Visualizing And Predicting Heart Disease With An Interactive Dashboard

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

```
df = pd.read_csv("C:\\Users\\User\\Downloads\\Heart_Disease_Prediction.csv")

df
```

Out[3]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Heart Disease
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presence
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absence
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presence
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absence
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absence
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
265	52	1	3	172	199	1	0	162	0	0.5	1	0	7	Absence
266	44	1	2	120	263	0	0	173	0	0.0	1	0	7	Absence
267	56	0	2	140	294	0	2	153	0	1.3	2	0	3	Absence
268	57	1	4	140	192	0	0	148	0	0.4	2	0	6	Absence
269	67	1	4	160	286	0	2	108	1	1.5	2	3	3	Presence

270 rows × 14 columns

```
df.head()
```

Out[4]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Heart Disease
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presence
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absence
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presence
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absence
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absence

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       270 non-null    int64
 1   Sex                       270 non-null    int64
 2   Chest pain type           270 non-null    int64
 3   BP                        270 non-null    int64
 4   Cholesterol               270 non-null    int64
 5   FBS over 120              270 non-null    int64
 6   EKG results               270 non-null    int64
 7   Max HR                    270 non-null    int64
 8   Exercise angina           270 non-null    int64
 9   ST depression             270 non-null    float64
 10  Slope of ST               270 non-null    int64
 11  Number of vessels fluro   270 non-null    int64
 12  Thallium                  270 non-null    int64
 13  Heart Disease             270 non-null    object
dtypes: float64(1), int64(12), object(1)
memory usage: 29.7+ KB
```

df.shape

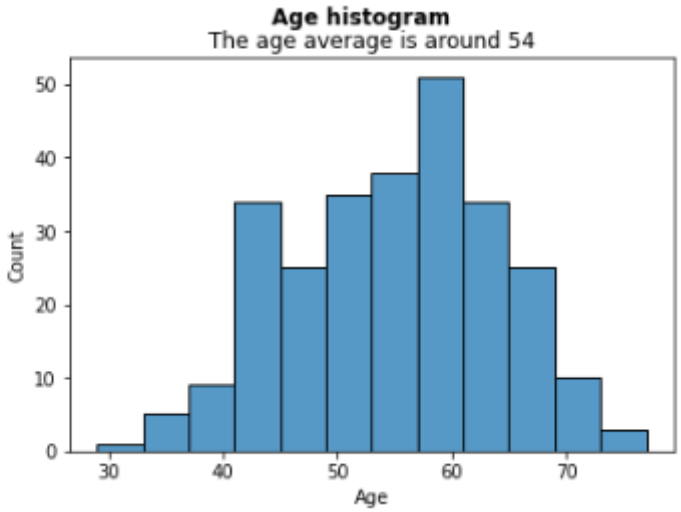
```
(270, 14)
```

df.nunique()

```
Out[8]: Age                       41
        Sex                        2
        Chest pain type            4
        BP                        47
        Cholesterol              144
        FBS over 120               2
        EKG results                3
        Max HR                    90
        Exercise angina            2
        ST depression             39
        Slope of ST                3
        Number of vessels fluro    4
        Thallium                   3
        Heart Disease              2
        dtype: int64
```

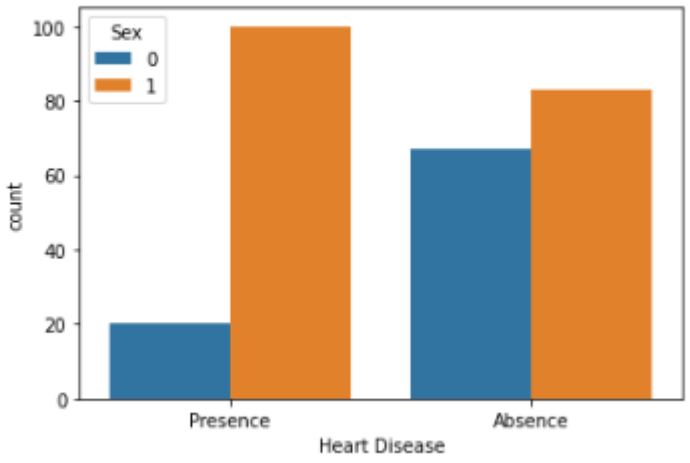
Data Exploration And Visualization

```
plt.suptitle('Age histogram', fontweight='heavy')

plt.title('The age average is around 54')

sns.histplot(data=df, x='Age')

