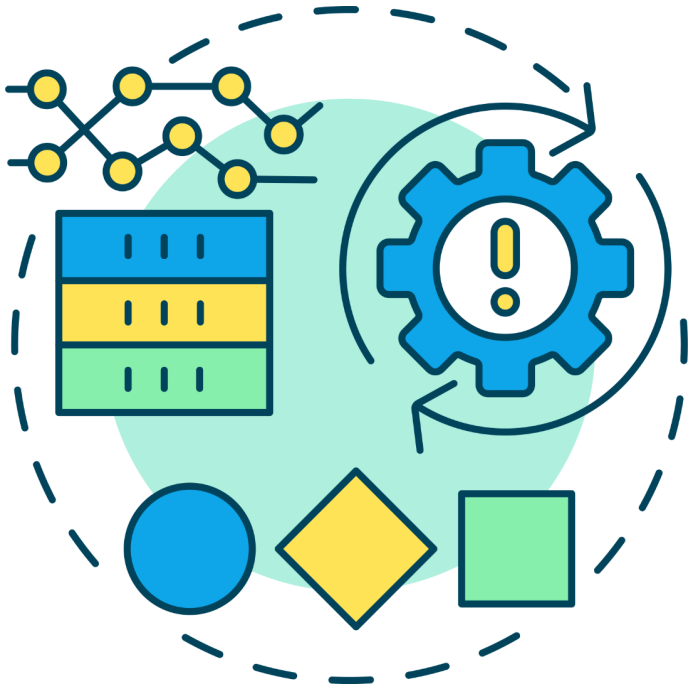
Assignment Lesson 6 - 7

- Regression Algorithms -

Regression, like classification, is another core of Data Science itself–to predict a numerical value based on the information in the data.



In this assignment, you will try to perform some Regression techniques, complete with Validation and Parameter Optimization, and give some reasoning and analysis about your choice of parameter and the obtained result.
Again, you are **free to use any dataset you are interested in** as long as it satisfies the following conditions:

- Contains at least **8 regular attributes** (can be numerical or categorical (before encoding). ID-like data (such as name) or time series data is not counted.)
- Contains at least **500 examples.**
- Contains **a numerical label attribute**

Feel free to priorly adjust the data (discretization or just select a few attributes) so it will fit those requirements. Finding a perfect dataset can be challenging this time. Some cleaning might be needed. You can look at kaggle.com for some datasets. If the dataset you used in the previous assignment still satisfied those conditions, you can use that dataset again. To complete this assignment, feel free to choose any statistical tool you like.

[!] *Three students who submit their assignments by the deadline with the most interesting data and data exploration results will receive an* **Amazon Gift Card worth 3000 yen**. *The result will be announced around 2 weeks after the deadline of the assignment.*

Prior Knowledge

Data Science Objective 4/4

Using the data you have, write down **what kind of prediction** you will do and **why it matters**.

The price of a house depends on different factors (location, renovation, number of rooms, size…). We are actually interested to see which factors impact the most the final price of a house. Can we predict the price of a house given its condition?

With 79 explanatory variables describing different aspect of residential homes in Ames, Iowa, we will try to predict the final price of each home. This would be helpful to seller to estimate the price of their house, but it is also price references to buyers preventing them from being scammed. In the point of view of architects, it is also provides what people prefer (what they want to pay for).

Upload the data to your Google Drive and paste the link here. It can be in a **csv or xls** format (not compressed like zip). Please make sure the sharing setting **allows Tohoku University members to see it.**

Link:
Original data: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data
Preprocessed data: https://drive.google.com/file/d/1xd6PuXoFihZnLiJzG85PAW0KVec_cohh/view?usp=share_link

- We merged the train and test data files from Kaggle (They had the same size) to get a bigger set
- We used one hot encoding for the categorical datas.
- The numerical data were normalized
- The missing values have been replaced by 0

Data Preparation & Exploration

Dataset 1/1

Give a screenshot of the **top 10 - 30 examples** of your dataset.

As the data set contains 79 columns (including categorical columns - which have been one-hot-encoded), not all columns will be displayed.

