey', 'kenneth cole reaction',

```
'body central', 'body glove', 'jerzees', 'empire', 'fruit of the loom', 'guy harvey', 'bebop', 'my michelle',
'axe', 'poetry', 'marmot', 'dana buchman', 'perry ellis', 'tampax', 'urban pipeline', 'jolt', 'ball', 'next level',
'kirra', 'deb', 'soda', 'jansport', 'helly hansen', 'gund', 'van heusen', 'madame alexander', 'j. jill', 'thermos',
'kim rogers', 'sunbeam', 'just my size', 'speechless', 'frontline', 'wubbanub', 'the first years', 'mr. coffee',
'play-doh', 'primitive', 'anchor hocking', 'gianni bini', 'studio y', 'michael stars', 'mighty fine', 'marika',
'garage', 'self-esteem', 'charlotte tilbury', 'trixxi', 'lysol', 'conair', "frederick's of hollywood", 'a plus child
supply', 'flying monkey', 'city triangles', 'harajuku lovers', 'jane iredale', 'suave', 'fun world', 'g by guess',
'the sak', 'kensie', 'jamberry', 'kyodan', 'christopher & banks', 'breathablebaby', 'custom accessories', 'sean
john', 'lucky brand jeans', 'enfagrow', 'romeo & juliet couture', 'farberware', 'matchbox', 'esprit', 'baby
bullet', 'cowgirl tuff', 'dragon ball z', 'anvil', 'mossy oak', 'c by champion', 'nokia', 'pro keds', 'dr. seuss',
'energie', 'cottonelle', 'rubbermaid', 'boon', 'stayfree', 'lauren conrad', 'element', 'kong', 'covington',
'buffalo', 'ivivva', 'bcbg', 'sugarpill', 'huggies little movers', 'paris blues', 'tultex', 'baublebar', 'hue', 'cable &
gauge', 'soma', 'vocal', 'mitchum', 'teva', 'zion rootswear', 'scotch', 'gaiam', 'tomtom', 'cello jeans', 'pilot',
'style & co', "altar'd state", 'miley cyrus', 'bcx', 'blizzard', 'venezia', 'isaac mizrahi', 'ellen tracy', 'keen', 'fox',
'rival', 'zeroxposur', 'zoo york', 'rocket dog', 'discovery kids', 'melissa', 'blue asphalt', 'furminator', 'nick &
nora', 'shoe dazzle', 'vanilla star', 'as seen on tv', 'creativity for kids', 'dial', 'soprano', 'george foreman', 'en
focus studio', 'mango', 'laura ashley', 'andrew christian', 'pinkblush']

dataset['good_brand_or_not'] = dataset['brand_name'].map(lambda x: 1 if x in good_brands else 0)

#Making binary for cat1
empty_df1 = pd.get_dummies(dataset['cat1'])
impFeature_list1 = list(empty_df1.columns.values)
impFeature_list1 = ['cat1_' + str(x) for x in impFeature_list1]
empty_df1.columns = impFeature_list1
#Making binary for cat2
empty_df2 = pd.get_dummies(dataset['cat2'])
impFeature_list2 = list(empty_df2.columns.values)
impFeature_list2 = ['cat2_' + str(x) for x in impFeature_list2]
empty_df2.columns = impFeature_list2
#Making binary for cat3
empty_df3 = pd.get_dummies(dataset['cat3'])
impFeature_list3 = list(empty_df3.columns.values)
impFeature_list3 = ['cat3_' + str(x) for x in impFeature_list3]
```

```
empty_df3.columns = impFeature_list3

#Merging them

dataset2 = pd.concat([dataset,empty_df1,empty_df2,empty_df3],axis=1)

dataset = dataset2

#Deleting unnecessary things

del dataset2,empty_df1,empty_df2,empty_df3

del
dataset['cat1'],dataset['cat2'],dataset['cat3'],dataset['cat4'],dataset['cat5'],dataset['item_description'],dat
aset['name'],dataset['categories'],dataset['category_name'],dataset['brand_name']

print("pre-processing on category feature completed")


#printing information of the final pre-processed dataFrame

print(dataset.shape)


print(dataset.info())


#printing final pre-processed data and its shape

dataset


#preparation of next step for building ANN model

test_id = dataset['test_id']

train_id = dataset['train_id']

del dataset['train_id'],dataset['test_id']

dataset_head = dataset.head()

#Separating the merged dataset into train and test

training = dataset[dataset['train_or_not'] ==1]

testing = dataset[dataset['train_or_not'] ==0]
```

```
#deleting train and test datasetset identifiers

del training['train_or_not']

del testing['train_or_not']


# getting insights of training dataset

training


# getting insights of training dataset

testing


train_y = training['price'].values

train_y = np.log(train_y+1)

test_y = testing['price'].values

test_y = np.log(test_y+1)


#Deleting price column from train and test dataset

del training['price']

del testing['price']

train_size = len(list(training.columns.values))

train_names = list(training.columns.values)


print(train_names)


#setting identified important columns

testing.columns = train_names

training.columns = train_names
```

```
price_training = training

price_testing = testing


print("Shape of new dataset")

print("Training dataset",(price_training.shape))

print("Testing dataset",(price_testing.shape))



# calculating the input nodes for autoencoder algorithm

input_node = len(list(price_training.columns.values))

print("there are ",input_node," nodes in input layer")


# Step 3: Feature reduction

# Preparation of data for feature reduction

price_training = price_training.values

price_testing = price_testing.values


sc_X = StandardScaler()

price_training = sc_X.fit_transform(price_training)

price_testing = sc_X.transform(price_testing)


# Defining early stopping- a regularization technique

# Below mentioned code for callback will stop the training when there is no improvement in

# The loss for two consecutive epochs.


callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=2)
```

```
# Build an autoencoder for feature reduction #


# Below code defines variable for number of input and reduced feature set neurons

input_dim = price_training.shape[1]

encoding_dim = 128  # Number of neurons in the encoded layer


# Define an autoencoder architecture

input_layer = Input(shape=(input_dim,))

encoded = Dense(encoding_dim, activation='relu')(input_layer)

decoded = Dense(input_dim, activation='relu')(encoded)


autoencoder = keras.Model(inputs=input_layer, outputs=decoded)

autoencoder.compile(optimizer='adam', loss='mean_squared_error')


# Train an autoencoder

history= autoencoder.fit(price_training, price_training, epochs=50, batch_size=32, shuffle=True,
validation_data=(price_testing, price_testing), callbacks=callback)


# Using trained autoencoder extract reduced feature set

reducedFeatures= keras.Model(inputs=input_layer, outputs=encoded)

# Below reduced featureset now has 128 neurons which wil serve as input to the ANN model

encodedFeature_train = reducedFeatures.predict(price_training)

encodedFeature_test = reducedFeatures.predict(price_testing)


print(encodedFeature_train.shape)

print(encodedFeature_test.shape)
```

```
#summary of the autoencoder model for feature reduction

autoencoder.summary()


# The final number of epochs required to autoencode model to efficient ratio

print("Final number of epochs used:", len(history.history['loss']))


# visualization of the model architecture, shapes of layer and their names

tf.keras.utils.plot_model(autoencoder, show_shapes=True, show_layer_names=True)


print(encodedFeature_train.shape)

print(train_y.shape)
```

