```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno

import warnings
warnings.filterwarnings('ignore')

sns.set()
plt.style.use('ggplot')


import pandas as pd
from google.colab import files

# Upload the file
uploaded = files.upload()

# Read the CSV file into a DataFrame
df = pd.read_csv(list(uploaded.keys())[0])

# Display the first few rows of the DataFrame
df.head()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving heart.csv to heart (5).csv
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHI
0	40	M	ATA	140	289	0	Normal	17
1	49	F	NAP	160	180	0	Normal	15
2	37	M	ATA	130	283	0	ST	9
3	48	F	ASY	138	214	0	Normal	10

```
import pandas as pd
from google.colab import files

# Upload the file
uploaded = files.upload()

# Read the CSV file into a DataFrame
df = pd.read_csv(list(uploaded.keys())[0])

# Display the column names
print("Column names:", df.columns)

# Display the first few rows of the DataFrame
print("First few rows of the DataFrame:")
print(df.head())

# Replace 'diagnosis' with the correct column name once identified
# For example, if the correct column name is 'HeartDisease':
# unique_values = df['HeartDisease'].unique()
# print("Unique values in 'HeartDisease':", unique_values)
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving heart.csv to heart (6).csv
Column names: Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
First few rows of the DataFrame:
   Age Sex ChestPainType  RestingBP  Cholesterol  FastingBS RestingECG  MaxHR
0   40   M           ATA        140          289          0     Normal    172
1   49   F           NAP        160          180          0     Normal    156
2   37   M           ATA        130          283          0         ST     98
3   48   F           ASY        138          214          0     Normal    108
4   54   M           NAP        150          195          0     Normal    122

  ExerciseAngina  Oldpeak ST_Slope  HeartDisease
0              N      0.0       Up             0
1              N      1.0     Flat             1
2              N      0.0       Up             0
3              Y      1.5     Flat             1
```

```
df.info()
```

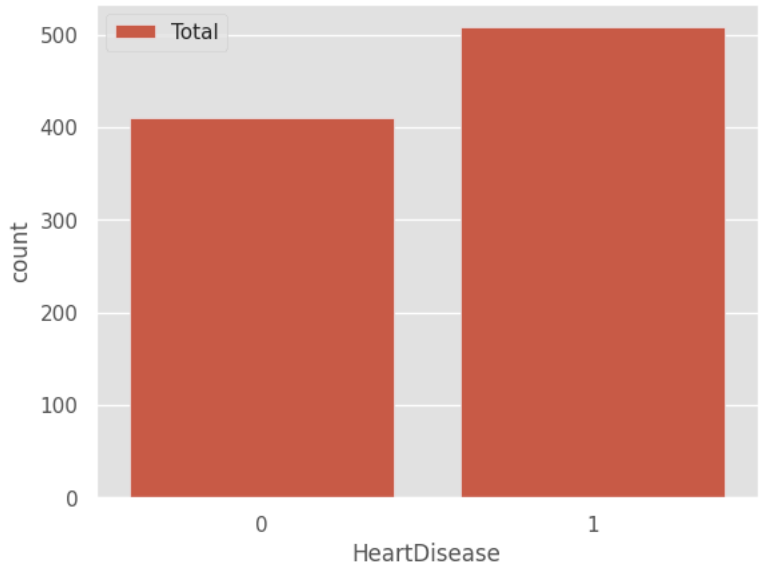
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

```
df.describe()
```

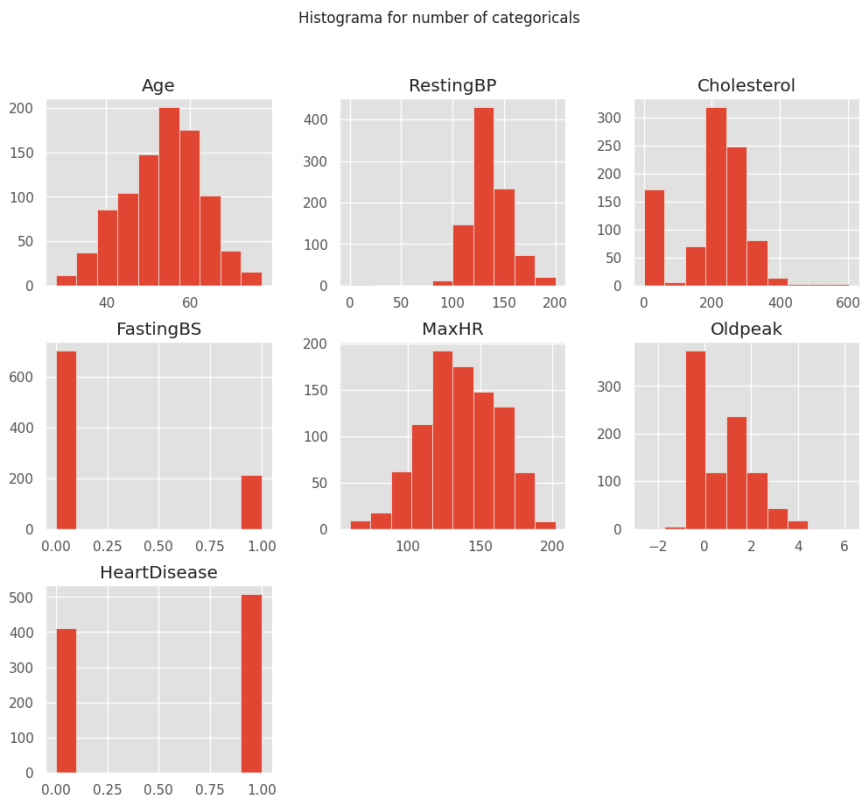
	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	Heart
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	91
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	

```
y=df['HeartDisease']
plot_sb = sns.countplot(df,x=y, label='Total')
Rain, NotRain =y.value_counts()
print('Have Heart Disease: ',Rain)
print('Not Have Heart Disease : ',NotRain)
```

```
Have Heart Disease:  508
Not Have Heart Disease :  410
```

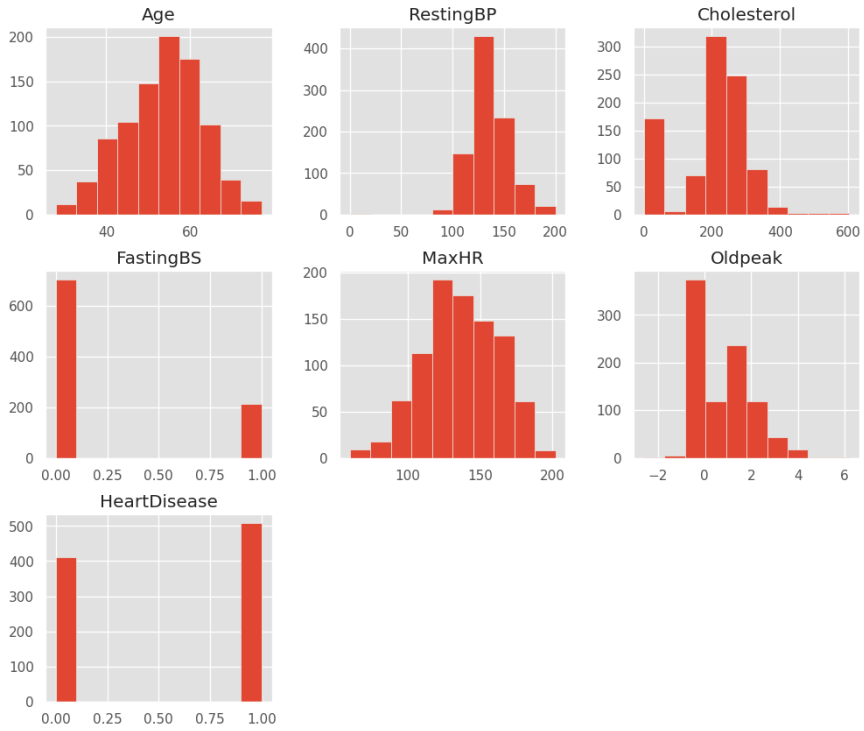



```
df.hist(figsize=(12, 10))
plt.suptitle('Histograma for number of categoricals')
plt.show()
```




```
df.hist(figsize=(12, 10))
plt.suptitle('Histograma for number of categoricals')
plt.show()
```

Histograma for number of categoricals



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")

!pip install xgboost catboost lightgbm

from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, ExtraTreesClassifier, AdaBoostClassifier
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from lightgbm import LGBMClassifier

from sklearn.metrics import confusion_matrix, classification_report

from google.colab import files
uploaded = files.upload()

df = pd.read_csv('heart.csv')

print(df.columns)

# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns

# Apply OneHotEncoder to categorical columns
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), categorical_cols)
    ],
    remainder='passthrough'
)

# Apply preprocessing
X = df.drop('HeartDisease', axis=1)  # Replace 'HeartDisease' with the actual target column name if different
y = df['HeartDisease']  # Replace 'HeartDisease' with the actual target column name if different

X = preprocessor.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packa
Collecting catboost
  Downloading catboost-1.2.5-cp310-cp310-manylinux2014_x86_64.whl (98.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.2/98.2 MB 7.5 MB/s eta 0:00:(
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pytho
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10,
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10,
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/d
Installing collected packages: catboost
Successfully installed catboost-1.2.5
Choose Files heart.csv
```

- **heart.csv**(text/csv) - 35921 bytes, last modified: 7/15/2024 - 100% done

```
Saving heart.csv to heart.csv
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS'
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
[[69  8]
 [13 94]]
              precision    recall  f1-score   support

           0       0.84      0.90      0.87        77
           1       0.92      0.88      0.90       107

    accuracy                           0.89       184
   macro avg       0.88      0.89      0.88       184
weighted avg       0.89      0.89      0.89       184
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")

!pip install xgboost catboost lightgbm

from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, ExtraTreesClassifier, AdaBoostClassifier
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from lightgbm import LGBMClassifier

from sklearn.metrics import confusion_matrix, classification_report

from google.colab import files
uploaded = files.upload()

df = pd.read_csv('heart.csv')

print(df.columns)

# Define LabelEncoders for categorical columns
sex = LabelEncoder()
cpt = LabelEncoder()
recg = LabelEncoder()
ea = LabelEncoder()
st = LabelEncoder()

# Encode categorical columns
df['Sex'] = sex.fit_transform(df['Sex'])
df['ChestPainType'] = cpt.fit_transform(df['ChestPainType'])
df['RestingECG'] = recg.fit_transform(df['RestingECG'])
```

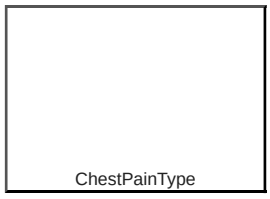
```
df['ExerciseAngina'] = ea.fit_transform(df['ExerciseAngina'])
df['ST_Slope'] = st.fit_transform(df['ST_Slope'])

# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns

# Apply OneHotEncoder to categorical columns
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), categorical_cols)
    ],
    remainder='passthrough'
)

# Apply preprocessing
X = df.drop('HeartDisease', axis=1)  # Replace 'HeartDisease' with the actual target column name if different
y = df['HeartDisease']  # Replace 'HeartDisease' with the actual target column name if different

X = preprocessor.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

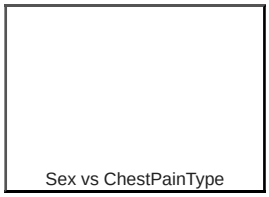
```
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: catboost in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pytho
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10,
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10,
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/d
Choose Files heart.csv
• heart.csv(text/csv) - 35921 bytes, last modified: 7/15/2024 - 100% done
Saving heart.csv to heart (1).csv
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS'
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
[[68  9]
 [10 97]]
              precision    recall  f1-score   support

