Machine Learning in Finance: Mock exam 2

Duration: 180 minutes. Part 1 and Part 2 are worth 50 points each. Work locally in Jupyter with downloaded course materials and personal notes. Internet and AI are not allowed.

Submission. Put all code and written answers in the answer notebook. Save it as `MLFIN_MOCK_2026_2_answers.ipynb`. Submit that notebook together with the supplied `MLFIN_MOCK_2026_2_blank_submission.pdf`. The PDF is intentionally blank; do not put answers in it.

The supplied `MLFIN_MOCK_2026_2_code_solutions.ipynb` is for self-checking. It contains code only; written explanations and answers to explanation-only questions are intentionally omitted.

Partial solutions can earn points. Show completed steps even if unfinished. Later Part 2 work that correctly follows an earlier answer can earn marks; state your assumptions. If a question tells you to use a particular method, use it. Otherwise, any correct method is accepted.

Part 1 (50 points)

Part 1 contains four independent programming problems. Each block supplies the inputs and definitions needed for that block; its later subparts may use your earlier result in the same block.

(A) **Count late loan payments (10 points)**

`payments` is a dictionary: each key is a borrower name, and its value is a list of whole-number days late for that borrower's successive payments. A zero means on time. For a chosen nonnegative `limit`, count an entry only when its days-late value is **strictly greater** than `limit`. For example, with `limit=1`, an entry of 1 is not counted but an entry of 2 is. An empty list has count zero.

```
payments = {
    'Ava': [0, 2, 4],
    'Bo': [1, 1, 0],
    'Cia': [3, 0, 2],
    'Dan': [],
}
```

(i) **7 points.** Write `late_counts(payments,limit)` using loops. It must return a new dictionary mapping every borrower to that borrower's late-payment count. Do not change `payments`. Call it with `limit=1` and print the dictionary.

(ii) **3 points.** Print the borrower with the largest count from your result and that count. If counts tie, choose the name that comes first alphabetically.

(B) **Repair a bank-balance path (13 points)**

A bank account starts with `start` euros. The integers in `changes` are deposits (positive) and withdrawals (negative), in order. `balance_path(start,changes)` must return a list beginning with the starting balance and then each updated balance. It must stop **immediately after** appending the first negative balance, ignoring later changes. Assume `start` is nonnegative. The supplied function runs but gives the wrong path.

```
def balance_path(start, changes):
    path = [start]
    for change in changes:
        balance = start + change
        path.append(balance)
    return path
```

```
start = 100
changes = [-20, -30, 10, -80, 50]
```

(i) **4 points.** Identify the two separate logic errors in the supplied function. State the correct path for the supplied input, including the starting balance.

(ii) **9 points.** Write the corrected complete `balance_path` function and call it on the supplied input. Print its returned list. Your function must also work when no change makes the balance negative.

(C) **Portfolio size versus portfolio value (12 points)**

A portfolio has four stocks. For stock $i = 1, \ldots, 4$, `shares[i]` is its share count and `prices[i]` is its euro price per share. Define its euro value $v_i = s_i p_i$, where s_i and p_i are those two inputs. Let $L = \{i : v_i \geq 200\}$ be the holdings worth at least 200 euros. Define

$$\text{count share} = \frac{|L|}{4}, \qquad \text{value share} = \frac{\sum_{i \in L} v_i}{\sum_{i=1}^{4} v_i}.$$

Here $|L|$ is the number of large holdings.

```
import numpy as np
shares = np.array([4, 10, 5, 8])
prices = np.array([50, 20, 80, 10])
```

(i) **7 points.** Complete all four `...` blanks and write the completed code in your answer cell. It must print the count share and value share of large holdings, both rounded to three decimals.

```
values = ...
large = ...
count_share = large.sum() / ...
value_share = values[...].sum() / values.sum()
print(round(float(count_share), 3), round(float(value_share), 3))
```

(ii) **5 points.** State the two printed shares and explain why the value share is higher than the count share in this portfolio. Your explanation must use the supplied holdings, not only the general definitions.

(D) **Align loan repayments with months (15 points)**

`months` lists five successive month ends and `balance` gives the outstanding euro loan balance at each month end. Between the preceding and current month ends, the amount repaid in month i is

$$r_i = b_{i-1} - b_i, \qquad i = 1, 2, 3, 4,$$

where b_i is `balance[i]`. Label r_i with `months[i]`, the month when that repayment appears. All balances here decrease.

```
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']
balance = [1000, 900, 750, 600, 450]
```

(i) **5 points.** Using a loop, build and print `repaid`, the four monthly repayments in month order.

(ii) **5 points.** Build and print `repayment_by_month` as a list of `(month,euros_repaid)` pairs in the correct month order. In one sentence, explain why using `months[:-1]` would give the wrong labels.

(iii) **5 points.** Print the earliest month with the largest repayment and that amount. If several months tie, choose the earliest. Also print whether the sum of `repaid` equals the first balance minus the last balance.

Part 2 (50 points)

A risk team monitors stocks and wants to know which ones may be especially volatile next month. Use a **DecisionTreeRegressor** to forecast each stock's next-month realised volatility, reported as an annualised percentage. In the simulated CSV, each row is a different stock observed on the same date. It records recent volatility and return, market value, trading turnover, and market beta. `next_vol_pct` is the observed next-month volatility; `stock_id` identifies the stock.

Column	Meaning
`stock_id`	Unique stock label; not a predictor.
`vol_20d_pct`	Annualised volatility over the previous 20 trading days, in percent.
`ret_20d_pct`	Return over the previous 20 trading days, in percent.
`market_cap_bn`	Company market value in billions of euros.
`turnover_pct`	Average daily shares traded as a percentage of outstanding shares.
`market_beta`	Sensitivity to broad-market moves; a dimensionless number.
`next_vol_pct`	Target: next-month realised volatility, annualised percent.

(A) **Prepare the stock forecasts (13 points)**

For a stock, let r_i be its past 20-day return in percent. Add an absolute-return predictor $a_i = |r_i|$, also in percent. Use 25% of stocks as test and the rest as train. Mean squared error (MSE) is the average squared forecast error, in squared percentage points.

(i) **2 points.** Load the CSV into `df`. Print its shape and the first three rows.

(ii) **4 points.** Add `abs_ret_20d_pct` to `df`. Print its first five values to one decimal and count stocks for which it is at least 5%.

(iii) **4 points.** Make `X` from prior volatility, absolute return, market value, turnover and market beta, in that order; make `y` from next-month volatility. Split into train and test with 25% test and `random_state=17`. Print both row counts.

(iv) **3 points.** The reference forecasts the train mean target for every test stock. Calculate and print its test MSE to two decimals.

(B) **Choose the amount of tree detail (10 points)**

For each tree, use `min_samples_leaf=20` and `random_state=31`. Compare `max_depth` values 2, 4, and 7 by mean val MSE over three shuffled train/val folds (seed 23). Choose the lowest mean; if tied, choose the shallower tree.

(i) **7 points.** Calculate and print the three mean val MSEs, labelled by depth and rounded to two decimals.

(ii) **3 points.** Choose the depth by the stated rule. Fit `tree` with that depth and the fixed settings on all train rows. Print the depth.

(C) **Scores and a residual picture (12 points)**

For test stock i, let y_i be actual and $\hat{y}_i$ forecast next-month volatility, both in annualised percentage points. Define the residual

$$e_i = y_i - \hat{y}_i.$$

(i) **5 points.** Print train MSE, test MSE, and the simple reference's test MSE, each to two decimals. State whether the fitted tree beats the reference on test rows and whether its train error is below its test error.

(ii) **7 points.** Plot test residuals on the vertical axis against tree test forecasts on the horizontal axis. Add a zero line, a title, and axis labels with units. In one sentence, compare the vertical spread at low and high forecasts.

(D) **Compare low and high forecasts (7 points)**

Let m be the median test forecast. The low group has $\hat{y}_i \leq m$ and the high group has $\hat{y}_i > m$. For either group G, define its mean absolute error by

$$\mathrm{MAE}(G) = \frac{1}{|G|} \sum_{i \in G} |y_i - \hat{y}_i|.$$

MAE is in percentage points.

(i) **4 points.** Calculate and print the test MAE and stock count separately for the low and high forecast groups, with MAE to two decimals.

(ii) **3 points.** Which group has higher test MAE? Relate this to the plot and explain why overall test MSE does not show the contrast.

(E) **A risk-manager note (8 points)**

The desk may use a high forecast to decide where to spend extra risk-review time.

(i) **8 points.** Advise the risk desk in your own words. Use the tree and reference test MSEs, both group MAEs, and the residual plot. Give one limitation of evidence from stocks observed on one date. Write at most five sentences.