cardio-vascular-disease-prediction

February 7, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
```

[2]:
```
data = pd.read_csv(r'/Users/akurisivanagendrareddy/Downloads/cardio_train2.csv')
data
```

[2]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	\
0	0	18393	2	168	62.0	110	80	1	1	
1	1	20228	1	156	85.0	140	90	3	1	
2	2	18857	1	165	64.0	130	70	3	1	
3	3	17623	2	169	82.0	150	100	1	1	
4	4	17474	1	156	56.0	100	60	1	1	
…	…	…	…	…	…	…	…	…	…	
69995	99993	19240	2	168	76.0	120	80	1	1	
69996	99995	22601	1	158	126.0	140	90	2	2	
69997	99996	19066	2	183	105.0	180	90	3	1	
69998	99998	22431	1	163	72.0	135	80	1	2	
69999	99999	20540	1	170	72.0	120	80	2	1	

	smoke	alco	active	cardio
0	0	0	1	0
1	0	0	1	1
2	0	0	0	1
3	0	0	1	1
4	0	0	0	0
…	…	…	…	…

```
69995      1     0       1       0
69996      0     0       1       1
69997      0     1       0       1
69998      0     0       0       1
69999      0     0       1       0

[70000 rows x 13 columns]
```

[3]:
```
scaler = StandardScaler()
numeric_features = ['age', 'ap_hi', 'cholesterol', 'gluc', 'weight']
data[numeric_features] = scaler.fit_transform(data[numeric_features])
```

[4]:
```
X = data.drop('ap_hi', axis=1)
y = data['ap_hi']
```

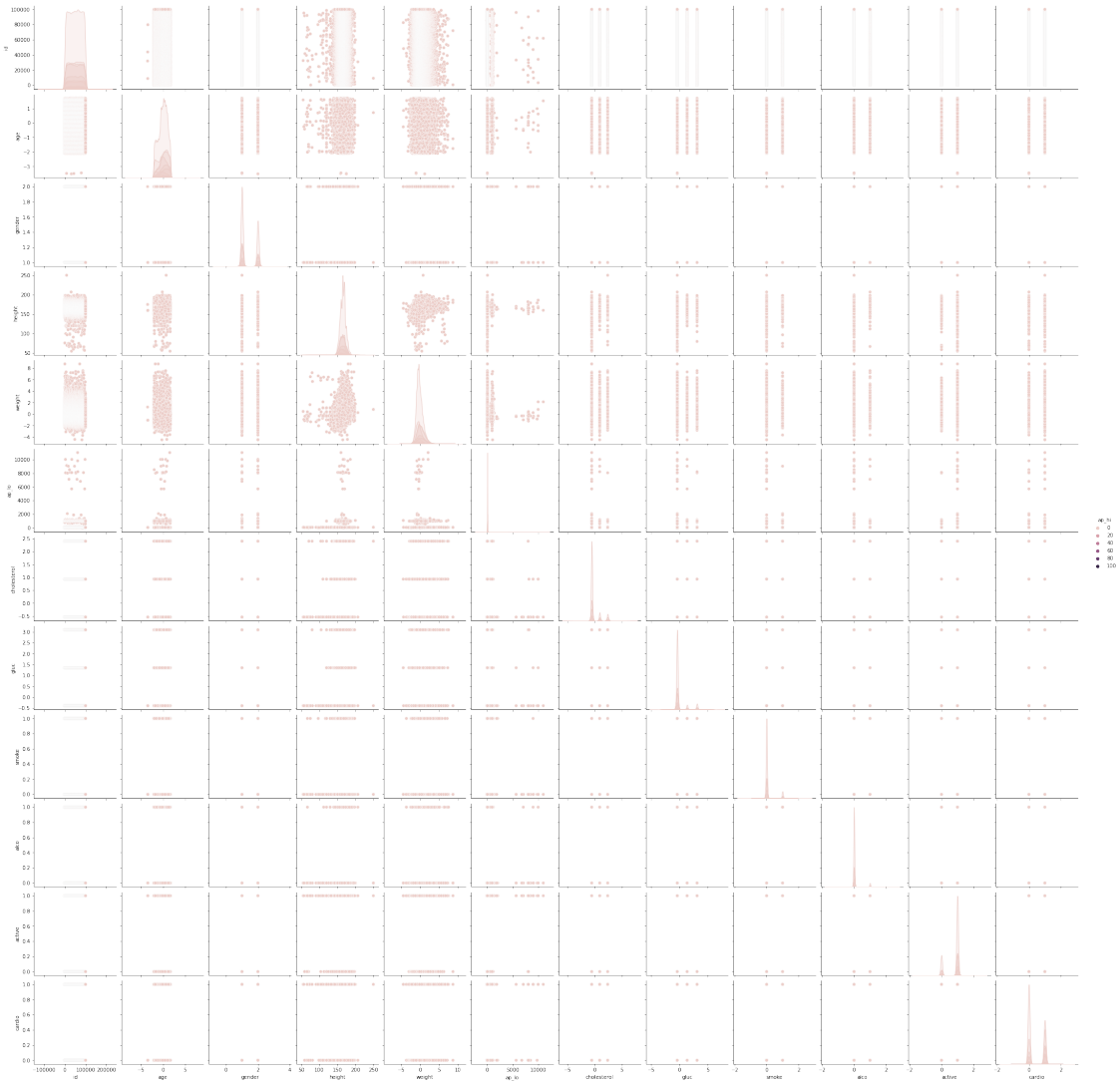
[5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[6]:
```
import seaborn as sns
import matplotlib.pyplot as plt
print(data.describe())
```

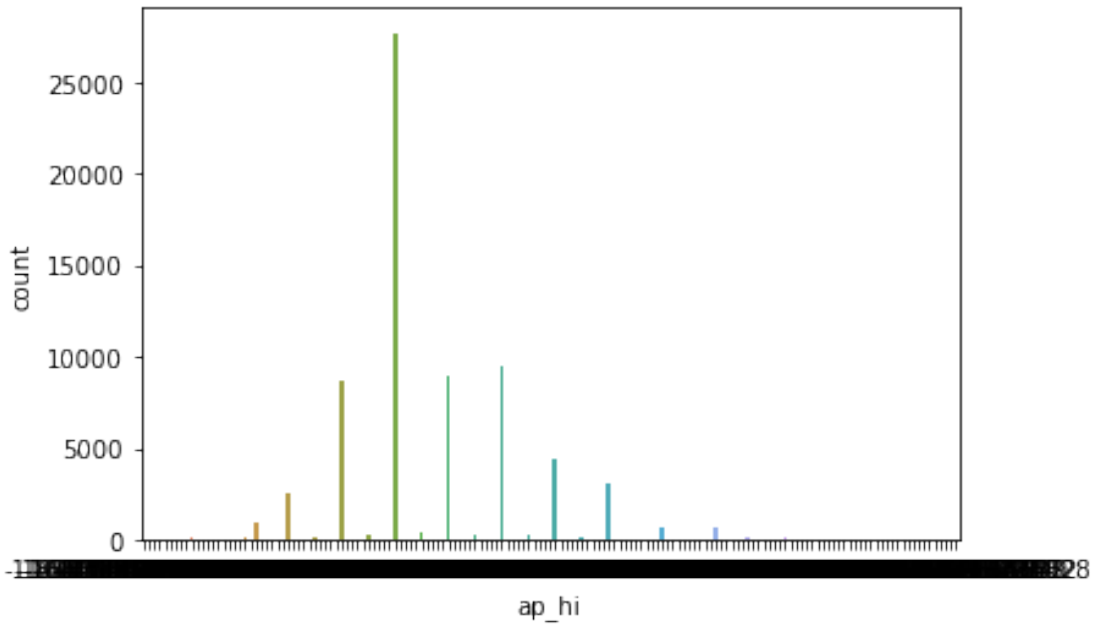
```
                 id           age        gender        height        weight  \
count  70000.000000  7.000000e+04  70000.000000  70000.000000  7.000000e+04
mean   49972.419900  5.202481e-16      1.349571    164.359229 -3.450534e-16
std    28851.302323  1.000007e+00      0.476838      8.210126  1.000007e+00
min        0.000000 -3.514407e+00      1.000000     55.000000 -4.460075e+00
25%    25006.750000 -7.315341e-01      1.000000    159.000000 -6.394770e-01
50%    50001.500000  9.489744e-02      1.000000    165.000000 -1.532192e-01
75%    74889.250000  7.531244e-01      2.000000    170.000000  5.414349e-01
max    99999.000000  1.720199e+00      2.000000    250.000000  8.738353e+00

