```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
sns.set_theme(color_codes=True)
```

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

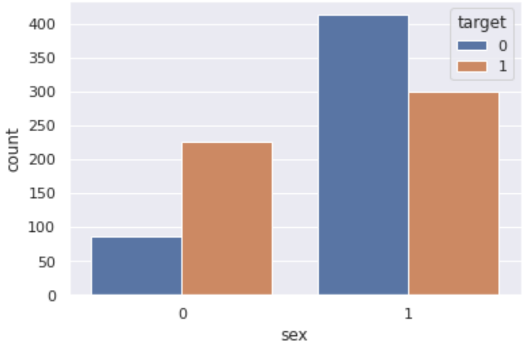
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tl
0	52	1	0	125	212	0	1	168	0	1.0	2	2	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	

1025 rows × 14 columns

Exploratory Data Analysis

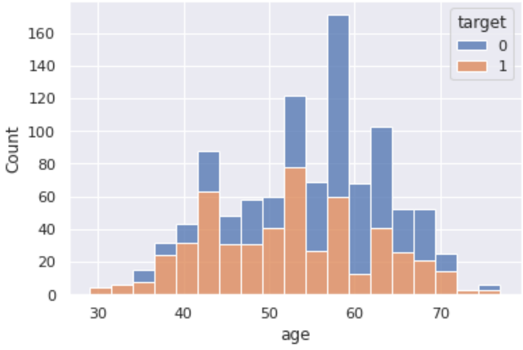
```
sns.countplot(data=df, x="sex", hue="target")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe8f8f72b0>
```



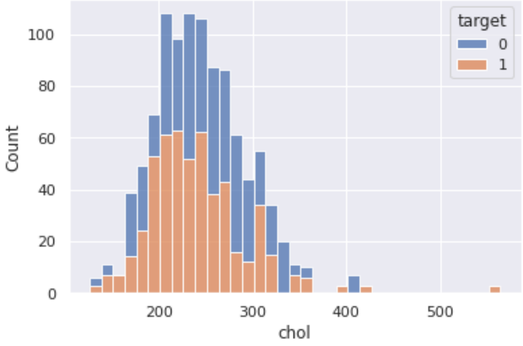
```
sns.histplot(data=df, x="age", hue="target", multiple="stack")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe8e6c5250>
```



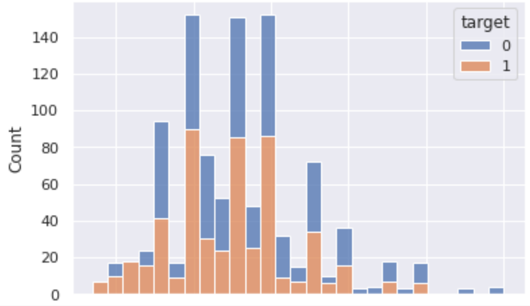
```
sns.histplot(data=df, x="chol", hue="target", multiple="stack")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe8e5df3d0>
```



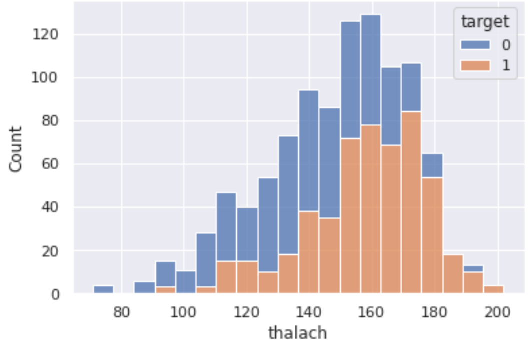
```
sns.histplot(data=df, x="trestbps", hue="target", multiple="stack")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe8e49aa30>
```



```
sns.histplot(data=df, x="thalach", hue="target", multiple="stack")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe8e3762b0>
```



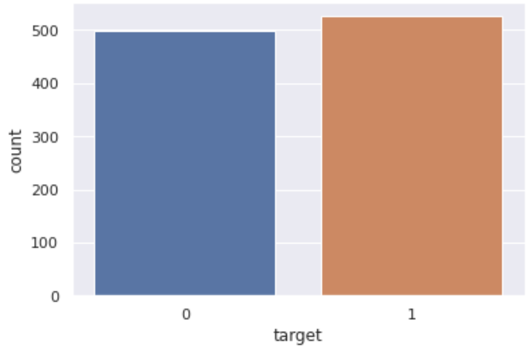
Data Preprocessing

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

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

```
sns.countplot(df['target'])
print(df.target.value_counts())
```

```
1    526
0    499
Name: target, dtype: int64
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: P
  warnings.warn(
```



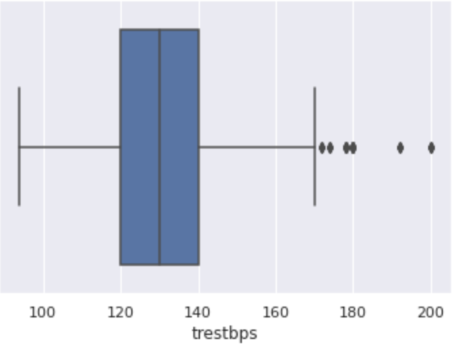
Outlier Detection Using Boxplot

```
sns.boxplot(x=df["age"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe8e2ef850>
```

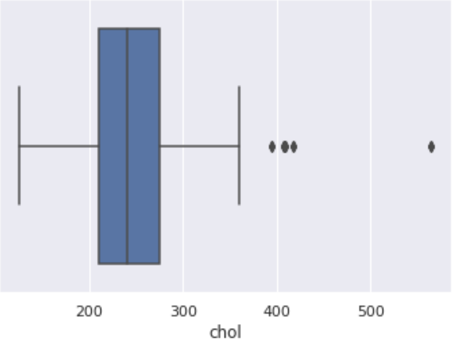
```
sns.boxplot(x=df["trestbps"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe8e2561f0>
```



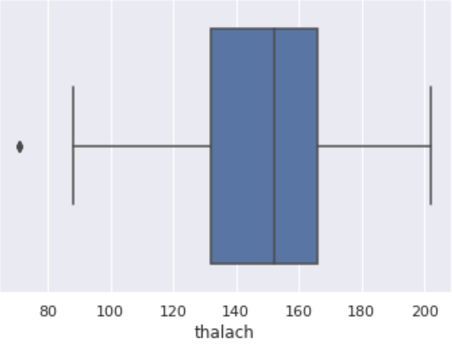
```
sns.boxplot(x=df["chol"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe8e262d90>
```



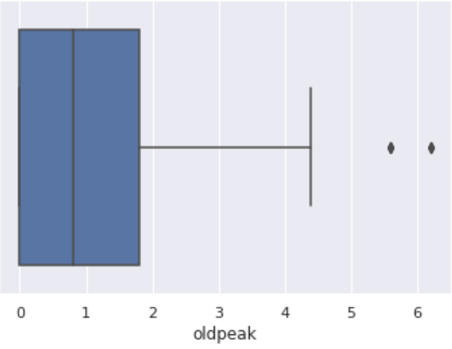
```
sns.boxplot(x=df["thalach"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe8e163670>
```



```
sns.boxplot(x=df["oldpeak"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe8e183100>
```



Outlier Removal Using Z-Score

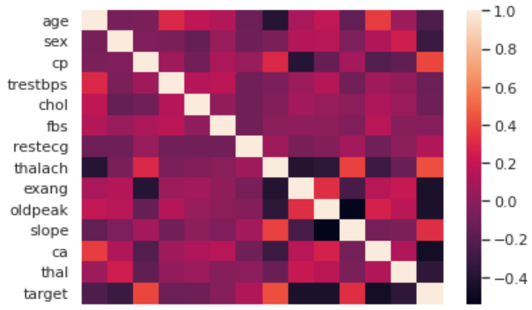
```
import scipy.stats as stats
z = np.abs(stats.zscore(df))
data_clean = df[(z<3).all(axis = 1)]
data_clean.shape
```

```
(969, 14)
```

Data Correlation using Heatmap

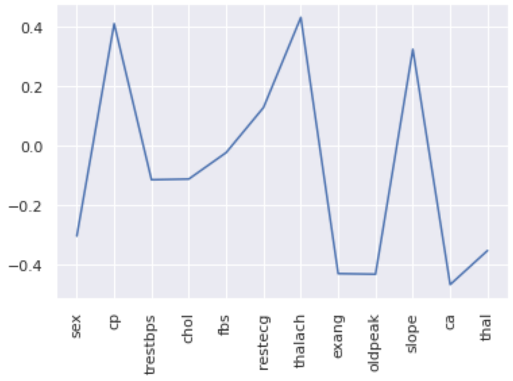
```
sns.heatmap(data_clean.corr(), fmt='.2g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe8e284070>
```



Correlation between Class and other attributes

```
corr = data_clean[data_clean.columns[1:]].corr()['target'][:-1]
plt.plot(corr)
plt.xticks(rotation=90)
plt.show()
```



Machine Learning Model Building

```
X = data_clean.drop('target', axis=1)
y = data_clean['target']
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,random_state=0)
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(random_state = 0)
dtree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(random_state=0)
```

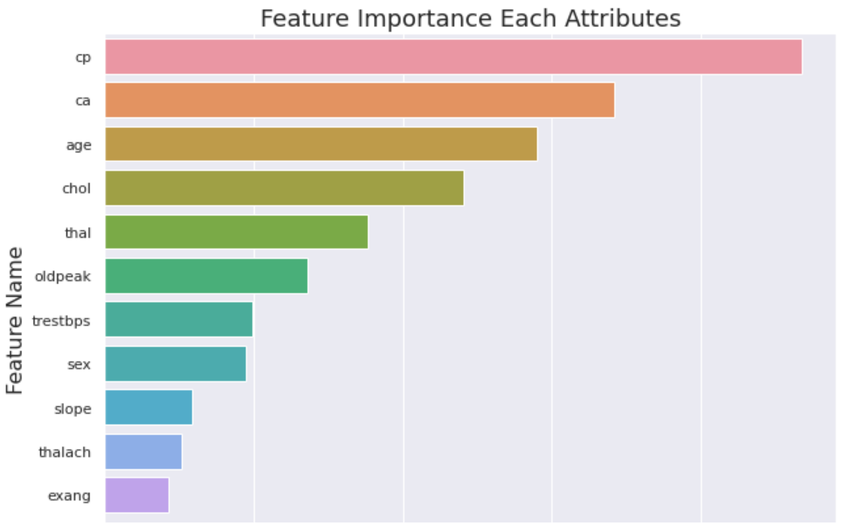
```
y_pred = dtree.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 100.0 %
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  1.0
Precision Score :  1.0
Recall Score :  1.0
```

```
#Feature Importance
imp_df = pd.DataFrame({
    "Feature Name": X_train.columns,
    "Importance": dtree.feature_importances_
})
fi = imp_df.sort_values(by="Importance", ascending=False)
plt.figure(figsize=(10,8))
sns.barplot(data=fi, x='Importance', y='Feature Name')
plt.title('Feature Importance Each Attributes', fontsize=18)
plt.xlabel ('Importance', fontsize=16)
plt.ylabel ('Feature Name', fontsize=16)
plt.show()
```



Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state=0)
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=0)
```

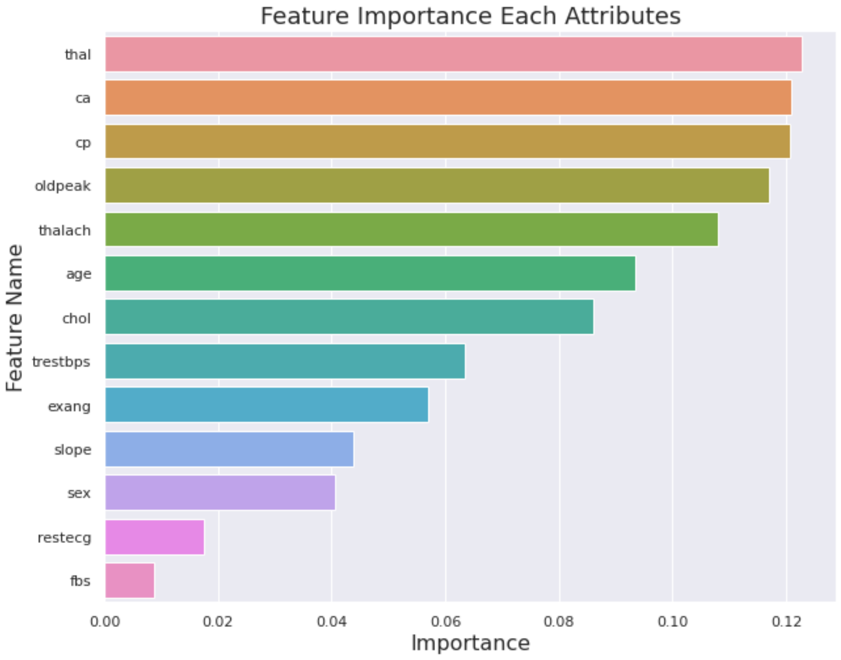
```
y_pred = rfc.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 100.0 %
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  1.0
Precision Score :  1.0
Recall Score :  1.0
```