           0       0.87      0.88      0.88        77
           1       0.92      0.91      0.91       107

    accuracy                           0.90       184
   macro avg       0.89      0.89      0.89       184
weighted avg       0.90      0.90      0.90       184
```

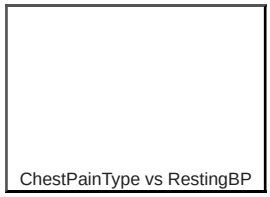
```
df.head()
```

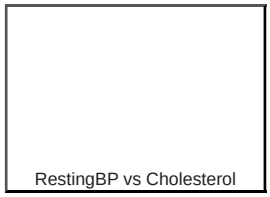
	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHI
0	40	1	1	140	289	0	1	17
1	49	0	2	160	180	0	1	15
2	37	1	1	130	283	0	2	9
3	48	0	0	138	214	0	1	10
4	54	1	2	150	195	0	1	12

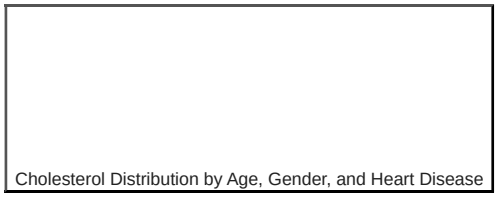
Next steps: Generate code with df View recommended plots

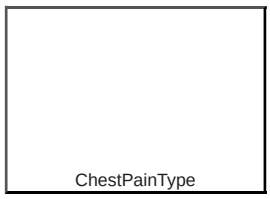


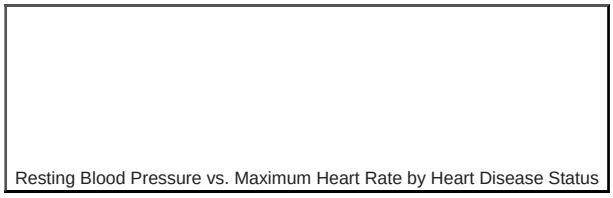


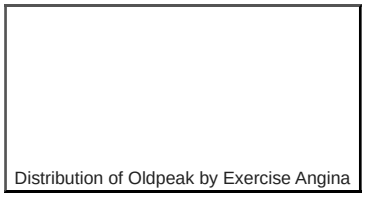






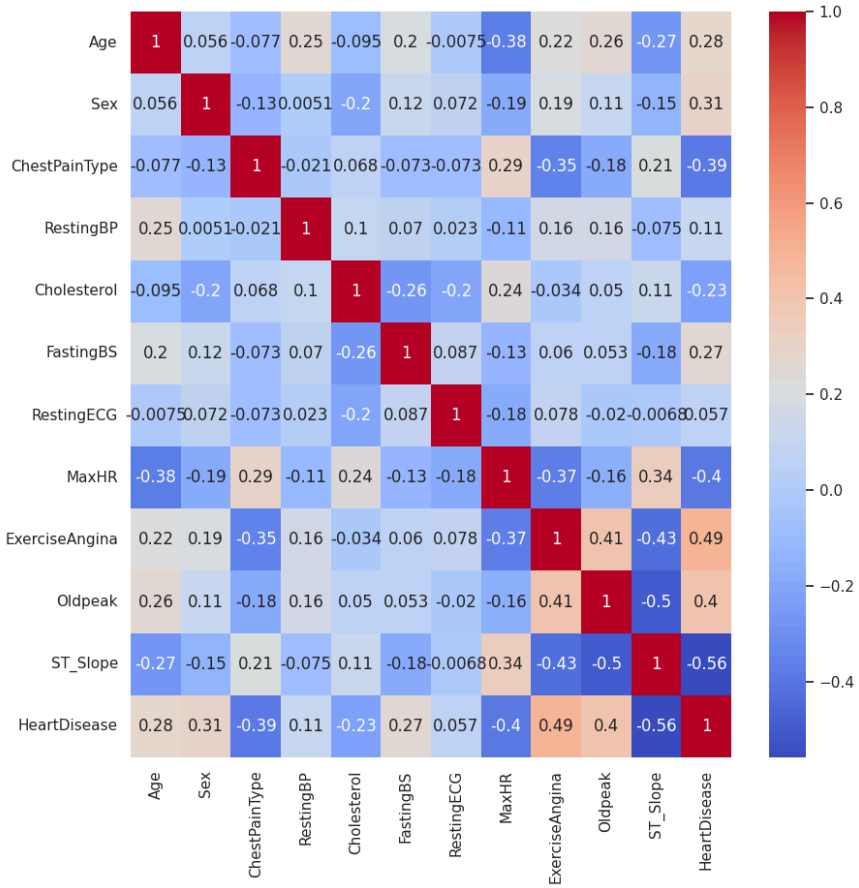







