MSc Project - Reflective Essay

Project Title:	Supply Chain Optimisation Tool- Using Data-Driven Decision Models
Student Name:	Vishal Makode
Student Number:	200875978
Supervisor Name:	Dr Flynn Castles
Programme of Study:	MSc IOT (Data)

Supply Chain Optimisation Tool- Using Data-Driven Decision Models

Companies worldwide may now start to achieve previously unimaginable things especially in supply chain and logistics domain as technologies like digital twins, machine learning (ML), and the internet of things (IoT) continue to mature and spread. These cutting-edge technologies can help businesses build an intelligent supply chain that anticipates and tracks the implications and effects of almost every decision they make.

Figure 1: Generic supply chain diagram.

Historically, when businesses attempted to comprehend demand, they relied solely on their sales, i.e., what they had sold over what time frame. This was helpful but insufficient. With improved access to point-of-sale data from merchants and the introduction of syndicated market data, consumer-focused businesses have improved their understanding of their end customers. As the AI-powered solutions burgeoned, and organisations were able to use this information, a whole new world of data opened for businesses. This can help them understand what influences demand at increasingly granular levels and meet that demand more effectively, potentially predicting demand before customers know what they want. For instance, machine learning algorithms study demand patterns and forecast product categories a consumer will require at a specific business based on relevant data, enhancing customer pleasure and loyalty.

This tool (SCM tool) aims to facilitate the same; a few novel features - provide an intuitive dashboard to the user with intelligent logic running at the backend, a data pipeline to support real-time analytics and easy customisations for businesses.
However, before enabling this tool to make decisions for the supply chain manager, understanding the supply chain and, more importantly, parameters affecting the supply chain and logistics were necessary.

The initial technical and development aspirations with the software were to make it deployable and cross-platform while adhering to production-compliant practices and services, keeping it functional for real-world use. Preliminary research incorporating similar work suggested that this was an arduous task; hence some planning and project breakdown was vital.

The project was segregated into separate components, i.e., A) Research. B) Architecture diagram. C) Business logic and Machine Learning. D) Deployability and scalability. E) Dashboarding. F) Functionality testing and reflection. This breakdown gave accountability to each step and apprehension about what needed to be done considering the time frame.

Coming across conflicting studies and contradictory data in preliminary research, the reliability, and validity of datasets and research papers were questioned. Tremendous amount of time and research was required in data gathering, cleaning, and pondering on the insights to be extracted from the datasets. What helped was quickly plotting a correlation heat map of every dataset as shown in Fig 2 to get insight into the data and understand which features were necessary, duplicated, or unwanted, as shown in the image. This was a quick litmus test to analyse whether the data could be used functionally. A dataset of supply chains used by the company DataaCo Global was selected (Constante, Silva and Pereira, 2019). The dataset includes a collection of their products, financial details (profit, loss, total sales, etc.), shipping details, and customer details such as sales, demographics, and transaction details. The data spans 91 MB engulfing details of 180,520 customers spanning 53 columns related to clothing, sports, and electronic supplies.

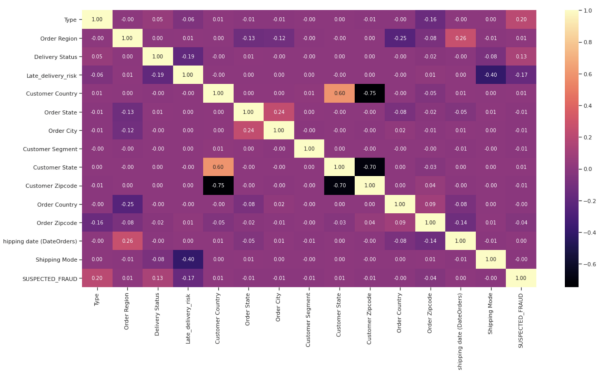

Figure 2: Exploratory data analysis by correlation plot.

As the program expanded, so did the accompanying code base, which caused the development environment to become overloaded each time the application was spawned, thus decreasing productivity. Suppose a single function or component of an application fails. In that case, the application is rendered inoperable—with distinct features such as a dashboard, business logic, database, and data parsing services. If a specific function began to use more processing power, the program's overall performance would suffer. The impact of a monolithic software design approach was felt during both the application development and deployment phases. It became crucial to divide the application into smaller components. Also, due to the interconnected nature of the monolithic approach, it has become harder to build or deploy modules independently. They must stay entirely reliant on others, increasing their overall development time. It was hard to manage multiple modules, so the microservice architecture approach made more sense. Microservices are a technique for decomposing big software projects into loosely linked modules or services

