Data Completion and Interpolation Tool

Jiawei Wu(jw1308)
Rutgers University
Piscataway, NJ, USA
Email: jiawei.wu1@rutgers.edu

***Abstract*—** I design and implement a tool to perform data completion and interpolation by using machine learning methods. The tool is able to deal with multiple missing features at the same time by assigning task priority. It is also capable of handle both categorical and numerical variables for regression and classification tasks. In this paper I will shows that the tool is not only easy to use under most cases but also have good performance by experimenting it using real-world dataset.

I. Introduction

A. Problem Description

Dataset has been playing an important role in various aspects in computer science and data science area. Especially for machine learning and data mining area, the datasets used in research papers are becoming increasing complex and massive. However, large datasets can unavoidably be dirty and untidy, making it hard to use directly. So it would be very important to have tools that deal with these issues.

In this project, I develop a tool handling the missing value by performing data completion and interpolation. Traditionally, missing value is handled by simply dropping the corresponding row. It is obvious that this can result in not only the loss in potential information, but the size of the dataset as well. To avoid such situation, I design and implement a tool that impute missing value using machine learning algorithms. The tool takes a dataset and predict the value of missing features using information from other non-missing features. The tool is capable of dealing with both categorical and numerical features as well as handling multiply missing features at the same time.

B. Project novelty

While designing the tool, I try to make the tool not only powerful enough to generate accurate result, but also easy to use under most cases. Strength of the model is listed as follows.

- The model is capable of handling both categorical and numerical variables.
- When there are multiple missing features, the model deals with them by generating task priority and using the generated data for new prediction.
- The model support various strategies for user to choose. It can also choose the best strategy by evaluating their performance.
- The framework it highly extendable, which makes it easy to add new predicting strategies at anytime.

II. Model

A. Model Description

Figure 1 shows the workflow of the model. For non-missing features, the model apply normalize or encoding methods so that they can be fed into the training algorithm later. For missing features, the model specify their imputing strategies and assign task priority. After a feature is imputed, it can be then regards as non-missing features and used for predictions for the latter missing features.

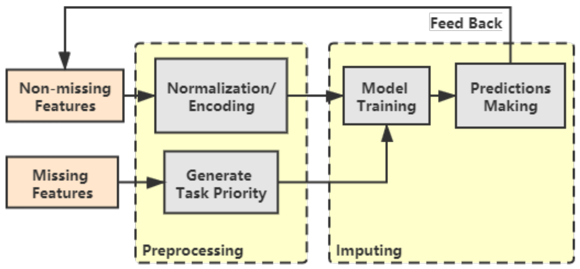

Fig. 1. Workflow of the model

B. Prepossessing

1) Categorizing Features: After taking a dataset, the model will firstly categorize all features into three groups: string value, continuous number and discrete number. This algorithm is designed based on two assumptions:

- Discrete features are only having less or equals to k distinct values, where k is a positive integer set by the user.
- String features can only be considered as Categorical features. Non-categorical string features such as name and address are not supported currently.

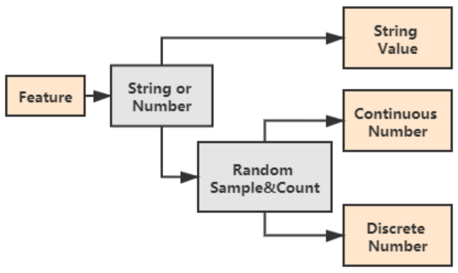

Fig. 2. Data Prepossessing

Figure 2 shows the logic of the algorithm. The algorithm iterate through each column and get the data type of the corresponding feature:

- If values of the feature is not number, mark it as string feature.
- Else, randomly sample N rows from the data and count how many distinct values does the feature have.
- If there are more than k different distinct values, mark it as continuous number.
- Else, mark it as discrete number.

The number of N and k are set as 100 and 25 as default but the user can change their values in a configuration file easily.

2) Handling Non-Missing Features: No-Missing Features are used as training data. The model firstly use algorithm from the previous section to get the category of the Non-Missing Features. Features are separated as continuous features and discrete features and are processed according to their category.

For each continuous feature, in order to balance their initial influence to the model, we apply mean normalization algorithm, which is defined as:

$$x_i' = \frac{x_i - \text{mean}(X)}{\max(X) - \min(X)} \tag{1}$$

For string features, we use one hot encoding algorithm to map string value to an array of integers. The idea behind one hot encoding is that we not only made this feature trainable by machine learning algorithm but guarantee that all embedded features are having the same distance between each other's as well.

As we are assuming that features with string value are discrete features, the embedded array can also be considered as array of discrete values.

