REPORT ON MINOR PROJECT ARTIFICIAL INTELLIGENCE

Cardiovascular disease prediction

JUNE 25, 2024

PARTHIV KUMAR NIKKU
BTech – Tifac core in cyber security at Amrita Vishwa Vidyapeetham

Cardiovascular Disease prediction

To achieve the goal of building a predictive and efficient model on cardiovascular disease prediction I followed these steps.

Data Pre-processing:

- Load the dataset and inspect it.
- Handle missing values if any.
- Perform data cleaning and format the data types.
- Normalize or standardize the data if necessary.
- Encode categorical variables.

Data Analysis and Visualization:

- Perform exploratory data analysis (EDA).
- Create various plots to understand the distribution of data and relationships between variables (e.g., histograms, box plots, scatter plots, etc.).
- Draw a correlation matrix to understand the relationships between different features.

Correlation Matrix:

- Compute and visualize the correlation matrix of the features to understand their relationships.

Model Building and Evaluation:

- Split the dataset into training and testing sets.
- Train various machine learning models including SVM, KNN, Decision Trees, Logistic Regression, and Random Forest.
- Evaluate the models using appropriate metrics (accuracy, precision, recall, F1-score, etc.).
- Select the best performing model for predicting heart disease.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Load the dataset
data = pd.read_csv('cardio_train.csv', sep=';')

# Display basic information about the dataset
print(data.info())

# Check for missing values
print(data.isnull().sum())

# Drop any rows with missing values (if any)
data.dropna(inplace=True)

# Encode categorical variables if any (assuming 'cardio' is the target variable)
le = LabelEncoder()
data['cardio'] = le.fit_transform(data['cardio'])

# Split data into features and target variable
X = data.drop(columns=['cardio'])
y = data['cardio']

# Normalize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

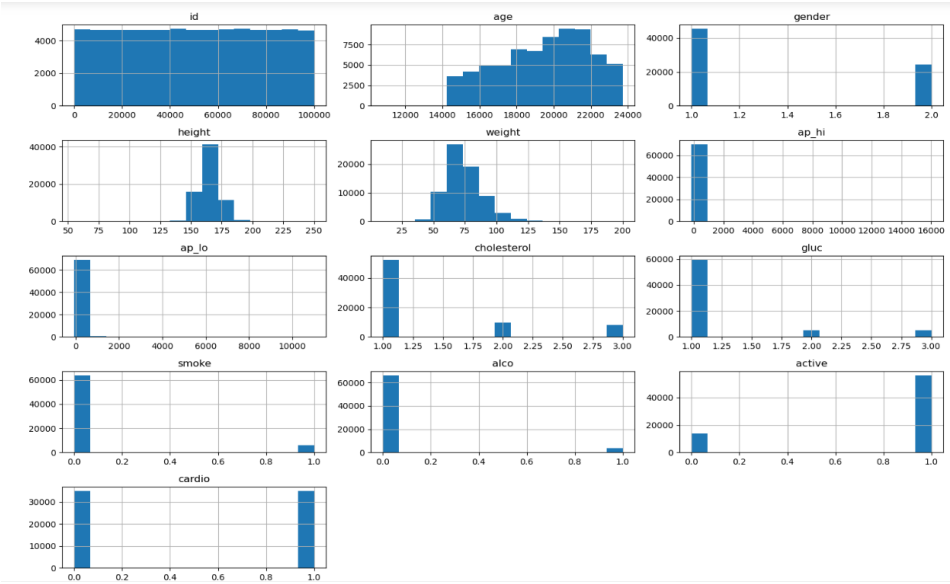
Output

```
class 'pandas.core.frame.DataFrame'
RangeIndex: 70000 entries, 0 to 69999


Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
  0     id           70000 non-null  int64
  1     age          70000 non-null  int64
  2     gender       70000 non-null  int64
  3     height       70000 non-null  int64
  4     weight       70000 non-null  float64
  5     ap_hi        70000 non-null  int64
  6     ap_lo        70000 non-null  int64
  7     cholesterol  70000 non-null  int64
  8     gluc         70000 non-null  int64
  9     smoke        70000 non-null  int64
  10    alco         70000 non-null  int64
  11    active       70000 non-null  int64
  12    cardio         70000 non-null   int64   dtypes: float64(1), int64(12)
 memory usage: 6.9 MB
None id
0 age
0 gender
0 height
0 weight
0 ap_hi
0 ap_lo
0 cholesterol
0 gluc
0 smoke
0 alco
```