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	SaleType_Oth
0	0.067320	-0.184443	-0.217841	0.646073	-0.507197	1.046078	0.896679	0.523038	0.580708	-0.293030	...	0
1	-0.873466	0.458096	-0.072032	-0.063174	2.187904	0.154737	-0.395536	-0.569893	1.177709	-0.293030	...	0
2	0.067320	-0.055935	0.137173	0.646073	-0.507197	0.980053	0.848819	0.333448	0.097840	-0.293030	...	0
3	0.302516	-0.398622	-0.078371	0.646073	-0.507197	-1.859033	-0.682695	-0.569893	-0.494771	-0.293030	...	0
4	0.067320	0.629439	0.518814	1.355319	-0.507197	0.947040	0.753100	1.381770	0.468770	-0.293030	...	0
5	-0.167877	0.672275	0.500430	-0.772420	-0.507197	0.715952	0.513801	-0.569893	0.637774	-0.293030	...	0
6	-0.873466	0.243916	-0.010665	1.355319	-0.507197	1.079091	0.992399	0.467277	2.035897	-0.293030	...	0
7	0.067320	0.000000	0.027119	0.646073	0.391170	0.055700	-0.539116	0.768390	0.916521	-0.103911	...	0
8	-0.167877	-0.784145	-0.513264	0.646073	-0.507197	-1.330831	-1.639892	-0.569893	-0.968860	-0.293030	...	0
9	3.124875	-0.826981	-0.348436	-0.772420	0.391170	-1.066730	-1.639892	-0.569893	0.898962	-0.293030	...	0
10	-0.873466	0.029737	0.130834	-0.772420	-0.507197	-0.208401	-0.921995	-0.569893	1.019679	-0.293030	...	0
11	0.067320	0.672275	0.222630	2.064566	-0.507197	1.112104	1.040259	1.024894	1.221606	-0.293030	...	0
12	-0.873466	0.000000	0.355000	-0.772420	0.391170	-0.307439	-1.065574	-0.569893	0.648748	-0.293030	...	0
13	-0.873466	0.929291	0.061352	0.646073	-0.507197	1.145116	1.088119	1.136418	-0.968860	-0.293030	...	0
14	-0.873466	0.000000	0.095332	-0.063174	-0.507197	-0.373465	-1.161294	0.612257	0.639969	-0.293030	...	0
15	-0.285475	-0.784145	-0.513264	0.646073	2.187904	-1.396856	0.800960	-0.569893	-0.968860	-0.293030	...	0
16	-0.873466	0.000000	0.136032	-0.063174	1.289537	-0.043338	-0.682695	0.433819	0.299766	-0.293030	...	0
17	0.772910	0.115409	0.078976	-1.481667	-0.507197	-0.142376	-0.826275	-0.569893	-0.968860	-0.293030	...	0
18	-0.873466	-0.141607	0.447177	-0.772420	-0.507197	1.079091	0.944539	-0.569893	0.449016	-0.293030	...	0
19	-0.873466	0.029737	-0.330685	-0.772420	0.391170	-0.439490	-0.921995	-0.569893	0.137347	-0.293030	...	0

20 rows × 331 columns

Preparing Training and Testing 5/5

Using random sampling, set 10% of the data aside as the hold-out test set. This test set will be used at the end of modeling. Give a screenshot of the **count of both datasets.**

```
Selected Ratio: 10%
Total Training Set:  2627
Total Test Set:  292
```

Summary Statistics 2/2

Give a screenshot of the summary statistics of your data (**mean**, **median, standard deviation, min, max, count**).

