In [1]:

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


from warnings import filterwarnings
filterwarnings(action='ignore')
```

In [2]:

```
thyroid_data = pd.read_csv("thyroid_data.csv")
thyroid_data
```

Out[2]:

| | S.no | Age | Sex | On Thyroxine | Query on Thyroxine | On Antithyroid Medication | Sick | Pregnant | Thyroid Surgery | I131 Treatment |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 41 | F | f | f | f | f | f | f | f |
| **1** | 1 | 23 | F | f | f | f | f | f | f | f |
| **2** | 2 | 46 | M | f | f | f | f | f | f | f |
| **3** | 3 | 70 | F | t | f | f | f | f | f | f |
| **4** | 4 | 70 | F | f | f | f | f | f | f | f |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3216** | 2774 | 82 | M | f | f | f | f | f | f | f |
| **3217** | 2776 | 79 | M | f | f | f | f | f | f | f |
| **3218** | 2782 | 50 | F | f | f | f | f | f | f | f |
| **3219** | 2786 | 73 | ? | f | f | f | f | f | f | f |
| **3220** | 2796 | 73 | M | f | t | f | f | f | f | f |

3221 rows × 28 columns

In [3]:

```
thyroid_data = thyroid_data.drop(['S.no'], axis = 1)
```

In [4]:

```
thyroid_data
```

Out[4]:

| | Age | Sex | On Thyroxine | Query on Thyroxine | On Antithyroid Medication | Sick | Pregnant | Thyroid Surgery | I131 Treatment | Hypoth |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 41 | F | f | f | f | f | f | f | f | |
| **1** | 23 | F | f | f | f | f | f | f | f | |
| **2** | 46 | M | f | f | f | f | f | f | f | |
| **3** | 70 | F | t | f | f | f | f | f | f | |
| **4** | 70 | F | f | f | f | f | f | f | f | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **3216** | 82 | M | f | f | f | f | f | f | f | |
| **3217** | 79 | M | f | f | f | f | f | f | f | |
| **3218** | 50 | F | f | f | f | f | f | f | f | |
| **3219** | 73 | ? | f | f | f | f | f | f | f | |
| **3220** | 73 | M | f | t | f | f | f | f | f | |

3221 rows × 27 columns

In [5]:

```
thyroid_data.shape
```

Out[5]:

```
(3221, 27)
```

In [6]:

```
## Columns
thyroid_data.columns
```

Out[6]:

```
Index(['Age', 'Sex', 'On Thyroxine', 'Query on Thyroxine',
       'On Antithyroid Medication', 'Sick', 'Pregnant', 'Thyroid Surgery',
       'I131 Treatment', 'Query Hypothyroid', 'Query Hyperthyroid', 'Lithiu
m',
       'Goitre', 'Tumor', 'Hypopituitary', 'Psych', 'TSH Measured', 'TSH',
       'T3 Measured', 'T3', 'TT4 Measured', 'TT4', 'T4U Measured', 'T4U',
       'FTI Measured', 'FTI', 'Category'],
      dtype='object')
```

In [7]:

```
# A quick fix needed
thyroid_data.loc[thyroid_data['Age'] == '455', 'Age'] = '45'
```

In [8]:

```
## Let's drop some unnecessary columns
thyroid_data = thyroid_data.drop(['TSH Measured','T3 Measured','TT4 Measured','T4U Measured
```

In [9]:

```
#Checking for null values
thyroid_data.isna().sum()
```

Out[9]:

```
Age                          0
Sex                          0
On Thyroxine                 0
Query on Thyroxine           0
On Antithyroid Medication    0
Sick                         0
Pregnant                     0
Thyroid Surgery              0
I131 Treatment               0
Query Hypothyroid            0
Query Hyperthyroid           0
Lithium                      0
Goitre                       0
Tumor                        0
Hypopituitary                0
Psych                        0
TSH                          0
T3                           0
TT4                          0
T4U                          0
FTI                          0
Category                     0
dtype: int64
```

In [10]:

```
thyroid_data.dtypes
```

Out[10]:

```
Age                          object
Sex                          object
On Thyroxine                 object
Query on Thyroxine           object
On Antithyroid Medication    object
Sick                         object
Pregnant                     object
Thyroid Surgery              object
I131 Treatment               object
Query Hypothyroid            object
Query Hyperthyroid           object
Lithium                      object
Goitre                       object
Tumor                        object
Hypopituitary                object
Psych                        object
TSH                          object
T3                           object
TT4                          object
T4U                          object
FTI                          object
Category                     object
dtype: object
```

In [11]:

```
n = len(thyroid_data[thyroid_data['Category'] == 'hyperthyroid'])
print("No of hyperthyroid in Dataset:",n)
```

