Machine-Learning in Finance
Mock Exam 2025

Exercise 1: Using random features with Monte-Carlo simulation to get a derivative pricer

For complex derivatives, we cannot obtain a closed-form solution for their prices. We can rely on a Monte-Carlo simulation to estimate its price in such cases. In this exercise, we will try to create a model that can predict the derivatives' price solely from their underlying parameters using random features. You should complete the following tasks by replacing the *#TODO* with the appropriate code:

1. **Load the data** from *data/exercise1.csv*. It contains 8 columns: the first 7 are features (e.g., initial price, volatility, barrier, etc), and the last column (price) is the target.

2. **Split the data** into train/test sets (80 % train, 20 % test).

3. **Fit a linear model** using only the 7 original features. Report the in-sample and out-of-sample Mean Squared Error (MSE).

4. **Create random features**: for each $k \in \{1, 10, 20, 50, 75, 100, 1000\}$, generate k new features via

$$X^{\mathrm{RF}} = \mathrm{ReLU}(XW), \quad W \sim \mathcal{N}(0,1) \in \mathbb{R}^{d \times k}.$$

5. **Run Ridge regression** on the augmented dataset for each k and each regularization weight $z \in \{0.1, 1, 10\}$. Report both in-sample and out-of-sample MSE.

6. **Plot Heatmap** of the out-of-sample MSE where the X axis is the ridge-regularization and the Y axis is the number of Random Features.

Exercise 2: Replicating payoff with MLP

According to the static replication theorem, any sufficiently smooth payoff at time T of an underlying can be replicated by a continuum of vanilla options with maturity T, a position in the underlying and zero-coupon bonds maturing at T. In this exam, we'll approximate this using a finite set of strikes and train an MLP to output the appropriate weights for these options given the underlying price. Mathematically, we have:

$$f(S_T) = (f(F) - f'(F)F) + f'(F)S_T + \int_0^F f''(K)(K - S_T)^+ dK + \int_F^\infty f''(K)(S_T - K)^+ dK$$

You should complete the following tasks by replacing the *#TODO* with the appropriate code:

1. **Load the data** from *data/exercise2.csv*. It contains 10 columns:
   - Column 1: underlying price at maturity S_T
   - Columns 2–9: payoffs of calls and puts at maturity
   - Column 10: target payoff $f(S_T)$
2. **Check for NaNs and outliers** using a Whiskey plot.
3. **Split the data** into train and test sets (80% / 20%).
4. **Build an MLP** in PyTorch to learn the optimal replication weights. Your model should be :

$$\hat{f}(S_T) = w_0\, S_T + \sum_{i=1}^{4} w_i\, \max(S_T - K_i, 0) + \sum_{i=5}^{9} w_i\, \max(K_i - S_T, 0).$$

5. **Evaluation**: report the replication MSE on the test set and plot true vs. replicated payoffs.

Exercise 3: Predicting Analyst Reactions from Textual Embeddings

In this exercise, you will build a simple predictive model that maps the language of earnings-related statements to analyst reaction scores.

Each sentence - drawn from earnings calls or financial reports - is paired with a numerical target representing the magnitude and direction of analyst response.

We model this relationship as:

$$y_i = f(x_i) + \varepsilon_i \tag{1}$$

where:

- x_i is the embedding of the i-th sentence,
- y_i is the corresponding analyst reaction score,
- ε_i is a noise term capturing unmodeled variation.

You will use a pretrained transformer model to extract *mean-pooled* sentence embeddings - i.e., by averaging token embeddings across each sentence. These fixed-length vectors will serve as inputs to a linear regression model trained to predict analyst responses.

The notebook includes several **TODO** sections to guide your implementation. Specifically, you will:

1. Tokenize the input sentences using the transformer tokenizer,
2. Extract mean-pooled sentence embeddings,
3. Train a linear regression model (Please use the sklearn function) on the provided data,
4. Apply the trained model to a new sentence to generate a prediction.

By the end of the exercise, you will compute the predicted analyst reaction to a new example sentence, evaluating how well your model captures the relationship between financial language and market expectations.