In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams["figure.figsize"] = (10, 20)
```

In [7]:

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

In [8]:

```
df.head()
```

Out[8]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	ca
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
id             0
age            0
gender         0
height         0
weight         0
ap_hi          0
ap_lo          0
cholesterol    0
gluc           0
smoke          0
alco           0
active         0
cardio         0
dtype: int64
```

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 13 columns):
id              70000 non-null int64
age             70000 non-null int64
gender          70000 non-null int64
height          70000 non-null int64
weight          70000 non-null float64
ap_hi           70000 non-null int64
ap_lo           70000 non-null int64
cholesterol     70000 non-null int64
gluc            70000 non-null int64
smoke           70000 non-null int64
alco            70000 non-null int64
active          70000 non-null int64
cardio          70000 non-null int64
dtypes: float64(1), int64(12)
memory usage: 6.9 MB
```

In [11]:

```
df.shape
```

Out[11]:

```
(70000, 13)
```

In [12]:

```
df['gender'].value_counts()
```

Out[12]:

```
1    45530
2    24470
Name: gender, dtype: int64
```

In [13]:

```
df1 = pd.get_dummies(data = df['gender'], drop_first = True, prefix = 'gender')
```

In [15]:

```
data = pd.concat([df, df1], axis = 1)
```

In [16]:

```
data.head()
```

Out[16]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	ca
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	

In [17]:

```
data = data.drop(['id'], axis = 1)
```

In [19]:

```
data = data.drop(['gender'], axis = 1)
```

In [20]:

```
data.head()
```

Out[20]:

	age	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	gender
0	18393	168	62.0	110	80	1	1	0	0	1	0	
1	20228	156	85.0	140	90	3	1	0	0	1	1	
2	18857	165	64.0	130	70	3	1	0	0	0	1	
3	17623	169	82.0	150	100	1	1	0	0	1	1	
4	17474	156	56.0	100	60	1	1	0	0	0	0	

In [21]:

```
X = data.drop(['cardio'], axis = 1)
y = data['cardio']
```

In [22]:

```
from sklearn.model_selection import train_test_split
```

In [23]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [24]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [25]:

```
clf = RandomForestClassifier(n_estimators = 10)
```

In [26]:

```
clf.fit(X_train, y_train)
```

Out[26]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=N
one,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [27]:

```
clf.score(X_train, y_train)
```

Out[27]:

```
0.9790178571428572
```

In [29]:

```
predict = clf.predict(X_test)
```

In [30]:

```
clf.score(X_test, y_test)
```

Out[30]:

```
0.6958571428571428
```

In [31]:

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

In [32]:

```
print(classification_report(predict, y_test))
```

```
              precision    recall  f1-score   support

           0       0.75      0.68      0.71      7743
           1       0.64      0.71      0.68      6257

    accuracy                           0.70     14000
   macro avg       0.70      0.70      0.69     14000
weighted avg       0.70      0.70      0.70     14000
```

In [33]:

```
print(accuracy_score(predict, y_test))
```

```
0.6958571428571428
```

In [35]:

```
sns.heatmap(data = 'data', hue = 'cardio')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-35-104b5855c490> in <module>
----> 1 sns.heatmap(data = 'data', hue = 'cardio')

~\Anaconda3\lib\site-packages\seaborn\matrix.py in heatmap(data, vmin, vmax,
cmap, center, robust, annot, fmt, annot_kws, linewidths, linecolor, cbar, cb
ar_kws, cbar_ax, square, xticklabels, yticklabels, mask, ax, **kwargs)
    515     plotter = _HeatMapper(data, vmin, vmax, cmap, center, robust, an
not, fmt,
    516                           annot_kws, cbar, cbar_kws, xticklabels,
--> 517                           yticklabels, mask)
    518
    519     # Add the pcolormesh kwargs here

~\Anaconda3\lib\site-packages\seaborn\matrix.py in __init__(self, data, vmi
n, vmax, cmap, center, robust, annot, fmt, annot_kws, cbar, cbar_kws, xtickl
abels, yticklabels, mask)
    108         else:
    109             plot_data = np.asarray(data)
--> 110             data = pd.DataFrame(plot_data)
    111
    112         # Validate the mask and convet to DataFrame

~\Anaconda3\lib\site-packages\pandas\core\frame.py in __init__(self, data, i
ndex, columns, dtype, copy)
    438                 mgr = init_dict({data.name: data}, index, columns, d
type=dtype)
    439             else:
--> 440                 mgr = init_ndarray(data, index, columns, dtype=dtype
, copy=copy)
    441
    442         # For data is list-like, or Iterable (will consume into lis
t)

~\Anaconda3\lib\site-packages\pandas\core\internals\construction.py in init_
ndarray(values, index, columns, dtype, copy)
    169     # by definition an array here
    170     # the dtypes will be coerced to a single dtype
--> 171     values = prep_ndarray(values, copy=copy)
    172
    173     if dtype is not None:

~\Anaconda3\lib\site-packages\pandas\core\internals\construction.py in prep_
ndarray(values, copy)
    293         values = values.reshape((values.shape[0], 1))
    294     elif values.ndim != 2:
--> 295         raise ValueError("Must pass 2-d input")
    296
    297     return values

ValueError: Must pass 2-d input
```

In [36]:

```
from sklearn.linear_model import LogisticRegression
```

In [37]:

```
model = LogisticRegression()
```

In [38]:

```
model.fit(X_train, y_train)
```

```
C:\Users\SURYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Speci
fy a solver to silence this warning.
  FutureWarning)
```

Out[38]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [39]:

```
model.score(X_train, y_train)
```

Out[39]:

```
0.711125
```

In [40]:

```
pre = model.predict(X_test)
```

In [42]:

```
model.score(X_test, y_test)
```

Out[42]:

```
0.7082142857142857
```

In [43]:

```
print(classification_report(pre, y_test))
```

```
              precision    recall  f1-score   support

           0       0.75      0.70      0.72      7650
           1       0.66      0.72      0.69      6350

    accuracy                           0.71     14000
   macro avg       0.71      0.71      0.71     14000
weighted avg       0.71      0.71      0.71     14000
```

In [44]:

```
print(accuracy_score(pre, y_test))
```

```
0.7082142857142857
```

In [ ]:

```

```