# Step 4: Building and training the ANN model

```
#Initializing the ANN

model = Sequential()

#extracting number of input neurons from reduced feature set

input_node=encodedFeature_test.shape[1]


#adding the input layer, one hidden layer and final output layer

model = tf.keras.Sequential([

   Dense(input_node, activation='relu', input_dim=input_node),

   Dense(64, activation='relu'),

   Dense(1, activation='relu')

])

#compiling ANN model

model.compile( optimizer='adam' , loss='mean_squared_logarithmic_error', metrics = ['mse']  )
```

```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=2)

start = datetime.now()

model.fit(encodedFeature_train, train_y ,batch_size=32,epochs=50, callbacks=callback)

stop = datetime.now()

execution_time = stop-start

print(execution_time)


# Step 5: Price prediction

# Predict prices

our_pred = model.predict(encodedFeature_test)


# visualization of the model architecure, shapes of layer and their names

tf.keras.utils.plot_model(model, show_shapes=True, show_layer_names=True)


#Execution time for the price prediction model

stop_real = datetime.now()

execution_time_real = stop_real start_real

print(execution_time_real)


# Step 6: Evaluation of the model and visualization

# plot loss during training the ANN model

plt.subplot(211)

plt.title('loss during training the ANN model')

plt.plot(history.history['loss'], label='loss')

plt.plot(history.history['val_loss'], label='val_loss')

plt.legend()
```

```
# Evaluate on test dataset

test_loss, test_accuracy = model.evaluate(encodedFeature_test, test_y)

print(f"Test Loss: {test_loss:.4f}, Test Accuracy: {test_accuracy:.4f}")


#Computes the mean of absolute of errors between labels and predictions.

mae=tf.keras.losses.mean_absolute_error(test_y, our_pred)

print("Mean Absolute Error (MAE):", mae.numpy())


# calculating a single scalar MAE value for better evaluation of the model

mean_mae = np.mean(mae)

print(f"Mean Absolute Error (MAE): {mean_mae:.4f}")


# Convert arrays to 1-dimensional arrays

test_y_1d = test_y.flatten()

our_pred_1d = our_pred.flatten()

df=pd.DataFrame({'Actual price':test_y_1d, 'predicted price':our_pred_1d})

df


# Creating a line plot representing actual vs predicted prices

plt.figure(figsize=(20, 10))

plt.plot(test_y, label='Actual Prices', marker='o', linestyle='-', color='blue')

plt.plot(our_pred, label='Predicted Prices', marker='o', linestyle='-', color='red')

plt.xlabel('dataset Points')

plt.ylabel('Prices')

plt.title('Actual Prices vs. Predicted Prices (Line Plot)')

plt.legend()

plt.grid(True)
plt.show()

# Create a scatter plot
plt.figure(figsize=(15, 7))
plt.scatter(test_y , our_pred.reshape(-1), c='blue', label='Actual vs. Predicted Prices', alpha=0.6)

# Add a regression line (best-fit line)
z = np.polyfit(test_y, our_pred.reshape(-1), 1)
p = np.poly1d(z)
plt.plot(test_y, p(test_y), c='red', label='Regression Line', linewidth=2)

# Add labels and title
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs. Predicted Prices with Regression Line')

# Add a legend
plt.legend()

# Display the plot
plt.grid(True)
plt.show()


# Calculate differences between actual and predicted prices
price_diff = (our_pred - test_y).flatten()
# Create a histogram of price differences with a single color
```

```
plt.figure(figsize=(8, 4))

plt.hist(price_diff, color='green', alpha=0.6, edgecolor='black')

plt.xlabel('Price Differences (Predicted - Actual)')

plt.ylabel('Frequency')

plt.title('Distribution of Price Differences')

plt.grid(True)

plt.show()


# Define a function to plot the learning curve

def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None, n_jobs=1, train_sizes=np.linspace(.1, 1.0, 5)):

    plt.figure()

    plt.title(title)

    if ylim is not None:

        plt.ylim(*ylim)

    plt.xlabel("Training examples")

    plt.ylabel("Mean Squared Error")

    train_sizes, train_scores, test_scores = learning_curve(

        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes, scoring='neg_mean_squared_error')

    train_scores_mean = -np.mean(train_scores, axis=1)

    train_scores_std = np.std(train_scores, axis=1)

    test_scores_mean = -np.mean(test_scores, axis=1)

    test_scores_std = np.std(test_scores, axis=1)

    plt.grid()


    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,

                     train_scores_mean + train_scores_std, alpha=0.1,
                 color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training error")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation error")

    plt.legend(loc="best")
    return plt

# Replace X and y with your actual data
X = test_y.reshape(-1, 1)  # Actual price
y = our_pred.reshape(-1)  # Predicted prices

# Create a linear regression model
from sklearn.linear_model import LinearRegression
estimator = LinearRegression()

title = "Learning Curves for price prediction"
cv = 10  # Number of cross-validation folds

# Plot the learning curve
plot_learning_curve(estimator, title, X, y, cv=cv)
plt.show()
```

Experiment phase II- Sale time Predictions

# Step 1: Installation and Setup

Import Section

```
# Importing necessary libraries

import pandas as pd

import numpy as np

import multiprocessing

%matplotlib inline


# Required to initialize NN

from keras.models import Sequential

# Required to build layers of ANN and autoencoder

import tensorflow as tf

from tensorflow import keras

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.layers import Input, Dense


# Required for visualization

import missingno as msno

import matplotlib.cm as cm

import scipy as sci

import seaborn as sns

import matplotlib.pyplot as plt


# Required for displaying learning_curve

from sklearn.model_selection import learning_curve
from datetime import datetime

start_real = datetime.now()

# Mounts google drive to load Test and Train dataset

from google.colab import drive

drive.mount('/content/drive')


# Retrieving train and test Mercari dataset records for processing

saleTime_train = pd.read_csv("/content/drive/My Drive/Project/Data/ANN/Mercari_train.csv",
encoding='utf-8-sig')

saleTime_test= pd.read_csv("/content/drive/My Drive/Project/Data/ANN/Mercari_test.csv",
encoding='utf-8-sig')


# Displaying the dimensions of the dataset (rows, column)

print(saleTime_train.shape)

print(saleTime_test.shape)


**Uncomment below code to run the model with limited dataset**


saleTime_train=saleTime_train[:80000]

saleTime_test=saleTime_test[:20000]


#checking content and format of the Mercari train dataset

saleTime_train.info()


#checking content and format of the Mercari test dataset

saleTime_test.info()
```

```
#Checking for missing values in the train dataset
# Creating a colormap using 'inferno' colors
cmap = plt.get_cmap('inferno')

# Generate the missing data visualization with custom colors
msno.bar(saleTime_train, sort='ascending', figsize=(10, 3), color=cmap(0.10))

# Display the plot
plt.show()

# Checking for missing values in the test dataset
# Creating colormap using 'inferno' colors
cmap = plt.get_cmap('inferno')

# Generate the missing dataset visualization with custom colors
msno.bar(saleTime_test, sort='ascending', figsize=(10, 3), color=cmap(0.10))

# Display the plot
plt.show()
```

