MODULE 4 · STUDY NOTES

Introducing Forecasting

Predicting the future from time series data — its patterns and pitfalls, and building forecast models with Amazon SageMaker Canvas and Amazon Forecast

Forecasting is machine learning aimed at the future: it predicts values that lie ahead by learning from historical data that carries a time component. That time element is both a gift and a burden — it lets a model learn from how values change, but it also breaks an assumption ordinary regression relies on, because points measured in sequence are correlated rather than independent. This module builds the whole picture. It defines forecasting and its use cases, then confronts the many ways raw time series data is messy — inconsistent timestamps, missing values, mismatched frequencies, outliers, seasonality, and autocorrelation — and how to clean and reshape it. It then turns to the tools: the five algorithms Amazon Forecast offers, the no-code path through Amazon SageMaker Canvas for importing data and building, evaluating, testing, and deploying a forecast model, the probabilistic quantile forecasts and accuracy metrics used to judge a model, and the Amazon Forecast service workflow with its supported domains and back testing.

LEARNING OBJECTIVES

- Describe the business problems solved by using machine learning forecasting
- Describe the challenges of working with time series data
- List the steps that are required to generate a forecast by using Amazon SageMaker Canvas
- Use Amazon SageMaker Canvas to make an in-app prediction

4.1 Forecasting overview

Forecasting predicts future values that are based on historical data, and it is an important area of machine learning precisely because so many opportunities to predict future outcomes rest on the past. Many of those opportunities involve a **time component**. That component is valuable — the model can derive meaning from how data points change over time — but it also makes **time series problems harder to handle** than other kinds of prediction. You can think of time series data as falling into two broad categories: **univariate** data has only one variable, and **multivariate** data has more than one.

TABLE 4.1 Common time series patterns

PATTERN	WHAT IT LOOKS LIKE
Trend	Values that increase, decrease, or stay the same (are stagnant) over time
Seasonal	A repeating pattern that is based on the seasons in a year
Cyclical	Some other form of a repeating pattern
Irregular	Changes over time that appear to be random or have no discernible pattern

WHERE FORECASTING IS USED

- **Sales and demand forecasts** — marketing applications such as sales forecasting and demand projections.
- **Inventory projections** — anticipating required inventory levels, often including information about delivery times.
- **Energy consumption** — determining when and where energy is needed.
- **Weather forecasts** — for governments and commercial applications such as agriculture.

Most time series datasets follow one of the four patterns above, and a single dataset can even carry more than one. Recognizing the pattern shapes every later choice — how you clean the data, which algorithm you pick, and how you read the forecast.

4.2 Processing time series data

Time series data is captured in chronological sequence over a defined period of time. Introducing time into a machine learning model has a positive impact, because the model can derive meaning from the change in data points over time. But time series data tends to be **correlated** — a dependency exists between data points. That is a problem: forecasting is a **regression** problem, and regression assumes the **independence** of data points. You must therefore develop a method for handling that data dependence, whose purpose is to increase the validity of the predictions.

THREE KINDS OF DATA FEED A FORECAST

- **Time series data** — the core measurements captured in sequence (for retail: timestamp, item, and quantity sold per period).
- **Related data** — extra series that augment the model, such as **price or promotions** (linked back by timestamp and item ID).
- **Metadata** — descriptive attributes that group results or explain predictions, such as a **brand name, category, or genre** (linked by item ID, since metadata rarely changes).

The more data you have, the better — but multiple sources bring **time and date challenges**. Timestamp formats differ and data can be incomplete; sometimes you can infer what's missing (if a month-and-day field cycles through the months and repeats after 12, you can add the year if you know when the data started). Much data is stored in **Universal Coordinated Time (UTC)** but not all of it, so check whether a timestamp is local or universal — as a general ML practice, standardize timestamps to UTC for consistency across time zones (for example, `2020-06-02T13:15:30Z`). Watch for ambiguity, too: for cars serviced at a garage, does the timestamp mark arrival, completion, pickup, or data entry? Other traps include a **timescale mismatch** (modeling hourly caloric intake when you only have daily data forces you to adjust the target timescale), having **no timestamp at all** (you may extrapolate a series from other signals such as wavelength measurements or vectors in an image), and **daylight savings time**, which differs worldwide and makes some times occur twice a year.

Missing values are common in real-world forecasting and make it harder for a model to generate a forecast. Values can be missing because there was no transaction, or because of measurement errors or a monitoring service that wasn't working. The classic retail example is an **out-of-stock** situation: if an item goes out of stock the day's sales fall to zero, and a forecast built on those zero values will be wrong. Several fill techniques exist — but they are not interchangeable.

TABLE 4.2 Techniques for handling missing values

TECHNIQUE	HOW IT FILLS THE GAP	WATCH OUT
Forward fill	Uses the last known value	Safe — only looks backward
Moving average	Averages the last known values	Smooths sudden changes
Backward fill	Uses the next known value after the gap	**Lookahead** — uses the future to compute the past; avoid it
Interpolation	Uses an equation to calculate the value	Assumes a smooth path between points
Zero fill	Leaves the value as 0	Right in retail — a missing sale means no order; investigate why

DOWNSAMPLING	UPSAMPLING
finer grain → coarser grain (e.g., hourly → daily)	*coarser grain → finer grain (e.g., daily → hourly)*
▪ For data at different frequencies, or finer than your question needs ▪ Combine values by **sum** (sales) or **average** (temperature) ▪ Your data decides the right combination	▪ The inverse of downsampling — difficult in most cases ▪ Needs extra data or domain knowledge ▪ Retail: one order per day at a set hour

Smoothing deals with outliers and anomalies. An order with an unusually large quantity — one that might never repeat — can distort a forecast, so removing such outliers is known as smoothing. You might smooth for **data preparation** (removing error values and outliers) or for **visualization** (reducing noise in a plot). Smoothing can yield cleaner data and a better model, but weigh the impact first: could smoothing compromise the model, does the model actually expect noisy data, and can you also smooth the data in production?

Seasonality is any repeating observation whose frequency is stable — hourly, daily, quarterly, or yearly; spring, summer, fall, winter; major holiday sales or the winter holiday season. Sales, for instance, tend to rise at the end of a quarter and into Q4, and consumer retail shows this even more strongly in Q4. A dataset can hold **multiple types of seasonality at once**, and you will often want to incorporate seasonality — and localized **holidays** — into the forecast.

Three related properties shape both data prep and algorithm choice. **Stationarity** describes how stable a system is: a stationary series has constant statistical properties over time, which tells you how much the past should inform the future — a low-stability system is poor for prediction. **Trends** must be handled with care, because adjusting a series for its trend makes it hard to compare against another trend-adjusted series, and dominant trends can lead you to **overestimate the correlation** between two series. **Autocorrelation** — how points in time are linearly related — is a special problem of time series: because separate observations are not independent, it behaves as noise and can make a model **overstate its accuracy**. Some algorithms in this module correct for it.

> **COMMON PITFALL**
>
> Correlations do not mean causation. Be cautious about acting on correlations that have no real-world meaning: two randomly generated series between 0 and 1 have low correlation, but give both the same slope (a shared trend) and they suddenly correlate strongly. Along with seasonality, stationarity, trends, and autocorrelation all influence which model you should select.

WORKED EXAMPLE – PANDAS FOR TIME SERIES

The **pandas** library was built with financial data analysis in mind, so it handles time series well. Set a DataFrame's index to a datetime and you can select by date, including partial-date ranges: `df['2010-01-04']` picks a day and `df['2010-02':'2010-03']` picks a range, while `df.index.weekday_name` extracts date parts. For aggregation there are built-in group and resample operations — `df.groupby('StockCode').resample('D').sum()` downsamples each item to daily sums — and `df['Quantity'].autocorr()` reports the series' autocorrelation.

4.3 Time series forecasting algorithms

One of the tasks in building a forecasting application is choosing an appropriate algorithm — the **type of dataset and its features** should determine the choice, since some algorithms handle seasonality and autocorrelation and others do not. **Amazon Forecast supports five algorithms**, and if you are unsure which fits, the **AutoML** feature tries them all and keeps the best predictor.

TABLE 4.3 The five Amazon Forecast algorithms

ALGORITHM	HOW IT WORKS AND WHERE IT FITS
ARIMA (Autoregressive Integrated Moving Average)	Removes autocorrelations that might influence the pattern of observations
DeepAR+	A supervised learning algorithm for forecasting one-dimensional time series; uses a recurrent neural network to train a model over multiple time series
ETS (Exponential Smoothing)	Useful for datasets with seasonality; uses a weighted average of all observations, with the weights decreasing over time
NPTS (Non-Parametric Time Series)	Bases predictions on sampling from past observations; specialized versions are available for seasonal and climatological datasets
Prophet	A Bayesian time series model; useful for datasets that span a long time period, have missing data, or have large outliers

COMMON PITFALL

Don't misclassify DeepAR+. Despite using a recurrent neural network, it is a **supervised** learning algorithm — it trains across multiple related one-dimensional time series. And ARIMA is the one that specifically **removes autocorrelation**, while ETS is the go-to for **seasonal** data.

4.4 Building a forecast model with Amazon SageMaker Canvas

Amazon SageMaker Canvas is a no-code visual interface for building ML models. It suits users who want to create ML models but have limited coding and cloud-infrastructure experience: it offers **ready-to-use models** for immediate predictions, lets you **build custom models**, and can even chat with popular large language models. As a managed service, Canvas provisions the needed AWS resources on your behalf and covers the workflow end to end — importing and preparing datasets, training models, and deploying them to make predictions. For datasets larger than **5 GB** or for advanced transformations, use **Amazon SageMaker Data Wrangler** to build a data flow, apply advanced transforms, or join datasets. Data can come from local files, Amazon S3, Amazon Redshift provisioned clusters (not Redshift Serverless), the AWS Glue Data Catalog through Amazon Athena, Amazon Aurora, Amazon RDS, Salesforce Data Cloud, Snowflake, JDBC databases, and more than 40 SaaS platforms. When you build your own model, Canvas trains **up to 250 models** and chooses the one that performs best.

TABLE 4.4 SageMaker Canvas custom model types

MODEL TYPE	ML EQUIVALENT	EXAMPLE QUESTION
Numeric prediction	Regression	What is the predicted price of a 2,000 sq ft house?
Categorical (two categories)	Binary classification	Will the customer churn — yes or no?
Categorical (three or more)	Multi-class classification	Is the user platinum, gold, or silver?
Time series forecasting	—	Will the company be able to fulfill shoe orders next quarter?
Image prediction	Single-label image classification	Is this an eggplant, a tomato, or a zucchini?
Text prediction	Multi-class text classification	Is this text positive, neutral, or negative?

BUILDING A CUSTOM FORECASTING MODEL – THE STEPS

- **Import your data** from sources such as Amazon Aurora, Amazon S3, or Amazon Redshift; optionally clean, format, and merge. A forecast dataset must have a **timestamp** column, a **target** column (the values you forecast), and an **item ID** column (unique identifiers).
- **Build the model** — a **quick build** takes 2–15 minutes and is reasonably accurate; a **standard build** takes 2–4 hours and is the most accurate.
- **Evaluate** the training performance and accuracy scores (wQL, MAPE, WAPE, RMSE, MASE).
- **Test** the model with batch or single predictions on a deployed test version.
- **Register** the model (optional) — a new version is added to the **Amazon SageMaker model registry**.
- **Deploy** to production — the configuration needs a model version, a deployment name, and the type and number of Amazon EC2 instances; the deployed model exposes a **URL** to invoke for predictions. The whole process can be run with SageMaker Canvas automations.

REFINING FORECASTS AND ITERATING ON THE MODEL

- **Group column** — group the forecast by each value in a column (such as an item ID) to get a separate, more specific forecast per group.
- **Holiday schedule** — add holidays (available for **251 countries**) to improve sales forecasts, since months like November and December differ sharply from January.
- **What-if scenario** — for a single item, change input values (for example, lower the price) to see how the forecast responds.
- **Model versions** — each build is a version (V1, V2, …); compare accuracy across versions, clone or create new ones, and delete those you no longer need to iteratively improve performance.

WORKED EXAMPLE – INVOKING A DEPLOYED MODEL IN PYTHON

A model deployed to a SageMaker **endpoint** is called programmatically with the runtime client. Load the payload with pandas (`data = pd.read_csv(csv_path)`), create the client (`client = boto3.client("runtime.sagemaker")`), then call `client.invoke_endpoint(...)` with the endpoint name, `ContentType="text/csv"`, the CSV body, and `Accept="application/json"`. The response carries the forecast — and testing single predictions on the production model also reports latency.

4.5 Evaluating a forecast model

Forecasts from these tools are **probabilistic**: rather than a single number, SageMaker Canvas and Amazon Forecast produce predictions at three distinct **quantiles** — 10 percent, 50 percent, and 90 percent — that show how much uncertainty is attached to each forecast. A quantile of Pn means the true value will be below the predicted value **n percent** of the time. You choose the quantile according to your business risk.

TABLE 4.5 Reading the P10 / P50 / P90 quantiles

QUANTILE	MEANING	CHOOSE IT WHEN
P10	The true value is below the forecast 10% of the time (a low, conservative estimate)	Overstock is costly — order low and accept selling out ~90% of the time
P50	The true value is below the forecast 50% of the time (the median)	Demand is moderate and overstock is not a concern
P90	The true value is below the forecast 90% of the time (a high estimate)	Understock is costly — lost revenue is high, so order high

WORKED EXAMPLE – READING A SHOE DEMAND FORECAST

A web retailer wants to predict how often it will be unable to fill orders for AnyCompany brand shoes. SageMaker Canvas forecasts demand of **1,000 pairs per month** with three quantiles: **P10** — 10% of the time fewer than **880** pairs are ordered; **P50** — 50% of the time fewer than **1,050**; **P90** — 90% of the time fewer than **1,200**. The retailer picks an inventory level from these values based on the relative cost of failing to fill orders versus holding excess stock.

Average Weighted Quantile Loss (wQL) evaluates the forecast as a whole by averaging the error at the P10, P50, and P90 quantiles — a lower value means a more accurate model, and wQL works best for models with **greater variability in the errors**. **Root Mean Square Error (RMSE)** takes the difference between each actual value and its forecast, squares those differences, averages them, and takes the square root; it represents the standard deviation of the prediction errors and is best when the errors are mostly the **same size** (few outliers). As with every forecasting metric, lower is better.

TABLE 4.6 RMSE, worked step by step

ACTUAL	PREDICTION	DIFFERENCE	DIFFERENCE SQUARED
2	4	2	4
4	8	4	16
8	13	5	25
5	9	4	16

WORKED EXAMPLE – FINISHING THE RMSE

Sum the squared differences (4 + 16 + 25 + 16 = 61), divide by the number of predictions (61 ÷ 4 = 15.25), and take the square root: √15.25 ≈ **3.905**. Because RMSE is the square root of the mean squared error, a few large misses inflate it quickly — which is exactly why it is trusted only when errors are of similar size.

TABLE 4.7 Further metrics SageMaker Canvas reports

METRIC	WHAT IT MEASURES	INTERPRETING IT
MAPE	Percent difference between mean forecast and actual, averaged over all time points	Lower is better; 0 = a perfect model
WAPE	Sum of the absolute error normalized by the sum of the absolute target	Lower is better; 0 = a perfect model
MASE	Forecast error scaled by the error of a simple baseline method	< 1 beats the baseline; > 1 is worse
Column impact	The relative impact of each input attribute on the forecast	Compared on a relative scale; larger = larger impact

COMMON PITFALL

Quantiles are not accuracy scores. P10, P50, and P90 express **uncertainty** and let you tune inventory to your risk — a P90 forecast is not "more accurate" than P10. Accuracy is judged separately, by **lower** wQL, RMSE, MAPE, WAPE, and MASE values.

4.6 Amazon Forecast

Amazon Forecast is a dedicated forecasting service that applies the same machine learning development pipeline you use throughout the course. At a high level the process is: **import** your data — as much historical and related data as you have — and do basic evaluation and feature engineering; **train a predictor** by choosing an algorithm (or letting **AutoML** choose) and selecting a **domain**; then **generate forecasts** by applying the trained model to an input dataset group. Amazon Forecast inspects the data, identifies key fields, selects and optimizes a model, and produces a predictor. You can query the resulting forecasts in the AWS console, export them to an **Amazon S3** bucket as comma-delimited files (optionally encrypted), or work with them through the API and CLI.

TABLE 4.8 Amazon Forecast supported domains

DOMAIN	WHAT IT FORECASTS
Retail	Product demand
Inventory planning	Raw materials requirements
EC2 capacity	Capacity demand for Amazon EC2
Work force	Workload projections
Web traffic	Projected traffic to one or more websites
Metrics	Metrics such as revenue, sales, or cash flow
Custom	A domain that cannot be mapped to any of the above

Selecting the right **domain improves the efficiency of the predictor**, because each domain expects specific types of data. The Retail domain, for example, expects item identifiers, a timestamp for the observation, and the number of sales for that item. If none of the predefined domains fits, choose the Custom domain.

TABLE 4.9 Data for a retail demand forecast

DATA TYPE	CONTENTS	LINKED BY
Time series	Transactional sales — timestamp, item, quantity	The core series
Metadata	Item attributes such as category or item color	Item ID (metadata rarely changes)
Related data	In-stock and promotion data, such as sales price	Timestamp + item ID

Because time series data is correlated across time, Amazon Forecast measures accuracy with **back testing** rather than a random holdout. When you import data, Forecast splits it into **training and test** sets: it trains the model on one part and tests it against the data held back. You can specify multiple **back test windows** to split the data several times, train, and compare metrics to find the best model; the default is **1**. The split is controlled by the **BackTestWindowOffset** parameter set when you create the predictor, and if you don't set it the algorithms use default values.

> COMMON PITFALL
>
> Don't validate a forecast with a random sample. In ordinary ML you hold back a random subset, but time series data is **correlated across time**, so Amazon Forecast uses **back testing** — a time-ordered train/test split — instead. Random holdouts would leak future information into training.

Module summary

- Forecasting predicts future values from historical data that carries a time component; series are univariate or multivariate, and most follow a trend, seasonal, cyclical, or irregular pattern. Common uses: sales and demand, inventory, energy, and weather.
- Time series data is captured in sequence and tends to be correlated, which breaks regression's assumption of independent points — so you must handle that dependence (autocorrelation) to keep predictions valid.
- A forecast draws on three kinds of data: the time series itself, related data (price, promotions), and metadata (brand, category) used to group or explain results.
- Data preparation dominates the work: reconcile timestamp formats and standardize to UTC; fill missing values (forward fill, moving average, interpolation, zero fill — never backward-fill lookahead); resample by downsampling or the harder upsampling; smooth outliers; and account for seasonality.
- Stationarity, trends, and autocorrelation shape both cleaning and algorithm choice; correlation is not causation, and shared trends can inflate it. The pandas library supports datetime indexing, resampling, and autocorrelation.
- Amazon Forecast supports five algorithms — ARIMA (removes autocorrelation), DeepAR+ (supervised RNN over many series), ETS (seasonal, weighted average), NPTS (samples past observations), and Prophet (Bayesian, long spans/missing data/outliers) — plus AutoML to pick the best.
- Amazon SageMaker Canvas builds forecast models with no code: import data (timestamp + target + item ID), quick or standard build, evaluate, test, register, and deploy to an endpoint; refine with group columns, holiday schedules, and what-if scenarios, and version models to iterate.

- Forecasts are probabilistic at P10/P50/P90 quantiles, chosen by over- versus under-stock risk; accuracy is judged by lower-is-better metrics (wQL, RMSE, MAPE, WAPE, MASE) plus column impact, with accuracy validated by back testing rather than a random holdout.

Review questions

1. **Why does a time component make forecasting harder than other prediction problems?**

   Answer: Time series data is captured in sequence and tends to be correlated — points depend on one another. Regression, however, assumes independent data points, so you must build a method to handle that dependence (autocorrelation), or the model's validity and accuracy suffer.

2. **Distinguish the three kinds of data used in a forecast, using a retail example.**

   Answer: The time series is the core measurement (timestamp, item, quantity sold). Related data augments it — price or promotions, linked by timestamp and item ID. Metadata describes and groups items — category, item color, or brand — linked by item ID because it rarely changes.

3. **You have gaps in your sales data. Which fill techniques are safe, which is dangerous, and when do you fill with zero?**

   Answer: Forward fill (last known value), moving average, and interpolation are safe. Backward fill uses future values to compute the past — a lookahead error to avoid. In retail, a missing sales value usually means no orders that day, so a zero fill is correct (though it's worth investigating why).

4. **What is the difference between downsampling and upsampling, and why is one much harder?**

   Answer: Downsampling moves to a coarser grain (hourly → daily), combining values by sum (sales) or average (temperature). Upsampling moves to a finer grain (daily → hourly) and is difficult — usually impossible without another data source, an irregular series to match, or domain knowledge to distribute the values.

5. **A student claims two data series are related because they correlate strongly. What should you check first?**

   Answer: Correlation is not causation. Check whether a shared trend is inflating the correlation — two random series given the same slope will correlate strongly with no real relationship — and be cautious about acting on correlations that lack real-world meaning.

6. **Match each Amazon Forecast algorithm to its strength: ARIMA, DeepAR+, ETS, NPTS, Prophet.**

   Answer: ARIMA removes autocorrelations; DeepAR+ is a supervised RNN that trains over multiple time series; ETS suits seasonal data via a time-decaying weighted average; NPTS samples from past observations (with seasonal/climatological variants); Prophet is a Bayesian model for long spans, missing data, or large outliers. AutoML tries them all.

7. **Walk through the steps to build and deploy a custom forecasting model in SageMaker Canvas.**

   Answer: Import data with timestamp, target, and item ID columns (optionally clean and merge); build with a quick build (2–15 min) or standard build (2–4 hours); evaluate accuracy scores; test with batch or single predictions; optionally register a version in the SageMaker model registry; then deploy to production (model version, deployment name, EC2 instance type/count) to get an endpoint URL to invoke.

8. **A retailer is very worried about lost revenue from understocking. Which quantile should it order on, and why?**

   Answer: P90 — the true demand falls below the P90 forecast 90% of the time, so ordering at that high estimate rarely leaves the retailer understocked. P10 would be the conservative choice when overstock is the bigger cost, and P50 (the median) when demand is moderate and overstock isn't a concern.