Here's a brief version of all the columns that we have:

- **SalePrice** - the property's sale price in dollars. This is the target variable that you're trying to predict.
- **MSSubClass**: The building class
- **MSZoning**: The general zoning classification
- **LotFrontage**: Linear feet of street connected to property
- **LotArea**: Lot size in square feet
- **Street**: Type of road access
- **Alley**: Type of alley access
- **LotShape**: General shape of property
- **LandContour**: Flatness of the property
- **Utilities**: Type of utilities available
- **LotConfig**: Lot configuration
- **LandSlope**: Slope of property
- **Neighborhood**: Physical locations within Ames city limits
- **Condition1**: Proximity to main road or railroad
- **Condition2**: Proximity to main road or railroad (if a second is present)
- **BldgType**: Type of dwelling
- **HouseStyle**: Style of dwelling
- **OverallQual**: Overall material and finish quality
- **OverallCond**: Overall condition rating
- **YearBuilt**: Original construction date
- **YearRemodAdd**: Remodel date
- **RoofStyle**: Type of roof
- **RoofMatl**: Roof material
- **Exterior1st**: Exterior covering on house
- **Exterior2nd**: Exterior covering on house (if more than one material)
- **MasVnrType**: Masonry veneer type
- **MasVnrArea**: Masonry veneer area in square feet
- **ExterQual**: Exterior material quality
- **ExterCond**: Present condition of the material on the exterior
- **Foundation**: Type of foundation
- **BsmtQual**: Height of the basement
- **BsmtCond**: General condition of the basement
- **BsmtExposure**: Walkout or garden level basement walls
- **BsmtFinType1**: Quality of basement finished area
- **BsmtFinSF1**: Type 1 finished square feet
- **BsmtFinType2**: Quality of second finished area (if present)
- **BsmtFinSF2**: Type 2 finished square feet
- **BsmtUnfSF**: Unfinished square feet of basement area

- **TotalBsmtSF**: Total square feet of basement area
- **Heating**: Type of heating
- **HeatingQC**: Heating quality and condition
- **CentralAir**: Central air conditioning
- **Electrical**: Electrical system
- **1stFlrSF**: First Floor square feet
- **2ndFlrSF**: Second floor square feet
- **LowQualFinSF**: Low quality finished square feet (all floors)
- **GrLivArea**: Above grade (ground) living area square feet
- **BsmtFullBath**: Basement full bathrooms
- **BsmtHalfBath**: Basement half bathrooms
- **FullBath**: Full bathrooms above grade
- **HalfBath**: Half baths above grade
- **Bedroom**: Number of bedrooms above basement level
- **Kitchen**: Number of kitchens
- **KitchenQual**: Kitchen quality
- **TotRmsAbvGrd**: Total rooms above grade (does not include bathrooms)
- **Functional**: Home functionality rating
- **Fireplaces**: Number of fireplaces
- **FireplaceQu**: Fireplace quality
- **GarageType**: Garage location
- **GarageYrBlt**: Year garage was built
- **GarageFinish**: Interior finish of the garage
- **GarageCars**: Size of garage in car capacity
- **GarageArea**: Size of garage in square feet
- **GarageQual**: Garage quality
- **GarageCond**: Garage condition
- **PavedDrive**: Paved driveway
- **WoodDeckSF**: Wood deck area in square feet
- **OpenPorchSF**: Open porch area in square feet
- **EnclosedPorch**: Enclosed porch area in square feet
- **3SsnPorch**: Three season porch area in square feet
- **ScreenPorch**: Screen porch area in square feet
- **PoolArea**: Pool area in square feet
- **PoolQC**: Pool quality
- **Fence**: Fence quality
- **MiscFeature**: Miscellaneous feature not covered in other categories
- **MiscVal**: $Value of miscellaneous feature
- **MoSold**: Month Sold
- **YrSold**: Year Sold
- **SaleType**: Type of sale
- **SaleCondition**: Condition of sale

