- Welcome
- Introduction: Machine Learning concepts
- Module 1. The Predictive Modeling Pipeline
- Module 2. Selecting the best model
- Module 3. Hyperparameter tuning
- **Module 4. Linear Models**

Module overview

Intuitions on linear models
Quiz M4

Non-linear feature engineering for linear models
Quiz M4

Regularization in linear model
Quiz M4

Wrap-up quiz

Wrap-up quiz 4

In this wrap-up quiz you will need to write some code in order to answer quiz questions:

- an empty notebook is available just below to write your code
- quiz questions are located after the notebook here
- the button Open Notebook at the bottom right of the screen allows you to open the notebook in full page at any time

+ Click here to see a demo video of the notebook user interface

Open Notebook

- Module 5. Decision tree models
- Module 6. Ensemble of models
- Module 7. Evaluating model performance
- Conclusion
- Appendix

Module 4 - Wrap-Up Quiz

Importing Data

In [1]:
```
import pandas as pd

ames_housing = pd.read_csv("../datasets/ames_housi
target_name = "SalePrice"
data = ames_housing.drop(columns=target_name)
target = ames_housing[target_name]
```

Selecting Only Numerical Data

In [2]:
```
numerical_features = [
    "LotFrontage", "LotArea", "MasVnrArea", "BsmtF
    "BsmtUnfSF", "TotalBsmtSF", "1stFlrSF", "2ndFl
    "GrLivArea", "BedroomAbvGr", "KitchenAbvGr", "
    "GarageCars", "GarageArea", "WoodDeckSF", "Ope
    "3SsnPorch", "ScreenPorch", "PoolArea", "MiscV
]

data_numerical = data[numerical_features]
```

Building Model

Ridge with $\alpha = 0$

In [3]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_validate

baseline_model = make_pipeline(StandardScaler(), R

cv_results = cross_validate(baseline_model, data_n
```


Powered by OVHcloud

Open the dataset `ames_housing_no_missing.csv` with the following command:

```
ames_housing =
pd.read_csv("../datasets/ames_housing_no_missing.csv")
target_name = "SalePrice"
data = ames_housing.drop(columns=target_name)
target = ames_housing[target_name]
```

`ames_housing` is a pandas dataframe. The column "SalePrice" contains the target variable.

To simplify this exercise, we will only used the numerical features defined below:

```
numerical_features = [
    "LotFrontage", "LotArea", "MasVnrArea", "BsmtFinSF1",
"BsmtFinSF2",
    "BsmtUnfSF", "TotalBsmtSF", "1stFlrSF", "2ndFlrSF",
"LowQualFinSF",
    "GrLivArea", "BedroomAbvGr", "KitchenAbvGr",
"TotRmsAbvGrd", "Fireplaces",
    "GarageCars", "GarageArea", "WoodDeckSF",
"OpenPorchSF", "EnclosedPorch",
    "3SsnPorch", "ScreenPorch", "PoolArea", "MiscVal",
]