plt.show()
```



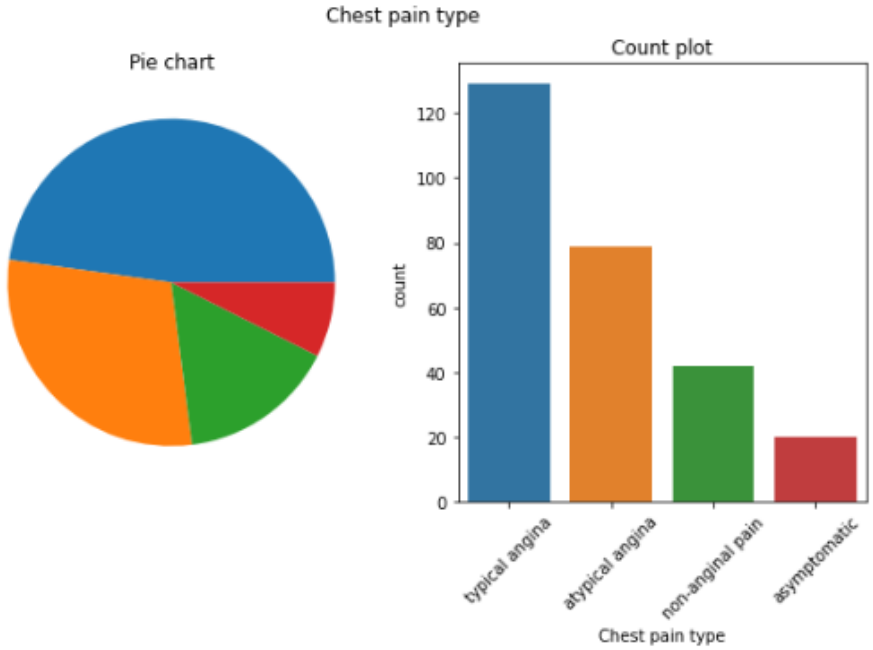

```
sns.countplot(x=df['Heart Disease'],hue='Sex',data=df)
```

```
Out[11]: <AxesSubplot:xlabel='Heart Disease', ylabel='count'>
```



It is observed that males have more chances to have disease than females

```
labels = ["typical angina", "atypical angina", "non-anginal pain", "asymptomatic"]
order = df['Chest pain type'].value_counts().index

plt.figure(figsize=(10,5))
plt.suptitle("Chest pain type")

plt.subplot(1,2,1)
plt.title('Pie chart')
plt.pie(df['Chest pain type'].value_counts(), textprops={'fontsize':12})
plt.subplots_adjust(left=0.125)

plt.subplot(1,2,2)
plt.title('Count plot')
sns.countplot(x='Chest pain type', data=df, order=order)
plt.xticks([0,1,2,3], labels, rotation=45)

plt.show()

df['Chest pain type'].value_counts()
```




```
Out[12]: 4    129
         3     79
         2     42
         1     20
         Name: Chest pain type, dtype: int64
```

labels = ["normal", 'aving ST-T wave abnormality', "showing probable or definite left ventricular hypertrophy by Estes' criteria"]

order = df['EKG results'].value_counts().index

plt.figure(figsize=(10,5))

plt.suptitle("EKG results")

plt.subplot(1,2,1)

plt.title('Pie chart')

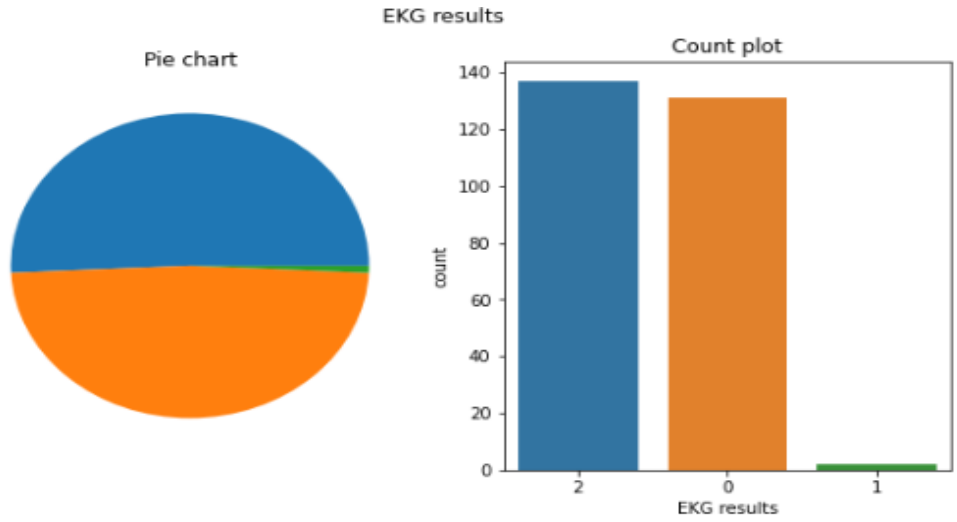
plt.pie(df['EKG results'].value_counts(), textprops={'fontsize':12})

plt.subplots_adjust(left=0.125)

plt.subplot(1,2,2)

