AI In Product Management

Prof. Zillur Rahman
Department of Management Studies
Indian Institute of Technology, Roorkee

Lecture - 43
Price Optimization using AI (Part 3)

Welcome to this NPTEL online certification course on artificial intelligence in product management. And we are talking about module 43, that is price optimization using AI. And now we will be talking about part 3 of this. So, this is what we are talking about. So, to give you an overview of this module, we will start with briefly introducing algorithms and how pricing algorithms work, how machine learning influence pricing, what is the role of data in AI-assisted pricing, what is data pricing,

Scraping and difference between data scraping and data crawling. Why data scraping is important? How to use it for getting competitors pricing? What are some of the important tools used for data scrapping and how to automate web scrapping with? RPAs.

What are the benefits and challenges of RPAs for web scrapping and how to automate RPAs for scrapping. Price optimization and forecasting using ML in crisis situation. The role of reinforcement learning in pricing and its challenges and the implementation of RL based dynamic pricing. So these are the things that we will cover in this module. So, to give an introduction to AI algorithms, AI pricing algorithms finds optimal prices for your products or services.

They analyze huge amounts of data. The data ranges from market trends and competition to external factors such as weather or economic conditions. AI pricing algorithms process and interpret this data. Then, they can make accurate predictions about consumer demand and set prices. Powered by machine learning and data analysis, AI pricing algorithms offer businesses new opportunities to optimize their pricing strategies.

One of the key advantages of AI pricing algorithms is their ability to handle vast amounts of data. Traditional pricing methods like Excel and ERP systems often rely on manual analysis, which can be time-consuming and prone to human error. In contrast, AI algorithms can process and analyze massive datasets in a fraction of the time, allowing businesses to make more informed pricing decisions. However, AI pricing algorithms can adapt and learn from new data, continuously improving their accuracy over time. This

adaptability is crucial in today's fast-paced business environment, where market conditions and consumer preferences can change rapidly.

Now, let us look at how pricing algorithms work. AI pricing algorithms work by gathering and analyzing large datasets to make informed pricing decisions. They use machine learning to understand complex market dynamics and continuously adjust pricing strategies to optimize profitability. This process involves a cycle of training, predictions, and adjustments to ensure that the pricing recommendations remain accurate and effective over time. We will break down the steps through which they typically operate.

The first step is collecting your historical data. The algorithms gather historical and real-time data from various sources. This includes internal data like sales history, production costs, inventory levels, as well as external data such as market trends, competitor pricing, and consumer behavior. The second step is processing the accumulated data. The collected data is cleaned and prepared for analysis.

AI algorithms require high-quality, relevant data to make accurate predictions. This stage often involves handling missing values, removing outliers, and ensuring the data is in a format suitable for analysis. The third step is selecting relevant factors. The algorithms identify which factors or features are most relevant to pricing. This could include product types, time of day, seasonality, customer segments, and more.

Selecting the right features is crucial. for the effectiveness of the pricing models. The fourth step is training the model. The AI uses the prepared data to train a machine learning model. This involves feeding the data into the model and adjusting the model's parameters until it can accurately predict prices.

Common techniques include regression analysis, decision trees, and neural networks. The fifth step is testing and validation. The model is tested on a separate set of data to validate its accuracy. This helps ensure that the model will perform well on real-world data and not just the data it was trained on.

The sixth step is predictive analysis. Once validated, the AI models analyze current market data and predict the most effective pricing strategy. It considers how different pricing options might affect demand, competitor responses, and overall profitability. The seventh step is dynamic pricing.

The AI algorithms continuously update their predictions based on new data. As market conditions change, the algorithms adjust their pricing recommendations to remain optimal. This is often referred to as dynamic or real-time pricing. When formed well, on real-world data and not just the data it was trained on.

The eighth step is price monitoring and adjustments. The performance of the AI pricing model is continuously monitored. If the market changes or the model's predictions become less accurate, it may be retrained or adjusted to improve its performance. Now we will understand how machine learning influences pricing. Machine learning plays a crucial role in driving the core functionality of AI pricing algorithms.

These algorithms are trained on massive datasets that contain valuable information about consumer behavior. By analyzing this data, machine learning models can identify patterns, correlations, and trends that human analysts might miss. This enables the algorithms to make accurate predictions about how customers will respond to different price points. One of the key advantages of machine learning is its ability to continuously learn from new data. As more information becomes available, the algorithms can update their models and improve their predictive capabilities over time.