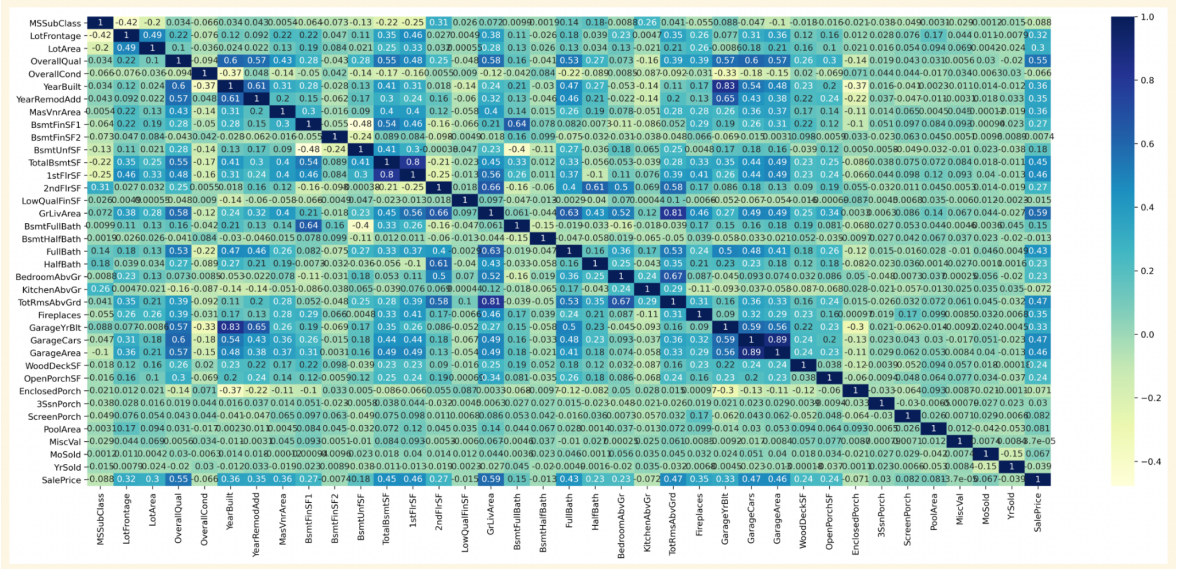
As there are a lot of them, we chose to present here only the numerical columns before preprocessing:

Attributes	Mean	Median	Standard Deviation	Min	Max
MSSubClass	57	50	43	20	1,9E+02
LotFrontage	69	68	23	21	3,1E+02
LotArea	**1E+04**	9,5E+03	7,9E+03	1,3E+03	2,2E+05
OverallQual	6,1	6	1,4	1	10
OverallCond	5,6	5	1,1	1	9
YearBuilt	2E+03	2E+03	30	1,9E+03	2E+03
YearRemodAdd	2E+03	2E+03	21	2E+03	2E+03
MasVnrArea	**1E+02**	0	1,8E+02	0	1,6E+03
BsmtFinSF1	**4,4E+02**	3,7E+02	4,6E+02	0	5,6E+03
BsmtFinSF2	**50**	0	1,7E+02	0	1,5E+03
BsmtUnfSF	**5,6E+02**	4,7E+02	4,4E+02	0	2,3E+03
TotalBsmtSF	1,1E+03	9,9E+02	4,4E+02	0	6,1E+03
1stFlrSF	1,2E+03	1,1E+03	3,9E+02	3,3E+02	5,1E+03
2ndFlrSF	**3,4E+02**	0	4,3E+02	0	2,1E+03
LowQualFinSF	**4,7**	0	46	0	1,1E+03
GrLivArea	1,5E+03	1,4E+03	5,1E+02	3,3E+02	5,6E+03
BsmtFullBath	0,43	0	0,52	0	3
BsmtHalfBath	0,061	0	0,25	0	2
FullBath	1,6	2	0,55	0	4
HalfBath	0,38	0	0,5	0	2
BedroomAbvGr	2,9	3	0,82	0	8
KitchenAbvGr	1	1	0,21	0	3
TotRmsAbvGrd	6,5	6	1,6	2	15
Fireplaces	0,6	1	0,65	0	4
GarageYrBlt	2E+03	2E+03	26	1,9E+03	2,2E+03
GarageCars	1,8	2	0,76	0	5
GarageArea	4,7E+02	4,8E+02	2,2E+02	0	1,5E+03
WoodDeckSF	**94**	0	1,3E+02	0	1,4E+03
OpenPorchSF	**47**	26	68	0	7,4E+02
EnclosedPorch	**23**	0	64	0	1E+03
3SsnPorch	**2,6**	0	25	0	5,1E+02
ScreenPorch	**16**	0	56	0	5,8E+02
PoolArea	**2,3**	0	36	0	8E+02
MiscVal	**51**	0	5,7E+02	0	1,7E+04
MoSold	6,2	6	2,7	1	12
YrSold	2E+03	2E+03	1,3	2E+03	2E+03
SalePrice	1,8E+05	1,8E+05	5,7E+04	3,5E+04	7,6E+05

We compared the mean and the median. When there is a significant difference between those two data, we marked the bigger value in red and in bolt.
We can see that in a lot of situations there are a lot of small numbers: for example there are a lot of houses with no open porch and some of them with a big porches. There night be some outliers.