```
plt.title('Count plot')

sns.countplot(x='EKG results', data=df, order=order)

plt.show()
```

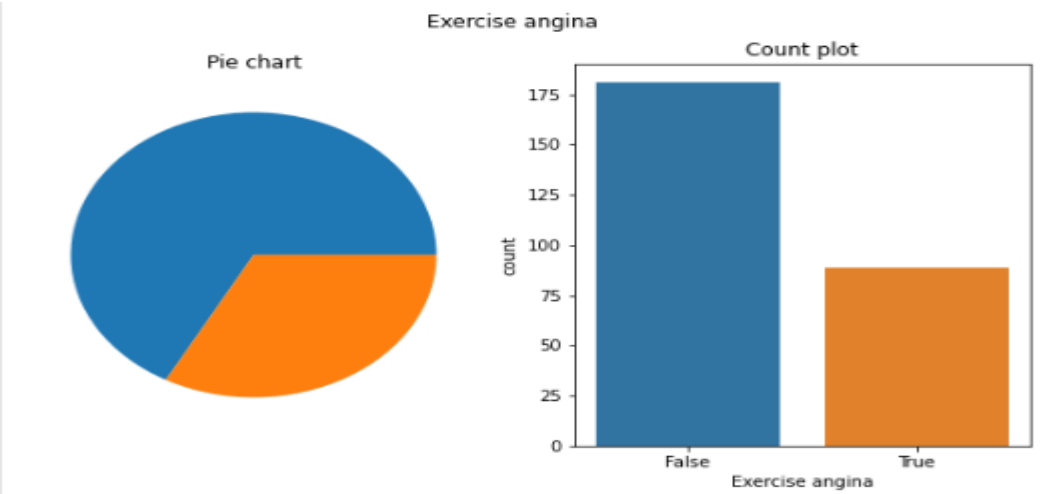
```
df['EKG results'].value_counts()
```




```
Out[13]: 2    137
         0    131
         1      2
         Name: EKG results, dtype: int64
```

```
labels = ["False", 'True']

order = df['Exercise angina'].value_counts().index


plt.figure(figsize=(10,5))

plt.suptitle("Exercise angina")


plt.subplot(1,2,1)

plt.title('Pie chart')

plt.pie(df['Exercise angina'].value_counts(), textprops={'fontsize':12})
```

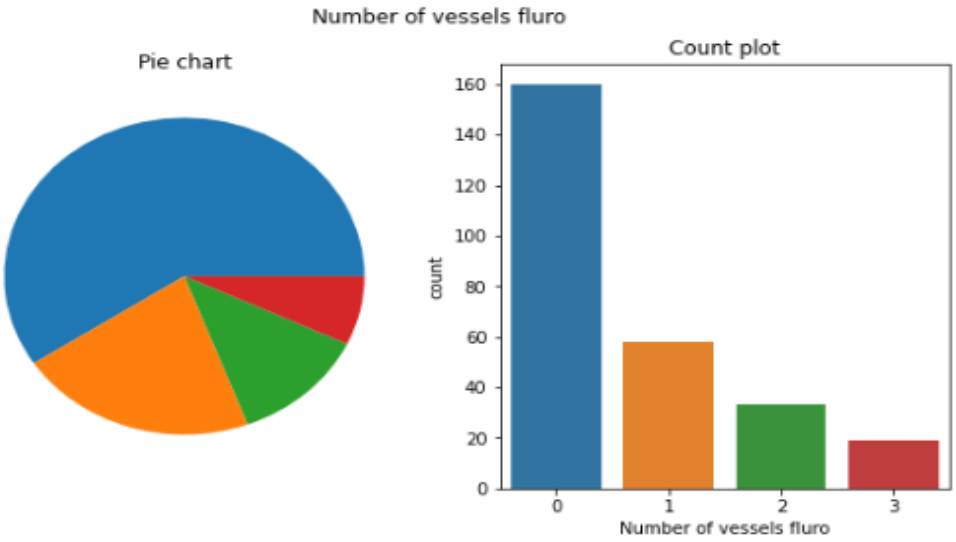
```
plt.subplots_adjust(left=0.125)


plt.subplot(1,2,2)
plt.title('Count plot')
sns.countplot(x='Exercise angina', data=df, order=order)
plt.xticks([0,1], labels=labels)


plt.show()


df['Exercise angina'].value_counts()
```




```
0    181
1     89
Name: Exercise angina, dtype: int64
```

```
labels = ["0", '1', '2', "3"]
order = df['Number of vessels fluro'].value_counts().index
```

```
plt.figure(figsize=(10,5))
plt.suptitle("Number of vessels fluro")

plt.subplot(1,2,1)
plt.title('Pie chart')
plt.pie(df['Number of vessels fluro'].value_counts(), textprops={'fontsize':12})
plt.subplots_adjust(left=0.125)

plt.subplot(1,2,2)
plt.title('Count plot')
sns.countplot(x='Number of vessels fluro', data=df, order=order)
plt.xticks([0,1,2,3], labels=labels)
plt.show()

df['Number of vessels fluro'].value_counts()
```



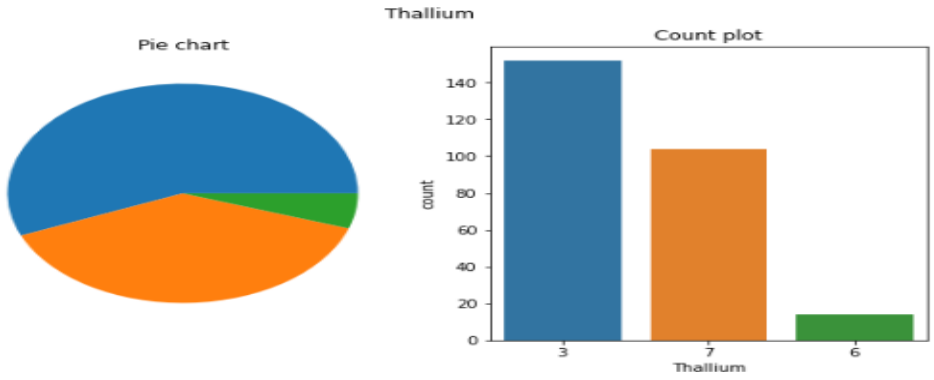