This iterative learning process ensures that the AI pricing algorithms stay up to date with market conditions and customer preferences. Now we will look at the role of data in AI pricing. The more comprehensive and diverse the data, the more accurate the algorithms can be in predicting consumer behavior and optimizing pricing. A pricing tool's function of data scraping gathers prices of competitors' product assortments across various e-commerce platforms.

This then helps move forward to set competitors' pricing strategically. Businesses collect data from various sources to create a holistic understanding of the market dynamics. Now let us look at the role of historical data. Historical sales data provides valuable insights into past purchasing patterns and customer preferences. By analyzing this data, AI pricing algorithms can identify seasonal trends, peak buying periods, and other factors that influence consumer behavior.

This information allows businesses to adjust their prices accordingly and maximize their revenue. The next thing that we will consider is customer demographics. Customer demographics also play a significant role in AI pricing algorithms. By understanding the characteristics of different customer segments, businesses can tailor their pricing strategies to meet the unique needs and preferences of each group.

Another example of pricing algorithms is that luxury brands might set higher prices for affluent customers while offering discounts to price-sensitive shoppers for selected products. Then comes the role of competitor pricing information. Competitors' pricing information is another crucial data source for AI pricing algorithms. By studying the prices set by competitors, businesses can benchmark their own pricing strategies and ensure they remain competitive in the market.

This information allows businesses to identify pricing gaps and opportunities, enabling them to adjust their prices strategically. Then, we will look at what data scraping is. In today's highly competitive market, knowing what your competitors charge for similar products or services is essential. By understanding your competitors' pricing strategies, you can make informed decisions about your pricing and stay competitive in the market. However, manually collecting competitors' prices can take time and effort. Data scraping with the right tools can help find and collect prices and automate the process, making it quick and smooth. Generally, data scraping or web scraping is the process of automatically collecting data from websites. It involves extracting data from HTML pages, parsing it, and storing it in a structured format such as spreadsheets or databases.

Data scraping can be conducted manually, but it is usually done using automated software tools. These tools use algorithms to extract specific data from websites, such as product information, customer reviews, and competitor prices. Price Scraping Price scraping is a process that focuses on extracting competitor prices from websites and online platforms. Similar to general data scraping, you can scrape prices using pricing platforms.

While you can gather this information manually, businesses prefer using an automated tool to collect, store in an organized structure, and analyze the data. This process significantly reduces the time and bandwidth required when done otherwise. So now, what is the difference between data scraping and data crawling? The main difference between data scraping and data crawling is the scope and the purpose of the data extraction. Data scraping is focused on specific data within a web page or a document, while data crawling is focused on the web pages or documents themselves.

Data scraping is usually done for a specific analysis or task. While data crawling is usually done for general exploration or indexing. Data scraping can be done on any webpage or document, while data crawling requires a starting point and a set of rules or

criteria to follow. So, why is data scraping important? Scraping data is crucial as it lets you gather ample information through a streamlined process.

It saves managers time and resources compared to manually collecting data, and it can provide insights that would be difficult to obtain otherwise. For example, data scraping can be used to monitor competitive pricing, collect customer reviews and feedback, and track social media mentions of your brand. It can also extract data from market research and analysis, and gather data for academic or scientific research. So, how can you get competitive pricing using data scraping? You can scrape data from websites manually using specific programming languages.

However, due to today's fast-paced business environment, you can scrape prices using ready-made software tools to extract data from websites. Data scraping can be done by using a variety of programming languages, including Python, JavaScript, and Ruby. Moreover, many scraping tools also have user interfaces that allow non-technical users to scrape data without writing code. So, how can you scrape prices from competitors' websites and online platforms?

So, here is how to scrape data from a website in general. Start with identifying the website you want to check and the data you want to collect. Then use a scraping tool to extract data from the website. Parse the data to extract specific information you need and store the data in a structured format such as a CSV file or database. In the next slide, we will look at strategic methods for finding competitor pricing using data scraping.

