Missing value imputation

How to deal with missing values?

1. **Deletion**. If the missing valued features are important to your data and there are not many samples with missing values. We can perform a row wise deletion. In other words: delete the data samples with missing values. If there are many data samples missing the same feature, we can delete the feature column.

2. **Imputation**. Replacing a missing value with another value based on a reasonable estimate. The easiest method is replacing missing values with mean or median value for that feature.

    a. Hot-deck imputation: replace missing value from a similar data sample.
    b. Cold-deck imputation: replace missing value from a similar data sample from different dataset.

Imputation could introduce some bias. There are more ways to do imputation on missing values, including using deep learning model to estimate missing value. However, your homework should be good enough if you use the above technique to handle missing values.

Data pre-processing

Split dataset: ***Keep in mind if your testing dataset is same for all the models.***

sklearn provides functions to help you split dataset. You can directly use the function or write your own function to split dataset with Pandas and Numpy. The sklearn data split function document is listed below:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Data pre-processing: the data pre-processing can be done using sklearn functions. Below is documentation of how to use those functions.

https://scikit-learn.org/stable/modules/preprocessing.html

Data loading documentation for Keras is listed below:

https://keras.io/api/data_loading/

Model

Documentation for Keras fully connected layer:

https://keras.io/api/layers/core_layers/dense/

Documentation for sklearn model:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Recommendation

You can organize your code by few classes and functions:

1. dataPreprocess.py: you can use this class to pre-process and visualize your data.
    a. pre_process(): use the function to pre-process data
    b. load_data(): generate dataset. This function may return training, validation, and testing dataloader
2. model.py: you can use this class to define your model architecture.
    a. layers(): customize your layers. You may not going to need it right now
    b. get_model(): build your model architecture
3. training.py: you can use this class to train your model.
    a. training(): get your dataset, load model and start a loop or use function to fit your model.
    b. save_model(): save the model for future use.
4. testing.py: you can use this class to testing your model's performance and make plot
    a. testing(): load testing dataset and evaluate your model's performance
    b. get_plot(): plot your training, validation and testing loss and performance.