3) Handling Missing Features: Missing Features are used as label data of machine learning algorithms. In this step we specify the predicting tasks according to the category of the feature. That is to say,

- For features of discrete values, we need to train a classifier.
- For features of continuous values, we need to train a regressor.

Notice that if there are multiple missing features, we assign classification tasks a higher priority than the regression task basing on two reasons:

- Regression model would be more precise if there are more features, as we are hoping to minimize the mean square error.
- After we finish the classification task of features of string value, we will be having more features as we are encoding this feature to an array for the next step.

C. Imputing

In the previous section,normalization and one hot encoding methods are applied to all features, enabling machine learning algorithms to train models over all features. Meanwhile, tasks are generated according to the category of missing features. In this section, I will briefly talk about the models that we choose for each task.

1) KNN: The basic idea of KNN[1] is that we calculate k nearest neighbors of the data basing on Euclidian distance and let the label of the data be the label with largest value close neighbors. Euclidian distance is defined as:

$$\text{ED}(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2}$$

2) Decision Tree: Decision tree[2] is a model model widely use in classification tasks. The model repeatedly picks features that can partition the data best basing on information gain, information gain ratio or Gini index of the features.

As decision tree only support discrete features when doing partition, extra steps are needed for continuous features:

- Acquire the interval of the value by getting the maximal and minimal value of the feature.
- Partition the interval into m intervals of same size.
- Data in the interval $[i, i+1]$ converted to value i.

By using this strategy, continuous values can be converted to discrete values, making it trainable by the decision tree classifier.

3) Random Forest: Random Forest[3] is based on decision tree model. In our model, Random Forest is built using following strategy:

- Assume that the training dataset is of size N, we sample the training dataset N times randomly. After that we get a new training dataset of size N.
- Assume that we are having M features for each training data. Build a decision tree by only picking K features randomly $(K \ll M)$
- Repeat the second step various time and we get a random forest.
- The prediction of the random forest is the prediction that most decision tree made in the forest.

Random forest is more robust than traditional decision tree and can make prediction of higher accuracy.

4) Naive Regression: Naive regression[4] is consider as a direct and immediate regression approach where the training error is

$$\text{err}_{\text{train}}(\underline{w}) = \frac{1}{m} \left\| \underline{y} - X\underline{w} \right\|^2 \tag{3}$$

Naive regression model is trained by minimizing the training error by finding the optimal $\underline{w}$.

5) Ridge Regression: Ridge regression[5] is based on naive regression whose training error is defined as

$$\mathrm{err}_{\mathrm{train}}(\underline{w}) = \frac{1}{m} \left\|\underline{y} - X\underline{w}\right\|^2 + \lambda \left\|\underline{w}\right\|_2^2 \tag{4}$$

By adding additional expression $\lambda \left\|\underline{w}\right\|_2^2$, ridge regression is able to prune the irrelevant features better than naive least square regression model by setting their wight to a very small number.

6) Lasso Regression: Lasso regression[6] is also based on naive regression whose training error is defined as

$$\mathrm{err}_{\mathrm{train}}(\underline{w}) = \frac{1}{m} \left\|\underline{y} - X\underline{w}\right\|^2 + \lambda \left\|\underline{w}\right\|_1 \tag{5}$$

By adding additional expression $\lambda \left\|\underline{w}\right\|_1$, ridge regression is able to prune the irrelevant features and those correlated features efficiently, which is pretty similar to ridge regression. However, it also tends to prune features with small weights, even though they are significant features.

7) Basic Completion: Basic Completion strategy is design as the baseline for other machine learning methods. Its logic is pretty simple but direct.

- If the missing feature is discrete, fill missing value with the mode of feature.
- If the missing feature is continuous, fill missing value with the mean of feature.

As a result, The predicting value will have the typical properties of the data set.

8) Best Model: Basing on all methods above, the model is able to find the best model for every tasks for the user. In order to do this, training data is further partitioned into smaller training data and validation data. Then the model iterate through all methods and select best model by testing them on validation data.

For classification tasks, the performance of the model is represents by its accuracy.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

Where TP, TN, FP and FN stands for number of true positive, true negative, false positive and false negative respectively.

For regression task, the performance is evaluated using root mean square error.

$$\text{RMSE}(y, y') = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2} \tag{7}$$

This strategy will find the model with highest accuracy or smallest root mean square error for the current task and use it for prediction.