```
No of hyperthyroid in Dataset: 77
```

In [12]:

```
n1 = len(thyroid_data[thyroid_data['Category'] == 'hypothyroid'])
print("No of hypothyroid in Dataset:",n1)
```

```
No of hypothyroid in Dataset: 220
```

In [13]:

```
n2 = len(thyroid_data[thyroid_data['Category'] == 'sick'])
print("No of sick in Dataset:",n2)
```

```
No of sick in Dataset: 171
```

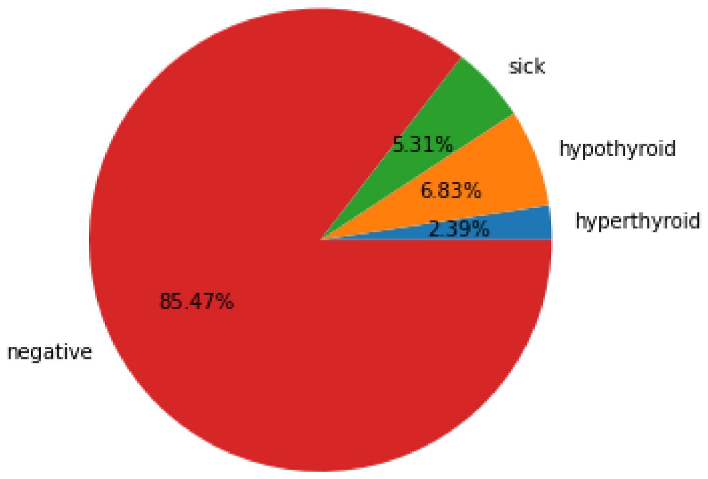
In [14]:

```
n3 = len(thyroid_data[thyroid_data['Category'] == 'negative'])
print("No of negative in Dataset:",n3)
```

```
No of negative in Dataset: 2753
```

In [15]:

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
l = ['hyperthyroid', 'hypothyroid', 'sick','negative']
s = [77,220,171,2753]
ax.pie(s, labels = l,autopct='%1.2f%%')
plt.show()
```



In [ ]:

```

```

In [16]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
```

In [17]:

```
train, test = train_test_split(thyroid_data, test_size = 0.20)
print(train.shape)
print(test.shape)
```

```
(2576, 22)
(645, 22)
```

In [18]:

```
thyroid_data.columns
```

Out[18]:

```
Index(['Age', 'Sex', 'On Thyroxine', 'Query on Thyroxine',
       'On Antithyroid Medication', 'Sick', 'Pregnant', 'Thyroid Surgery',
       'I131 Treatment', 'Query Hypothyroid', 'Query Hyperthyroid', 'Lithiu
m',
       'Goitre', 'Tumor', 'Hypopituitary', 'Psych', 'TSH', 'T3', 'TT4', 'T4
U',
       'FTI', 'Category'],
      dtype='object')
```

In [19]:

```
thyroid_data.dtypes
```

Out[19]:

```
Age                          object
Sex                          object
On Thyroxine                 object
Query on Thyroxine           object
On Antithyroid Medication    object
Sick                         object
Pregnant                     object
Thyroid Surgery              object
I131 Treatment               object
Query Hypothyroid            object
Query Hyperthyroid           object
Lithium                      object
Goitre                       object
Tumor                        object
Hypopituitary                object
Psych                        object
TSH                          object
T3                           object
TT4                          object
T4U                          object
FTI                          object
Category                     object
dtype: object
```

In [ ]:

```

```

In [20]:

```
def convert_category(dataframe, column):

    if column == 'Sex':
        conditionF = dataframe[column] == 'F' # For sex column
        conditionT = dataframe[column] == 'M' # For sex column
    else:
        conditionF = dataframe[column] == 'f'
        conditionT = dataframe[column] == 't'

    dataframe.loc[conditionF, column] = 0
    dataframe.loc[conditionT, column] = 1
```

In [21]:

```
# Binarize Category Columns
binary_cols = ['Age', 'Sex', 'On Thyroxine', 'Query on Thyroxine',
       'On Antithyroid Medication', 'Sick', 'Pregnant', 'Thyroid Surgery',
       'I131 Treatment', 'Query Hypothyroid', 'Query Hyperthyroid', 'Lithium',
       'Goitre', 'Tumor', 'Hypopituitary', 'Psych', 'TSH', 'T3', 'TT4', 'T4U',
       'FTI']

for col in binary_cols: convert_category(thyroid_data, col)
```

In [22]:

```
# Convert '?' to np.nan and convert numeric data to numeric dtype
for col in thyroid_data.columns:
    if col != 'Category':
        thyroid_data.loc[thyroid_data[col] == '?', col] = np.nan
        thyroid_data[col] = pd.to_numeric(thyroid_data[col])
```

In [23]:

```
from sklearn.impute import SimpleImputer

curr_columns = thyroid_data.columns.difference(['Category'])

imputer = SimpleImputer(missing_values=np.nan, strategy='median')
imputed_data = imputer.fit_transform(thyroid_data.drop('Category', axis=1))
imputed_data = pd.DataFrame(imputed_data, columns=curr_columns)
```

In [24]:

```
thyroid_data = pd.concat([
                    imputed_data.reset_index(),
                    thyroid_data['Category'].reset_index()],
                    axis=1).drop('index', axis=1)
```

In [25]:

```
thyroid_data.dtypes
```

Out[25]:

```
Age                          float64
FTI                          float64
Goitre                       float64
Hypopituitary                float64
I131 Treatment               float64
Lithium                      float64
On Antithyroid Medication    float64
On Thyroxine                 float64
Pregnant                     float64
Psych                        float64
Query Hyperthyroid           float64
Query Hypothyroid            float64
Query on Thyroxine           float64
Sex                          float64
Sick                         float64
T3                           float64
T4U                          float64
TSH                          float64
TT4                          float64
Thyroid Surgery              float64
Tumor                        float64
Category                      object
dtype: object
```

In [26]:

```
thyroid_data.describe()
```

Out[26]:

| | Age | FTI | Goitre | Hypopituitary | I131 Treatment | Lithium | Antith Medic |
|---|---|---|---|---|---|---|---|
| **count** | 3221.000000 | 3221.000000 | 3221.000000 | 3221.000000 | 3221.000000 | 3221.000000 | 3221.00 |
| **mean** | 52.406085 | 0.306116 | 0.106489 | 0.014902 | 0.010866 | 0.043775 | 0.01 |
| **std** | 19.104151 | 0.460950 | 0.308510 | 0.121180 | 0.103689 | 0.204626 | 0.11 |
| **min** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| **25%** | 37.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| **50%** | 55.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| **75%** | 68.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| **max** | 94.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.00 |

8 rows × 21 columns

In [27]:

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

X = thyroid_data.drop('Category', axis=1)
y = thyroid_data['Category']

col_names = X.columns

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

scaler = MinMaxScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train))
X_test = pd.DataFrame(scaler.transform(X_test))
```

In [28]:

```
#Using LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
prediction = model.predict(X_test)
print('Accuracy:',metrics.accuracy_score(prediction,y_test))
```

```
Accuracy: 0.8496124031007752
```

In [29]:

```
#Confusion matrix
from sklearn.metrics import confusion_matrix,classification_report
confusion_mat = confusion_matrix(y_test,prediction)
print("Confusion matrix: \n",confusion_mat)
print(classification_report(y_test,prediction))
```

```
Confusion matrix:
 [[  0   0  16   0]
 [  0   2  42   0]
 [  3   2 546   0]
 [  0   0  34   0]]
              precision    recall  f1-score   support

hyperthyroid       0.00      0.00      0.00        16
 hypothyroid       0.50      0.05      0.08        44
    negative       0.86      0.99      0.92       551
        sick       0.00      0.00      0.00        34

    accuracy                           0.85       645
   macro avg       0.34      0.26      0.25       645
weighted avg       0.77      0.85      0.79       645
```

In [30]:

```
#Using KNN Neighbors
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=5)
model2.fit(X_train,y_train)
y_pred2 = model2.predict(X_test)
```

In [31]:

```
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(y_test,y_pred2))
```

```
Accuracy Score: 0.8232558139534883
```

In [32]:

```
#Using GaussianNB
from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(X_train,y_train)
y_pred3 = model3.predict(X_test)
```

In [33]:

```
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(y_test,y_pred3))
```

```
Accuracy Score: 0.10077519379844961
```

In [34]:

```
#Using Decision Tree
from sklearn.tree import DecisionTreeClassifier
model4 = DecisionTreeClassifier(criterion='entropy',random_state=7)
model4.fit(X_train,y_train)
y_pred4 = model4.predict(X_test)
```

In [35]:

```
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(y_test,y_pred4))
```

```
Accuracy Score: 0.7162790697674418
```

In [36]:

```
# Fitting Naive Bayes Classification to the Training set with linear kernel
from sklearn.naive_bayes import GaussianNB
nvclassifier = GaussianNB()
nvclassifier.fit(X_train, y_train)
```

