Practical – Development/Programming Assessment

Course: Fullstack AI – Batch - 4

You have to program/Develop : 1 question from Section A and 1 question from section B

==========

Section A: Machine Learning [Marks 30]

Question 1:

This dataset contains Real Estate listings in the US broken by State and zip code.

The dataset has 1 CSV file with 10 columns -

1. realtor-data.csv (2,226,382 entries)
   - brokered by (categorically encoded agency/broker)
   - status (Housing status - a. ready for sale or b. ready to build)
   - price (Housing price, it is either the current listing price or recently sold price if the house is sold recently)
   - bed (# of beds)
   - bath (# of bathrooms)
   - acre_lot (Property / Land size in acres)
   - street (categorically encoded street address)
   - city (city name)
   - state (state name)
   - zip_code (postal code of the area)
   - house_size (house area/size/living space in square feet)
   - prev_sold_date (Previously sold date)

NB:

1. `brokered by` and `street` addresses were categorically encoded due to data privacy policy
2. `acre_lot` means the total land area, and `house_size` denotes the living space/building area

Acknowledgements

Data was collected from -

- https://www.realtor.com/ - A real estate listing website operated by the News Corp subsidiary Move, Inc. and based in Santa Clara, California. It is the second most visited real estate listing website in the United States as of 2024, with over 100 million monthly active users.

Inspiration

- Can we predict housing prices based on the features?
- How are housing price and location attributes correlated?
- What is the overall picture of the USA housing prices w.r.t. locations?
- Do house attributes (bedroom, bathroom count) strongly correlate with the price? Are there any hidden patterns?

Tasks & Deliverables

1. **Data Access & Integrity**
    - Verify file size, schema, and encoding. Document the exact file(s) used.
    - Check basic completeness: % of missing values in `price`, `house_size`, `bed`, `bath`, `acre_lot`, `city`, `state`, `zip_code`. Identify if there are regional biases or heavy-tailed distributions (e.g., luxury segments).
2. **EDA (Exploratory Data Analysis)**
    - **Target inspection:** Distribution of `price` (consider log-transform if highly skewed).
    - **Feature relationships:** Pairwise plots and correlation heatmaps among `price`, `bed`, `bath`, `house_size`, `acre_lot`.
    - **Geographic variation:** Price by `state` and by `zip_code` (top 50 zips with highest median price).
    - **Market status:** Compare `for_sale` vs `off_market`/`sold` if present and decide whether to filter or encode `status`.
3. **Preprocessing**
    - Handle missing values: impute or drop with rationale (e.g., impute `house_size` with regional medians).
    - **Outliers:** Detect extreme `price`, `acre_lot`, `house_size` values via IQR/quantiles per state; decide on capping or exclusion.
    - **Categorical encoding:** Encode `state` and (optionally) high-cardinality `zip_code` (e.g., target encoding or hashing).
    - **Scaling:** Standardize/normalize continuous features if using linear or distance-based models.
4. **Feature Engineering**
    - Create features such as **`price_per_sqft`** (if `house_size` available), **`lot_sqft`** (acre→sqft), **age since `prev_sold_date`**, and interaction terms (e.g., bed×bath).
    - Aggregate **geospatial signals**: state-level or zip-level medians of price/size to capture neighborhood effects.

5. **Modeling**
    - Train at least **three** regressors:
        - **Regularized Linear** (Ridge/Lasso)
        - **Tree-based** (Random Forest / XGBoost / LightGBM)
        - **Elastic Net or Gradient Boosting**
    - Use **time-aware splits** if leveraging `prev_sold_date`; otherwise stratified K-fold on price quantiles to stabilize evaluation.
    - Implement a **scikit-learn** **`Pipeline`** + **`ColumnTransformer`** to avoid leakage.
    - Perform **hyperparameter tuning** via `GridSearchCV` or `RandomizedSearchCV`.
6. **Evaluation**
    - Report **RMSE**, **MAE**, and **R^2** on a held-out test set.
    - Compare models and justify the final choice.
    - Provide **feature importance** (permutation importance or SHAP) for interpretability.
7. **Robustness & Deployment Readiness**
    - Stress-test on different sub-markets (e.g., CA, NY, TX) to check stability.
    - Document assumptions (e.g., log-price modeling) and any ethical considerations (pricing fairness across regions).
    - Package the best pipeline for inference and show example predictions.

Deliverables:

- Clean, reproducible **script** (pandas, NumPy, scikit-learn, Seaborn).
- Short **report** summarizing EDA insights, modeling choices, metrics, and recommended production pipeline.

Or

Question 2:

You are analyzing the European used-car market using large, real listings datasets from the Netherlands and pan-European sources. These datasets include technical specification fields (brand, model, year, engine power/capacity, mileage, gearbox, fuel type), enriched equipment embeddings, and market attributes that are well-suited to multiple classification objectives

Used Cars Market — Dutch listings (334,851 rows, 32 features)
Why this one? It's **cleaned, standardized, and ML-ready**, with rich automotive attributes (brand/model, power, engine capacity, fuel, transmission, weight, emissions), plus **TF-IDF equipment embeddings** (A1–A4) computed per (Brand, Model, YearBuilt) that are excellent for classification

🔧 Problem Statement (Classification)

Business Goal:
Build a model to **classify the vehicle's body type** (e.g., *Hatchback, SUV, Saloon/Sedan, Estate/Wagon, Coupe, Convertible, MPV/Van*) **from structured technical features and equipment embeddings**. This helps marketplaces and dealers **auto-tag listings**, improve search relevance, and standardize catalog metadata across Europe.

Target:

- `body_type` (multi-class classification).
  Present in the Dutch dataset; names may vary slightly per dataset.

Predictor Features (examples available in datasets):

- Technical specs: `Brand`, `Model`, `YearBuilt`, `Power (kW/HP)`, `EngineCapacity (L)`, `EmptyWeight`, CO_2`Emission`, `Gears`, `Drive type`, `Fuel type`, `Transmission`.
- Usage: `Mileage`, `Previous owners`.
- **Equipment embeddings:** `A1–A4` (Comfort/Convenience, Media, Safety/Security, Extras) converted via **context-aware TF-IDF** within (Brand, Model, YearBuilt).

□ Tasks

1. **Data Access & Understanding**
   - Download and load the dataset(s). Verify schema, row counts, and feature availability. Document any field differences between the Dutch and pan-European sets.
2. **EDA (Exploratory Data Analysis)**
   - Distribution of `body_type` classes; identify long-tail classes and class imbalance.
   - Relationships between `body_type` and **engine/power/weight/mileage**; inspect role of **equipment embeddings** w.r.t. body type (e.g., Safety features richer in SUVs).
   - Market segments by **brand/model/year**—spot whether certain brands dominate specific body types.
3. **Preprocessing**
   - **Missing values:** Impute numericals (median by Brand–Model–Year); impute categorical via most frequent per segment or "Unknown."

- **Categorical encoding:** One-hot or target-encode high-cardinality features (`Brand`, `Model`).
- **Scaling:** Standardize continuous features (power, engine size, emissions, weight, mileage).
- **Class imbalance:** Apply **stratified splits**, and consider **class-weighting** or **SMOTE** where appropriate.

4. **Feature Engineering**
   - Interactions: `Power/Weight ratio`, log-transform `Mileage`, age of vehicle (`current_year - YearBuilt`).
   - Region signals (if using pan-Europe dataset): derive **country/market** from `location` if available; analyze cross-region generalization.
   - Use **A1–A4 equipment embeddings** directly as numeric features; optionally reduce dimensionality (PCA) to capture latent equipment clusters.
5. **Modeling**
   - Train and compare **at least three** classifiers:
     - Linear baseline (Logistic Regression with class weights)
     - **Tree-based** (RandomForest or LightGBM/XGBoost)
     - **Regularized** (ElasticNet or linear SVM)
   - Wrap preprocessing + model in a **scikit-learn `Pipeline`** to avoid leakage.
6. **Evaluation**
   - Report **macro-F1**, **weighted-F1**, **accuracy**, and **per-class precision/recall** (focus on minority classes like *Coupe/Convertible*).
   - Conduct **cross-validation**; show **confusion matrix**.
   - **Feature importance** / **SHAP** to explain predictors driving each body type.
7. **Robustness & Generalization**
   - **Out-of-market test:** Train on **Dutch dataset**, test on **pan-Europe** (or UK subset) to assess generalization across countries.
   - Analyze performance drift by **model year** and **brand segment**.
8. **Deployment Readiness**
   - Export the best pipeline; include inference examples for raw listing JSON → predicted `body_type`.
   - Document assumptions and limitations (e.g., dependence on equipment embedding quality).

🎁 Expected Deliverables

- Clean, reproducible **Python notebook/script** (pandas, NumPy, scikit-learn, Seaborn/Matplotlib).
- Short **report** covering EDA insights, modeling choices, metrics (macro/weighted F1), cross-market generalization, and recommended production pipeline.

About Dataset

```
Data columns (total 32 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   Brand                      334851 non-null  object
 1   Model                      334851 non-null  object
 2   BodyType                   334851 non-null  object
 3   Category                   334851 non-null  object
 4   Drive                      332953 non-null  object
 5   Doors                      334851 non-null  int64
 6   Km                         334851 non-null  int64
 7   YearBuilt                  334851 non-null  int64
 8   Power_kW_HP                334851 non-null  int64
 9   EngineCapacity             334851 non-null  float64
 10  Cylinders                  334327 non-null  float64
 11  EmptyWeight                334178 non-null  float64
 12  EmissionClass              334452 non-null  object
 13  FuelType                   334727 non-null  object
 14  Price                      334851 non-null  int64
 15  Seats                      334851 non-null  float64
 16  FullServiceHistory         334851 non-null  bool
 17  Transmission               334851 non-null  object
 18  Gears                      239814 non-null  float64
 19  CO2Emission                330416 non-null  float64
 20  Color                      334851 non-null  object
 21  Warranty                   334851 non-null  object
 22  APK                        213311 non-null  object
 23  FuelConsumption            46982 non-null   float64
 24  PreviousOwners             100446 non-null  float64
 25  NonSmokingCar              334851 non-null  bool
 26  APK_month_diff             334851 non-null  int64
 27  EngineCapacity_Corrected   334851 non-null  float64
 28  A1                         334851 non-null  float64
 29  A2                         334851 non-null  float64
 30  A3                         334851 non-null  float64
 31  A4                         334851 non-null  float64
dtypes: bool(2), float64(13), int64(6), object(11)
memory usage: 77.3+ MB
```

Used Cars Market — 334,851 Listings (2006–2018)

Cleaned, enriched and machine-learning-ready dataset

This dataset contains **334,851 real used-car listings** from the Dutch used-car market.
All records were collected **manually from publicly available listings** and then cleaned, standardized and enriched for analytics and Machine Learning.
No automated scraping tools were used.

It includes **32 high-quality features**, engineered for price prediction, automotive analytics, and clustering.

🚗 Dataset Overview

- **334,851 listings**
- Years **2006–2018**
- **32 structured variables**
- Detailed technical specifications
- TF-IDF equipment embeddings
- Extensive cleaning + enrichment pipeline
- ML-ready format with minimal missing data

This dataset provides a **deep and accurate** picture of the used-car market in the Netherlands.

🔥 Key Features

✓ High-quality automotive attributes

- Brand, Model, YearBuilt
- Power (kW / HP)
- Engine capacity (L)
- Cylinders
- Vehicle weight
- Fuel type & consumption
- Transmission, gears, drive type
- Color, emissions, previous owners

✓ Enriched technical data

Missing values were reconstructed using grouped statistical mapping across:

- **Brand → Model → Year → Power → EngineCapacity**
- **Brand → Model → Year → Seats**
- **Brand → Model → Year → Drive type**
- **EngineCapacity → CO_2Emission**
- **Model → Year → EmptyWeight**
- Many other hierarchical combinations

  This makes the dataset **far more complete** than typical automotive datasets.

  ✓ Cleaned and standardized

- Major outliers removed or corrected
- Impossible values fixed
- Normalized numerical scales
- Empty strings converted to NaN
- Uniform formatting across all fields

□ TF-IDF Equipment Embeddings (A1–A4)

The dataset originally contained four text-heavy equipment columns:

- `ComfortConvenience`
- `EntertainmentMedia`
- `SafetySecurity`
- `Extras`

  These were converted using **context-aware TF-IDF**, computed **inside each (Brand, Model, YearBuilt) group**.
  This generated four compact, expressive numeric features:

- **A1**
- **A2**
- **A3**
- **A4**

  These represent the *richness and complexity of vehicle equipment* without messy text fields — ideal for ML models.

Steps - TO Do: [Neat, Clean, well-documented code]

Loading in DataFrame, Perform “Descriptive Statistics” Max-Operations, Input data analysis based on seaborn, Performing core operation of SK Learn, Perform metric analysis

[10 Marks for exceptional creative deep data analytical skills demonstration. For standard implementation – these marks will NOT be given.]

Need demonstration of all skills, learn in this course. End to end. For standard implementation, no marks , will be given. AI generated code or copy pasted code, will get only zero marks.

Section B: Deep Learning [Marks 30]

Process, 1 out of 2 dataset

Time SERIES – RNN , LSTM or GRU

Steps - TO Do:[Neat, Clean, well-documented code]

Design Model with related layers, Performing core operation of Tensflow-Keras based deep learning, Perform Model metric analysis

[10 Marks for exceptional creative deep data analytical skills demonstration. For standard implementation – these marks will NOT be given.]

Dataset

📓 Detailed DL Time-Series Question — Stock Price Forecasting (US Software)

Context:
You'll forecast near-term stock prices for two US software leaders using historical OHLCV data (daily bars). Use **sequence models** (vanilla RNN, LSTM, GRU) and compare performance across architecture and hyperparameters.

Datasets ():

1. **Microsoft (MSFT)** — full daily history, 1986–2024/25
    - Columns: Date, Open, High, Low, Close, Adj Close, Volume (daily) as described on the dataset page.
2. **Adobe (ADBE)** — full daily history, 1986–2024

Why these two? Both are major US software firms with long trading histories, suitable for **long horizon** sequence modeling and **walk-forward evaluation**.

🎯 Business Objective

Build and compare **RNN/LSTM/GRU** models to **predict next-day closing price** (Close_{t+1}) from a rolling **lookback window** of past N days using **multivariate inputs** (Open, High, Low, Close, Volume, optionally Adj Close). Produce **probabilistic intervals** (optional) and **directional accuracy** (up/down) alongside regression error metrics.

2) Preprocessing & Feature Engineering

- **Sorting & alignment:** Ensure chronological order by Date; remove non-trading days if present. *(Datasets are daily OHLCV by construction.)*
- **Scaling:** Fit StandardScaler/MinMaxScaler on **train only** (no leakage).
- **Feature set:** Use multivariate inputs: [Open, High, Low, Close, Volume] (+ Adj Close optional). Consider engineered features: returns = log(Close_t / Close_{t-1}), High-Low range, rolling MA (5/20), RSI(14).
- **Supervised transformation:** Create **sliding windows** (lookback N days → predict Close_{t+1}); experiment with N ∈ {30, 60, 120}.

3) Train/Validation/Test Strategy

- **Walk-forward split:**
    - Train: oldest → cutoff1
    - Validation: cutoff1 → cutoff2
    - Test: cutoff2 → latest
- **Rolling origin evaluation:** Perform **walk-forward validation** with re-fitting or updating to mimic live forecasting.

- Report performance **per decade** (e.g., 2000s, 2010s, 2020s) to show regime stability.

4) Models to Implement

- **Vanilla RNN** (tanh)
- **LSTM** (1–2 stacked layers)
- **GRU** (1–2 stacked layers)
  All in **TensorFlow/Keras**; maintain consistent input pipelines.

Architectural & Training hints:

- Hidden sizes: {32, 64, 128}; Dropout {0.1, 0.3}; Learning rate {1e-3, 5e-4}
- Optimizer: Adam; Loss: MSE (primary), MAE (monitor)
- **EarlyStopping** on validation loss with patience=10

(MSFT & ADBE pages confirm daily OHLCV availability and long historical spans appropriate for deep sequence models.)

5) Baselines

- **Naïve:** Close_{t+1} = Close_t
- **Classical:** ARIMA (univariate Close), and simple **MLP** on returns to benchmark DL gains.

6) Evaluation Metrics

- **Regression:** RMSE, MAE, MAPE, sMAPE, and **R^2** on **test**.
- **Directional:** Accuracy of sign(ΔClose); precision/recall for "Up" class if you turn direction into classification.
- **Economic:** Cumulative PnL of a frictionless "long if predicted up, flat otherwise" strategy; max drawdown.
- **Statistical significance:** Diebold–Mariano test (optional) vs naïve baseline.

7) Robustness & Generalization

- Train on **MSFT**, test on **ADBE** (transfer) to assess cross-ticker generalization.
- Sensitivity analysis for **window size**, **scaler choice**, **feature set (with/without engineered indicators)**.
- **Stability across regimes** (dot-com crash, GFC, COVID, AI rally) using decade slices.

8) Interpretability (Optional)

- **Saliency/Integrated Gradients** on sequences to see which timesteps/features matter most.
- Partial dependence via **feature perturbations** (e.g., shock Volume).

9) Deliverables

- Reproducible **notebook(s)** for both tickers: data prep, rolling windows, model training (RNN/LSTM/GRU), evaluation, plots.
- **Comparison table** of metrics for all models and baselines on **held-out test** for both MSFT and ADBE.
- Short **technical report**: data description (source, date ranges), methodology, metrics, regime analysis, limitations.

□ Minimal Start-Up Checklist (you'll fill in with code)

1. **Load & inspect** MSFT and ADBE CSVs; confirm columns & date span. *(MSFT: 1986–2024; ADBE: 1986–2024 per pages.)*
2. **Create lookback sequences** for multivariate OHLCV.
3. **Build three models** (RNN/LSTM/GRU) with identical input/output shapes.
4. **Walk-forward validation**; **EarlyStopping**; **learning-rate scheduling** optional.
5. **Compare** test RMSE/MAE and **directional accuracy** across models; **plot predictions vs actuals** for recent years.

==========

About Dataset

📈 Microsoft Stock Price History (1986–2025)

This dataset provides daily historical stock price data for **Microsoft Corporation (MSFT)** from **March 13, 1986 to April 6, 2025**. It includes essential trading information such as open, high, low, close, adjusted close prices, and daily trading volume.
Whether you're a data scientist, financial analyst, or machine learning enthusiast, this dataset is perfect for building models, visualizing trends, or exploring the evolution of one of the world's largest tech companies.

📂 Dataset Overview

Column Name	Description
date	(Trading date)
open	Opening price of the stock
high	Highest price during the day
low	Lowest price during the day
close	Closing price of the stock
adj_close	Adjusted closing price (accounting for splits/dividends)
volume	Number of shares traded on the day

Summary

- **Date Range**: 1986-03-13 to 2025-04-06
- **Total Entries**: 9,843

- **Average Close Price**: ~$64.63
- **Max Price (Close)**: $467.56
- **Max Volume**: Over 1 billion shares
- **Missing Values**: None ✅

🔍 Potential Use Cases

- Time-series forecasting using LSTM, ARIMA, or Prophet
- Backtesting trading strategies
- Analyzing long-term financial trends and volatility
- Visualizing market behavior around major events (e.g., dot-com bubble, COVID-19)
- Comparing real vs adjusted stock prices

💡 Project Ideas

- 📉 Predict next-day prices using deep learning
- 📈 Create interactive visualizations with Plotly
- 🕯️ Train an ML model to detect bullish/bearish patterns
- 📊 Calculate technical indicators like RSI, MACD, Bollinger Bands

---

Adobe Stock Data 1986-2024

Adobe Stock Data from 1986 to 2024

About Dataset

📊 Adobe Stock Dataset (1986-2024)

This dataset contains historical stock data for **Adobe Inc.** (ticker symbol: ADBE) obtained from Yahoo Finance. The dataset spans from **1986 to 2024**, offering a rich insight into Adobe's stock performance over nearly four decades. The data provides essential information about Adobe's market behavior, which can be used for various analyses, such as trend analysis, forecasting, and financial modeling.

🤔💻 How the Data is Made

The data is sourced from Yahoo Finance, where stock prices are recorded for every trading day. It includes key market information like opening, closing, highest, and lowest prices of the stock on any given day, along with the trading volume and adjusted close prices.

- **Opening Price**: The first price of the stock traded during market hours.
- **Closing Price**: The last price at which the stock was traded during market hours.
- **Adjusted Close**: The closing price adjusted for dividends and stock splits to reflect a true value over time.
- **High/Low Prices**: The highest and lowest prices at which the stock traded throughout the day.
- **Volume**: The number of shares traded during the day.
  The data is processed daily, and over the years, it has been aggregated to offer a long-term view of Adobe's stock performance.

📋 Column Descriptions

Column Name	Description
📅 **Date**	The date when the stock data was recorded. Represents each trading day.
📈 **Adj Close**	The adjusted closing price accounting for corporate actions like dividends.
📉 **Close**	The final price at which the stock was traded on that day.

Column Name	Description
High	The highest price that Adobe’s stock reached on a given day.
Low	The lowest price Adobe’s stock reached during a trading day.
Open	The price at which Adobe’s stock opened for trading at the start of the day.
Volume	The total number of shares traded during the day. Indicates market activity.

Time Span of Data

- **Start Date**: 1986 (The year Adobe was first publicly listed on the stock market)
- **End Date**: 2024 (Latest available data)

Key Insights from the Dataset

- **Market Trends**: Track the upward and downward trends in Adobe's stock value, identifying key periods of growth or decline.
- **Volatility**: Analyze the fluctuations in stock prices using the high and low values to understand Adobe’s stock market volatility.
- **Volume Activity**: Understand market sentiment and investor interest by examining trading volumes.

- **Stock Performance**: Assess Adobe’s performance over time using adjusted closing prices, which account for stock splits and dividends.
  This dataset offers a detailed, long-term view of Adobe's stock and is a valuable resource for anyone interested in financial analysis, stock price prediction, or market behavior study.

Need demonstration of all skills, learn in this course. End to end. For standard implementation, no marks , will be given. AI generated code or copy pasted code, will get only zero marks.