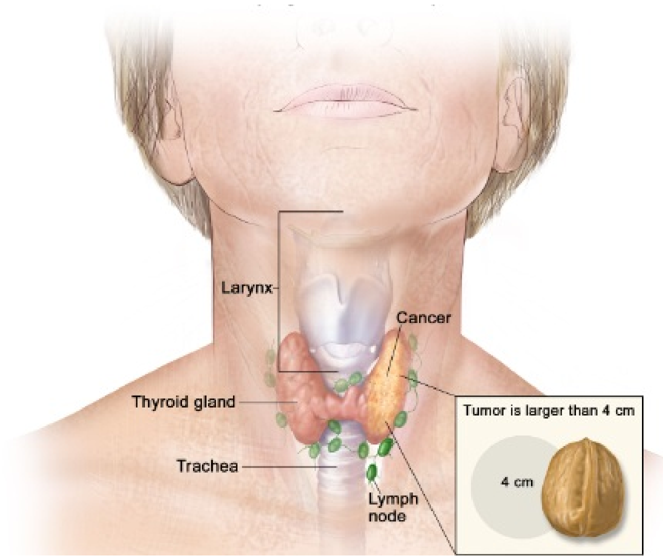


Thyroid Cancer Recurrence Database Overview

This dataset, complied over 15 years at the crossroads of AI and Medicine,is rich in predictive features aimed at understanding the recurrence of well-differentiated thyroid cancer.Each patient,meticulously tracked for a minimum of 10 years.The following list outlines key columns :

1. **Age**: Patient's age
2. **Gender**: Patient's gender
3. **Smoking**: Smoking status
4. **Hx Smoking**: History of smoking(binary)
5. **Hx Radiotherapy**: History of radiotherapy(binary)
6. **Thyroid Function**: Thyroid function details.
7. **Physical Examination**: Results of physical examination.
8. **Adenopathy**: Presence of adenopathy(binary)
9. **Pathology**: Pathology details.
10. **Focality**: Focality information
11. **Risk**: Risk assessment.
12. **T,N,M,Stage**: Cancer staging attributes.
13. **Response**: Response details.
14. **Recurred**: Dependent variable indicating recurrence(binary)

Importing Packages

In [ ]:

```
import numpy as np
import pandas as pd
```

Importing Dataset

In [2]:

```
df=pd.read_csv("C:/Users/hp/OneDrive/Desktop/Thyroid_Diff.csv")
```

Exploratory Data Analysis (EDA)

In [3]:

```
df.head()
```

Out[3]:

	Age	Gender	Smoking	Hx Smoking	Hx Radiothreapy	Thyroid Function	Physical Examination	Adenopathy	Pathology	Focality	Risk	T	N	M	Stage	
0	27	F	No	No	No	Euthyroid	Single nodular goiter-left	No	Micropapillary	Uni-Focal	Low	T1a	N0	M0	I	Ir
1	34	F	No	Yes	No	Euthyroid	Multinodular goiter	No	Micropapillary	Uni-Focal	Low	T1a	N0	M0	I	
2	30	F	No	No	No	Euthyroid	Single nodular goiter-right	No	Micropapillary	Uni-Focal	Low	T1a	N0	M0	I	
3	62	F	No	No	No	Euthyroid	Single nodular goiter-right	No	Micropapillary	Uni-Focal	Low	T1a	N0	M0	I	
4	62	F	No	No	No	Euthyroid	Multinodular goiter	No	Micropapillary	Multi-Focal	Low	T1a	N0	M0	I	

In [4]:
```
df.tail()
```

Out[4]:

	Age	Gender	Smoking	Hx Smoking	Hx Radiothreapy	Thyroid Function	Physical Examination	Adenopathy	Pathology	Focality	Risk	T	N	M	St
378	72	M	Yes	Yes	Yes	Euthyroid	Single nodular goiter-right	Right	Papillary	Uni-Focal	High	T4b	N1b	M1	
379	81	M	Yes	No	Yes	Euthyroid	Multinodular goiter	Extensive	Papillary	Multi-Focal	High	T4b	N1b	M1	
380	72	M	Yes	Yes	No	Euthyroid	Multinodular goiter	Bilateral	Papillary	Multi-Focal	High	T4b	N1b	M1	
381	61	M	Yes	Yes	Yes	Clinical Hyperthyroidism	Multinodular goiter	Extensive	Hurthel cell	Multi-Focal	High	T4b	N1b	M0	
382	67	M	Yes	No	No	Euthyroid	Multinodular goiter	Bilateral	Papillary	Multi-Focal	High	T4b	N1b	M0	

Data PreProcessing

In [5]:
```
df.duplicated().sum()
```

Out[5]:
```
19
```

In [6]:
```
df.drop_duplicates(inplace=True)
```

In [7]:
```
df.duplicated().sum()
```

Out[7]:
```
0
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
Age                     0
Gender                  0
Smoking                 0
Hx Smoking              0
Hx Radiothreapy         0
Thyroid Function        0
Physical Examination    0
Adenopathy              0
Pathology               0
Focality                0
Risk                    0
T                       0
N                       0
M                       0
Stage                   0
Response                0
Recurred                0
dtype: int64
```

Splitting The Data into independent and dependent

In [9]:
```
X=df.iloc[:, 0:16] # IV
Y=df.iloc[:,16:17]
Y
```

Out[9]:

	Recurred
0	No
1	No
2	No
3	No
4	No
...	...
378	Yes
379	Yes
380	Yes
381	Yes
382	Yes

364 rows × 1 columns

In [10]:

```
# Handling the categorical data --X
X=pd.get_dummies(X, drop_first=True)
X
```

Out[10]:

	Age	Gender_M	Smoking_Yes	Hx Smoking_Yes	Hx Radiothreapy_Yes	Thyroid Function_Clinical Hypothyroidism	Thyroid Function_Euthyroid	Thyroid Function_Subclinical Hyperthyroidism	Function_S Hypot
0	27	0	0	0	0	0	1	0	
1	34	0	0	1	0	0	1	0	
2	30	0	0	0	0	0	1	0	
3	62	0	0	0	0	0	1	0	
4	62	0	0	0	0	0	1	0	
...	...	...	...	...	...	...	...	...	
378	72	1	1	1	1	0	1	0	
379	81	1	1	0	1	0	1	0	
380	72	1	1	1	0	0	1	0	
381	61	1	1	1	1	0	0	0	
382	67	1	1	0	0	0	1	0	

364 rows × 40 columns

In [11]:

```
Y=pd.get_dummies(Y, drop_first=True)
Y
```

Out[11]:

	Recurred_Yes
0	0
1	0
2	0
3	0
4	0
...	...
378	1
379	1
380	1
381	1
382	1

364 rows × 1 columns

In [12]:

```
# data splitting
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test=train_test_split(X, Y, test_size=0.3, random_state=2)
```

In [13]:

```
X_train
```

Out[13]:

	Age	Gender_M	Smoking_Yes	Hx Smoking_Yes	Hx Radiothreapy_Yes	Thyroid Function_Clinical Hypothyroidism	Thyroid Function_Euthyroid	Thyroid Function_Subclinical Hyperthyroidism	Function_S Hypot
159	24	0	0	0	0	0	1	0	
115	37	0	0	0	0	0	1	0	
375	59	0	0	0	0	0	1	0	
296	51	0	0	1	0	0	1	0	
80	27	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	
318	30	0	0	0	0	0	1	0	
22	36	0	0	0	0	0	1	0	
78	35	0	0	0	0	0	1	0	
15	42	0	0	0	0	0	1	0	
184	67	0	0	0	0	0	1	0	

254 rows × 40 columns

In [14]:
```
Y_train
```

Out[14]:

	Recurred_Yes
159	0
115	0
375	1
296	0
80	0
...	...
318	1
22	0
78	0
15	0
184	0

254 rows × 1 columns

In [15]:
```
print("Shape of X_train-->", X_train.shape)
print("Shape of Y_train-->", Y_train.shape)
print("Shape of X_test-->", X_test.shape)
print("Shape of Y_test-->", Y_test.shape)
```

```
Shape of X_train--> (254, 40)
Shape of Y_train--> (254, 1)
Shape of X_test--> (110, 40)
Shape of Y_test--> (110, 1)
```

In [16]:
```
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=0)
```

In [17]:
```
from sklearn.preprocessing import StandardScaler
ss= StandardScaler()
X_train=ss.fit_transform(X_train)
X_test=ss.transform(X_test)
```

In [18]:
```
from sklearn.linear_model import LinearRegression
mlr=LinearRegression()
mlr.fit(X_train,Y_train)
```

Out[18]:
```
LinearRegression()
```

In [19]:
```
X_test
```

Out[19]:
```
array([[ 1.08790265, -0.47733437, -0.36372479, ..., -1.0749677 ,
         2.36931137, -0.58396389],
       [-0.03145143,  2.09496751,  2.74933147, ...,  0.93026051,
        -0.42206356, -0.58396389],
       [-1.01911679, -0.47733437, -0.36372479, ..., -1.0749677 ,
        -0.42206356,  1.71243465],
       ...,
       [ 1.68050187, -0.47733437, -0.36372479, ..., -1.0749677 ,
         2.36931137, -0.58396389],
       [-0.88742808, -0.47733437, -0.36372479, ...,  0.93026051,
        -0.42206356, -0.58396389],
       [ 0.03439293, -0.47733437, -0.36372479, ..., -1.0749677 ,
         2.36931137, -0.58396389]])
```

In [20]:
```
print("regression coffiecient--",mlr.coef_)
```

```
regression coffiecient-- [[ 0.01358067  0.04606817 -0.02257455  0.00905311  0.00953028  0.02948823
   0.03557591  0.00878127 -0.00638305 -0.06336698 -0.00478465 -0.04647725
  -0.06486336 -0.00584694  0.01087814  0.00633522 -0.00840969  0.0026246
  -0.04487456  0.00111679 -0.02340344  0.00560606 -0.05208046 -0.11598266
   0.02795634  0.02305345  0.02685346  0.02353439  0.00576156 -0.00131718
   0.01283653  0.02255439 -0.03239784  0.04108073  0.00112558  0.01048471
   0.01858543 -0.20271772 -0.1302626   0.16392676]]
```

In [21]:
```
print("Y_intercept---",mlr.intercept_)
```

```
Y_intercept--- [0.29896907]
```

In [22]:
```
Y_pred=mlr.predict(X_test)
```

In [23]:
```
Y_pred
```

Out[23]:
```
array([[ 3.55539775e-01],
       [-2.29322483e-02],
       [ 9.06636424e-01],
       [ 1.24117209e-01],
       [ 7.44931957e-01],
       [ 9.81735120e-01],
       [-1.73202438e-01],
       [-5.13220977e-02],
       [-2.21304335e-02],
       [-9.30874082e-02],
       [ 1.09182949e+00],
       [-3.88031486e-02],
       [ 1.21208514e+00],
       [ 7.04583203e-02],
       [ 1.62774345e-01],
       [ 1.28328231e+00],
       [-7.82306306e-03],
       [-4.32499012e-02],
       [-2.57072761e-02],
       [ 3.69655879e-02],
       [ 4.90432235e-01],
       [-3.22976614e-02],
       [ 1.19761492e+00],
       [ 1.01687255e+00],
       [-1.26057870e-03],
       [ 3.85979606e-02],
       [ 1.61837753e-03],
       [ 5.81588992e-01],
       [ 8.23100420e-02],
       [ 1.14111378e-02],
       [-3.01834172e-03],
       [ 1.02236928e+00],
       [ 1.96181689e-02],
       [ 1.04102423e-01],
       [ 1.91853032e-02],
       [ 6.88043253e-02],
       [ 4.69058335e-02],
       [-1.94478016e-02],
       [ 3.80788011e-01],
       [ 9.22646453e-01],
       [-4.86394658e-02],
       [ 1.61352140e-01],
       [ 1.09658299e+00],
       [-1.54533236e-01],
       [-1.22738522e-03],
       [-8.20483027e-02],
       [-1.40276359e-01],
       [ 9.77683544e-01],
       [ 3.64970154e-01],
       [ 1.05878811e-01],
       [ 5.84539868e-03],
       [ 7.76235532e-01],
       [ 4.88488066e-03],
       [ 7.60171807e-01],
       [ 3.31178230e-02],
       [ 3.73482254e-02],
       [ 1.14546947e-02],
       [ 2.87775883e-02],
       [ 8.64447292e-01],
       [ 5.36517214e-01],
       [ 1.79803210e-02],
       [-1.96885216e-02],
       [ 9.68741437e-01],
       [ 1.16109720e+00],
       [ 3.00556422e-02],
       [ 7.99530494e-02],
       [ 4.19397163e-01],
       [-4.23134353e-02],
       [-1.21727326e-01],
       [ 4.29232551e-01],
       [ 1.59496835e-01],
       [-4.50626232e-02],
       [ 1.89032583e-01]])
```

In [24]:
```
from sklearn.metrics import r2_score
z=r2_score(Y_test,Y_pred)*100
z
```

Out[24]:
```
79.37270316465978
```

Decision Tree

In [25]:
```
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor()
dtr.fit(X_train, Y_train)
```

```
# Make predictions on new data (X_test)
Y1_pred = dtr.predict(X_test)
```

In [26]:
```
from sklearn.metrics import r2_score
z1=r2_score(Y_test,Y1_pred)*100
z1
```

Out[26]:
```
66.57509157509158
```

Random Forest Regression Model

In [27]:
```
from sklearn.ensemble import RandomForestRegressor
rfr=RandomForestRegressor()
rfr.fit(X_train, Y_train)

# Make predictions on new data (X_test)
Y2_pred =rfr.predict(X_test)
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_55072\1089883793.py:3: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  rfr.fit(X_train, Y_train)
```

In [28]:
```
from sklearn.metrics import r2_score
z2=r2_score(Y_test,Y2_pred)*100
z2
```

Out[28]:
```
80.29935897435897
```

SVR

In [29]:
```
from sklearn.svm import SVR

# Create and fit the model (adjust hyperparameters like 'C' and 'kernel')
svrm = SVR(kernel='linear')
svrm.fit(X_train, Y_train)

# Make predictions on new data (X_test)
Y3_pred =svrm.predict(X_test)
```

```
C:\Users\hp\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rav
el().
  y = column_or_1d(y, warn=True)
```

In [31]:
```
from sklearn.metrics import r2_score
z3=r2_score(Y_test,Y3_pred)*100
z3
```

Out[31]:
```
77.41640879945099
```

Table

In [32]:
```
pd.DataFrame(([mlr,dtr,rfr,svrm],[z,z1,z2,z3]),index=('Model','r2_score'))
```

Out[32]:

	0	1	2	3
Model	LinearRegression()	DecisionTreeRegressor()	(DecisionTreeRegressor(max_features='auto', ra...	SVR(kernel='linear')
r2_score	79.372703	66.575092	80.299359	77.416409

Visualization

In [33]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

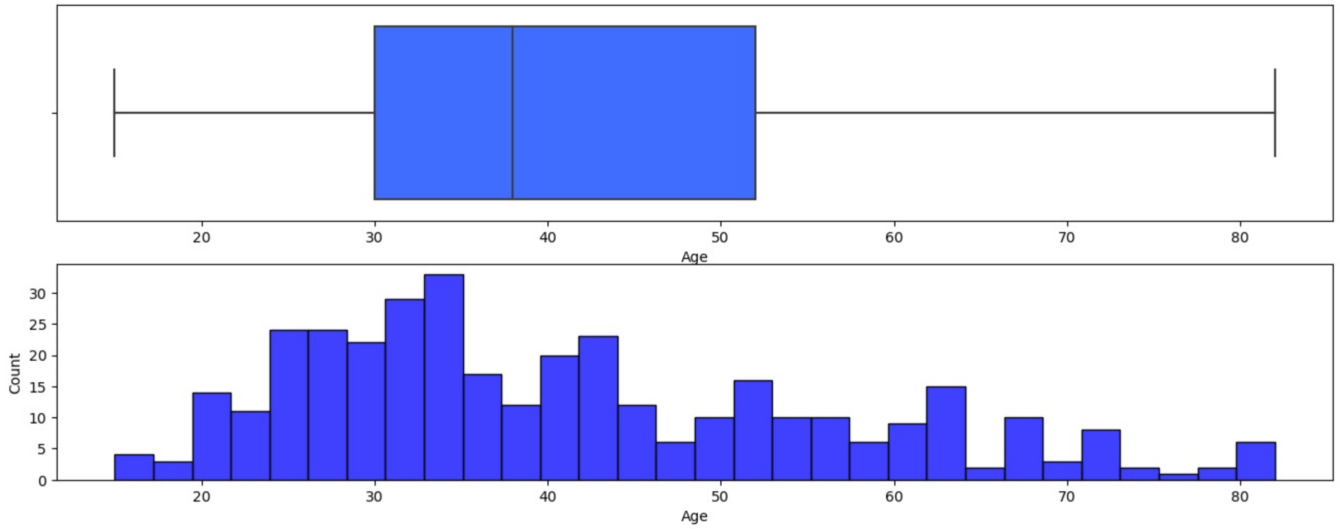
In [34]:
```
target='Recurred'
columns=df.columns.tolist()
```

In [35]:
```
plt.figure(figsize=(16,6))
plt.subplot(2,1,1)
sns.boxplot(df[columns[0]],orient='h', boxprops=dict(facecolor='#416DFE'))
plt.subplot(2,1,2)
sns.histplot(df[columns[0]],bins=30,color='b')
```

```
C:\Users\hp\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable a
s a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[35]:

```
<AxesSubplot:xlabel='Age', ylabel='Count'>
```



In [36]:

```
print("\033[1m"+"Skewness"+"\033[0m",df[columns[0]].skew())
```

Skewness 0.6782692112593017

Skewness 0.7197318617338616

Inference:

The age distribution in the dataset is positively skewed to the right, indicating that most patients fall within the younger age range. There is a prominent peak between the ages of 30 to 40 , suggesting a concentration of patients in this age group.

Gender

In [37]:

```
df[columns[1]]
```

Out[37]:

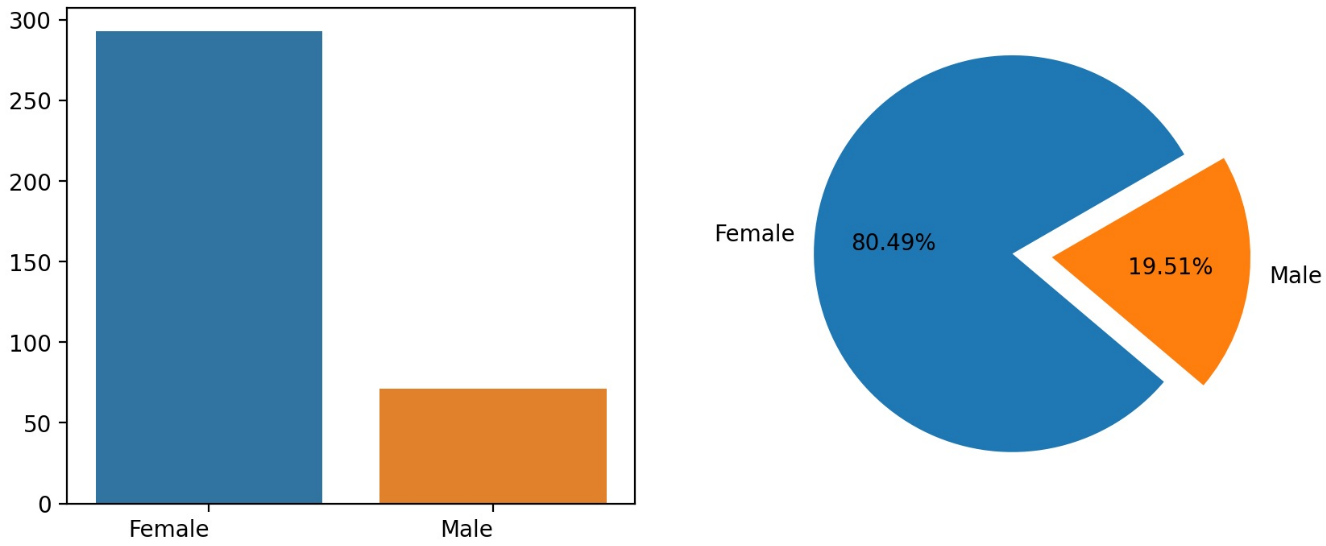
```
0      F
1      F
2      F
3      F
4      F
      ..
378    M
379    M
380    M
381    M
382    M
Name: Gender, Length: 364, dtype: object
```

In [38]:

```
age=df[columns[1]].value_counts()
```

In [39]:

```
plt.figure(figsize=(10,4),dpi=200)
plt.subplot(1,2,1)
custom_labels = ('Female','Male')
sns.barplot(x=age.index,y=age.values)
plt.xticks([0,1],custom_labels, rotation=0, ha='right')
plt.subplot(1,2,2)
plt.pie(labels=custom_labels,x=age.values,autopct='%.2f%%',startangle=30,explode=(0.1,.1))
plt.show()
```



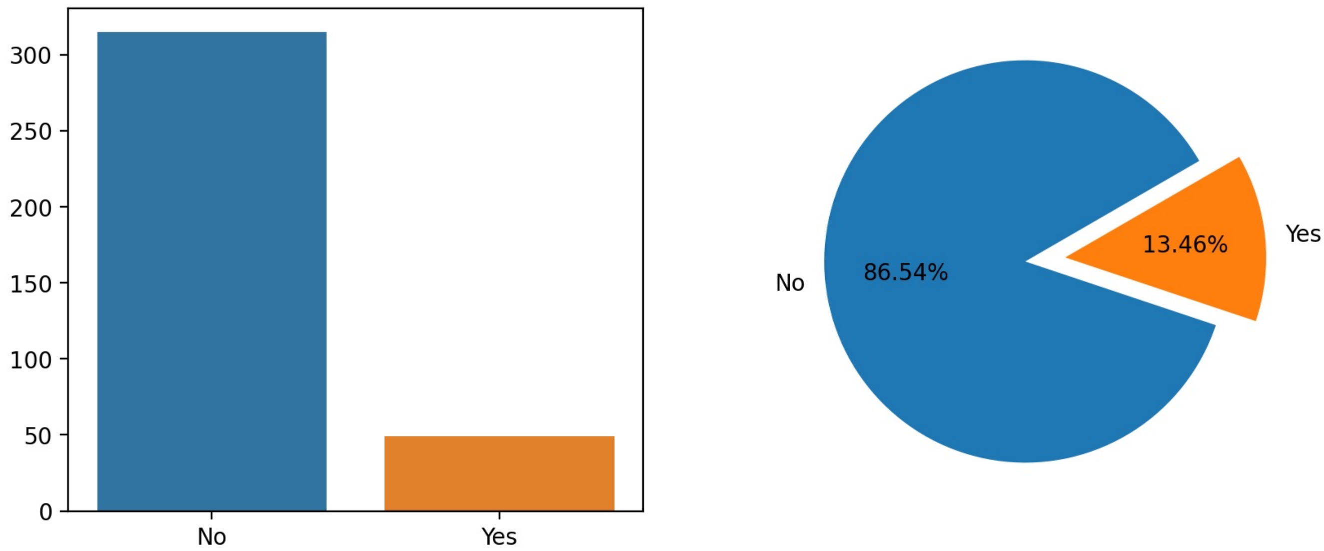
Inference:

Aproximately 81% of the individuals are females with frequency of 312, indicating a predominant representation of female patients in the study on well-differentiated thyroid cancer recurrence . This gender distribution provides valuable context for further analysis and may prompt considerations regarding the impact of gender on the prediction of cancer recurrence.

Smoking

In [40]:

```
Smoking = df[columns[2]].value_counts()
plt.figure(figsize=(10,4),dpi=200)
plt.subplot(1,2,1)
sns.barplot(x=Smoking.index,y=Smoking.values)
plt.subplot(1,2,2)
plt.pie(labels=Smoking.index,x=Smoking.values,autopct='%.2f%%',startangle=30,explode=(0.1,.1))
plt.show()
```



Inference:

The dataset indicates that around 87% of individuals are non-smokers , with 49 cases identified as smokers .

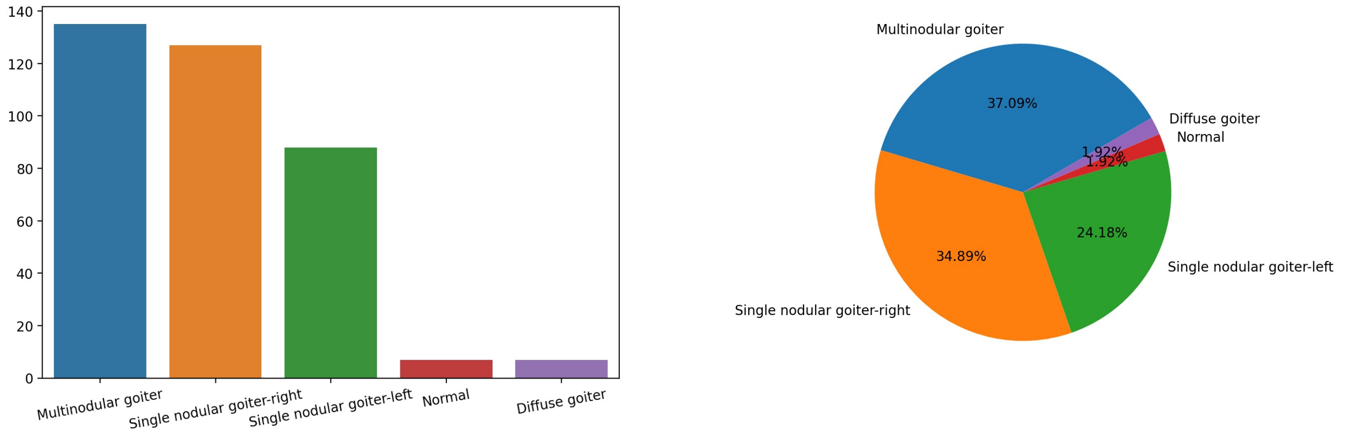
In [41]:

```
def visualize(data,figsize=(10,4),ticks_rotation=0,lab_disatance=1.1,pct_dist=0.6,textprops=None):
    data = data.value_counts()
    colors = ('#ffa600','#ff6361','#d45087','#a05195','#f95d6a','#ff7c43')
    fig = plt.figure(figsize=figsize,dpi=200)
    plt.subplot(1,2,1)
    plt.title('')
    sns.barplot(x=data.index,y=data.values)
    plt.xticks(rotation=ticks_rotation)
    plt.subplot(1,2,2)
    plt.pie(labels=data.index,x=data.values,autopct='%.2f%%',startangle=30)
    plt.show()
```

Results of Physical Examination

In [42]:

```
# visualize(df[columns[6]],ticks_rotation=40,figsize=(14,5))
visualize(df[columns[6]],(17,5),ticks_rotation=10,textprops=dict(size=10),pct_dist=1.2,lab_disatance=1.37)
```



The distribution of Physical Examination findings indicates diverse statuses among patients:

Multinodular goiter: 140 cases

Single nodular goiter-right: 140 cases

Single nodular goiter-left: 89 cases

Normal: 7 cases

Diffuse goiter: 7 cases

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js