```
corr_matrix=df.corr(method='pearson')
plt.figure(figsize=(10,10))
sns.heatmap(corr_matrix,annot=True,cmap='coolwarm')
```

```
<Axes: >
```



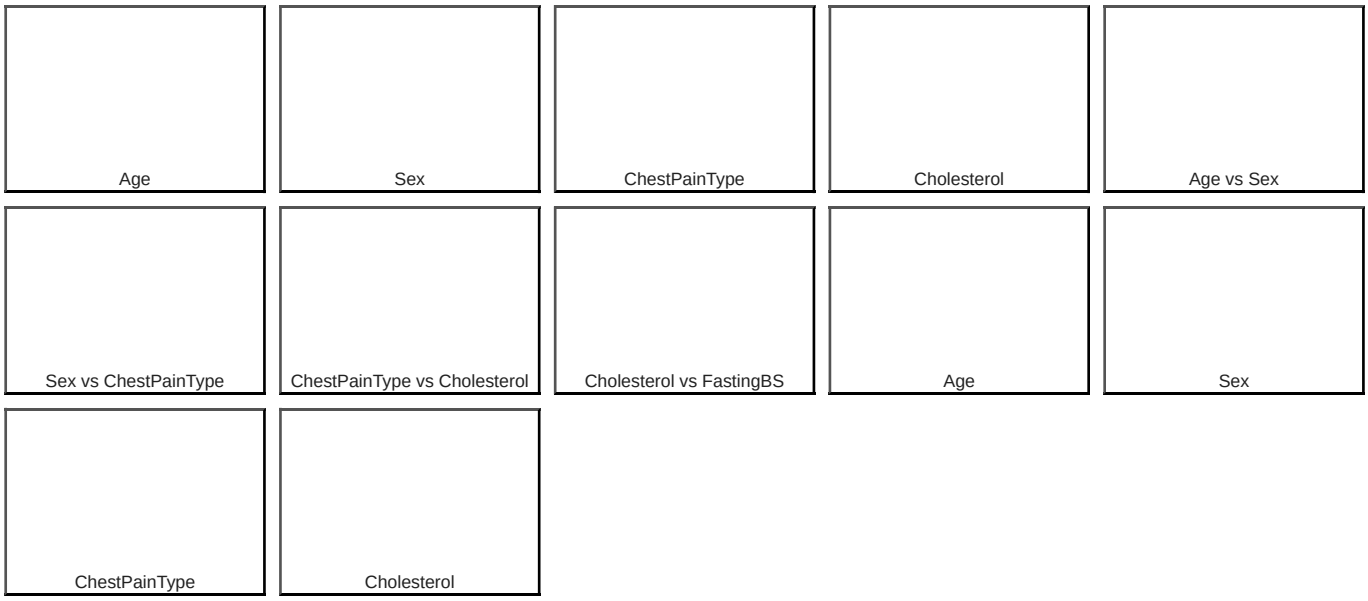

```
thresh_hold=0.2
select_feat=corr_matrix.index[abs(corr_matrix['HeartDisease'])>=thresh_hold].to_list()
select_feat.remove('HeartDisease')
print(select_feat)
```

```
['Age', 'Sex', 'ChestPainType', 'Cholesterol', 'FastingBS', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope']
```

```
selected_features=df[select_feat]
selected_features.head()
```

	Age	Sex	ChestPainType	Cholesterol	FastingBS	MaxHR	ExerciseAngina	Oldp
0	40	1	1	289	0	172	0	
1	49	0	2	180	0	156	0	
2	37	1	1	283	0	98	0	
3	48	0	0	214	0	108	1	
4	54	1	2	195	0	122	0	

Next steps: Generate code with `selected_features` View recommended plots




```
target=df['HeartDisease']
target
```

```
0      0
1      1
2      0
3      1
4      0
      ..
913    1
914    1
915    1
916    1
917    0
Name: HeartDisease, Length: 918, dtype: int64
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")

!pip install xgboost catboost lightgbm

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, ExtraTreesClassifier, AdaBoostClassifier
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from lightgbm import LGBMClassifier

from sklearn.metrics import confusion_matrix, classification_report

from google.colab import files
uploaded = files.upload()

df = pd.read_csv('heart.csv')

print(df.columns)

# Define LabelEncoders for categorical columns
sex = LabelEncoder()
cpt = LabelEncoder()
recg = LabelEncoder()
ea = LabelEncoder()
st = LabelEncoder()

# Encode categorical columns
df['Sex'] = sex.fit_transform(df['Sex'])
df['ChestPainType'] = cpt.fit_transform(df['ChestPainType'])
df['RestingECG'] = recg.fit_transform(df['RestingECG'])
```

```
df['ExerciseAngina'] = ea.fit_transform(df['ExerciseAngina'])
df['ST_Slope'] = st.fit_transform(df['ST_Slope'])

# Separate features and target
X = df.drop('HeartDisease', axis=1)  # Replace 'HeartDisease' with the actual target column name if different
y = df['HeartDisease']  # Replace 'HeartDisease' with the actual target column name if different

# Define selected features (for the example, we use all features; adjust as needed)
selected_features = X

# Scale the features
scaler = StandardScaler()
selected_features = scaler.fit_transform(selected_features)

print(selected_features)

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(selected_features, y, test_size=0.2, random_state=42)

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

# Predict and evaluate
y_pred = model.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: catboost in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pytho
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10,
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10,
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/d
Choose Files heart.csv
• heart.csv(text/csv) - 35921 bytes, last modified: 7/15/2024 - 100% done
Saving heart.csv to heart (2).csv
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS'
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
[[-1.4331398   0.51595242  0.22903206 ... -0.8235563  -0.83243239
   1.05211381]
 [-0.47848359 -1.93816322  1.27505906 ... -0.8235563   0.10566353
  -0.59607813]
 [-1.75135854  0.51595242  0.22903206 ... -0.8235563  -0.83243239
   1.05211381]
 ...
 [ 0.37009972  0.51595242 -0.81699495 ...  1.21424608  0.29328271
  -0.59607813]
 [ 0.37009972 -1.93816322  0.22903206 ... -0.8235563  -0.83243239
  -0.59607813]
 [-1.64528563  0.51595242  1.27505906 ... -0.8235563  -0.83243239
   1.05211381]]
[[68  9]
 [12 95]]
              precision    recall  f1-score   support

           0       0.85      0.88      0.87        77
           1       0.91      0.89      0.90       107

    accuracy                           0.89       184
   macro avg       0.88      0.89      0.88       184
weighted avg       0.89      0.89      0.89       184
```

```
x_train,x_test,y_train,y_test=train_test_split(selected_features,target,test_size=0.2)
```

```
from sklearn.linear_model import LogisticRegression

LOG_model=LogisticRegression()
LOG_model.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
log_pred=LOG_model.predict(x_test)


from sklearn.metrics import accuracy_score  # Import the missing function

accuracy = accuracy_score(y_test, log_pred)

print("Accuracy=", int(accuracy * 100), '%')
```