```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

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

```
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
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	

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

	0
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.shape
```

```
(1025, 14)
```

```
df.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
df.dtypes
```

	0
age	int64
sex	int64
cp	int64
trestbps	int64
chol	int64
fbs	int64
restecg	int64
thalach	int64
exang	int64
oldpeak	float64
slope	int64
ca	int64
thal	int64
target	int64

dtype: object

```
# Unique values in each column
for i in df.columns:
    print(i, df[i].unique())
```

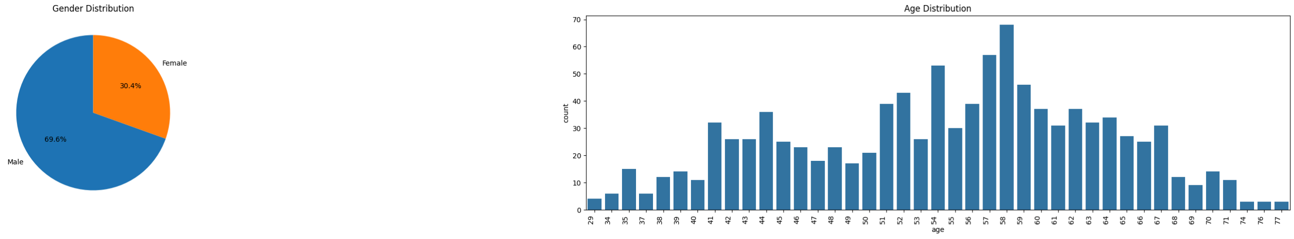
```
age [52 53 70 61 62 58 55 46 54 71 43 34 51 50 60 67 45 63 42 44 56 57 59 64
 65 41 66 38 49 48 29 37 47 68 76 40 39 77 69 35 74]
sex [1 0]
cp [0 1 2 3]
trestbps [125 140 145 148 138 100 114 160 120 122 112 132 118 128 124 106 104 135
 130 136 180 129 150 178 146 117 152 154 170 134 174 144 108 123 110 142
 126 192 115  94 200 165 102 105 155 172 164 156 101]
chol [212 203 174 294 248 318 289 249 286 149 341 210 298 204 308 266 244 211
 185 223 208 252 209 307 233 319 256 327 169 131 269 196 231 213 271 263
 229 360 258 330 342 226 228 278 230 283 241 175 188 217 193 245 232 299
 288 197 315 215 164 326 207 177 257 255 187 201 220 268 267 236 303 282
 126 309 186 275 281 206 335 218 254 295 417 260 240 302 192 225 325 235
 274 234 182 167 172 321 300 199 564 157 304 222 184 354 160 247 239 246
 409 293 180 250 221 200 227 243 311 261 242 205 306 219 353 198 394 183
 237 224 265 313 340 259 270 216 264 276 322 214 273 253 176 284 305 168
 407 290 277 262 195 166 178 141]
fbs [0 1]
restecg [1 0 2]
thalach [168 155 125 161 106 122 140 145 144 116 136 192 156 142 109 162 165 148
 172 173 146 179 152 117 115 112 163 147 182 105 150 151 169 166 178 132
 160 123 139 111 180 164 202 157 159 170 138 175 158 126 143 141 167  95
 190 118 103 181 108 177 134 120 171 149 154 153  88 174 114 195 133  96
 124 131 185 194 128 127 186 184 188 130  71 137  99 121 187  97  90 129
 113]
exang [0 1]
oldpeak [1.  3.1 2.6 0.  1.9 4.4 0.8 3.2 1.6 3.  0.7 4.2 1.5 2.2 1.1 0.3 0.4 0.6
 3.4 2.8 1.2 2.9 3.6 1.4 0.2 2.  5.6 0.9 1.8 6.2 4.  2.5 0.5 0.1 2.1 2.4
 3.8 2.3 1.3 3.5]
slope [2 0 1]
ca [2 0 1 3 4]
thal [3 2 1 0]
target [0 1]
```

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000

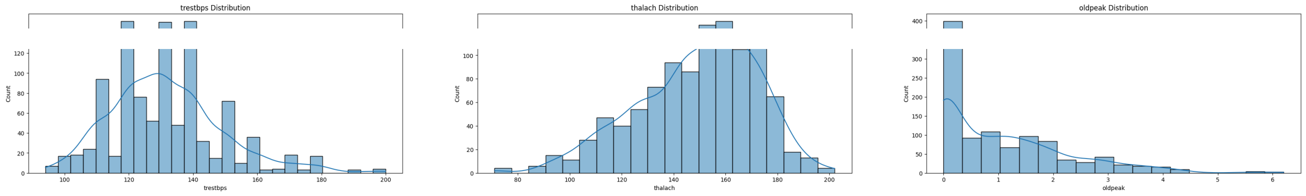
```
fig, ax = plt.subplots(1,2,figsize=(40, 5))
ax[0].pie(df['sex'].value_counts(), labels = ['Male', 'Female'], autopct='%1.1f%%', start
ax[0].set_title('Gender Distribution')
sns.countplot(x = 'age', data = df, ax = ax[1]).set_title('Age Distribution')
ax[1].set_xticklabels(ax[1].get_xticklabels(), rotation=90, ha='right')
```

```
<ipython-input-16-8f43a2313710>:5: UserWarning: FixedFormatter should only be used tc
  ax[1].set_xticklabels(ax[1].get_xticklabels(), rotation=90, ha='right')
[Text(0, 0, '29'),
 Text(1, 0, '34'),
 Text(2, 0, '35'),
 Text(3, 0, '37'),
 Text(4, 0, '38'),
 Text(5, 0, '39'),
 Text(6, 0, '40'),
 Text(7, 0, '41'),
 Text(8, 0, '42'),
 Text(9, 0, '43'),
 Text(10, 0, '44'),
 Text(11, 0, '45'),
 Text(12, 0, '46'),
 Text(13, 0, '47'),
 Text(14, 0, '48'),
 Text(15, 0, '49'),
 Text(16, 0, '50'),
 Text(17, 0, '51'),
 Text(18, 0, '52'),
 Text(19, 0, '53'),
 Text(20, 0, '54'),
 Text(21, 0, '55'),
 Text(22, 0, '56'),
 Text(23, 0, '57'),
 Text(24, 0, '58'),
 Text(25, 0, '59'),
 Text(26, 0, '60'),
 Text(27, 0, '61'),
 Text(28, 0, '62'),
 Text(29, 0, '63'),
 Text(30, 0, '64'),
 Text(31, 0, '65'),
 Text(32, 0, '66'),
 Text(33, 0, '67'),
 Text(34, 0, '68'),
 Text(35, 0, '69'),
 Text(36, 0, '70'),
 Text(37, 0, '71'),
 Text(38, 0, '74'),
 Text(39, 0, '76'),
 Text(40, 0, '77')]
```



```
fig, ax = plt.subplots(1,3,figsize=(40, 5))
sns.histplot(x = 'trestbps', data = df, ax = ax[0], kde = True).set_title('trestbps Distr
sns.histplot(x = 'thalach', data = df, ax = ax[1], kde = True).set_title('thalach Distrib
sns.histplot(x = 'oldpeak', data = df, ax = ax[2], kde = True).set_title('oldpeak Distrib
```

```
Text(0.5, 1.0, 'oldpeak Distribution')
```



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.drop(columns = ['target']), df['ta
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.tree import DecisionTreeClassifier
# Create Decision Tree object
dtc = DecisionTreeClassifier(random_state=0, max_depth= 12, min_samples_leaf=2, min_sampl
```

```
# Training the model
dtc.fit(X_train, y_train)
```

```
▾                       DecisionTreeClassifier                    i ?

DecisionTreeClassifier(class_weight='balanced', max_depth=12,
                       min_samples_leaf=2, random_state=0)
```

```
# Training accuracy
dtc.score(X_train, y_train)*100
```

```
99.7560975609756
```

```
# Predicting the test set results
dtc_pred = dtc.predict(X_test)
```

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
```

```
#Training the model
lr.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: Conver
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
▾ LogisticRegression ⓘ ?
LogisticRegression()
```

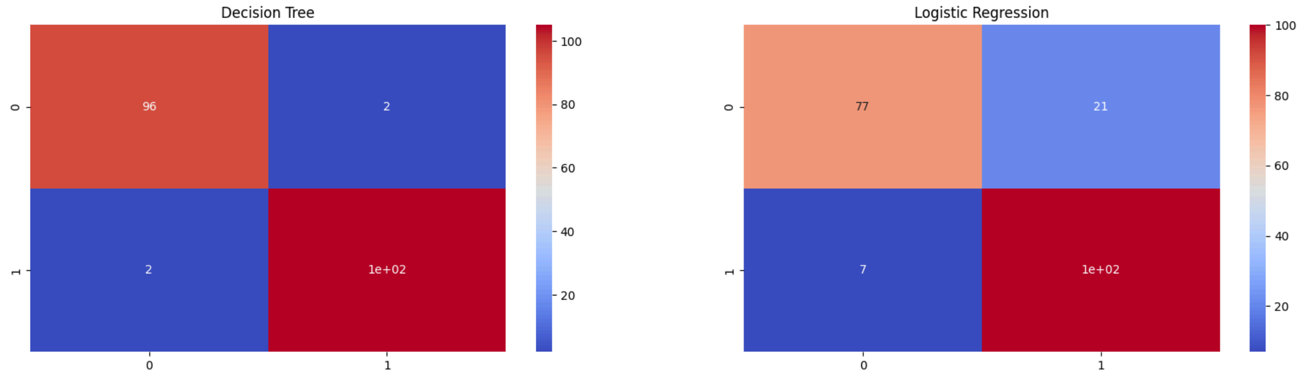
```
#Training accuracy
lr.score(X_train, y_train)*100
```

```
86.21951219512195
```

```
#Predicting the test set results
lr_pred = lr.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
fig, ax  = plt.subplots(1,2, figsize = (20,5))
sns.heatmap(confusion_matrix(y_test, dtc_pred), annot = True, cmap = 'coolwarm', ax = ax[
sns.heatmap(confusion_matrix(y_test, lr_pred), annot = True, cmap = 'coolwarm', ax = ax[1
```

```
Text(0.5, 1.0, 'Logistic Regression')
```



Decision Tree

Logistic Regression

```
print('Decision Tree')
print('Accuracy Score: ', accuracy_score(y_test, dtc_pred)*100)
print('Precision Score: ', precision_score(y_test, dtc_pred)*100)
print('Recall Score: ', recall_score(y_test, dtc_pred)*100)
print('F1 Score: ', f1_score(y_test, dtc_pred)*100)
```

```
Decision Tree
Accuracy Score:  98.04878048780488
Precision Score:  98.13084112149532
Recall Score:  98.13084112149532
F1 Score:  98.13084112149532
```

```
print('Logistic Regression')
print('Accuracy Score: ', accuracy_score(y_test, lr_pred)*100)
print('Precision Score: ', precision_score(y_test, lr_pred)*100)
print('Recall Score: ', recall_score(y_test, lr_pred)*100)
print('F1 Score: ', f1_score(y_test, lr_pred)*100)
```

```
Logistic Regression
Accuracy Score:  86.34146341463415
Precision Score:  82.64462809917356
Recall Score:  93.45794392523365
F1 Score:  87.71929824561403
```

Start coding or generate with AI.