In this second part, you will work on the house prices data set assembled and published by Dean De Cock. It's a set of 2,930 observations with 82 attributes each. The goal is to use the first 2,430 ones to fit and evaluate different models and use them to make predictions for the last 500 ones. Note that we don't provide the prices for those 500 houses, your task is to estimate them.

Project files

Download the `house-prices.zip` archive from the resource section. You should see the following files

- Detailed data description - `documentation.txt`
- The 2,430 data points with prices - `house-prices.csv`
- The remaining 500 without prices - `house-prices-test.csv`
- Example submission file - `predictions-example.csv`
- Data set source - `source.txt`

Create a `house-prices-solution.ipynb` notebook to write your analysis. The notebook should also run with the `exts-ml` environment.

A quick look at the data

Here are the first five entries from `house-prices.csv`

	Order	PID	MS SubClass	MS Zoning	...	Yr Sold	Sale Type	Sale Condition	SalePrice
0	484	528275070	60	RL	...	2009	WD	Normal	236000
1	2586	535305120	20	RL	...	2006	WD	Normal	155000
2	2289	923228250	160	RM	...	2007	WD	Normal	75000
3	142	535152150	20	RL	...	2010	WD	Normal	165500
4	2042	903475060	190	RM	...	2007	WD	Normal	122000

5 rows × 82 columns

You can find a detailed description of each variable in the `documentation.txt` file, but there are a few things to know.

- The `Order` and `PID` variables are identifiers. They are not useful to predict house prices.
- The variables are not necessarily encoded consistently. For instance, `MS SubClass` (the type of dwelling) and `MS Zoning` (zoning classification) are both categorical variables, but one is encoded with numerical values and the other with short labels.
- The data isn't clean: there are incorrect and missing values, outliers and inconsistencies

You should address all these issues during the **data cleaning** stage.

Data cleaning

Your analysis should include the necessary **data cleaning steps**. For instance

- Find and handle incorrect, missing values
- Correct inconsistencies in the variables
- Handle outliers

You are free to choose your preferred approach to handle each step. For instance, you might want to replace missing values with the average or the most frequent value or create an `missing` category. In any case, **justify your choices**!

Feature encoding

Your analysis should include the necessary **feature encoding steps**. The `documentation.txt` file labels each variable with its type. For categorical ones, it uses the ordinal, nominal and discrete classification.

- Ordinal variables - you can order the categories
- Nominal variables - no possible ordering
- Discrete variables - categories are integer values

The encoding depends on the type of variable and its meaning. For instance, the kitchen quality variable is on a scale from excellent to poor. Hence, it's an ordinal variable, and you can choose to apply one-hot encoding or define a numerical scale ex. excellent corresponds to 5 and poor to 1. In any case, **justify your choices**!

Feature engineering

Your analysis should also include **feature engineering**. Here are a few ideas

- Create indicator variables ex. year of construction is older than some threshold
- Transformations ex. log-transforms, polynomials

Suggestion: write down your feature engineering ideas during the data exploration stage.

> ***Warning***: *Be careful when adding total counts (ex. the total number of rooms, living surface) and other linear combinations of the input features. Those variables don't add "modeling*

> *power" to the model if you keep the original features in the data and can lead to ill-conditioning and numerical issues. On the other hand, if you create such variables and remove the original features, it can be seen as a way to compress the information on fewer dimensions which can be useful for the simple and intermediate models where the number of variables is limited.*

Model fitting

Your analysis should include an appropriate **baseline** and evaluate three different models ranging in complexity

- A **simple model** with two variables (three with the target variable)
- An **intermediate model** (between 10 and 20 variables)
- A **complex model** with all variables

The number of variables is only given as an indication, it's not a strict range. Also, it corresponds to the variables count before one-hot encoding. For the simple and intermediate models, you can choose the variables. You are free to choose your preferred approach for this variable selection step, but you should include a short comment to **explain your choice**.

> ***Example***: *I decide to choose variables* `v1` , `v2` *and* `v3` *for my simple model because I think that they provide a good overview of the house - or - I choose these variables because they are the most correlated with the target - or - I decide to test the SelectKBest object that I found in Scikit-learn to do automatic feature selection.*

Regularization

Your analysis should include **regularization** for at least the complex model, tune its strength with **grid search** and plot the **validation curves**.

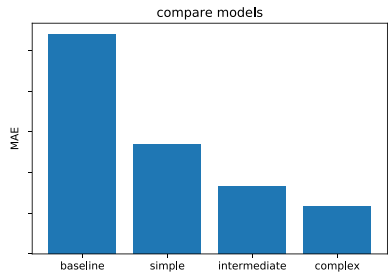
Communicating the results

You are free to use any appropriate cost function to fit your models, with or without prices transformation ex. log-transformed. You can also choose the train/test split ratio ex. 50-50 split. However, **explain your choices** in the notebook.

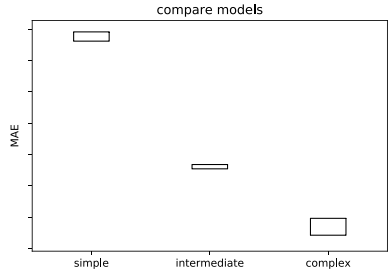
You can also track different metrics to evaluate the performance of your models. However, make sure to print the **mean absolute error** (MAE) score **in dollars** for each one.

> ***Example***: *My simple model has an MAE of 25,123 thousand dollars - or - The mean MAE score of my model is* `..` ± `..` *dollars (std)*

Your analysis should also include a final visualization which summarizes the different models MAE scores. For instance, using a bar chart



or even a box-plot if you decided to evaluate your models on several train/test splits



Compute predictions

Make predictions for the 500 new houses from `house-prices-test.csv` using each model and save them in a `.csv` file.

- Predictions from your simple model - `predictions-simple-model.csv`
- Predictions from your intermediate model - `predictions-intermediate-model.csv`
- Predictions from your complex model - `predictions-complex-model.csv`

Your `.csv` files should contain two columns: the house `PID` and the predicted price. You can find a sample submission file in `predictions-example.csv` .

PID	SalePrice

0	909279080	420124
1	907126050	543610
2	528144030	460166
3	535452060	417207
4	911202100	327230

RESOURCES

house-prices.zip 210 KB