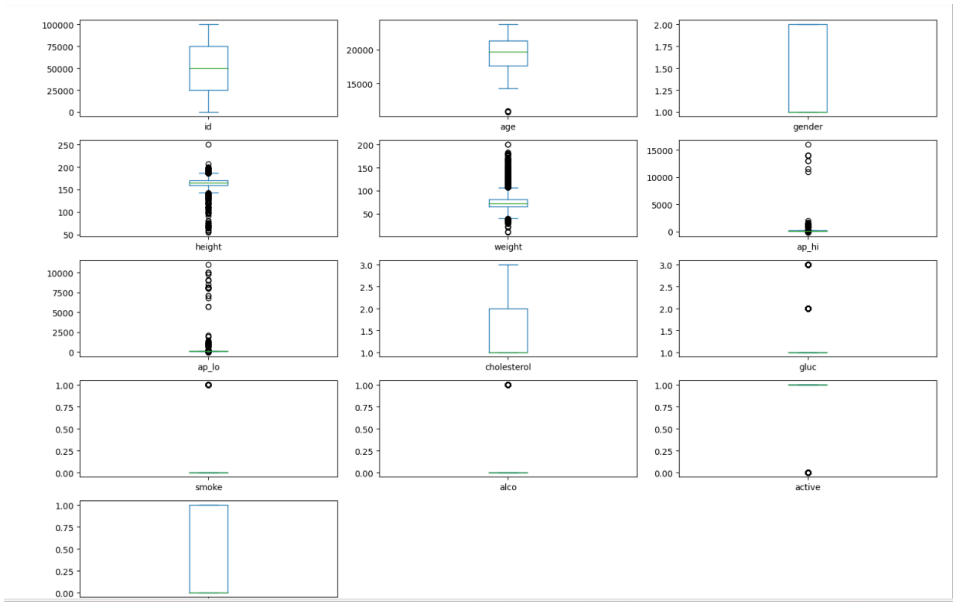
```
0 active
0 cardio
0 dtype: int64
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings


# Histogram for each feature
data.hist(bins=15, figsize=(15, 10), layout=(5, 3))
plt.tight_layout()
plt.show()

# Box plots for each feature
data.plot(kind='box', subplots=True, layout=(5, 3), figsize=(15, 10), sharex=False, sharey=False)
plt.tight_layout()
plt.show()

# Pair plot to see relationships between features
sns.pairplot(data)
plt.show()

# Heatmap of the correlation matrix
corr_matrix = data.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.show()
```

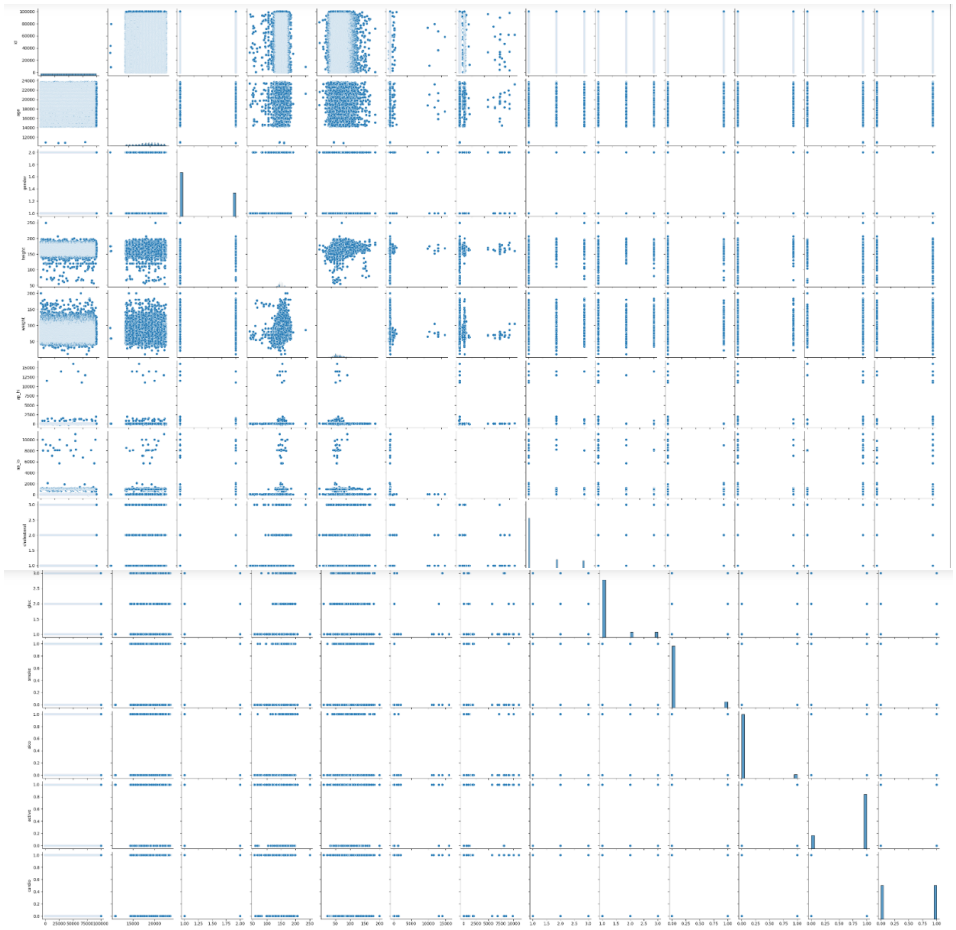
Outputs for each :



