FACULTY OF INFORMATION TECHNOLOGY

ITADAA4-12

Decision Support System determining whether Patients have Heart Disease for Healthcare Professionals

STUDENT NAME:

NAME	SURNAME	STUDENT NUMBER
LINGTON	SKHOSANA	EDUV4931502@VOSSIE.NET

Lecture: Mr Mokgapi London Mashabela **Date: 24 July 2024**

Table of Contents

1. QUESTION 1

1.1 Creating database connection from csv file to SQLite.

#importing necessary libraries

import pandas as pd

import sqlite3

#connecting to the csv file to create a dataframe

df= pd.read_csv(r"C:\Users\skhosanal\OneDrive - Inkomati-Usuthu Catchment Management Agency\Python Scripts\HeartDeseaseProject\Heart.csv")

print("Dataframe created")

#Establishing a connection to sqlite database

conn=sqlite3.connect('Heart.db')

print("Connection established")

2. QUESTION 2

2.1 Data preprocessing

a) df.head(10)

jupyter DSS_Heart_Chronic_Patients Last Checkpoint: 4 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help   Trusted   Python 3 (ipykernel)

```
dataframe converted into a Table Database

In [39]: df.head(10)
Out[39]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

The data frame contains 303 rows and 14 columns when one runs the df. Shape function. The dataframe does not contain any null values through the df.isnull().sum() function.

jupyter DSS_Heart_Chronic_Patients Last Checkpoint: 3 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run | Code

```
9   57   1   2   150   168   0   1   174   0   16   2   0   2   1
```

In [40]:
```
df.shape
```

Out[40]:
```
(303, 14)
```

In [42]:
```
df.isnull().sum()
```

Out[42]:
```
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
```

df.info()

In [43]:
```
data_dup=df.duplicated().any
```

In [46]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

data_dup.duplicated().any

In [43]:
```
data_dup=df.duplicated().any
```

In [46]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
```

df.describe()

```
In [47]: df.describe()
```

Out[47]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.31
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.61
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.0
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.0
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.0
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.0

The dataset contains 303 records with a maximum age of seventy-seven (77) and a minimum age of twenty-nine (29), and an average age of fifty-four (54). Chest pain is more common in males than females. Males have high cholesterol being present and more common in the age of fifty -five and above. The datasets show males being prone to heart disease condition. Although the number of heart vessels colored with fluoroscopy is mild and normal, the status of their heart is reversible, and only 50% are fixed defects.

#Fetching the data from sqlite Heart_Disease_Patients Database.

