

CmpE 544: Term Project

Comparison of Imputation Algorithms

Due on Jan 12, 10:00

E. ALPAYDIN

Arman Boyacı

armanboyaci@gmail.com

Contents

1	Introduction	3
2	Theoretical Background	4
2.1	Data with Mixed Variables	4
2.2	Missing Data	4
2.2.1	Ad-hoc Methods	4
2.2.2	Modern Approach	5
3	Data	5
3.1	About Data	5
3.2	Preprocessing	6
4	Classification Algorithm: KNN	7
4.1	Results	8
5	Missingness: Different Scenarios	9
5.1	Results	9
6	Discussion	10
7	Conclusion	10

1 Introduction

Missing data can be occurred from several reasons: lost surveys, refusal to answer some survey questions, skipped questions, procedural mistakes or computer malfunctions. One can try to ignore those missing entries however this approach may be misleading especially for small data sets. Samples containing missing values also hold some value and the information can be easily recovered by imputation. In this project my main aim is to compare different imputations strategies. The dataset that I've chosen contains both numerical and categorical variables. So, my second aim is to deal with mixed variables.

In order to compare imputation algorithms I need a classifier, since they will be compared according to their accuracy rates. As a classifier, k nearest neighbor algorithm is chosen. First, classification algorithm will be run on full data, then some of the values will be missed randomly on training set. Algorithm will be executed on clean data, which means data containing no missing value. Finally missing values will be forecasted with some imputation techniques and will be filled up and accuracy will be recalculated on this imputed data.

2 Theoretical Background

2.1 Data with Mixed Variables

In machine learning, data often contains both numeric and categorical values. Categorical values cannot be considered as numerical, in other words labeling categories will not produce meaningful results many categorical domains are not ordered.

2.2 Missing Data

Missing data can effect properties of estimators (mean, percentage, variance, regression parameter values as well as confidence intervals). As a consequence, classification process can also be effected.

Missigness Mechanism There exist three missingness mechanism: Missing At Random(MAR), Missing Completely At Random (MCAR), Missing Not At Random (MNAR). Let r be a random variable which represents missingness of a random variable. Data Y can be splitted into observer and missing parts denoted by $Y = (Y_O, Y_M)$. In MCAR situation $Pr(r|Y_O, Y_M) = Pr(r)$. In MAR situation, $Pr(r|Y_O, Y_M) = Pr(r|Y_O)$. And finally in MNAR situation $Pr(r|Y_O, Y_M) = Pr(r|Y_O, Y_M)$.

2.2.1 Ad-hoc Methods

They create a single complete dataset. The answers are invalid unless some strong assumptions hold. Typical problem is that we can not reflect the real variance of the variable by imputing single value.

Analysis of Completers Only We remove data containing missing values. When the missing observations are not a completely random selection of data, invalid inferences can be made.

Imputation of Simple and Class Mean For the categorical variables, we can use the median. In this situation, variances will be wrongly estimated.

Imputation of Regression Mean We can generate unbiased estimates of means but the variability of the imputation is too small, so the estimated precision of regression coefficients will be wrong and inferences will be misleading. Artificially inflate correlations. Standard errors too small. Artificially p-values low.

2.2.2 Modern Approach

Multiple Imputation One of the main problems with the single stochastic imputation methods is the need for developing appropriate variance formulae for each different setting. Multiple imputation attempts to provide a procedure that can get the appropriate measures of precision relatively simply in any setting. We use this relationship to complete the data set by drawing the missing observations from the distribution of $Y_{miss}|Y_{observed}$. We do this K (typically 5) times, giving rise to K (typically 5) complete data sets. We analyse each of these data sets in the usual way. We combine the results using particular rules.

If we knew the missing values, then estimating the model parameters would be straightforward. Similarly, if we knew parameters of the data model, then it would be possible to obtain unbiased prediction for the missing values. An iterative method can be used: first predict the missing values based on assumed values for the parameters, use these predictions to update the parameter estimates, and repeat. The sequence of parameters converges to maximum-likelihood estimates.