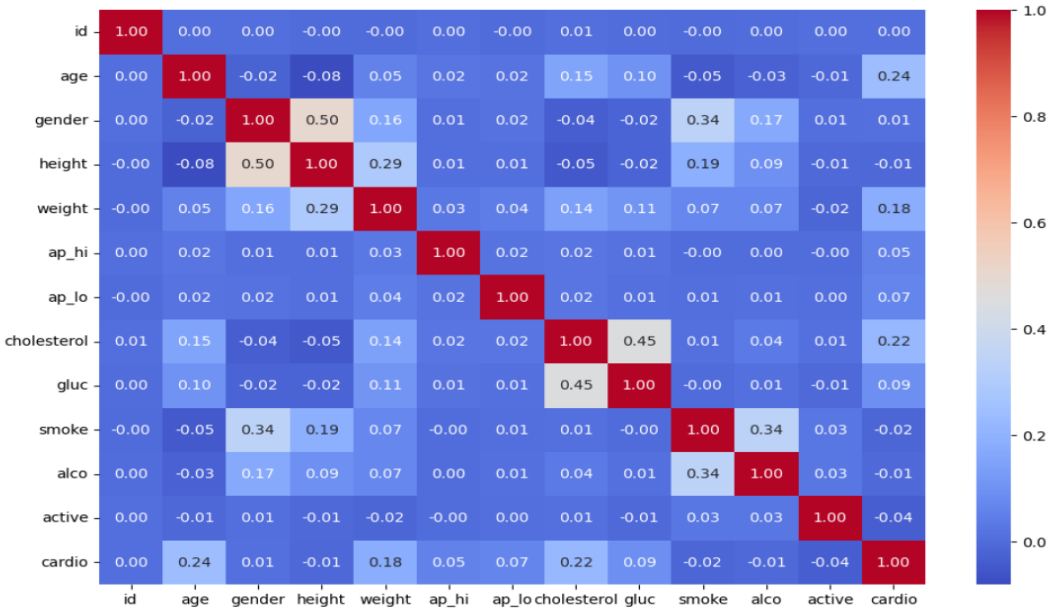
The above graph shows the histogram for each feature