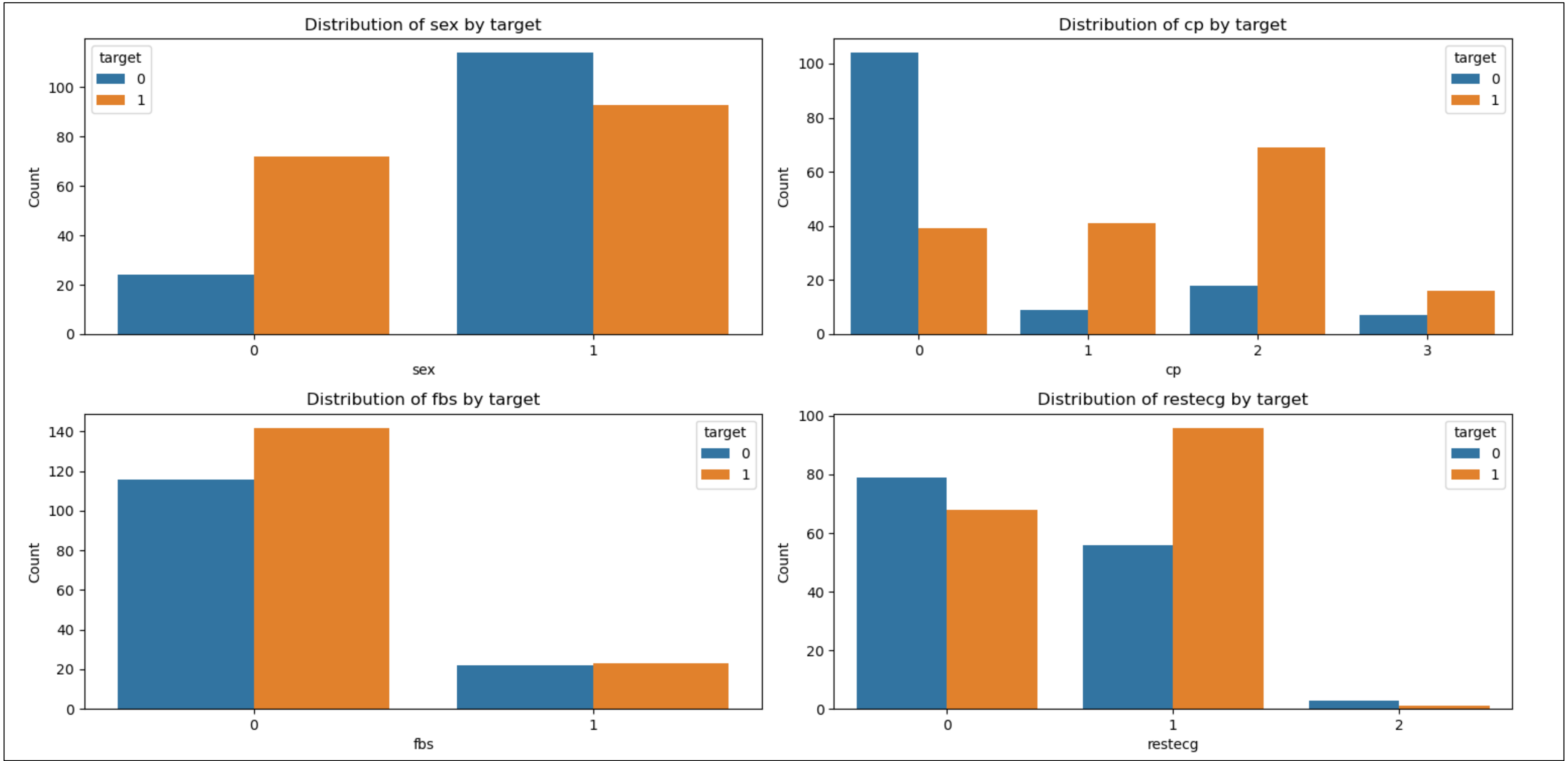
```
conn=sqlite3.connect('Heart.db')
cur=conn.cursor()
cur.execute("SELECT * FROM Heart")
rows = cur.fetchall()

for row in rows:
    print(row)

conn.commit()
conn.close()
```

b) Plotting of distribution of classes on the 8 categorical variables based on the target variables.

```
#setting up the matplotlib figure
%matplotlib inline
fig, axes=plt.subplots(nrows=4, ncols=2, figsize=(15, 15))
axes = axes.flatten()
categorical_vars=['sex','cp','fbs','restecg','exang','slope','ca','thal']
# Plot countplots for each categorical variable
for i, var in enumerate(categorical_vars):
    sns.countplot(x=var, hue='target', data=df, ax=axes[i])
    axes[i].set_title(f'Distribution of {var} by target')
    axes[i].set_xlabel(var)
    axes[i].set_ylabel('Count')
plt.tight_layout()
plt.show()
```



