Heart Disease Prediction

Kelompok Bernat tidur 2

Rahmat Nur Ibrahim Santosa / 13516009

Michelle Eliza Gananjaya / 13516015

Erma Safira Nurmasyita / 13516072

Daniel Ryan Levyson / 13516132

Rinda Nur Hafizha / 13516151

Dataset Cleaning

Metode Cleaning

Pertama-tama, dihitung jumlah nilai `'?'` pada tiap kolom. Jika banyaknya melebihi setengah dari jumlah data, maka kolom tersebut tidak diikutsertakan pada proses training.

Untuk menangani nilai `'?'` pada setiap kolom lainnya, jika nilai suatu kolom numerikal, nilai `'?'` diganti menjadi nilai median dari nilai pada kolom tersebut. Jika niali suatu kolom kategorikal, nilai `'?'` diganti menjadi nilai modus dari nilai-nilai pada kolom tersebut.

In [1]:

```python
# IMPORT DATASET TRAIN AND TEST
import pandas as pd
import numpy as np
dataset = pd.read_csv('data/tubes2_HeartDisease_train.csv')
test_set = pd.read_csv('data/tubes2_HeartDisease_test.csv')
```

In [2]:

```python
# DATASET ClEANING
# Column Dataset Train, No. of '?' values
## Column 4 = 46 kosong - numerical
## Column 5 = 24 - numerical
## Column 6 = 78 - categorical
## Column 7 = 1 - categorical, replace with 0
## Column 8 = 43 - numerical
## Column 9 = 43 - categorical
## Column 10 = 48 - numerical
## Column 11 = 261 - categorical
## Column 12 = 513 - DROP
## Column 13 = 407 - DROP

# Drop column 12 and 13
import math
dataset = dataset.drop(columns=['Column12', 'Column13'], axis=1)
test_set = test_set.drop(columns=['Column12', 'Column13'])

# Replace '?' value in Column7 with 0
data_len = len(dataset['Column7'])
for i in range(0, data_len):
    if dataset['Column7'][i] == '?' or math.isnan(float(dataset['Column7'][i])):
        dataset['Column7'][i] = '0'
```

```
c:\users\hp\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:23:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

In [3]:

```
# Replace '?' with nan
col = 'Column'
for i in range(1, 12):
    column = col + str(i)
    for j in range(0, data_len):
        if dataset[column][j] == '?':
            dataset[column][j] = np.nan

#dataset = dataset.drop(deleterow)
```

```
c:\users\hp\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:7: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  import sys
```

In [4]:

```
# Replace nan in categorical data with mode value (Column 6, 9, 11)
mode6 = dataset['Column6'].mode()
mode9 = dataset['Column9'].mode()
mode11 = dataset['Column11'].mode()
for i in range(0, data_len):
    if math.isnan(float(dataset['Column6'][i])):
        dataset['Column6'][i] = mode6
for i in range(0, data_len):
    if math.isnan(float(dataset['Column9'][i])):
        dataset['Column9'][i] = mode9
for i in range(0, data_len):
    if math.isnan(float(dataset['Column11'][i])):
        dataset['Column11'][i] = mode11

# Replace nan in numerical data with median value (Column 4, 5, 8, 10)
median4 = dataset['Column4'].median()
median5 = dataset['Column5'].median()
median8 = dataset['Column8'].median()
median10 = dataset['Column10'].median()
for i in range(0, data_len):
    if math.isnan(float(dataset['Column4'][i])):
        dataset['Column4'][i] = median4
for i in range(0, data_len):
    if math.isnan(float(dataset['Column5'][i])):
        dataset['Column5'][i] = median5
for i in range(0, data_len):
    if math.isnan(float(dataset['Column8'][i])):
        dataset['Column8'][i] = median8
for i in range(0, data_len):
    if math.isnan(float(dataset['Column10'][i])):
        dataset['Column10'][i] = median10
```

```
c:\users\hp\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:7: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  import sys
c:\users\hp\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  # Remove the CWD from sys.path while we load stuff.
c:\users\hp\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:13:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  del sys.path[0]
c:\users\hp\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:22:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
c:\users\hp\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:25:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
c:\users\hp\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:28:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
c:\users\hp\appdata\local\programs\python\python37-32\lib\site-packages\ipykernel_launcher.py:31:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

