ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

Метод прогнозування серцево-судинних захворювань на основі методів машинного навчання

Кваліфікаційна робота
Другий (магістерський) рівень

Автор:
Вракіна К.П.
студ. гр. КСМм–22–2

Керівник:
Мартовицький В.О.
доц. кафедри ЕОМ

1

Мета і задачі роботи

Мета: дослідження і окресслення методів машинного навчання, які можуть бути ефективно застосовані для прогнозування серцевих захворювань та корекції поведінкових факторів ризику в профілактиці серцево-судинних захворювань (ССЗ).

Задачі:

- Вивчення медичного матеріалу та знайомство з відомими експериментами
- Вивчення технології і алгоритмів машинного навчання
- Розробка архітектури програми, її розробка та тестування

2

Машинне навчання та його призначення

Термін «машинне навчання» був введений у 1959 році Артуром Самуелем, співробітником IBM і піонером у галузі комп'ютерних ігор і штучного інтелекту.

Машинне навчання – один з методів функціонування штучного інтелекту, а саме – практичної реалізації його можливостей шляхом створення алгоритмів для виявлення закономірностей під час аналізу великих даних, та їх подальше використання для самонавчання.

Дві головні мети сучасного Машинного навчання:

- класифікація даних на основі розроблених моделей
- спрогнозувати майбутні результати на основі цих моделей.

3

Актуальність Роботи

Актуальність теми використання машинного навчання як основної технології прогнозування серцево-судинних захворювань на основі даних пацієнтів є величезною та багатогранною. Ця актуальність випливає з гострих глобальних проблем охорони здоров'я, пов'язаних із серцево-судинними захворюваннями, і трансформаційного потенціалу, який пропонує машинне навчання для вирішення цих проблем.

4

Фреймворки Машинного навчання

- **TensorFlow** і **PyTorch** - прямі конкурентами через свою схожість (обидва надають багатий набір інструментів лінійної алгебри та можуть виконувати регресійний аналіз).

PyTorch вважається більш пітонічним.**TensorFlow** може запустити модель швидше та з деякими налаштуваннями, а PyTorch вважається більш настроюваним, дотримуючись більш традиційного підходу до об'єктно-орієнтованого програмування через створення класів та має швидкий час навчання.

- **Scikit-learn** існує вже давно, але має велике застереження: він не створений для роботи в кластері.
- **Spark ML** створено для роботи в кластері, оскільки це суть Apache Spark, тобто він може обробляти дійсно велике множення матриці, беручи фрагменти матриці та запускаючи це обчислення на різних серверах
- **Scikit** — ще один пакет Python, який може виконувати багато корисних завдань машинного навчання: лінійна регресія, регресії дерева рішень, регресії випадкового лісу, K-nearest neighbour, SVM, стохастичні моделі градієнтного спуску.

5

Сфери використання і призначення Машинного навчання

- *Розпізнавання зображень* (позначення рентгенівських знімків як ракових чи ні, призначення імені сфотографованому обличчю)
- *Розпізнавання обличчя на зображенні*
- *Розпізнавання мовлення* (перетворення голосу в текстовий файл, наприклад, голосовий пошук)
- *Діагностика захворювань (*постановка діагнозу або рекомендація варіанту лікування, розпізнавання ракової тканини*)*
- *Екстракція* (створення моделі для прогнозування розладів голосових зв'язок, розробка методів профілактики, діагностики та лікування захворювань)

6

Методи і прийоми боротьби із Серцево-судинними захворюваннями (ССЗ)

Алгоритми класифікації

Системи постійного моніторингу та оповіщення

Регресійні моделі

Геномний аналіз

Глибоке навчання

Методи ансамблю

Обробка природної мови (nlp)

Обробка даних у реальному часі

Алгоритми кластеризації

Виявлення аномалії

Прогностичне моделювання прихильності до лікування

Аналіз часових рядів

Незбалансована обробка даних

Передача навчання

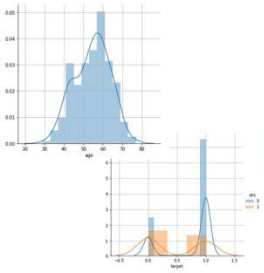
Зрозумілість моделі

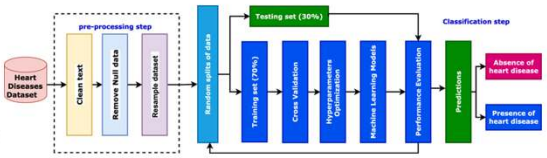
Зменшення розмірності

Міждисциплінарна співпраця

7

Розробка програми



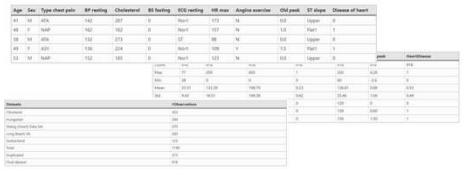


Основні етапи розробки програми.

У запропонованій схемі класифікації випадків захворювань серця, спочатку проводиться пошуковий аналіз.

Розподіл даних за віком та статтю.

Перед навчанням вибраних моделей важливо звернути увагу на відсутні значення холестерину.

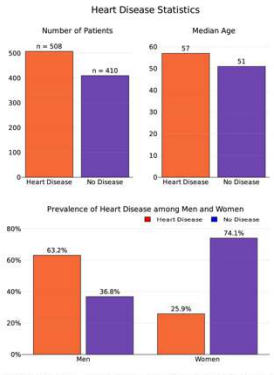


Набори даних і зведена статистика

Алгоритм машинного навчання буде правильно навчено після попередньої обробки та нормалізації наборів даних. Після модифікації даних їх довільно класифікують на навчальний набір і тестовий набір, при цьому 70% рядків призначають навчальному набору, а 30% — тестовому.

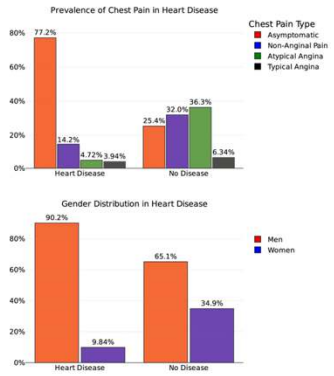
8

Результати



Статистика серцевих захворювань

Близько 63% чоловіків мають серцеві захворювання, тоді як приблизно 25% жінок мають серцеві захворювання. Жінка має ймовірність захворювання серця 25,91%. Чоловік має ймовірність захворювання серця 63,17%.



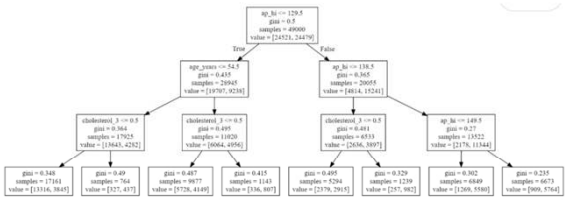
Безсимптомний біль у грудях при серцевих захворюваннях у майже 77%, відсутність болю в грудях (безсимптомний) є найпоширенішим симптомом у пацієнтів із серцевими захворюваннями.
Пацієнт із безсимптомним болем у грудях (ASY) приблизно в шість разів частіше страждає серцевими захворюваннями, ніж пацієнт із атиповою стенокардією (АТА).

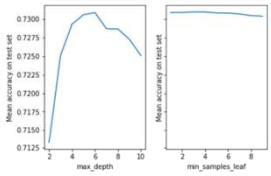
Поширеність безсимптомного болю в грудній клітині серед даних про захворювання серця

Цікаво, що існує негативний зв'язок між холестерином і хворобами серця. Чоловіки приблизно в 2,44 рази частіше страждають серцевими захворюваннями, ніж жінки. Існують чіткі відмінності між типами болю в грудях. Пацієнти з безсимптомним болем у грудях (ASY) приблизно в шість разів частіше страждають від серцевих захворювань, ніж у пацієнтів з атиповою стенокардією (АТА).

9

Приклад роботи програми

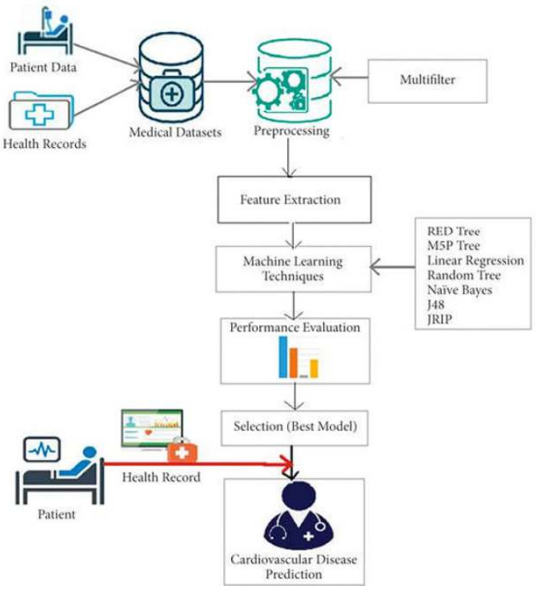




Спочатку відбувається контроль стану ЧСС пацієнта. Наступний етап використовується в системах підтримки медичних рішень для прогнозування та діагностики захворювань серця. Завдяки тренуванню техніки глибокого та машинного навчання, яка аналізує дані користувача для виявлення захворювань серця.

10

Схема взаємодії людини та Машинного навчання



Висновки

У цьому дослідженні було використано різні методи попередньої обробки та алгоритми машинного навчання для проведення поглибленого аналізу та отримання результатів. Було використано різноманітні класифікації навчання під наглядом, щоб класифікувати діагноз серцевих проблем.

Цю роботу можна продовжити для діагностики серцевих захворювань за допомогою зрозумілої техніки машинного навчання, щоб встановити точність, справедливість, прозорість і результати моделі. Крім того, використання самостійно створеного набору даних дозволить розширити це дослідження.

ДОДАТОК Б

Обробка програми

Б.1 Код обробки програми

```
df = pd.read_csv('../input/cardio_train.csv', sep=';')
df.head()

df['age_years'] = np.floor(df['age'] / 365.25)
new_df = pd.get_dummies(df, columns=['cholesterol', 'gluc'])
new_df.head()

from sklearn.model_selection import train_test_split
X = new_df.drop(['id', 'cardio', 'age'], axis=1)
y = new_df['cardio']
X_train, X_valid, y_train, y_valid = train_test_split(X,
y,test_size=0.3, random_state=2019)

from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(max_depth=3, random_state=2019)
tree.fit(X_train, y_train)

DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=3,
                       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=2019, splitter='best')


from sklearn.tree import export_graphviz

export_graphviz(tree, out_file='tree.dot',
feature_names=X.columns)
print(open('tree.dot').read())

digraph Tree {
node [shape=box] ;
0 [label="ap_hi <= 129.5\ngini = 0.5\nsamples = 49000\nvalue =
[24521, 24479]"] ;
1 [label="age_years <= 54.5\ngini = 0.435\nsamples =
28945\nvalue = [19707, 9238]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
```

```
2 [label="cholesterol_3 <= 0.5\ngini = 0.364\nsamples =
17925\nvalue = [13643, 4282]"] ;
1 -> 2 ;
3 [label="gini = 0.348\nsamples = 17161\nvalue = [13316, 3845]"]
;
2 -> 3 ;
4 [label="gini = 0.49\nsamples = 764\nvalue = [327, 437]"] ;
2 -> 4 ;
5 [label="cholesterol_3 <= 0.5\ngini = 0.495\nsamples =
11020\nvalue = [6064, 4956]"] ;
1 -> 5 ;
6 [label="gini = 0.487\nsamples = 9877\nvalue = [5728, 4149]"] ;
5 -> 6 ;
7 [label="gini = 0.415\nsamples = 1143\nvalue = [336, 807]"] ;
5 -> 7 ;
8 [label="ap_hi <= 138.5\ngini = 0.365\nsamples = 20055\nvalue =
[4814, 15241]"] ;
0 -> 8 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
9 [label="cholesterol_3 <= 0.5\ngini = 0.481\nsamples =
6533\nvalue = [2636, 3897]"] ;
8 -> 9 ;
10 [label="gini = 0.495\nsamples = 5294\nvalue = [2379, 2915]"]
;
9 -> 10 ;
11 [label="gini = 0.329\nsamples = 1239\nvalue = [257, 982]"] ;
9 -> 11 ;
12 [label="ap_hi <= 149.5\ngini = 0.27\nsamples = 13522\nvalue =
[2178, 11344]"] ;
8 -> 12 ;
13 [label="gini = 0.302\nsamples = 6849\nvalue = [1269, 5580]"]
;
12 -> 13 ;
14 [label="gini = 0.235\nsamples = 6673\nvalue = [909, 5764]"] ;
12 -> 14 ;
}
```

```
from sklearn.metrics import accuracy_score

y_pred = tree.predict(X_valid)
accuracy_score(y_valid, y_pred)
```

```
from sklearn.model_selection import GridSearchCV

tree_params = {'max_depth': np.arange(2, 11),
               'min_samples_leaf': np.arange(1, 10)
              }

tree_grid = GridSearchCV(tree, tree_params, cv=5,
scoring='accuracy') # Cross validation by 5 blocks
tree_grid.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=DecisionTreeClassifier(class_weight=None,
```

```
                                                criterion='gini',
max_depth=3,
                                                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=2019,
                                                splitter='best'),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': array([ 2,  3,  4,  5,  6,
7,  8,  9, 10]),
                         'min_samples_leaf': array([1, 2, 3, 4,
5, 6, 7, 8, 9])},
             pre_dispatch='2*n_jobs', refit=True,
return_train_score=False,
             scoring='accuracy', verbose=0)

tree_grid.best_params_

{'max_depth': 6, 'min_samples_leaf': 3}

tree_grid.best_estimator_

DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=6,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0,
min_impurity_split=None,
                       min_samples_leaf=3, min_samples_split=2,
                       min_weight_fraction_leaf=0.0,
presort=False,
                       random_state=2019, splitter='best')


tree_grid.best_score_

0.7309795918367347

from sklearn.model_selection import GridSearchCV

tree_params = {'max_depth': np.arange(2, 11)}

tree_grid = GridSearchCV(tree, tree_params, cv=5,
```

```
scoring='accuracy')
tree_grid.fit(X_train, y_train)

GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=DecisionTreeClassifier(class_weight=None,
                                              criterion='gini',
max_depth=3,
                                              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=2019,
                                              splitter='best'),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': array([ 2,  3,  4,  5,  6,
7,  8,  9, 10])},
             pre_dispatch='2*n_jobs', refit=True,
return_train_score=False,
             scoring='accuracy', verbose=0)

tree_grid.best_params_

{'max_depth': 6}

tree_params_2 = {'min_samples_leaf': np.arange(1, 10),
                 'max_depth': [6]}

tree_grid_2 = GridSearchCV(tree, tree_params_2, cv=5,
scoring='accuracy')
tree_grid_2.fit(X_train, y_train)

GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=DecisionTreeClassifier(class_weight=None,
                                              criterion='gini',
max_depth=3,
                                              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=2019,
                                                splitter='best'),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': [6],
                         'min_samples_leaf': array([1, 2, 3, 4,
5, 6, 7, 8, 9])},
             pre_dispatch='2*n_jobs', refit=True,
return_train_score=False,
             scoring='accuracy', verbose=0)

tree_grid_2.best_params_

{'max_depth': 6, 'min_samples_leaf': 3}

import matplotlib.pyplot as plt

fig, ax = plt.subplots(nrows=1, ncols=2, sharey=True)
ax[0].plot(tree_params['max_depth'],
tree_grid.cv_results_['mean_test_score']) # accuracy vs
max_depth
ax[0].set_xlabel('max_depth')
ax[0].set_ylabel('Mean accuracy on test set')

ax[1].plot(tree_params_2['min_samples_leaf'],
tree_grid_2.cv_results_['mean_test_score']) # accuracy vs
min_samples_leaf
ax[1].set_xlabel('min_samples_leaf')
ax[1].set_ylabel('Mean accuracy on test set')

pd.DataFrame(tree_grid.cv_results_).head().T

best_tree = tree_grid.best_estimator_
y_pred = best_tree.predict(X_valid)
accuracy_score(y_valid, y_pred)

0.7320952380952381

export_graphviz(best_tree, out_file='best_tree.dot',
feature_names=X.columns)
print(open('best_tree.dot').read())

digraph Tree {
node [shape=box] ;
0 [label="ap_hi <= 129.5\ngini = 0.5\nsamples = 49000\nvalue =
[24521, 24479]"] ;
1 [label="age_years <= 54.5\ngini = 0.435\nsamples =
28945\nvalue = [19707, 9238]"] ;
```

```
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="cholesterol_3 <= 0.5\ngini = 0.364\nsamples =
17925\nvalue = [13643, 4282]"] ;
1 -> 2 ;
3 [label="age_years <= 43.5\ngini = 0.348\nsamples =
17161\nvalue = [13316, 3845]"] ;
2 -> 3 ;
4 [label="cholesterol_2 <= 0.5\ngini = 0.239\nsamples =
4568\nvalue = [3935, 633]"] ;
3 -> 4 ;
5 [label="ap_hi <= 113.5\ngini = 0.214\nsamples = 4121\nvalue =
[3618, 503]"] ;
4 -> 5 ;
6 [label="gini = 0.154\nsamples = 1741\nvalue = [1595, 146]"] ;
5 -> 6 ;
7 [label="gini = 0.255\nsamples = 2380\nvalue = [2023, 357]"] ;
5 -> 7 ;
8 [label="weight <= 81.5\ngini = 0.412\nsamples = 447\nvalue =
[317, 130]"] ;
4 -> 8 ;
9 [label="gini = 0.374\nsamples = 338\nvalue = [254, 84]"] ;
8 -> 9 ;
10 [label="gini = 0.488\nsamples = 109\nvalue = [63, 46]"] ;
8 -> 10 ;
11 [label="ap_hi <= 119.5\ngini = 0.38\nsamples = 12593\nvalue =
[9381, 3212]"] ;
3 -> 11 ;
12 [label="weight <= 64.5\ngini = 0.323\nsamples = 4152\nvalue =
[3312, 840]"] ;
11 -> 12 ;
13 [label="gini = 0.249\nsamples = 1599\nvalue = [1366, 233]"] ;
12 -> 13 ;
14 [label="gini = 0.362\nsamples = 2553\nvalue = [1946, 607]"] ;
12 -> 14 ;
15 [label="cholesterol_1 <= 0.5\ngini = 0.404\nsamples =
8441\nvalue = [6069, 2372]"] ;
11 -> 15 ;
16 [label="gini = 0.476\nsamples = 795\nvalue = [485, 310]"] ;
15 -> 16 ;
17 [label="gini = 0.394\nsamples = 7646\nvalue = [5584, 2062]"]
;
15 -> 17 ;
18 [label="gluc_3 <= 0.5\ngini = 0.49\nsamples = 764\nvalue =
[327, 437]"] ;
2 -> 18 ;
19 [label="ap_lo <= 71.5\ngini = 0.368\nsamples = 325\nvalue =
[79, 246]"] ;
18 -> 19 ;
20 [label="height <= 152.5\ngini = 0.477\nsamples = 84\nvalue =
[33, 51]"] ;
19 -> 20 ;
21 [label="gini = 0.278\nsamples = 6\nvalue = [5, 1]"] ;
20 -> 21 ;
```

```
22 [label="gini = 0.46\nsamples = 78\nvalue = [28, 50]"] ;
20 -> 22 ;
23 [label="age_years <= 51.5\ngini = 0.309\nsamples = 241\nvalue
= [46, 195]"] ;
19 -> 23 ;
24 [label="gini = 0.259\nsamples = 170\nvalue = [26, 144]"] ;
23 -> 24 ;
25 [label="gini = 0.405\nsamples = 71\nvalue = [20, 51]"] ;
23 -> 25 ;
26 [label="weight <= 80.5\ngini = 0.492\nsamples = 439\nvalue =
[248, 191]"] ;
18 -> 26 ;
27 [label="ap_lo <= 74.5\ngini = 0.469\nsamples = 348\nvalue =
[217, 131]"] ;
26 -> 27 ;
28 [label="gini = 0.366\nsamples = 87\nvalue = [66, 21]"] ;
27 -> 28 ;
29 [label="gini = 0.488\nsamples = 261\nvalue = [151, 110]"] ;
27 -> 29 ;
30 [label="height <= 171.5\ngini = 0.449\nsamples = 91\nvalue =
[31, 60]"] ;
26 -> 30 ;
31 [label="gini = 0.313\nsamples = 67\nvalue = [13, 54]"] ;
30 -> 31 ;
32 [label="gini = 0.375\nsamples = 24\nvalue = [18, 6]"] ;
30 -> 32 ;
33 [label="cholesterol_3 <= 0.5\ngini = 0.495\nsamples =
11020\nvalue = [6064, 4956]"] ;
1 -> 33 ;
34 [label="age_years <= 60.5\ngini = 0.487\nsamples =
9877\nvalue = [5728, 4149]"] ;
33 -> 34 ;
35 [label="ap_hi <= 119.0\ngini = 0.472\nsamples = 7292\nvalue =
[4511, 2781]"] ;
34 -> 35 ;
36 [label="weight <= 91.5\ngini = 0.415\nsamples = 1854\nvalue =
[1310, 544]"] ;
35 -> 36 ;
37 [label="gini = 0.402\nsamples = 1755\nvalue = [1265, 490]"] ;
36 -> 37 ;
38 [label="gini = 0.496\nsamples = 99\nvalue = [45, 54]"] ;
36 -> 38 ;
39 [label="active <= 0.5\ngini = 0.484\nsamples = 5438\nvalue =
[3201, 2237]"] ;
35 -> 39 ;
40 [label="gini = 0.5\nsamples = 1094\nvalue = [564, 530]"] ;
39 -> 40 ;
41 [label="gini = 0.477\nsamples = 4344\nvalue = [2637, 1707]"]
;
39 -> 41 ;
42 [label="weight <= 52.5\ngini = 0.498\nsamples = 2585\nvalue =
[1217, 1368]"] ;
34 -> 42 ;
```

```
43 [label="ap_lo <= 79.5\ngini = 0.448\nsamples = 115\nvalue =
[76, 39]"] ;
42 -> 43 ;
44 [label="gini = 0.334\nsamples = 52\nvalue = [41, 11]"] ;
43 -> 44 ;
45 [label="gini = 0.494\nsamples = 63\nvalue = [35, 28]"] ;
43 -> 45 ;
46 [label="active <= 0.5\ngini = 0.497\nsamples = 2470\nvalue =
[1141, 1329]"] ;
42 -> 46 ;
47 [label="gini = 0.478\nsamples = 520\nvalue = [205, 315]"] ;
46 -> 47 ;
48 [label="gini = 0.499\nsamples = 1950\nvalue = [936, 1014]"] ;
46 -> 48 ;
49 [label="weight <= 70.5\ngini = 0.415\nsamples = 1143\nvalue =
[336, 807]"] ;
33 -> 49 ;
50 [label="age_years <= 60.5\ngini = 0.466\nsamples = 538\nvalue
= [199, 339]"] ;
49 -> 50 ;
51 [label="height <= 161.5\ngini = 0.489\nsamples = 366\nvalue =
[156, 210]"] ;
50 -> 51 ;
52 [label="gini = 0.458\nsamples = 163\nvalue = [58, 105]"] ;
51 -> 52 ;
53 [label="gini = 0.499\nsamples = 203\nvalue = [98, 105]"] ;
51 -> 53 ;
54 [label="age_years <= 63.5\ngini = 0.375\nsamples = 172\nvalue
= [43, 129]"] ;
50 -> 54 ;
55 [label="gini = 0.407\nsamples = 144\nvalue = [41, 103]"] ;
54 -> 55 ;
56 [label="gini = 0.133\nsamples = 28\nvalue = [2, 26]"] ;
54 -> 56 ;
57 [label="height <= 185.5\ngini = 0.35\nsamples = 605\nvalue =
[137, 468]"] ;
49 -> 57 ;
58 [label="age_years <= 60.5\ngini = 0.343\nsamples = 600\nvalue
= [132, 468]"] ;
57 -> 58 ;
59 [label="gini = 0.377\nsamples = 389\nvalue = [98, 291]"] ;
58 -> 59 ;
60 [label="gini = 0.27\nsamples = 211\nvalue = [34, 177]"] ;
58 -> 60 ;
61 [label="gini = 0.0\nsamples = 5\nvalue = [5, 0]"] ;
57 -> 61 ;
62 [label="ap_hi <= 138.5\ngini = 0.365\nsamples = 20055\nvalue
= [4814, 15241]"] ;
0 -> 62 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
63 [label="cholesterol_3 <= 0.5\ngini = 0.481\nsamples =
6533\nvalue = [2636, 3897]"] ;
62 -> 63 ;
64 [label="age_years <= 58.5\ngini = 0.495\nsamples =
```

```
5294\nvalue = [2379, 2915]"] ;
63 -> 64 ;
65 [label="ap_lo <= 88.5\ngini = 0.5\nsamples = 3797\nvalue =
[1839, 1958]"] ;
64 -> 65 ;
66 [label="age_years <= 41.5\ngini = 0.498\nsamples =
2245\nvalue = [1189, 1056]"] ;
65 -> 66 ;
67 [label="gini = 0.439\nsamples = 157\nvalue = [106, 51]"] ;
66 -> 67 ;
68 [label="gini = 0.499\nsamples = 2088\nvalue = [1083, 1005]"]
;
66 -> 68 ;
69 [label="smoke <= 0.5\ngini = 0.487\nsamples = 1552\nvalue =
[650, 902]"] ;
65 -> 69 ;
70 [label="gini = 0.482\nsamples = 1391\nvalue = [562, 829]"] ;
69 -> 70 ;
71 [label="gini = 0.496\nsamples = 161\nvalue = [88, 73]"] ;
69 -> 71 ;
72 [label="age_years <= 61.5\ngini = 0.461\nsamples =
1497\nvalue = [540, 957]"] ;
64 -> 72 ;
73 [label="smoke <= 0.5\ngini = 0.482\nsamples = 849\nvalue =
[343, 506]"] ;
72 -> 73 ;
74 [label="gini = 0.475\nsamples = 779\nvalue = [303, 476]"] ;
73 -> 74 ;
75 [label="gini = 0.49\nsamples = 70\nvalue = [40, 30]"] ;
73 -> 75 ;
76 [label="weight <= 55.5\ngini = 0.423\nsamples = 648\nvalue =
[197, 451]"] ;
72 -> 76 ;
77 [label="gini = 0.499\nsamples = 25\nvalue = [13, 12]"] ;
76 -> 77 ;
78 [label="gini = 0.416\nsamples = 623\nvalue = [184, 439]"] ;
76 -> 78 ;
79 [label="gluc_3 <= 0.5\ngini = 0.329\nsamples = 1239\nvalue =
[257, 982]"] ;
63 -> 79 ;
80 [label="weight <= 84.5\ngini = 0.287\nsamples = 748\nvalue =
[130, 618]"] ;
79 -> 80 ;
81 [label="weight <= 80.5\ngini = 0.314\nsamples = 497\nvalue =
[97, 400]"] ;
80 -> 81 ;
82 [label="gini = 0.285\nsamples = 436\nvalue = [75, 361]"] ;
81 -> 82 ;
83 [label="gini = 0.461\nsamples = 61\nvalue = [22, 39]"] ;
81 -> 83 ;
84 [label="age_years <= 50.5\ngini = 0.228\nsamples = 251\nvalue
= [33, 218]"] ;
80 -> 84 ;
```

```
85 [label="gini = 0.098\nsamples = 58\nvalue = [3, 55]"] ;
84 -> 85 ;
86 [label="gini = 0.263\nsamples = 193\nvalue = [30, 163]"] ;
84 -> 86 ;
87 [label="height <= 156.5\ngini = 0.384\nsamples = 491\nvalue =
[127, 364]"] ;
79 -> 87 ;
88 [label="age_years <= 40.5\ngini = 0.247\nsamples = 104\nvalue
= [15, 89]"] ;
87 -> 88 ;
89 [label="gini = 0.0\nsamples = 1\nvalue = [1, 0]"] ;
88 -> 89 ;
90 [label="gini = 0.235\nsamples = 103\nvalue = [14, 89]"] ;
88 -> 90 ;
91 [label="weight <= 83.5\ngini = 0.411\nsamples = 387\nvalue =
[112, 275]"] ;
87 -> 91 ;
92 [label="gini = 0.451\nsamples = 239\nvalue = [82, 157]"] ;
91 -> 92 ;
93 [label="gini = 0.323\nsamples = 148\nvalue = [30, 118]"] ;
91 -> 93 ;
94 [label="ap_hi <= 149.5\ngini = 0.27\nsamples = 13522\nvalue =
[2178, 11344]"] ;
62 -> 94 ;
95 [label="gluc_3 <= 0.5\ngini = 0.302\nsamples = 6849\nvalue =
[1269, 5580]"] ;
94 -> 95 ;
96 [label="age_years <= 60.5\ngini = 0.294\nsamples =
6133\nvalue = [1099, 5034]"] ;
95 -> 96 ;
97 [label="cholesterol_1 <= 0.5\ngini = 0.303\nsamples =
5034\nvalue = [938, 4096]"] ;
96 -> 97 ;
98 [label="gini = 0.278\nsamples = 1622\nvalue = [270, 1352]"] ;
97 -> 98 ;
99 [label="gini = 0.315\nsamples = 3412\nvalue = [668, 2744]"] ;
97 -> 99 ;
100 [label="ap_lo <= 62.5\ngini = 0.25\nsamples = 1099\nvalue =
[161, 938]"] ;
96 -> 100 ;
101 [label="gini = 0.497\nsamples = 13\nvalue = [6, 7]"] ;
100 -> 101 ;
102 [label="gini = 0.245\nsamples = 1086\nvalue = [155, 931]"] ;
100 -> 102 ;
103 [label="age_years <= 62.5\ngini = 0.362\nsamples =
716\nvalue = [170, 546]"] ;
95 -> 103 ;
104 [label="weight <= 90.5\ngini = 0.377\nsamples = 643\nvalue =
[162, 481]"] ;
103 -> 104 ;
105 [label="gini = 0.353\nsamples = 489\nvalue = [112, 377]"] ;
104 -> 105 ;
106 [label="gini = 0.439\nsamples = 154\nvalue = [50, 104]"] ;
```

```
104 -> 106 ;
107 [label="alco <= 0.5\ngini = 0.195\nsamples = 73\nvalue = [8,
65]"] ;
103 -> 107 ;
108 [label="gini = 0.176\nsamples = 72\nvalue = [7, 65]"] ;
107 -> 108 ;
109 [label="gini = 0.0\nsamples = 1\nvalue = [1, 0]"] ;
107 -> 109 ;
110 [label="ap_lo <= 68.0\ngini = 0.235\nsamples = 6673\nvalue =
[909, 5764]"] ;
94 -> 110 ;
111 [label="ap_hi <= 249.5\ngini = 0.483\nsamples = 66\nvalue =
[27, 39]"] ;
110 -> 111 ;
112 [label="weight <= 58.5\ngini = 0.436\nsamples = 56\nvalue =
[18, 38]"] ;
111 -> 112 ;
113 [label="gini = 0.0\nsamples = 8\nvalue = [0, 8]"] ;
112 -> 113 ;
114 [label="gini = 0.469\nsamples = 48\nvalue = [18, 30]"] ;
112 -> 114 ;
115 [label="weight <= 80.5\ngini = 0.18\nsamples = 10\nvalue =
[9, 1]"] ;
111 -> 115 ;
116 [label="gini = 0.0\nsamples = 9\nvalue = [9, 0]"] ;
115 -> 116 ;
117 [label="gini = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
115 -> 117 ;
118 [label="gluc_3 <= 0.5\ngini = 0.231\nsamples = 6607\nvalue =
[882, 5725]"] ;
110 -> 118 ;
119 [label="ap_hi <= 12255.0\ngini = 0.225\nsamples =
6014\nvalue = [778, 5236]"] ;
118 -> 119 ;
120 [label="gini = 0.225\nsamples = 6011\nvalue = [776, 5235]"]
;
119 -> 120 ;
121 [label="gini = 0.444\nsamples = 3\nvalue = [2, 1]"] ;
119 -> 121 ;
122 [label="height <= 166.5\ngini = 0.289\nsamples = 593\nvalue
= [104, 489]"] ;
118 -> 122 ;
123 [label="gini = 0.242\nsamples = 390\nvalue = [55, 335]"] ;
122 -> 123 ;
124 [label="gini = 0.366\nsamples = 203\nvalue = [49, 154]"] ;
122 -> 124 ;
}
```