ALT/FNDATA ACADEMY · PROBLEM SET

Problem set 9: Machine learning on auction data

Module 9. Total: 100 points. Format: no-code sandbox, with an optional API and code extension in Problem 5 using a shared class API key from your instructor.

Before you start

This problem set works with the documented fields on one category table (designer, model, item_title, sale_date, usd_price_decimal, sale_estimates_high_usd_price, status, vendor, stock_ticker) to frame two modeling tasks, predicting a realized price and classifying whether a lot sells, without fitting an actual model in the required portion. Choose one category table and, optionally, one brand or house within it to narrow your examples. The discipline this problem set tests is the same discipline a working analyst needs before touching a single line of modeling code: knowing which fields would have existed at the moment of prediction, and refusing to let the outcome leak into the inputs.

Problem 1 (20 points): Features versus leakage

List the documented fields on your chosen table, and sort them into two groups: fields that would be known before a sale (legitimate features) and fields that are known only after a sale (possible targets, not features). Then, in three to four sentences, explain what target leakage means in plain terms and give one specific example of how it could enter a model built on this table, including the case of using usd_price_decimal to predict usd_price_decimal.

Problem 2 (20 points): A point-in-time train and test split

Write a SQL query against your chosen table that labels each row train or test based on whether sale_date falls before or on/after a cutoff date you choose, and counts the rows on each side. Then, in three to four sentences, explain why this split must be made by sale_date rather than by randomly assigning rows to train or test, and describe specifically what a random split would hide about a model's real-world performance.

```
-- your query here
```

Problem 3 (20 points): Export a training extract

Write a SQL query against your chosen table that produces a training-shaped extract: only pre-sale fields (drawn from your Problem 1 list) plus the two possible targets, usd_price_decimal and status, ordered by sale_date. Restrict to rows with a valid, non-zero high estimate. In one or two sentences, state which rows in this extract you would use for the price-regression task and which you would use for the sold/unsold classification task, and why those two subsets differ.

```
-- your query here
```

Problem 4 (20 points): Regression or classification, and reading the metric

In 200 to 300 words, frame the two tasks side by side: predicting usd_price_decimal as a regression problem, and predicting status (sold or unsold) as a classification problem. For each task, name one legitimate pre-sale feature from your Problem 1 list and explain why it is legitimate. Then explain one specific reason a classification accuracy figure for the sold/unsold task could look strong while the model is still unfit for use, referencing what you would need to check about the class balance of sold versus unsold lots before trusting that figure.

Problem 5 (20 points): Baseline models on a time-based split (optional code extension)

Using the shared class API key your instructor has provided, write the endpoint and JSON request body you would send to POST /v1/tables/{name}/query to retrieve the same training extract you built in Problem 3. Then, describe in a short paragraph, or in actual Python if you complete the optional code track, how you would load the exported rows, split them by your Problem 2 cutoff date, and fit one baseline regression model to predict usd_price_decimal for sold lots and one baseline classification model to predict status, using only the pre-sale features from Problem 1.

```
POST /v1/tables/<table_name>/query
Header: X-API-Key: <class key>
Body:
{
  ...
}
```

Submission. Turn in this file with your SQL, JSON body, feature lists, and written answers filled in.