In [8]:

```python
from random import choice
import numpy as np
import math

def generate_fold(k, dataset):
    dataset_size = len(dataset)
    fold = [[] for i in range (k)]
    nums = [i for i in range(k)]
    sizes = [0 for i in range(k)]
    normal_size = math.floor(dataset_size / k)
    max_size = math.ceil(dataset_size / k)
    size = max_size
    max_size_counter = dataset_size % k

    for val in dataset:
        idx = choice(nums)
        fold[idx].append(val)
        sizes[idx] += 1

        if sizes[idx] == size:
            nums.remove(idx)

        if sizes[idx] == max_size:
            max_size_counter -= 1
            if max_size_counter == 0:
                size = normal_size

                temp = []
                for num in nums:
                    if sizes[num] == size:
                        temp.append(num)
                for t in temp:
                    nums.remove(t)
    return fold

def seperate(dataset):
    params = []
    lables = []
    datasize = len(dataset[0])
    for data in dataset:
        params.append(data[:datasize-1])
        lables.append(data[datasize-1])
    return params, lables

def parse_dataset(frame):
```

```
def parse_dataset(frame):
    dataset = []
    for index, row in frame.iterrows():
        dataset.append(row.values.tolist())
    return dataset

def get_trainingset(index, folds):
    training_set = []
    for i in range(len(folds)):
        if i != index:
            for data in folds[i]:
                training_set.append(data)
    return training_set

def pseudo_clean(dataset):
    cleaned = []
    idx = 0
    for data in dataset:
        row = []
        for val in data:
            row.append(float(val))
        cleaned.append(row)
        idx += 1
    return cleaned
```

In [9]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import tree

def train(trainer, folds):
    classifiers = []
    for i in range (len(folds)):
        training_set = get_trainingset(i, folds)
        training_params, training_lables = seperate(training_set)
        classifiers.append(trainer.fit(training_params, training_lables))
    return classifiers

def train_naive_bayes(folds):
    return train(GaussianNB(), folds)

def train_decision_tree(folds):
    return train(tree.DecisionTreeClassifier(), folds)

def train_knn(neighbor, folds):
    return train(KNeighborsClassifier(neighbor), folds)

def train_mlp(folds):
    return train(MLPClassifier(learning_rate_init=0.01, max_iter=300), folds)
```

In [7]:

```
dataset = parse_dataset(dataset)
dataset = pseudo_clean(dataset)
folds = generate_fold(10, dataset)

nb_models = train_naive_bayes(folds)
dt_models = train_decision_tree(folds)
knn_models = train_knn(3, folds)
mlp_models = train_mlp(folds)
```

In [10]:

```
from sklearn.metrics import accuracy_score

def generate_accuracies(models):
    for i in range(len(models)):
        test_set = folds[i]
        test_params, test_lables = seperate(test_set)
        predictions = models[i].predict(test_params)
        accuracy = accuracy_score(np.array(test_lables), predictions)
        print(str(round(accuracy * 100, 2)) + '%')
```

In [11]:

```
print("Naive Bayes Accuracies:")
generate_accuracies(nb_models)
print("Decision Tree Accuracies:")
generate_accuracies(dt_models)
print("K-Nearest Neighbors Accuracies:")
generate_accuracies(knn_models)
print("Multi-Layer Perceptron Accuracies:")
generate_accuracies(mlp_models)
```

```
Naive Bayes Accuracies:
56.41%
47.44%
52.56%
70.51%
53.85%
62.82%
55.13%
57.69%
72.73%
53.85%
Decision Tree Accuracies:
100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
47.44%
K-Nearest Neighbors Accuracies:
65.38%
64.1%
62.82%
76.92%
66.67%
62.82%
64.1%
70.51%
68.83%
47.44%
Multi-Layer Perceptron Accuracies:
48.72%
47.44%
48.72%
48.72%
30.77%
47.44%
41.03%
42.31%
53.25%
42.31%
```

Menyimpan Model ke File Eksternal

In [19]:

```
from sklearn.externals import joblib

joblib.dump(nb_models, 'nb_models.joblib')
joblib.dump(dt_models, 'dt_models.joblib')
joblib.dump(knn_models, 'knn_models.joblib')
joblib.dump(mlp_models, 'mlp_models.joblib')
```

Out[19]:

```
['mlp_models.joblib']
```

Analisis Hasil Training

Berdasarkan accuracy dari hasil prediksi tiap model, model Decision Tree adalah yang terbaik karena memiliki hasil yang paling bagus.

Evaluasi model terbaik yang telah disimpan

In [23]:

```
model_dt = joblib.load('dt_models.joblib')
print(model_dt)
```

```
[DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth
=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth
=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth
=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth
=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth
=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth
=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth
=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth
=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth
=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')]
```

In [15]:

```
generate_accuracies(model_dt)
```

```
100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
47.44%
```

In [ ]:

```

```

In [ ]:

```

```