```
Out[15]: 0    160
         1     58
         2     33
         3     19
         Name: Number of vessels fluro, dtype: int64
```

```
labels = ["3", '7', '6']

order = df['Thallium'].value_counts().index


plt.figure(figsize=(10,5))

plt.suptitle("Thallium")


plt.subplot(1,2,1)

plt.title('Pie chart')

plt.pie(df['Thallium'].value_counts(), textprops={'fontsize':12})

plt.subplots_adjust(left=0.125)


plt.subplot(1,2,2)

plt.title('Count plot')

sns.countplot(x='Thallium', data=df, order=order)
```

plt.xticks([0,1,2], labels=labels)

plt.show()

df['Thallium'].value_counts()




```
Out[16]: 3    152
         7    104
         6     14
         Name: Thallium, dtype: int64
```

```
target = df['Heart Disease'].map({'Presence':1, 'Absence':0})
inputs = df.drop(['Heart Disease'], axis=1)
```

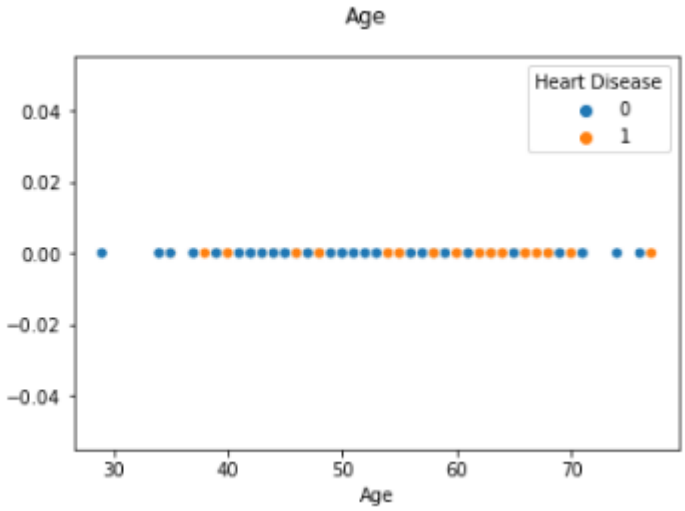
target = df['Heart Disease'].map({'Presence':1, 'Absence':0})

inputs = df.drop(['Heart Disease'], axis=1)

plt.suptitle("Age")

sns.scatterplot(data=df, x='Age', y=np.zeros(len(df['Age'])), hue=target)

plt.show()

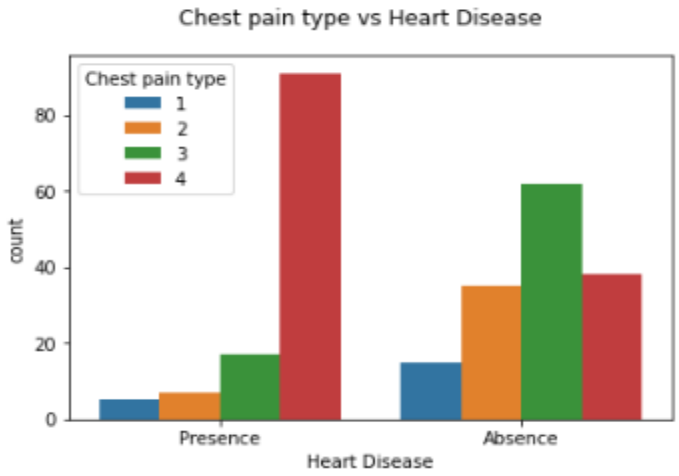


Heart disease based on age – As You Can See That Older people have more chance to have heart disease.

plt.suptitle('Chest pain type vs Heart Disease')

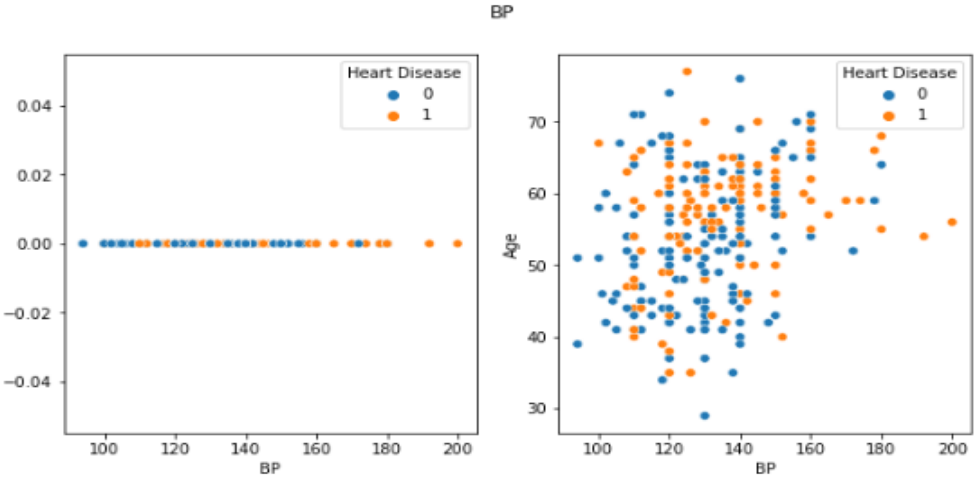
sns.countplot(data=df, x='Heart Disease', hue='Chest pain type')

plt.show()



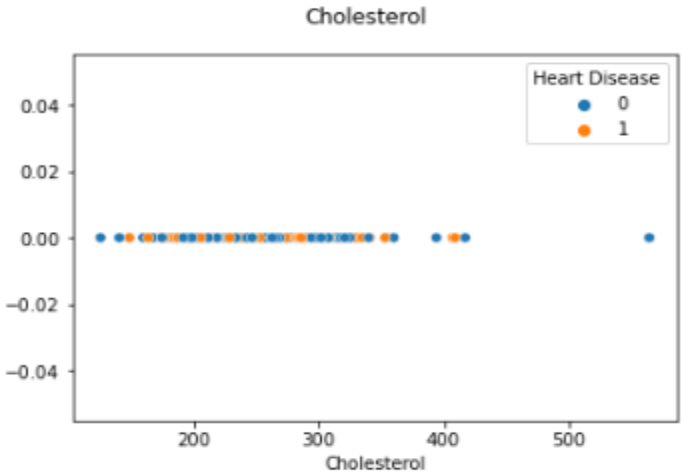
Heart disease based on Chest pain type - 4th type of chest pain dominate in heart disease.

```
plt.figure(figsize=(10,5))


plt.subplot(1,2,1)

plt.suptitle("BP")

sns.scatterplot(data=df, x='BP', y=np.zeros(len(df['BP'])), hue=target)


plt.subplot(1,2,2)

sns.scatterplot(data=df, x='BP', y='Age', hue=target)



plt.show()
```



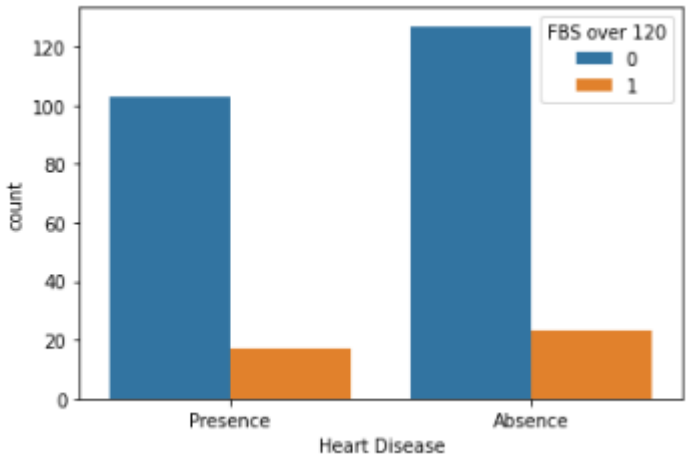
Heart Disease based on BP - Persons with high BP have more chance to get heart disease.

```
plt.suptitle("Cholesterol")

sns.scatterplot(data=df, x='Cholesterol', y=np.zeros(len(df['Cholesterol'])),
hue=target)

plt.show()
```



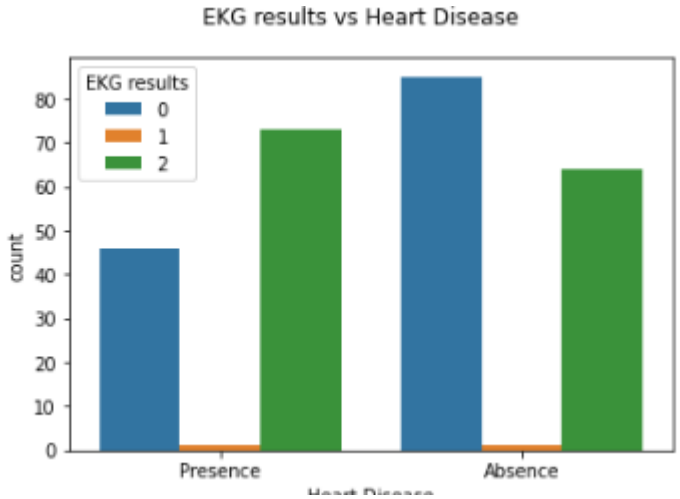
Cholesterol - Higer Cholesterol does not influence on heart disease.

```
ax = sns.countplot(x='Heart Disease', hue='FBS over 120', data=df)

plt.show()
```



FBS over 120 - AIso increased FBS over 120 does not imply on heart disease prediction.

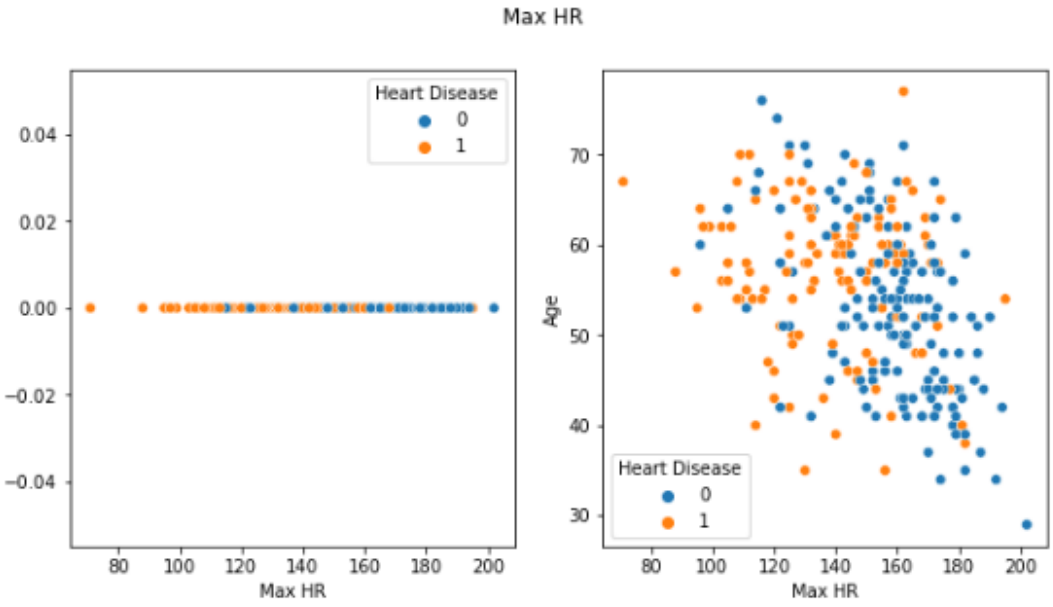
```
plt.suptitle('EKG results vs Heart Disease')

sns.countplot(data=df, x='Heart Disease', hue='EKG results')

plt.show()
```



EKG results - The 2nd value of EKG could influence on heart disease prediction.

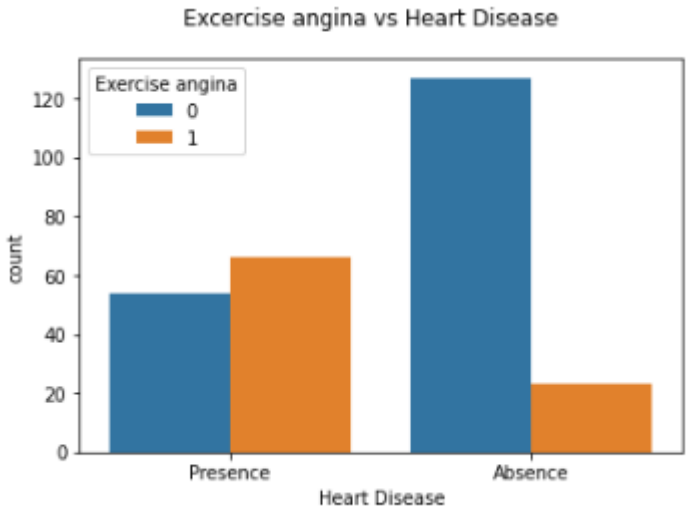
```
plt.figure(figsize=(10,5))


plt.subplot(1,2,1)
```

```
plt.suptitle("Max HR")

sns.scatterplot(data=df, x='Max HR', y=np.zeros(len(df['Max HR'])), hue=target)


plt.subplot(1,2,2)

sns.scatterplot(data=df, x='Max HR', y='Age', hue=target)


plt.show()
```



Max HR - From the first graph we can observe that people with lower HR max have a higher likelihood of heart disease than those with higher HR max. Furthermore, we can observe explicit cut of/threshold where below 120 HR max objects have a higher probability to have problem with heart.

```
plt.suptitle('Excercise angina vs Heart Disease')

sns.countplot(data=df, x='Heart Disease', hue='Exercise angina')

plt.show()
```



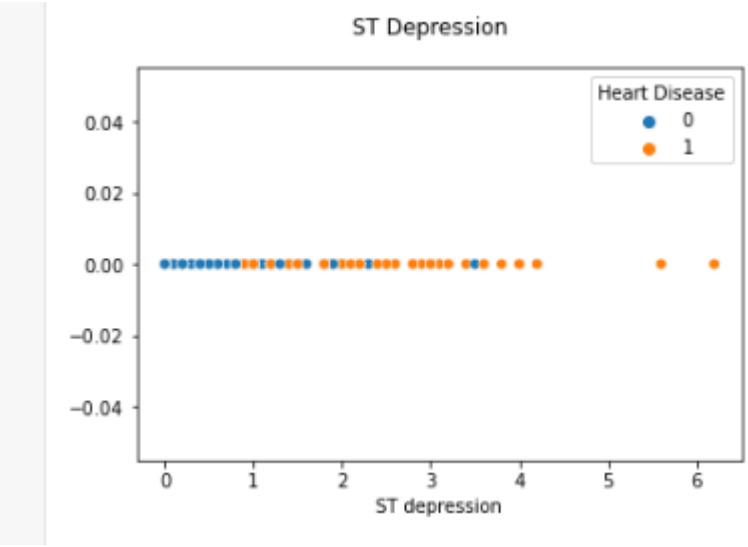
Excercise angina - Chest pain after a high excercise intensity or stress can casues a presence of heart failure.

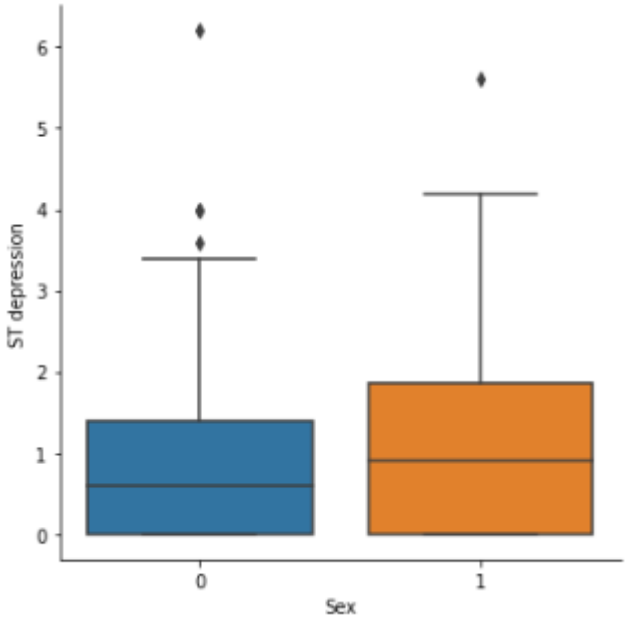
plt.suptitle("ST Depression")

sns.scatterplot(data=df, x='ST depression', y=np.zeros(len(df['ST depression'])), hue=target)

ax = sns.catplot(x='Sex', y='ST depression', kind='box', data = df)

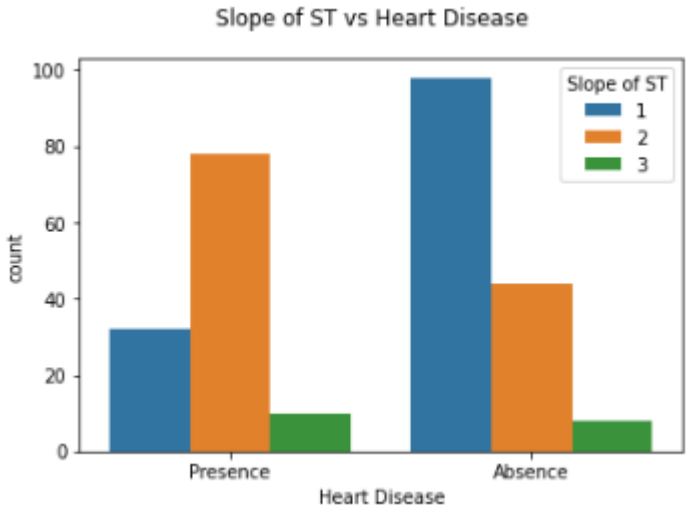
plt.show()





ST depression - Increased ST deprsesion increase heart disease. Nevetheless it can be observe on the bottom figure that males have higher probability of having depression.

```
plt.suptitle('Slope of ST vs Heart Disease')

sns.countplot(data=df, x='Heart Disease', hue='Slope of ST')

plt.show()
```




```
Chest_pain_type = pd.get_dummies(df['Chest pain type'], prefix='Chest pain type', drop_first=True)

EKG_results = pd.get_dummies(df['EKG results'], prefix='EKG results', drop_first=True)

Number_of_vessels_fluro = pd.get_dummies(df['Number of vessels fluro'], prefix='Number of vessels fluro', drop_first=True)

Thallium = pd.get_dummies(df['Thallium'], prefix='Thallium', drop_first=True)


frames = [df, Chest_pain_type, EKG_results, Number_of_vessels_fluro, Thallium]

df = pd.concat(frames, axis=1)


df.drop(columns = ['Chest pain type', 'EKG results', 'Number of vessels fluro', 'Thallium', 'Slope of ST'])

target = df['Heart Disease'].map({'Presence':1, 'Absence':0})
```

```
inputs = df.drop(['Heart Disease'], axis=1)
```

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Age	270.0	54.433333	9.109067	29.0	48.0	55.0	61.0	77.0
Sex	270.0	0.677778	0.468195	0.0	0.0	1.0	1.0	1.0
Chest pain type	270.0	3.174074	0.950090	1.0	3.0	3.0	4.0	4.0
BP	270.0	131.344444	17.861608	94.0	120.0	130.0	140.0	200.0
Cholesterol	270.0	249.659259	51.686237	126.0	213.0	245.0	280.0	564.0
FBS over 120	270.0	0.148148	0.355906	0.0	0.0	0.0	0.0	1.0
EKG results	270.0	1.022222	0.997891	0.0	0.0	2.0	2.0	2.0
Max HR	270.0	149.677778	23.165717	71.0	133.0	153.5	166.0	202.0
Exercise angina	270.0	0.329630	0.470952	0.0	0.0	0.0	1.0	1.0
ST depression	270.0	1.050000	1.145210	0.0	0.0	0.8	1.6	6.2
Slope of ST	270.0	1.585185	0.614390	1.0	1.0	2.0	2.0	3.0
Number of vessels fluro	270.0	0.670370	0.943896	0.0	0.0	0.0	1.0	3.0
Thallium	270.0	4.696296	1.940659	3.0	3.0	3.0	7.0	7.0
Chest pain type_2	270.0	0.155556	0.363107	0.0	0.0	0.0	0.0	1.0
Chest pain type_3	270.0	0.292593	0.455798	0.0	0.0	0.0	1.0	1.0
Chest pain type_4	270.0	0.477778	0.500434	0.0	0.0	0.0	1.0	1.0
EKG results_1	270.0	0.007407	0.085906	0.0	0.0	0.0	0.0	1.0
EKG results_2	270.0	0.507407	0.500874	0.0	0.0	1.0	1.0	1.0
Number of vessels fluro_1	270.0	0.214815	0.411456	0.0	0.0	0.0	0.0	1.0
Number of vessels fluro_2	270.0	0.122222	0.328151	0.0	0.0	0.0	0.0	1.0
Number of vessels fluro_3	270.0	0.070370	0.256245	0.0	0.0	0.0	0.0	1.0
Thallium_6	270.0	0.051852	0.222140	0.0	0.0	0.0	0.0	1.0
Thallium_7	270.0	0.385185	0.487543	0.0	0.0	0.0	1.0	1.0

```
one_target = int(np.sum(target))

zero_counter = 0

indices_to_remove = []


for i in range(target.shape[0]):

  if target[i] == 0:

    zero_counter += 1

    if zero_counter > one_target:

      indices_to_remove.append(i)


print("Indices before balancing data:", target.shape[0])

print("Idices to delete:", len(indices_to_remove))
```

```
Indices before balancing data: 270
Idices to delete: 30
```

```
balanced_inputs = inputs.drop(indices_to_remove, axis=0)

balanced_targets = target.drop(indices_to_remove, axis=0)


#reset indices

reset_inputs = balanced_inputs.reset_index(drop=True)

reset_targets = balanced_targets.reset_index(drop=True)
```

```
print("Inputs after balancing data:", reset_inputs.shape[0])

print("Targets after balancing data:", reset_targets.shape[0])
```

```
Inputs after balancing data: 240
Targets after balancing data: 240
```

```
from sklearn.preprocessing import MinMaxScaler

scaled_inputs = MinMaxScaler().fit_transform(balanced_inputs)
```

Splitting It Into Training And Testing

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(scaled_inputs, balanced_targets,
test_size=0.2)
```

Applying Different Models

```
from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score


lr = LinearRegression()

lr.fit(X_train, y_train)


predicted = lr.predict(X_test)
```

```
RMSE = np.sqrt(mean_squared_error(y_test, predicted))

r2 = r2_score(y_test, predicted)


print('Root mean squared error: ', RMSE)

print("r2: ", r2)
```

```
Root mean squared error:  0.36416508651208407
r2:  0.4611150822223089
```

```
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

from sklearn import metrics

from sklearn.metrics import roc_curve


logit = LogisticRegression()

logit.fit(X_train, y_train)


predicted_logit = logit.predict(X_test)


LogisticRegressionScore = accuracy_score(predicted_logit, y_test)


print("Logistic Regression score: ", LogisticRegressionScore)
```

```
Logistic Regression score:  0.875
```

```
from sklearn.naive_bayes import GaussianNB

gauss = GaussianNB()
gauss.fit(X_train, y_train)

gauss_pred = gauss.predict(X_test)

gauss_score = accuracy_score(gauss_pred, y_test)
print("Gaussian Naive Bayes score: ", gauss_score)
```

```
Gaussian Naive Bayes score:  0.7708333333333334
```

```
from sklearn.neighbors import KNeighborsClassifier

KNC = KNeighborsClassifier(n_neighbors=2)
KNC.fit(X_train, y_train)

KNC_pred = KNC.predict(X_test)
```

```
KNC_accuracy = metrics.accuracy_score(y_test, KNC_pred)

print("KNeighbourClassifier score: ", KNC_accuracy)
```

```
KNeighbourClassifier score:  0.75
```

```
from sklearn.ensemble import RandomForestClassifier

rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1)
rnd_clf.fit(X_train, y_train)

rnd_clf_pred = rnd_clf.predict(X_test)

rnd_clf_accuracy = metrics.accuracy_score(y_test, rnd_clf_pred)
print("RandomForest score: ", rnd_clf_accuracy)
```

```
RandomForest score:  0.8333333333333334
```