So, this section discusses how to find competitors' pricing and other metrics to start an analysis. In short, you would have to identify your direct competitors and then understand the metrics you need. You also need to define your objective in gathering this information and how you can implement it in your pricing strategy. So, here the first step is to identify your competitors. Before you can start scraping competitors' pricing data, you need to identify who your competitors are.

First, list all the companies that offer similar products or services to yours; you can use search engines or industry directories to find them. Then you can categorize your competitors in three ways: primary, secondary, and tertiary. Once you have done that, you can focus on the direct competitors to determine your immediate pricing decisions. Step two is to determine the data you want to collect.

Once you have identified your competitors, you must determine the data you want to collect. Of course, pricing is the most critical data to manage, but you may also want to collect other data. such as product descriptions, customer reviews, and shipping costs. You can scrape all your competitors' prices and record them in a structured system. For a comprehensive analysis, you may want to gather the following: price index, product availability, and competitors' additional offers.

The third step is to choose a data scraping tool. In this step, you can scrape data from websites, but instead of scraping prices manually, you should use a price scraping tool to do the work for you. These data scraping tools extract various data types from websites and are not limited to just prices. Of the many data scraping tools, popular ones include Scrapy, Beautiful Soup, and Selenium.

These tools allow you to extract data from websites and store it in a structured format like Excel or Google Sheets for organized storage. The fourth step is to create a data scraping script. Once you have chosen a data scraping tool, you need to create a scraping script. Your scraping script should include the URLs of the websites you want to scrape, the data you wish to collect, and any rules you want to apply to the scraping process. The fifth step is to run your data scraping script.

Once you have created your scraping script, you need to run it. The scraping process can take some time depending on the number of websites you are scraping and the amount of data you are collecting. The sixth step is to analyze your scraped data. After the data scraping process is complete, you need to analyze the data you have collected. You can use tools such as Excel or Python to analyze the data and identify trends in your competitors' pricing strategies. Step 7 is to make informed decisions about your pricing strategy. Finally, you can use your collected data to make informed decisions about your pricing strategy. By understanding your competitors' pricing strategies, you can set your prices at a competitive and profitable level.

Now, let us look at the various data web scraping tools. So, we have listed down and discussed some of the popular and widely used web scraping tools. The first is Bright Data. It provides more than 230 web scraper APIs. Covering a wide range of data sources such as social media, e-commerce, real estate, and travel.

Additionally, it offers proxy-based APIs like Web Unlocker and SERP API. These scraper APIs are equipped with a proxy tool that allows for targeting at both country and city levels across any location. The APIs feature capabilities such as JavaScript

rendering, IP rotation, and anti-detection measures. Another is Oxylabs. It is a premium proxy provider that offers a variety of web scraping APIs, including the Web Scraper API, SERP Scraper API, Real Estate Scraper, and E-Commerce Scraper API.

Their web scrapping APIs allow targeting at the country level in 195 locations while the SERP Scraper API provides more precise targeting options such as city and coordinate level targeting. Their Web Scraper API supports headless browsers to render and extract data from JavaScript heavy websites another web scrapping tool is Nimble it offers general purpose SERP e-commerce and maps APIs featuring integrated rotating residential proxies and unlocker proxy solutions the web API is capable of handling Batch requests allowing up to 1000 URLs in each batch. The Nimble Web API offers three methods for data delivery.

First is real-time data is collected and instantly returned to the user. Second is cloud storage. Collected data is sent to the user's chosen cloud storage service. And the third is push-pull. Data is stored in Emble servers and can be accessed through a provided URL for download.

Another is Apify. Apify is a developer-focused web scrapping platform that offers pre-made scrappers and automation tools called actors. Actors are designed to automate actions or extract data from social media sites, e-commerce platforms, review sites, job portals, and other websites. Every actor can be accessed via API using Python. javascript or HTTP requests you can use actors as they are asked to modify them for your use case or create your own developers can create and run actors in various programming languages such as javascript typescript and python by using code templates universal scrappers or the open source web scrapping library that is crawly Apify runs on a cloud-based infrastructure

with built-in scrapping features such as data center and residential proxies, automatic IP rotation, capture solving, monitoring, scheduling, and integrations. Another is Smartproxy. Smartproxy provides four web scrapper API services like social media, service scrapping, e-commerce, and web scrapping APIs. These APIs come with a residential proxy network and support country-level targeting, while the SERP scrapping API also enables coordinate-level targeting.

The scrapping APIs from Smartproxy include essential features like proxy rotation, entry detection methods, and JavaScript rendering. SOAX It offers social media, SERP, and e-

commerce API. The vendor provides built-in proxy management capabilities and handles pagination. Users can set a max page parameter for Scraping Multiple Pages.

Zyte, it provides an API with sophisticated proxy management features and browse automation capabilities. The Scraper API allows for handling request headers, cookies, and toggling JavaScripts. Now, how to automate web scrapping with RPAs? RPAs stand for Robotic Process Automation, which is a type of software that performs a repetitive task by replicating human interactions with GUI, GUI stands for graphical user interface elements.

The interest in RPA is rising as the technology matures and vendors provide low- or no-code interfaces to build RPA boards. The global RPA market is expected to reach $120 million by 2027. To reach $11 billion by 2027, RPA is one of the top candidates to automate any repetitive task, and typical rule-based processes can be 70-80% automated. When done manually, web scraping can be a tedious task.

With many clicks, scrolls, and copy-paste repetitions to extract the designated data. That is why it is compelling to use RPA to automate web scraping. RPA bots perform repetitive tasks by replicating GUI processes, which a human user would typically do to perform a task. For web scraping, users would find the URL they want to scrape,

inspect the pages to find data relevant to their searches, write code (for example, in Python), or use an extension to extract the data and export it to a table. RPA bots can be programmed to log into the designated URL, scroll through multiple pages, extract specific data, and transform it into the required format. The bot can also enter the extracted data directly into another application or system for different uses. That is, send an email, modify spreadsheet fields, etc. Now, what are the benefits of RPA in web scraping?

So, web scrapping technology provides the following benefits. First is to eliminate manual data entry errors. Second is to extract images, text and videos. Third is reduce time for data extraction and entry. Fourth is automatically monitor websites and portals regularly for data changes.

So, web scraping tools or RPA software allow users to build scraping bots without writing code or script for data extraction. RPAs can be the right tool for web scrapping, especially if more data processing needs to be done on scrapped data. Different technologies can be easily integrated into the RPA bots used in scraping. For example, a

machine learning API integrated to the scraping bot could identify companies' websites from their extracted logos.

Now, what are the challenges in using RPAs for web scraping? Since RPA bots rely on GUI elements to organize the wanted data, it is difficult to automate web scrapping when pages do not display content in a consistent manner. Some of the examples are as follows. One is pop-up ads. Pop-up elements are ads.

can hide GUI elements from the bot's vision and disable it from extracting the underlying data. The solution would be to use an ad blocker extension for the browser used for web scrapping. Another is "load more" button. Typically, the bot will scroll down a page, extract the data, and export it to the output file. Some web pages, especially product pages,

load data in parts and allow users to explore more product via load more button. When this is the case, the bot will stop extracting the data by the end of the page instead of exploring more products. The solution is to create an if loop within the bot program to click load more GUI element if it exists until no more buttons appear on the web pages. Go to the next page, same as in the show more button, some content may be loaded in the following pages. The solution would also be to create a loop to click on the next page GUI element to open the following URL.

Scrape Protection System Websites like LinkedIn use sophisticated technology to protect their websites from getting scrapped. In such cases, users have a few options. one work with a company that provides the website data in a data as a service manner the second is in such a model the supplier handles all the programming and manual verification and provides clean data via an api or csv download build your own data pipeline you can rely on web scrapping software or rps in combination with proxy servers to build bots that acts in a manner that is not distinguishable from human. Now let us look at price optimization and demand forecasting with machine learning during a crisis.

In May 2023, the World Health Organization officially declared the end of the COVID-19 global public health emergency. However, back in 2020, we observed its direct impact on consumer spending. Businesses all over the world had to revise their annual sales forecasts and strategies. They were not operating in a business-as-usual manner, leading to a question: Are we still able to use machine learning to predict demand in completely unexpected scenarios like this?

The answer is yes, but with new things to consider. In a business-as-usual scenario, machine learning is likely to leverage historical data and correlated external data to bring insights such as seasonality, relevant sales data, and competitors' reactions. During a crisis, as the market is not behaving as usual, the historical insights are likely to fall short in predicting future sales. To fight back, we would need to increase the importance of shorter-term information, for example, daily sales, with the understanding that the recent past is much more suitable to predict the future.