# Step 2: Pre-processing of the dataset

```
# Removing outlier with price more than 200
saleTime_train['outliers'] = saleTime_train['price'].map(lambda x: 1 if x >200 else 0)
saleTime_train = saleTime_train[saleTime_train['outliers'] ==0]
del saleTime_train['outliers']
```

```
#Getting the length of item description

saleTime_train['length'] = saleTime_train['item_description'].apply(lambda x: len(str(x)))

saleTime_test['length'] = saleTime_test['item_description'].apply(lambda x: len(str(x)))


#Merging test and train dataset for further preprocessing of the textual features

dataset = pd.concat([saleTime_train,saleTime_test])

#Defining a variable to spliot

dataset['train_or_not'] = dataset['train_id'].map(lambda x: 1 if x.is_integer() else 0)


#lowering letters

dataset['brand_name'] = dataset['brand_name'].map(lambda x: str(x).lower())

dataset['category_name'] = dataset['category_name'].map(lambda x: str(x).lower())

dataset['item_description'] = dataset['item_description'].map(lambda x: str(x).lower())

dataset['name'] = dataset['name'].map(lambda x: str(x).lower())


dataset['len_description'] = dataset['item_description'].map(lambda x: len(str(x).split()))


#Nan values in brand

dataset['brand_nan'] = dataset['brand_name'].map(lambda x: 1 if x =="nan" else 0)


#Number of unique brand names

print(len(set(dataset['brand_name'])))

print('brand_name in train',len(set(saleTime_train['brand_name'])))

print('brand_name in test',len(set(saleTime_test['brand_name'])))
```

```
#checking the duplicate 'brand_name' records comparing with test and train dataset

train_categories= list(set(saleTime_train['brand_name']))

test_categories= list(set(saleTime_test['brand_name']))


in_test_not_in_train = [x for x in test_categories if x not in train_categories]

print(len(in_test_not_in_train))


in_train_not_in_test = [x for x in train_categories if x not in test_categories]

print(len(in_train_not_in_test))


#category to make it easier to work with the individual categories for analysis

dataset['categories'] = dataset['category_name'].map(lambda x: list(str(x).split('/')))


# Items with no descriptions

dataset['no_description'] = dataset['item_description'].map(lambda x: 1 if str(x) =='no description yet' else
0)

print(len(dataset[dataset['no_description']==1]))


print('Blank brand_name column & no description',len(dataset[(dataset['brand_name']=='nan') &
(dataset['no_description'] ==1)]))


#No brand name and no desc

no_desc_no_brand = dataset[(dataset['brand_name']=='nan') & (dataset['no_description'] ==1)]

no_desc_no_brand['test'] = no_desc_no_brand['test_id'].apply(lambda x: 1 if x.is_integer() else 0)

no_desc_no_brand = no_desc_no_brand[no_desc_no_brand['test'] ==0]


#No of rows whose price is bigger than 200
```

```
print("No of rows whose price is bigger than 200 in above category",len(no_desc_no_brand[no_desc_no_brand['price'] >200]))


no_desc_no_brand['price'].describe()

del no_desc_no_brand


**Logic for sale time calculation- output feature- start**


#calculating sale time for marketplace ecommerce products- defining dynamic column here sale_time


# Convert date columns to datetime format

#data['sell_date'] = pd.to_datetime(data['sell_date'])

dataset['date_first_available'] = pd.to_datetime(dataset['date_first_available'], format='%d-%m-%Y')

dataset['sell_date'] = pd.to_datetime(dataset['sell_date'], format='%d-%m-%Y')


# Calculate the difference between the two date columns

dataset['sale_time'] = dataset['sell_date'] - dataset['date_first_available']

print(dataset[['date_first_available', 'sell_date', 'sale_time']])


#converting timedel

dataset['sale_time']=dataset['sale_time'].dt.days


#removing unwanted columns

dataset=dataset.drop('date_first_available', axis=1)

dataset=dataset.drop('sell_date', axis=1)
```

```
**Logic for sale time calculation- output feature- end**


# Display final sale time column (prediction column)

dataset['sale_time']


dataset['price'].describe().apply(lambda x: format(x, 'f'))


#Length of categories

dataset['len_categories'] = dataset['categories'].map(lambda x: len(x))


# Value counts for item_condition_id

item_condition = dataset['item_condition_id'].value_counts()[:5].to_frame().reset_index()


plt.figure(figsize=(10, 6))

sns.barplot(x='index', y='item_condition_id', data=item_condition, palette='inferno')

plt.title('Item Condition ID')

plt.xlabel('Condition ID')

plt.ylabel('Count')

plt.show()


#Making binary 'item_condition_id' feature column

ic_list = list(set(dataset['item_condition_id']))


for i in ic_list:

    dataset['item_condition_id'+str(i)] = dataset['item_condition_id'].map(lambda x: 1 if x==i else 0)
```

```
del dataset['item_condition_id']


#Correlation between len_description and price feature

corr = dataset[['len_description','sale_time','shipping','len_categories','length','price']].corr()


# Set up the matplot figure

f,ax = plt.subplots(figsize=(12,9))


#Draw the heatmap using seaborn

sns.heatmap(corr, cmap='inferno', annot=True)


#Determined important features based on above correlation matrix

most_imp = ['cat3_full-length', 'cat2_jewelry', 'cat3_tracksuits & sweats', 'item_description_case', 'name_michael', 'name_ring', 'name_nike', 'item_description_price', 'name_pink', 'cat3_headphones', 'len_description', 'item_description_[rm]', 'cat1_electronics', 'cat3_sticker', 'length', 'item_description_silver', 'cat3_consoles', 'item_condition_id5', 'item_description_-', 'brand_old navy', 'item_condition_id4', 'brand_forever 21', 'name_palette', 'cat2_cat2_other', 'name_bracelet', 'item_description_set', 'cat3_hoodie', 'name_boys', 'cat3_makeup palettes', 'name_purse', 'name_bundle', 'brand_lululemon', 'cat2_makeup', 'brand_beats', 'cat1_home', 'item_description_high', 'cat2_home appliances', 'item_description_full', 'cat2_dresses', 'brand_apple', 'brand_ugg australia', 'cat1_beauty', 'name_sleeve', 'brand_kate spade', 'brand_gymshark', "cat2_women's accessories", 'cat2_diapering', 'item_description_bag', 'name_air', 'name_eagle', "cat2_men's accessories", 'name_one', 'brand_lularoe', 'cat3_socks', 'name_jacket', 'name_coach', 'item_description_super', 'name_jordan', "brand_victoria's secret", 'brand_air jordan', 'cat1_women', 'item_description_fits', 'name_lace', 'item_description_new', 'name_top', 'brand_louis vuitton', 'brand_chanel', 'cat3_shoes', 'cat2_bags and purses', 'item_description_shipping', 'cat2_underwear', 'item_condition_id1', 'name_lularoe', 'item_description_comes', 'item_description_item', 'item_description_bundle', 'brand_samsung', 'name_disney', 'cat1_handmade', 'brand_supreme', 'brand_lilly pulitzer', 'item_description_see', 'shipping', 'name_case', 'cat2_cell phones & accessories', 'name_funko', 'name_silver', 'item_description_firm', 'cat3_shipping supplies', 'brand_tiffany & co.', 'item_description_8',
'brand_michael kors', 'brand_tory burch', 'cat2_sweaters', 'brand_kendra scott', 'name_vs', 'name_adidas', 'name_reserved', 'cat3_backpack style', 'brand_adidas', 'item_description_secret', 'item_condition_id2', 'item_description_ship', 'item_description_red', 'cat2_coats & jackets', 'cat3_hair styling tools', 'cat3_fleece jacket', 'cat2_shoes', 'item_description_free', 'name_shorts', 'brand_american eagle', 'len_categories', 'name_kors', 'brand_senegence', 'name_girls', 'item_description_x', 'name_makeup', 'name_shirt', 'item_description_box', 'item_description_body', 'cat3_pants, tights, leggings', 'name_set', 'item_description_gold', 'item_description_condition', 'brand_puma', 'brand_birkenstock', 'name_lot', 'item_description_cute', 'cat2_trading cards', 'cat1_men', 'item_description_7', 'item_description_beautiful', 'brand_miss me', 'name_gold', "cat2_women's handbags", 'cat3_necklaces', 'brand_other_brand', 'cat3_dining & entertaining', 'cat2_tops & blouses', 'brand_dooney & bourke', 'item_description_save', 'item_condition_id3', 'brand_rae dunn', 'brand_pink', 'item_description_plus', 'cat3_jerseys', 'cat3_cosmetic bags', 'cat3_flats', 'cat3_athletic', 'brand_beats by dr. dre', 'brand_nan', 'cat1_kids', 'brand_free people', 'cat2_computers & tablets', 'cat3_cases, covers & skins', 'cat2_cameras & photography', 'cat3_boots', 'item_description_original']

print(len(most_imp))


itemdesc_imp = [x[17:] for x in most_imp if 'item_description_' in x]

name_imp = [x[5:] for x in most_imp if 'name_' in x]

brand_imp = [x[6:] for x in most_imp if 'brand_' in x]

