Machine Learning in Finance: Mock exam 3

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_3_answers.ipynb`. Submit that notebook together with the supplied `MLFIN_MOCK_2026_3_blank_submission.pdf`. The PDF is intentionally blank; do not put answers in it.

The supplied `MLFIN_MOCK_2026_3_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) **Bond cashflow gains (12 points)**

`price_eur` gives the euro purchase price P_n of one unit of bond n. `payments_eur[n]` gives its three future euro payments p_{n1}, p_{n2}, p_{n3}. Define the simple gain, ignoring payment timing, by

$$G_n = \sum_{j=1}^{3} p_{nj} - P_n.$$

The dictionary keys are bond names.

```
price_eur = {'Bond A': 102, 'Bond B': 99, 'Bond C': 105, 'Bond D': 98}
payments_eur = {
    'Bond A': [4, 4, 104], 'Bond B': [3, 3, 103],
    'Bond C': [2, 2, 102], 'Bond D': [5, 5, 105],
}
```

(i) **6 points.** Build `gain_eur`, a dictionary mapping each bond name to its simple gain. Print it.

(ii) **6 points.** Select bonds whose gain is at least 10 euros. Sort the selected names by descending gain; break a tie by alphabetical bond name. Print each name and gain in that order.

(B) **A portfolio with winning stocks (12 points)**

The three positions have `units[i]` shares. `before[i]` and `after[i]` are euro prices per share at the start and end of one period. For stock i, define $r_i = \text{after}_i/\text{before}_i - 1$; it is a winner when $r_i > 0$. The portfolio's start and end values are $V_0 = \sum_i u_i b_i$ and $V_1 = \sum_i u_i a_i$, where u_i is units, b_i is before, and a_i is after. Its return is

$$R = \frac{V_1}{V_0} - 1.$$

```
import numpy as np
units = np.array([10, 20, 5])
before = np.array([10, 20, 40])
after = np.array([11, 18, 44])
```

(i) **6 points.** Use NumPy to make a Boolean array `winner` and print it. Print the mean price return of winning stocks to three decimals.

(ii) **6 points.** Calculate the portfolio's total start and end values and its return. Print the return to three decimals. Explain why the portfolio loses money although two stocks win.

(C) **Interest on a bank balance (12 points)**

For a nonnegative bank balance b in euros, annual interest $I(b)$ in euros is 2% of the first 1,000 euros and 1% of any amount above 1,000 euros:

$$I(b) = \begin{cases} 0.02b, & 0 \le b \le 1000, \\ 20 + 0.01(b - 1000), & b > 1000. \end{cases}$$

(i) **6 points.** Complete the three blanks so `interest(balance)` returns one float for the stated rule. Write the completed function.

```
def interest(balance):
    if ...:
        return float(...)
    return float(...)
```

(ii) **6 points.** Using a loop over whole-euro balances from 0 through 5,000, find the smallest balance with annual interest at least 35 euros. Print the balance and its interest. Stop the loop at the first qualifying balance.

(D) **Variation in fund-unit prices (14 points)**

The array `nav` gives one fund unit's euro price on five consecutive days. Let `changes` be the four day-to-day changes in euros. Their sample variance is $s^2 = \sum_{i=1}^{4}(c_i - \bar{c})^2/(4-1)$, where $\bar{c}$ is their mean.

```
import numpy as np
nav = np.array([100, 101, 100, 103, 105])
```

(i) **7 points.** Make and print the four-element array `changes`. Calculate the sample variance directly from the displayed formula and print it to six decimals.

(ii) **7 points.** Calculate the mean squared deviation of `changes` using `np.mean`, then multiply it by 4/3 to obtain a second sample-variance calculation. Print both sample variances to six decimals and print whether they agree to absolute tolerance `1e-12`. In one sentence, explain the factor 4/3.

Part 2 (50 points)

You work with a lender's collections team. After sending a payment reminder, the team needs to decide when to follow up. It asks you to use **LinearRegression** to forecast how many days each loan will take to be paid. The simulated CSV has one reminded loan per row. It records the amount due (`loan_amount_eur`), the borrower's monthly income (`monthly_income_eur`), how late their previous payment was (`prior_late_days`), whether automatic payment is active (`autopay`), and the loan term (`term_months`). The outcome, `days_to_payment`, counts days from the reminder to payment. `loan_id` identifies the loan.

Column	Meaning
`loan_id`	Loan identifier
`loan_amount_eur`	Amount due, euros
`monthly_income_eur`	Borrower's monthly income, euros
`prior_late_days`	Days late on the borrower's previous payment
`autopay`	1 if automatic payment is enabled, 0 otherwise
`term_months`	Loan term, months
`days_to_payment`	Days from this reminder until payment; target

(A) **Prepare the loan data (12 points)**

For loan i, let A_i be `loan_amount_eur` and M_i be `monthly_income_eur`. Add the dimensionless ratio

$$R_i = \frac{A_i}{M_i}.$$

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

(ii) **3 points.** Add `loan_to_income` to `loans` using the displayed formula. Print its first five values to three decimals.

(iii) **4 points.** Make `X` from loan amount, monthly income, prior late days, autopay, term and `loan_to_income`, in that order; make `y` from `days_to_payment`. Divide rows randomly into train (60%), val (20%), and test (20%), using seed 17 for reproducibility. Print the train, val and test row counts.

(iv) **3 points.** The reference forecasts the train mean `days_to_payment` for every val loan. Calculate and print its val MAE to two decimals.

(B) **Fit and validate a linear forecast (11 points)**

Use `LinearRegression` with its default settings.

(i) **7 points.** Fit the linear model on train rows. Predict on val rows and print its val MAE to two decimals. Say which val MAE is lower, this model's or the reference's.

(ii) **4 points.** Refit the same estimator on train and val rows combined. Predict on test rows. Print the first five test forecasts to two decimals and the number of forecasts.

(C) **Check errors and a predictor relationship (16 points)**

For a set G of test loans, let y_i be actual days to payment and $\hat{y}_i$ the model forecast. Define

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

Compare `autopay=0` and `autopay=1`. The final test reference forecasts the mean target over train and val rows combined.

(i) **6 points.** Calculate and print overall test MAE for the final linear model and the reference mean over train and val combined, both to two decimals. For each `autopay` value 0 and 1, print the number of test loans and the linear model's test MAE to two decimals.

(ii) **6 points.** Plot `prior_late_days` horizontally and `days_to_payment` vertically for train and val rows combined. Distinguish `autopay=0` and `autopay=1` by colour or symbol. Include a legend, a title, and axis labels with day units.

(iii) **4 points.** Describe the main direction of the relationship in your plot and one visible difference between the two `autopay` groups. State why the plot alone cannot show that switching on automatic payment causes faster payment.

(D) **Explain the forecast to the loan team (11 points)**

The team uses forecasts to decide which payments need earlier follow-up. Work from the fitted model and the measured test results.

(i) **5 points.** Print the final fitted coefficient for `autopay` to two decimals. Explain its unit and what its sign and size mean when the other listed predictors are held fixed.

(ii) **6 points.** In your own words, assess whether the linear forecasts would help the loan team plan follow-up. Use the overall test MAEs, the two autopay-group MAEs, and the coefficient or plot to support your view. Give one limitation of this single split in at most five sentences.