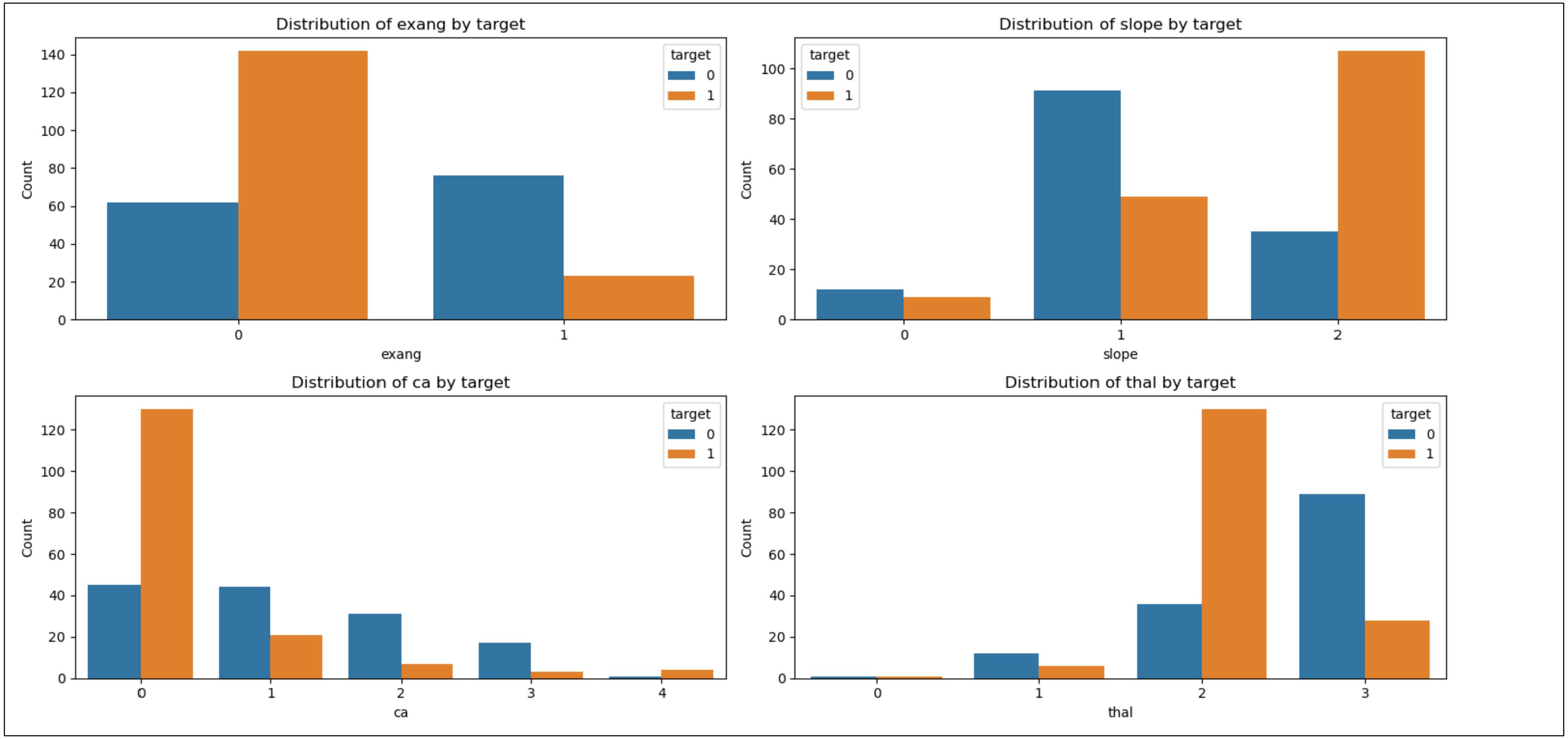
Observation

i. There are more male patients than female patients in the datasets, and males have a higher proportion of heart disease than females. However, in the category of females, more females have a heart condition than those who do not have.

ii. Patients who tested positive for atypical angina category one (1) and two (2) chest pain appear to have a relative proportion of heart disease whereas those in category zero (0) and three(3) appear to have a low proportion of heart disease.

iii. Patients who tested negative for fasting blood sugar less than (120mg/dl), shown to have a higher proportion of heart conditions, however, the graph indicates a lower risk of heart disease for those above (120mg/dl).

iv. There is a high number of patients in the abnormal electrocardiographic category one (1) with a high proportion of heart disease compared to those in category zero (0) and two (2) wherein the latter have a relatively low presence of the disease whereas those in category zero have a low presence compared to those who do not have in that category.



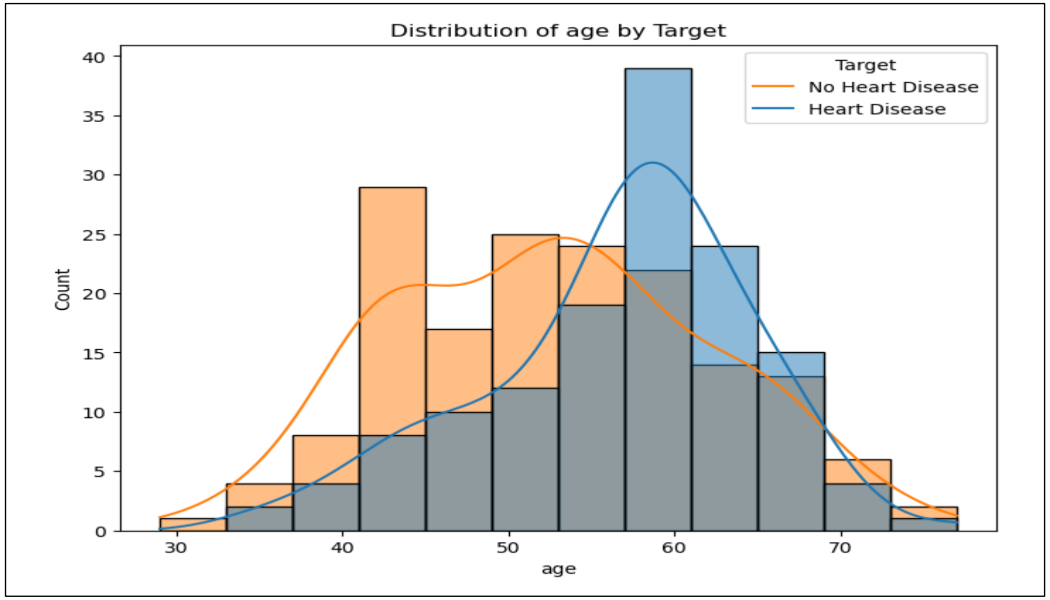
v. Patients without exercise-induced angina equal to zero(0) seem to have a higher proportion of heart disease compared to those with exercise-induced angina in category 1. Therefore, it means that those who performed the exercise are less prone to heart disease than those who didn't perform the exercise.

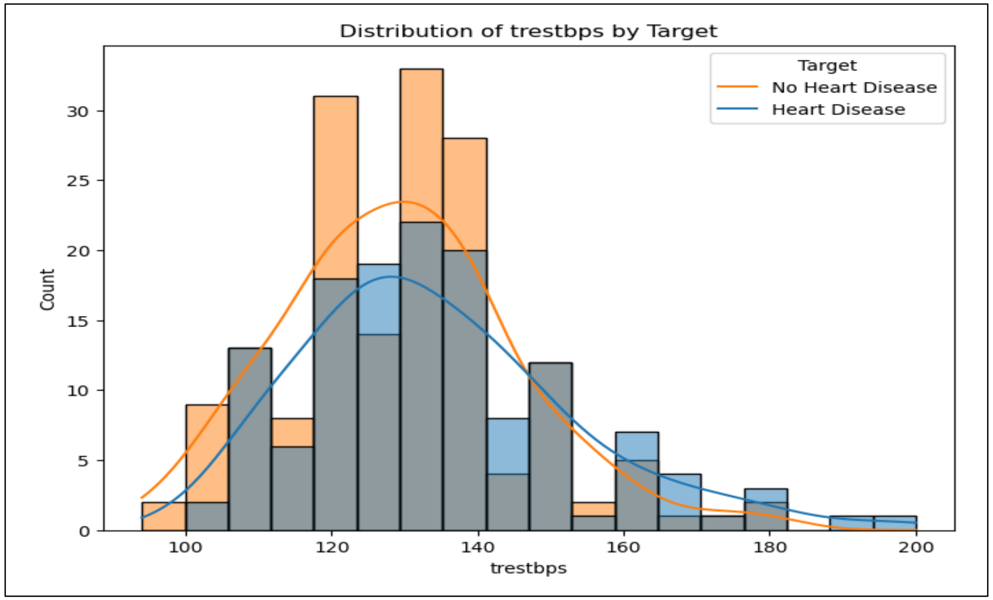
vi. Patients with a flat slope in category two (2) appear to have a relatively higher proportion of patients with heart disease than those in category zero(0) and category one (1).

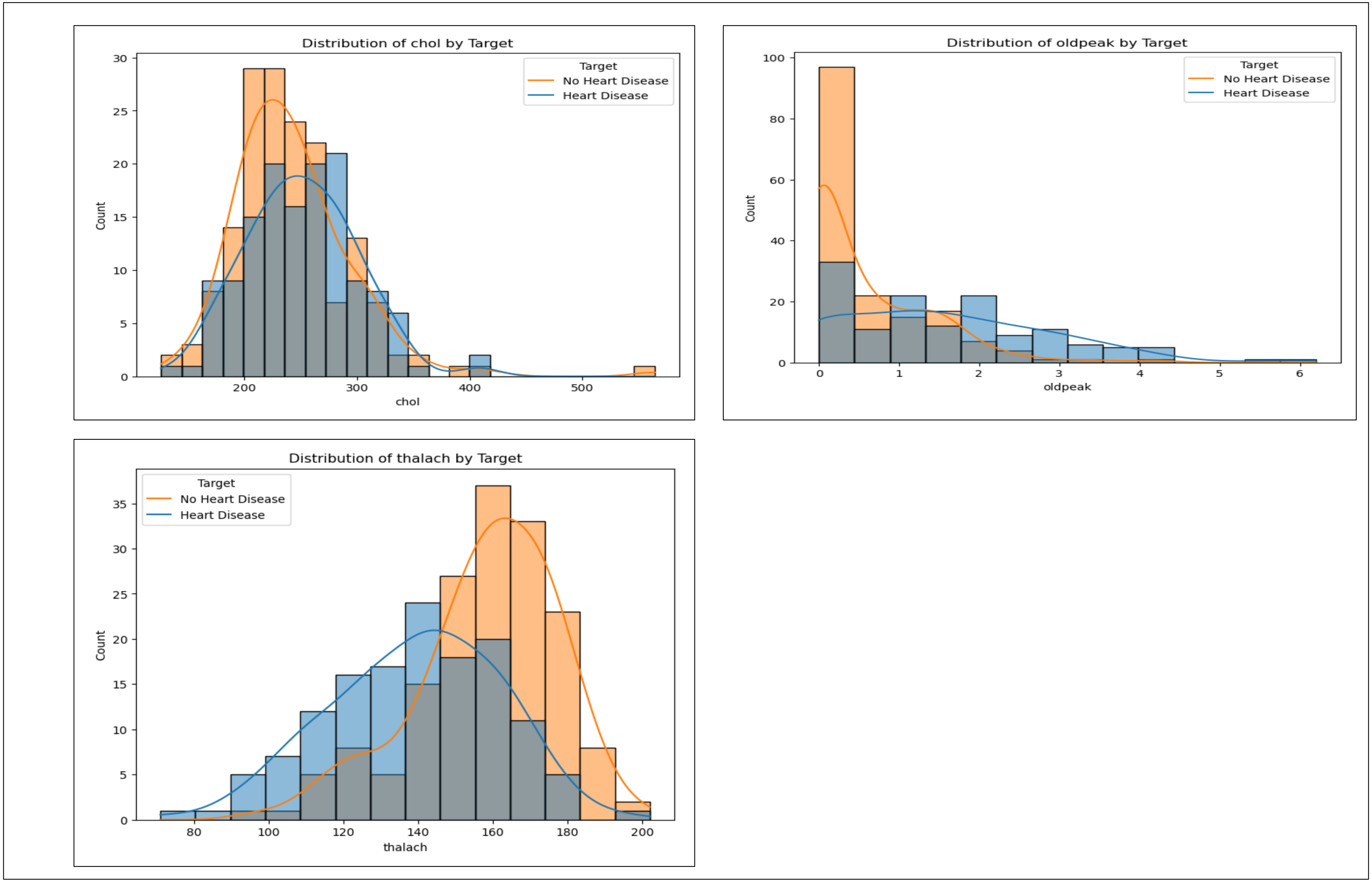
vii. Patients with major fluoroscopy in category zero(0) -mild appear to have a higher prevalence of patients with heart disease compared to those in other categories. Category four which is severe indicates the lowest number of patients with a low prevalence of the disease.

viii. Patients in category zero (0) of the status of the heart indicate a very low absence of the disease than those in category one (1) indicate a small population of those who suffer from a heart condition and those in category two (2) which is fixed-defect indicate a higher proportion of heart disease than those in other categories, and finally category three(3) reversible defect have relatively a low population of those who have the condition versus those who do not have in the same category.

c) Plot the distribution of classes for the numeric variables based on the target variable.






Distribution of chol by Target
Target
No Heart Disease
Heart Disease
Count
0
5
10
15
20
25
30
200
300
400
500
chol
Distribution of oldpeak by Target
Target
No Heart Disease
Heart Disease
Count
0
20
40
60
80
100
0
1
2
3
4
5
6
oldpeak
Distribution of thalach by Target
Target
No Heart Disease
Heart Disease
Count
0
5
10
15
20
25
30
35
80
100
120
140
160
180
200
thalach

Observations

i. There seems to be a higher frequency of heart disease in older patients from age 60 upwards and the trend decreases in the elderly generation.

ii. Those with a resting blood pressure of between 140-200 show a prevalence of succumbing to heart disease than those who are in the same range but don’t have the heart condition.

iii. Patients with high cholesterol constitute a higher number of those who don’t have a heart condition than those who have and have a heart condition.

iv. ST depression induced by exercise relative to rest constitutes a relatively low population of patients who suffer from the disease, than those who don’t however constitute a larger portion of healthy patients.

v. Patients with a maximum heart rate of 80-140 have a higher prevalence of heart disease than those who have a rate of 150-200 is dominated by patients who are free from a heart condition.

3. QUESTION 3

3.1 Modelling heart disease prediction problem through machine learning

 # Convert categorical variables into numerical format

cate_val = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']

cont_val =['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

```
jupyter DSS_Heart_Chronic_Patients Last Checkpoint: 3 minutes ago (autosaved)          Logout
File  Edit  View  Insert  Cell  Kernel  Widgets  Help          Trusted   Python 3 (ipykernel) O
Run  Code

In [53]: #Data Processing
         cate_val = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
         cont_val =['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

In [54]: cate_val

Out[54]: ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']

In [55]: cont_val

Out[55]: ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

In [56]: df['cp'].unique()

Out[56]: array([3, 2, 1, 0], dtype=int64)
```

 # Feature scaling

from sklearn.preprocessing import StandardScaler

st= StandardScaler()

df[cont_val]= st.fit_transform(df[cont_val])

```
In [58]: #feature Scaling
         from sklearn.preprocessing import StandardScaler

In [59]: st= StandardScaler()
         df[cont_val]= st.fit_transform(df[cont_val])

In [60]: df.head()

Out[60]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	0.952197	1	3	0.763956	-0.256334	1	0	0.015443	0	1.087338	0	0	1	1
1	-1.915313	1	2	-0.092738	0.072199	0	1	1.633471	0	2.122573	0	0	2	1
2	-1.474158	0	1	-0.092738	-0.816773	0	0	0.977514	0	0.310912	2	0	2	1
3	0.180175	1	1	-0.663867	-0.198357	0	1	1.239897	0	-0.206705	2	0	2	1

X = df.drop('target',axis=1)# dropping dependent variable

y = df['target']

from sklearn.model_selection import train_test_split

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

X_train, X_test, y_train,y_test #Testing the results

```
In [61]: #Splitting The dataset into Training Set and Test Set
         X = df.drop('target',axis=1)# dependent variable
         y = df['target']

In [29]: y

Out[29]: 0    1
         1    1
         2    0
         3    0
         4    0
         5    0
         6    0
         7    0
         8    0
         9    0
```

```
In [62]: from sklearn.model_selection import train_test_split

In [63]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

In [64]: y_test

Out[64]: 179    0
         228    0
         111    1
         246    0
         60     1
               ..
         249    0
         104    1
         300    0
         193    0
         184    0
         Name: target, Length: 61, dtype: int64
```

3.2.1 Logistic Regression

Explanation: Logistic Regression is a binary classification algorithm that models the probability of a binary outcome. It's a linear model that predicts the probability that a given instance belongs to a particular class.

- **Advantages:**
  - Simple and easy to interpret.
  - Low computational cost, making it efficient for large datasets.
- **Disadvantages:**
  - Assumes a linear relationship between features and the log-odds of the outcome.
  - May underperform when the relationship between features and target variable is non-linear.

3.2.2 Random Forest

- Explanation: Random Forest is an ensemble learning method that builds multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees.
- **Advantages:**
  - High accuracy and robustness against overfitting.
  - Handles non-linear relationships and interactions between features well.
- **Disadvantages:**
  - More complex than individual decision trees, making it harder to interpret.
  - Can be computationally expensive and slow to train on large datasets with many trees.

3.2.3 Support Vector Machine(SVM):

- Explanation: SVM is a powerful supervised learning algorithm used for classification and regression tasks. It finds the hyperplane that best separates the classes in the feature space.
- **Advantages:**
  - Effective in high-dimensional spaces and when the number of features is greater than the number of samples.
  - Versatile, as it can use different kernel functions to handle non-linear decision boundaries.
- **Disadvantages:**
  - Memory-intensive and computationally expensive, especially for large datasets.

- Can be sensitive to the choice of kernel parameters and regularization.

3.2.4 Determining the best model for predicting a heart disease

Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression()
log.fit(X_train,y_train)
y_pred1 = log.predict(X_test)
from sklearn.metrics import accuracy score
accuracy_score(y_test,y_pred1)
    accuracy_score: 0.8524590163934426
```

SVM (Support Vector Machine) Model

```
from sklearn import svm
svm = svm.SVC()
svm.fit(X_train,y_train)
y_pred2 = svm.predict(X_test)
accuracy_score(y_test,y_pred2)
    accuracy_score: 0.8688524590163934
```

Random Forest Classifier Model

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(X_train,y_train)
y_pred3 = rf.predict(X_test)
accuracy_score(y_test,y_pred3)
accuracy_score: accuracy_score: 0.8688524590163934
```

Model Comparison

```
final_data = pd.DataFrame({'Models': ['LR','SVM','RFC'],
                    'ACC': [accuracy_score(y_test,y_pred1),
                        accuracy_score(y_test,y_pred2),
                        accuracy_score(y_test,y_pred3)]})
```

```
In [76]: final_data = pd.DataFrame({'Models': ['LR','SVM','RFC'],
                                    'ACC': [accuracy_score(y_test,y_pred1),
                                            accuracy_score(y_test,y_pred2),
                                            accuracy_score(y_test,y_pred3)]})
```

```
In [77]: final_data
```

Out[77]:

	Models	ACC
0	LR	0.852459
1	SVM	0.868852
2	RFC	0.868852

NB: Based on the scaled data all three models are best fit to be deployed and run the web application for the prediction of heart disease, both SVM and RFC scored the same score of 87% however, am going to use a Random Forest Classifier for the prediction due to the reasons mentioned in its advantages:

- High accuracy and robustness against overfitting.
- Handles non-linear relationships and interactions between features well.

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(X,y)
```

Testing the model on new Data

```
new_data = pd.DataFrame({
    'age':52,
    'sex':1,
    'cp':0,
    'trestbps':125,
    'chol':212,
    'fbs':0,
    'restecg':1,
    'thalach':168,
    'exang':0,
    'oldpeak':1.0,
    'slope':2,
```

```
    'ca': 2,
    'thal':3,
}, index = [0])
```

```
In [ ]: from sklearn.ensemble import RandomForestClassifier

In [ ]: rf=RandomForestClassifier()
        rf.fit(X,y)

In [88]: new_data=pd.DataFrame({
             'age':52,
             'sex':1,
             'cp':0,
             'trestbps':125,
             'chol':212,
             'fbs':0,
             'restecg':1,
             'thalach':168,
             'exang':0,
             'oldpeak':1.8,
             'slope': 2,
             'ca': 2,
             'thal':3,
         },index=[0])

In [89]: new_data

Out[89]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	52	1	0	125	212	0	1	168	0	1.8	2	2	3

```
p=rf.predict(new_data)
if p[0]==0:
    print("Patient is safe, No heart disease.")
else:
    print("Patient is Likely to develop a heart disease!")
```

```
Output: Patient is safe No heart disease!
```

#SAVING THE DEVELOPED MODEL(RANDOM FOREST)

```
import joblib
joblib.dump(rf,'random_forest_model.joblib')
```

#LOADING THE MODEL AND PERFORMING PREDICTION (RANDOM FOREST)

```
model=joblib.load('random_forest_model.joblib')
model.predict(new_data)
```

QUESTION 4

4.1 Deployment of a model in a web application using Streamlit

See the Code in the GitHub account: lingtonskhosana@gmail.com

https://heart-disease-project-dcjsceil23aqgkocplnnxi.streamlit.app/

https://github.com/SkhosanaL/Heart-Disease-Project