OVERVIEW OF MACHINE LEARNING

Machine learning is employed for a variety of tasks including:

- **Image Classification**: Identifying objects within images.
- **Language Translation**: Converting text from one language to another.
- **Data Handling**: Managing and analyzing large volumes of sensor data.
- **Prediction**: Forecasting future values based on current data.

Various strategies and algorithms are utilized, each tailored to different types of data and problems.

SUPERVISED VS UNSUPERVISED LEARNING

- **Supervised Learning**: A model is trained using labeled data, where each input has a known output. The goal is to predict the output for new data based on patterns learned from labeled examples.
  Example: Labeling images of food to classify them into categories like pizza, burgers, etc.

- **Unsupervised Learning**: A model is trained with unlabeled data. The model identifies patterns and structures within the data on its own.
  Example: Providing images of various foods without labels and allowing the model to group similar items, such as separating images of pizza from other foods.

Understanding Algorithms

An algorithm in machine learning is a mathematical function customized to fit specific data, designed to solve problems efficiently and accurately.

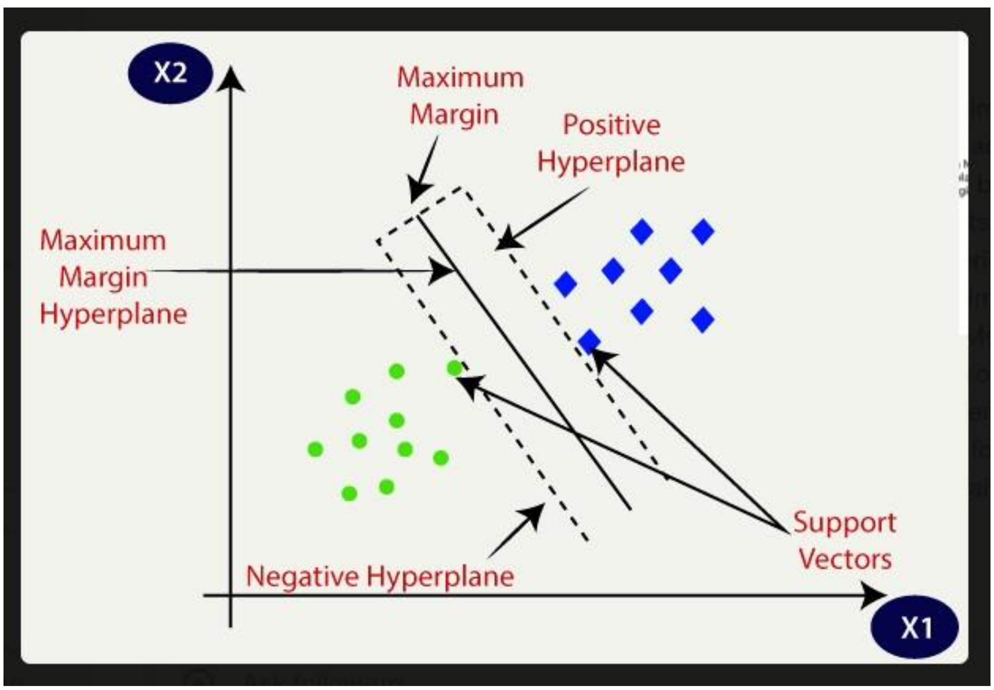
- **Support Vector Machines (SVMs)**: Supervised learning models used for classification, regression, and outlier detection, noted for their effectiveness in complex problems.
  - **Classification and Regression**: SVMs can classify data points into different categories or predict continuous values.
  - **Support Vector Regression (SVR)**: An extension of SVM for regression tasks.

How SVMs Work

➢ **Linear SVM**: Finds the best straight line (or hyperplane) that separates different classes of data with maximum margin.

- **Non-Linear SVM**: Uses kernel functions to transform data into higher dimensions where linear separation is possible.

- The Support Vector Machine (SVM) algorithm works by finding the optimal hyperplane that segregates data points of different classes in an n-dimensional space.
- This hyperplane is positioned to maximize the margin, which is the distance between thehyperplane and the closest data points from each class.
- The data points closest to the hyperplane are called support vectors and influence the position and orientation of the hyperplane.
- SVMs use kernel functions to transform data into higher-dimensional spaces, enabling thealgorithm to handle nonlinear classification problems.
- SVMs are effective for binary classification tasks and can be extended to handle multiclassproblems by combining binary classifiers.
- The SVM algorithm aims to identify a hyperplane that distinctly separates classes with the largest margin, making it suitable for various applications such as text classification, image classification, and anomaly detection.



Types of SVMs

1. **Simple SVM**: Used for linear classification and regression.
2. **Kernel SVM**: Handles non-linear data by using kernel functions to fit a hyperplane.

Pros and Cons of SVMs

Pros:

- Effective on high-dimensional datasets.
- Handles small to medium-sized datasets well.
- Memory efficient due to the use of support vectors.
- Customizable kernel functions.

Cons:

- Not ideal for very large datasets.
- Requires careful tuning to avoid overfitting.
- Doesn't provide probability estimates directly.

Hyperplanes

In SVM, a hyperplane is a decision boundary that separates different classes of data points. The number of dimensions of the hyperplane is one less than the number of features:

- **2D Space**: The hyperplane is a line.
- **3D Space**: The hyperplane is a plane.
- **Higher Dimensions**: Visualization is challenging, but the principle remains the same.

Margin

The margin is the distance between the hyperplane and the nearest data points from either class, known as support vectors. The objective is to maximize this margin, as a larger margin signifies better separation between classes, enhancing the model's ability to generalize to unseen data.

Cost Function and Gradient Updates

The SVM algorithm aims to maximize the margin between data points and the hyperplane. The hinge loss function is used to achieve this.

Hinge Loss Function

Defined as:

$$L(y,f(x))=\max(0,1-y\cdot f(x))$$

where y is the true label, and f(x) is the predicted value. The cost is zero if the prediction and actual label are correctly classified and otherwise depends on the margin.

Regularization

A regularization parameter is added to the cost function to balance margin maximization and loss. The regularized cost function is:

$$\text{Cost} = \text{Hinge Loss}+\lambda\|w\|^2$$

where λ is the regularization parameter.

Gradient Updates

Gradients are used to update the weights in the model:

- **No Misclassification**: Only the gradient from the regularization term is considered.
- **Misclassification**: The gradient includes both the loss and the regularization term.

ADVANTAGES of SVM	**DISADVANTAGES of SVM**
Productive in high-dimensional spaces.	SVM is not suitable for large datasets due to itscomputational intensity.
Effective when the number of dimensions exceedsthe number of specimens.	SVM underperforms when the number of properties for each data point exceeds the number of training data specimens.
Memory systematic, making it efficient in memoryusage	SVM does not perform well when the dataset hasmore noise, i.e., target classes are overlapping.
Works well with an understandable margin ofdissociation between classes. .	SVM does not provide probabilistic clarificationfor the classification.

QUESTION

You are given a dataset from Kaggle containing information about various medical conditions of patients. The dataset includes features such as age, sex, blood pressure, cholesterol levels, and whether the patient has a specific medical condition. Your task is to build a Support Vector Machine (SVM) model to predict the presence or absence of the medical condition based on the given features.

DATASET

You can use the **"Heart Disease UCI"** dataset available on Kaggle, which contains information about various medical attributes and the presence or absence of heart disease. You can access the dataset (https://www.kaggle.com/datasets/redwankarimsony/heart-disease-data)

About Dataset

This is a multivariate type of dataset which means providing or involving a variety of separate mathematical or statistical variables, multivariate numerical data analysis. It is composed of 14 attributes which are age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, oldpeak — ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, number of major vessels and Thalassemia. This database includes 76 attributes, but all published studies relate to the use of a subset of 14 of them. The Cleveland database is the only one used by ML researchers to date. One of the major tasks on this dataset is to predict based on the given attributes of a patient that whether that particular person has heart disease or not and other is the experimental task to diagnose and find out various insights from this dataset which could help in understanding the problem more.

Run All Code Draft Session (33m)

```python
# Import Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.preprocessing import StandardScaler,MinMaxScaler,LabelEncoder
from sklearn.impute import KNNImputer,SimpleImputer
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.model_selection import train_test_split,GridSearchCV,cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostin
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
import warnings
warnings.filterwarnings('ignore')
```

Notebook

Input

Add Input Upload

DATASETS

heart-disease-data

heart_disease_uci.csv

Output (56KB / 19.5GB)

/kaggle/working

Session options

Schedule a notebook to run

IMPORTING LIBRARIES

LOADING THE DATASET

```python
df=pd.read_csv('/kaggle/input/heart-disease-data/heart_disease_uci.csv')
df.head()
```

[8]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal	2
2	3	67	Male	Cleveland	asymptomatic	120.0	229.0	False	lv hypertrophy	129.0	True	2.6	flat	2.0	reversable defect	1
3	4	37	Male	Cleveland	non-anginal	130.0	250.0	False	normal	187.0	False	3.5	downsloping	0.0	normal	0
4	5	41	Female	Cleveland	atypical angina	130.0	204.0	False	lv hypertrophy	172.0	False	1.4	upsloping	0.0	normal	0

EXPLORATORY DATA ANALYSIS

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   id        920 non-null    int64  
 1   age       920 non-null    int64  
 2   sex       920 non-null    object 
 3   dataset   920 non-null    object 
 4   cp        920 non-null    object 
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object 
 8   restecg   918 non-null    object 
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object 
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object 
 13  ca        309 non-null    float64
 14  thal      434 non-null    object 
 15  num       920 non-null    int64  
dtypes: float64(5), int64(3), object(8)
memory usage: 115.1+ KB
```

[10]:
```
df.shape
```

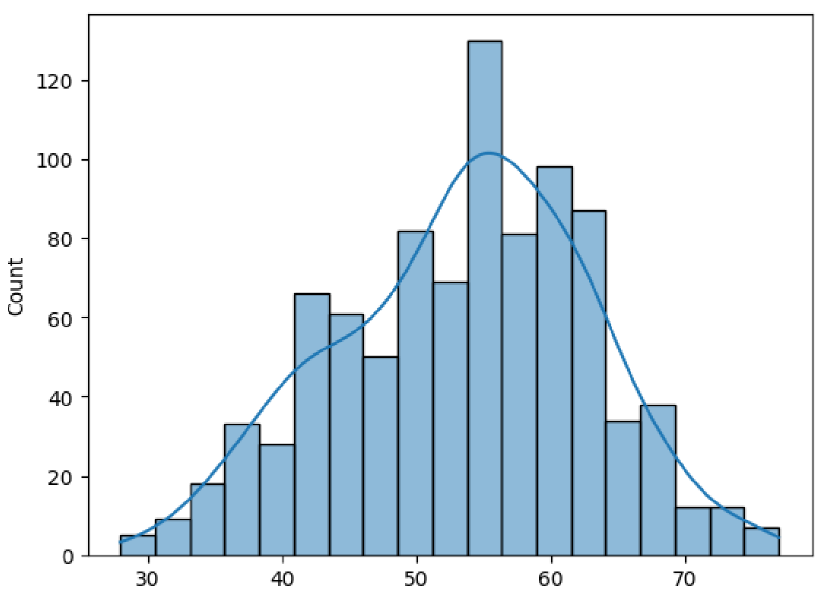
[10]: (920, 16)

```
#age column
df['age'].min(),df['age'].max()
```

[11]: (28, 77)

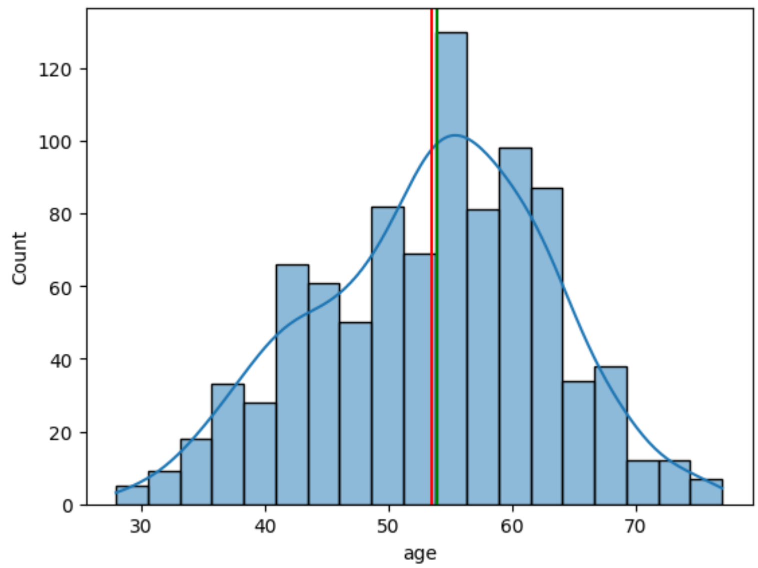
```
sns.histplot(df['age'],kde=True)
```

[12]: <Axes: xlabel='age', ylabel='Count'>



```python
sns.histplot(df['age'],kde=True)
plt.axvline(df['age'].mean(),color='red')
plt.axvline(df['age'].mode()[0],color='blue')
plt.axvline(df['age'].median(),color='green')
# print the values of mean, median, and mode
print("Mean :", df['age'].mean())
print('Mode :', df['age'].mode()[0])
print('Median :', df['age'].median())
```

```
Mean : 53.51086956521739
Mode : 54
Median : 54.0
```



OUTPUT

```python
#  Calculate the percentages of male and female in the data.Print how much percent males are more than female

male_count = df['sex'].value_counts()['Male']
female_count = df['sex'].value_counts()['Female']

total_count = male_count + female_count

male_percentage = (male_count / total_count) * 100
female_percentage = (female_count / total_count) * 100

male_more_than_female = ((male_count-female_count)/female_count)*100

print(f"Males: {male_percentage:.2f}%")
print(f"Females: {female_percentage:.2f}%")
print(f"Males are {male_more_than_female:.2f}% more than females")
```

```
Males: 78.91%
Females: 21.09%
Males are 274.23% more than females
```

[15]:
```python
# Lets deal with dataset column
df['dataset'].unique()
```

[15]:
```
array(['Cleveland', 'Hungary', 'Switzerland', 'VA Long Beach'],
      dtype=object)
```

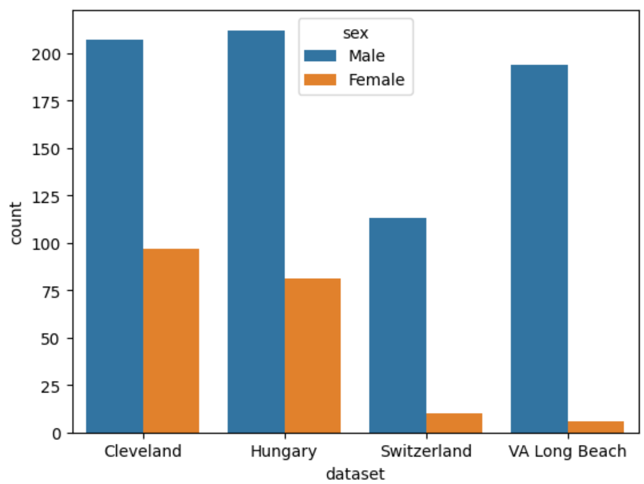
```python
df['dataset'].value_counts()
```

[16]:
```
dataset
Cleveland        304
Hungary          293
VA Long Beach    200
Switzerland      123
Name: count, dtype: int64
```

```python
#  print the countplot of dataset column

sns.countplot(data=df,x='dataset',hue='sex')
```

OUTPUT

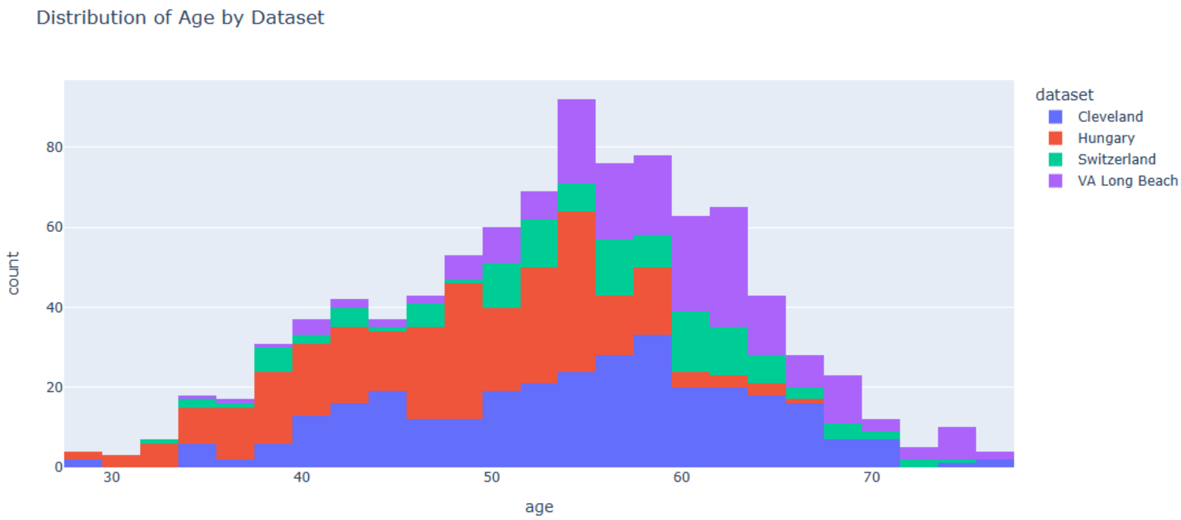
[17]: <Axes: xlabel='dataset', ylabel='count'>




```python
#  groupby sex column by dataset  column

df.groupby('sex')['dataset'].value_counts()
```

[18]:
```
sex     dataset
Female  Cleveland         97
        Hungary           81
        Switzerland       10
        VA Long Beach      6
Male    Hungary          212
        Cleveland        207
        VA Long Beach    194
        Switzerland      113
Name: count, dtype: int64
```

```python
#  Make a plot of age column using plotly amd color using dataset

fig = px.histogram(df, x="age", color="dataset",
             title="Distribution of Age by Dataset")
fig.show()
```




```python
#  groupby using dataset column Print the mean median mode of age column separately

for dataset in df['dataset'].unique():
  df_subset = df[df['dataset'] == dataset]
  print(f"Dataset: {dataset}")
  print(f"Mean age: {df_subset['age'].mean():.2f}")
  print(f"Median age: {df_subset['age'].median():.2f}")
  print(f"Mode age: {df_subset['age'].mode()[0]}")
  print()
```

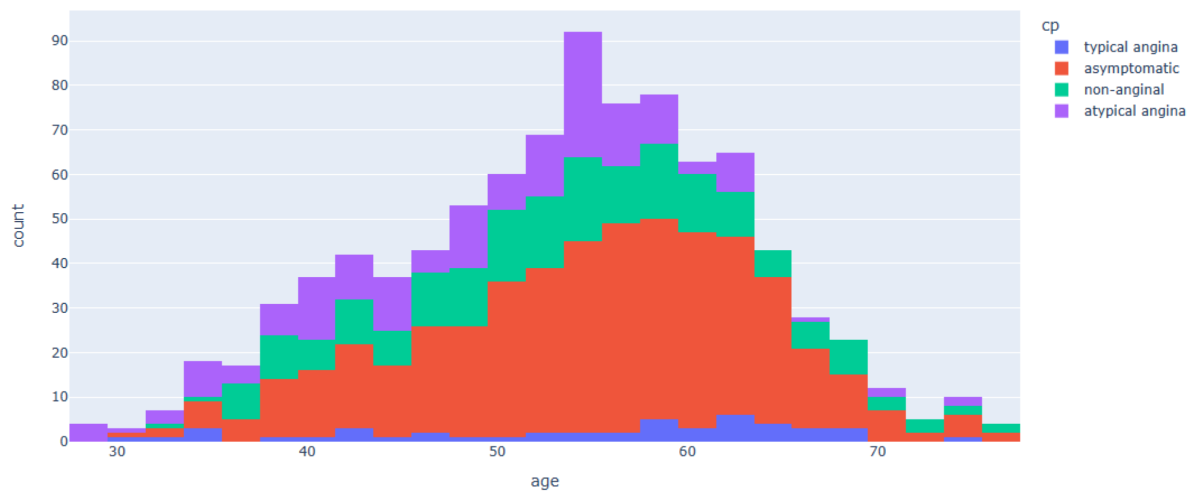
```
Dataset: Cleveland
Mean age: 54.35
Median age: 55.50
Mode age: 58

Dataset: Hungary
Mean age: 47.89
Median age: 49.00
Mode age: 54

Dataset: Switzerland
Mean age: 55.32
Median age: 56.00
Mode age: 61

Dataset: VA Long Beach
Mean age: 59.35
Median age: 60.00
Mode age: 62
```

[21]:
```python
# Now exploring Chest pain Column (cp)
df['cp'].value_counts()
```

[21]:
```
cp
asymptomatic       496
non-anginal        204
atypical angina    174
typical angina      46
Name: count, dtype: int64
```

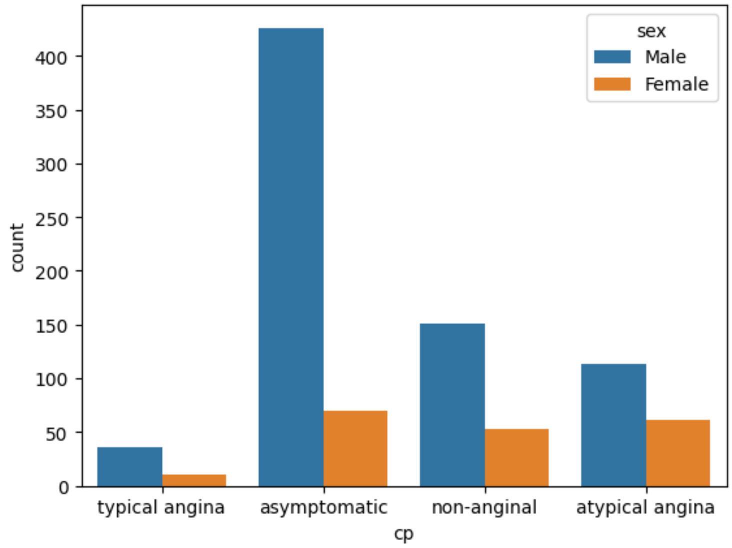
```python
#  draw the plot of age column in plotly and color using cp

fig = px.histogram(data_frame=df, x="age", color="cp")
fig.show()
```

OUTPUT



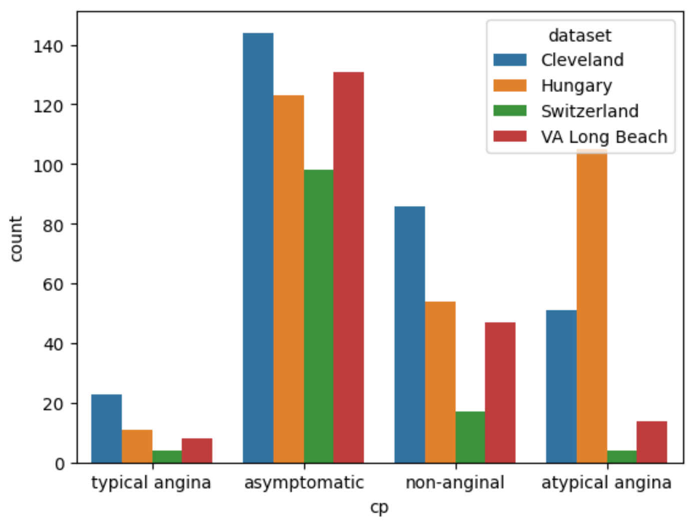

```python
# countplot of cp column using sex column

sns.countplot(data=df, x="cp", hue="sex")
plt.show()
```

OUTPUT




```
#  countplot of cp column using dataset column

sns.countplot(data=df, x="cp", hue="dataset")
plt.show()
```

OUTPUT

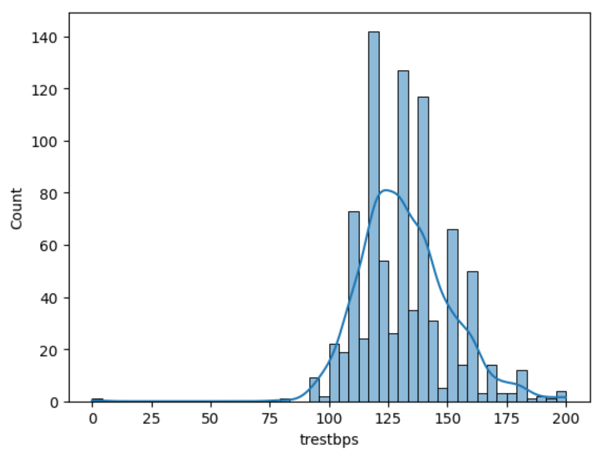


```python
# Exploring the trestbps resting blood pressure resting blood pressure in mm Hg on admission to the hospital
df['trestbps'].describe()
```

```
[25]: count    861.000000
mean     132.132404
std       19.066070
min        0.000000
25%      120.000000
50%      130.000000
75%      140.000000
max      200.000000
Name: trestbps, dtype: float64
```

```python
#] create a histplot of  trestbps

sns.histplot(data=df, x="trestbps", kde=True)
plt.show()
```

OUTPUT




```python
# prompt: percentage of missing values in trestbps column

missing_values = df['trestbps'].isnull().sum()
total_values = len(df['trestbps'])
missing_percentage = (missing_values / total_values) * 100
print(f"Percentage of missing values in trestbps column: {missing_percentage:.2f}%")
```

```
Percentage of missing values in trestbps column: 6.41%
```

```python
# prompt: Impute the missing values in trestbps column using iterative imputer and print missing values

from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

# Create an IterativeImputer object
imputer = IterativeImputer(max_iter=10)

# Impute the missing values in the trestbps column
df['trestbps'] = imputer.fit_transform(df['trestbps'].values.reshape(-1, 1))

# Print the number of missing values in the trestbps column
missing_values = df['trestbps'].isnull().sum()
print(f"Number of missing values in trestbps column after imputation: {missing_values}")
```

```
Number of missing values in trestbps column after imputation: 0
```

```python
#  Lets check missing values in other columns

# Check for missing values in other columns
missing_values_per_column = df.isnull().sum()

# Print the number of missing values in each column
for column, missing_count in missing_values_per_column.items():
  if missing_count > 0:
    print(f"Column: {column}, Missing Values: {missing_count}")
```

```
Column: chol, Missing Values: 30
Column: fbs, Missing Values: 90
Column: restecg, Missing Values: 2
Column: thalch, Missing Values: 55
Column: exang, Missing Values: 55
Column: oldpeak, Missing Values: 62
Column: slope, Missing Values: 309
Column: ca, Missing Values: 611
Column: thal, Missing Values: 486
```

[30]:

```python
missing_data_cols = df.isnull().sum()[df.isnull().sum() > 0].index.tolist()
```

+ Code + Markdown

```python
classifier_cols = ['thal', 'ca', 'slope', 'exang', 'restecg','fbs', 'cp', 'sex', 'num']
bool_cols = ['fbs', 'exang']
regressor_cols = ['oldpeak', 'thalch', 'chol', 'trestbps', 'age']
```

```python
for col in missing_data_cols:
    print("Missing Values", col, ":", str(round((df[col].isnull().sum() / len(df)) * 100, 2))+"%")
    if col in classifier_cols:
        df[col] = impute_categorical_missing_data(col)
    elif col in regressor_cols:
        df[col] = impute_continuous_missing_data(col)
    else:
        pass
```

OUTPUT

```
Missing Values chol : 3.26%
MAE = 44.46702247191012

Missing Values fbs : 9.78%
The feature 'fbs' has been imputed with 78.92 accuracy

Missing Values restecg : 0.22%
The feature 'restecg' has been imputed with 66.3 accuracy

Missing Values thalch : 5.98%
MAE = 16.963005780346682

Missing Values exang : 5.98%
The feature 'exang' has been imputed with 76.88 accuracy

Missing Values oldpeak : 6.74%
MAE = 0.5572848837209302

Missing Values slope : 33.59%
The feature 'slope' has been imputed with 68.29 accuracy

Missing Values ca : 66.41%
The feature 'ca' has been imputed with 66.13 accuracy

Missing Values thal : 52.83%
The feature 'thal' has been imputed with 71.26 accuracy
```

Missing Values Imputed Sucessfully

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

```
[38]: id          0
      age         0
      sex         0
      dataset     0
      cp          0
      trestbps    0
      chol        0
      fbs         0
      restecg     0
      thalch      0
      exang       0
      oldpeak     0
      slope       0
      ca          0
      thal        0
      num         0
      dtype: int64
```

```
df.head()
```

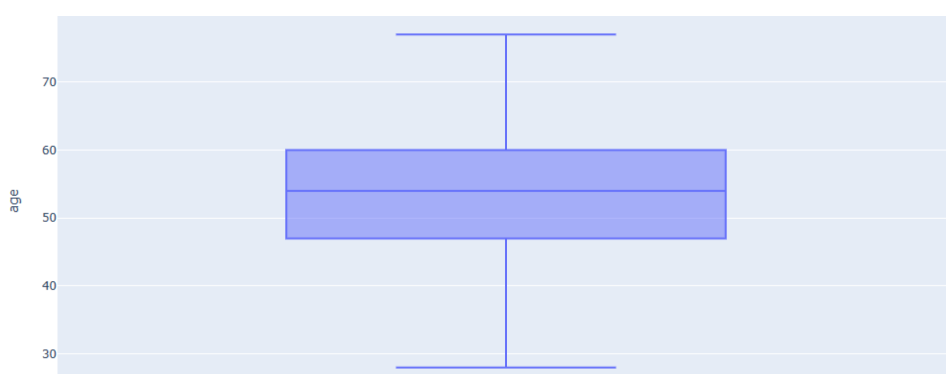
[40]:

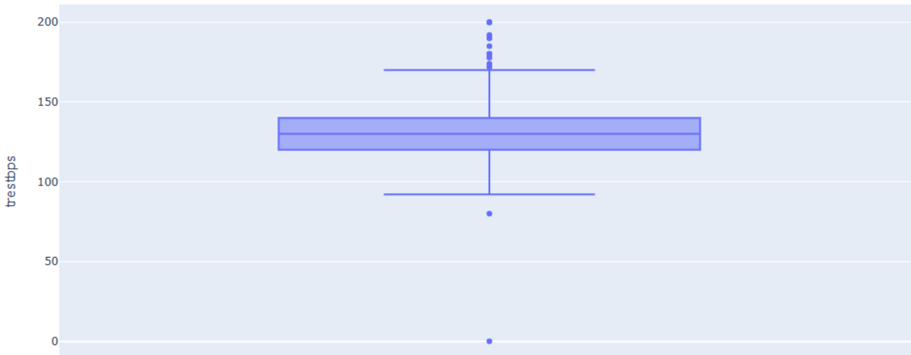
	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal	2
2	3	67	Male	Cleveland	asymptomatic	120.0	229.0	False	lv hypertrophy	129.0	True	2.6	flat	2.0	reversable defect	1
3	4	37	Male	Cleveland	non-anginal	130.0	250.0	False	normal	187.0	False	3.5	downsloping	0.0	normal	0
4	5	41	Female	Cleveland	atypical angina	130.0	204.0	False	lv hypertrophy	172.0	False	1.4	upsloping	0.0	normal	0

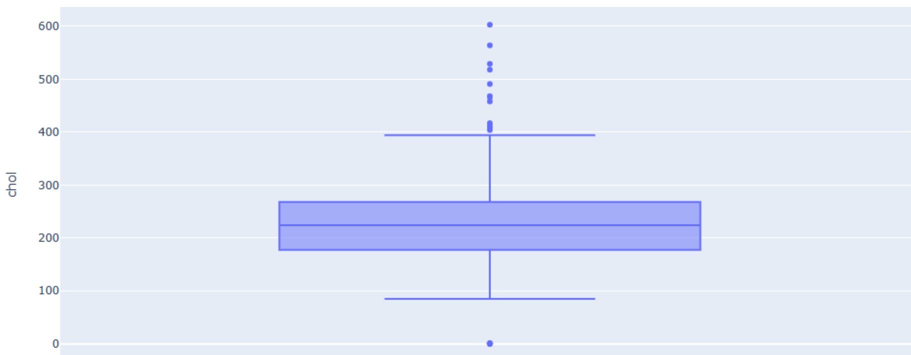
Dealing with OUTLIERS

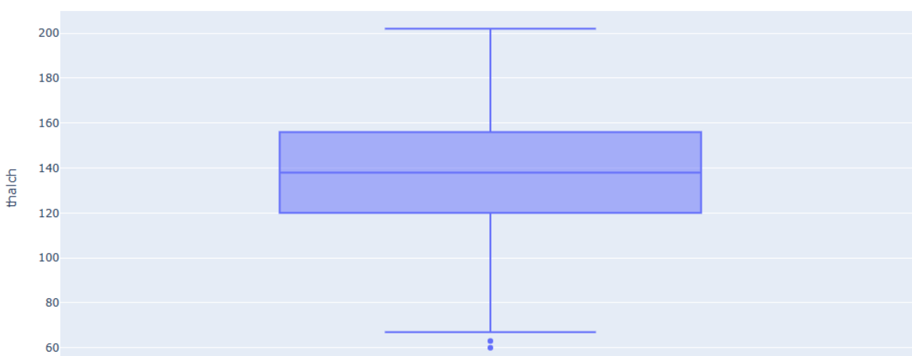
```
fig = px.box(data_frame=df, y='age')
fig.show()

fig = px.box(data_frame=df, y='trestbps')
fig.show()

fig = px.box(data_frame=df, y='chol')
fig.show()

fig = px.box(data_frame=df, y='thalch')
fig.show()

fig = px.box(data_frame=df, y='oldpeak')
fig.show()
```

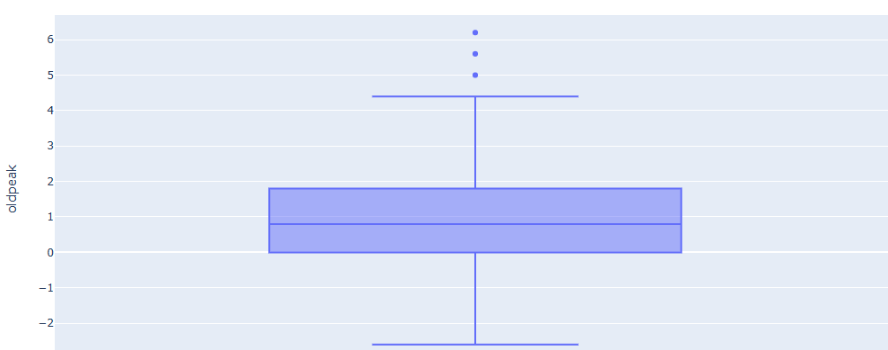
OUTPUT




200
150
100
50
0
trestbps


600
500
400
300
200
100
0
chol


200
180
160
140
120
100
80
60
thalch


6
5
4
3
2
1
0
−1
−2
−3
oldpeak

```python
# print the row with trestbp 0
df[df['trestbps']==0]
```

[42]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
753	754	55	Male	VA Long Beach	non-anginal	0.0	0.0	False	normal	155.0	False	1.5	flat	0.0	reversable defect	3

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 919 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        919 non-null    int64
 1   age       919 non-null    int64
 2   sex       919 non-null    object
 3   dataset   919 non-null    object
 4   cp        919 non-null    object
 5   trestbps  919 non-null    float64
 6   chol      919 non-null    float64
 7   fbs       919 non-null    object
 8   restecg   919 non-null    object
 9   thalch    919 non-null    float64
 10  exang     919 non-null    object
 11  oldpeak   919 non-null    float64
 12  slope     919 non-null    object
 13  ca        919 non-null    float64
 14  thal      919 non-null    object
 15  num       919 non-null    int64
dtypes: float64(5), int64(3), object(8)
memory usage: 122.1+ KB
```

```python
# prompt: print the row in df where chol is 0

df[df['chol']==0]
```

[48]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
597	598	32	Male	Switzerland	typical angina	95.000000	0.0	False	normal	127.00	False	0.700	upsloping	0.0	reversable defect	1
598	599	34	Male	Switzerland	asymptomatic	115.000000	0.0	False	normal	154.00	False	0.200	upsloping	0.0	reversable defect	1
599	600	35	Male	Switzerland	asymptomatic	132.132404	0.0	False	normal	130.00	True	1.510	flat	0.0	reversable defect	3
600	601	36	Male	Switzerland	asymptomatic	110.000000	0.0	False	normal	125.00	True	1.000	flat	0.0	fixed defect	1
601	602	38	Female	Switzerland	asymptomatic	105.000000	0.0	False	normal	166.00	False	2.800	upsloping	0.0	normal	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
818	819	43	Male	VA Long Beach	asymptomatic	122.000000	0.0	False	normal	120.00	False	0.500	upsloping	0.0	reversable defect	1
819	820	63	Male	VA Long Beach	non-anginal	130.000000	0.0	True	st-t abnormality	160.00	False	3.000	flat	0.0	reversable defect	0
822	823	48	Male	VA Long Beach	non-anginal	102.000000	0.0	False	st-t abnormality	110.00	True	1.000	downsloping	0.0	reversable defect	1
839	840	56	Male	VA Long Beach	asymptomatic	132.132404	0.0	False	lv hypertrophy	118.56	True	1.645	downsloping	0.0	reversable defect	1
840	841	62	Male	VA Long Beach	non-anginal	132.132404	0.0	True	st-t abnormality	118.62	False	2.144	downsloping	2.0	reversable defect	2

171 rows × 16 columns

Applying Machine Learning For Prediction

[49]:

```
#  Split the data with target label "num"

X = df.drop('num', axis=1)
y = df['num']
```

```
# prompt: Encode X data using label encoder for all categorical and object columns

le = LabelEncoder()
for col in X.select_dtypes(include=['object', 'category']):
  X[col] = le.fit_transform(X[col])
```

[51]:

```
#  Train test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

```
#  import the best multi classification models

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```python
# prompt: Iterate over the models and evaluate their performance using cross validation.Save the best models

models = [
    LogisticRegression(),
    KNeighborsClassifier(),
    SVC(),
    XGBClassifier(),
    RandomForestClassifier(),
    AdaBoostClassifier(),
    GradientBoostingClassifier(),
     GaussianNB(),
    DecisionTreeClassifier()
]

for model in models:
    model.fit(X_train, y_train)
    scores = cross_val_score(model, X_train, y_train, cv=10, scoring='accuracy')
    print(f"Model: {model.__class__.__name__}, CV Mean Accuracy: {scores.mean():.2f}")

best_model = max(models, key=lambda model: cross_val_score(model, X_train, y_train, cv=10, scoring='accuracy').mean())

print(f"Best Model: {best_model.__class__.__name__}")
```

OUTPUT

```
Model: LogisticRegression, CV Mean Accuracy: 0.54
Model: KNeighborsClassifier, CV Mean Accuracy: 0.57
Model: SVC, CV Mean Accuracy: 0.59
Model: XGBClassifier, CV Mean Accuracy: 0.66
Model: RandomForestClassifier, CV Mean Accuracy: 0.67
Model: AdaBoostClassifier, CV Mean Accuracy: 0.59
Model: GradientBoostingClassifier, CV Mean Accuracy: 0.67
Model: GaussianNB, CV Mean Accuracy: 0.60
Model: DecisionTreeClassifier, CV Mean Accuracy: 0.63
Best Model: RandomForestClassifier
```

The Best performing model is Random Forest Classifier

```python
# Save the random forest classifier model

import pickle

# Save the model
with open('heart_disease_model.pkl', 'wb') as file:
    pickle.dump(best_model, file)
```

Comprehensive Report About The Insights

Age Distribution

- There are 920 data points, and the average age is approximately **53.5** years, with an average difference of **9.4** years.
- The youngest person is **28** years old, and the oldest is **77** years.
- The majority of ages fall between **47** to **60** years.

Gender Distribution

- Approximately **77.77%** of individuals with heart disease in the dataset are males.
- Approximately **22.22%** of individuals with heart disease in the dataset are females.

Age and Heart Disease

- The age range **54-55** has the highest occurrence of heart disease.

Regional Analysis

"In the dataset, Hungary has the highest heart disease prevalence at 32.7%, followed by Cleveland (31.7%), VA Long Beach (22.4%), and Switzerland (13.2%), considering both males and females."

FEMALE	**MALE**
Switzerland: **10**	Switzerland: **113**
Cleveland: **97**	Cleveland: **207**
Hungary: **81**	Hungary: **212**
VA Long Beach: **6**	VA Long Beach: **194**

Age Statistics

MEAN AGE	MEDIAN AGE	MODE OF AGE
Switzerland: 55.32	Switzerland: 56.0	Switzerland: **61**
Cleveland: 54.35	Cleveland: 55.5	Cleveland: **58**
Hungary: 477.89	Hungary: 49.0	Hungary: **54**
VA Long Beach: 59.35	VA Long Beach: 60.0	VA Long Beach: **62, 63**

Chest Pain Types

- Asymptomatic: VA Long Beach (129), Cleveland, Hungary, Switzerland.
- Atypical Angina: Hungary (104), Cleveland, VA Long Beach, Switzerland.
- Non-Anginal: Cleveland (80), Hungary, VA Long Beach, Switzerland.
- Typical Angina: Cleveland (22), Hungary, VA Long Beach, Switzerland.

Chest Pain by Gender

- Asymptomatic more in males (404) than females (64).
- Atypical Angina more in males (110) than females (61).
- Non-Anginal more in males (143) than females (52).
- Typical Angina more in males (35) than females (10).

Additional Information

- Unique values in different columns:
- **fbs**: True False
- **restecg**: 'lv hypertrophy' 'normal' 'st-t abnormality'
- **exang**: False True
- **slope**: 'downsloping' 'flat' 'upsloping'
- **thal**: 'fixed defect' 'reversible defect' 'normal'
- Mean values for various health indicators:
- **chol**: 243.26
- **trestbps**: 131.90
- **thalach**: 137.07
- **oldpeak**: 0.93
- **ca**: 0.41

EXPLANATION

Task 1: Load and preprocess the dataset

The dataset is loaded into a Data Frame, missing values are handled by filling them with the mean of therespective columns, and the index is reset.

Task 2: Exploratory data analysis (EDA)

The dataset is explored to understand the distribution of features, and relationships between differentfeatures and the target variable are visualized using count plots and heatmaps.

Task 3: Feature engineering

Categorical variables are encoded, and numerical features are scaled using StandardScaler.

Task 4: Model selection and training

The dataset is split into training and testing sets, and the SVM model is trained using the radial basisfunction kernel.

Task 5: Model evaluation

The performance of the SVM model is evaluated using accuracy, precision, recall, and F1-score. Aconfusion matrix is visualized to understand the model's performance in detail.

Task 6: Hyperparameter tuning

The hyperparameters of the SVM model are fine-tuned using grid search, and the model is trained with thebest hyperparameters.

Task 7: Conclusion

The best hyperparameters are summarized, and the performance of different kernel functions and hyperparameter configurations is compared. Insights are provided into the features that have the mostsignificant impact on predicting the presence or absence of heart disease.

Overall, the given dataset is loaded into a Data Frame, preprocessed, and explored using EDA. Relevant features are extracted, categorical variables are encoded, and numerical features are scaled. The SVM modelis trained using the radial basis function kernel, and hyperparameters are fine-tuned using grid search. The performance of the SVM model is evaluated, and insights are provided into the features that have the most significant impact on predicting the presence or absence of heart disease.