[SCM Book Review] Data Science For Supply Chain Forecasting, tác giả Nicolas Vandeput, xuất bản bởi Walter de Gruyter GmbH, năm 2021

Trước khi bàn về phần nội dung, mình và các bạn cũng nên dành chút ít thời gian nghía qua phần bìa sách. Người ta hay bảo "Don't judge a book by its cover" nhưng nếu bỏ qua quả bìa này thì sẽ là một thiếu sót lớn. Bìa của Data Science For Supply Chain Forecasting là bức tranh "The School of Athens", được vẽ trong khoảng từ năm 1509 đến 1511, thời kỳ Renaissance (Phục Hưng), bởi danh họa Raphael Sanzio da Urbino với "purchase order" từ Giáo hoàng Julius II. Hiện tại bức tranh này đang được trưng bày tại Apostolic Palace thành Vatican.

Nhìn qua bức tranh, có thể thấy bối cảnh của nó được đặt tại một trường triết học Hy Lạp cổ đại với vô số các triết gia nổi tiếng như Euclid (người viết ra tiên đề "với một điểm nằm ngoài một đường thẳng, chỉ có thể vẽ một đường thẳng song song với đường thẳng đó" mà ta được học vào lớp 7), Pythagoras (với định lý "bình phương cạnh huyền bằng tổng bình phương hai cạnh góc vuông" và bộ môn Thần Số Học nổi tiếng với Gen Z ngày nay), Zeno (khởi xướng trường phái Stoicism/Khắc kỷ), Socrates (với danh hiệu triết gia khôn ngoan nhất thành Athen), v.v. Về cơ bản, bức tranh mô tả cuộc tranh luận xuyên suốt nhiều năm trong triết học cổ điển phương Tây giữa hai trường phái Duy Tâm và Duy Vật, với đại diện là Aristotle và Plato được đặt ở trung tâm. Dẫn sang câu chuyện về supply chain, ắt hẳn các bạn cũng đã từng nghe đâu đó rằng demand planning là sự kết hợp hài hòa giữa hai yếu tố Art (dùng business sense để đánh giá tình hình thị trường, đưa ra chiến lược) và Science (dùng số liệu phân tích để dự báo và hoạch định vận hành). Có thể hiểu sự liên kết về ý nghĩa của demand planning và thông điệp triết học nằm sau chiếc bìa sách là như thế. Thực ra, trong bìa sách vẫn còn một nhân vật thú vị nữa – cũng nằm ngay giữa bức tranh, dưới chân của Aristotle và Plato – là Diogenes. Ông Diogenes này có triết lý về con chó rất "dị", nếu có thời gian các bạn nên tìm hiểu thêm.

Xong phần bìa sách, giờ ta đi vào nội dung chính. Tổng quan, Data Science For Supply Chain Forecasting được viết ra nhằm mang đến cho đối tượng độc giả là những nhà nghiên cứu cũng như những cá nhân đang trực tiếp làm việc tại các doanh nghiệp một cái nhìn "số má" hơn về demand planning nói chung và demand forecasting nói riêng. Cũng bởi vì sách được viết hướng đến đối tượng đã có nền tảng sẵn nên tác giả đã bỏ hoàn toàn những khái niệm cơ bản mà tập trung vào cách ứng dụng khoa học dữ liệu vào thực tiễn.

Một điểm đặc biệt của của cuốn sách là ngoài những cơ sở lý thuyết, logic và công thức toán học, tác giả còn hướng dẫn cách triển khai các forecasting model trên hai nền tảng Excel và Python, qua đó giúp người đọc nâng cao kỹ năng lập trình và tận mắt chứng kiến cách mà các hệ thống demand forecasting ngoài thị trường đang hoạt động. Mình đã chạy toàn bộ code có trong sách và upload tại GitHub: nicolas-vandeput-books-

review/data-science-for-supply-chain-forecasting at main · thanhtranviet248/nicolas-vandeput-books-review . Khi xem, các bạn lưu ý rằng code của mình không giống hoàn toàn tác giả vì sẽ có một số trường hợp mình làm cho gọn hơn, giải thích cho dễ hiểu hơn, hay cập nhật syntax cho phù hợp với version Python và các package hiện tại. Mọi người xem nếu thấy bổ ích thì cho mình xin 1 follow và 1 sao cho repository này nhé.

Về cấu trúc, sách được chia ra làm 3 phần: Statistical Forecasting (Dự báo sử dụng các phương pháp thống kê); Machine Learning (Dự báo sử dụng các thuật toán học máy); và Data-Driven Forecasting Process Management (Dự báo định tính và quản lý giá trị gia tăng trong quy trình dự báo). Dưới đây là tóm tắt nội dung và một vài đánh giá của mình cho từng phần của cuốn sách.

Phần 1: Statistical Forecasting

Chương 1: Moving Average

Về Moving Average thì mình cho rằng model này quá đơn giản rồi nên cũng không có nhiều điều để nói. Mở đầu chương, tác giả trình bày công thức tính của Naïve (Đơn Giản) và Moving Average (Trung Bình Động) rồi sau đó chỉ ra là chúng không thể sử dụng trong các trường hợp demand pattern có tính seasonal hay trend, cũng như trọng số được phân bổ đều cho các mốc thời gian được đưa vào model (ví dụ với Naïve là 100% và Moving Average là 33.33% nếu lấy trung bình 3 tháng). Cuối cùng, model được demo nhẹ nhàng trên Excel và Python, chắc là mới mở đầu nên tác giả chưa muốn hardcore với người đọc.

Chương 2: Forecasting KPI

Với cá nhân mình, chương 2 là là một trong những chương thú vị nhất của cuốn sách. Đầu tiên, tác giả Nicolas liệt kê các forecasting KPI quen thuộc như Error, Bias, MAPE, MAE, RMSE cùng công thức tính của chúng. Sau đó, anh đem những chỉ số này so sánh với nhau, chỉ ra đối với dạng demand pattern và dữ liệu ra sao thì nên sử dụng cái nào cho phù hợp. Ý nghĩa công thức tính từng KPI sẽ được đào sâu, đối với KPI dạng phần trăm thì phải lưu ý điều gì hay với KPI dạng số tuyệt đối thì sao. Phần này tác giả viết rất kỹ nên sẽ có nhiều thông tin giá trị, khi đọc các bạn nên tập trung vì nó sẽ có nhiều ảnh hưởng đến toàn bộ cuốn sách về sau khi chúng ta đụng vào những forecasting model phức tạp hơn.

Chương 3: Exponential Smoothing, Chương 5: Double Exponential Smoothing, Chương 7: Double Smoothing With Damped Trend, Chương 9: Triple Exponential Smoothing, Chương 11: Triple Additive Exponential Smoothing

Tuy các chương 3, 5, 7, 9 và 11 đều viết về Exponential Smoothing (San Bằng Hàm Mũ) nhưng độ khó sẽ tăng dần qua từng chương. Tác giả bắt đầu chương 3 với ý nghĩa và công thức của Simple Exponential Smoothing cùng tham số alpha. Điểm yếu của

Simple Exponential Smoothing là chỉ làm việc được khi demand pattern không có tính trend và seasonal cũng được chỉ ra.

Sang chương 5, yếu tố trend cùng tham số beta được thêm vào để tạo ra Double Exponential Smoothing. Nhưng đến đây mọi thứ vẫn chưa tốt hơn là mấy vì theo phương pháp này tính xu hướng sẽ kéo dài mãi mãi, từ đó dẫn đến sự over-estimate trend trong một quãng thời gian dài.

Sang đến chương 7, tham số phi tham gia vào model để hình thành Double Exponential Smoothing With Damped Trend với mục đích giảm đi tính xu hướng theo thời gian. Tuy nhiên, khả năng ghi nhận tính seasonal vẫn chưa được giải quyết.

Tiếp tục với chương 9, yếu tố seasonal và tham số gamma đã có mặt để tạo ra Triple Exponential Smoothing (hay có tên gọi khác là Holt-Winter). Lúc này, mọi vấn đề của model sơ khởi Simple Exponential Smoothing đã được giải quyết khi model đã có khả năng ghi nhận các thành phần level, trend và seasonal.

Những tưởng mọi thứ đã hoàn thiện, nhưng không, Nicolas cho rằng nếu chúng ta đưa yếu tố seasonal vào model dưới dạng multiplicative (phép nhân) thì vẫn sẽ có những trường hợp khó mà ứng dụng được. Ví dụ với Intermittent Products, nhu cầu sản phẩm tại những thời điểm sẽ giảm xuống tiệm cận hoặc bằng 0, với độ biến động lớn của demand như thế sẽ gây ảnh hưởng lên khả năng ghi nhận tính seasonal của model. Vậy để giải quyết, chúng ta đến với chương 11 và model Triple Additive Exponential Smoothing. Với Triple Additive Exponential Smoothing, seasonal index không được nhân vào nữa mà thay vào đó là cộng. Ta biến Forecast = (Level + Trend)*Seasonal trở thành Forecast = Level + Trend + Seasonal.

Cuối chương, tác giả nhấn mạnh một điều rằng không có model nào là hoàn hảo mà người làm forecast phải dựa vào dữ liệu, vào demand pattern, rồi kiểm thử nhiều lần để biết đâu là model phù hợp nhất.

Chương 4: Underfitting và Chương 8: Overfitting

Tại chương 4 và chương 8, 2 khái niệm overfitting và underfitting được đưa vào thảo luận. Các bạn cũng cần phải chú ý 2 chương này vì overfitting và underfitting sẽ được nhắc đến rất nhiều ở phần sau của cuốn sách. Cụ thể, underfitting nghĩa là model không “describe fully the reality” hoặc có thể hiểu đơn giản là ta không thể lượng hóa hết được toàn bộ các tác nhân tác động đến demand có trong thực tế, từ đó làm giảm forecast accuracy. Ngược lại, với overfitting, model sẽ ghi nhận một cách vô cùng chi tiết các dữ liệu, bao gồm cả các điểm random, noise, hay outlier, từ đó cho ra kết quả tốt trên bộ dữ liệu quá khứ hay train set, nhưng khi kiểm thử trên test set thì kết quả sẽ không đạt như mong đợi.

Chương 10: Outliers

Đây là chương bổ trợ cho hai chương 4 và 8 bên trên, nội dung sẽ bàn về về cách xử lý các điểm outliers. Có 3 cách được nêu, bao gồm: Winsorization (loại bỏ các điểm quan sát cao nhất và thấp nhất dựa theo percentile); Standard Deviation (sử dụng độ lệch chuẩn của demand để loại bỏ các outliers); và Error Standard Deviation (đây là phương pháp mà mình thấy hay nhất vì nó có khả năng phản ánh được năng lực của forecasting model khi sử dụng độ lệch chuẩn của Error để loại outliers thay vì dựa hoàn toàn vào độ biến động của demand).

Chương 6: Model Optimization

Đây cũng là một chương hay, nhưng chưa "tới". Tại chương 6, tác giả sử dụng Excel Solver để thiết lập optimization model và chạy vòng lặp trên Python nhằm tìm ra các giá trị alpha, beta, phi, gamma, v.v. sao cho sai số dự báo đo bằng RMSE đạt cực tiểu. Đối với mình thì chương này có hơi "cưỡi ngựa xem hoa" vì tác giả không đi sâu vào nguyên lý thiết lập optimization model mà chỉ demo công cụ thôi (thực ra cũng chấp nhận được vì đây là sách forecasting chứ không phải operations research).

Phần 2: Machine Learning

Chương 12: Machine Learning

Sang chương 12, Machine Learning (ML) được giới thiệu như là một công cụ giúp ta dễ dàng đạt được kết quả dự báo vượt trội – trích "easily achieve outstanding performance" – với lời hứa hẹn rằng cứ tiếp nối từng chương, các model sẽ ngày càng manh mẽ hơn.

ML khác với các statistical model ở chỗ, từng thuật toán ML sẽ có từng cách tiếp cận riêng và sử dụng các cách tiếp cận đó để học từ dataset rồi thiết lập các mối quan hệ giữa các biến thay vì sử dụng một công thức với các mối quan hệ được xác định sẵn bởi con người. Ví dụ, ML sẽ không tự cho rằng có sự tồn tại sẵn của yếu tố seasonal hay trend mà thay vào đó sẽ tự tìm ra chúng trong quá trình học. Một điểm hay khác của ML là nó sẽ học trên toàn bộ dataset thay vì chỉ dựa vào một vùng data nhỏ như các statistical model, từ đó giúp cho kết quả đầu ra khách quan hơn.

Dataset norway_new_car_sales_by_make.csv (đã được upload trên GitHub của mình) sẽ được sử dụng để thực hành từ giờ đến hết cuốn sách. Tại cuối chương 12, tác giả hướng dẫn người đọc import dữ liệu vào Python, xử lý để tạo thành định dạng mong muốn, rồi sau đó chia bộ dữ liệu thành hai phần là train set và test set. Cuối cùng là fit chúng vào model đơn giản nhất của ML là Linear Regression (Hồi Quy Tuyến Tính) và tính ra forecast accuracy (MAE, RMSE) với mục đích dùng làm benchmark cho các model sau này (tại đây, ý của anh Nicolas là cho dù model phức tạp đến mấy mà không thắng được Linear Regression thì cũng đem vứt).

Chương 13: Tree

Mô hình Tree hay Decision Tree (Cây Quyết Định) bao gồm hai dạng là Regression và Classification nhưng trong cuốn sách này ta sẽ chỉ tập trung vào Regression Tree để giải quyết vấn đề demand forecasting thôi.

Các thành phần của một tree như node (điểm mà tree sẽ rẽ nhánh bằng một câu hỏi Yes/No), leaf (các kết quả đầu ra thuộc hai nhóm Yes/No) và feature (các trường dữ liệu mà node dùng để đặt ra câu hỏi Yes/No) cũng được giải thích trực quan sinh động bằng trò chơi "Guess Who?". Tiếp đó, tác giả minh họa cách một tree được phát triển dựa trên dataset norway_new_car_sales_by_make.csv và giải thích các parameter trong package scikit-learn như max_depth, min_samples_split, min_samples_leaf rồi cuối cùng là chạy demo trên Python. Khi có kết quả đầu ra, hai chỉ số MAE và RMSE được đem ra thảo luận vì chúng là thành phần của parameter criterion (parameter giúp cho model biết được rằng cần phải dùng feature nào và split point là bao nhiêu trong quá trình train). Kết quả so sánh cho thấy, khi criterion là MAE thì thời gian train sẽ lâu hơn (gấp MSE 40 lần nếu sử dụng dataset hiện tại và có thể sẽ lâu hơn nữa nếu dataset dài hơn). Một điểm thú vị khác là kể cả khi ta train model dựa trên MAE thì khi kiểm trên test set thì kết quả RMSE lại tốt hơn MAE. Nghe hơi lạ nhưng điều này không quá hiếm gặp vì các parameter hiện tại chưa được tối ưu, và một cách tình cờ nào đó, dataset lại phù hợp hơn với MSE so với MAE.

Chương 14: Parameter Optimization

Khi setup parameter cho một model, làm sao để chúng ta biết các parameter đó đã tối ưu hay chưa, ví dụ max_depth của một tree nên là 5 hay 6? Tác giả đã chạy thực nghiệm vòng lặp mô hình tree với max_depth đi từ 1 đến 14, kết quả cho thấy max_depth càng cao thì mức độ chính xác của model trên train set càng cao (ừ thì do overfitting đó). Còn với test set thì mức độ chính xác sẽ tốt nhất với max_depth bằng 6. Vậy thì ta kết luận luôn được không? Đương nhiên là không vì nếu parameter được tối ưu dựa vào kết quả của test set không khác gì mô hình overfitting với test set. Lúc này, 2 phương án tối ưu parameter nhưng hạn chế overfitting được đề xuất là K-Fold Cross-Validation và Random Search.

Với K-Fold Cross-Validation, ta sẽ chia train set thành các fold nhỏ hơn rồi lần lượt train model dựa vào các fold đó. Ví dụ với 100 hàng dữ liệu thuộc train set, ta chia nhỏ ra thành 5 folds (4 fold train và 1 fold test) với mỗi fold gồm 20 hàng dữ liệu. Sau đó lần lượt chạy model cho từng tập hợp rồi tính trung bình kết quả để ra output cuối cùng. Lợi ích của K-Fold Cross-Validation là sẽ giúp ta thử được nhiều parameter và chọn được parameter tối ưu nhất nhưng vẫn hạn chế được overfitting vì phương pháp này dựa trên train set được chia nhỏ, còn test set vẫn sẽ được giữ lại để kiểm định.

Đi cùng với K-Fold Cross-Validation là các phương pháp search nhằm trả lời cho câu hỏi "Những parameters nào nên được chọn để tìm ra điểm tối ưu (với K-Fold)?" vì ta không thể kiểm định toàn bộ parameter với hạn chế trong thời gian tính toán. Có 2 phương

pháp search được đề cập, là Grid Search (Uniform Search) và Random Search. Với Grid Search, ta trải đều range của các parameter ra, ví dụ max_depth từ 5 đến 15 và min_samples_split từ 2 đến 12. Với mỗi parameter có 11 option như vậy thì tổng cộng có 121 tổ hợp model. Tuy nhiên, để chạy hết 121 model thì cũng rất tốn tài nguyên nên ta có thể chọn chạy một phần nhỏ, ví dụ 25 model, vậy số parameter có thể được test là 5 option của max_depth và 5 option của min_samples_split. Chuyển sang cách search thứ hai – Random Search – ta sẽ không search trải đều nữa mà sẽ làm nó một cách ngẫu nhiên, từ đó với 25 model được test thì ta có thể scan qua được 10 option của max_depth và 9 option của min_samples_split.

Với phần thực hành, sau khi có được bộ parameter tối ưu với K-Fold Cross-Validation cùng Random Search và sử dụng parameter đó để kiểm thử trên test set thì kết quả đã được cải thiện từ 21.1% đến 18.9% với MAE. Nói thêm, để tiếp tục giảm overfitting, ta nên tăng thêm search space cho parameter và tăng số lượng fold, điều này vô hình trung sẽ làm tăng thời gian tính toán.

Chương 15: Forest

Mô hình Forest hay Random Forest (Rừng Ngẫu Nhiên) có thể được tóm tắt bằng câu tục ngữ "Một cây làm chẳng nên non, ba cây chụm lại nên hòn núi cao". Forest thuộc nhóm ensemble bagging model (một mô hình lớn được cấu thành bởi nhiều mô hình con chạy song song với kết quả cuối cùng là trung bình kết quả của các mô hình con). Ý tưởng của Forest chính là cho chạy nhiều Tree đồng thời với mỗi tree sẽ khác nhau một chút về parameter, các điểm dữ liệu của train set (thực hiện bằng parameter bootstrapping và max_samples) và số lượng feature (với parameter max_features). Sau khi chạy demo model Forest, ta cũng có thể tối ưu hóa model bằng Random Search và K-Fold Cross-Validation cùng với parameter n_estimators giúp tăng số lượng tree có trong forest.

Chương 16: Feature Importance

Chương 16 nhắc lại về khả năng interpretable của các statistical model vì chúng cho ta thấy ngay mối quan hệ giữa các biến có trong model. Điển hình là Exponential Smoothing, ta có thể phân tích demand trên ba yếu tố level, trend và seasonal theo kiểu "demand này có tính seasonal không?", "demand này có tính trend không?". Và nếu kết quả forecast có vấn đề thì ta có thể lật lại các yếu tố này để đánh giá xem nguyên nhân đến từ đâu. Ngược lại, các ML model thì không được như thế – chúng rất khó để diễn giải – ta không thể biết được thông qua kết quả của Forest rằng liệu demand của chúng ta có tính seasonal hay trend hay không. Tuy nhiên, có một công cụ có thể hỗ trợ ta hiểu được mức độ tác động của các feature lên mô hình: Feature Importance.

Với dataset norway_new_car_sales_by_make.csv, ta dự báo demand tháng tiếp theo bằng 12 feature là demand của 12 tháng trước đó. Câu hỏi đặt ra là "Demand tháng nào trong 12 tháng trước có vai trò quan trọng hơn trong việc xác định demand của tháng tiếp theo?". Với tính năng Feature Importance trên scikit-learn, mức độ quan trọng của

từng feature sẽ được đo bằng khả năng đóng góp của feature đó cho forecast accuracy (tối thiểu hóa MAE hoặc RMSE). Feature Importance là một tính năng hữu ích giúp ngăn chặn overfitting vì ta biết những feature nào là quan trọng và feature nào không, và với feature nào không quan trọng thì sẽ bị loại ra khỏi model để giảm noise.

Chương 17: Extremely Randomized Trees

Model Extremely Randomized Trees hay Extra Trees được ra đời năm 2006 bởi các nhà khoa học người Bỉ là Pierre Geurts, Damien Ernst, và Louis Wehenkel. Model này giống gần như hoàn toàn với Random Forest, điểm khác biệt là Extremely Randomized Trees khi phân nhánh thì sẽ không chủ động chọn feature và split point với mục tiêu tối ưu criterion MAE hay MSE. Thay vào đó, split point được chọn ngẫu nhiên tại từng feature rồi chọn ra feature/split option đem lại kết quả tốt nhất trong các kết quả ngẫu nhiên ấy. Cũng bởi vì không cố gắng tối ưu criterion nên Extremely Randomized Trees train sẽ nhanh hơn nhiều so với Random Forest. Ở phần thực hành, Nicolas cũng cho chạy demo model rồi sử dụng K-Fold Cross-Validation cùng Random Search để tìm ra bộ parameter tối ưu.

Chương 18: Feature Optimization #1

Tính đến chương 18, các model ML như Linear Regression, Decision Tree, Random Forest và Extremely Random Trees đã được đề cập cùng với cách tối ưu parameter của chúng thông qua K-Fold Cross-Validation và Random Search. Cơ mà vẫn còn một thứ mà chúng ta vẫn chưa tối ưu – các feature được sử dụng để train model. Vấn đề mà chương 18 đặt ra là "Bao nhiêu feature và những feature nào nên được sử dụng để train model để có forecast accuracy tốt nhất?". Để trả lời câu hỏi này, tác giả đã thực nghiệm bằng cách chạy model nhiều lần.

Ý tưởng của thực nghiệm là train model với số lượng feature range từ 6 đến 50 (cho chạy vòng lặp, với lần đầu chỉ dùng demand của 6 tháng trước để forecast cho đến vòng lặp cuối là dùng demand của 50 tháng trước để forecast nhằm quan sát xem với số lượng feature là bao nhiêu thì MAE sẽ đạt giá trị cực tiểu). Kết quả của lần chạy đầu tiên cho thấy, với Random Forest, MAE train set sẽ nhỏ nhất với số feature là 40 và test set là 30. Với Extremely Random Trees thì lần lượt là 40 và 12. Từ output này, ta có thể thấy được rằng với càng nhiều feature được đưa vào train thì forecast accuracy trên train set sẽ tăng (vì model ghi nhận nhiều noise và số lượng train loop ít đi dẫn đến overfitting). Và ta cũng không thể optimize model dựa trên kết quả trên test set vì nó cũng dẫn đến model overfitting với test set.

Với lần thực nghiệm thứ hai, train set sẽ được chia thành các fold và validation set. Kết quả thu về thì cũng khá giống lần chạy đầu nhưng tốt hơn ở điểm là pattern MAE của validation set và test set lần này tương tự nhau chứ không đôi đường đôi ngã. Một số điểm cần lưu ý là nếu số lượng fold càng nhiều thì mức độ biến thiên ngẫu nhiên của MAE cũng sẽ giảm đi, nhưng ta phải bù lại bằng thời gian tính toán và vẫn sẽ có khả năng

dẫn mô hình bị overfitting khi số lượng feature được đưa vào train ngày càng tăng thêm sau mỗi vòng lặp.

Với lần thực nghiệm cuối cùng, holdout set được đưa vào sử dụng. Chức năng của holdout set cũng tương tự như test set – được giữ không cho model chạm vào trong quá trình train – để rồi đến khi chạy xong toàn bộ vòng lặp, ta sẽ kiểm tra xem số feature là bao nhiêu để forecast accuracy trên holdout set là thấp nhất và so sánh pattern MAE giữa holdout set và test set xem có tương đồng với nhau hay không. Tin vui là có, số lượng feature để đạt được MAE thấp nhất cho holdout set với Forest là 28 và với Extremely Random Trees là 26 – cải thiện đáng kể so với con số 40 ban đầu.

Thế tóm lại thì bao nhiêu feature (hay demand của bao nhiêu tháng) nên được đưa vào để train model? Tác giả bảo rằng cứ chạy thực nghiệm đi rồi chọn ra cho phù hợp chứ không có câu trả lời đúng sai trắng đen rõ ràng đâu – trích "you will unfortunately never select a perfect model". Còn theo mình nghĩ thì nó có thể là số lượng feature sao cho kết quả forecast accuracy trên holdout set là tốt nhất và pattern của forecast accuracy trên holdout và test set tương tự nhau (không overfitting). Sau khi có được số feature "đẹp" rồi ta lại tiếp tục phân tích Feature Importance và chốt con số cuối cùng.

Chương 19: Adaptive Boosting

Câu hỏi đặt ra bởi Michael Kearns và Leslie Valiant trong những năm cuối thập niên 80 của thế kỉ trước "Can a set of weak learners create a single strong learner?" đã đặt nền móng cho một dạng ensemble model mới, thứ đã được Yoav Freund và Robert E. Schapire công bố vào năm 1997, gọi là boosting. Cách vận hành của boosting là các model con sẽ được thêm vào theo thứ tự, từ đó các model vào sau sẽ có thể sửa lỗi cho các model trước nhằm tăng độ chính xác của dự báo.

Ý tưởng của Freund và Schapire là chạy trước một weak model (ví dụ Tree) với trọng số cho các điểm dữ liệu trong train set là ngang bằng nhau, sau khi fit model thì tính ra prediction error và model confidence rồi lại chạy tiếp một weak model thứ hai với train set được điều chỉnh trọng số. Quá trình này lặp đi lặp lại cho đến khi đạt đến số lượng weak model tối đa (xác định thông qua parameter n_estimators) và kết quả cuối cùng được tổng hợp bằng cách lấy trung bình có trọng số kết quả của toàn bộ weak model. Cách chạy model này gọi là Adaptive Boosting hay AdaBoost.

Các phần còn lại của chương 19 như giải thích cụ thể từng parameter, chạy demo trên Python với scikit-learn hay tối ưu hóa parameter với K-Fold Cross-Validation và Random Search thì sẽ tương tự những chương trước. Thực ra cũng có một điểm cần lưu ý đó là tốc độ train của AdaBoost sẽ lâu hơn nhiều khi so sánh với các model bagging, tiếp theo, cũng như là vì lý do kỹ thuật nên AdaBoost chỉ có thể cho ra duy nhất một kết quả forecast tại một thời điểm, nếu ta muốn cho ra nhiều kết quả hơn thì phải dùng MultiOutputRegressor.

Chương 20: Demand Drivers and Leading Indicators

Từ đầu cuốn sách cho đến hiện tại, ta chỉ mới làm forecast theo dạng autoregressive (tự hồi quy) bằng cách dùng các endogenous factor (yếu tố nội sinh), nghĩa là dùng demand quá khứ để forecast demand tương lai, còn câu chuyện đưa một exogeneous factor (yếu tố ngoại vi) vào model thì chưa. Sự ảnh hưởng của các yếu tố ngoại vi này lên kết quả forecast có thể được ví dụ bằng câu hỏi "Do customers buy ice cream on Friday if they know that it will be a sunny weekend?". Chương 20 sẽ đề cập đến vấn đề này.

Nói về tác động của các yếu tố ngoại vi, tác giả ví dụ bằng sự tương quan giữa GDP và sản lượng xe bán ra hàng năm tại Norway. Kết quả cho thấy sự tỉ lệ thuận giữa hai biến này, theo biểu đồ chuỗi thời gian, khi GDP tăng thì sản lượng xe bán ra tại Norway cũng tăng và ngược lại. Tuy nhiên, vẫn tồn tại những vấn đề liên quan đến việc sử dụng demand driver khi forecast là ta phải biết giá trị tương lai của các demand driver đó, ví dụ, planning horizon cho tiệm kem của bạn là 1 tháng, vậy bạn phải forecast trước số lượng kem bán ra trong 30 ngày tới (với demand driver là thời tiết), điều này dẫn đến việc bạn cũng sẽ phải có thông tin dự báo thời tiết chính xác trong 30 ngày. Rõ ràng sự "double forecasting" này rất là tricky. Tác giả chốt lại phần lý thuyết chương 20 bằng việc gợi ý cho người đọc rằng tùy thuộc vào planning horizon mà một số leading indicator có thể hữu dụng hơn so với những leading indicator khác. Bạn có thể kiểm tra các leading indicator khác nhau dựa trên data availability và forecast accuracy tương ứng để chọn ra leading indicator thích hợp.

Chương 21: Extreme Gradient Boosting

Năm 2001, Jerome H. Friedman đề xuất ra một ý tưởng mới cho các mô hình boost trees, gọi là Gradient Boosting. Hướng tiếp cận của Gradient Boosting và Adaptive Boosting nhìn chung sẽ giống nhau, chúng đều là ensemble boosting model với cách vận hành là thêm lần lượt các model con vào nhằm cải thiện kết quả dự báo đầu ra. Điểm khác biệt chính của Gradient Boosting là các model mới sẽ được thêm vào để cải thiện phần residual errors của model trước đó thay vì cập nhật trọng số của data train set như AdaBoost.

Cải tiến dần theo thời gian, đến năm 2016, Chen và Guestrin từ University of Washington đã cho trình làng một thuật toán mới dựa trên Gradient Boosting, gọi là Extreme Gradient Boosting (hay XGBoost). Kể từ ngày ấy, XGBoost đã được sử dụng rất rộng rãi và mang lại nhiều kết quả đáng nể. Để so sánh XGBoost với Gradient Boosting và AdaBoost, Nicolas khẳng định ba điều. Một, XGBoost nhanh hơn. Hai, XGBoost nhìn chung cho ra kết quả tốt hơn. Ba, XGBoost cho phép có nhiều parameters hơn để optimize.

Đến phần thực hành, ta sẽ đi qua lần lượt các đoạn code Python để chạy model XGBoost, phân tích Feature Importance, sử dụng XGBoost cùng MultiOutputRegressor để thực hiện dự báo cho đồng thời nhiều mốc thời gian trong tương lai. Với XGBoost, có

một tính năng đáng quan tâm là Early Stopping, tính năng này cho phép model dừng tạo thêm model con khi loss function dựa trên evaluation set đạt thấp nhất. Một model XGBoost có thể được tối ưu bằng nhiều parameter, có thể kể đến như subsample, colsample_bynode, colsample_bylevel, colsample_bytree, min_child_weight, reg_alpha, reg_lambda, max_depth và learning_rate. Một tip mà tác giả Nicolas đem đến cho chúng ta khi optimize các parameter là ta có thể chạy Random Search hai lần, với lần đầu đặt range của parameter rộng hơn và lần hai thì thu hẹp range của parameter lại chỉ xoay quanh giá trị tối ưu của lần chạy đầu.

Sau cùng, với các parameter tối ưu, XGBoost cho ra kết quả forecast accuracy với MAE tốt nhất trên test set tính đến hiện tại. Xét thêm về tốc độ train, với dataset norway_new_car_sales_by_make.csv và n_estimators=100, XGBoost nhanh hơn AdaBoost 4.5 lần và xấp xỉ với Forest và Extremely Randomized Trees.

Chương 22: Categorical Features

Như đã đề cập tại chương 20, forecasting model có thể được enrich bằng cách thêm vào những dữ liệu bên ngoài cạnh dữ liệu chính historical demand, ví dụ như thông tin về các kênh bán hàng (MT, GT, Horeca, Ecommerce, v.v.), các thị trường mục tiêu (Bắc, Trung, Nam, v.v.), hay các nhóm sản phẩm (Homecare, Healthcare, F&B, v.v.).

Về mặt kỹ thuật, hiện tại không nhiều package ML như scikit-learn có thể trực tiếp làm việc với data dạng categorical (phân loại), vì thế để train model ta phải biến đổi categorical data trở thành numerical data.

Có hai cách để biến đổi data được trình bày trong sách là Integer Enconding và One-hot Enconding. Integer Enconding sẽ có hiệu quả hơn khi áp dụng lên các dữ liệu có tính ordinal (thứ bậc), ví dụ học lực (học sinh giỏi, khá, trung bình, yếu), hay phân khúc sản phẩm (luxury, premium, standard, low-cost). Về One-hot Encoding, nó sẽ được áp dụng lên các data khó xác định rõ thứ bậc như thị trường kinh doanh (EU, US, China, India, v.v.). Ta sẽ tạo ra các trường thông tin có dạng binary (nhị phân), ví dụ với cột EU, nếu sản phẩm được bán tại đây thì giá trị tại trường này sẽ là 1, ngược lại là 0.

Tương tự những chương trước, với phần thực hành của chương 22, tác giả Nicolas hướng dẫn chúng ta chạy Python để encode trên dataset norway_new_car_sales_by_make.csv.

Chương 23: Clustering

Sang chương 23, một dạng học khác của ML là Unsupervised Learning được đem đến cho độc giả thông qua thuật toán K-means Clustering. Giải thích nhanh, hiện nay ML có thể được chia thành 4 dạng học là Supervised Learning, Unsupervised Learning, Semi-Supervised Learning và Reinforcement Learning. Cuốn sách này chỉ bàn đến 2 dạng chính là Supervised và Unsupervised.

Mô tả ngắn gọn, Supervised Learning là khi model ML được cung cấp cả data input và output mong muốn. Thuật toán sẽ tự tìm hiểu mối quan hệ giữa các data này. Nó được gọi là "supervised" (có giám sát) vì bạn cho model biết output mong muốn là gì. Ngược lại, với Unsupervised Learning, model chỉ được cung cấp data input mà không có output cụ thể. Thuật toán sẽ tự tổ chức và phân loại các quan sát dữ liệu khác nhau. Đây gọi là "unsupervised" (không giám sát) vì bạn không gán nhãn cho các mẫu theo cách đã định sẵn và yêu cầu bất kỳ output cụ thể nào.

Một thuật toán nổi tiếng trong nhánh Unsupervised Learning là K-means Clustering với chức năng phân loại input data thành các cụm khác nhau dựa vào sự "tương đồng" về feature của chúng. Chi tiết về cách thuật toán K-means vận hành cùng với các thông tin kỹ thuật khác như distance, inertia bạn có thể xem rõ trong sách.

Phần thực hành, ta sẽ được ứng dụng K-means Clustering trên dataset norway_new_car_sales_by_make.csv bằng cách phân các thương hiệu xe thành 4 cluster khác nhau dựa vào tính seasonal của demand, cuối cùng là sử dụng package seaborn để vẽ ra heatmap trực quan hóa kết quả đầu ra của từng cluster.

Chương 24: Feature Optimization #2

Đến chương 24, bộ dataset tổng hợp tất cả các dạng feature đã bàn đến ở những chương trước được tạo ra. Lúc này, training feature đã có historical demand (trung bình, trung vị, trung bình động 4 tháng, max, min và cụ thể từ M-12 đến M-1), phân khúc sản phẩm (Luxury, Premium, Standard, Low-cost), thương hiệu (Mazda, McLaren, Peugeot, v.v.), tháng (từ 1 đến 12), GDP (trung bình, trung bình động 4 tháng và cụ thể từ M-12 đến M0). Mục đích của tác giả là sử dụng toàn bộ feature để chạy model (XGBoost) rồi sau đó loại dần ra với Feature Importance, chỉ giữ lại những feature có giá trị đóng góp cao lên forecast accuracy nhằm tránh overfitting nhưng vẫn đảm bảo kết quả đầu ra tốt, đây gọi là kỹ thuật backward elimination. Sau đây là top 10 feature (trên tổng số 96 feature) đóng góp đến 80% lên forecast accuracy, theo thứ tự từ lớn đến bé: Demand MA4, Demand Mean, Demand M-1, Month, Exo Mean, Demand M-2, Demand M-3, Demand M-12, Demand M-10 và Demand M-6.

Sau khi loại bỏ đi những feature không đem lại nhiều giá trị, ta chạy lại model và thu được kết quả forecast cải thiện chút xíu, từ 17.4 MAE xuống 17.1 MAE. Ừ thì chấp nhận thôi, biết sao giờ.

Chương 25: Neural Network

Chương 25 bắt đầu với sự liệt kê hàng loạt những thành tựu nổi bật của lĩnh vực Deep Learning (DL) và Neural Network (NN). Tại cuộc thi ImageNet Large Scale Visual Recognition Challenge tổ chức năm 2012, đội của Alex Krizhevsky từ University of Toronto đã sử dụng Convolutional Neural Networks/CNN (Mạng Neuron Tích Chập) và giành chức vô địch với error rate chỉ có 15.3% trong khi error rate của đội á quân là

26.2%. Kế tiếp, vào năm 2016, AlphaGo – một AI phát triển bởi DeepMind – đã chiến thắng nhà vô địch cờ vây thế giới Lee Sedol. Rồi đến năm 2020, OpenAI cho ra mắt thị trường một AI có thể làm thơ, kể chuyện, giải phương trình, lập trình website, v.v. – GPT-3. Và tính đến ngày bài viết này lên sóng thì GPT 4.5 đã được hoàn thành.

Tóm tắt về NN, đây là một dạng model ML lấy cảm hứng từ cách thức hoạt động của bộ não con người. Nó bao gồm các neuron (đơn vị xử lý) được liên kết với nhau qua các connection (đoạn kết nối) có weight (trọng số). Cấu trúc của một NN bao gồm input layer (nhận dữ liệu đầu vào), hidden layers (trung gian nơi thông tin được xử lý), và output layer (dự đoán kết quả cuối cùng). Một thành phần không kém phần quan trọng khác trong NN là activation function (hàm kích hoạt), nó giúp quyết định liệu một neuron có hoạt động hay không và chuyển đổi output từ layer trước thành input cho layer sau. Nếu không có activation function, NN chỉ đơn giản là một phép toán tuyến tính (tương tự như Linear Regression) và không thể mô phỏng các mối quan hệ phi tuyến phức tạp giữa các biến. Một số activation function có thể kể đến như sigmoid, ReLu, tanh, Leaky ReLU, Softmax, v.v.

Tiếp theo, tác giả mô tả cách một NN thực hiện forecast bằng ví dụ gồm 2 input (MA4 = 15 và M12 = 10), sau đó cho các input này đi qua 2 lớp hidden layer với ReLu activation function, cuối cùng cho ra kết quả dự báo là 16.6 tại output layer. Đây gọi là Forward Propagation.

Bàn về phương thức học của NN, như có nhắc đến ở trên, Forward Propagation (Lan Truyền Tiến) là khi NN nhận data input, tính tổng có trọng số tại mỗi neuron rồi cho nó đi qua activation function, cứ như thế cho đến khi có được kết quả output. Mục tiêu của Forward Propagation là tính toán kết quả output dựa vào các trọng số hiện tại rồi tính loss function. Ngược lại. Backpropagation (Lan Truyền Ngược) là sau khi ta có được kết quả loss function, trọng số sẽ được cập nhật bằng các thuật toán như Gradient Descent, Mini-Batch Gradient Descent hay Adam (Adaptive Moment Estimation). Quá trình Forward Propagation và Backpropagation sẽ được lặp đi lặp lại nhiều lần cho đến khi NN đạt được độ chính xác mong muốn. Trích tác giả “During the training phase of a neural network, forward propagation and backpropagation will be performed in turns. Forward propagation will be used to generate predictions (as data flow through the network). Backpropagation will be used to tune the weights based on the latest prediction errors.”

Ở phần thực hành, ta sẽ sử dụng MLPRegressor từ package scikit-learn và sau đó optimize các parameter bằng Random Search. Tuy vẫn cho ra kết quả có thể đem so sánh với XGBoost, scikit-learn không phải package dành riêng cho deep learning. Nếu muốn xây dựng các NN phức tạp hơn, các bạn có thể xem qua các package như Keras, TensorFlow và PyTorch.

Phần 3: Data-Driven Forecasting Process Management

Chương 26: Judgmental Forecasts

Tại chương 26, Nicolas Vandeput khẳng định judgmental forecast (dự báo phán đoán định tính) có thể enrich cho quantitative forecast baseline bằng nhiều cách khác nhau. Tuy nhiên, judgemental forecast cũng có thể gây ra nhiều rủi ro vì lý do bias.

Năm 2009, Oliva và Watson đã đề xuất ra một framework nhằm xem xét các yếu tố dẫn đến bias từ judgemental forecast, có thể kể đến như: Misalignment of Incentives (mỗi stakeholders trong chuỗi cung ứng đều có những mục tiêu và động cơ riêng, ví dụ nhân viên sales sẽ muốn lượng bán ra nhiều hơn so với forecast để nhận bonus, vì thế người ấy sẽ muốn số forecast thấp. Hoặc nhân viên customer service sẽ muốn lượng inventory cao để lúc nào cũng có hàng giao cho khách, từ đó gây ra overforecasting), Power and Influence (đôi khi một số phòng ban sẽ có quyền lực cao hơn những phòng ban khác, từ đó họ có thể tác động lên số forecast để phù hợp với mục tiêu của riêng họ), Process Blind Spots and Data Quality (forecast có thể bị over- hoặc under-pessimistic vì nó dựa vào nguồn dữ liệu bị thiếu, bị bias. Ví dụ trong ngành retail, khi làm forecast đôi khi ta tập trung vào số lượng sản phẩm mới launch mà không quan tâm đến những sản phẩm ngưng bán, hoặc tập trung vào các chiến dịch marketing mà quên mất phần pricing, v.v.). Và một cái blind spot rất là quen thuộc là khi làm demand forecasting là người ta chỉ nhìn vào số sales trong quá khứ chứ không phải số demand, trích "The most common case of informational blind spot is to look at historical sales rather than demand. Most supply chains do not track the real demand, but only the sales." Cùng với những lý do kể trên thì ta còn có những Individual Cognitive Bias như Confirmation Bias (con người có xu hướng hướng góc nhìn của mình vào những điều mà họ cho là đúng và bỏ qua những ý kiến trái chiều) và Anchoring Bias (điều đầu tiên ta nghĩ đến sẽ tác động lớn đến kết quả cuối cùng). Vậy để giải quyết những vấn đề trên ta cần phải làm gì?

Đầu tiên chính là Accountability and Accuracy Tracking. Là một data scientist, chúng ta phải luôn keep track kết quả của model cũng như hiệu quả của từng công đoạn trong quy trình dự báo. Nhằm giảm bớt những bias mang tính intentional, ta có thể track riêng mức sai số được tạo ra bởi sự tweak của các bộ phận liên quan. Ví dụ số model chạy ra là 100, sales bảo đẩy lên 110, marketing bảo đẩy lên 120, finance bảo giảm xuống 105, demand planner chốt số 110. Thì khi có actual demand, ta phải đánh giá lại xem sự bias đến từ đâu.

Một giải pháp nữa đó là Group Forecast, đơn giản là involve nhiều stakeholder vào quy trình forecast để tránh góc nhìn chủ quan của một số ít người. Điều này cũng tương đồng với cách vận hành của những ensemble machine learning model.

Tiếp theo là Independence and Anonymity. Forecast không bao giờ nên là một phần của chính trị công sở (mặc dù trong thực tế thì khác), thì để đảm bảo kết quả không bị bias

thì lãnh đạo doanh nghiệp phải tạo được một sự độc lập và phi chính trị trong quy trình forecast.

Discuss Assumption là giải pháp cuối cùng được Nicolas đưa ra. Theo anh, việc mọi người tập trung vào chọn ra một con số chính xác cho sales hay demand sẽ gây ra bias và influence, thay vào đó, các phòng ban có thể tập trung nhiều vào đánh giá tác động của những assumption và leading indicator có tác động lên số forecast demand như market share, market growth rate, macroeconomics, pricing, new product launches, marketing campaign, v.v.

Chương 27: Forecast Value Added

Chương 27, cũng là chương cuối cùng của sách, bàn về cách để chúng ta sử dụng judgemental forecast nhằm cải thiện kết quả đầu ra của forecast baseline. Có 2 vấn đề chính được thảo luận, thứ nhất là cách sử dụng KPI sao cho thông minh để quản lý toàn bộ danh mục sản phẩm và cách tập trung vào những sản phẩm quan trọng, thứ hai là đánh giá được giá trị của từng stakeholder đóng góp vào quy trình demand forecasting.

Nói về hướng thứ nhất, Portfolio KPI, khi thảo luận về forecast KPI tại chương 2 của cuốn sách, ta đang giả định rằng từng sản phẩm sẽ được track riêng lẻ và forecating model cũng được tối ưu cho riêng từng sản phẩm như thế. Tuy nhiên, trong thực tế thì một demand planner phải quản lý cả một category hàng chứ không phải một vài SKU, từ đó dẫn đến việc theo dõi KPI cho từng sản phẩm như vậy rất mất thời gian và không hiệu quả. Vậy giải pháp là?

Một, Smart Weighting. Sử dụng chỉ số weighted forecast error với trọng số có thể đến từ các yếu tố như mức độ đóng góp vào doanh thu, chi phí sản xuất, hoặc sản phẩm chiến lược, v.v. Với chỉ số này, ta có thể nhìn nhận vấn đề trên hai phương diện là sai số dự báo cùng với mức độ quan trọng của sản phẩm, từ đó tập trung vào những sản phẩm cốt lõi nhưng có sai số dự báo cao.

Hai, Forecast Horizon and Granularity. Chỉ nên tập trung vào forecast với mức độ forecast granularity (ngành hàng, thương hiệu, SKUs) và forecast horizon (năm, tháng, ngày) thật sự có giá trị cho chuỗi cung ứng của bạn. Trích “Forecasts are only helpful for a supply chain if they empower its stakeholders to take action. This means that you have to choose the granularity of your model and which lags to focus on, based on the rhythm of your supply chain and the decisions you can make in response to the forecast.”

Đặt tiếp câu hỏi “What Is a Good Forecast Error?”. Trong cuốn sách này, ta đã bàn nhiều về các model rồi, và ví dụ một model chạy ra forecast với kết quả MAE = 20%, ta có thể nhận định con số này là tốt hay không tốt được không? Hay forecast và demand phải giống nhau 99% thì mới là tốt? Nhận định từ tác giả, forecast accuracy của model phụ thuộc nhiều vào mức độ phức tạp nội tại và tính ngẫu nhiên của demand, ví dụ bạn

forecast số lượng xe bán ra tại Norway theo tháng sẽ dễ hơn forecast số lượng sales của một chiếc smarphone, tại cửa hàng A, vào một ngày cụ thể. Nhìn chung sẽ rất khó để đánh giá một kết quả forecast có tốt hay không nếu thiếu đi benchmark.

Nói về benchmark, để biết được một model đang perform như thế nào, ta có thể so sánh nó với những model đơn giản như Naïve, Seasonal Naïve hay Linear Regression, từ đó sẽ thấy được "how much extra accuracy" ta có với model hiện tại.

Bên cạnh benchmark, Forecast Value Added (FVA) – một concept được ra đời năm 2002 bởi Gilliland – cũng là một công cụ giá trị trong đánh giá kết quả dự báo. Ví dụ, bạn chạy Naïve model thu được MAE 52%, sau đó bạn chạy một ML model và thu được MAE 45%, như vậy giá trị gia tăng của ML model là 52-45 = 7 (point).

Ứng dụng của FVA không chỉ nằm ở việc so sánh hai forecasting model mà ta cũng có thể đưa nó vào đánh giá giá trị gia tăng trong toàn quy trình demand forecasting. Bằng cách này, người quản lý quy trình dự báo có thể thấy được những bước nào hay những bên tham gia nào có đóng góp tích cực lên kết quả dự báo và ngược lại. Cụ thể về triển khai FVA các bạn có thể xem chi tiết trong sách.

Cuối cuốn sách, tác giả đưa ra recommendation nên thúc đẩy sự cộng tác giữa người và máy, giữa định tính và định lượng vì điều này vẫn luôn "show better result". Trích "As a demand planner, you should work together with ML as an efficient duo by letting the ML do the heavy lifting while you focus on bringing new insights by using your network and communication skills."