Out[36]:

```
GaussianNB()
```

In [37]:

```
# Predicting the Test set results
y_pred = nvclassifier.predict(X_test)
print(y_pred)
```

```
['hypothyroid' 'hyperthyroid' 'hypothyroid' 'hyperthyroid' 'hypothyroid'
 'hyperthyroid' 'hyperthyroid' 'hyperthyroid' 'hypothyroid' 'hypothyroid'
 'hyperthyroid' 'hyperthyroid' 'hyperthyroid' 'hyperthyroid'
 'hyperthyroid' 'hyperthyroid' 'hyperthyroid' 'hypothyroid' 'hyperthyroi
d'
 'hyperthyroid' 'hyperthyroid' 'hypothyroid' 'hyperthyroid' 'hypothyroid'
 'hypothyroid' 'hyperthyroid' 'hyperthyroid' 'hyperthyroid' 'hypothyroid'
 'hyperthyroid' 'hypothyroid' 'hypothyroid' 'hyperthyroid' 'hyperthyroid'
 'hypothyroid' 'hyperthyroid' 'hyperthyroid' 'hypothyroid' 'hyperthyroid'
 'hyperthyroid' 'hyperthyroid' 'hyperthyroid' 'hyperthyroid'
 'hyperthyroid' 'hypothyroid' 'hyperthyroid' 'hyperthyroid' 'hypothyroid'
 'hyperthyroid' 'hyperthyroid' 'hyperthyroid' 'hyperthyroid'
 'hyperthyroid' 'hyperthyroid' 'hypothyroid' 'hypothyroid' 'hypothyroid'
 'hyperthyroid' 'hyperthyroid' 'hyperthyroid' 'hypothyroid' 'hypothyroid'
 'hyperthyroid' 'hypothyroid' 'hyperthyroid' 'hyperthyroid' 'hyperthyroi
d'
 'hyperthyroid' 'hyperthyroid' 'hypothyroid' 'hyperthyroid' 'hyperthyroi
d'
 'hypothyroid' 'hypothyroid' 'hyperthyroid' 'hypothyroid' 'hypothyroid'
```

In [38]:

```
#lets see the actual and predicted value side by side
y_compare = np.vstack((y_test,y_pred)).T
#actual value on the left side and predicted value on the right hand side
#printing the top 5 values
y_compare[:15,:]
```

Out[38]:

```
array([['negative', 'hypothyroid'],
       ['negative', 'hyperthyroid'],
       ['negative', 'hypothyroid'],
       ['negative', 'hyperthyroid'],
       ['negative', 'hypothyroid'],
       ['negative', 'hyperthyroid'],
       ['hyperthyroid', 'hyperthyroid'],
       ['negative', 'hyperthyroid'],
       ['negative', 'hypothyroid'],
       ['hypothyroid', 'hypothyroid'],
       ['negative', 'hyperthyroid'],
       ['negative', 'hyperthyroid'],
       ['negative', 'hyperthyroid'],
       ['negative', 'hyperthyroid'],
       ['negative', 'hyperthyroid']], dtype=object)
```

In [39]:

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[ 16   0   0   0]
 [  0  40   1   3]
 [306 229   7   9]
 [ 15  17   0   2]]
```

In [40]:

```
#finding accuracy from the confusion matrix.
a = cm.shape
corrPred = 0
falsePred = 0

for row in range(a[0]):
    for c in range(a[1]):
        if row == c:
            corrPred +=cm[row,c]
        else:
            falsePred += cm[row,c]
print('Correct predictions: ', corrPred)
print('False predictions', falsePred)
print ('\n\nAccuracy of the Naive Bayes Clasification is: ', corrPred/(cm.sum()))
```

```
Correct predictions:  65
False predictions 580


Accuracy of the Naive Bayes Clasification is:  0.10077519379844961
```

In [41]:

```
#Using Support Vector
from sklearn.svm import SVC
model1 = SVC()
```

In [42]:

```
model1.fit(X_train,y_train)

pred_y = model1.predict(X_test)

from sklearn.metrics import accuracy_score
print("Acc=",accuracy_score(y_test,pred_y))
```

```
Acc= 0.8542635658914729
```

In [46]:

```
results = pd.DataFrame({
    'Model': ['Logistic Regression','KNN','Decision tree','GaussianNayeBayes','Support Vect
    'Score': [0.8496124031,0.8232558,0.7162790,0.1007751,0.85426356]})

result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df.head(9)
```

Out[46]:

| | Model |
|---|---|
| **Score** | |
| **0.854264** | Support Vector Machines |
| **0.849612** | Logistic Regression |
| **0.823256** | KNN |
| **0.716279** | Decision tree |
| **0.100775** | GaussianNayeBayes |

In [ ]:

In [ ]:

In [ ]:

In [ ]: