Data Imputation

Learn about data imputation and the various methods to accomplish it.

Chapter Goals:

- Learn different methods for imputing data

A. Data imputation methods

In real life, we often have to deal with data that contains missing values. Sometimes, if the dataset is missing too many values, we just don't use it. However, if only a few of the values are missing, we can perform data imputation to substitute the missing data with some other value(s).

There are many different methods for data imputation. In scikit-learn, the `SimpleImputer` transformer performs four different data imputation methods.

The four methods are:

- Using the mean value
- Using the median value
- Using the most frequent value
- Filling in missing values with a constant

The code below shows how to perform data imputation using mean values from each column.

```
# predefined data
print('{}\n'.format(repr(data)))

from sklearn.impute import SimpleImputer
imp_mean = SimpleImputer()
transformed = imp_mean.fit_transform(data)
print('{}\n'.format(repr(transformed)))
```

In NumPy arrays, missing data is represented by the `np.nan` value. In the above example, we replaced each missing value with the mean of the values in its column.

The default imputation method for `SimpleImputer` is using the column means. By using the `strategy` keyword argument when initializing a `SimpleImputer` object, we can specify a different imputation method.

The code below demonstrates various initialization strategies for `SimpleImputer`.

```
# predefined data
print('{}\n'.format(repr(data)))

from sklearn.impute import SimpleImputer
imp_median = SimpleImputer(strategy='median')
transformed = imp_median.fit_transform(data)
print('{}\n'.format(repr(transformed)))

imp_frequent = SimpleImputer(strategy='most_frequent')
transformed = imp_frequent.fit_transform(data)
print('{}\n'.format(repr(transformed)))
```

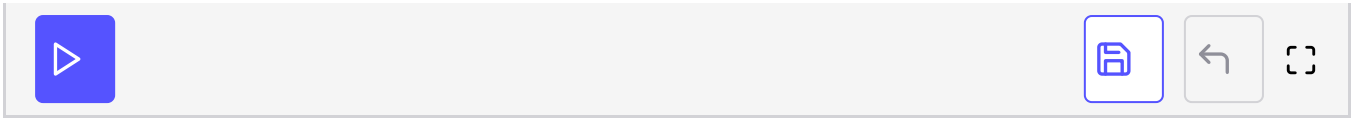
The `'median'` strategy fills in missing data with the median from each column, while the `'most_frequent'` strategy uses the value that appears the most for each column.

The final imputation method that `SimpleImputer` provides is to fill in missing values with a specified constant. This can be useful if there is already a suitable substitute for missing data (e.g. 0 or -1).

The code below demonstrates how to fill in missing data with a specific constant. The `fill_value` keyword argument is used when initializing the `SimpleImputer` object, to specify the constant.

```
# predefined data
print('{}\n'.format(repr(data)))

from sklearn.impute import SimpleImputer
imp_constant = SimpleImputer(strategy='constant',
                             fill_value=-1)
transformed = imp_constant.fit_transform(data)
print('{}\n'.format(repr(transformed)))
```



B. Other imputation methods

The `SimpleImputer` object only implements the four imputation methods shown in section **A**. However, data imputation is not limited to those four methods.

There are also more advanced imputation methods such as k-Nearest Neighbors (filling in missing values based on similarity scores from the kNN algorithm) and MICE (applying multiple chained imputations, assuming the missing values are randomly distributed across observations).

In most industry cases these advanced methods are not required, since the data is either perfectly cleaned or the missing values are scarce. Nevertheless, the advanced methods could be useful when dealing with open source datasets, since these tend to be more incomplete.