Unit	Data		Imputation 1		Imputation 2		Imputation 3		Imputation 4	
	Y_M	Y_O	Y_M	Y_O	Y_M	Y_O	Y_M	Y_O	Y_M	Y_O
1	1.1	3.4	1.1	3.4	1.1	3.4	1.1	3.4	1.1	3.4
2	1.5	3.9	1.5	3.9	1.5	3.9	1.5	3.9	1.5	3.9
3	2.3	2.6	2.3	2.6	2.3	2.6	2.3	2.6	2.3	2.6
4	3.6	1.9	3.6	1.9	3.6	1.9	3.6	1.9	3.6	1.9
5	0.8	2.2	0.8	2.2	0.8	2.2	0.8	2.2	0.8	2.2
6	3.6	3.3	3.6	3.3	3.6	3.3	3.6	3.3	3.6	3.3
7	3.8	1.7	3.8	1.7	3.8	1.7	3.8	1.7	3.8	1.7
8	?	0.8	0.2	0.8	0.8	0.8	0.3	0.8	2.3	0.8
9	?	2.0	1.7	2.0	2.4	2.0	1.8	2.0	3.5	2.0
10	?	3.2	2.7	3.2	2.5	3.2	1.0	3.2	1.7	3.2

3 Data

3.1 About Data

The data was extracted from the census bureau database found at <http://www.census.gov/ftp/pub/DES>. Prediction task is to determine whether a person makes over 50K a year.

Variables:

- Contionous: age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week

- Ordinal: education
- Nominal: workclass, marital-status, occupation, relationship, race, sex, native-country

Variables are in different scale, for instance: First variable is in interval [17,90], third variable is in interval [12285;1484705]. Variances of variables can not be assumed equal: Variance of variable one is equal to 186, on the other hand, second variable's variance is 5.

Data also contains an information read-me file. It gives us results for different algorithms. The following table contains error rates of some of the algorithms.

Algorithm	Error %
C4.5	15.54
Naive-Bayes	16.12
Nearest-neighbor(1)	21.42
Nearest-neighbor(3)	20.35

3.2 Preprocessing

Data contains a mixture of variables: categorical and numeric. We should first think about how all these variable will be handled. Arbitrary given labeling for categorical variables will be misleading. For instance, native-country variable contains more than 30 category. They are not comparable to each other. In the literature, for this kind of situations (handling categorical variables), one of the solution is that categorical variables are splitted into indicator variables. We have done it the same process in my Statistical Inference course, last year. We were using a free statistical tool: it has a build-in function for handling categorical variables by splitting them indicator variables.

For the simplification some of the categories are merged. For instance, native-country variable contains more than 30 category (country) , but I merged them into US and non-US.

marital status: Married(All Others) , Never-Married(Never-Married)

native-country: US(United-States) , Non-US(All Others)

education:

High(Masters,Doctorate) ,

Medium(11th, 9th, 12th, 10th, Bachelors, Some-college, HS-grad, Prof-school) ,

Low(1st-4th, 5th-6th, 7th-8th, Preschool)

workclass:

Private(Private, Self-emp-not-inc, Self-emp-inc),

Gov(Federal-gov, Local-gov, State-gov),

Without-Pay (Without-pay, Never-worked)

Variables after simplification:

- Contionous: age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week
- Binary: marital-status
- Ordinal: education, workclass
- Nominal: occupation, relationship, race, sex, native-country

For ordered variables, the assumption is that the distance between classes are equal(for instance the distance between low-medimum and medium-high for education variable). Nominal variables containing k category will be transformed to k independent indicator variables. As a consequence, variable number has increased to 42 (raw data contains 14 variables).

Gender	Male	Female
'Male'	1	0
'Female'	0	1
'Male'	1	0
...

4 Classification Algorithm: KNN

The nearest-neighbor is a non-parametric method where a new observation is classified according to closest observation in the training set. The determination of similarity is based on distance measures (typically euclidean or mahalanobis distance measures).Data contains both numerical and categorical values. Categorical values are converted to binary variables. On the other hand, numerical variables are in different scales. They should be normalized. In general we use mahalanobis distance for the normalization. However in this case covariance matrix is not invertible, possible because of presence of categorical variables. Therefore, I'll try different methods: (1) Normalizing with maximum value of each variable. Binary variable stay the same since maximum value is 1. Other variables are [0,1] interval. (2) Normalizing with the variance of the variable are applied only on numerical values. Both training and test sets are normalized.

After being normalized, I should look for k nearest neighbors of each element in the test test. (a) Euclidean distance is my first choice. Secondly, I tried (b) weighted Euclidean distance. Weights are determined according to absolute value

of the correlation coefficients between each variable with the class identifier. My assumption is that if a variable has a high correlation with the class identifier then it must have more importance when we calculate the distance.

4.1 Results

All in all, I should determine three things: Normalization method(1,2), Distance Measure (a,b) and the hyper-parameter k of the algorithm.

In this section a portion of the full dataset is used. Training set size was 4000 and test set size was. 2000. I started by determining convenient k value for the algorithm. Euclidean distance is used and variables are normalized according to their max value.

k	Accuracy	Accuracy %
1	1458	72.90
3	1519	75.95
5	1550	77.50
7	1561	78.05
9	1574	78.70

Then, I tried different normalization techniques. k=5 and euclidean distance is used.

Normalization	Accuracy	Accuracy %
Non	1185	59.25
Max	1550	77.50
Var	1548	77.40

Finally, Euclidean and weighted Euclidean distance measures are compared for k=5 and normalization with max values.

Distance Measure	Accuracy	Accuracy %
Euclidean	1550	77.50
Weighted Euclidean	1541	77.05

For the next section, k-nearest neighbor algorithm will be used with the following setting k=5, Euclidean distance and normalization with the max values. I also tested k-nn on full data.

k	Accuracy	Accuracy %	Error %
1	11848	78.67	21.33
3	12169	80.80	19.20
5	12292	81.62	18.37

I have slightly better results than results reported on read-me file.

5 Missingness: Different Scenarios

In this section, I will talk about missing data situations. For the simplification each time only one variable will have missed values. This assumption does not play a role for the loss of accuracy because we throw all the missing rows, it does not matter whether there are more than one variable or not. However it can be important for the imputation process. I have three questions in my mind. (1) How much accuracy am I loosing for the specific missing rate. (2) Is there any dependency on which variable we have missing values. (3) For which training set size, we can omit samples having missing values.

As the samples are iid, I assumed that missing values are occurred in the first m samples which is calculated according to missing rate value.

5.1 Results

Training Set Size ¹ Runned 5 times, Average values, Variable: 'Capital Gain', Test Set Size: 15060

Training Set Size	Full Data	5%	25%	50%
20	9857	9802	9806	9944
40	10121	10221	10171	9315
50	10091	10051	10021	9733
100	10554	10595	10447	10490
200	10847	10845	10816	10436

Loss Rates Loss Rate = (Full - Clean) / Full

Training Set Size	5%	25%	50%
20	0.5	0.5	(0.8)
40	(0.9)	(0.49)	7.9
50	0.39	0.69	3.5
100	(0.38)	1.01	0.6
200	0.01	0.28	3.78

Different Variables Training Set Size = 50, Error Rate 50, Run 5

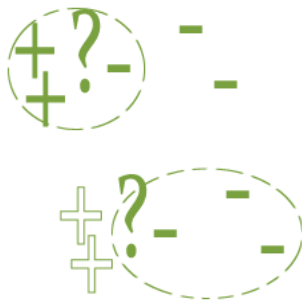
¹A result in parantheses means the opposite sign of other results

Variable	Full	Cleaned 50%	Loss Rate
'Capital-Gain'	10091	9733	3.5
'Education-num'	10162	10595	(4.2)
'Hours-per-week'	10371	9017	13.0

Different Imputation Algorithms Gain Rate = (Algo - Clean) / Clean

Technique	Gain Rate %
Mean Imputation	9.8
Class-Mean	9.9
Regression	5.8

6 Discussion



- In some cases, accuracy of the algorithm increases when we remove missing data. This is quite surprising since theoretically we should lose statistical power when we have smaller data. The reason of this can be illustrated on the figure above.
- We can not conclude that when we decrease training set size, loss rate is increasing.
- There is not enough information to say that class-mean imputation is better than mean imputation, both better than regression imputation.
- According to these results, I can say that for a missing rate 5%, omitting missing values does not decrease significantly our accuracy even for small training set size. An imputation technique can be necessary only for 50% missing rate.

7 Conclusion

In this study, I tried to compare different imputation algorithms. KNN algorithms has been chosen as a classifier to measure the performance. First, the implementation

of the algorithm causes some problem since data contains both numerical and categorical variables and moreover numerical variables are in different scales. Categorical variables are transformed to binary variables and applied normalization techniques to rid of different scales. In the imputation part, I noticed that there was not enough loss of statistical power. In some situations, performance of the classifier has been increased even when 50% of the training data set were missing. According to the results we can say that we should prefer mean imputation over regression imputation. As a future work, a different algorithm can be chosen as a classifier (possibly a parametric classifier). Results should also be verified on new data. If we encounter a situation that loss of accuracy is relatively big and ad-hoc methods like mean imputation and regression imputations methods are not enough to gain power we can try Multiple Imputation method. Moreover, instead of classification, regression situation can be considered.

References

- <http://www.missingdata.org.uk>
- Eric R. Buhi, Patricia Goodson, Torsten B. Neilands, Out of Sights: Not Out of Mind: Strategies for Handling Missing Data
- Z. Huang, Clustering Large Data Sets with Mixed Numeric and Categorical Values
- J.L. Schafer, M.K. Olsen, Multiple Imputation for Multivariate Missing-Data Problems: A Data Analyst's Perspective, Multivariate Behavioral Research, 1998

Matlab Code

Listing 1: Generating Data

```
function [training_set test_set] = my_data()
% Data
data_train = importdata('adult.data');
data_test = importdata('adult.test');
5 % Some Calculations
% Priorities
% =< 50k (Class 1)
Prior_1_train = size(data_train(data_train(:,15)==1),1);
Prior_1_test = size(data_test(data_test(:,15)==1),1);
10 % > 50k (Class 2)
Prior_2_train = size(data_train(data_train(:,15)==2),1);
Prior_2_test = size(data_test(data_test(:,15)==2),1);
% Clean Data (Without Missing Values)
% Training
15 keep_matrix = [];
for i=1:size(data_train,1)
    keep = 1;
    for j = 1:size(data_train,2)
        if data_train(i,j) == -1
20             keep = 0;
        end
    end
    if keep == 1
        keep_matrix = [keep_matrix i];
25    end
end
train_c = data_train(keep_matrix,:);
% Test
keep_matrix = [];
30 for i=1:size(data_test,1)
    keep = 1;
    for j = 1:size(data_test,2)
        if data_test(i,j) == -1
35             keep = 0;
        end
    end
    if keep == 1
        keep_matrix = [keep_matrix i];
    end
end
```

```
40 end
test_c = data_test(keep_matrix,:);
% Preprocessing
% M: Merged, I: Indicator, C: Clean, N: Normalized
% Simplification on categorical variables (merging)
45 train_m = merge(data_train);
test_m = merge(data_test);
train_cm = merge(train_c);
test_cm = merge(test_c);

50 % Change Order: First Continuous Variables Second Ordered Variables
% Third Categorical Variables
train_mi = indicator(train_m);
test_mi = indicator(test_m);
train_cmi = indicator(train_cm);
55 test_cmi = indicator(test_cm);

% Normalizing
max_values_train = max(train_cmi);
max_values_test = max(test_cmi);

60 %Numerical Variables
num_var = [1 5 9 38 39 40];

avg_values_train = mean(train_cmi);
65 avg_values_test = mean(test_cmi);

cov_train = cov(train_cmi);
cov_test = cov(test_cmi);

70 var_train = diag(cov_train);
var_test = diag(cov_test);

train_cmin = train_cmi;
test_cmin = test_cmi;

75 for i=1:size(train_cmi,2)
%for i=1:size(num_var,2)
train_cmin(:,i) = train_cmi(:,i)/max_values_train(i);
test_cmin(:,i) = test_cmi(:,i)/max_values_test(i);

80 % train_cmin(:,num_var(i)) = train_cmi(:,num_var(i))/avg_values_train(n
```

```

%     test_cmin(:,num_var(i)) = test_cmi(:,num_var(i))/avg_values_test(num_v

%     train_cmin(:,num_var(i)) = train_cmi(:,num_var(i))/var_train(num_var(i)
85 %     test_cmin(:,num_var(i)) = test_cmi(:,num_var(i))/var_test(num_var(i));
end

% Shuffle
randomize_train = randperm(size(train_cmin,1));
90 randomize_test = randperm(size(test_cmin,1));
train_cmin(:, :) = train_cmin(randomize_train, :);
test_cmin(:, :) = test_cmin(randomize_test, :);

% Detect Classes
95 train_class_1 = find(train_cmin(:,43)==0);
train_class_2 = find(train_cmin(:,43)==1);
test_class_1 = find(test_cmin(:,43)==0);
test_class_2 = find(test_cmin(:,43)==1);

100 % Select 4000 sample from training set, equal prior
temp_train = [train_class_1(1:25); train_class_2(1:25) ];
% Shuffle
randomize_train = randperm(50);

105 % Select 2000 sample form test set, equal prior
temp_test = [test_class_1(1:2000); test_class_2(1:2000) ];
% Shuffle
randomize_test = randperm(4000);

110 training_set = train_cmin( temp_train(randomize_train) ,:);
test_set = test_cmin( temp_test(randomize_test) ,:);

% Full Data Set
% training_set = train_cmin;
115 test_set = test_cmin;

end

```

Listing 2: Merging Procedure

```

function data_new = merge(data_old)
data_new = data_old;
% column 2

```

```
temp = data_old(:,2);
5 data_new(temp==2,2) = 1;
  data_new(temp==3,2) = 2;
  data_new(temp==4,2) = 2;
  data_new(temp==5,2) = 2;
  data_new(temp==6,2) = 3;
10 data_new(temp==7,2) = 3;
  data_new(temp==8,2) = 3;
  % column 4
temp = data_old(:,4);
  data_new(temp==2,4) = 1;
15 data_new(temp==3,4) = 1;
  data_new(temp==4,4) = 1;
  data_new(temp==5,4) = 2;
  data_new(temp==6,4) = 2;
  data_new(temp==7,4) = 2;
20 data_new(temp==8,4) = 2;
  data_new(temp==9,4) = 2;
  data_new(temp==10,4) = 2;
  data_new(temp==11,4) = 2;
  data_new(temp==12,4) = 2;
25 data_new(temp==13,4) = 2;
  data_new(temp==14,4) = 2;
  data_new(temp==15,4) = 3;
  data_new(temp==16,4) = 2;
  % column 6
30 temp = data_old(:,6);
  data_new(temp==5,6) = 1;
  data_new(temp==1,6) = 2;
  data_new(temp==2,6) = 2;
  data_new(temp==3,6) = 2;
35 data_new(temp==4,6) = 2;
  data_new(temp==6,6) = 2;
  data_new(temp==7,6) = 2;
end
```

Listing 3: Creating Indicator Variables

```
function data_new = indicator(data_old)
% age
data_new(:,1) = data_old(:,1);
% workclass
```

```
5 data_new(:,2) = (data_old(:,2)==1);
data_new(:,3) = (data_old(:,2)==2);
data_new(:,4) = (data_old(:,2)==3);
% fnlwt
data_new(:,5) = data_old(:,3);
10 % education
data_new(:,6) = (data_old(:,4)==1);
data_new(:,7) = (data_old(:,4)==2);
data_new(:,8) = (data_old(:,4)==3);
% education-num
15 data_new(:,9) = data_old(:,5);
% marital-status
data_new(:,10) = (data_old(:,6)==1);
% occupation
data_new(:,11) = (data_old(:,7)==1);
20 data_new(:,12) = (data_old(:,7)==2);
data_new(:,13) = (data_old(:,7)==3);
data_new(:,14) = (data_old(:,7)==4);
data_new(:,15) = (data_old(:,7)==5);
data_new(:,16) = (data_old(:,7)==6);
25 data_new(:,17) = (data_old(:,7)==7);
data_new(:,18) = (data_old(:,7)==8);
data_new(:,19) = (data_old(:,7)==9);
data_new(:,20) = (data_old(:,7)==10);
data_new(:,21) = (data_old(:,7)==11);
30 data_new(:,22) = (data_old(:,7)==12);
data_new(:,23) = (data_old(:,7)==13);
data_new(:,24) = (data_old(:,7)==14);
% relationship
data_new(:,25) = (data_old(:,8)==1);
35 data_new(:,26) = (data_old(:,8)==2);
data_new(:,27) = (data_old(:,8)==3);
data_new(:,28) = (data_old(:,8)==4);
data_new(:,29) = (data_old(:,8)==5);
data_new(:,30) = (data_old(:,8)==6);
40 % race
data_new(:,31) = (data_old(:,9)==1);
data_new(:,32) = (data_old(:,9)==2);
data_new(:,33) = (data_old(:,9)==3);
data_new(:,34) = (data_old(:,9)==4);
45 data_new(:,35) = (data_old(:,9)==5);
% sex
```

```

data_new(:,36) = (data_old(:,10)==1);
data_new(:,37) = (data_old(:,10)==2);
% capital-gain
50 data_new(:,38) = data_old(:,11);
% capital-loss
data_new(:,39) = data_old(:,12);
% hours-per-week
data_new(:,40) = data_old(:,13);
55 % native-country
data_new(:,41) = (data_old(:,14)==1);
data_new(:,42) = (data_old(:,14)==2);
% 50k
data_new(:,43) = (data_old(:,15)==2);
60 end

```

Listing 4: Euclidean Distance

```

function y = my_dist(x,0)
% Last column contains class information
% Do the calculation without it
%temp = zeros(size(O,1), (size(O,2)-1) );
5 temp = (repmat(x, size(O,1),1)-O(:,1:(end-1))));
y = sum(temp.^2,2);
end

```

Listing 5: Weighted Euclidean Distance

```

function y = my_dist2(x,0)
% Last column contains class information
% Do the calculation without it
%temp = zeros(size(O,1), (size(O,2)-1) );
5 kor = corrcoef(O);
% Corrolation Coefficients are the weights
w = abs(kor(1:(end-1),end));
temp = (repmat(x, size(O,1),1)-O(:,1:(end-1))));
y = (temp.^2)*w;
10 end

```

Listing 6: K Nearest Neighbor Algorithm

```

function [accuracy] = my_knn(k,train_set,test_set)
% Classification
% K-NN