              ap_hi         ap_lo   cholesterol          gluc         smoke  \
count  7.000000e+04  70000.000000  7.000000e+04  7.000000e+04  70000.000000
mean  -1.518658e-15     96.630414 -7.881616e-16  1.843571e-15      0.088129
std    1.000007e+00    188.472530  1.000007e+00  1.000007e+00      0.283484
min   -1.810381e+00    -70.000000 -5.393221e-01 -3.957199e-01      0.000000
25%   -5.725127e-02     80.000000 -5.393221e-01 -3.957199e-01      0.000000
50%   -5.725127e-02     80.000000 -5.393221e-01 -3.957199e-01      0.000000
75%    7.261016e-02     90.000000  9.307354e-01 -3.957199e-01      0.000000
max    1.031826e+02  11000.000000  2.400793e+00  3.099157e+00      1.000000

               alco        active        cardio
count  70000.000000  70000.000000  70000.000000
mean       0.053771      0.803729      0.499700
std        0.225568      0.397179      0.500003
min        0.000000      0.000000      0.000000
```

25%	0.000000	1.000000	0.000000
50%	0.000000	1.000000	0.000000
75%	0.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000

[7]:
```
sns.pairplot(data, hue='ap_hi', diag_kind='kde')
plt.show()
```

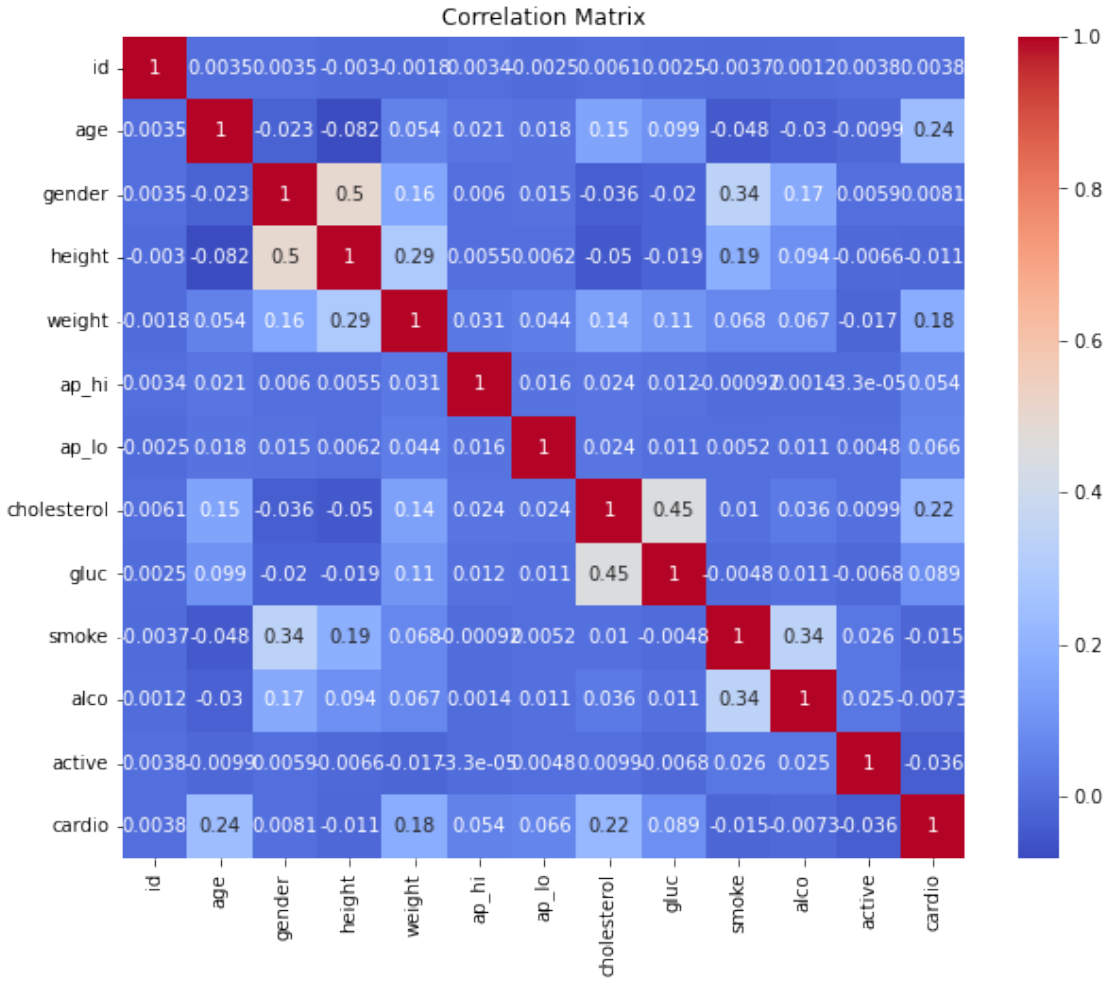


[8]:
```
sns.countplot(x='ap_hi', data=data)
plt.show()
```



[9]:
```
corr_matrix = data.corr()
```

[10]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



[11]:
```
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