Practically, this means adjusting the feature engineering process to weigh the shorter-term sales lags rather than the historical ones. Additionally, the demand forecasting problem will also require the incorporation of more real-time market data than before, as well as external macroeconomic and social data. On the real-time data, this means regularly updating available market data such as sales data, customer churn, sales intent (for example, added-to-cart items), traffic to competitors' sites, competitors' prices, among others. On the macroeconomic level, data such as consumer spending, unemployment, GDP, and even community mobility segmented by city or region could also be considered, although these are mostly reported on a monthly basis.

Stock market indicators, i.e., S&P, BSE Sensex, NSE, Nifty50 could potentially be considered as proxies for real-time macroeconomic trends. Finally, there might also be positive results from incorporating social data. If we take the pandemic as a use case, we would use reported COVID cases or current government policies, including lockdown durations, to generate scenarios, forecast, and consider them for modeling future demand. To summarize, in any abruptly new economic scenario,

Our analysis still concludes that machine learning would be greatly leveraged to build accurate demand forecasts and optimize pricing strategies. The key adaptation to a business-as-usual scenario would be to incorporate some real-time data, such as market and macroeconomic data. And adapt the model to consider near-term lags versus historical data. It is also worth noting that business understanding and human judgment will still play a key role in creating this solution. Next comes reinforcement learning and pricing.

In real-world scenarios, for price optimization projects in business, classical demand forecasting approaches provide an excellent starting point. In further steps, RL models can be built on top of an existing machine learning system. One of RL's most interesting

aspects is that it can learn without needing expert domain knowledge beyond the system rules. For example, some representation of the state of the environment

With available actions in each state and finally the rewards or penalties received when moving from a state to the next one. The link between the way we believe we think and this field of AI is clear. Behaviorism, which is the natural intelligence counterpart of RL theory, is not considered the alternative way to explain how the mind works. But it accounts for a large part of the advancements in psychological research. That similarity in the way people and animals learn from their environment seems to bias our common sense and make us believe that we have finally solved the problem of intelligence and that RL is now the only path to follow.

But should we allow this particular path to solve all our machine learning problems? Intense research pushes this field forward, and from that point of view, we are in a very sweet spot where computer science meets psychology, just like how behavioral economics evolved. Psychologists and economists learn to program in Python, and all that motivation is truly powerful, but it does not mean that we should throw away our previous tools. The great power of many of our analytical tools comes from the fact that they enable different ways to look at the problem. Sometimes these tools support, and other times they completely override our intuition. Combining tools, processing different visualizations, and leveraging the best-performing machinery in each specific domain brings us the best results if we do it right. We should ask ourselves what different visualizations,

components, and sub-problems will help us optimize the pricing policy. So, the first step is that a great pricing system is first a good forecasting system. In some cases, forecasting may be so inaccurate that it is not feasible for a pricing system, and thus a different approach is needed. The point is, when available, the best pricing system also provides forecasting and is based on it. Good forecasting yields pricing strategies.

It is because if you can predict how many sales you have, you will have for each product as a function of the price and you make it with good accuracy then you get the optimum price for this product. So, profit will be equal to predict to save sales prices into price minus cost. Now, let us look at the challenges in using reinforcement learning for pricing. However, that is not the case as there are many subtle things.

implicit behind the good forecasting requirement the first challenge is supply chain management what if we run out of stock in the middle of our predicted sales the second is

outlier predictions if you do not consider sales like black friday big billion day while predicting The third is product of cannibalization if you are trying to sell a package of two or more expensively than the two separate units. The fourth is exploration-exploitation trade-off. What if we are always selecting prices near the historically set prices? Maybe the system gets high forecasting accuracy for the prices that it sets live.

Still, we never discover that the rest of the predicted demand curve was wrong, underestimating sales for some prices that were never previously tested. If some new prices are never explored, there is no way to tell that the selected prices are the best. This dilemma is well known in the RL theory, but we will have to deal with it even if we don't use RL. The fifth is data requirement.

AI algorithms require large amounts of historical data to learn effective pricing policies, which may pose challenges for businesses with limited data availability or quality. Computational complexity: training RL models for dynamic pricing can be computationally intensive, requiring significant computational resources and expertise in machine learning techniques. Regulatory constraints: businesses must navigate regulatory constraints and ethical considerations when implementing pricing strategies, particularly regarding fairness and transparency in pricing decisions. Now, how to implement RL-based dynamic pricing? So, the first step here is problem formulation.