data_numerical = data[numerical_features]
```

Start by fitting a ridge regressor (`sklearn.linear_model.Ridge`) fixing the penalty `alpha` to 0 to not regularize the model. Use a 10-fold cross-validation and pass the argument `return_estimator=True` in `sklearn.model_selection.cross_validate` to access all fitted estimators fitted on each fold. As discussed in the previous notebooks, use an instance of `sklearn.preprocessing.StandardScaler` to scale the data before passing it to the regressor.

Question 1 (1/1 point)

How large is the largest absolute value of the weight (coefficient) in this trained model?

b) Between 1.0 (1e0) and 100,000.0 (1e5)

c) Larger than 100,000.0 (1e5)

Hint: Note that the estimator fitted in each fold of the cross-validation procedure is a pipeline object. To access the coefficients of the `Ridge` model at the last position in a pipeline object, you can use the expression `pipeline[-1].coef_` for each pipeline object fitted in the cross-validation procedure. The `-1` notation is a negative index meaning "last position".

You have used 1 of 1 submissions

Question 2 (1/1 point)

Repeat the same experiment by fitting a ridge regressor (`sklearn.linear_model.Ridge`) with the default parameter (i.e. `alpha=1.0`).

How large is the largest absolute value of the weight (coefficient) in this trained model?

a) Lower than 1.0

b) Between 1.0 and 100,000.0

c) Larger than 100,000.0

You have used 1 of 1 submissions

Question 3 (1/1 point)

What are the two most important features used by the ridge regressor? You can make a box-plot of the coefficients across all folds to get a good insight.

- [x] b) `"GarageCars"` and `"GrLivArea"` ✓
- [ ] c) `"TotalBsmtSF"` and `"GarageCars"`

You have used 1 of 1 submissions

Question 4 (1/1 point)

Remove the feature `"GarageArea"` from the dataset and repeat the previous experiment.

What is the impact on the weights of removing `"GarageArea"` from the dataset?

- [ ] a) None
- [ ] b) Completely changes the order of the most important features
- [x] c) Decreases the standard deviation (across CV folds) of the `"GarageCars"` coefficient

✓

Select all answers that apply

You have used 1 of 2 submissions

Question 5 (1/1 point)

What is the main reason for observing the previous impact on the most important weight(s)?

b) Removing the "GarageArea" feature reduces the noise in the dataset

c) Just some random effects

EXPLANATION

solution: a)

The number of cars that can fit in the garage is indeed strongly dependent on the area of the garage. This could be checked by computing a correlation coefficient (e.g. the Pearson, Spearman or Kendall correlation coefficients) between the two columns.

Correlated features typically cause unstable estimation of the the matching linear model coefficients, even with some level of regularization. As a result we can expect comparatively larger standard deviations of their coefficients when the two correlated features are included in the linear model.

There is no reason that the measurement of the garage area would be more noisy than most other features.

One way to check the above analysis holds would be to drop the "GarageCars" feature instead of "GarageArea" and check that the coefficient is of "GarageArea" gets to the most important in magnitude along with a small standard deviation.

You have used 1 of 1 submissions

Question 6 (1/1 point)

Now, we will search for the regularization strength that maximizes the generalization performance of our predictive model. Fit a `sklearn.linear_model.RidgeCV` instead of a `Ridge` regressor on the

the regularization strength.

What is the effect of tuning `alpha` on the variability of the weights of the feature `"GarageCars"`? Remember that the variability can be assessed by computing the standard deviation.

- a) The variability does not change after tuning `alpha`
- b) The variability decreased after tuning `alpha` ✔
- c) The variability increased after tuning `alpha`

You have used 1 of 1 submissions

Question 7 (1/1 point)

Check the parameter `alpha_` (the regularization strength) for the different ridge regressors obtained on each fold.

In which range does `alpha_` fall into for most folds?

- a) between 0.1 and 1
- b) between 1 and 10
- c) between 10 and 100
- d) between 100 and 1000 ✔

You have used 1 of 1 submissions

the numerical and categorical columns:

- categorical features can be selected if they have an `object` data type;
- use an `OneHotEncoder` to encode the categorical features;
- numerical features should correspond to the `numerical_features` as defined above. This is a subset of the features that are not an `object` data type;
- use an `StandardScaler` to scale the numerical features.

The last step of the pipeline should be a `RidgeCV` with the same set of `alphas` to evaluate as previously.

Question 8 (1/1 point)

By comparing the cross-validation test scores fold-to-fold for the model with `numerical_features` only and the model with both `numerical_features` and `categorical_features`, count the number of times the simple model has a better test score than the model with all features. Select the range which this number belongs to:

- (x) a) [0, 3]: the simple model is consistently worse than the model with all features
- ( ) b) [4, 6]: both models are almost equivalent
- ( ) c) [7, 10]: the simple model is consistently better than the model with all features

You have used 1 of 1 submissions

In this Module we saw that non-linear feature engineering may yield a more predictive pipeline, as long as we take care of adjusting the regularization to avoid overfitting.

hyperparameter values) to better model the non-linear influence of the numerical features.

Furthermore, let the new pipeline model feature interactions by adding a new `Nystroem` step between the preprocessor and the `RidgeCV` estimator. Set `kernel="poly"`, `degree=2` and `n_components=300` for this new feature engineering step.

Question 9 (1/1 point)

By comparing the cross-validation test scores fold-to-fold for the model with both `numerical_features` and `categorical_features`, and the model that performs non-linear feature engineering; count the number of times the non-linear pipeline has a better test score than the model with simpler preprocessing. Select the range which this number belongs to:

- ○ a) [0, 3]: the new non-linear pipeline is consistently worse than the previous pipeline
- ○ b) [4, 6]: both models are almost equivalent
- ◉ c) [7, 10]: the new non-linear pipeline is consistently better than the previous pipeline

You have used 1 of 1 submissions

YOUR EXPERIENCE

According to you, the 'Wrap-up Quiz' of this module was

- ○ **Too easy, I got bored**
- ○ **Adapted to my skills**
- ○ **Difficult but I was able to follow**

Submit

To follow this lesson, I spent:

- less than 30 minutes
- 30 min to 1 hour
- 1 to 2 hours
- 2 to 4 hours
- more than 4 hours
- I don't know

Submit

FORUM (EXTERNAL RESOURCE)

There are no more M4. Wrap-up quiz 4 topics. Ready to start a new conversation?

Terms and conditions