Step 12: Designing My Deployment Solution

My production solution will be a webapp which can accept features about a London property and display the predicted price.

Components

Part One: Building the property prediction models

I have developed a python script which can create models capable of predicting property prices.

Input

The input will a csv file, which contains property samples. The data is hosted on GitHub. The data has been collected on a single occasion, and is one month of London property prices. As such, it provides a snapshot in time of property prices.

It would be possible to collect data again at a later date and retrain the model, to produce more up-to-date property price predictions.

Each sample includes a list of features about each property, and the property price.

Parameters

The python script is configurable based on multiple parameters

- Algorithm
  - This determines the model which should be produced. Options for the output model include:
    - Linear Regression
    - KNN
    - XGBoost
    - CatBoost
- Dataset Version Number
  - This determines which version of the dataset should be used.
    - The best performing dataset version is version 11, which contains 80 features
    - However, there have been models which have been well-performing with smaller feature sets. For example, KNN performs well with dataset version 06, which only contains 8 features. (Due to the nature of the KNN algorithm, it fails to finish training in a reasonable time with version 11 and no option for this model has been made available)
- Algorithm Detail

  - this determines how the hyperparameters should be explored for optimisation.
    * RandomizedSearchCV is the most commonly-used option
    * There are also options available for GridSearchCV, or running/re-running a custom hyperparameter space (useful for verifying a hyperparameter set performs consistently well)
- Data Detail
  - Typically only used with CatBoost, this ensures that features are not dummied before processing. CatBoost does not require dummy features, and useful insight can be gained from feature importance if dummying is not carried out.

Processing

The python script will train the requested model using the training csv file, and find an optimal machine learning model for predicting property prices with the available data.

Output

The output will be a machine learning model. I have successfully used this script to create multiple machine learning models capable of predicting London property prices.

Part Two: WebApp

The webapp will be created using Streamlit and hosted using either Heroku or Streamlit Share.

I will upload my best performing prediction models into the webapp, and create a user interface which will allow the user to predict property prices from a series of pre-selected properties.

Input

The inputs will be:

- a sample property, with all the features for that property. A sample property will be picked at random for the user.
- a choice of which well-performing model to use to make property predictions

Processing

The webapp will use the selected model to accept the property features and make a prediction about what the property price should be.

Output

The predicted price will be displayed to the user.

The webapp will also display the known actual price to the user, in order to see how accurately the model predicted the property price.