that interact over basic Application Programming Interfaces (APIs) and make software convenient to deploy over the cloud. To enable and exploit microservices architecture, services docker and Kubernetes were used.
Docker is an open platform for developing, shipping, and running applications. Docker enables separating applications from infrastructure so the software can be delivered quickly (Docker Documentation, 2020). With Docker, we can manage infrastructure like we manage applications. Docker provides the ability to package and run an application in a loosely isolated environment called a container. Isolation and security allow us to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so there is no need to rely on what is currently installed on the host (Docker Documentation, 2020). Kubernetes is a portable, extensible, open-source platform for managing containerised workloads and services that facilitates declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available and well documented. Containerising everything turned out to be a blessing for the development process, the software got segregated, and each module was responsible for a single task. There was some overhead to the containerisation as well, communication was one of them. For asynchronous messaging amongst the services, tool like RabbitMQ was used. RabbitMQ supports several messaging protocols, directly and through the use of plugins. The RabbitMQ Cluster Kubernetes Operator provides a consistent and easy way to deploy RabbitMQ clusters to Kubernetes and run them. RabbitMQ clusters deployed using the Operator can be used by applications running on Kubernetes or outside Kubernetes (www.rabbitmq.com, n.d.). I managed to get services up and running to start working on the business logic.

Conceptually, it was straightforward. The algorithm (i.e., the machine) learns associations from a training dataset (i.e., the one selected above) and may then apply these correlations to new data and throw out answers. However, understanding what questions to ask was essential to making the forecast. Per se, based on the last 'x' period of demand, what will the demand be during the following period(t)? As it happens, machine learning can generate very accurate predictions, given certain points are made clear: Which data to feed the algorithm so it understands the proper relation? Which machine algorithm to use as there are many different ones. Which parameters to use in our model? Each machine learning algorithm can be tweaked to improve its accuracy. How to score each algorithm? It was challenging to pick the data analysis method and machine learning model to employ, as the model's performance varied a lot depending on the factors present in the data. One of the significant flaws of this approach is that the business logic relies too much on data which is not ideal as the idea was to make it as general purpose as possible. As always, there was no definitive one-size-fits-all answer. Yet, some experiments were conducted to find the best one for the dataset.

For their performance, nine primary machine learning classifiers and seven regressors were evaluated against neural networks. Important supply chain-related variables were identified, and machine learning models were trained to detect future demand, fraudulent transactions, late order fulfilment, sales revenue, and the frequency of customer requests.

This project utilises Logistic Regression, Linear Discriminant Analysis, Extreme Gradient Boosting, Support Vector Machines, k-Nearest Neighbours, Random Forest classification, and Decision Tree classification to detect fraud and predict late delivery based on accuracy and recall score.

The idea is that organisations will be able to uncover hidden trends and make better decisions in the future. For this project, for instance, there are some fascinating insights, like Western Europe and Central America are the regions with the highest sales. The company also lost most of its revenue from these regions only—furthermore, these regions had the highest number of fraudulent transactions with the most delayed deliveries. The company should be careful when customers are using wire transfers as the company was scammed with more than 100k by a single customer. All the orders with the risk of late delivery are delivered late every time. Most orders with Men's Footwear and Women's Apparel category products are causing late delivery. These products are also suspected of fraud the most. The neural network classifier model trained for fraud detection outperformed all machine learning classifier models. Compared with other classification machine learning models, the Decision Tree model did an excellent job identifying orders with later delivery and detecting fraudulent transactions.
For further study, machine learning models can be compared with diverse datasets to confirm whether the models are performing reasonably better or not. Moreover, the performance of these machine learning models can be improved with hyperparameter tuning.

We have already tackled a good chunk of work at this stage, including data collection, cleaning, consolidation of several data sources, applying learning algorithms, and creating a mix of meaningful metrics. Now it was time to make a user-friendly dashboard which aids in the decision-making process of the user, which turned out to be surprisingly hard. Research tells me to adhere to particular dashboard design “best practices” to present data optimally, making it simple to assess and act upon. Some preliminary requirements with the dashboard were: Ensure that the user always looks at the latest information by having dashboards interact directly in real-time with the source data. They are customised for business and the specific driving of the organisation. Accessible via mobile devices. And most crucial, it helps to drive decisions. I prioritised simplicity and went for a clean layout design, with the right kind of charts as shown in Fig 3, hoping that the dashboard would answer

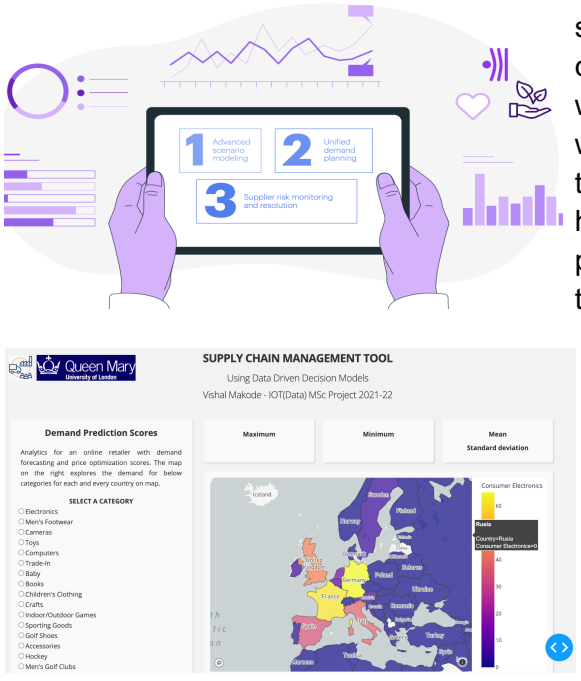

Figure 3: Actual dashboard from SCM tool.

questions like- For a particular product category, which region of the map is the demand going to grow? For a particular product category, which region will have more lead time? What are the correlations between multiple supply chain parameters? For example, shipping date is positively correlated (0.58) with order region, and shipping date is negatively correlated (-0.75) with late delivery risk. Given the time frame and my lack of user-interface design experience, the dashboard layout was designed to plot answers to some of the above questions.

Reflection on this experience was enlightening. It was realised that a more robust approach to picking appropriate journals and research papers should be adopted. Given the time frame, one crucial thing I skipped was, having an in-depth technical discussion with managers and data scientists working in the supply chain domain. There are several application categories where SCM-tool may be employed. Demand, supply, on-time delivery, and frauds may all be predicted or anticipated using machine learning and the business logic of SCM-tool. Many essential supply chain operation components like lead time and demand prediction may be automated, which can also assist in identifying or anticipating deviations from regular operations.
Nevertheless, it was unclear what the requirements should be, making it more challenging to gauge the learning pipeline and, more importantly, which input features and output the industry uses and demands. The lack of open data was also a hindrance. Most of the datasets available online were incomprehensive and fudged.
Furthermore, as mentioned above, the business logic relied too much on the data. The dataset obtained in the end belonged to a Spanish company; hence a tremendous amount of time was spent cleaning, formatting, catering to all language discrepancies and engineering that data to make it usable. This problem could have been addressed by approaching industries and requesting legit data.
Initial intentions with this project about following the "production compliant" software approach were hugely successful as it prepared me with the software and tools used in the industry like docker, Kubernetes, git versioning, cloud architecture, data engineering, and jupyter notebook, to name a few. The code base with an architecture diagram was pushed to a git repo from the get-go. Designing a scalable and robust cloud architecture was vital as each application is unique and will have a custom set of requirements. Multiple real-world examples of architecture diagrams were studied to use as base reference architecture and customise accordingly to meet SCM tools requirements. The diagrams demonstrate concepts such as multiple services, data pipelines, or multi-cloud deployments. Using the cloud has drawbacks like, increasing the application complexity tenfold and making it sluggish. Many optimisations, like reducing the frequency of API calls and other software tweaks, could have been done if the time permitted.
The project provided a different perspective, a context for academic learning, and an opportunity to put theory into practice which was an incredible learning experience. I am grateful to have this working set up and excited to continue developing this platform further to a product standard or to a point where it makes a positive difference.

References –

[1] Caballero, S. and Rice, J. (2018). *Artificial Intelligence/Machine Learning + Supply Chain Planning Summary Report*. [online] Available at: https://ctl.mit.edu/sites/ctl.mit.edu/files/2020-07/AI_Machine_Learning_Supply_Chain_Planning_MIT_CTL_Nov_18_RT.pdf [Accessed 5 Dec. 2021].

[2] Supply chain analytics and AI in driving relevance, resilience and responsibility. (n.d.). [online] Available at: https://www.accenture.com/_acnmedia/PDF-163/Accenture-Supply-Chain-AI.pdf [Accessed 13 Aug. 2022].

[3] Constante, Fabian; Silva, Fernando; Pereira, António (2019), “DataCo SMART SUPPLY CHAIN FOR BIG DATA ANALYSIS”, Mendeley Data, V5, doi: 10.17632/8gx2fvg2k6.5

[4] www.rabbitmq.com. (n.d.). *RabbitMQ Cluster Operator for Kubernetes — RabbitMQ*. [online] Available at: https://www.rabbitmq.com/kubernetes/operator/operator-overview.html [Accessed 18 Aug. 2022].

[5] Documentation. (2020). *Docker overview*. [online] Available at: https://docs.docker.com/get-started/overview/#:~:text=Docker%20is%20an%20open%20platfom