```

```

performance = 0;
5 for i=1: size (test_set,1)
    %for i=1:100
        dist_vector = my_dist (test_set (i,1:(end-1)),train_set);
        [value nn] = sort (dist_vector);
        estimated_class = mode (train_set (nn(1:k), end));
10 if estimated_class == test_set (i,end)
            performance = performance + 1;
        end
        if rem (i,100) == 0
            %disp(i);
15 end
end
accuracy = performance;
end

```

Listing 7: Imputation Procedure

```

function [mean_imp class_mean regr_imp clean]=imputation(training_set,rate)
    variable=40;
    % randperm ile farklı datalar seçilecek!
    temp = ceil ((rate/100)*size (training_set,1));
5 % Clean Data Set
    clean = training_set (temp+1:end,:);
    % Mean Imputation
    mean_imp = training_set;
    mean_imp (1:temp,variable) = ...
10         repmat (mean (training_set (temp+1:end,variable)),temp,1);
    % Class Mean Imputation
    class_mean = training_set;

    temp1 = find (training_set (temp+1:end,end)==0);
15    temp2 = find (training_set (temp+1:end,end)==1);

    avg1 = mean (training_set (temp1,variable));
    avg2 = mean (training_set (temp2,variable));

20    temp3 = find (training_set (1:temp,end)==0);
    temp4 = find (training_set (1:temp,end)==1);

    class_mean (temp3,variable) = avg1;
    class_mean (temp4,variable) = avg2;

```

```

25      % Multivariate Linear Regression
      regr_imp = training_set;
      temp_vector = [1:(variable-1) (variable+1):(size(training_set,2)-1)];
      beta = regress(training_set(temp+1:end,variable), ...
30          training_set(temp+1:end,temp_vector));
      regr_imp(1:temp,variable) = regr_imp(1:temp,temp_vector) * beta;
end

```

Listing 8: Generating Results

```

function [x y] = main(sim_length)
for i=1:sim_length
    [training_set test_set] = my_data();
    result(i,1) = my_knn(5,training_set,test_set);
5    [mean_imp class_mean regr_imp clean] = imputation(training_set,5);
    result(i,2) = my_knn(5,clean,test_set);
    result(i,3) = my_knn(5,mean_imp,test_set);
    result(i,4) = my_knn(5,class_mean,test_set);
    result(i,5) = my_knn(5,regr_imp,test_set);
10    disp(['%5 ' 'Clean: ' int2str(result(i,2)) ...
          ' Mean: ' int2str(result(i,3)) ...
          ' Class-Mean: ' int2str(result(i,4)) ...
          ' Regression: ' int2str(result(i,5)) ] )
    [mean_imp class_mean regr_imp clean] = imputation(training_set,25);
15    result(i,6) = my_knn(5,clean,test_set);
    result(i,7) = my_knn(5,mean_imp,test_set);
    result(i,8) = my_knn(5,class_mean,test_set);
    result(i,9) = my_knn(5,regr_imp,test_set);
    disp(['%25 ' 'Clean: ' int2str(result(i,6)) ...
20          ' Mean: ' int2str(result(i,7)) ...
          ' Class-Mean: ' int2str(result(i,8)) ...
          ' Regression: ' int2str(result(i,9)) ] )
    [mean_imp class_mean regr_imp clean] = imputation(training_set,50);
    result(i,10) = my_knn(5,clean,test_set);
25    result(i,11) = my_knn(5,mean_imp,test_set);
    result(i,12) = my_knn(5,class_mean,test_set);
    result(i,13) = my_knn(5,regr_imp,test_set);
    disp(['%50 ' 'Clean: ' int2str(result(i,10)) ...
30          ' Mean: ' int2str(result(i,11)) ...
          ' Class-Mean: ' int2str(result(i,12)) ...
          ' Regression: ' int2str(result(i,13)) ] )

```

```
end  
x = result;  
y = result*(100/size(test_set,1));  
35 end
```