ALT/FNDATA ACADEMY · MODULE 9

Machine learning on auction data

This session introduces machine learning through two concrete tasks built on auction and resale records: predicting a realized price and classifying whether a lot will sell.

90 minutes · sandbox.altfndata.com

SESSION AT A GLANCE

What today looks like

- Length: 90 minutes. A 60 minute and a 120 minute variant are noted under Teaching notes.
- Level: advanced undergraduates or graduate students in an introductory machine learning, applied data science, or quantitative methods course, who have taken or are concurrently taking a first course in statistics or machine learning.
- Access: the no-code sandbox at sandbox.altfndata.com is the primary path, and is all that is needed for the core session. No API key is required. An optional Python extension, fitting a baseline regression model and a baseline classification model on an exported training extract, requires a class API key from info@altfndata.com and a standard Python data science environment.
- Goal: students leave able to separate pre-sale features from outcome fields for a price-prediction and a sold-versus-unsold task, explain target leakage in their own words, and justify a point-in-time train/test split by sale_date rather than a random split.

OBJECTIVES

By the end of class students can

- Frame a realized-price prediction task as regression and a sold-versus-unsold task as classification, and identify which documented fields are legitimate predictive features for each.
- Explain target leakage in plain terms and identify at least two ways it could enter a model built on this dataset, including using realized price to predict realized price.
- Explain why a train/test split for this data must be made by sale_date rather than at random, and describe what a random split would hide about a model's real-world performance.
- Use the sandbox to explore candidate features and export a training extract limited to fields that would be known before a sale.
- Interpret a simple accuracy or error metric for each task and identify at least one reason the metric could look good while the model is still unfit for use.
- (Extension) Load an exported training extract in Python, fit a baseline regression model and a baseline classification model, and evaluate each on a time-based holdout.

AGENDA

How the session runs

- 0 to 15 min: Introduce the two tasks, price prediction and sold/unsold classification, and preview the leakage trap that sits inside both.
- 15 to 30 min: Feature framing. As a class, sort the documented fields into "known before the sale" and "known only after the sale," and build the before/after diagram on the board.
- 30 to 45 min: The leakage and point-in-time discussion, anchored on why a random train/test split would overstate performance and why a split by sale_date is required instead.
- 45 to 60 min: Guided demo, use the sandbox to explore candidate features for one category and export a training extract limited to pre-sale fields.
- 60 to 75 min: Small-group exercise, groups sketch a feature list and a train/test split plan for a category or brand of their choosing, and identify one leakage risk specific to their plan.
- 75 to 85 min: Class discussion, groups present their feature lists and leakage risks for critique.
- 85 to 90 min: Wrap-up and homework assignment, including a note on the optional Python extension for students who want to fit an actual model.

LIVE DEMO

The in-class walkthrough

- Open sandbox.altfndata.com, sign in, and select the watches data table from the SQL editor dropdown.
- Open the schema tab and list the documented fields together as a class: designer, model, item_title, sale_date, usd_price_decimal, sale_estimates_high_usd_price, status, vendor, stock_ticker.
- Sort those fields on the board into two columns, known before the sale (designer, model, item_title, sale_estimates_high_usd_price, vendor, sale_date, category) and known only after the sale (usd_price_decimal, status). Point out that sale_estimates_high_usd_price belongs in the "before" column because a house sets its estimate ahead of the sale, while usd_price_decimal and status are outcomes.
- Run the first guided query below, which pulls a training-shaped extract using only pre-sale fields plus the two possible targets, and explain that the two targets, usd_price_decimal for regression and status for classification, must never also appear among the features for their own task.
- Introduce the point-in-time rule directly: split the exported rows so that everything before a chosen cutoff date is training data and everything on or after that date is test data, and explain that a model can only ever use information that existed at the time it would have made its prediction.
- Run the second guided query, which counts rows before and after a candidate cutoff date, and use the counts to choose a cutoff that leaves a reasonably sized test set.

LIVE DEMO

The core query

- Run this in the sandbox SQL editor:

```
SELECT designer, model, item_title, vendor, sale_date,
       sale_estimates_high_usd_price,
       usd_price_decimal,
       status
FROM all_watches_data
WHERE sale_estimates_high_usd_price > 0
ORDER BY sale_date;
```

DISCUSS

Questions for the room

- Why does using usd_price_decimal as a feature to predict usd_price_decimal count as leakage, even though it feels like an obviously circular mistake to name that way?
- sale_estimates_high_usd_price is set by the auction house before the sale. Is it a legitimate feature for both tasks, and what would make it less reliable as a predictor for a lot type the house rarely handles?
- What specifically does a random train/test split hide about a model's performance that a date-based split reveals?
- If a classification model achieves 90 percent accuracy predicting sold versus unsold, what question would you ask before trusting that number, given that most lots in many categories do sell?
- How might designer or vendor identity leak information about the outcome in a subtler way than an obvious target field, and how would you check for that?

HOMEWORK

This week's assignment

- Each student selects one category table and drafts a one-page modeling plan covering both tasks:
- predicting usd_price_decimal for sold lots and classifying status as sold or unsold.
- The plan must list the exact features to be used for each task, drawn only from documented fields known before a sale, state the train/test cutoff date chosen and the row counts on each side of it (using a query like query 2), and include a short paragraph identifying one specific leakage risk in the plan and how it was avoided.
- Students who complete the optional Python extension may additionally submit a notebook that fits the two baseline models on an exported extract and reports a basic error or accuracy metric on the time-based test set; this is optional and not required for full credit.
- Grading criteria:

NEXT

Wrapping up

- Restate the one-sentence takeaway before students leave.
- Point to the facilitator guide for discussion guidance and teaching notes.
- Questions or a class API key: info@altfndata.com