cat1_imp= [x[5:] for x in most_imp if 'cat1_' in x]

cat2_imp= [x[5:] for x in most_imp if 'cat2_' in x]

cat3_imp= [x[5:] for x in most_imp if 'cat3_' in x]

other_imp = ['len_description', 'length', 'item_condition_id5', 'item_condition_id4', 'item_condition_id1', 'shipping', 'item_condition_id2', 'len_categories', 'item_condition_id3']


# New Pre-processed feature dataset based on correlation

# presented with relation to the main feature column and length associated with it

print("Length of new features related to each parent feature" )

print("Length of name: ",len(name_imp))
print("Length of item description: ",len(itemdesc_imp))

print("Length of brand: ",len(brand_imp))

print("Length of category_1: ",len(cat1_imp))

print("Length of category_2: ",len(cat2_imp))

print("Length of cat egory_3: ",len(cat3_imp))

print("Length of other important features: ",len(other_imp))

print("Total length of pre-processed features: ", len(most_imp))


new_features = ['product Name', 'Item Description', 'Brand', 'Category 1', 'Category 2', 'Category 3', 'Other Important Features', ]

new_featurelengths = [len(name_imp), len(itemdesc_imp), len(brand_imp), len(cat1_imp), len(cat2_imp), len(cat3_imp), len(other_imp)]


#creating a pie chart representation

colors = cm.inferno(np.linspace(0.2, 1, len(new_features)))

plt.figure(figsize=(8, 8))

plt.pie(new_featurelengths, labels=new_features, autopct='%1.1f%%', startangle=140, colors=colors, shadow=True, explode=np.linspace(0.01,0.01, len(new_features)))

plt.title('Distribution of extracted features related to each parent feature and % weightage \n')

plt.axis('equal')

plt.show()


#Creating columns for parent "name" column

for i in name_imp:

  dataset['name_'+str(i)] = dataset['name'].map(lambda x: 1 if i in x else 0)


print("preprocessing on name feature completed")
```

```
#Creating columns for parent "item_description_" column
for i in itemdesc_imp:
    dataset['item_description_'+str(i)] = dataset['item_description'].map(lambda x: 1 if i in x else 0)
print("preprocessing on description feature completed")

#Creating columns for parent "brands" column
most_common_brands = brand_imp
#If a brand not in common brands, it was labeled as other_brand
other_brand = "other_brand"
dataset['brand_name'] = dataset['brand_name'].map(lambda x: x if x in most_common_brands else 
other_brand)

# Creating dummnies for brand_name column
empty_df = pd.get_dummies(dataset['brand_name'])
impFeature_list = list(empty_df.columns.values)
impFeature_list = ['brand_'+str(x) for x in impFeature_list]
empty_df.columns = impFeature_list
print(impFeature_list)

dataset2 = pd.concat([dataset,empty_df],axis=1)
dataset = dataset2
del dataset2,empty_df
print("preprocessing on brand feature completed")

print(list(dataset.columns.values))
print(len(list(dataset.columns.values)))


#Creating separate  olumns for each category

dataset['categories']= dataset['categories'].map(lambda x: list(x)+[0,0,0,0])

dataset['cat1']=dataset['categories'].map(lambda x: x[0])

dataset['cat2']=dataset['categories'].map(lambda x: x[1])

dataset['cat3']=dataset['categories'].map(lambda x: x[2])

dataset['cat4']=dataset['categories'].map(lambda x: x[3])

dataset['cat5']=dataset['categories'].map(lambda x: x[4])


most_common_cat1=dataset['cat1'].value_counts().sort_values(ascending=False)[:11]

most_common_cat2=dataset['cat2'].value_counts().sort_values(ascending=False)[:70]

most_common_cat3=dataset['cat3'].value_counts().sort_values(ascending=False)[:90]

most_common_cat4=dataset['cat4'].value_counts().sort_values(ascending=False)[:100]

most_common_cat5=dataset['cat5'].value_counts().sort_values(ascending=False)[:100]


#Bucketing features for Category 1

cat1_b1 = ['women','vintage & collectibles','sports & outdoors','nan','home']

cat1_b2 = ['other','beauty','handmade']

cat1_b3 = ['men','electronics','beauty']

dataset['cat1_fe'] = dataset['cat1'].map(lambda x: 1 if x in cat1_b1 else 2 if x in cat1_b2 else 3)


#Bucketing features for Category 4

dataset['cat4_tablet'] = dataset['cat4'].map(lambda x: 1 if x =='tablet' else 0)


#Bucketing features for Category 5
```

```
ebook = ['ebook access','ebook readers']

dataset['cat5_ebook'] = dataset['cat5'].map(lambda x: 1 if x in ebook else 0)


#Considering the top 3 categories

most_common_cat1=cat1_imp

most_common_cat2=cat2_imp

most_common_cat3=cat3_imp

cat1_list = list(most_common_cat1)

cat2_list = list(most_common_cat2)

cat3_list = list(most_common_cat3)


#If a category not in cat1, it was labeled as 'cat1_other'

cat1_other = "cat1_other"

dataset['cat1'] = dataset['cat1'].map(lambda x: x if x in cat1_list else cat1_other)

#If a category not in cat2, it was labeled as 'cat2_other'

cat2_other = "cat2_other"

dataset['cat2'] = dataset['cat2'].map(lambda x: x if x in cat2_list else cat2_other)

#If a category not in cat3, it was labeled as 'cat3_other'

cat3_other = "cat3_other"

dataset['cat3'] = dataset['cat3'].map(lambda x: x if x in cat3_list else cat3_other)


cat1_exp = ['electronics']

dataset['cat1_exp'] = dataset['cat1'].map(lambda x: 1 if x in cat1_exp else 0)


cat2_exp = ["women's handbags","cell phones & accessories","shoes"]

dataset['cat2_exp'] = dataset['cat2'].map(lambda x: 1 if x in cat2_exp else 0)
```

```
cat3_exp = ["cell phones & smartphones","shoulder bag","athletic","totes & shoppers","messenger &
crossbody"]

dataset['cat3_exp'] = dataset['cat3'].map(lambda x: 1 if x in cat3_exp else 0)
```

```
good_brands = ['forever 21', 'american eagle', 'under armour', 'old navy', 'hollister', "carter's", 'brandy
melville', 'gap', 'charlotte russe', 'ralph lauren', 'converse', 'h&m', 'express', 'abercrombie & fitch', 'nyx', 'hot
topic', 'calvin klein', "levi's®", 'anastasia beverly hills', 'torrid', 'tommy hilfiger', 'mossimo', 'aeropostale',
'columbia', 'guess', 'urban outfitters', 'target', 'xhilaration', 'maybelline', 'american apparel', 'maurices',
'elmers', 'rue21', "l'oreal", 'smashbox', 'champion', 'fashion nova', 'lucky brand', 'wet n wild', 'banana
republic', 'toms', 'popsockets', 'wet seal', 'ann taylor loft', 'colourpop cosmetics', 'hello kitty', 'it cosmetics',
'merona', "osh kosh b'gosh", 'crocs', 'rue', 'e.l.f.', 'avon', 'revlon', "the children's place", 'starbucks', 'stila',
'jessica simpson', 'new york & company', 'lane bryant', 'pacific sunwear', 'skechers', 'motherhood
maternity', 'nine west', "children's place", 'no boundaries', 'simply southern', 'athleta', 'roxy', 'fox racing',
'covergirl', 'bareminerals', 'aldo', 'gildan', 'new era', 'bare escentuals', 'silver jeans co.', 'yankee candle',
'bullhead', 'lacoste', 'lc lauren conrad', 'faded glory', 'hollister co.', 'hot wheels', 'billabong', 'laura mercier',
'tupperware', 'white house black market', 'affliction', 'stride rite', 'mac cosmetic', 'crest', 'sally hansen',
'nickelodeon', 'cacique', 'aéropostale', 'bobbi brown', "candie's", 'gillette', 'tobi', 'volcom', 'sperrys', 'mudd',
'gerber', 'leap frog', 'diamond supply co.', 'cato', 'nautica', 'laura geller', 'my little pony', 'disney princess',
'danskin', 'cherokee', 'mossimo supply co.', 'lime crime', 'vtech', 'sperry', 'dc shoes', 'daytrip', 'kenneth cole
new york', 'dickies', 'stussy', 'pampered chef', 'cotton on', 'the limited', 'neutrogena', 'inc international
concepts', 'ardell', 'hanna anderson', 'liz lange', 'so', 'comfort colors', 'liz claiborne', 'hurley', 'eddie bauer',
'bcbgeneration', "burt's bees", 'ann taylor', "chico's", "dr. brown's", 'nerf', 'thebalm', 'garnier', 'papaya',
'aden & anais', 'bongo', 'melissa & doug', 'fila', 'dove', 'make up for ever', 'american rag', 'ed hardy',
'sonoma', 'beautyblender®', 'aerie', 'petsmart', 'huggies', 'sesame street', 'ikea', 'anne klein', 'febreze',
'origins', 'pier one', 'worthington', 'munchkin', 'ivory ella', 'floam', 'bonne bell', 'ambiance apparel', 'avent',
'converse shoes', 'full tilt', 'dkny', 'vanity', 'shiseido', 'wrangler', 'lokai', 'arizona', 'the body shop', 'spanx',
'apt.', 'jumping beans', 'hourglass cosmetics', 'hard candy', 'a.n.a', 'obey', 'sperry top-sider', 'boppy', 'schick',
'rock & republic', 'simply vera vera wang', 'ben nye', 'almay', 'thrasher magazine', "lands' end", 'jennifer
lopez', 'infantino', 'bke', "o'neill", 'rimmel', 'chaps', 'disney pixar cars', 'croft & barrow', 'op', "gilligan &
o'malley", 'colgate', 'bdg', 'eos', 'rvca', 'pampers', 'dermablend', 'wilton', 'delia*s', 'modcloth', 'fabletics',
```