Correlation Table 3/3

Give a screenshot of a **correlation matrix** of the numerical attributes.



Here are the attributes that have a correlation level superior as 0.70:

- "**GarageCars**" (Size of garage in car capacity) and "**GarageArea**" (Size of garage in square feet) are highly correlated (**0.89**)
  - = Indeed, the bigger a garage is, the more cars can fit in it.
- "**GarageYrBlt**"(Year the garage was built) and "**YearBuilt**" (Year the house was built) are highly correlated (**0.83**)
  - = Most people build their garage at the same time as their house (and not afterwords).
- "**TotRmsAbvGrd**" (Total rooms above grade (does not include bathrooms)) and "**GrLivArea**" (Above grade (ground) living area square feet) are also correlated (**0.81**).
  - = Hypothesis: The bigger the living area is, the more rooms there are in a house
- "**1stFlrSF**" (First Floor square feet) and "**TotalBsmtSF**" (Total square feet of basement area) are highly correlated (**0.8**)
  - = The basement and the first floor are generally proportionnal.

Regression Modeling

[!] In this assignment, the effort of getting high accuracy and the analysis is more important than the value of the accuracy itself. Getting high performance does not mean that you will get the full grade.

Model Preparation

Model Validation 3/3

State one model validation that you will use for all of the models. The validation method of your choice **should be applied to all the models you will make** below. (ex: k-fold validation with k = 5, etc.)

Validation Method: **K-fold validation with K= 5**

Parameter Optimization 2/2

State one parameter optimization that you will use for all of the models. The method of your choice **should be applied to all the models you will make** below. (ex: Random Grid)

Parameter Optimization Method: **Grid Search**

Regression Tree 15/15

Perform regression on your model using the regression tree method. Adjust the pruning parameter range to obtain the best performance. Give screenshot(s) of the code (for Python) or operators architecture & the parameters of the model (for Rapidminer).

```
regressor = sklearn.tree.DecisionTreeRegressor(
    max_depth=7,
    min_samples_split = 27,
    criterion = "friedman_mse"
)
```

We first try the best value for max_depth from 1 to 50 with a step = 1

Then, we did the same for min_samples_split from 1 to 200 with a step = 1

Finally we try the different criterion (but this value did not impact the final result)

Performance Evaluation from Validation: (ex: RMSE = ???, R2Score = ???)

```
R2 Score = 0.40
Root Mean Square Log Error = 0.053

The RMSLE is the main evaluation method as it penalizes the under estimate more
than over estimate. It is useful in prediction of sales.
```

Analysis（30 - 100 words）

(Hint: As the first model you tried, what do you think the performance evaluation results mean? Is it good enough?)

The obtained result is quite bad. Only 40% of the variability observed in the target variable is explained by this regression model. However, the RMSLE seems to be small. It seems that a regression tree is a too simple model for our dataset. Indeed, after preprocessing the data, we had more than 300 columns, which might be too many attributes for a decision tree model. To verify this hypothesis, we could try to select fewer attributes, the ones that we estimate the most relevant.

Linear Model 8/10 5/5

Try out multiple linear models from the Generalized Linear model. Just show 1 model you think is the best for your data. The regularization method needs to be applied as well to your model (any regularization method can be used.) Give screenshot(s) of the code (for Python) or operators architecture & the parameters of the model (for Rapidminer).

```
regressor = sklearn.linear_model.Ridge(
    alpha=481
)
```

Performance Evaluation from Validation:

```
R2 Score = 0.46
Root Mean Square Log Error = 0.044
We tried every alpha from 1 to 1000 with a step = 1
```

Analysis (30 - 100 words)

(Hint: Compared to other linear models, why the one you choose works well for your data?))

This time, the model performed better. We chose the Ridge model because it improves efficiency, even though the model is less interpretable due to the high number of features (compared to Lasso). Many attributes in our dataset are a little bit correlated (the coefficient is between 0.5 and 0.7), so that is also maybe why the Ridge model performs better than Lasso, ElasticNet (which performs well to highly correlated values), and, of course, also a Linear model without regularization.

Ensemble Modeling

Random Forest 15/15 5/5

Perform regression on your model using the random forest method. Give screenshot(s) of the code (for Python) or operators architecture and the parameters of the model (for Rapidminer).

```
regressor = RandomForestRegressor(
    n_estimators=90,
)
```

Performance Evaluation from Validation:

```
R2 Score = 0.48
Root Mean Square Log Error = 0.045
We tried every n_estimators from 1 to 1000 with a step = 10
```

Analysis (30 - 100 words)

(Hint: How much better is random forest compared to singular regression tree?)

Random forest gives a higher R2 Score and a way small root mean square log error than the singular regression tree. The random forest is indeed an aggregation of multiple decision trees. However, it considers only a random subset of the training set attributes when splitting each node in a decision tree. So even if each sub-tree doesn't use all the attributes' information, the whole model does. The output is more stable. So far, the random forest gives the best performances.

Stacking 10/10 10/10

Perform regression on your model using the stacking method. Choose any combination of models you like as your best model, whatever it takes to get the best performance. The choice of the final estimator/meta-learner is up to you. Give screenshot(s) of the code (for Python) or operators architecture and the parameters of the model (for Rapidminer).

```
regressor = StackingRegressor(
    estimators=[
    (
        'Ridge',
        sklearn.linear_model.Ridge(
            alpha=481
        )
    ),
    (
        'RandomForest',
        sklearn.ensemble.RandomForestRegressor(
            n_estimators=90,
        )
    ),
        (
        'SGD',
        sklearn.linear_model.SGDRegressor(
            learning_rate="constant"
        )
    )
    ],
    final_estimator=sklearn.tree.DecisionTreeRegressor(max_depth=3)
    )
```

Performance Evaluation from Validation:

```
R2 Score = 0.45
Root Mean Square Log Error = 0.046

We chose our 2 best algorithms
We added a SGD model and tried different learning rates (["constant", "optimal", "invscaling", "adaptive"])
We tried different max_depth for our final estimator (2 to 5)
We added an ElasticNet Model with different alphas.
```

Analysis（30 - 100 words）

(Hint: Does combining some different algorithms really gives better performance?)

The result we obtained is not that much different from the previous ones. The Random Forest remains our best model, even though the stacking one is made of it. We also realised that adding an ElasticNet model to the the estimators did not impact the final performances. We also realized that using a SGD model gives better result than an Elastic Net model. It might be because an Elastic Net is fundamentally close to a Ridge model, so we guess that the two models give the same information.

Final Testing 5/5

Test all the models you've made with the 10% hold-out test set you set aside in the beginning. Write in their performance

Regression Tree:

R2 Score = 0.30

Root Mean Square Log Error = 0.463

Linear Model:

R2 Score = 0.45

Root Mean Square Log Error = 0.045

Random Forest:

R2 Score = 0.45

Root Mean Square Log Error = 0.046

Stacking:

R2 Score = 0.30

Root Mean Square Log Error = 0.055

Analysis（30 - 100 words）

(Hint: How do all the models performed on the test dataset compare to the one you obtained from validation?)

We can see that the regression tree performs once again poorly. Indeed, the Root Mean Square Log Error is ten times higher than with the train data set. The stacking model also performs a bit poorly compared to the previous results. However, the random forest and the linear model give quite similar outputs, which shows that these two models are stable. Both don't make any reduction of attributes. Overall, we would say that the Random Forest gives the best efficiency. However, the ridge model is probably the most stable. It might be because of the correlation among the data and that the sale prices might be a linear problem (Indeed, the more we have rooms, or a bigger apartment, the higher the price would be).
In this study, we did not pay attention to which attribute was the most important in deciding the prices, but if we took a look at the coefficients of the different models, we could get an answer.