Markov Decision Process Formulation In dynamic pricing, the pricing problem can be represented as an MDP consisting of states (S), representing the current market conditions, which include factors such as demand, competitor pricing, time of day, seasonality, inventory levels, and customer characteristics. Then comes A: actions correspond to pricing decisions, such as setting the price for each product or service. The transition function (T) describes the probability of transitioning from one state to another based on the chosen action. And then comes the reward function (R), which provides feedback on the desirability of the state-action pairs, typically reflecting revenue or profit generated from each pricing decision.

The second step is model selection. Q-learning: a simple and widely used RL algorithm suitable for discrete action spaces. It iteratively updates the Q-values, which represent the expected future rewards for taking a particular action in a given state. Deep Q-networks: a variant of Q-learning that uses deep neural networks to approximate the Q-values, enabling the handling of large state spaces and continuous action spaces. Policy gradient

methods: directly learn the policy (the mapping from states to actions) without explicitly computing Q-values, offering flexibility in handling complex action spaces.

Stochastic policies. The third is feature engineering. So, here the first step is extraction. Extract relevant features from historical data. Features may include product attributes,

For example, category, brand, popularity, customer demographics (e.g., age, gender, location), competitor prices, time of day, day of the week, seasonality, promotions, and any other factors influencing purchasing decisions. Pre-process and normalize features. Scale and pre-process features to ensure they are in a consistent range and format suitable for training the RL model. The second step is reward design. So, define appropriate reward functions.

Rewards should align with business objectives, such as maximizing revenue, profits, or customer satisfaction, while considering long-term goals and constraints. Revenue-based rewards directly tie rewards to revenue generated from each pricing decision. Profit-based rewards incorporate cost considerations. For example, product cost, shipping cost, to optimize profit margins.

The next step is customer satisfaction rewards. Introduce rewards for customer satisfaction metrics. For example, repeat purchase rate and customer lifetime value to encourage customer-centric pricing strategies. The third step is service training and evaluation. It starts with data preparation.

Split historical data into training, validation, and test sets. Preprocess data and code features for input into the RL model. Train the RL model. Use the selected RL algorithm to train the pricing policy on historical data, iteratively updating the policy parameters to maximize cumulative rewards. Evaluation metrics:

Evaluate the performance of the trained model using metrics such as revenue, profits, customer satisfaction, and market share. Conduct simulations or A/B testing to assess the impact of the RL-based pricing strategy compared to baseline approaches from our competitors. The fourth step is deployment and monitoring. Deployment strategy: Deploy the trained RL-based pricing system in a production environment, integrating it with the e-commerce platform's pricing infrastructure. Continuous monitoring: Monitor the performance of the deployed model in real time, tracking key metrics such as revenue, profits, and customer satisfaction.

Adaptive learning – Incorporate mechanism for adaptive learning, allowing the model to continuously adapt to changing market dynamics, competitor behavior and customer preferences. Regular updates, schedule periodic updates and retraining of the RL model using new data to ensure that it remains effective and up-to-date with evolving market conditions. By following these implementation steps, businesses can effectively leverage reinforcement learning for dynamic pricing enabling them to optimize revenues, profits and customer satisfaction in a rapidly evolving landscape. We will learn about dynamic pricing in depth in the next modules.

So to conclude, in this module, we have first introduced AI algorithms and understood how pricing algorithms work. Then we have understood how machine learning influences pricing. Thereafter, we have understood the role of data in AI-assisted pricing. We have learned what is data scrapping and the difference between data scrapping versus data scrawling. And we have also learned the importance of data scrapping and how to use it for getting competitor pricing.

Then we have explored some of the important tools used for data scrapping and how to automate web scrapping with RPAs. We have discussed the benefits and challenges of RPA for web scrapping and how to automate RPA for scrapping. We have also discussed the use of machine learning in price optimization and forecasting in crisis situations. Then we have discussed the role of reinforcement learning in pricing and its challenges. Finally, we have learnt how to implement RL based dynamic pricing.

These are some of the sources from which the material for this module was taken. Thank you.