'ymi', 'venus', 'la hearts', 'dressbarn', 'disney pixar', "kiehl's", 'style&co.', 'soffe', 'playtex', 'tommee tippee', 'xoxo', 'vigoss', 'speedo', 'hanes', 'rave', 'paper mate', 'tommy bahama', 'sinful by affliction', 'derek heart', 'refuge', 'sanuk', 'talbots', 'elizabeth arden', 'olay', 'zella', 'lalaloopsy', 'avenue', 'pokemon usa', 'pampers swaddlers', "francesca's collections", 'gilly hicks', 'kendall & kylie', 'zumba', 'la idol', 'bumbo', 'arizona jean company', 'decree', 'huggies snug & dry', 'glade', 'dreamworks', 'franco sarto', "st. john's bay", 'nivea', 'chinese laundry', 'incipio', 'us polo assn', "claire's", 'boohoo', "lulu's", 'kotex', 'cabi', 'obey clothing', 'jones new york', 'crayola', 'disney jr.', 'everlast', 'lee', 'material girl', 'catalina', 'art', 'levi strauss & co.', 'bic', 'nick jr.', 'l.e.i.', "tilly's", 'dockers', 'russell athletic', 'capezio', 'kiplling', 'avia', 'charming charlie', 'air wick', 'nike golf', 'kimchi blue', 'hydraulic', 'dollhouse', 'geneva', 'justfab', 'kardashian kollection', 'pur minerals', 'nollie', 'lucy activewear', 'partylite', 'bobbie brooks', 'the hundreds', "dr. scholl's", 'hamilton beach', 'young & reckless', 'always', 'life is good', 'reef', 'southern marsh', 'brooks brothers', 'unionbay', 'izod', 'elle', 'fila sport', 'playskool', 'lenox', 'aerosoles', 'coty', 'baby phat', 'danskin now', 'moda international', 'bravado', 'sharpie', 'george', 'kodak', 'loft', 'belkin', 'apt. 9', 'red cherry', 'huf', 'a pea in the pod', 'john deere', 'new directions', 'robeez', 'twenty one', 'premier designs', 'ivanka trump', 'accessory workshop', 'ecko unltd.', 'safety st', 'max studio', 'hot kiss', 'jj cole collections', 'baby einstein', 'madden girl', 'tek gear', 'cynthia rowley', 'xersion', 'nostalgia electrics', 'bisou bisou', 'mam baby', 'huggies little snugglers', 'thrasher', 'arden b', 'angie', "summer's eve", 'nuk', 'quiksilver', 'precious moments', 'neff', 'degree', 'luvs', 'keds', 'maidenform', 'm.i.a.', 'pillow pets', 'celebrity pink', 'tahari', 'bass', "a'gaci", 'a. byer', 'one clothing', 'carbon', 'corningware', 'bright starts', 'scott paper', 'daisy fuentes', 'nhl', 'kut from the kloth', 'jaclyn smith', 'white stag', 'bandolino', 'cartoon network', 'silver jeans', 'missguided', 'fashion bug', 'jakks pacific', 'antonio melani', 'stance', 'rainbow shops', 'stüssy', 'qupid', 'belly bandit®', 'brita', 'browning', 'aveeno', 'lrg', "cabela's", 'oster', '% pure', 'huggies pull-ups', 'silence + noise', 'liz lange for target', 'ashley stewart', 'monopoly', 'k-swiss', 'willow', 'nascar', 'southpole', 'nicole miller', 'manic panic', 'mally beauty', 'homedics', 'charter club', 'minnetonka', 'angels', 'okie dokie', 'tonka', '47 brand', 'nuby', 'dress barn', 'gloria vanderbilt', 'love culture', 'aroma', 'rewind', 'realtree', 'machine', 'top paw', 'rampage', 'on the byas', 'rocawear', 'crooks & castles', 'ferasali', 'eyeshadow', 'coldwater creek', 'hasbro games', 'divided', 'jada toys', 'jockey', 'kenneth cole reaction', 'body central', 'body glove', 'jerzees', 'empire', 'fruit of the loom', 'guy harvey', 'bebop', 'my michelle', 'axe', 'poetry', 'marmot', 'dana buchman', 'perry ellis', 'tampax', 'urban pipeline', 'jolt', 'ball', 'next level', 'kirra', 'deb', 'soda', 'jansport', 'helly hansen', 'gund', 'van heusen', 'madame alexander', 'j. jill', 'thermos', 'kim rogers', 'sunbeam', 'just my size', 'speechless', 'frontline', 'wubbanub', 'the first years', 'mr. coffee', 'play-doh', 'primitive', 'anchor hocking', 'gianni bini', 'studio y', 'michael stars', 'mighty fine', 'marika', 'garage', 'self-esteem', 'charlotte tilbury', 'trixxi', 'lysol', 'conair', "frederick's of hollywood", 'a plus child supply', 'flying monkey', 'city triangles', 'harajuku lovers', 'jane iredale', 'suave', 'fun world', 'g by guess', 'the sak', 'kensie', 'jamberry', 'kyodan', 'christopher & banks', 'breathablebaby', 'custom accessories', 'sean john',

```
'lucky brand jeans', 'enfagrow', 'romeo & juliet couture', 'farberware', 'matchbox', 'esprit', 'baby bullet', 'cowgirl tuff', 'dragon ball z', 'anvil', 'mossy oak', 'c by champion', 'nokia', 'pro keds', 'dr. seuss', 'energie', 'cottonelle', 'rubbermaid', 'boon', 'stayfree', 'lauren conrad', 'element', 'kong', 'covington', 'buffalo', 'ivivva', 'bcbg', 'sugarpill', 'huggies little movers', 'paris blues', 'tultex', 'baublebar', 'hue', 'cable & gauge', 'soma', 'vocal', 'mitchum', 'teva', 'zion rootswear', 'scotch', 'gaiam', 'tomtom', 'cello jeans', 'pilot', 'style & co', "altar'd state", 'miley cyrus', 'bcx', 'blizzard', 'venezia', 'isaac mizrahi', 'ellen tracy', 'keen', 'fox', 'rival', 'zeroxposur', 'zoo york', 'rocket dog', 'discovery kids', 'melissa', 'blue asphalt', 'furminator', 'nick & nora', 'shoe dazzle', 'vanilla star', 'as seen on tv', 'creativity for kids', 'dial', 'soprano', 'george foreman', 'en focus studio', 'mango', 'laura ashley', 'andrew christian', 'pinkblush']

dataset['good_brand_or_not'] = dataset['brand_name'].map(lambda x: 1 if x in good_brands else 0)

#Making binary for cat1
empty_df1 = pd.get_dummies(dataset['cat1'])
impFeature_list1 = list(empty_df1.columns.values)
impFeature_list1 = ['cat1_' + str(x) for x in impFeature_list1]
empty_df1.columns = impFeature_list1
#Making binary for cat2
empty_df2 = pd.get_dummies(dataset['cat2'])
impFeature_list2 = list(empty_df2.columns.values)
impFeature_list2 = ['cat2_' + str(x) for x in impFeature_list2]
empty_df2.columns = impFeature_list2
#Making binary for cat3
empty_df3 = pd.get_dummies(dataset['cat3'])
impFeature_list3 = list(empty_df3.columns.values)
impFeature_list3 = ['cat3_' + str(x) for x in impFeature_list3]
empty_df3.columns = impFeature_list3
#Merging them
```

```
dataset2 = pd.concat([dataset,empty_df1,empty_df2,empty_df3],axis=1)

dataset = dataset2

#Deleting unnecessary things

del dataset2,empty_df1,empty_df2,empty_df3

del
dataset['cat1'],dataset['cat2'],dataset['cat3'],dataset['cat4'],dataset['cat5'],dataset['item_description'],dat
aset['name'],dataset['categories'],dataset['category_name'],dataset['brand_name']

print("pre-processing on category feature completed")


#printing information of the final pre-processed dataFrame

print(dataset.shape)


print(dataset.info())


#printing final pre-processed data and its shape

dataset


# Compute the correlation matrix

correlation_matrix = dataset.corr()


# Create a heatmap

plt.figure(figsize=(12, 8))

sns.heatmap(correlation_matrix, cmap='coolwarm', annot=False)

plt.title('Correlation Heatmap of the reduced feature set')


# Display the plot

plt.show()
```

```
#preparation of next step for building ANN model

test_id = dataset['test_id']

train_id = dataset['train_id']

del dataset['train_id'],dataset['test_id']

dataset_head = dataset.head()

#Separating the merged dataset into train and test

training = dataset[dataset['train_or_not'] ==1]

testing = dataset[dataset['train_or_not'] ==0]


#deleting train and test datasetset identifiers

del training['train_or_not']

del testing['train_or_not']


# getting insights of training dataset

training


# getting insights of training dataset

testing


#Adding feature to predict toy values for train and test dataset

train_y = training['sale_time'].values

test_y = testing['sale_time'].values


#Deleting columns to predict from training and testing dataset
```

```
del training['sale_time']

del testing['sale_time']

train_size = len(list(training.columns.values))

train_names = list(training.columns.values)


print(train_names)


#setting identified important columns

testing.columns = train_names

training.columns = train_names


saleTime_training = training

saleTime_testing = testing


print("Shape of new dataset")

print("Training dataset",(saleTime_training.shape))

print("Testing dataset",(saleTime_testing.shape))




# calculating the input nodes for autoencoder algorithm

input_node = len(list(saleTime_training.columns.values))

print("there are ",input_node," nodes in input layer")

