## **Smart Bricks Technical Assessment Task - Case Study**

Automated valuation models (AVMs) are sophisticated AI models used to estimate the market value of real estate properties. These models leverage advanced AI algorithms combined with comprehensive property databases to generate automated property valuations. The core purpose of an AVM is to provide a fast, objective, and cost-effective way to assess a property's worth at a given point in time (current or future).

The key functionality of an AVM is to analyse recent sales data for comparable properties in the local market (you may augment any additional contexts: like lat-long, marco-economic, world-events etc.). By examining factors like location, size, condition, and other relevant attributes, the AVM can identify similar properties and use their sale prices to estimate the probable market value of the target property. AVMs may also incorporate additional data sources, such as historical valuation records from professional appraisers, to enhance the accuracy of their estimates.

The AI based AVMs offer several advantages over traditional manual appraisal methods. AVMs can generate property valuations much more quickly and at a lower cost. They also aim to provide more objective and consistent estimates by minimising human bias. However, AVMs can have limitations and may not be appropriate for all real estate transactions, particularly those involving unique or complex properties.

Overall, AI based AVM models have become an important tool in the real estate industry, enabling faster, cheaper, and more data-driven property assessments in many contexts. Your goal is to build the AVM model based on the following requirements. You are provided with a sample dataset, `requirements.txt` file. People follow the provided folder structures– if you make changes please document the reason for changes to the structure in the Readme.md file.

**This exercise should be limited to the Dubai market.**

*Please note in the attached folder, there are two CSV files, one for rentals and the other for sales transactions.*

### Requirements

This assignment requires you to have the following installed on your local machine or implicit knowledge of:

0. A machine capable of running Linux-based operating systems OR Windows Subsystem for Linux (WSL).

1. Python 3.10 or higher.

2. Docker.

3. Basic knowledge of git.

4. Basic knowledge of machine learning libraries (scikit-learn, XGBoost, TensorFlow/Keras).

5. Basic knowledge of data manipulation libraries (pandas, numpy).

6. Basic knowledge of bash.

7. At least ~6GB of free space on your local machine for the virtual environment and data. You may use a smaller virtual environment if you do not have enough space by adjusting the `requirements.txt` file.

### Step-by-step

1. Clone this repository/folder to your local machine and set it up as a new project in your favourite IDE.

2. Set up a new virtual environment and install the required packages (via `requirements.txt`, you may use any python package manager of your choice).

3. Implement the data preprocessing pipeline in `preprocess.py`:

- Handle missing data

- Perform feature engineering

- Encode categorical variables

- Scale numerical features

4. Implement feature selection in `feature\_selection.py`:

- Correlation analysis

- Feature importance using tree-based models

- Recursive Feature Elimination (RFE)

- Univariate feature selection

- Combine selected features

5. Implement base models and hyperparameter optimization in `base\_models.py`:

- XGBoost

- Random Forest

- Support Vector Regression

- Use Bayesian Optimization for hyperparameter tuning

6. Implement the meta-learner in `meta\_learner.py`:

- Create a neural network that takes predictions from base models as input

- Implement hyperparameter optimization for the meta-learner

7. Create `main.py` that orchestrates the entire pipeline:

- Data loading and preprocessing

- Feature selection

- Base model training and prediction

- Meta-learner training and final prediction

8. Implement model evaluation metrics in `evaluation.py`:

- RMSE (Root Mean Squared Error)

- R2 Score

- MAE (Mean Absolute Error)

9. Create a `config.py` file to store all configurable parameters (e.g., file paths, model parameters, etc.).

10. Provide some example code in your `main.py` to save intermediate results during the model training process.

11. Set up a `test.py` file and provide unit tests for key components of your pipeline.

12. Set up the `Dockerfile` for containerizing your application.

13. Create a `README.md` file with clear instructions on how to run your code, including:

- Environment setup

- Data requirements

- How to train the model

- How to make predictions using the trained model

14. Please use git and regularly (may be one feature at a time) commit code to your local repository. Use `git bundle` to create a bundle of your repository or zip it to share it with us.

### Optional

These are optional steps that you may complete if you have time:

1. You can augment the dataset provided with additional features not available in the provided data set to improve the performance on the model.

2. Implement a simple API using Flask or FastAPI to serve predictions.

3. Create a basic web interface for inputting property details and displaying predictions.

4. Implement cross-validation in your model training process.

5. Add data visualisation for feature importance and model performance.

6. Implement an ensemble method that combines multiple meta-learners.

7. Optimise your code for performance (e.g., using parallel processing where applicable).

8. Implement a feature to explain individual predictions (e.g., using SHAP values).

9. Create a Jupyter notebook demonstrating the use of your model with example data.

10. Provide an architecture diagram of your AVM system, showing how different components interact.

### Submission

Please provide any additional notes, explanations, or comments pertaining to your implementation here. Include any challenges you faced, assumptions you made, or ideas for future improvements.

This task is designed to evaluate the candidate's ability to implement a complex machine learning pipeline for real estate valuation. It covers various aspects of the data science workflow, from data preprocessing to model deployment, and allows candidates to showcase their skills in machine learning, software engineering, and problem-solving.