III. Experiments

A. Dataset

The dataset I use is called Sloan Digital Sky Survey DR14[7], the data consists of 10,000 observations of space taken by the SDSS. Every observation is described by 17 feature columns and 1 class column which identifies it to be either a star, galaxy or quasar. There are 4 discrete features one is of string vale and two is of integer value and 14 continuous features in total, detailed information is shown in table I.This dataset is accessible using the link:
https://www.kaggle.com/lucidlenn/sloan-digital-sky-survey

Feature	Feature Type	Description
objid	Continuous	Object Identifier
ra, dec	Continuous	Coordinate Information
u, g, r, i, z	Continuous	Bands of the Telescope
field	Continuous	Field Number
run, rerun, camcol	Discrete	Fields Information
specobjid	Continuous	Object Identifier
class	String	Label Information
redshift	Continuous	Physical phenomenon
plate, fibered	Continuous	ID Information
mjd	Continuous	Time Information

TABLE I
DATASET DESCRIPTION

Notice that the feature *objectid* and *specobjid* are all features related to id, so we are not going to consider them as the features with missing values. So we are going to regard them as non-missing features and feed them into the model.

B. Experiment Setup

Users are allowed to change the parameters of machine learning model by editing the configuration file *configure.txt*. All supported parameters are listed in the table II.

Parameter	Description	Default
model-k	Max Distinct Value of Discrete Features	25
model-n	Sample Size of Features	100
knn-k	Value of k for KNN	20
dt-max_depth	Max Depth of Decision Tree	10
dt-min_num	Min Size of Data in Tree Node	-1
dt-k	Interval of Continuous Feature	10
rf-n	Number of Features for Each Tree	10
rf-k	Interval of Continuous Feature	10
rf-forest_size	Forest Size	40
ridge-lambda	Value of λ for Ridge Regression	0.1
lasso-lambda	Value of λ for Lasso Regression	0.01

TABLE II
PARAMETER DESCRIPTION

C. Executing the Tool

The tool is implemented using python and external libraries that I use are *argparse*, *numpy*, *csv* and *math*. All machine learning algorithms are implemented from scratch.

The python file takes 6 arguments shown in table III.

Argument	Description	Default
file_path	Target dataset	-
header	Whether table has header	True
target	List of target columns	-
using	List of columns using	-
classification_model	Classification model	basic_completion
regression_model	Regression model	basic_completion

TABLE III
ARGUMENT DESCRIPTION

The list of supported classification model includes: *knn*, *decision_tree*, *random_forest*, *basic_completion* and *best_model*.

The list of supported regression model includes: *naive_regression*, *ridge_regression*, *lasso_regression*, *basic_completion* and *best_model*.

For example, if we want to impute columns $[1, 2, 5]$ with information from columns $[3, 4, 6]$, using decision tree as classification model and naive regression as regression model, just execute:

Notice that although $[3, 4, 6]$ are used as training data, they are allowed to have missing value. If such situations are detected, the model would try to impute their missing value before checking columns $[1, 2, 5]$.

```
Impute.py --file_path 'a.csv' --target 1 2 5 --using 3 4 6
--classification_model decision_tree --regression_model
naive_regression
```

D. Result and Analysis

In the following experiment, we set all parameter as default and evaluate the performance of the model for each feature. The root mean square error of each continuous features are shown in table IV. An interesting observation is that the model is having very good performance on some features while relatively poor performance on the others. The main reason behind this is that not all features in the dataset are correlated with each other.

Feature	Naive	Ridge	Lasso	Baseline
ra	36.089	36.863	36.519	47.764
dec	14.087	14.499	14.158	25.205
u	0.218	0.460	0.399	0.828
g	0.104	0.251	0.330	0.945
r	0.094	0.282	0.569	1.067
i	0.149	0.352	0.679	1.142
z	0.164	0.439	0.164	1.203
field	100.998	104.249	100.998	162.551
redshift	0.202	0.204	0.202	0.388
plate	3.287	278.975	3.287	1787.178
fibered	165.188	166.538	165.188	206.278
mjid	346.945	379.162	346.945	1511.021

TABLE IV
RMSE OF REGRESSION TASKS

This observation can also be made for categorical features. Table V shows the classification accuracy of the model for each features. Although the overall accuracy is very high, it is still obviously that model can make better prediction for features such as run and $class$.

Feature	KNN	DT	RF	Baseline
run	0.941	0.974	0.970	0.306
rerun	1.000	1.000	1.000	1.000
camcol	0.647	0.636	0.601	0.166
class	0.877	0.873	0.888	0.500

TABLE V
ACCURACY OF CLASSIFICATION TASKS

Figure 3 shows the features that can be predicted successfully by the model.

