**Mahmoud Ayman Kharoof Smart Bricks Automated Valuation Model (AVM) - Project Documentation**

**Project Overview**

The Automated Valuation Model (AVM) leverages AI and machine learning to estimate real estate property values in the Dubai market. This project involves data preprocessing, feature selection, base model implementation, meta-learner creation, and containerization with Docker. The workflow is designed for efficient, data-driven property valuation using advanced AI algorithms.

**Table of Contents**

1. Environment Setup
2. Folder Structure
3. Data Preprocessing
4. Feature Selection
5. Base Model Implementation
6. Meta-Learner Implementation
7. Evaluation Metrics
8. GitHub Desktop for Version Control
9. Docker Setup
10. Future Improvements

**Environment Setup**

**1. Install WSL**

* Ensure you have WSL2 with Ubuntu installed on your system.
* Update your WSL environment:

sudo apt update && sudo apt upgrade -y

**2. Install Python**

* Install Python 3.10 or higher:

sudo apt install python3 python3-venv python3-pip -y

**3. Install Docker**

* Install Docker CE in WSL:

sudo apt install docker-ce docker-ce-cli containerd.io -y

* Allow Docker to run without sudo:

sudo usermod -aG docker $USER

newgrp docker

**Folder Structure**

The project adheres to the following structure:

├── preprocess.py # Data preprocessing pipeline

├── feature\_selection.py # Feature selection logic

├── base\_models.py # Base model implementation

├── meta\_learner.py # Meta-learner implementation

├── main.py # Orchestrates the entire pipeline

├── evaluation.py # Model evaluation metrics

├── config.py # Configuration parameters

├── Dockerfile # Docker container setup

├── requirements.txt # Python dependencies

├── test.py # Unit tests for key components

├── data/

│ ├── sales.csv # Sales data

│ ├── rentals.csv # Rentals data

└── README.md # Documentation

**Data Preprocessing**

The preprocess.py script handles:

1. **Missing Data Handling**:
   * Fill missing values with statistical methods (mean/median for numerical, mode for categorical data).
2. **Feature Engineering**:
   * Create interaction terms, binning, and derived features.
3. **Encoding Categorical Variables**:
   * Use LabelEncoder or OneHotEncoder for categorical features.
4. **Scaling Numerical Features**:
   * Apply StandardScaler or MinMaxScaler for uniform scaling.

**Feature Selection**

The feature\_selection.py script implements:

1. **Correlation Analysis**:
   * Drop features with high correlation to reduce redundancy.
2. **Tree-Based Feature Importance**:
   * Use XGBoost to rank features by importance.
3. **Recursive Feature Elimination (RFE)**:
   * Use RFE for stepwise feature elimination.
4. **Univariate Feature Selection**:
   * Select features based on statistical scores.

**Base Model Implementation**

The base\_models.py script includes:

1. **Model Selection**:
   * Implement XGBoost, Random Forest, and Support Vector Regression (SVR).
2. **Hyperparameter Tuning**:
   * Use Bayesian Optimization for tuning model parameters.

**Meta-Learner Implementation**

The meta\_learner.py script contains:

1. **Neural Network Meta-Learner**:
   * Combines predictions from base models as input.
2. **Hyperparameter Optimization**:
   * Tune meta-learner parameters using grid search or random search.

**Evaluation Metrics**

The evaluation.py script evaluates models using:

1. **Root Mean Squared Error (RMSE)**:
   * Measure of prediction accuracy.
2. **R-Squared (R²)**:
   * Proportion of variance explained by the model.
3. **Mean Absolute Error (MAE)**:
   * Average of absolute differences between predicted and actual values.

**GitHub Desktop for Version Control**

**1. Install GitHub Desktop**

* Download GitHub Desktop from [here](https://desktop.github.com/).

**2. Clone Repository**

* Open GitHub Desktop and clone the repository.

**3. Regular Commits**

* Make regular commits after completing each task:
  + Example: After implementing data preprocessing, commit with a descriptive message like:

“Added data preprocessing pipeline in preprocess.py”

**4. Sync Changes**

* Sync changes with the remote repository to keep backups and enable collaboration.

**Docker Setup**

**1. Create Dockerfile**

Add the following content to the Dockerfile:

FROM python:3.10-slim

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

CMD ["python", "main.py"]

**2. Build Docker Image**

docker build -t avm-app .

**3. Run Docker Container**

docker run -it avm-app

**Future Improvements**

1. **Augment Data**:
   * Include additional features from external sources.
2. **Deploy Model**:
   * Use Flask or FastAPI to create an API for serving predictions.
3. **Cross-Validation**:
   * Add cross-validation to ensure robust model performance.
4. **Data Visualization**:
   * Visualize feature importance and prediction results.
5. **Explainable AI**:
   * Use SHAP or similar tools to explain individual predictions.