```
#Feature Importance
imp_df = pd.DataFrame({
    "Feature Name": X_train.columns,
    "Importance": rfc.feature_importances_
})
fi = imp_df.sort_values(by="Importance", ascending=False)
plt.figure(figsize=(10,8))
sns.barplot(data=fi, x='Importance', y='Feature Name')
plt.title('Feature Importance Each Attributes', fontsize=18)
plt.xlabel ('Importance', fontsize=16)
plt.ylabel ('Feature Name', fontsize=16)
plt.show()
```



AdaBoost

```
from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(random_state=0)
ada.fit(X_train, y_train)
```

```
AdaBoostClassifier(random_state=0)
```

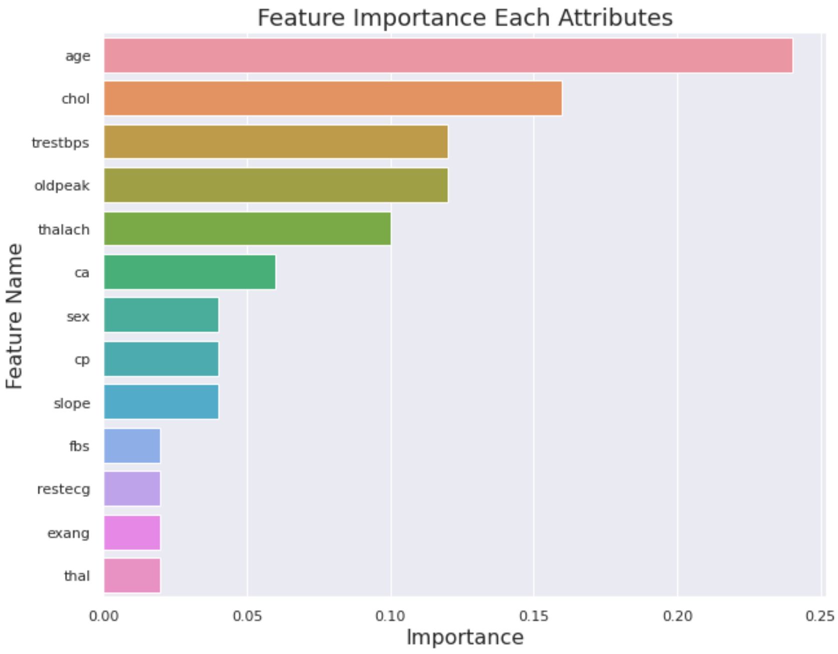
```
y_pred = ada.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

```
Accuracy Score : 93.3 %
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print('F-1 Score : ',(f1_score(y_test, y_pred)))
print('Precision Score : ',(precision_score(y_test, y_pred)))
print('Recall Score : ',(recall_score(y_test, y_pred)))
```

```
F-1 Score :  0.9365853658536586
Precision Score :  0.9411764705882353
Recall Score :  0.9320388349514563
```

```
#Feature Importance
imp_df = pd.DataFrame({
    "Feature Name": X_train.columns,
    "Importance": ada.feature_importances_
})
fi = imp_df.sort_values(by="Importance", ascending=False)
plt.figure(figsize=(10,8))
sns.barplot(data=fi, x='Importance', y='Feature Name')
plt.title('Feature Importance Each Attributes', fontsize=18)
plt.xlabel ('Importance', fontsize=16)
plt.ylabel ('Feature Name', fontsize=16)
plt.show()
```



Colab paid products - Cancel contracts here

✓ 1s completed at 11:01 AM