به نام خدا

گزارش پروژه درس یادگیری ماشین

نگارش

علی ابراهیمی، آمیار سی‌وسه مرده، امیر پارسا جهانبانی

استاد

دکتر شریفی زارچی

زمستان ۱۴۰۴

فهرست

فاز اول – Data Visualization

در این فاز هدف ما این بود که قبل از هر نوع مدل‌سازی، کاملاً بفهمیم داده‌ها چه رفتاری دارند، فروشگاه‌ها چطور عمل می‌کنند و چه متغیرهایی احتمالاً روی فروش اثر جدی دارند. به عبارتی، این مرحله برای این بود که «با داده‌ها زندگی کنیم» و رفتارشان را از نزدیک ببینیم، نه اینکه مستقیم سراغ مدل برویم.

در ادامه، تمام کارهایی که در کد انجام شده و برداشت‌هایی که از آن داشتیم، مرحله‌به‌مرحله توضیح داده می‌شود.

بارگذاری و آماده‌سازی اولیه داده‌ها

ابتدا دو فایل train.csv و store.csv خوانده شدند. فایل train شامل رکوردهای فروش روزانه هر فروشگاه بود و فایل store شامل اطلاعات ثابت مربوط به هر فروشگاه (مثل StoreType و اطلاعات رقابت).

ستون Date به فرمت datetime تبدیل شد تا بتوانیم ویژگی‌های زمانی استخراج کنیم و تحلیل سری زمانی انجام دهیم. سپس داده‌ها بر اساس Store و Date مرتب شدند. این کار مخصوصاً برای محاسبه rolling و تحلیل روند بسیار مهم است، چون اگر ترتیب زمانی به‌هم بخورد، میانگین‌های متحرک و تحلیل روند اشتباه می‌شود.

ستون StateHoliday به صورت رشته‌ای استانداردسازی شد تا مقادیر ناسازگار (مثلاً 0.0) اصلاح شوند. همچنین اگر ستون Open مقدار گمشده داشت، با 1 پر شد (فرض اینکه فروشگاه باز بوده).

در نهایت دو جدول train و store بر اساس Store با هم merge شدند و یک دیتافریم نهایی به نام df ساخته شد. این دیتافریم پایه تمام تحلیل‌های بعدی بود.

ایجاد ویژگی‌های زمانی

برای اینکه رفتار فروش را بهتر بفهمیم، از ستون تاریخ ویژگی‌های زیر استخراج شد:
سال(Year)
ماه(Month)
شماره هفته(WeekOfYear)
روز هفته محاسبه‌شده(DayOfWeek_calc)
و یک شاخص ساده برای آخر هفته بودن(IsWeekend)

این ویژگی‌ها هنوز وارد مدل نشده‌اند، اما برای تحلیل روند و بررسی رفتار فروش بسیار مفید هستند.

فیلتر کردن روزهای باز

برای اینکه مقایسه‌ها عادلانه باشد، فقط روزهایی در نظر گرفته شد که فروشگاه باز بوده (Open = 1) و فروش بیشتر از صفر بوده است. این کار باعث شد تحلیل‌ها تحت تأثیر روزهای تعطیل کامل یا رکوردهای غیرعادی قرار نگیرند. دیتافریم df_open برای همین هدف ساخته شد.

دسته‌بندی فروشگاه‌ها بر اساس سطح فروش

برای اینکه تحلیل روند فقط محدود به یک فروشگاه خاص نباشد، فروشگاه‌ها بر اساس میانگین فروش روزانه دسته‌بندی شدند.

نکته مهم این است که از mean استفاده شد نه .sum چون اگر از مجموع فروش استفاده کنیم، فروشگاهی که مدت بیشتری فعال بوده ممکن است صرفاً به خاطر تعداد روز بیشتر، مجموع بالاتری داشته باشد. اما میانگین روزانه معیار عادلانه‌تری است.

بر اساس صدک‌های 33٪ و 66٪، فروشگاه‌ها به سه گروه تقسیم شدند:

فروش پایین

فروش متوسط

فروش بالا

از هر گروه به صورت تصادفی 3 فروشگاه انتخاب شد تا رفتار آن‌ها بررسی شود. این کار کمک کرد تحلیل ما فقط به یک نمونه خاص محدود نشود.

تحلیل روند فروش(Trend Analysis)

برای هر گروه، فروش به صورت هفتگی تجمیع شد (resampleهفتگی با مجموع فروش). سپس برای نرم‌تر شدن نمودار، میانگین متحرک سه‌هفته‌ای روی آن اعمال شد.

نتایج نشان داد:

- فروشگاه‌های گروه High سطح فروش بالاتر و نوسان بزرگ‌تری دارند.
- فروشگاه‌های Low پایدارتر اما با دامنه کوچک‌تر هستند.
- در همه گروه‌ها الگوی فصلی هفتگی و دوره‌های اوج و افت تکرارشونده مشاهده شد.

همچنین یک نمودار مقایسه‌ای نرمال‌شده رسم شد. در این نمودار فروش هر فروشگاه بر میانه خودش تقسیم شد تا شکل روندها با هم مقایسه شود.

این کار نشان داد که اگرچه سطح فروش متفاوت است، اما شکل نوسان بسیاری از فروشگاه‌ها مشابه است. این موضوع نشان می‌دهد که عوامل فصلی مشترک در کل شبکه فروش وجود دارد.

بررسی اثر ویژگی‌ها بر فروش

در این بخش اثر چند متغیر کلیدی بررسی شد:

اثرPromo

میانگین و میانه فروش در حالت Promo=1 و Promo=0 محاسبه شد. همچنین boxplot رسم شد تا توزیع کامل دیده شود.

نتیجه واضح بود:
در روزهای پروموشن، میانگین فروش به شکل محسوسی بالاتر است.

نرخ uplift نسبی نیز محاسبه شد:
(میانگین فروش در - Promo=1 میانگین درPromo=0) تقسیم بر میانگین در حالت بدون پروموشن.

این عدد نشان می‌دهد پروموشن چه درصدی فروش را افزایش می‌دهد. این متغیر یکی از مهم‌ترین عوامل مدل خواهد بود.

اثرSchoolHoliday

در تعطیلات مدارس، در برخی فروشگاه‌ها فروش افزایش جزئی داشت، اما اثر آن به شدت Promo نبود. توزیع فروش در حالت SchoolHoliday=1نسبتاً گسترده‌تر بود.

اثرStateHoliday

در تعطیلات رسمی، فروش در بسیاری از فروشگاه‌ها کاهش داشت یا رفتار متفاوتی نشان می‌داد. اما این اثر یکنواخت نبود و بستگی به نوع فروشگاه داشت.

اثر Promo بر اساسStoreType

در این قسمت بررسی شد که آیا اثر پروموشن در همه انواع فروشگاه یکسان است یا نه.

برای هر StoreType میانگین و میانه فروش در حالت Promo=0 و Promo=1 محاسبه شد و نمودار خطی رسم شد.

نتایج نشان داد:
در بعضیStoreType ها اثر Promo بسیار قوی‌تر از سایرین است.
یعنی همه فروشگاه‌ها به یک شکل به پروموشن واکنش نشان نمی‌دهند.

این نتیجه اهمیت تعامل بین StoreType و Promo را در مدل‌سازی نشان می‌دهد.

تحلیل همبستگی(Correlation Analysis)

در انتها ماتریس همبستگی بین متغیرهای عددی محاسبه شد.

نکته مهم: ستون Customers حذف شد تا data leakage رخ ندهد، چون Customers عملاً نماینده مستقیم فروش است و استفاده از آن پیش‌بینی را غیرواقعی می‌کند.

در ماتریس همبستگی مشاهده شد:

- Promoهمبستگی مثبت با Sales دارد.
- برخی ویژگی‌های زمانی همبستگی متوسط دارند.
- برخی متغیرها تقریباً بی‌اثر هستند.

همچنین 15 متغیری که بیشترین همبستگی را با Sales داشتند لیست شدند. این لیست سرنخ اولیه برای مهندسی ویژگی در فاز بعدی بود.

فاز دوم – Feature Engineering

اول از همه فایل‌های train و store رو خوندیم و با هم یکی کردیم. ستون تاریخ رو تبدیل به datetime کردیم که بتونیم راحت ازش سال و ماه و بقیه چیزها رو استخراج کنیم. داده‌ها رو بر اساس Store و Date مرتب کردیم، چون وقتی قراره لگ و میانگین متحرک بسازیم، اگر ترتیب زمانی درست نباشه کل فیچرها خراب می‌شن.

بعد شروع کردیم به ساختن فیچرهای زمانی. از روی تاریخ، سال، ماه، روز، شماره هفته، شماره روز سال و این‌که آخر هفته هست یا نه رو استخراج کردیم. حتی این‌که اول ماهه یا آخر ماه هم مشخص کردیم. اینا به مدل کمک می‌کنن بفهمه مثلاً فروش آخر ماه فرق داره یا نه، یا تابستون‌ها چه رفتاری داریم.

بعد رفتیم سراغ تعطیلات. فقط نگفتیم امروز تعطیله یا نه. یه کار قشنگ‌تر کردیم: حساب کردیم چند روز مونده تا تعطیلی بعدی و چند روز از تعطیلی قبلی گذشته. این خیلی مهمه چون فروش معمولاً قبل تعطیلی یه رفتاری داره، بعد تعطیلی یه رفتار دیگه. اینجوری مدل می‌تونه اون موج قبل و بعد تعطیلات رو یاد بگیره.

برای اینکه فصل‌ها رو بهتر بفهمه، از سینوس و کسینوس استفاده کردیم .(Fourier terms) این کار کمک می‌کنه مدل به شکل نرم‌تر بفهمه که سال یه سیکل داره. یعنی به جای اینکه فقط بگیم "ماه 1، ماه 2..."، یه موج پیوسته بهش می‌دیم که بتونه الگوی سالانه رو بهتر یاد بگیره.

از اطلاعات مربوط به فروشگاه هم فیچر ساختیم. مثلاً حساب کردیم رقیب از کی وارد بازار شده و چند روز از اون تاریخ گذشته. یا اینکه Promo2تو این ماه خاص فعاله یا نه. چون داشتن Promo2 به تنهایی کافی نیست؛ مهم اینه که اون ماه جزو بازه پروموشن باشه یا نه. حتی یه ستون ساده ساختیم که بگه فاصله تا رقیب اصلاً معلوم هست یا نه.

اما مهم‌ترین قسمت این فاز ساخت lag و rolling بود. چون فروش امروز خیلی به فروش دیروز و هفته قبل وابسته است. برای هر فروشگاه فروش 1 روز قبل، 7 روز قبل، 14 روز قبل و 28 روز قبل رو ساختیم. بعد میانگین و انحراف معیار متحرک 7، 14 و 28 روزه رو هم اضافه کردیم.

اینجا یه نکته خیلی مهم رعایت شد: قبل از محاسبه میانگین متحرک، فروش رو یک روز شیفت دادیم. یعنی فروش امروز تو محاسبه خودش وارد نشد. اگر این کار رو نمی‌کردیم، مدل عملاً جواب امروز رو از خودش می‌فهمید و یه RMSE الکی عالی می‌داد. این دقیقاً همون جاییه که خیلی‌ها اشتباه می‌کنن و بدون اینکه بفهمن داده آینده رو وارد گذشته می‌کنن.

یه فیچر دیگه هم ساختیم که میانگین تجمعی فروش هر فروشگاه تا قبل از امروز رو نشون می‌ده. این کمک می‌کنه مدل بفهمه سطح کلی فروش این فروشگاه چقدره.

بعد یه تعامل ساده بین Promo و SchoolHoliday ساختیم. چون ممکنه اثر پروموشن تو تعطیلات مدرسه فرق داشته باشه. این تعامل‌های ساده بعضی وقتا خیلی تاثیرگذار می‌شن.

ستون‌های دسته‌ای مثل نوع فروشگاه و Assortment و StateHoliday رو هم به صورت one-hot درآوردیم تا مدل بتونه باهاشون کار کنه.

بعد از اینکه همه فیچرها ساخته شد، رفتیم سراغ تقسیم زمانی. 42 روز آخر رو گذاشتیم برای تست، 42 روز قبلش برای اعتبارسنجی، و بقیه رو برای آموزش. این تقسیم کاملاً زمانی بود، نه تصادفی. چون تو سری زمانی نباید آینده رو قاطی گذشته کرد.

پر کردن مقادیر گمشده رو عمداً بعد از تقسیم انجام دادیم. مدین هر ستون فقط روی train حساب شد و همون مقدار روی val و test هم اعمال شد. این کار برای این بود که هیچ اطلاعاتی از آینده وارد train نشه.

بعد ستون‌هایی که نباید وارد مدل می‌شدن حذف شدند. مهم‌ترینش Customers بود، چون تقریباً مستقیم با Sales مرتبطه و استفاده ازش باعث میشه مدل تقلبی خوب بشه.

ویژگی‌ها با StandardScaler اسکیل شدند، اما فقط فیچرها، نه Store. Store شناسه است و معنی عددی نداره که اسکیل بشه. اسکیلر فقط روی train fit شد و بعد روی val و test اعمال شد.

در آخر، خروجی نهایی ذخیره شد. فیچرها اسکیل‌شده هستند، Salesبا log1p تبدیل شده تا توزیعش نرمال‌تر بشه و مدل راحت‌تر یاد بگیره، و ستون‌های Store و Date هم برای تحلیل‌های بعدی نگه داشته شدند.

فاز سوم – Learn & Estimate

اول کار، فایل‌های خروجی فاز ۲ یعنیtrain_features.csv ، val_features.csvو test_features.csv رو لود کردیم و لیست ستون‌های فیچر رو هم از feature_cols.json برداشتیم. یک "هارد فیکس" هم گذاشته شد که اگر به هر دلیلی ستون Store داخل feature_cols مانده باشد، حذفش کنیم. این کار خیلی مهمه چون Store شناسه است، اگر به عنوان فیچر وارد مدل بشه، مدل ممکنه به جای یادگیری الگو، بره سمت حفظ کردن رفتار فروشگاه‌ها و نتیجه‌ها غیرواقعی خوب یا عجیب بشه. بعدش هم Date دوباره datetime شد و Store با خیال راحت به int تبدیل شد که وسطmerge ها یا گروپ‌بندی‌ها به مشکل نخوره.

از همون اول تصمیم پروژه این بود که Sales داخل فایل‌های فاز ۲ به شکل log1p ذخیره شده (یعنی .((log(1+Sales پس اینجا هر متریکی که حساب می‌کردیم، باید اول خروجی مدل و y واقعی رو برگردونیم به مقیاس اصلی. دقیقاً همین کار انجام شد: توی تابع rmspe و rmse اول np.expm1 زدیم روی y_true و y_pred تا برگردن به فروش واقعی، بعد metric رو حساب کردیم. حتی ماسک گذاشتیم که y_true>0باشه، چون RMSPE با صفرها مشکل داره. این باعث شد گزارش خطا واقعی و قابل اعتماد باشه، نه مصنوعی.

بعد رفتیم سراغ baseline اول: میانگین متحرک ۷ روزه. برای اینکه baseline واقعاً ساده و منصفانه باشه، مستقیم از train.csv خام استفاده کردیم. برای هر فروشگاه، فروش رو یک روز شیفت دادیم و 7 rolling mean روزه گرفتیم. یعنی پیش‌بینی امروز فقط از گذشته ساخته می‌شد. برای جاهایی که rolling موجود نبود (مثلاً اول‌های سری)، مقدار پیش‌بینی رو با میانگین فروش همان فروشگاه پر کردیم و اگر باز هم چیزی نشد، با میانگین کل پر شد. نتیجه این baseline روی validation این بود که RMSE حدود 3424.65 و RMSPE حدود 0.3030 شد. این عددها به ما گفتن اگر خیلی ساده پیش بریم، درصد خطا نسبتاً بالاست و جا برای بهتر شدن خیلی داریم.

بعد baseline دوم رو ساختیم: یک مدل خطی، ولی نه با کتابخانه آماده؛ خودمون با گرادیان‌دیسنت و ریدج (L2) پیاده‌سازی کردیم. وزن‌ها با مقدارهای کوچک تصادفی شروع شدن، آموزش به صورت mini-batch انجام شد batch_size) بزرگ برای سرعت)، و هر epoch روی validation RMSPEحساب شد. برای اینکه مدل overfit نکنه، بهترین وزن‌ها بر اساس کمترین RMSPE validation نگه داشته شد و آخر کار هم همون بهترین وزن‌ها برگردونده شد. روند آموزش هم مشخص بود که RMSPE از حدود 0.90 اول کار کم‌کم پایین اومد و بعد از چندین epoch تثبیت شد.

خروجی نهایی مدل خطی این بود که روی validation حدود RMSPE 0.2248 و روی تست حدود RMSPE 0.2048 گرفتیم. یعنی نسبت به moving average بهتر شد، ولی هنوز فاصله زیادی با یک مدل قوی‌تر داشت. این دقیقاً همون چیزی بود که انتظار داشتیم: مدل خطی وقتی فیچرهای سری‌زمانی خوبی داریم، بد نیست، اما چون رابطه‌ها تو فروش معمولاً غیرخطی و پر از interaction هست، مدل خطی یه سقف مشخص داره.

بعد برای اینکه بفهمیم مدل خطی بیشتر به چی تکیه کرده، "اهمیت ویژگی" رو با قدر مطلق وزن‌ها حساب کردیم و ۲۰ تا ویژگی برتر رو رسم کردیم. چیزی که تو نمودار خیلی واضح بود اینه که Open بیشترین وزن رو داشت و خیلی از بقیه بزرگ‌تر بود. این منطقیه چون وقتی فروشگاه بسته است، فروش عملاً نزدیک صفر می‌شه و اگر این ستون درست وارد مدل شده باشه، اثرش خیلی شدید دیده می‌شه. بعد ازOpen ، Promoو چندتا lag مثل Sales_lag_14 و Sales_lag_7 و همچنین weekofyear و چندتا ویژگی فصلی و فروش تجمعی فروشگاه دیده می‌شدن. خلاصه‌اش این بود که مدل خطی بیشتر به سیگنال‌های خیلی مستقیم و ساده تکیه کرده.

اما بخش اصلی فاز ۳، XGBoostبود. اینجا رفتیم سراغ مدل درختی تقویتی چون معمولاً برای دیتای Rossmann خیلی خوب جواب می‌ده و می‌تونه تعامل‌ها و غیرخطی بودن‌ها رو راحت‌تر یاد بگیره. داده‌ها رو به فرم DMatrix تبدیل کردیم و یک custom metric هم نوشتیم که دقیقاً RMSPE رو روی مقیاس واقعی حساب کنه (با .(expm1 اینجا هم همان داستان برگرداندن از log به فروش واقعی رعایت شد.

برای تنظیم هایپرپارامترها، کد طوری نوشته شد که اگر RUN_TUNING=True باشه، گرید سرچ اجرا بشه و ترکیب‌های مختلف max_depthو eta رو امتحان کنه. اما برای اینکه اجرای پروژه طولانی و چندساعته نشه، گرید سرچ خاموش بود و از یک ترکیب از قبل انتخاب‌شده استفاده شد؛ ترکیبی که داخل همان فضای جستجو قرار داشت و گفته شده کمترین خطا را داده. پارامترهای نهایی این‌ها بودند : max_depth=8، eta=0.05، subsample=0.8، .colsample_bytree=0.7هدف هم reg:squarederror بود و eval_metric پیش‌فرض خاموش شد تا فقط همان RMSPE سفارشی معیار اصلی باشد.

مدل با early stopping آموزش داده شد. یعنی اگر روی validation بهتر نشد، ادامه نده که overfit نشه. طبق لاگ آموزش، RMSPE از مقدارهای بالا خیلی سریع پایین اومد و بعد از چندصدround ، نزدیک بهترین حالت تثبیت شد. در نهایت بهترین round حدود 231 بود (تو خروجی هم مشخص بود) و مدل روی تست به RMSPE حدود 0.1111 رسید. این عدد خیلی مهمه چون یعنی XGBoost تقریباً خطا رو نسبت به مدل خطی تقریباً نصف کرد و نسبت به moving average هم خیلی بهتر شد. پس نتیجه واضح این فاز این شد که مدل درختی تقویتی با اختلاف بهترین گزینه‌ی کلاسیک ماست.

بعد از آموزشXGBoost ، نمودار feature importance هم رسم شد. تو این نمودار چندتا نکته خیلی واضح بود . CompetitionDistanceجزو مهم‌ترین‌ها بود، یعنی فاصله تا رقیب واقعاً روی فروش اثر داشته و مدل ازش به عنوان یک سیگنال قوی استفاده کرده. بعدش Sales_lag_1 خیلی بالا بود که کاملاً طبیعی است چون فروش دیروز معمولاً بهترین پیش‌بینی برای امروز است . DayOfWeekهم جزو بالایی‌ها بود، یعنی مدل به شدت الگوی هفتگی رو یاد گرفته .Sales_lag_7 و Sales_lag_14 هم بالا بودند که یعنی اثر "هفته قبل" و "دو هفته قبل" نقش پررنگی داشته. بعدش هم rolling ها مثل Sales_roll_std_7 و Sales_roll_mean_7 و همچنین Sales_cum_mean_store دیده می‌شدن که یعنی مدل هم سطح کلی فروشگاه رو می‌فهمه، هم نوسان‌های کوتاه‌مدت رو. حتی Fourier termها (sin/cos) هم تو ۲۰ تای اول ظاهر شدند که یعنی سیگنال فصل‌ها واقعاً کمک کرده، ولی به اندازهlag ها تعیین‌کننده نبوده.

فاز چهارم – Uncertainty & Sales Classification

اول کار، دوباره خروجی‌های فاز ۲ رو لود کردیم(train/val/test) ، لیست فیچرها رو از JSON برداشتیم و مثل قبل مطمئن شدیم ستون Storeاشتباهی وارد فیچرها نشده. چون هنوز هم Store شناسه است و اگر بیاد تو مدل، خیلی راحت مدل به جای یاد گرفتن، میره سمت حفظ کردن. بعد همه train/val/test رو با هم چسبوندیم (df_all) و بر اساس Store و Date مرتب کردیم، چون تو این فاز کلی شیفت و افست داریم و اگر ترتیب درست نباشه همه چی خراب میشه.

نکته خیلی مهم این فاز اینه که چون ما پیش‌بینی چندروزه داریم، نمی‌تونیم همون split قبلی رو خام استفاده کنیم. چرا؟ چون وقتی می‌خوای ۷ روز جلوتر رو پیش‌بینی کنی، باید مطمئن باشی لیبل (y) از آینده‌ی مجاز میاد و با بخش‌های val/test قاطی نمیشه. برای همین یک ماسک زمانی مخصوص هر افق h ساختیم. یعنی برای هر) h از ۱ تا ۷) گفتیم:
آخرین روزی که می‌تونه تو train باشه، باید h روز قبل از شروع val تموم بشه، و همین‌طور برای val و test هم مرزها رو عقب کشیدیم. این کار باعث شد هیچ‌وقت y" آینده" از بیرون بازه خودش وارد آموزش نشه. این دقیقاً همون کاریه که خیلی‌ها انجام نمی‌دن و بعد متوجه نمی‌شن چرا نتایج‌شون الکی خوبه.

حالا اصل داستان: ما برای هر افق) h ۱ تا ۷ روز آینده) اومدیم y رو با (shift(-h ساختیم. یعنی اگر امروز ۲۰۱۵-۰۶-۰۱ باشه و h=3 باشه، لیبل میشه فروش ۳ روز بعد. بعد ردیف‌هایی که y ندارن (نزدیک انتهای سری) حذف شدن و X و y آماده شد. تو خروجی هم دقیقاً می‌بینی که با زیاد شدن h تعداد نمونه‌ها کم میشه، چون هر چی افق جلوتر میره، ته سری‌ها لیبل ندارن و باید بریزن بیرون. این طبیعیه و درست هم هست.

مدلی که برای multi-step استفاده کردیم "Quantile Regression" بود، آن هم از صفر با گرادیان‌دیسنت. یعنی برای هر h سه مدل جدا ساختیم: یکی برای ۰.۱(p10) ، یکی برای ۰.۵(p50) ، یکی برای ۰.۹ .p50 (p90). همون پیش‌بینی مرکزیه (مثل(median ، و p10/p90 بازه‌ای میدن که می‌تونه نقش عدم‌قطعیت رو بازی کنه. اینجا به جای loss معمولی، از pinball loss استفاده شد که مخصوص کوانتایل‌هاست.

بعد از اینکه برای هر افق مدل‌ها آموزش داده شدن، برای validation چند تا چیز رو حساب کردیم:
RMSE، RMSPE، MAPEو R2 برایp50 ، و pinball loss برای هر کوانتایل.
بعد هم دو تا شاخص خیلی مهم مربوط به uncertainty رو حساب کردیم:
یکی coverage یعنی چند درصد از y واقعی داخل بازه p10 تا p90 افتاده، و یکی width یعنی میانگین پهنای بازه (p10-p90) چقدره.

اینجا دقیقاً یه مشکل مهم تو نتایجت دیده میشه که تو اسکرین‌شات هم خیلی واضح هست coverage :برای بازه p10-p90 تقریباً صفر شده. یعنی چی؟ یعنی مدل‌های کوانتایل به شکلی آموزش دیدن که p10 و p90 درست "دور "y رو نگرفتن، یا حتی بدتر، احتمالاً ترتیب کوانتایل‌ها خراب شده (مثلاً p10 از p90 بزرگ‌تر شده یا برعکس)، یا scale/log handling یه جا اشتباه شده و خروجی‌ها به شکل معناداری از y پرت شدن. چون coverage=0.0 برای همهhorizon ها واقعاً نشونه اینه که uncertainty ما از نظر کالیبراسیون بد دراومده و بازه‌ها اصلاً واقع‌بینانه نیستن. اینو تو گزارش به عنوان "چالش/ایراد مشاهده‌شده" می‌نویسیم، نه اینکه پنهانش کنیم، چون اتفاقاً خیلی هم حرفه‌ایه که بگی اینجا کالیبراسیون interval درست نشد و دلیلش رو حدس بزنی.

یه نشونه دیگه هم تو جدول هست val_rmspe_p50 :تقریباً ۱ شده و R2 منفی خیلی بزرگه. این هم یعنی پیش‌بینی‌های p50 از نظر مقیاس یا برگشت ازlog ، درست نمی‌نشینن. چون اگر مدل یه چیز خیلی پرت بده، RMSPEمیره نزدیک ۱ یا بالاتر، و R2 منفی میشه. پس برداشت گزارش این میشه که "مدل quantile خطی برای multi-step تو این پیاده‌سازی، مثل XGBoost موفق نبود و احتمالاً به خاطر ضعف مدل خطی/هایپرپارامترها یا مشکل در پیاده‌سازیscale/log ، خروجی‌ها کالیبره نشدن."

ولی با این حال، از نظر ساختار پروژه، پیاده‌سازی فاز ۴ دقیق و درست طراحی شده: مدل برای هر افق جداست، leakageکنترل شده، و هم خروجی متریک و هم نمودارها ذخیره شدن. یعنی فریم‌ورک درسته، فقط نتیجه مدل اینجا مثل فاز ۳ خوب نشده.

بعد از بخش کوانتایل، ما یک "جدول پیش‌بینی نمونه" هم ساختیم. یعنی از یک تاریخ مبدا (یک روز قبل از شروع (test رفتیم برای چند فروشگاه تصادفی، برای هر کدام فروش ۱ تا ۷ روز آینده رو با p10/p50/p90 نوشتیم و تو forecast_sample_next_horizons.csv ذخیره کردیم. این بخش خیلی شبیه چیزیه که تو واقعیت به مدیر می‌دن: می‌گه برای فروشگاهX ، فردا حدوداً اینه، بدبینانه اینه، خوشبینانه اینه.

بعد رفتیم سراغ عدم‌قطعیت با .ensemble ایده‌اش این بود که به جای اینکه مستقیم از p10/p90 استفاده کنیم، چندتا مدل median جدا آموزش بدیم (K=7) تا) با seed متفاوت، بعد روی validation ببینیم پراکندگی پیش‌بینی‌ها چقدره. از روی میانگین و std اون‌ها یک بازه‌ی تقریبی ۹۰٪ ساختیم (با .(z≈1.6449 اینجا هم خروجی مهمش coverage_approx90 بود. تو اسکرین‌شات می‌بینیم coverage حدود 0.172 بوده، یعنی خیلی کمتر از 0.90. این یعنی حتی ensemble هم بازه‌ها رو درست کالیبره نکرده و uncertainty خیلی کم‌عرض بوده یا خطا زیاد بوده. خلاصه نتیجه این میشه که "عدم‌قطعیت تولید شد، اما کالیبراسیون ضعیف بود و نیاز به تنظیم دارد".

حالا بخش آخر و جذاب‌تر: طبقه‌بندی فروش.

اینجا گفتیم به جای اینکه عدد دقیق فروش رو بگیم، فروش رو سه‌کلاس کنیم .Low / Medium / High :برای اینکه کلاس‌ها بالانس باشن، آستانه‌ها رو از داده train برداشتیم: یک‌سوم پایین (t1) و دوسوم پایین .(t2) یعنی تقریباً هر کلاس یک‌سوم داده‌ها رو می‌گیره. این باعث میشه مدل طبقه‌بندی با مشکل imbalance شدید مواجه نشه.

این طبقه‌بندی رو برای دو افق انجام دادیم h=1 :و .h=7 یعنی هم فروش فردا رو کلاس‌بندی کردیم، هم فروش ۷ روز بعد رو.

مدل هم یک softmax regression بود که باز از صفر با GD پیاده‌سازی شده. بعد با دو معیار اصلی ارزیابی کردیم F1-macro :وROC-AUC (OvR macro).تو نتایجت (طبق جدول اسکرین‌شات) برایh=1 ، روی validation F1 حدود 0.72 و روی test F1 حدود 0.78 شده و AUC هم بالاست (حدود 0.88 تا 0.93). برای h=7 هم حتی بهتره F1 test :حدود 0.815 و AUC test حدود 0.944. این یعنی با اینکه مدل رگرسیون چندگامی (کوانتایل) خوب درنیومده، ولی طبقه‌بندی سطح فروش خیلی خوب جواب داده. یعنی مدل می‌تونه تشخیص بده فروش اون روز "کمه یا متوسطه یا زیاده" حتی اگر عدد دقیقش رو خیلی دقیق نزنه.

در آخر هم خروجی‌ها کامل ذخیره شدن داخل artifacts/phase4/ شامل متریک‌ها، نمودار RMSPE نسبت به افق، نمودارcoverage ، نمودارwidth ، جدول پیش‌بینی نمونه، متریک‌های ensemble و همین‌طور confusion matrix های val/test برای h=1 وh=7.

جمع‌بندی خیلی خودمونی فاز ۴ اینه: ما یک سیستم کامل ساختیم که برای هر فروشگاه بتونه ۱ تا ۷ روز آینده رو پیش‌بینی کنه و حتی بازه اطمینان بده. از نظر طراحی و ضد لیک بودن، کار تمیزه و درست انجام شده. اما نتایج قسمت quantile و uncertainty نشون داد که این

مدل خطیِ کوانتایل تو این اجرا خوب کالیبره نشده وcoverage ها افتضاح شده (تقریباً صفر). در مقابل، بخش طبقه‌بندی خیلی خوب جواب داده و برای تصمیم‌گیری سطحی (کم/متوسط/زیاد) عملکرد قوی داشته.

فاز پنجم - Deep

اول کاری که کردیم این بود که خروجی‌های فاز ۲ رو لود کردیمtrain_features :، val_features، test_featuresو لیست ستون‌های فیچر رو از feature_cols.json گرفتیم. مثل فازهای قبل یه چک جدی گذاشتیم که ستون Store توی feature_cols نباشه. چون Store فقط ID هست و اگر بیاد تو مدل، مدل ممکنه به جای یاد گرفتن الگو، صرفاً فروشگاه‌ها رو حفظ کنه یا وزن عجیب بهش بده. پس Store رو از فیچرها حذف کردیم و تارگت هم همون Sales بود که (طبق فاز ۲) به شکل log1p ذخیره شده.

حالا تفاوت اصلی دیپ‌لرنینگ با مدل‌های قبلی اینه که اینجا مدل انتظار داره به جای یک ردیف مستقل، "یه دنباله از روزهای پشت‌سرهم" رو ببینه. برای همین ما یه Sliding Window ساختیم. یعنی گفتیمLOOK_BACK = 14 ، پس برای هر فروشگاه، میایم ۱۴ روز گذشته رو به عنوان ورودی X می‌دیم و فروش روز بعدش رو به عنوان y می‌گیریم. این کار رو فروشگاه‌به‌فروشگاه انجام دادیم که دیتای دو فروشگاه قاطی نشه. این خیلی مهمه چون اگر ردیف‌ها رو همینجوری پشت‌هم بذاری، ممکنه انتهای فروشگاه ۱ و ابتدای فروشگاه ۲ کنار هم بیفتن و مدل فکر کنه اینا ادامه هم هستن! برای همین ما groupby(Store)زدیم، تاریخ رو سورت کردیم، بعد توی هر فروشگاه جداگانه پنجره‌های ۱۴تایی ساختیم.

پس خروجی این مرحله این شد که X_train و X_val و X_test به شکل سه‌بعدی ساخته شدن:
تعداد نمونه‌ها، ۱۴، تعداد فیچرها
یعنی هر نمونه دقیقاً یک "سکانس ۱۴ روزه از فیچرها" بود.

بعد رفتیم سراغ ساخت دو مدل:

1. LSTM
LSTMرو به شکل ساده ساختیم: یک لایه LSTM با units قابل تنظیم، بعد Dropout برای جلوگیری ازoverfitting ، بعد یک Dense 64 باReLU ، و در آخر Dense(1) برای خروجی عددی Loss .هم MSE گذاشتیم و optimizer هم.adam این معماری خیلی پیچیده نیست، ولی برای پروژه دانشگاهی کاملاً استاندارد و قابل دفاعه.

2. TCNدر عمل یک مدل Conv1D ساده
اینجا یه مدل کانولوشنی ساختیم Conv1D :با فیلتر ۶۴ و کرنل ۲، بعدMaxPooling ، بعدFlatten ، و.Dense حقیقتش این معماری بیشتر CNN" روی سری‌زمانی" حساب میشه تا TCN واقعی، ولی از نظر ایده، همون داستان رو داره: با کانولوشن روی زمان، الگوهای کوتاه‌مدت رو می‌گیره و سریع‌تر از LSTM هم می‌تونه یاد بگیره.

بعدش نوبت Optuna شد. ما کد Optuna رو گذاشتیم که بتونه units و dropout رو برای LSTM تیون کنه. یعنی اگر RUN_OPTUNA=Trueمی‌شد، با چند تا trial مدل رو می‌ساخت، چند epoch سریع آموزش می‌داد، و val_loss رو معیار انتخاب می‌گرفت. حتی early stopping هم گذاشتیم که زمان الکی تلف نشه.

ولی چون این پروژه اگر Optuna روشن می‌شد ممکن بود طولانی بشه، RUN_OPTUNAرو False گذاشتیم و گفتیم پارامترها رو از قبل یک مقدار مناسب انتخاب می‌کنیم units=64 :و .dropout=0.2 این همون چیزیه که تو خروجی هم ثبت شده.

بعد از مشخص شدن پارامترها، هر دو مدل رو نهایی آموزش دادیم. هر کدوم epoch ۱۰ با batch_size=256 و validation_data اجرا شدند. یعنی مدل‌ها همزمان روی val کنترل می‌شدن که خیلی overfit نکنن.

بعد رسیدیم به ارزیابی روی .test اینجا چونy ها log هستند، ما هم y_test رو expm1 کردیم هم پیش‌بینی‌ها رو expm1 کردیم تا همه چیز برگرده به مقیاس واقعی یورو. بعد RMSE و RMSPE رو حساب کردیم RMSPE .رو هم فقط روی جاهایی حساب کردیم که y_true>0باشه تا تقسیم بر صفر پیش نیاد.

نتایج دقیقاً طبق خروجی که فرستادی این بود:

- LSTM
  RMSE = 3631.13یورو
  RMSPE = 0.4835
- TCN
  RMSE = 2489.15یورو
  RMSPE = 0.3768

تو این پیاده‌سازی، مدل TCN از LSTM بهتر جواب داده. یعنی هم خطای مطلقش کمتره RMSE) پایین‌تر)، هم درصد خطاش بهتره RMSPE)پایین‌تر). این اتفاق هم خیلی عجیب نیست؛ چون توRossmann ، خیلی وقتا مدل‌های کانولوشنی ساده یا مدل‌های درختی (مثل XGB) از LSTM بهتر عمل می‌کنن، مخصوصاً وقتی داده tabular زیادی داریم و سکانس‌ها خیلی بلند نیستن LSTM .معمولاً وقتی خیلی خوب میشه که سری‌زمانی "واقعاً "sequence-heavy باشه و features خیلی تمیز و پیوسته باشن و مدل هم زمان کافی برای تیون داشته باشه.

بعد از این، یک نمودار مقایسه‌ای هم کشیدیم برای آخرین ۱۰۰ نمونه تست: خط واقعی فروش با خط چین مشکی، و دو خط پیش‌بینی برای LSTMو .TCN این نمودار خیلی به درد گزارش می‌خوره چون مصحح می‌بینه مدل‌ها چقدر trend رو می‌گیرن و چقدر lag دارن.

حالا بخش جذاب بعدی.SHAP :

اینجا هدف این بود که فقط نگیم "مدل فلان خطا رو داد"، بلکه یه ذره هم توضیح بدیم "چرا این پیش‌بینی رو کرده؟ به چی نگاه کرده؟". چون مدل‌های عمیق معمولاً black-box هستن. برای همین از SHAP KernelExplainer استفاده کردیم. ولی چون SHAP برای deep model خیلی سنگینه، اومدیم یک نمونه کوچک برداشتیم (۲۰ تا نمونه از(X_test ، بعد اون رو reshape کردیم به ۲ بعدی (چون SHAP اینجا با ورودی ۲بعدی راحت‌تر کار می‌کنه)، و یک wrapper نوشتیم که دوباره این ۲بعدی رو به ۳بعدی تبدیل کنه و بده به.model.predict

برای اینکه SHAP سریع‌تر بشه، از shap.kmeans برای ساختن background استفاده کردیم (۱۰ مرکز). همونwarning های قرمز تو اسکرین‌شات هم از همین قسمت میاد sklearn :کِی‌مینز روی MKL یه هشدار مربوط بهthread ها میده که خیلی جدی نیست، بیشتر یه warningسیستمیه و روی نتیجه تاثیر خاصی نداره. مهم اینه که SHAP اجرا شده و خروجی‌ها ذخیره شده.

در نهایت دو تا plot SHAP ذخیره شد: یکی برای LSTM و یکی برای .TCN اینا به شکل summary plot نشون میدن کدوم فیچرها به طور کلی بیشتر اثر گذاشتن. فقط یه نکته: چون ما ورودی رو flatten کردیم (۱۴ روز × تعداد فیچرها)، اگر اسم فیچرها رو دقیق نگذاشته باشیم، SHAPممکنه ستون‌ها رو با شماره نشون بده یا خیلی خوانا نباشه. ولی برای گزارش همین که "feature importance" رو نشون بدیم و بگیم مدل روی چه چیزهایی حساسه، کافی و قابل دفاعه.

آخر کار هم یک فایل metrics_deep_learning.csv ذخیره کردیم. نکته جالب اینجا اینه که به خاطر یک باگ ImportError یا مشکل pandas.to_csvدر محیط، از ماژول استاندارد csv استفاده شد تا خروجی متریک‌ها حتماً ذخیره بشه. این هم یه نکته مثبت تو گزارشه: یعنی پروژه رو مقاوم کردیم که وسط باگ محیطی زمین نخوره.