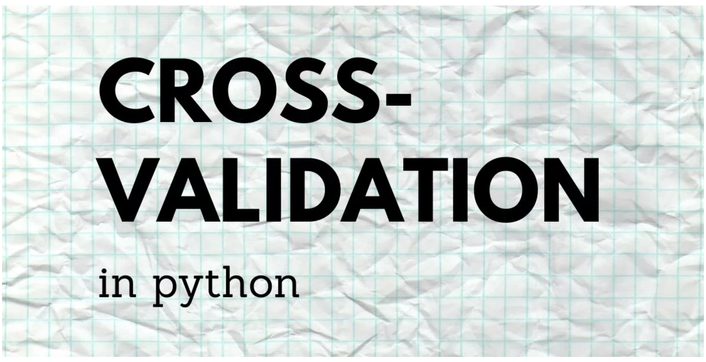
Linear Regression with Cross Validation

K-Fold Cross-Validation in Python Using SKLearn



Splitting a dataset into training and testing set is an essential and basic task when comes to getting a machine learning model ready for training. To determine if our model is overfitting or not we need to test it on unseen data (Validation set).

If a given model does not perform well on the validation set then it's gonna perform worse when dealing with real live data. This notion makes Cross-Validation probably one of the most important concepts of machine learning which ensures the stability of our model.

Cross-Validation is just a method that simply reserves a part of data from the dataset and uses it for testing the model(Validation set), and the remaining data other than the reserved one is used to train the model.

Cross-Validation Intuition

Let's first see why we should use cross validation.

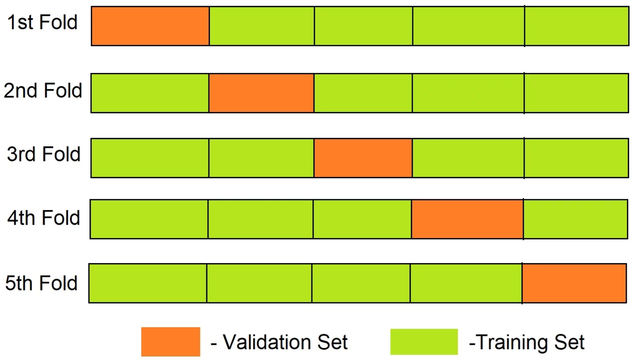
- It helps us with model evaluation finally determining the quality of the model.
- Crucial to determining if the model is generalizing well to data.
- To check if the model is overfitting or underfitting.
- Finally, it lets us choose the model which had the best performance.

There are many types of Cross Validation Techniques:

- Leave one out cross validation
- k-fold cross validation
- Stratified k-fold cross validation
- Time Series cross validation

Implementing the K-Fold Cross-Validation

The dataset is split into 'k' number of subsets, k-1 subsets then are used to train the model and the last subset is kept as a validation set to test the model. Then the score of the model on each fold is averaged to evaluate the performance of the model.



Import Data

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 47
Data columns (total 5 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Petrol_tax                  48 non-null     float64
 1   Average_income              48 non-null     int64
 2   Paved_Highways              48 non-null     int64
 3   Population_Driver_licence(%)  48 non-null     float64
 4   Petrol_Consumption          48 non-null     int64
dtypes: float64(2), int64(3)
memory usage: 2.0 KB
```

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol_Consumption
0	9.0	3571	1976	0.525	541
1	9.0	4092	1250	0.572	524
2	9.0	3865	1586	0.580	561
3	7.5	4870	2351	0.529	414
4	8.0	4399	431	0.544	410

K-fold Cross Validation using scikit learn

```
Number of training indices: 38
Number of validation indices: 10

Number of training indices: 38
Number of validation indices: 10

Number of training indices: 38
Number of validation indices: 10

Number of training indices: 39
Number of validation indices: 9

Number of training indices: 39
Number of validation indices: 9
```

Automating K-fold Cross Validation with cross_val_score() Function

```
<class 'numpy.ndarray'>
array([-0.22573437,  0.62535306, -0.08289785, -0.16796305,  0.30273973])

The average R^2 is: 0.0903
```

Scoring The Cross Validation Results with Different Metrics

```
[51.00294324 57.57120066 51.95759179 53.68447862 80.36175829]

58.91559451839531

[ 4879.40339844  4390.71601106  3639.35979059  4332.2200856
 12820.82586046]

[ 69.85272649  66.26247815  60.3271066   65.81960259 113.229085 75]

75.09819991833851
```

Logistic Regression

```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error', 'fractal dimension error',
       'worst radius', 'worst texture', 'worst perimeter', 'worst area',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry', 'worst fractal dimension',
       'target'],
      dtype='object')

accuracy of each fold: 0.91228

accuracy of each fold: 0.94737

accuracy of each fold: 0.97368

accuracy of each fold: 0.97368

accuracy of each fold: 0.95575

Avg accuracy : 0.9525539512498806
```