```

# Step 3: Feature reduction

```
# Preparation of data for feature reduction

saleTime_training = saleTime_training.values
```

```
saleTime_testing = saleTime_testing.values


sc_X = StandardScaler()

saleTime_training = sc_X.fit_transform(saleTime_training)

saleTime_testing = sc_X.transform(saleTime_testing)


# defining early stopping- a regularization technique

# Below mentioned code for callback will stop the training when there is no improvement in

# The loss for two consecutive epochs


callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)


# Build an autoencoder for feature reduction #


# Below code defines variable for number of input and reduced feature set neurons

input_dim = saleTime_training.shape[1]

encoding_dim = 128  # Number of neurons in the encoded layer


# Define an autoencoder architecture

input_layer = Input(shape=(input_dim,))

encoded = Dense(encoding_dim, activation='relu')(input_layer)

decoded = Dense(input_dim, activation='relu')(encoded)


autoencoder = keras.Model(inputs=input_layer, outputs=decoded)

autoencoder.compile(optimizer='adam', loss='mean_squared_error')
```

```
# Train an autoencoder

history=  autoencoder.fit(saleTime_training,  saleTime_training,  epochs=50,  batch_size=32,  shuffle=True,
validation_data=(saleTime_testing, saleTime_testing), callbacks=callback)


# Using trained autoencoder extract reduced feature set

reducedFeatures= keras.Model(inputs=input_layer, outputs=encoded)

# Below reduced dataset now has 128 neurons which wil serve as input to the ANN model

encodedFeature_train = reducedFeatures.predict(saleTime_training)

encodedFeature_test = reducedFeatures.predict(saleTime_testing)



#summary of the autoencoder model for feature reduction

autoencoder.summary()


# The final number of epochs required to autoencode model to efficient ratio

print("Final number of epochs used:", len(history.history['loss']))


# visualization of the model architecure, shapes of layer and their names

tf.keras.utils.plot_model(autoencoder, show_shapes=True, show_layer_names=True)


print(encodedFeature_train.shape)

print(train_y.shape)
```

# Step 4: Building and training the ANN model

```
#Initializing the ANN

model = Sequential()
```

```
#extracting number of input neurons from reduced feature set

input_node=encodedFeature_test.shape[1]


#adding the input layer, one hidden layer and final output layer

model = tf.keras.Sequential([

   Dense(input_node, activation='relu', input_dim=input_node),

   Dense(64, activation='relu'),

   Dense(1)

])

#compiling ANN model

model.compile( optimizer='adam' , loss='mean_squared_logarithmic_error', metrics = ['mse']  )


#implementing early stopping

callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=2)

start = datetime.now()

model.fit(encodedFeature_train, train_y ,batch_size=32,epochs=50, callbacks=callback)

stop = datetime.now()

execution_time = stop-start

print(execution_time)
```

# Step 5: Sale time Forecast

```
# Predict sale time

our_pred = model.predict(encodedFeature_test)


# visualization of the model architecure, shapes of layer and their names

tf.keras.utils.plot_model(model, show_shapes=True, show_layer_names=True)
```

```
model.summary()


stop_real = datetime.now()

execution_time_real = stop_real start_real

print(execution_time_real)
```

# Step 6: Evaluation of the model and visualization

```
# plot loss during training the ANN model

plt.subplot(211)

plt.title('loss during training the ANN model')

plt.plot(history.history['loss'], label='loss')

plt.plot(history.history['val_loss'], label='val_loss')

plt.legend()


# Evaluate on test data

test_loss, test_accuracy = model.evaluate(encodedFeature_test, test_y)

print(f"Test Loss: {test_loss:.4f}, Test Accuracy: {test_accuracy:.4f}")


#Computes the mean of squares of errors.

mae=tf.keras.losses.mean_absolute_error(test_y, our_pred)

print("Mean Absolute Error (MAE):", mae.numpy())


# calculating a single scalar MAE value for better evaluation of the model

mean_mae = np.mean(mae)

print(f"Mean Absolute Error (MAE): {mean_mae:.4f}")
```

```
# Convert arrays to 1-dimensional arrays

test_y_1d = test_y.flatten()

our_pred_1d = our_pred.flatten()

df=pd.DataFrame({'Actual':test_y_1d, 'predicted sale time':our_pred_1d})

df


# Creating a line plot representing actual vs predicted sale time

plt.figure(figsize=(20, 10))

plt.plot(test_y, label='Actual sale time', marker='o', linestyle='-', color='blue')

plt.plot(our_pred, label='Predicted sale time', marker='o', linestyle='-', color='red')

plt.xlabel('Data Points')

plt.ylabel('sale time')

plt.title('Actual sale time vs. Predicted sale time (Line Plot)')

plt.legend()

plt.grid(True)

plt.show()


# Create a scatter plot

plt.figure(figsize=(15, 7))

plt.scatter(test_y , our_pred.reshape(-1), c='blue', label='Actual vs. Predicted sale time', alpha=0.6)


# Add a regression line (best-fit line)

z = np.polyfit(test_y, our_pred.reshape(-1), 1)

p = np.poly1d(z)

plt.plot(test_y, p(test_y), c='red', label='Regression Line', linewidth=2)
```

```
# Add labels and title

plt.xlabel('Actual sale time')

plt.ylabel('Predicted sale time')

plt.title('Actual vs. Predicted sale time with Regression Line')


# Add a legend

plt.legend()


# Display the plot

plt.grid(True)

plt.show()



# Calculate differences between actual and predicted sale time

saleTime_diff = (our_pred - test_y).flatten()

# Create a histogram for above value

plt.figure(figsize=(8, 4))

plt.hist(saleTime_diff, color='green', alpha=0.6, edgecolor='black')

plt.xlabel('Differences between predicted and actual sale time')

plt.ylabel('Frequency')

plt.title('Distribution of sale time Differences')

plt.grid(True)

plt.show()


# Define a function to plot the learning curve
```

```
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None, n_jobs=1, train_sizes=np.linspace(.1,
1.0, 5)):
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Mean Squared Error")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes, scoring='neg_mean_squared_error')
    train_scores_mean = -np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = -np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training error")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation error")
    plt.legend(loc="best")

    return plt


# Replace X and y with your actual data

X = test_y.reshape(-1, 1)  # Actual sale time

y = our_pred.reshape(-1)  # Predicted sale time


# Create a linear regression model

from sklearn.linear_model import LinearRegression

estimator = LinearRegression()


title = "Learning Curves for sale time prediction"

cv = 10  # Number of cross-validation folds


# Plot the learning curve

plot_learning_curve(estimator, title, X, y, cv=cv)

plt.show()
```

Appendix E – The full set of features learnt in the functional testing phase

No.	Feature	Type	Description
1	Id	Numeric	the id of the listing
2	name	Categorical	The name of the product
3	item_condition_id	Numeric	The condition of the items provided by the seller
4	Category_name	Categorical	Category of the product
5	brand_name	Categorical	The brand name for the product
7	price	Numeric	The price of the product in USD
8	shipping	Categorical	The indicator of shipping paid by seller or buyer
9	item_description	Categorical	The full description of the product
10	date_first_available	Date Time	The date product is uploaded on the online store
11	sell_date	Date Time	The date product is sold through the online store
12	sale_time	Days	dynamically created

Appendix F – Contents of the original Input Files (.tsv)

Description of the first twenty e-commerce product details from test.csv file is as follows:

test_id name item_condition_id category_name brand_name shippingitem_description

0 "Breast cancer ""I fight like a girl"" ring" 1 Women/Jewelry/Rings 1 Size 7

1 "25 pcs NEW 7.5""x12"" Kraft Bubble Mailers" 1 Other/Office supplies/Shipping Supplies 1 "25 pcs NEW 7.5""x12"" Kraft Bubble Mailers Lined with bubble wrap for protection Self Sealing (peel-and-seal), adhesive keeps contents secure and tamper proof Durable and lightweight Kraft material helps save on postage Approved by UPS, FedEx, and USPS."

2 Coach bag 1 Vintage & Collectibles/Bags and Purses/Handbag Coach 1 Brand new coach bag. Bought for [rm] at a Coach outlet.

3 Floral Kimono 2 Women/Sweaters/Cardigan 0 -floral kimono -never worn -lightweight and perfect for hot weather

4 Life after Death 3 Other/Books/Religion & Spirituality 1 Rediscovering life after the loss of a loved one by Tony Cooke. Paperback in good condition 2003. ♥ ♥ Bundle and save! ♥ ♥ Book, death, grief, bereavement SHLF.SW.5.15

Five iPhone 6 Plus or 6s Plus Vodka pink case 1 Electronics/Mobiles & Accessories/Cases, Covers & Skins 1 One Absolut Vodka in Pink for iPhone 6 Plus and fits the 6s Plus iPhone. These are made of a flexible rubber material. Brand new case Also have in 6 and 6s size in pink Size: iPhone 6 Plus and 6s Plus Free shipping

6 Vintage Cameo Pendant & Brooch Pin 3 Women/Jewelry/Necklaces Vintage 1 "Two vintage Cameo pieces. 1. Silver metal Locket pendant with filigree, green background, and ivory cameo. It is 2"". 2. Gold tone metal, mirrored background with gray cameo. Pinback and a loop for a chain. It is also 2"". Free shipping."

7 Rose Gold Stainless Steel Quartz Watch 1 Women/Women's Accessories/Watches 1 "Brand new Price firm No trades Box included with 1yr warranty card and dust cloth. Metal: Stainless steel Finish: Polished Width: 12MM Length: 8"" Can be adjusted smaller Stainless steel will not rust, tarnish, change colours or turn skin green. Watch features quartz movement. Face is rose gold and stainless-steel case. Band is stainless steel rose gold."

8 Daisy Marc Jacobs 3.4oz 3 Beauty/Fragrance/Women MARC JACOBS 0 Brand new No box 100% authentic Firm price NO offers

9 Rose Brushes and Silicone Sponge 1 Beauty/Tools & Accessories/Makeup Brushes & Tools 1 All new. 12 pcs makeup brushes and one Silicone beauty Sponge

10 BNWT Coach coated canvas wristlet 1 Handmade/Bags and Purses/Wristlet 0 Authentic. Retail [rm]. Bramble rose. BNWT. Comes with box.

11 """John Carpenters Halloween""" 2 Electronics/Media/DVD 0 --NO FREE SHIPPING-- •No Scratches• ◆All PRICES Are FIRM Since Mercāri Charges 10% Sellers Fee on Every Single Transaction. ◆Ask Me to Bundle Items to Save On Shipping Fees Only. I DO NOT Discount on Bundles. ◆ADULT OWNED 100% AUTHENTIC MOVIES! #Horror Movie

12 Tuff Jeans Cowgirl Tuff Co. 3 Women/Jeans/Boot Cut Cowgirl Tuff 0 27 waist thirty-three length Decorated with copper and silver hardware Silver studded pocket line, Silver studded leg hem line. Studded Flair Bottoms Bronco Style Cowgirl Tuff Co.

13 Lululemon crop size 6 3 Women/Athletic Apparel/Pants, Tights, Leggings Lululemon 0 Super soft cropped sweat pant. Good condition - no flaws. Tags Leggings LulaRoe Nike Small

14 Versace Eros 200 ml 6.7 FL oz for men 1 Beauty/Fragrance/Men 1 Brand new authentic money back guarantee Free shipping

15 Loot crate exclusive kill bill socks 1 Men/Athletic Apparel/Socks 0 Brand new loot crate exclusive kill bill vol.1 men's socks

16 Tea party set 3 Kids/Toys/Dress Up & Pretend Play Fisher-Price 0 Comes with fisher price tea pot Talks batteries not included two cups two spoons and three plates Dessert shapes that fit in yellow thing M

17 Jordan 2 Kids/Boys (4+)/Coats & Jackets 0 Boys XL michael Jordan zip up jacket

18 MISS ME (media & entertainment) CROSS BLING JEANS 2 Women/Jeans/Boot Cut Miss Me 0 Women’s size 25 Inseam 29 In excellent condition No fraying or wear Have the distressed look on the front Price and shipping firm

19 Elephant Choker 1 Women/Jewelry/Necklaces 1 BRAND NEW ○ PRICE FIRM ○ △FREE SHIPPING△

20 3 SCARY STORIES BOOKS 2 TELL IN THE DARK 3 Other/Books/Children's Books 1 Ships same/next day. Free Shipping!! Can make a great gift... No writing or used, just imperfections on cover(s). Description The Scary Stories Set includes these three books: Scary Stories to Tell in the Dark, More Scary Stories to Tell in the Dark, and Scary Stories 3: More Tales to Chill Your

Bones. All three books are written by Alvin Schwartz with drawings by the original illustrator, Stephen Gammell. Paperback Publisher: Scholastic Language: English Product Dimensions: 7.5 x 5.3 inches Children's kid’s toddlers spooky Halloween best seller camping books boy’s girls’ gift present birthday

Description of the first twenty e-commerce product details from train.csv file is mentioned below:

train_id	name	item_condition_id	category_name	brand_name	price	shipping	item_description
0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts		10.0	1	No description yet
1	Razer Black Widow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & Parts	Razer	52.0	0	This keyboard is in excellent condition and works like it came out of the box. All the ports are assessed and work perfectly. The lights are customizable via the Razer Synapse app on your PC.
2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	1	Adorable top with a hint of lace and a keyhole in the back! The pale pink is a one times, and I also have three times available in white!
3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents		35.0	1	New with tags. Leather horses. Retail for [rm] each. Stand about a foot high. They are being sold as a pair. Any questions please ask. Free shipping. Just got out of storage
4	24K GOLD plated rose	1	Women/Jewelry/Necklaces		44.0	0	Complete with certificate of authenticity
5	Bundled items requested for Ruie	3	Women/Other/Other		59.0	0	Banana republic bottoms, Candies skirt with matching blazer,Amy Byers suit, Loft bottoms and cami top.
6	Acacia pacific tides santorini top	3	Women/Swimwear/Two-Piece	Acacia Swimwear	64.0	0	Size small but straps slightly shortened to fit xs, besides that, perfect condition
7	Girls cheer and tumbling bundle of 7	3	Sports & Outdoors/Apparel/Girls	Soffe	6.0	1	You get three pairs of Sophie cheer shorts size small and medium girls and two sports bra/boy shorts spandex matching sets in small and medium girls. All items total retail for [rm] in store and you can take him today for less than the price of one item at the store!)
8	Girls Nike Pro shorts	3	Sports & Outdoors/Apparel/Girls	Nike	19.0	0	Girls Size small Plus green. Three shorts total.

9	Porcelain clown doll checker pants VTG	3	Vintage & Collectibles/Collectibles/Doll		8.0	0	"I realized his pants are on backwards after the picture. They were very dirty, so I hand washed them. He has a stuffed body and painted porcelain head, hands, and feet. Back before clowns were too scary. 9"" tall. No chips or cracks but minor paint loss in a few places. Clown Circus Doll Collectible"

10	Smashbox primer	2	Beauty/Makeup/Face	Smashbox	8.0	1	0.25 oz Full size is 1oz for [rm] in Sephora

11	New vs pi k body mists	1	Beauty/Fragrance/Women	Victoria's Secret 34.0	0	(5) new vs pink body mists (2.5 oz each) Fresh & Clean Sun kiss Cool and bright Total flirt Sweet and flirty

12	Black Skater dress	2	Women/Dresses/Above Knee, Mini	rue	16.0	0	Xl, excellent condition

13	Sharpener and eraser	1	Other/Office supplies/School Supplies	Scholastic	4.0	1	No description yet

14	HOLDS for Dogs2016 Minnetonka boots	3	Women/Shoes/Boots	UGG Australia	43.0	0	Authentic. Suede fringe boots. Great condition! Size seven. If you are between the sizes 5.5-7 and love wearing thick socks during the winter they would be perfect for you as well (I did last winter) :)

15	Sephora tarte birthday gift	1	Beauty/Makeup/Makeup Sets	Tarte	11.0	1	Brand new. Deluxe travel size products. Contains: Amazonian Clay 12-hour Blush in Paaarty - .05oz/1.5g Tarteist Lip Paint in Birthday Suit - .034oz/1ml

16	Glitter Eyeshadow	1	Beauty/Makeup/Eyes	Wet n Wild	6.0	1	2 glitter eyeshadows; one in Brass and one in Bleached.

17	New: Baby K'tan active baby carrier	1	Kids/Gear/Backpacks & Carriers		29.0	1	Brand new in box Size: Medium Color: Coral Retails for [rm] The Baby K’tan ACTIVE is made of a breathable hi-tech performance fabric that wicks away moisture and sweat, blocks over 90% of the sun’s harmful UVA and UVB rays, and provides a unique temperature control. - Ergonomic positioning for healthy infant development. - Evenly distributes weight across back and shoulders. - Double-loop design slips on like a t-shirt.

18	"Too Faced Limited ""Merry Macaroons"""	1	Beauty/Makeup/Makeup Palettes	Too Faced	25.0	1	"This AUTHENTIC palette by Too Faced is brand new in mint condition still in original box. It is part of the. Christmas 2016 collection. It has twelve pretty eye shadow colours and a small sized” “Better Than Sex"" mascara. Never even swatched. Impeccable shape. Price includes 2-day priority shipping with insurance."

19 Cream/ Beige Front Cross Shirt 2 Women/Tops & Blouses/Blouse Anthropologie 27.0
0 Fancy, dressy or casual! Dress it up or down 100% polyester; washed once, never dried. Size: Small Brand: lush Purchased from Francesca's Tags: Free People, Anthropology, Dry Goods, Francesca's

20 Torrid Nautical Peplum Tube Top 3 Women/Tops & Blouses/Tank, Cami Torrid 13.0
1 Size 1. Worn once. Excellent condition