The above graph shows the box diagram for each feature



Ths above graphs are the pair plots to observe the relationships bw the features



This is the heatmap for the data set

```
# Display the correlation matrix
print(corr_matrix)
```

```
                 id       age    gender    height    weight     ap_hi  \


 id           1.000000  0.003457  0.003502 -0.003038 -0.001830  0.003356
age           0.003457  1.000000 -0.022811 -0.081515  0.053684  0.020764
gender        0.003502 -0.022811  1.000000  0.499033  0.155406  0.006005
height       -0.003038 -0.081515  0.499033  1.000000  0.290968  0.005488
weight       -0.001830  0.053684  0.155406  0.290968  1.000000  0.030702
ap_hi         0.003356  0.020764  0.006005  0.005488  0.030702  1.000000
ap_lo        -0.002529  0.017647  0.015254  0.006150  0.043710  0.016086
cholesterol   0.006106  0.154424 -0.035821 -0.050226  0.141768  0.023778
gluc          0.002467  0.098703 -0.020491 -0.018595  0.106857  0.011841
smoke        -0.003699 -0.047633  0.338135  0.187989  0.067780 -0.000922
alco          0.001210 -0.029723  0.170966  0.094419  0.067113  0.001408
active        0.003755 -0.009927  0.005866 -0.006570 -0.016867 -0.000033
cardio       0.003799  0.238159  0.008109 -0.010821  0.181660  0.054475

                ap_lo  cholesterol      gluc     smoke      alco    active  \
id           -0.002529     0.006106  0.002467 -0.003699  0.001210  0.003755
age           0.017647     0.154424  0.098703 -0.047633 -0.029723 -0.009927
gender        0.015254    -0.035821 -0.020491  0.338135  0.170966  0.005866
height        0.006150    -0.050226 -0.018595  0.187989  0.094419 -0.006570
weight        0.043710     0.141768  0.106857  0.067780  0.067113 -0.016867
ap_hi         0.016086     0.023778  0.011841 -0.000922  0.001408 -0.000033
ap_lo         1.000000     0.024019  0.010806  0.005186  0.010601  0.004780
cholesterol   0.024019     1.000000  0.451578  0.010354  0.035760  0.009911
gluc          0.010806     0.451578  1.000000 -0.004756  0.011246 -0.006770
smoke         0.005186     0.010354 -0.004756  1.000000  0.340094  0.025858
alco          0.010601     0.035760  0.011246  0.340094  1.000000  0.025476
active        0.004780     0.009911 -0.006770  0.025858  0.025476  1.000000
cardio       0.065719     0.221147  0.089307 -0.015486 -0.007330 -0.035653

                cardio
id           0.003799
age          0.238159
gender       0.008109
height      -0.010821
weight       0.181660
ap_hi        0.054475
ap_lo        0.065719
cholesterol  0.221147
gluc         0.089307
smoke       -0.015486
alco        -0.007330
active      -0.035653
cardio       1.000000
```

```python
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

# Define the models
models = {
    'Support Vector Machine': SVC(),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier()
}

# Train and evaluate each model
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:.4f}')
    print(classification_report(y_test, y_pred))
    print('-----------------------------------------------')
```

```
Support    Vector    Machine    Accuracy:    0.7282
 precision    recall  f1-score   support

               0       0.71      0.76      0.74
   10461           1       0.74      0.70      0.72
                        10539

    accuracy                           0.73     21000
macro avg        0.73      0.73      0.73     21000
weighted avg       0.73      0.73      0.73     21000

---------------------------------------------
K-Nearest       Neighbors       Accuracy:       0.6248
 precision    recall  f1-score   support

               0       0.62      0.64      0.63
   10461           1       0.63      0.61      0.62
                        10539

    accuracy                           0.62     21000
macro avg        0.62      0.62      0.62     21000
weighted avg       0.62      0.62      0.62     21000

---------------------------------------------
Decision Tree Accuracy: 0.6350             precision
 recall  f1-score   support

               0       0.63      0.64      0.63
   10461           1       0.64      0.63      0.64
                        10539

    accuracy                           0.63     21000
macro avg        0.63      0.63      0.63     21000
weighted avg       0.63      0.63      0.63     21000

---------------------------------------------
Logistic       Regression       Accuracy:       0.7198
 precision    recall  f1-score   support

               0       0.70      0.76      0.73
   10461           1       0.74      0.68      0.71
                        10539

    accuracy                           0.72     21000
macro avg        0.72      0.72      0.72     21000
weighted avg       0.72      0.72      0.72     21000

---------------------------------------------
Random Forest Accuracy: 0.7268             precision
 recall  f1-score   support

---------------------------------------------
```

In [6]:
```python
# Building the model
import joblib

# Train the final model
final_model = RandomForestClassifier()
final_model.fit(X_train, y_train)

# Save the model
joblib.dump(final_model, 'cardio_model.pkl')

# predictions = loaded_model.predict(new_data)
```

Out[6]:
```
['cardio_model.pkl']
```

```
              0       0.71      0.75      0.73
  10461           1       0.74      0.70      0.72
                  10539

    accuracy                           0.73     21000
macro avg        0.73      0.73      0.73     21000
weighted avg       0.73      0.73      0.73     21000
In                      [                      ]:
```