[12]:
```
models = {
    'SVM': SVC(),
    'KNN': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier()
}
y_train = (y_train > 0).astype(int)
y_test = (y_test > 0).astype(int)
```

[13]:
```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f'{name} Accuracy: {accuracy}')
    print(classification_report(y_test, y_pred))
```

```
SVM Accuracy: 0.5881428571428572
              precision    recall  f1-score   support

           0       0.59      1.00      0.74      8234
           1       0.50      0.00      0.00      5766

    accuracy                           0.59     14000
   macro avg       0.54      0.50      0.37     14000
weighted avg       0.55      0.59      0.44     14000

KNN Accuracy: 0.718
              precision    recall  f1-score   support

           0       0.72      0.86      0.78      8234
           1       0.72      0.51      0.60      5766

    accuracy                           0.72     14000
   macro avg       0.72      0.69      0.69     14000
weighted avg       0.72      0.72      0.71     14000

Decision Tree Accuracy: 0.7655714285714286
              precision    recall  f1-score   support

           0       0.80      0.80      0.80      8234
           1       0.71      0.72      0.72      5766

    accuracy                           0.77     14000
   macro avg       0.76      0.76      0.76     14000
weighted avg       0.77      0.77      0.77     14000


/Users/akurisivanagendrareddy/opt/anaconda3/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
```

```
  n_iter_i = _check_optimize_result(
```

```
Logistic Regression Accuracy: 0.7539285714285714
              precision    recall  f1-score   support

           0       0.75      0.87      0.81      8234
           1       0.76      0.58      0.66      5766

    accuracy                           0.75     14000
   macro avg       0.76      0.73      0.73     14000
weighted avg       0.76      0.75      0.75     14000

Random Forest Accuracy: 0.8332857142857143
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      8234
           1       0.85      0.73      0.78      5766

    accuracy                           0.83     14000
   macro avg       0.84      0.82      0.82     14000
weighted avg       0.83      0.83      0.83     14000
```

[17]:
```
import joblib
joblib.dump(best_model, 'heart_disease_detection_model.pkl')
```

[17]:
```
['heart_disease_detection_model.pkl']
```

[ ]:

[ ]: