In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:

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

In [4]:

```
df.head()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

In [5]:

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

Out[5]:

```
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
```

In [6]:

```
for feature in df.columns:
    print(f"{feature} unique values are: {df[feature].unique()}")
```

```
age unique values are: [52 53 70 61 62 58 55 46 54 71 43 34 51 50 60 67 45 6
3 42 44 56 57 59 64
 65 41 66 38 49 48 29 37 47 68 76 40 39 77 69 35 74]
sex unique values are: [1 0]
cp unique values are: [0 1 2 3]
trestbps unique values are: [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 unique values are: [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 unique values are: [0 1]
restecg unique values are: [1 0 2]
thalach unique values are: [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 unique values are: [0 1]
oldpeak unique values are: [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 unique values are: [2 0 1]
ca unique values are: [2 0 1 3 4]
thal unique values are: [3 2 1 0]
target unique values are: [0 1]
```

Numerical Features

In [7]:

```
numerical_feature = [feature for feature in df.columns if df[feature].dtype != 'O']
print(f"Total number of numerical features are: {len(numerical_feature)}")
print(numerical_feature)
```

```
Total number of numerical features are: 14
['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exan
g', 'oldpeak', 'slope', 'ca', 'thal', 'target']
```

Discrete features

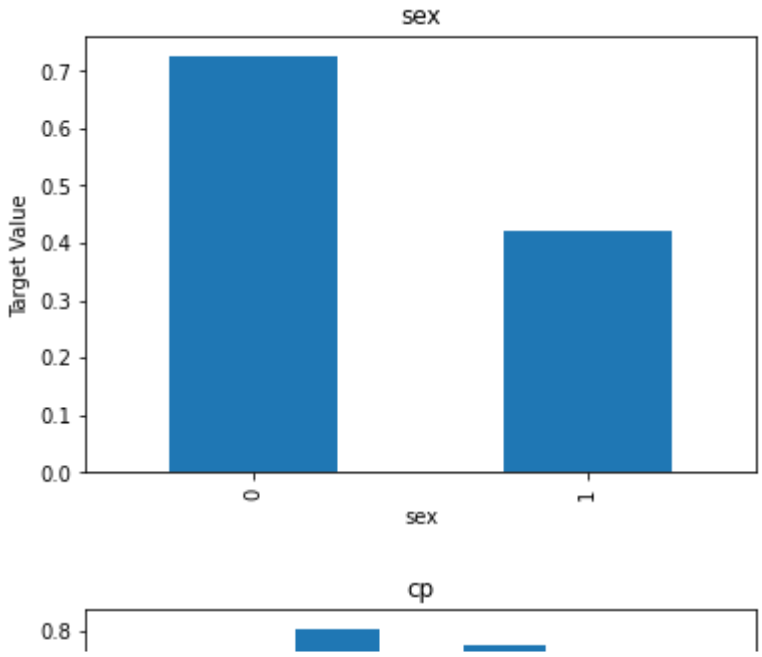
In [10]:

```
discrete_feature = [feature for feature in numerical_feature if len(df[feature].unique()) <
print(f"The total number of discrete features are {len(discrete_feature)}")
print(discrete_feature)
```

```
The total number of discrete features are 9
['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
```

In [13]:

```
for feature in discrete_feature:
    data = df.copy()
    data.groupby(feature)['target'].mean().plot.bar()
    plt.xlabel(feature)
    plt.title(feature)
    plt.ylabel('Target Value')
    plt.show()
```



In [14]:

```
df['target'].unique()
```

Out[14]:

```
array([0, 1], dtype=int64)
```

In [22]:

```
filterTarget = df['target'] == 1
```

In [23]:

```
df[filterTarget]
```

Out[23]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tai
5	58	0	0	100	248	0	0	122	0	1.0	1	0	2	
10	71	0	0	112	149	0	1	125	0	1.6	1	0	2	
12	34	0	1	118	210	0	1	192	0	0.7	2	0	2	
15	34	0	1	118	210	0	1	192	0	0.7	2	0	2	
16	51	0	2	140	308	0	0	142	0	1.5	2	1	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1011	45	1	1	128	308	0	0	170	0	0.0	2	0	2	
1014	44	0	2	108	141	0	1	175	0	0.6	1	0	2	
1019	47	1	0	112	204	0	1	143	0	0.1	2	0	2	
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	

526 rows × 14 columns

In [24]:

```
filterNotTarget = df['target'] == 0
```

In [25]:

```
df[filterNotTarget]
```

Out[25]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1017	53	1	0	123	282	0	1	95	1	2.0	1	2	3	
1018	41	1	0	110	172	0	0	158	0	0.0	2	0	3	
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	

499 rows × 14 columns

Continous Feature

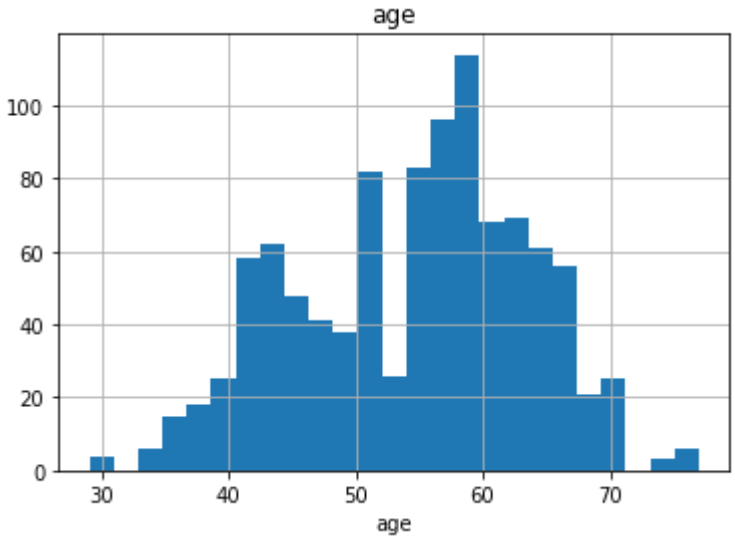
In [26]:

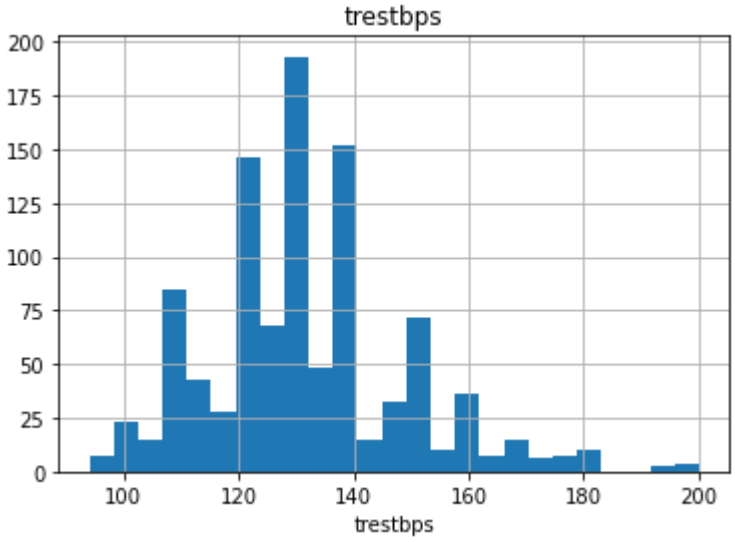
```
continous_feature = [feature for feature in df.columns if feature not in discrete_feature]

print(f"Total number of continous features are: {len(continous_feature)}")
print(continous_feature)
```

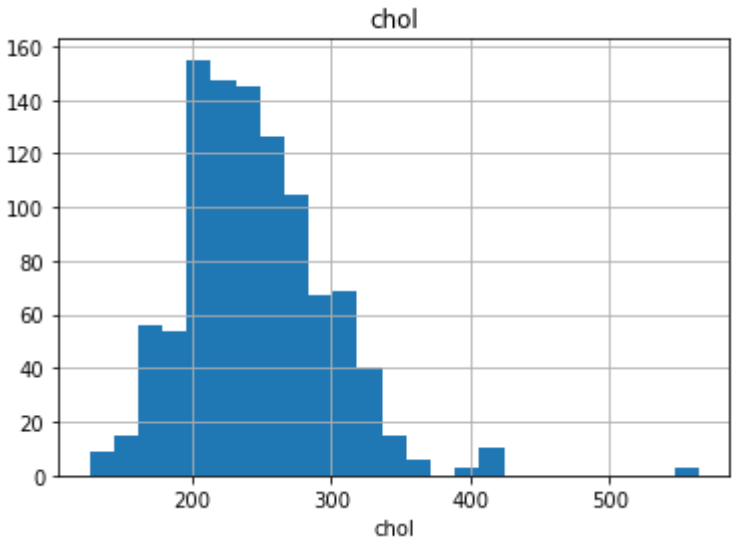
```
Total number of continous features are: 5
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

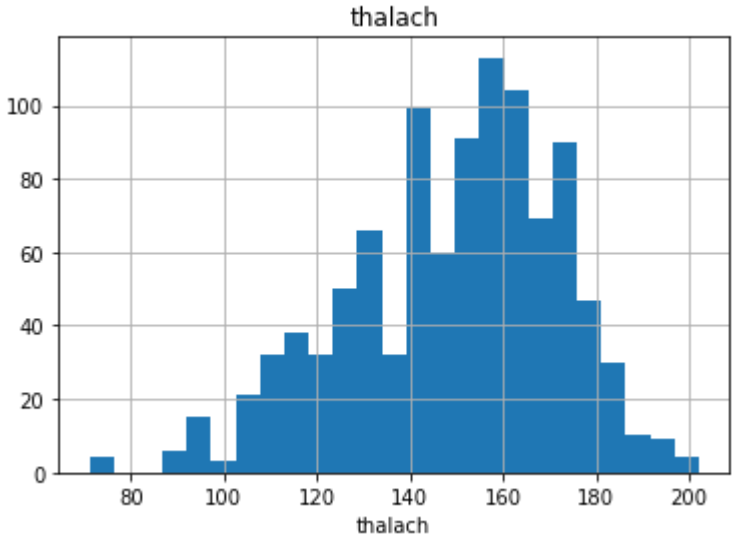
In [28]:

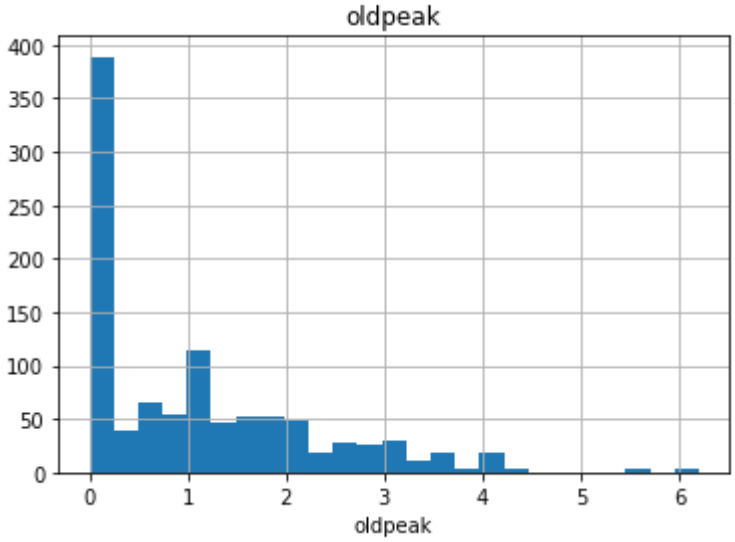
```
for feature in continous_feature:
    data = df.copy()
    data[feature].hist(bins=25)
    plt.xlabel(feature)
    plt.title(feature)
    plt.show()
```











In [29]:

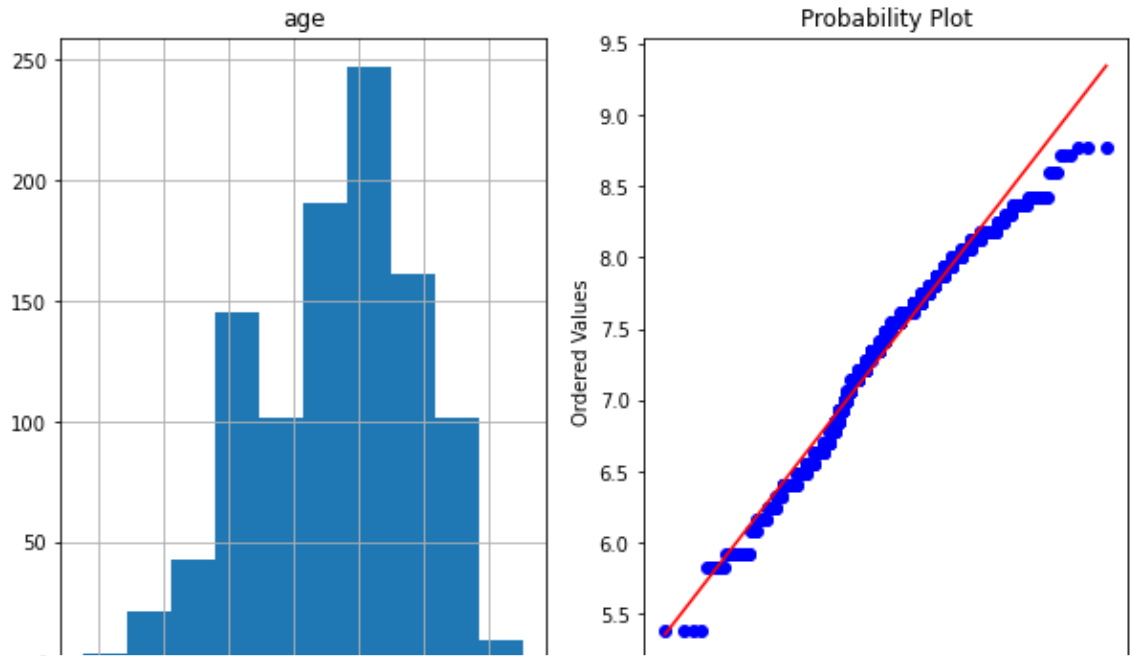
```
import pylab
import scipy.stats as stat
```

In [36]:

```
def plot_data(df, feature):
    plt.figure(figsize=(10, 6))
    plt.subplot(1, 2, 1)
    df[feature].hist()
    plt.title(feature)
    plt.subplot(1, 2, 2)
    stat.probplot(df[feature], dist='norm', plot=pylab)
    plt.show()
```

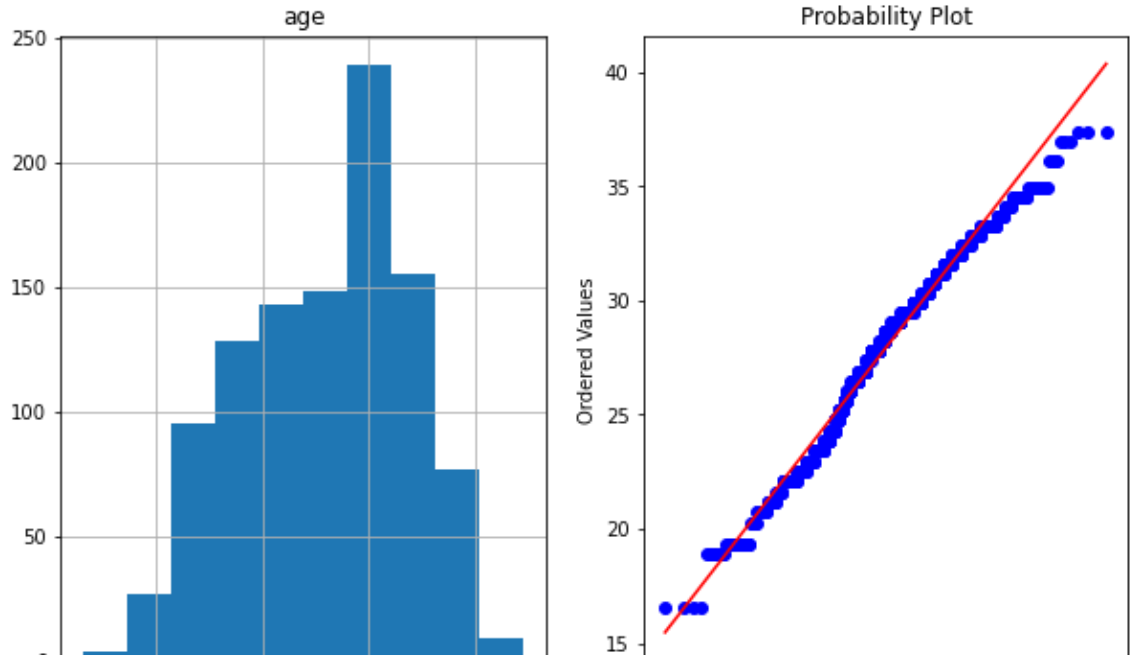
In [37]:

```
# Using Square Root Transformation
for feature in continous_feature:
    data = df.copy()
    data[feature] = data[feature] ** 0.5
    plot_data(data, feature)
```



In [38]:

```
# Using exponential Transformation
for feature in continous_feature:
    data = df.copy()
    data[feature] = (data[feature] ** (1 / 1.2))
    plot_data(data, feature)
```



In [39]:

```
# Using Exponential Transformation for the training dataset
for feature in continous_feature:
    df[feature] = (df[feature] ** (1 / 1.2))
```

In [40]:

```
df.head()
```

Out[40]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
0	26.915521	1	0	55.901699	86.818688	0	1	71.519489	0	1.000000	2
1	27.346173	1	0	61.438419	83.736222	1	0	66.876943	1	2.567250	0
2	34.481147	1	0	63.261575	73.641796	0	1	55.901699	1	2.217232	0
3	30.745026	1	0	64.350426	83.736222	0	1	69.027408	0	0.000000	2
4	31.164470	0	0	60.706133	114.013456	1	1	48.725341	0	1.707240	1

In [41]:

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

In [43]:

```
X.head()
```

Out[43]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
0	26.915521	1	0	55.901699	86.818688	0	1	71.519489	0	1.000000	2
1	27.346173	1	0	61.438419	83.736222	1	0	66.876943	1	2.567250	0
2	34.481147	1	0	63.261575	73.641796	0	1	55.901699	1	2.217232	0
3	30.745026	1	0	64.350426	83.736222	0	1	69.027408	0	0.000000	2
4	31.164470	0	0	60.706133	114.013456	1	1	48.725341	0	1.707240	1

In [44]:

```
Y = df['target']
```

In [45]:

```
Y
```

Out[45]:

```
0       0
1       0
2       0
3       0
4       0
       ..
1020    1
1021    0
1022    0
1023    1
1024    0
Name: target, Length: 1025, dtype: int64
```

In [46]:

```
X = np.array(X)
```

In [47]:

```
Y = np.array(Y)
```

In [48]:

```
X.shape, Y.shape
```

Out[48]:

```
((1025, 13), (1025,))
```

In [49]:

```
df.shape
```

Out[49]:

```
(1025, 14)
```

In [50]:

```
from sklearn.model_selection import train_test_split
```

In [51]:

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, shuffle=True)
```

In [52]:

```
X_train.shape, y_train.shape
```

Out[52]:

```
((768, 13), (768,))
```

In [53]:

```
X_test.shape, y_test.shape
```

Out[53]:

```
((257, 13), (257,))
```

In [54]:

```
n_estimators = [int(x) for x in np.linspace(start=200, stop=2000, num=10)]
criterion = ["gini", "entropy", "log_loss"]
min_samples_leaf = [1 ,2 , 4, 6, 8, 10]
max_features = ["sqrt", "log2", None]
min_samples_split = [2, 5, 6, 8, 10, 12]
```

In [55]:

```
random_grid = {
    'n_estimators' : n_estimators,
    'criterion' : criterion,
    'min_samples_split' : min_samples_split,
    'min_samples_leaf' : min_samples_leaf,
    'max_features' : max_features
}
```

In [56]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [57]:

```
rf = RandomForestClassifier()
```

In [58]:

```
from sklearn.model_selection import KFold
```

In [59]:

```
cv = KFold(n_splits=5, random_state=True, shuffle=True)
```

In [60]:

```
from sklearn.model_selection import RandomizedSearchCV
```

In [61]:

```
rf_randomcv = RandomizedSearchCV(estimator=rf,param_distributions=random_grid, n_iter=100,
```

In [62]:

```
rf_randomcv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
```

Out[62]:

```
RandomizedSearchCV(cv=KFold(n_splits=5, random_state=True, shuffle=True),
                   estimator=RandomForestClassifier(), n_iter=100, n_jobs=-
1,
                   param_distributions={'criterion': ['gini', 'entropy',
                                                      'log_loss'],
                                        'max_features': ['sqrt', 'log2', Non
e],
                                        'min_samples_leaf': [1, 2, 4, 6, 8,
10],
                                        'min_samples_split': [2, 5, 6, 8, 1
0,
                                                              12],
                                        'n_estimators': [200, 400, 600, 800,
                                                         1000, 1200, 1400, 1
600,
                                                         1800, 2000]},
                   random_state=100, verbose=2)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [63]:

```
best_params = rf_randomcv.best_params_
```

In [64]:

```
best_params
```

Out[64]:

```
{'n_estimators': 2000,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'criterion': 'entropy'}
```

In [65]:

```
from sklearn.model_selection import GridSearchCV
```

In [67]:

```
param_grid = {
    'criterion' : [best_params['criterion']],
    'min_samples_leaf' : [best_params['min_samples_leaf']-2, best_params['min_samples_leaf'
                          best_params['min_samples_leaf'], best_params['min_samples_leaf']+1
                          best_params['min_samples_leaf']+2],
    'min_samples_split' : [best_params['min_samples_split']-2, best_params['min_samples_spl
                           best_params['min_samples_split'], best_params['min_samples_split'
                           best_params['min_samples_split']+2],
    'max_features' : [best_params['max_features']],
    'n_estimators' : [best_params['n_estimators']-200, best_params['n_estimators']-100,
                      best_params['n_estimators'], best_params['n_estimators']+100,
                      best_params['n_estimators']+200]
}
```

In [68]:

```
rf = RandomForestClassifier()
```

In [69]:

```
grid_searchcv = GridSearchCV(estimator=rf, param_grid=param_grid, cv=cv,n_jobs=-1, verbose=
```

In [70]:

```
grid_searchcv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 125 candidates, totalling 625 fits
```

In [71]:

```
best_grid = grid_searchcv.best_estimator_
```

In [72]:

```
best_grid
```

Out[72]:

```
RandomForestClassifier(criterion='entropy', min_samples_split=3,
                       n_estimators=1800)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [73]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

In [74]:

```
y_pred = best_grid.predict(X_test)
```

In [75]:

```
print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[115   3]
 [  3 136]]
0.9766536964980544
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       118
           1       0.98      0.98      0.98       139

    accuracy                           0.98       257
   macro avg       0.98      0.98      0.98       257
weighted avg       0.98      0.98      0.98       257
```

In [76]:

```
pip install pandoc
```

```
Defaulting to user installation because normal site-packages is not writeabl
e
Collecting pandoc
  Downloading pandoc-2.2.tar.gz (29 kB)
Collecting plumbum
  Downloading plumbum-1.8.0-py3-none-any.whl (117 kB)
Collecting ply
  Using cached ply-3.11-py2.py3-none-any.whl (49 kB)
Requirement already satisfied: pywin32 in c:\programdata\anaconda3\lib\site-
packages (from plumbum->pandoc) (302)
Building wheels for collected packages: pandoc
  Building wheel for pandoc (setup.py): started
  Building wheel for pandoc (setup.py): finished with status 'done'
  Created wheel for pandoc: filename=pandoc-2.2-py3-none-any.whl size=29557
sha256=227fd1ad15065a9baf5e374dc139f144401d242065c9d033899f451ac4f46e4e
  Stored in directory: c:\users\19138\appdata\local\pip\cache\wheels\2d\da\b
1\54ff0401ef9b07b60c7fc9cffe616f243cf27dc3d04bd5d5ef
Successfully built pandoc
Installing collected packages: ply, plumbum, pandoc
Successfully installed pandoc-2.2 plumbum-1.8.0 ply-3.11
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:

```

```