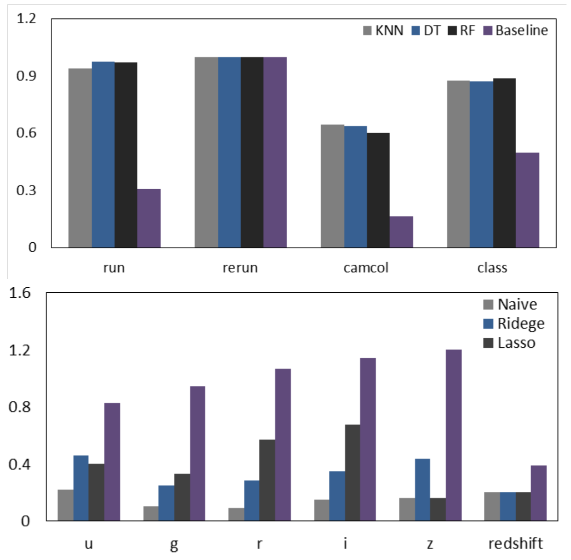

Fig. 3. Valuable features

This meets our expectation as if we check the description of these features, we can find out that:

- *u*, *g*, *r*, *i*, *z* represent the response of the 5 bands of the telescope.
- *run rerun* and *camcol* are features providing information describing a field within an image taken.
- *redshift* is related to a physical phenomenon, which can also have influence on the experiment.

Figure 4 shows the features that cause bad overall predicting performance.

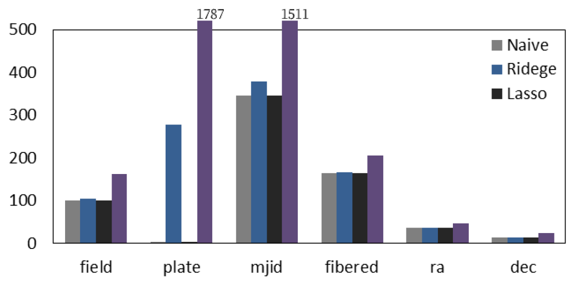

Fig. 4. Invaluable features

By taking a closer look to them we can find out that:

- These features can be useless information such as id or time. For example, *filed*, *fibered* and *plate* are id information and *mjid* is time information.
- The features *ra* and *dec* are not strictly related to the experiment as they only provide coordinate information while the aim of this dataset is to classify whether an object is a galaxy, star or quasar.

Basing on this observation, we can categorize all features into two groups: valuable features and invaluable features. valuable features are those features that are highly relevant to each others. They are also those features that machine learning algorithms would give highest weight on. On the contrary, invaluable features are the features that are irrelevant to each others. When fed into machine learning algorithms, this kind of features would be pruned or given smallest weight. The observing of valuable and invaluable features can give us a better understanding of the dataset.

IV. Conclusion and Future Work

A. Conclusion

For this project, I designed and implemented a tool performing data completion and interpolation. The tool is easy to use by executing the python file with at most 6 arguments. Meanwhile, parameters related to the model is accessible with a configuration file directly, which makes it easy for users to edit. The tool is also powerful as users can choose from multiple predicting algorithm or simply ask the tool to pick the best model for them. By experiment, we also show that the model is able to make good prediction and help separate valuable and invaluable features.

B. Future Work

Although we are prove that the tool is powerful and easy to use, some improvements can be made to bring it to a higher level.

Firstly, we can add more machine learning algorithms to the support list. This is very easy as data prepossessing has already been done separately and all we need to do is feeding the data into new algorithms directly.

Secondly, the parameters of machine learning algorithms are currently stored in a configuration file. It would be better if the tool can scan the dataset and generate suitable parameters for the users automatically.

Lastly, it would be interesting if we can extend the model by asking it to generate new rows. This may show us how the model is understand the dataset and reveal the hidden relationship among features.

References

[1] E. Fix, *Discriminatory analysis: nonparametric discrimination, consistency properties*. USAF school of Aviation Medicine, 1985, vol. 1.

[2] J. R. Quinlan, "Induction of decision trees," *MACH. LEARN*, vol. 1, pp. 81–106, 1986.

[3] Tin Kam Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, 1995, pp. 278–282 vol.1.

[4] H. L. Seal, "Studies in the history of probability and statistics. xv the historical development of the gauss linear model," *Biometrika*, vol. 54, no. 1-2, pp. 1–24, 1967.

[5] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[6] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[7] B. Abolfathi, D. Aguado, G. Aguilar, C. A. Prieto, A. Almeida, T. T. Ananna, F. Anders, S. F. Anderson, B. H. Andrews, B. Anguiano *et al.*, "The fourteenth data release of the sloan digital sky survey: First spectroscopic data from the extended baryon oscillation spectroscopic survey and from the second phase of the apache point observatory galactic evolution experiment," *The Astrophysical Journal Supplement Series*, vol. 235, no. 2, p. 42, 2018.