Exploratory Data Analysis (EDA)

In [3]:
```
#Importing the Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plot
import seaborn as sns
from sklearn.utils import resample
from imblearn.over_sampling import SMOTENC,RandomOverSampler,KMeansSMOTE
from sklearn.impute import KNNImputer
from sklearn.preprocessing import LabelEncoder
sns.set()
```

In [4]:
```
df = pd.read_csv("InputFile.csv")
```

In [5]:
```
df
```

Out[5]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant
0	41	F	f	f	f	f	f
1	23	F	f	f	f	f	f
2	46	M	f	f	f	f	f
3	70	F	t	f	f	f	f
4	70	F	f	f	f	f	f
...	...	...	...	...	...	...	...
3767	30	F	f	f	f	f	f
3768	68	F	f	f	f	f	f
3769	74	F	f	f	f	f	f
3770	72	M	f	f	f	f	f
3771	64	F	f	f	f	f	f

3772 rows × 30 columns

Problem Statement : To build a classification methodology to predict the type of Thyroid a person has ,based on the below features.

age - Age of the person

sex - Male or Female

on_thyroxine - true or false

on_antithyroid_medication - true or false

sick - true or false

pregnant - true or false

thyroid_surgery - true or false

I131_treatment - true or false

query_hypothyroid - true or false

query_hyperthyroid -true or false

lithium - true or false

goitre - true or false

tumor - true or false

hypopituitary- true or false

psych - true or false

TSH_measured - true or false

TSH - thyroid stimulating hormone floating value

T3_measured - true or false

T3 - triiodothyronine value

TT4_measured- true or false

TT4 - Thyroxine value

T4U_measured- true or false

T4U - numerical value

FTI_measured- true or false

FTI -Free Thyroxine Index

TBG_measured- true or false

TBG -Thyroid-Binding Globulin value

referral_source - different sources of referals

Class - different types of thyroid

In [6]:
```
df.describe()
```

Out[6]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	preg
count	3772	3772	3772	3772	3772	3772	
unique	94	3	2	2	2	2	
top	59	F	f	f	f	f	
freq	95	2480	3308	3722	3729	3625	

4 rows × 30 columns

In [8]:
```
for column in df.columns:
    count = df[column][df[column]=="?"].count()
    if count != 0:
        print(column, count)
```

```
age 1
sex 150
TSH 369
T3 769
TT4 231
T4U 387
FTI 385
TBG 3772
```

In [9]:
```
df = df.drop(["TBG"], axis = 1)
```

In [10]:
```
df[["T4U_measured", "T4U"]]
```

Out[10]:

	T4U_measured	T4U
0	t	1.14
1	f	?
2	t	0.91
3	f	?
4	t	0.87
...	...	...
3767	f	?
3768	t	1.08
3769	t	1.07
3770	t	0.94
3771	t	1.07

3772 rows × 2 columns

In [11]:
```
df = df.drop(['TSH_measured','T3_measured','TT4_measured','T4U_measured','F
```

In [12]:
```
for column in df.columns:
    count = df[column][df[column] == "?"].count()
    if count!=0:
        df[column] = df[column].replace("?", np.nan)
```

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
age                          1
sex                        150
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                        369
T3                         769
TT4                        231
T4U                        387
FTI                        385
referral_source              0
Class                        0
dtype: int64
```

In [15]:
```
df['sex'] = df['sex'].map({"F" : 0, "M" : 1})
```

In [16]:
```
for column in df.columns:
    if len(df[column].unique()) == 2:
        df[column] = df[column].map({"f":0, "t":1})
```

In [17]:
```
df.head()
```

Out[17]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	tl
0	41	0.0	0	0	0	0	0	
1	23	0.0	0	0	0	0	0	
2	46	1.0	0	0	0	0	0	
3	70	0.0	1	0	0	0	0	
4	70	0.0	0	0	0	0	0	

5 rows × 23 columns

In [18]:
```
df = df.drop(columns=["referral_source"], axis = True)
```

In [19]:
```
df["Class"].unique()
```

Out[19]:
```
array(['negative', 'compensated_hypothyroid', 'primary_hypothyroid',
       'secondary_hypothyroid'], dtype=object)
```

In [20]:
```
lbn = LabelEncoder()
df["Class"] = lbn.fit_transform(df["Class"])
```

In [21]:
```
df.head()
```

Out[21]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	tl
0	41	0.0	0	0	0	0	0	
1	23	0.0	0	0	0	0	0	
2	46	1.0	0	0	0	0	0	
3	70	0.0	1	0	0	0	0	
4	70	0.0	0	0	0	0	0	

5 rows × 22 columns

In [22]:
```
df.describe(include='all')
```

Out[22]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	
count	3771	3622.000000	3772.000000	3772.000000	3772.000000	377
unique	93	NaN	NaN	NaN	NaN	
top	59	NaN	NaN	NaN	NaN	
freq	95	NaN	NaN	NaN	NaN	
mean	NaN	0.315295	0.123012	0.013256	0.011400	
std	NaN	0.464698	0.328494	0.114382	0.106174	
min	NaN	0.000000	0.000000	0.000000	0.000000	
25%	NaN	0.000000	0.000000	0.000000	0.000000	
50%	NaN	0.000000	0.000000	0.000000	0.000000	
75%	NaN	1.000000	0.000000	0.000000	0.000000	
max	NaN	1.000000	1.000000	1.000000	1.000000	

11 rows × 22 columns

In [23]:
```
imputer = KNNImputer(n_neighbors=3, weights='uniform', missing_values=np.na
new_arr = imputer.fit_transform(df)
```

In [25]:
```
new_df = pd.DataFrame(new_arr)
```

In [26]:
```
new_df
```

Out[26]:

	0	1	2	3	4	5	6	7	8	9	...	12	13	14	15	16	
0	41.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.300000	2.5000
1	23.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	4.100000	2.0000
2	46.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.980000	1.6333
3	70.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.160000	1.9000
4	70.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.720000	1.2000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3767	30.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	1.566667	2.5333
3768	68.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.000000	2.1000
3769	74.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	5.100000	1.8000
3770	72.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.700000	2.0000
3771	64.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.000000	2.2000

3772 rows × 22 columns

In [27]:
```
new_df.isnull().sum()
```

Out[27]:
```
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
11    0
12    0
13    0
14    0
15    0
16    0
17    0
18    0
19    0
20    0
21    0
dtype: int64
```

In [29]:
```
new_df.describe()
```

Out[29]:

	0	1	2	3	4	5	
count	3772.000000	3772.000000	3772.000000	3772.000000	3772.000000	3772.000000	3772.
mean	51.737275	0.308855	0.123012	0.013256	0.011400	0.038971	0.
std	20.082478	0.458819	0.328494	0.114382	0.106174	0.193552	0.
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	36.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
50%	54.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
75%	67.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.
max	455.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.

8 rows × 22 columns

In [32]:
```
columns = df.columns
```

In [34]:
```
new_df = pd.DataFrame(new_arr, columns=columns)
```

In [35]:
```
new_df
```

Out[35]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnan
0	41.0	0.0	0.0	0.0	0.0	0.0	0.(
1	23.0	0.0	0.0	0.0	0.0	0.0	0.(
2	46.0	1.0	0.0	0.0	0.0	0.0	0.(
3	70.0	0.0	1.0	0.0	0.0	0.0	0.(
4	70.0	0.0	0.0	0.0	0.0	0.0	0.(
...	...	...	...	...	...	...	..
3767	30.0	0.0	0.0	0.0	0.0	0.0	0.(
3768	68.0	0.0	0.0	0.0	0.0	0.0	0.(
3769	74.0	0.0	0.0	0.0	0.0	0.0	0.(
3770	72.0	1.0	0.0	0.0	0.0	0.0	0.(
3771	64.0	0.0	0.0	0.0	0.0	0.0	0.(

3772 rows × 22 columns

In [39]:
```
columns = ['age','TSH','T3','TT4','T4U','FTI']

plot.figure(figsize=(10,15),facecolor='white')
plotnumber = 1

for column in columns:
    ax = plot.subplot(3,2,plotnumber)
    sns.distplot(new_df[column])
    plot.xlabel(column,fontsize=10)
    plotnumber+=1
plot.show()
```

```
similar flexibility) or `histplot` (an axes-level function for histogr
ams).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(new_df[column])
C:\Users\Dinesh\AppData\Local\Temp\ipykernel_39476\1990233037.py:8: Us
erWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.
14.0.

Please adapt your code to use either `displot` (a figure-level functio
n with
similar flexibility) or `histplot` (an axes-level function for histogr
ams).

For a guide to updating your code to use the new functions, please see
```

In [40]:
```
plot.figure(figsize=(10,15),facecolor='white')
plotnumber = 1

for column in columns:
    new_df[column]+=1
    ax = plot.subplot(3,2,plotnumber)
    sns.distplot(np.log(new_df[column]))
    plot.xlabel(column,fontsize=10)
    plotnumber+=1
plot.show()
```

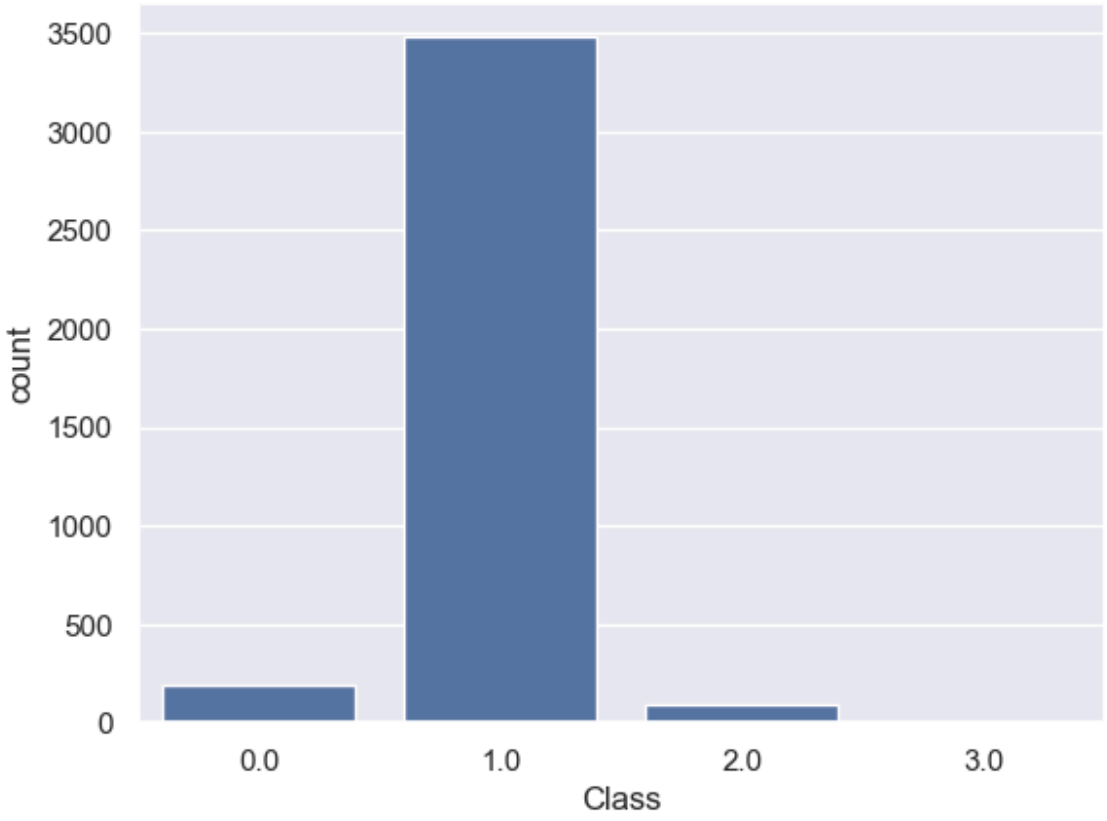
```
C:\Users\Dinesh\AppData\Local\Temp\ipykernel_39476\3553379851.py:7: Us
erWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.
14.0.

Please adapt your code to use either `displot` (a figure-level functio
n with
similar flexibility) or `histplot` (an axes-level function for histogr
ams).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(np.log(new_df[column]))
C:\Users\Dinesh\AppData\Local\Temp\ipykernel_39476\3553379851.py:7: Us
erWarning:
```

In [41]:
```
new_df = new_df.drop(columns="TSH")
```

In [43]:
```
new_df.shape
```

Out[43]:
```
(3772, 21)
```

In [52]:
```
sns.countplot(data = new_df, x = "Class")
plot.show()
```



In [54]:
```
x = new_df.drop(["Class"], axis = 1)
y = new_df["Class"]
rdsmple = RandomOverSampler()
x_sampled, y_sampled = rdsmple.fit_resample(x, y)
```

In [55]:
```
x_sampled.shape
```

Out[55]: (13924, 20)

In [56]:
```
y_sampled.shape
```

Out[56]: (13924,)

In [62]:
```
df_y = pd.DataFrame(y_sampled, columns=["Class"])
```

In [63]:
```
df_y
```

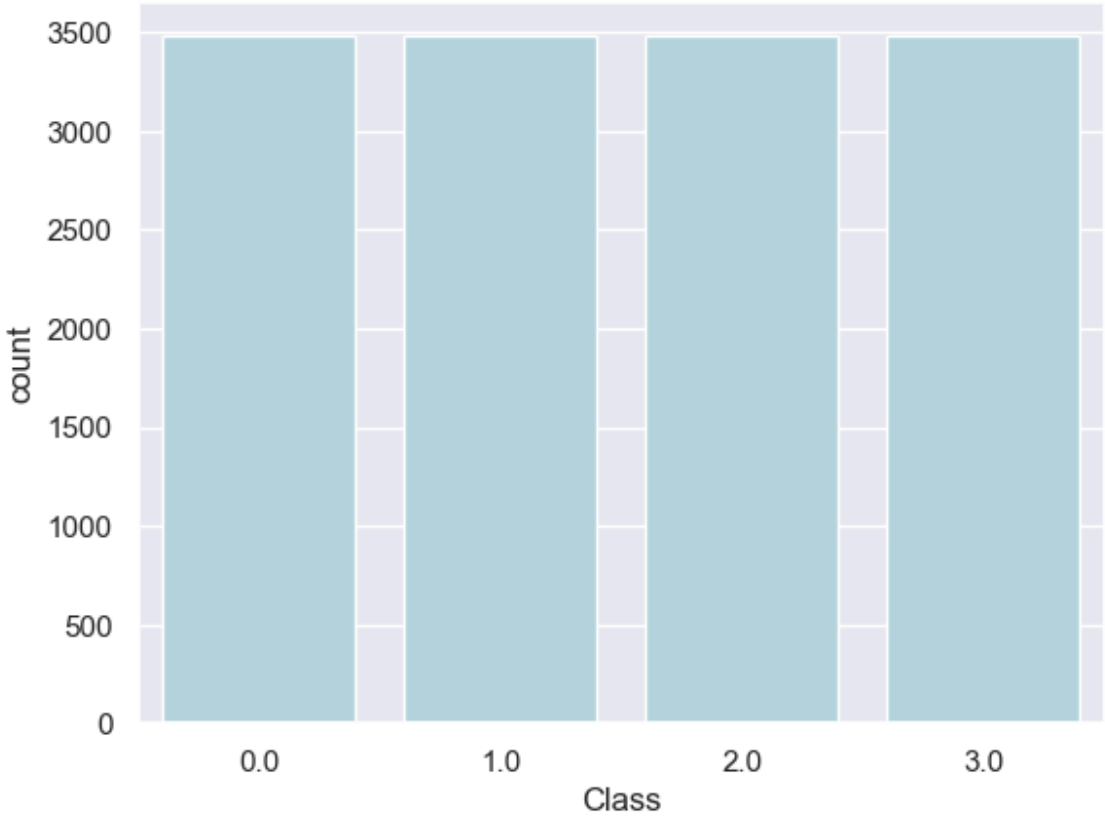
Out[63]:

	Class
0	1.0
1	1.0
2	1.0
3	1.0
4	1.0
...	...
13919	3.0
13920	3.0
13921	3.0
13922	3.0
13923	3.0

13924 rows × 1 columns

In [67]:
```
sns.countplot(data = df_y, x = "Class", color="lightblue")
```

Out[67]:
```
<AxesSubplot: xlabel='Class', ylabel='count'>
```



In [68]:
```
df_x = pd.DataFrame(x_sampled, columns = x.columns)
```

In [69]:
```
df_x
```

Out[69]:

nant	thyroid_surgery	I131_treatment	query_hypothyroid	query_hyperthyroid	lithium	goitre
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...
0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0

In [71]:
```
merged_df = pd.merge(df_x, df_y, left_index=True, right_index=True)
```

In [72]:
```
merged_df
```

Out[72]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregna
0	42.0	0.0	0.0	0.0	0.0	0.0	0
1	24.0	0.0	0.0	0.0	0.0	0.0	0
2	47.0	1.0	0.0	0.0	0.0	0.0	0
3	71.0	0.0	1.0	0.0	0.0	0.0	0
4	71.0	0.0	0.0	0.0	0.0	0.0	0
...	...	...	...	...	...	...	
13919	42.0	1.0	0.0	0.0	0.0	0.0	0
13920	47.0	0.0	0.0	0.0	0.0	0.0	0
13921	42.0	1.0	0.0	0.0	0.0	0.0	0
13922	47.0	0.0	0.0	0.0	0.0	0.0	0
13923	47.0	0.0	0.0	0.0	0.0	0.0	0

13924 rows × 21 columns

In [81]:
```
merged_df.to_csv("Data/preprocessed_data.csv", index=False)
```

In [76]:
```
# for saving the label encoder for model output printing to front end
from joblib import dump
dump(lbn, "Encoder/label_encoder_class.joblib")
```

Out[76]:
```
['Encoder/label_encoder_class.joblib']
```

In [ ]:

In [ ]:

Clustering

In [1]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from kneed import KneeLocator
import pandas as pda
```

```
C:\Users\Dinesh\AppData\Roaming\Python\Python39\site-packages\scipy\__init__.py:177: UserWarning: A NumPy version >=1.18.5 and <1.26.0 is required for this version of SciPy (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
df = pd.read_csv('Data/preprocessed_data.csv')
```

In [3]:
```
df
```

Out[3]:

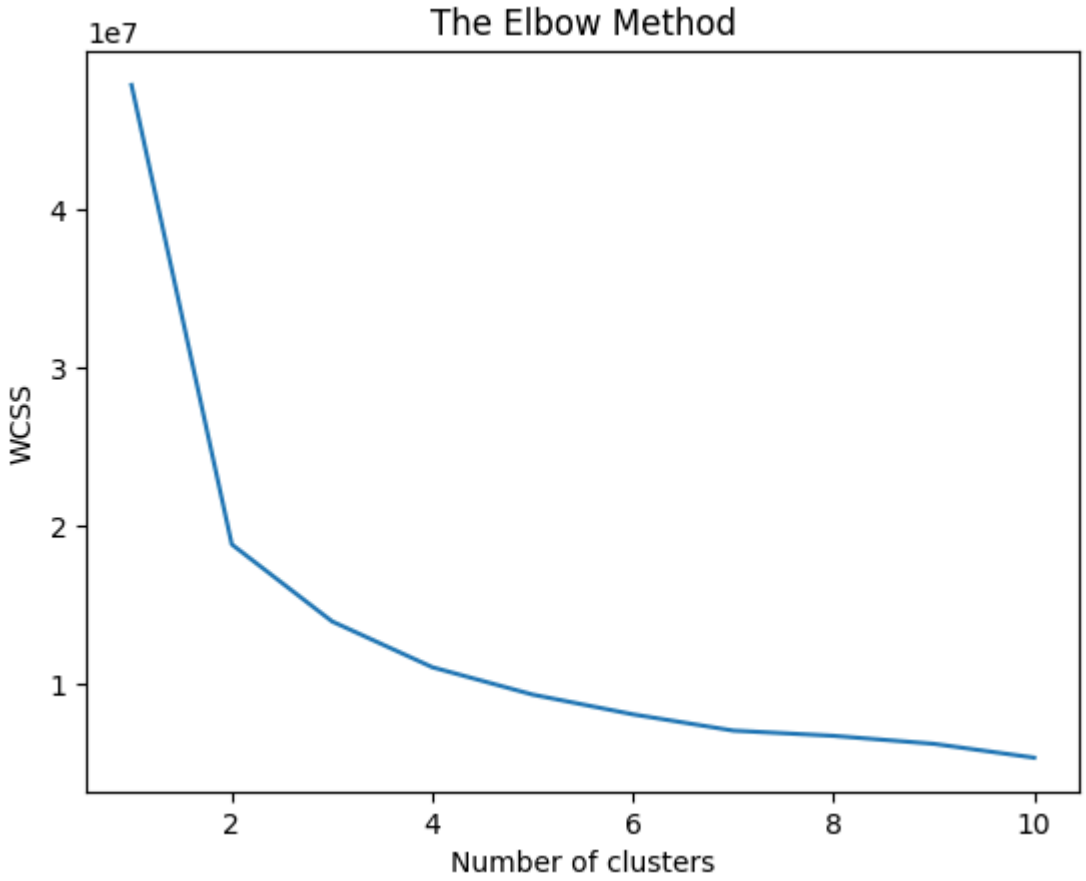
	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregna
0	42.0	0.0	0.0	0.0	0.0	0.0	0
1	24.0	0.0	0.0	0.0	0.0	0.0	0
2	47.0	1.0	0.0	0.0	0.0	0.0	0
3	71.0	0.0	1.0	0.0	0.0	0.0	0
4	71.0	0.0	0.0	0.0	0.0	0.0	0
...	...	...	...	...	...	...	
13919	42.0	1.0	0.0	0.0	0.0	0.0	0
13920	47.0	0.0	0.0	0.0	0.0	0.0	0
13921	42.0	1.0	0.0	0.0	0.0	0.0	0
13922	47.0	0.0	0.0	0.0	0.0	0.0	0
13923	47.0	0.0	0.0	0.0	0.0	0.0	0

13924 rows × 21 columns

In [4]:
```
X = df.drop(["Class"], axis = 1)
y = df["Class"]
```

In [5]:
```
wcss = [] #within cluster sum of square
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

In [9]:
```
plt.plot(range(1,11),wcss) # creating the graph between WCSS and the number
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [11]:
```
kn = KneeLocator(range(1, 11), wcss, curve='convex', direction='decreasing'
num_cluster = kn.knee
```

In [14]:
```
#creating the number of cluster using KMeans++ as the number of cluster is
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
```

In [15]:
```
y_kmeans = kmeans.fit_predict(X)
```

In [16]:
```
# created the cluster and stored the number of cluster
y_kmeans
```

Out[16]:
```
array([2, 0, 2, ..., 1, 1, 1])
```

In [17]:
```
X['Cluster'] = y_kmeans
```

In [18]:
```
X.head()
```

Out[18]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	t
0	42.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	24.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	47.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	71.0	0.0	1.0	0.0	0.0	0.0	0.0	
4	71.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 21 columns

In [19]:
```
X["Label"] = y
```

In [20]:
```
X.head()
```

Out[20]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	t
0	42.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	24.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	47.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	71.0	0.0	1.0	0.0	0.0	0.0	0.0	
4	71.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 22 columns

In [21]:
```
X.to_csv("Data/Cluster_data.csv", index = False)
```

In [22]:
```
import pickle
```

In [24]:
```
with open("Cluster_model/clustering.pkl", 'wb') as f:
    pickle.dump(kmeans, f)
```

In [ ]:
```

```

Model_Selection_Training_Cluster_1

In [2]:
```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics  import roc_auc_score,accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
import pickle
```

```
C:\Users\Dinesh\AppData\Roaming\Python\Python39\site-packages\scipy\__ini
t__.py:177: UserWarning: A NumPy version >=1.18.5 and <1.26.0 is required
for this version of SciPy (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [3]:
```
df = pd.read_csv("Data/Cluster_data.csv")
```

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

Out[4]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	t
0	42.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	24.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	47.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	71.0	0.0	1.0	0.0	0.0	0.0	0.0	
4	71.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 22 columns

In [5]:
```
list_of_clusters = df["Cluster"].unique()
```

In [6]:
```
list_of_clusters
```

Out[6]:
```
array([2, 0, 1], dtype=int64)
```

In [7]:
```
cluster_data_2 = df[df["Cluster"] == list_of_clusters[2]]
```

In [8]:
```
cluster_data_2.head()
```

Out[8]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant
40	45.0	1.0	0.0	0.0	0.0	0.0	0.0
88	40.0	0.0	0.0	0.0	0.0	0.0	0.0
89	50.0	0.0	0.0	0.0	0.0	0.0	0.0
91	81.0	1.0	0.0	0.0	0.0	0.0	0.0
116	51.0	1.0	0.0	0.0	0.0	0.0	0.0

5 rows × 22 columns

In [9]:
```
cluster_feature_2 = cluster_data_2.drop(["Cluster", "Label"], axis = 1)
cluster_label_2 = cluster_data_2["Label"]
```

In [10]:
```
x_train, x_test, y_train, y_test = train_test_split(cluster_feature_2, clus
```

Random Forest

In [11]:
```
clf = RandomForestClassifier()
```

In [12]:
```
clf.fit(x_train, y_train)
```

Out[12]:
```
RandomForestClassifier()
```
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 [13]:
```
clf.score(x_test, y_test)
```

Out[13]:
```
0.9965936739659368
```

In [14]:
```
clf.score(x_train, y_train)
```

Out[14]:
```
1.0
```

In [15]:
```
param_clf = {"n_estimators": [10, 50, 100, 130, 160, 200, 250],
             "criterion": ['gini', 'entropy'],
            "max_depth": range(2, 8, 1),
             "max_features": ['sqrt', 'log2']
}
```

In [16]:
```
grid = GridSearchCV(estimator=clf, param_grid=param_clf, cv=10, n_jobs=-1,
```

In [16]:
```
grid.fit(x_train, y_train)
```

Out[16]:
```
GridSearchCV(cv=10, error_score='raise', estimator=RandomForestClassifier
(),
             n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 8),
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [10, 50, 100, 130, 160, 200, 25
0]})
```

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 [17]:
```
grid.best_params_
```

Out[17]:
```
{'criterion': 'entropy',
 'max_depth': 7,
 'max_features': 'sqrt',
 'n_estimators': 10}
```

In [17]:
```
clf = RandomForestClassifier(criterion='entropy', max_depth=7, max_features
```

In [18]:
```
clf.fit(x_train, y_train)
```

Out[18]:
```
RandomForestClassifier(criterion='entropy', max_depth=7, n_estimators=10)
```

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 [19]:
```
clf.score(x_test, y_test)
```

Out[19]:
```
0.9961070559610705
```

In [20]:
```
clf.score(x_train, y_train)
```

Out[20]:
```
0.9979136240350511
```

In [22]:
```
prediction_clf = clf.predict_proba(x_test)
```

In [25]:
```
if len(y_test.unique()) == 1:
    clf_score = accuracy_score(y_test, prediction_clf)
else:
    clf_score = roc_auc_score(y_test, prediction_clf, multi_class="ovr")
```

In [26]:
```
clf_score
```

Out[26]:
```
0.9932519073016862
```

KNN

In [27]:
```
knn = KNeighborsClassifier()
```

In [28]:
```
knn.fit(x_train, y_train)
```

Out[28]:
```
KNeighborsClassifier()
```
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 [29]:
```
knn.score(x_test, y_test)
```

Out[29]:
```
0.9951338199513382
```

In [30]:
```
param_knn = {
    'n_neighbors' : [i for i in range(5, 25)],
    'algorithm' : ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'leaf_size' : [10, 15, 20, 25, 30, 35, 40, 50],
    'p' : [1,2],
    'weights' : ['uniform', 'distance']
}
```

In [44]:
```
grid_knn = GridSearchCV(estimator=knn, param_grid=param_knn, cv = 10, error
```

In [45]:
```
grid_knn.fit(x_train, y_train)
```

Out[45]:
```
GridSearchCV(cv=10, error_score='raise', estimator=KNeighborsClassifier
(),
             n_jobs=-1,
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'b
rute'],
                         'leaf_size': [10, 15, 20, 25, 30, 35, 40, 50],
                         'n_neighbors': [5, 6, 7, 8, 9, 10, 11, 12, 13, 1
4, 15,
                                         16, 17, 18, 19, 20, 21, 22, 23,
24],
                         'p': [1, 2], 'weights': ['uniform', 'distanc
e']})
```

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 [46]:
```
grid_knn.best_params_
```

Out[46]:
```
{'algorithm': 'auto',
 'leaf_size': 10,
 'n_neighbors': 5,
 'p': 1,
 'weights': 'distance'}
```

In [31]:
```
knn = KNeighborsClassifier(algorithm='auto', leaf_size=10, n_neighbors = 5,
```

In [32]:
```
knn.fit(x_test, y_test)
```

Out[32]:
```
KNeighborsClassifier(leaf_size=10, p=1, weights='distance')
```

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 [33]:
```
knn.score(x_train, y_train)
```

Out[33]:
```
0.9912372209472147
```

In [34]:
```
knn.score(x_test, y_test)
```

Out[34]:
```
1.0
```

In [35]:
```
prediction_score = knn.predict_proba(x_test)
```

In [36]:
```
if len(y_test.unique()) == 1:
    knn_score = accuracy_score(y_test, prediction_score)
else:
    knn_score = roc_auc_score(y_test, prediction_score, multi_class="ovr")
```

In [37]:
```
knn_score
```

Out[37]:
```
1.0
```

SVM

In [39]:
```
svm = SVC()
```

In [40]:
```
svm.fit(x_train, y_train)
```

Out[40]:
```
SVC()
```
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 [41]:
```
svm.score(x_test, y_test)
```

Out[41]:
```
0.9712895377128954
```

In [42]:
```
svm.score(x_train, y_train)
```

Out[42]:
```
0.968078447736282
```

In [43]:
```
param_svc = {
    'kernel' :['linear', 'poly', 'rbf', 'sigmoid']
}
```

In [63]:
```
grid_svc = GridSearchCV(estimator=svm, param_grid=param_svc, cv = 10, n_jo
```

In [64]:
```
grid_svc.fit(x_train, y_train)
```

Out[64]:
```
GridSearchCV(cv=10, error_score='raise', estimator=SVC(), n_jobs=-1,
             param_grid={'kernel': ['linear', 'poly', 'rbf', 'sigmoid']})
```
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 [65]:
```
grid_svc.best_params_
```

Out[65]:
```
{'kernel': 'linear'}
```

In [44]:
```
svm = SVC(kernel = 'linear')
```

In [45]:
```
svm.fit(x_train, y_train)
```

Out[45]:
```
SVC(kernel='linear')
```
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 [46]:
```
svm.score(x_test, y_test)
```

Out[46]:
```
0.9664233576642336
```

In [47]:
```
svm.score(x_train, y_train)
```

Out[47]:
```
0.9609847694554559
```

In [58]:
```
prediction_svm = svm.score(x_test, y_test)
```

In [59]:
```
prediction_svm
```

Out[59]:
```
0.9664233576642336
```

In [ ]:
```

```

Decision Tree

In [60]:
```
dt = DecisionTreeClassifier()
```

In [61]:
```
dt.fit(x_train, y_train)
```

Out[61]:
```
DecisionTreeClassifier()
```
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 [62]:
```
dt.score(x_test, y_test)
```

Out[62]:
```
0.9965936739659368
```

In [63]:
```
dt.score(x_train, y_train)
```

Out[63]:
```
1.0
```

In [64]:
```
param_dt = {"criterion" : ['gini', 'entropy'],
            "splitter" : ['best', 'random'],
            "max_depth" : range(2, 40, 1),
            "min_samples_split" : range(2, 10, 1),
            "min_samples_leaf" : range(1, 10, 1)
}
```

In [65]:
```
grid_dt = GridSearchCV(estimator=dt, param_grid=param_dt, n_jobs=3, cv = 10
```

In [66]:
```
grid_dt.fit(x_train, y_train)
```

Out[66]:
```
GridSearchCV(cv=10, error_score='raise', estimator=DecisionTreeClassifier
(),
             n_jobs=3,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 40),
                         'min_samples_leaf': range(1, 10),
                         'min_samples_split': range(2, 10),
                         'splitter': ['best', 'random']})
```

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 [68]:
```
grid_dt.best_params_
```

Out[68]:
```
{'criterion': 'gini',
 'max_depth': 33,
 'min_samples_leaf': 1,
 'min_samples_split': 5,
 'splitter': 'random'}
```

In [69]:
```
dt = DecisionTreeClassifier(criterion='gini', max_depth=33, min_samples_lea
```

In [71]:
```
dt.fit(x_train, y_train)
```

Out[71]:
```
DecisionTreeClassifier(max_depth=33, min_samples_split=5, splitter='rando
m')
```

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 [72]:
```
dt.score(x_test, y_test)
```

Out[72]:
```
0.9975669099756691
```

In [73]:
```
dt.score(x_train, y_train)
```

Out[73]:
```
1.0
```

In [74]:
```
prediction_dt = dt.predict_proba(x_test)
```

In [76]:
```
dt_score = roc_auc_score(y_test, prediction_dt, multi_class='ovr')
```

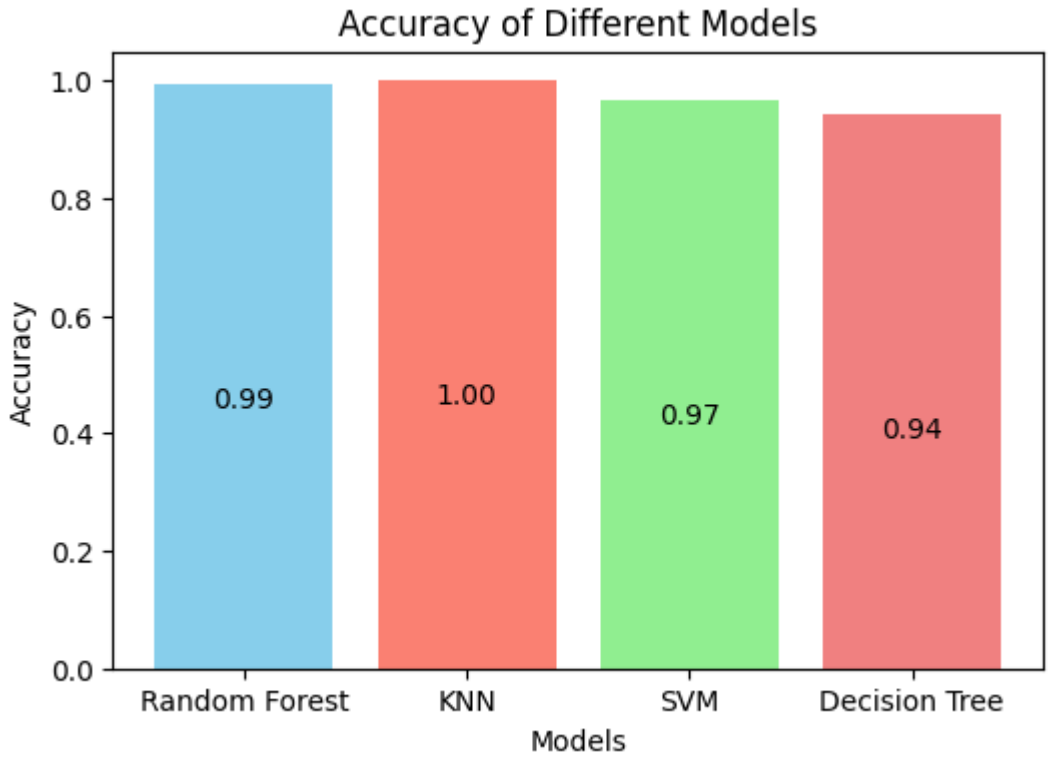
In [77]:
```
dt_score
```

Out[77]:
```
0.9426115627073657
```

Model Accuracy

In [79]:
```
import matplotlib.pyplot as plt
```

In [102]:
```
models = ['Random Forest', 'KNN', 'SVM', 'Decision Tree']

accuracies = [clf_score, knn_score, prediction_svm, dt_score]
colors = ['skyblue', 'salmon', 'lightgreen', 'lightcoral']


plt.figure(figsize=(6, 4))
bars = plt.bar(models, accuracies, color=colors)
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy of Different Models')
for bar, acc in zip(bars, accuracies):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() - 0.55, f'
plt.show()
```



Here the KNN is performing well on the cluster data 2 so we are using knn to predict the data

In [101]:
```
with open("Models/KNN2/knn.pkl", 'wb') as f:
    pickle.dump(knn, f)
```

In [ ]:
```

```

Model_Selection_Training_Cluster_2_0

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics  import roc_auc_score,accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import pickle
```

```
C:\Users\Dinesh\AppData\Roaming\Python\Python39\site-packages\scipy\__init__.py:177: UserWarning: A NumPy version >=1.18.5 and <1.26.0 is required for this version of SciPy (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
df = pd.read_csv("Data/Cluster_data.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	t
0	42.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	24.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	47.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	71.0	0.0	1.0	0.0	0.0	0.0	0.0	
4	71.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 22 columns

In [4]:
```
list_of_clusters = df["Cluster"].unique()
```

In [5]:
```
list_of_clusters
```

Out[5]:
```
array([2, 0, 1], dtype=int64)
```

Cluster 2

In [6]:
```
cluster_data_2 = df[df["Cluster"] == list_of_clusters[0]]
```

In [7]:
```
cluster_data_2.head()
```

Out[7]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	t
0	42.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	47.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	71.0	0.0	1.0	0.0	0.0	0.0	0.0	
5	19.0	0.0	1.0	0.0	0.0	0.0	0.0	
8	67.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 22 columns

In [8]:
```
cluster_feature_2 = cluster_data_2.drop(["Cluster", "Label"], axis = 1)
cluster_label_2 = cluster_data_2["Label"]
```

In [9]:
```
x_train, x_test, y_train, y_test = train_test_split(cluster_feature_2, clus
```

Random Forest

In [10]:
```
clf = RandomForestClassifier()
```

In [11]:
```
clf.fit(x_train, y_train)
```

Out[11]:
```
RandomForestClassifier()
```
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 [12]:
```
clf.score(x_test, y_test)
```

Out[12]:
```
0.9928774928774928
```

In [13]:
```
clf.score(x_train, y_train)
```

Out[13]:
```
1.0
```

In [14]:
```
param_clf = {"n_estimators": [10, 50, 100, 130, 160, 200, 250],
             "criterion": ['gini', 'entropy'],
            "max_depth": range(2, 8, 1),
             "max_features": ['sqrt', 'log2']
}
```

In [17]:
```
grid = GridSearchCV(estimator=clf, param_grid=param_clf, cv=10, n_jobs=-1,
```

In [23]:
```
grid.fit(x_train, y_train)
```

Out[23]:
```
GridSearchCV(cv=10, error_score='raise', estimator=RandomForestClassifier
(),
             n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 8),
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [10, 50, 100, 130, 160, 200, 25
0]})
```

In [24]:
```
grid.best_params_
```

Out[24]:
```
{'criterion': 'entropy',
 'max_depth': 7,
 'max_features': 'sqrt',
 'n_estimators': 100}
```

In [15]:
```
clf = RandomForestClassifier(criterion='entropy', max_depth=7, max_features
```

In [16]:
```
clf.fit(x_train, y_train)
```

Out[16]:
```
RandomForestClassifier(criterion='entropy', max_depth=7)
```

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 [17]:
```
clf.score(x_test, y_test)
```

Out[17]:
```
0.886039886039886
```

In [18]:
```
clf.score(x_train, y_train)
```

Out[18]:
```
0.9016493585827734
```

In [19]:
```
prediction_clf = clf.predict_proba(x_test)
if len(y_test.unique()) == 1:
    clf_score_2 = accuracy_score(y_test, prediction_clf)
else:
    clf_score_2 = roc_auc_score(y_test, prediction_clf, multi_class="ovr")
clf_score_2
```

Out[19]:
```
0.9848749197961503
```

KNN

In [20]:
```
knn = KNeighborsClassifier()
```

In [21]:
```
knn.fit(x_train, y_train)
```

Out[21]:
```
KNeighborsClassifier()
```
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 [22]:
```
knn.score(x_test, y_test)
```

Out[22]:
```
0.9301994301994302
```

In [23]:
```
knn.score(x_train, y_train)
```

Out[23]:
```
0.9578497251069029
```

In [48]:
```
param_knn = {
    'n_neighbors' : [i for i in range(5, 25)],
    'algorithm' : ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'leaf_size' : [8,9,11,12,21,24,23,45,67,78,],
    'p' : [1,2],
    'weights' : ['uniform','distance']
}
```

In [49]:
```
grid = GridSearchCV(estimator=knn, param_grid=param_knn, cv=10, n_jobs=-1,
```

In [50]:
```
grid.fit(x_train, y_train)
```

Out[50]:
```
GridSearchCV(cv=10, error_score='raise',
             estimator=KNeighborsClassifier(leaf_size=10, weights='distan
ce'),
             n_jobs=-1,
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'b
rute'],
                         'leaf_size': [8, 9, 11, 12, 21, 24, 23, 45, 67,
78],
                         'n_neighbors': [5, 6, 7, 8, 9, 10, 11, 12, 13, 1
4, 15,
                                         16, 17, 18, 19, 20, 21, 22, 23,
24],
                         'p': [1, 2], 'weights': ['uniform', 'distanc
e']})
```

In [51]:
```
grid.best_params_
```

Out[51]:
```
{'algorithm': 'auto',
 'leaf_size': 8,
 'n_neighbors': 5,
 'p': 2,
 'weights': 'distance'}
```

In [24]:
```
knn = KNeighborsClassifier(algorithm='auto', leaf_size=8, n_neighbors=5,p=2
```

In [25]:
```
knn.fit(x_train, y_train)
```

Out[25]:
```
KNeighborsClassifier(leaf_size=8, weights='distance')
```
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 [26]:
```
knn.score(x_test, y_test)
```

Out[26]:
```
0.9387464387464387
```

In [27]:
```
knn.score(x_train,y_train)
```

Out[27]:
```
1.0
```

In [28]:
```
prediction_score = knn.predict_proba(x_test)
if len(y_test.unique()) == 1:
    knn_score_2 = accuracy_score(y_test, prediction_score)
else:
    knn_score_2 = roc_auc_score(y_test, prediction_score, multi_class="ovr'
```

SVM

In [29]:
```
svm = SVC()
```

In [30]:
```
svm.fit(x_train, y_train)
```

Out[30]:
```
SVC()
```
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 [31]:
```
svm.score(x_test, y_test)
```

Out[31]:
```
0.6723646723646723
```

In [32]:
```
svm.score(x_train,y_train)
```

Out[32]:
```
0.6945632254123396
```

In [70]:
```
param_svm = {
    'kernel' :['linear', 'poly', 'rbf', 'sigmoid']
}
```

In [71]:
```
grid = GridSearchCV(estimator=svm, param_grid=param_svm, cv=10, n_jobs=3, e
```

In [72]:
```
grid.fit(x_train, y_train)
```

Out[72]:
```
GridSearchCV(cv=10, error_score='raise', estimator=SVC(), n_jobs=3,
             param_grid={'kernel': ['linear', 'poly', 'rbf', 'sigmoid']})
```

In [73]:
```
grid.best_params_
```

Out[73]:
```
{'kernel': 'linear'}
```

In [33]:
```
svm = SVC(kernel='linear')
```

In [34]:
```
svm.fit(x_train, y_train)
```

Out[34]:
```
SVC(kernel='linear')
```
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 [35]:
```
svm.score(x_test, y_test)
```

Out[35]:
```
0.7165242165242165
```

In [36]:
```
svm.score(x_train, y_train)
```

Out[36]:
```
0.7532070861331704
```

In [37]:
```
prediction_svm_2 = svm.score(x_test,y_test)
```

Decision Tree

In [38]:
```
dt = DecisionTreeClassifier()
```

In [39]:
```
dt.fit(x_train, y_train)
```

Out[39]:
```
DecisionTreeClassifier()
```
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 [40]:
```
dt.score(x_test, y_test)
```

Out[40]:
```
0.9672364672364673
```

In [41]:
```
dt.score(x_train, y_train)
```

Out[41]:
```
1.0
```

In [87]:
```
parm_dt = {"criterion" : ['gini', 'entropy'],
              "splitter" : ['best', 'random'],
              "max_depth" : range(2, 40, 1),
              "min_samples_split" : range(2, 10, 1),
              "min_samples_leaf" : range(1, 10, 1)}
```

In [94]:
```
grid = GridSearchCV(estimator=dt, param_grid=parm_dt, cv=10, n_jobs=3, erro
```

In [95]:
```
grid.fit(x_train, y_train)
```

Out[95]:
```
GridSearchCV(cv=10, error_score='raise', estimator=DecisionTreeClassifier
(),
             n_jobs=3,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 40),
                         'min_samples_leaf': range(1, 10),
                         'min_samples_split': range(2, 10),
                         'splitter': ['best', 'random']})
```

In [96]:
```
grid.best_params_
```

Out[96]:
```
{'criterion': 'gini',
 'max_depth': 35,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'splitter': 'random'}
```

In [42]:
```
dt = DecisionTreeClassifier(criterion='gini', max_depth= 35, min_samples_le
```

In [43]:
```
dt.fit(x_train, y_train)
```

Out[43]:
```
DecisionTreeClassifier(max_depth=35, splitter='random')
```

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 [44]:
```
dt.score(x_test, y_test)
```

Out[44]:
```
0.9786324786324786
```

In [45]:
```
dt.score(x_train, y_train)
```

Out[45]: 1.0

In [46]:
```
prediction_dt = dt.predict_proba(x_test)
dt_score_2 = roc_auc_score(y_test, prediction_dt, multi_class='ovr')
```

Cluster 0

In [47]:
```
cluster_data_0 = df[df["Cluster"] == list_of_clusters[1]]
```

In [48]:
```
cluster_data_0.head()
```

Out[48]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	t
1	24.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	71.0	0.0	0.0	0.0	0.0	0.0	0.0	
6	60.0	0.0	0.0	0.0	0.0	0.0	0.0	
7	81.0	0.0	0.0	0.0	0.0	0.0	0.0	
9	69.0	1.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 22 columns

In [49]:
```
cluster_features_0 = cluster_data_0.drop(columns = ['Cluster','Label'],axis
```

In [50]:
```
cluster_label_0 = cluster_data_0['Label']
```

In [51]:
```
x_train, x_test, y_train,y_test = train_test_split(cluster_features_0, clus
```

Random Forest

In [52]:
```
clf_0 = RandomForestClassifier()
```

In [53]:
```
clf_0.fit(x_train, y_train)
```

Out[53]: RandomForestClassifier()

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 [54]:
```
clf_0.score(x_test, y_test)
```

Out[54]:
```
0.9922644163150492
```

In [55]:
```
clf_0.score(x_train, y_train)
```

Out[55]:
```
1.0
```

In [56]:
```
param_clf_0 = {"n_estimators" : [10, 20, 67, 240, 110, 100, 250],
               "criterion" : ['gini', 'entropy'],
               "max_depth" : range(2,8,1),
               "max_features" : ['sqrt', 'log2']
}
```

In [120]:
```
grid = GridSearchCV(estimator =clf_0, param_grid = param_clf_0,cv= 10, n_jo
```

In [121]:
```
grid.fit(x_train, y_train)
```

Out[121]:
```
GridSearchCV(cv=10, error_score='raise', estimator=RandomForestClassifier
(),
             n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 8),
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [10, 20, 67, 240, 110, 100, 25
0]})
```

In [122]:
```
grid.best_params_
```

Out[122]:
```
{'criterion': 'gini',
 'max_depth': 7,
 'max_features': 'log2',
 'n_estimators': 250}
```

In [57]:
```
clf_0 = RandomForestClassifier(criterion= 'gini', max_depth= 7,max_features
```

In [58]:
```
clf_0.fit(x_train, y_train)
```

Out[58]:
```
RandomForestClassifier(max_depth=7, max_features='log2', n_estimators=25
0)
```

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 [59]:
```
clf_0.score(x_test, y_test)
```

Out[59]:
```
0.8291139240506329
```

In [60]:
```
clf_0.score(x_train, y_train)
```

Out[60]:
```
0.8395173453996984
```

In [61]:
```
prediction_clf = clf_0.predict_proba(x_test)
if len(y_test.unique()) == 1:
    clf_score_0 = accuracy_score(y_test, prediction_clf)
else:
    clf_score_0 = roc_auc_score(y_test, prediction_clf, multi_class="ovr")
clf_score_0
```

Out[61]:
```
0.934690768293213
```

KNN

In [62]:
```
knn_0 = KNeighborsClassifier()
```

In [63]:
```
knn_0.fit(x_train, y_train)
```

Out[63]:
```
KNeighborsClassifier()
```
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 [64]:
```
knn_0.score(x_test, y_test)
```

Out[64]:
```
0.9205344585091421
```

In [65]:
```
knn_0.score(x_train, y_train)
```

Out[65]:
```
0.9282051282051282
```

In [132]:
```
param_knn_0 = {'n_neighbors' : [i for i in range(5, 25)],
    'algorithm' : ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'leaf_size' : [8,9,11,12,21,50,20,60,70,100],
    'p' : [1,2],
    'weights' : ['uniform','distance']

}
```

In [133]:
```
grid = GridSearchCV(estimator = knn_0, param_grid = param_knn_0, cv= 10, n_
```

In [137]:
```
grid.fit(x_train, y_train)
```

Out[137]:
```
GridSearchCV(cv=10, error_score='raise', estimator=KNeighborsClassifier
(),
             n_jobs=-1,
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'b
rute'],
                         'leaf_size': [8, 9, 11, 12, 21, 50, 20, 60, 70,
100],
                         'n_neighbors': [5, 6, 7, 8, 9, 10, 11, 12, 13, 1
4, 15,
                                         16, 17, 18, 19, 20, 21, 22, 23,
24],
                         'p': [1, 2], 'weights': ['uniform', 'distanc
e']})
```

In [138]:
```
grid.best_params_
```

Out[138]:
```
{'algorithm': 'auto',
 'leaf_size': 8,
 'n_neighbors': 5,
 'p': 1,
 'weights': 'distance'}
```

In [66]:
```
knn_0 = KNeighborsClassifier(algorithm= 'auto', leaf_size= 8, n_neighbors=
```

In [67]:
```
knn_0.fit(x_train, y_train)
```

Out[67]:
```
KNeighborsClassifier(leaf_size=8, p=1, weights='distance')
```
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 [68]:
```
knn_0.score(x_test, y_test)
```

Out[68]:
```
0.9240506329113924
```

In [69]:
```
knn_0.score(x_train, y_train)
```

Out[69]:
```
1.0
```

In [70]:
```
prediction_score = knn_0.predict_proba(x_test)
if len(y_test.unique()) == 1:
    knn_score_0 = accuracy_score(y_test, prediction_score)
else:
    knn_score_0 = roc_auc_score(y_test, prediction_score, multi_class="ovr'
```

SVM

In [71]:
```
svm_0 = SVC()
```

In [72]:
```
svm_0.fit(x_train, y_train)
```

Out[72]:
```
SVC()
```
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]:
```
svm_0.score(x_test, y_test)
```

Out[73]:
```
0.6315049226441631
```

In [74]:
```
svm_0.score(x_train, y_train)
```

Out[74]:
```
0.6132730015082957
```

In [148]:
```
param_svm_0 = {
    'kernel' :['linear', 'poly', 'rbf', 'sigmoid']
}
```

In [149]:
```
grid = GridSearchCV(estimator = svm_0, param_grid = param_svm_0, cv= 10, n_
```

In [150]:
```
grid.fit(x_train, y_train)
```

Out[150]:
```
GridSearchCV(cv=10, error_score='raise', estimator=SVC(), n_jobs=-1,
             param_grid={'kernel': ['linear', 'poly', 'rbf', 'sigmoid']})
```

In [151]:
```
grid.best_params_
```

Out[151]:
```
{'kernel': 'linear'}
```

In [75]:
```
svm_0 = SVC(kernel= 'linear')
```

In [76]:
```
svm_0.fit(x_train, y_train)
```

Out[76]:
```
SVC(kernel='linear')
```
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 [77]:
```
svm_0.score(x_test, y_test)
```

Out[77]:
```
0.6870604781997187
```

In [78]:
```
svm_0.score(x_train, y_train)
```

Out[78]:
```
0.6911010558069381
```

In [79]:
```
prediction_svm_0 = svm_0.score(x_test,y_test)
```

Decision Tree

In [80]:
```
dt_0 = DecisionTreeClassifier()
```

In [81]:
```
dt_0.fit(x_train, y_train)
```

Out[81]:
```
DecisionTreeClassifier()
```
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 [82]:
```
dt_0.score(x_test, y_test)
```

Out[82]:
```
0.9683544303797469
```

In [83]:
```
dt_0.score(x_train, y_train)
```

Out[83]:
```
1.0
```

In [163]:
```
param_dt_0 = { "criterion" : ['gini', 'entropy'],
               "splitter" : ['best', 'random'],
               "max_depth" : range(2, 40, 1),
               "min_samples_split" : range(2, 10, 1),
               "min_samples_leaf" : range(1, 10, 1)

}
```

In [164]:
```
grid = GridSearchCV(estimator = dt_0, param_grid = param_dt_0, cv= 10, n_jo
```

In [165]:
```
grid.fit(x_train, y_train)
```

Out[165]:
```
GridSearchCV(cv=10, error_score='raise', estimator=DecisionTreeClassifier
(),
             n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 40),
                         'min_samples_leaf': range(1, 10),
                         'min_samples_split': range(2, 10),
                         'splitter': ['best', 'random']})
```

In [166]:
```
grid.best_params_
```

Out[166]:
```
{'criterion': 'gini',
 'max_depth': 39,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'splitter': 'random'}
```

In [84]:
```
dt_0 = DecisionTreeClassifier(criterion= 'gini', max_depth= 39, min_samples
```

In [85]:
```
dt_0.fit(x_train, y_train)
```

Out[85]:
```
DecisionTreeClassifier(max_depth=39, splitter='random')
```
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 [86]:
```
dt_0.score(x_test, y_test)
```

Out[86]:
```
0.9669479606188467
```

In [87]:
```
dt_0.score(x_train,y_train)
```

Out[87]:
```
1.0
```

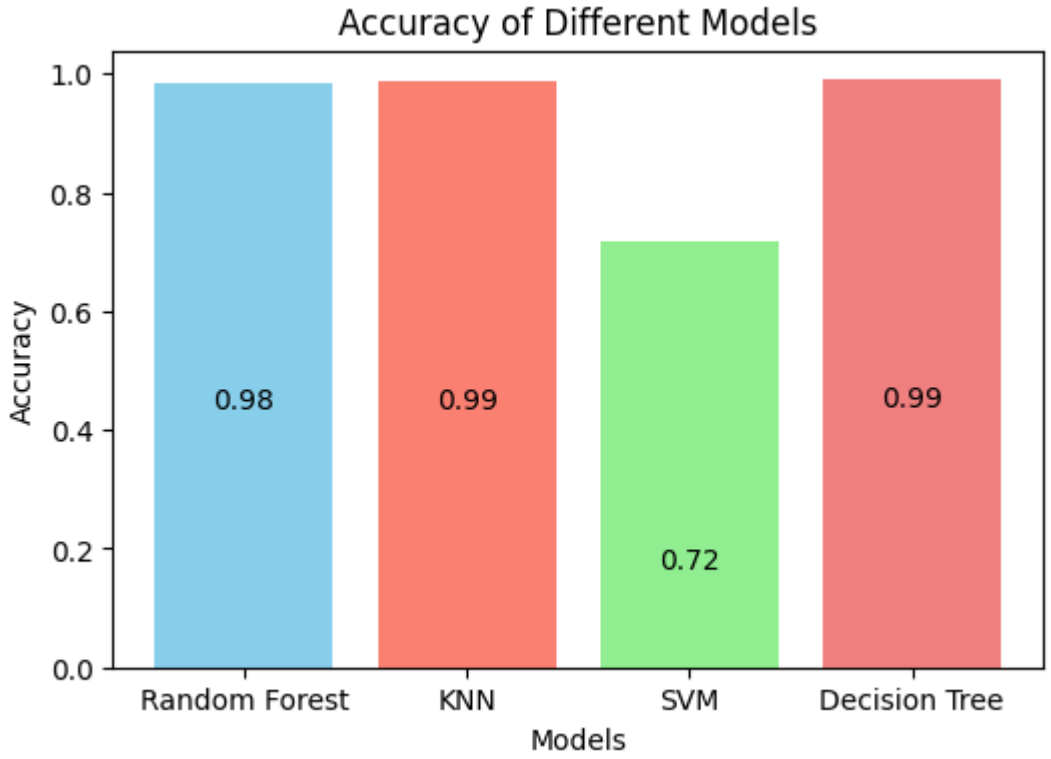
In [88]:
```
prediction_dt = dt_0.predict_proba(x_test)
dt_score_0 = roc_auc_score(y_test, prediction_dt, multi_class='ovr')
```

Model Accuracy

Cluster number 2

In [89]:
```
import matplotlib.pyplot as plt
```

In [90]:
```
models = ['Random Forest', 'KNN', 'SVM', 'Decision Tree']

accuracies = [clf_score_2, knn_score_2, prediction_svm_2, dt_score_2]
colors = ['skyblue', 'salmon', 'lightgreen', 'lightcoral']


plt.figure(figsize=(6, 4))
bars = plt.bar(models, accuracies, color=colors)
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy of Different Models')
for bar, acc in zip(bars, accuracies):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() - 0.55, f'
plt.show()
```

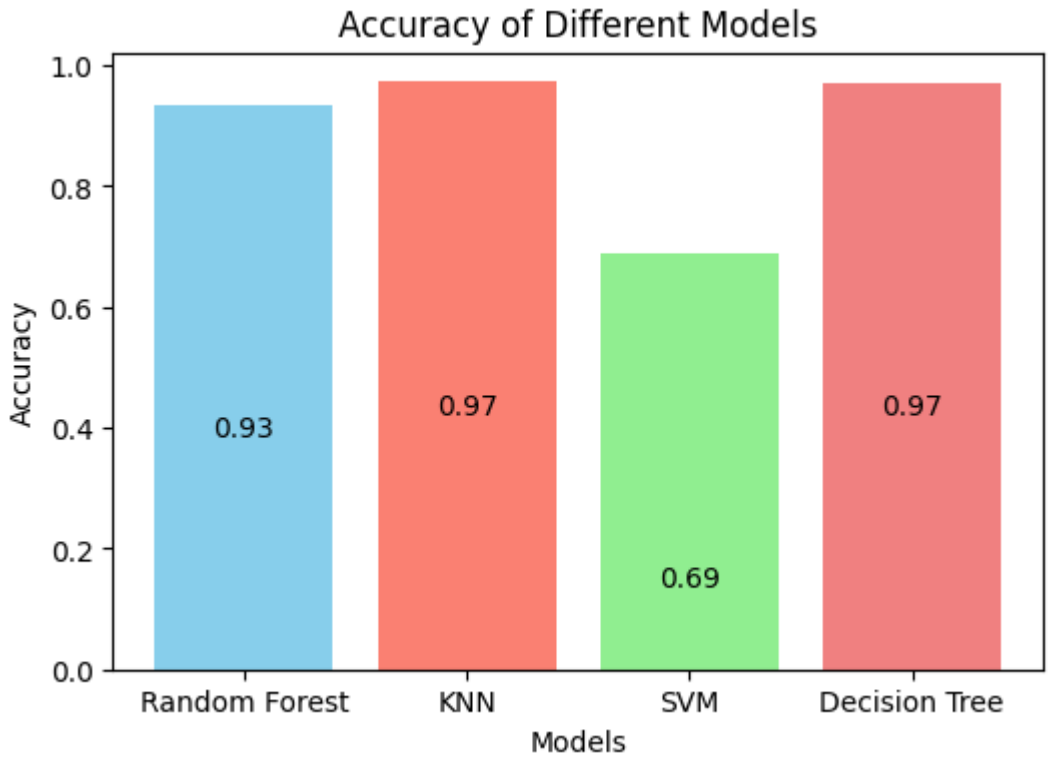


As Decision Tree has the highest accuracy score we will consider DT as our Model for cluster number 2

In [91]:
```
with open("dt.pkl", 'wb') as f:
    pickle.dump(dt,f)
```

Cluster number 0

In [92]:
```
models = ['Random Forest', 'KNN', 'SVM', 'Decision Tree']

accuracies = [clf_score_0, knn_score_0, prediction_svm_0, dt_score_0]
colors = ['skyblue', 'salmon', 'lightgreen', 'lightcoral']


plt.figure(figsize=(6, 4))
bars = plt.bar(models, accuracies, color=colors)
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Accuracy of Different Models')
for bar, acc in zip(bars, accuracies):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() - 0.55, f'
plt.show()
```



As we have chosen KNN and DT for other two clusters so we will be considering Random Forest for cluster number 0

In [94]:
```
with open("clf.pkl", 'wb') as f:
    pickle.dump(clf_0,f)
```

Flask API for Backend

```
from flask import Flask, request, render_template
import pickle
import os
import joblib

app = Flask(__name__)

kmeans = pickle.load(open("../Cluster_model/clustering.pkl", "rb"))

encoder = joblib.load(open("../Encoder/label_encoder_class.joblib", "rb"))

def find_model(cluster):
    if(cluster == 0):
        with open('../Models/CLF0/clf.pkl', 'rb') as f:
            model = pickle.load(f)
    elif(cluster == 1):
        with open('../Models/KNN1/knn.pkl', 'rb') as f:
            model = pickle.load(f)
    else:
        with open('../Models/DT2/dt.pkl', 'rb') as f:
            model = pickle.load(f)
    return model

@app.route("/", methods=['GET'])
def index():
    return render_template("main.html")

@app.route("/predict", methods=['POST'])
def predict():
    if request.method == 'POST':
        age = int(request.form["age"])
        T3 = float(request.form["T3"])
        TT4 = float(request.form["TT4"])
        T4U = float(request.form["T4U"])
        FTI = float(request.form["FTI"])
        sex_ = request.form['sex']
        if (sex_ == "Male"):
            sex = 1
        else:
            sex = 0

        thyroxine = request.form["on_thyroxine"]
        if (thyroxine == "true"):
            on_thyroxine = 1
        else:
            on_thyroxine = 0

        query_thyroxine = request.form["query_thyroxine"]
        if (query_thyroxine == "true"):
            query_on_thyroxine = 1
        else:
```

```
        query_on_thyroxine = 0

    antithyroid_medication = request.form["antithyroid_medication"]
    if( antithyroid_medication == "true"):
        on_antithyroid_medication = 1
    else:
        on_antithyroid_medication = 0

    I131_treatment_ = request.form["I131_treatment"]
    if(I131_treatment_ == "true"):
        I131_treatment = 1
    else:
        I131_treatment = 0

    query_hypothyroid_ = request.form["query_hypothyroid"]
    if(query_hypothyroid_ == "true"):
        query_hypothyroid = 1
    else:
        query_hypothyroid = 0

    hypopituitary_ = request.form["hypopituitary"]
    if(hypopituitary_ == "true"):
        hypopituitary = 1
    else:
        hypopituitary = 0

    psych_ = request.form["psych"]
    if(psych_ == "true"):
        psych = 1
    else:
        psych = 0

    sick_ = request.form['sick']
    if (sick_ == 'true'):
        sick = 1
    else:
        sick = 0

    lithium_ = request.form["lithium"]
    if(lithium_ == "true"):
        lithium = 1
    else:
        lithium = 0

    pregnant_ = request.form['pregnant']
    if (pregnant_ == 'true'):
        pregnant = 1
    else:
        pregnant = 0

    thyroid_surgery_ = request.form['thyroid_surgery']
    if (thyroid_surgery_ == 'true'):
        thyroid_surgery = 1
    else:
```

```
            thyroid_surgery = 0

        goitre_ = request.form['goitre']
        if(goitre_ == 'true'):
            goitre = 1
        else:
            goitre = 0

        tumor_ = request.form['tumor']
        if (tumor_ == 'true'):
            tumor = 1
        else:
            tumor = 0

        cluster_output = kmeans.predict([[age,
                                    sex, on_thyroxine, query_on_thyroxine,
on_antithyroid_medication, sick, pregnant, thyroid_surgery,  I131_treatment,
                                     query_hypothyroid, query_hypothyroid,
lithium,  goitre, tumor, hypopituitary, psych, T3,
                                     TT4,
                                     T4U,
                                     FTI]])

        cluster_num = cluster_output

        model = find_model(cluster_num)

        prediction_output = model.predict([[age,
                                    sex, on_thyroxine, query_on_thyroxine,
on_antithyroid_medication, sick, pregnant, thyroid_surgery,  I131_treatment,
                                     query_hypothyroid, query_hypothyroid,
lithium,  goitre, tumor, hypopituitary, psych, T3,
                                     TT4,
                                     T4U,
                                     FTI
                                    ]])

        prediction = int(prediction_output[0])

        encoded_prediction = encoder.inverse_transform([prediction])

        return f"<div style='display: flex; justify-content: center; align-items: center;
height: 100vh;'><h1>You have {encoded_prediction[0]}</h1></div>"


if __name__ == '__main__':
    app.run(debug=True)
```

HTML & CSS For Frontend

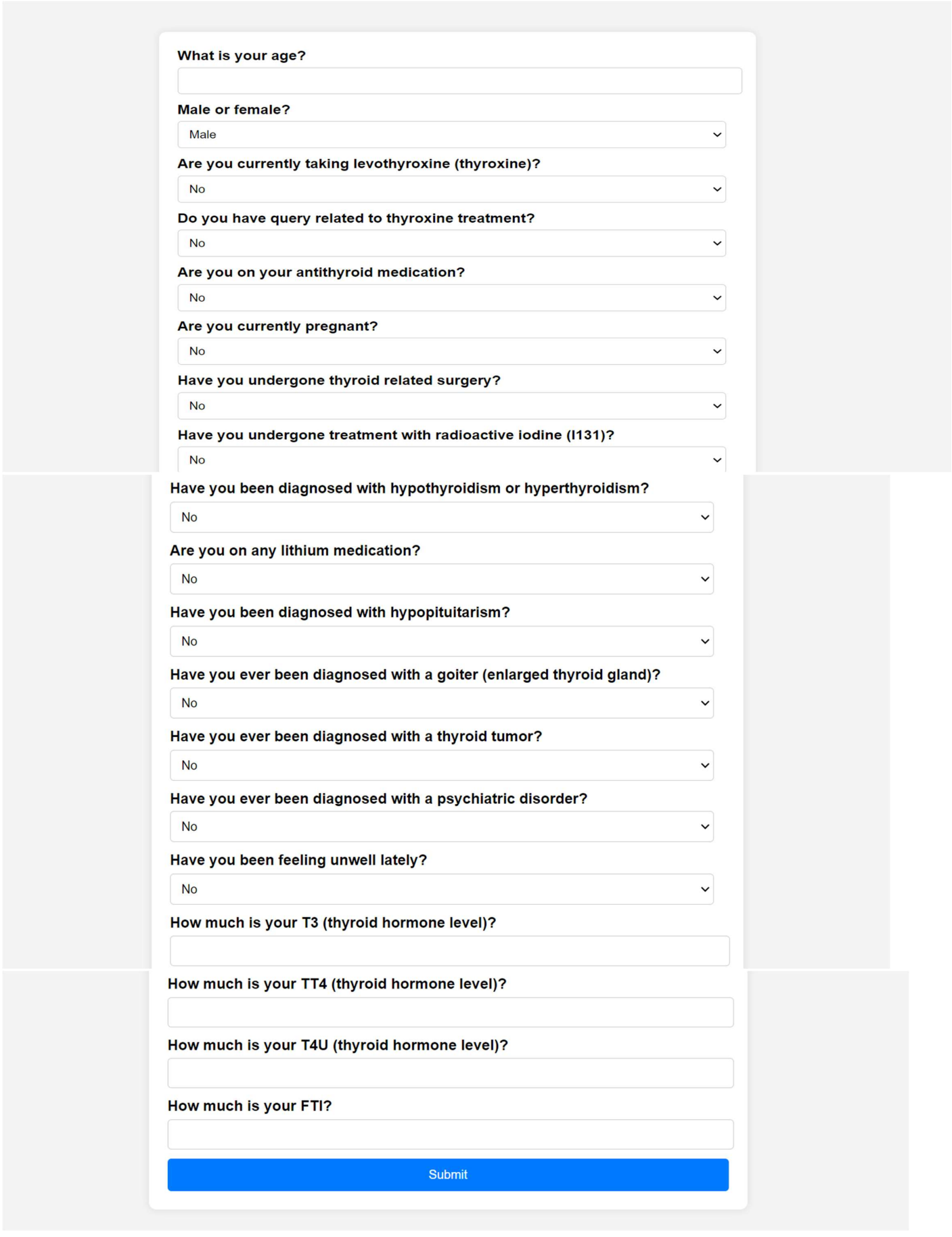
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Thyroid Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 600px;
            margin: 50px auto;
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        form label {
            display: block;
            margin-bottom: 5px;
            font-weight: bold;
        }
        form input[type="text"], form select, form input[type="checkbox"] {
            width: calc(100% - 12px);
            padding: 8px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        form input[type="submit"] {
            width: 100%;
            padding: 10px;
            background-color: #007bff;
            color: #fff;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        form input[type="submit"]:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <div class="container">
    <form action="/predict" method="post">
        <label for="age">What is your age?</label>
```

```
        <input type="text" name="age"><br>

        <label for="sex">Male or female?</label>
        <select name="sex">
            <option value="Male">Male</option>
            <option value="Female">Female</option>
        </select><br>


        <label for="on_thyroxine">Are you currently taking levothyroxine
(thyroxine)?</label>
        <select name="on_thyroxine">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>


        <label for="query_thyroxine">Do you have query related to thyroxine
treatment?</label>
        <select name="query_thyroxine">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>


        <label for="antithyroid_medication">Are you on your antithyroid
medication?</label>
        <select name="antithyroid_medication">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>







        <label for="pregnant">Are you currently pregnant?</label>
        <select name="pregnant">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>


        <label for="thyroid_surgery">Have you undergone thyroid related surgery?</label>
        <select name="thyroid_surgery">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>


        <label for="I131_treatment">Have you undergone treatment with radioactive iodine
(I131)?</label>
        <select name="I131_treatment">
            <option value="true">Yes</option>
```

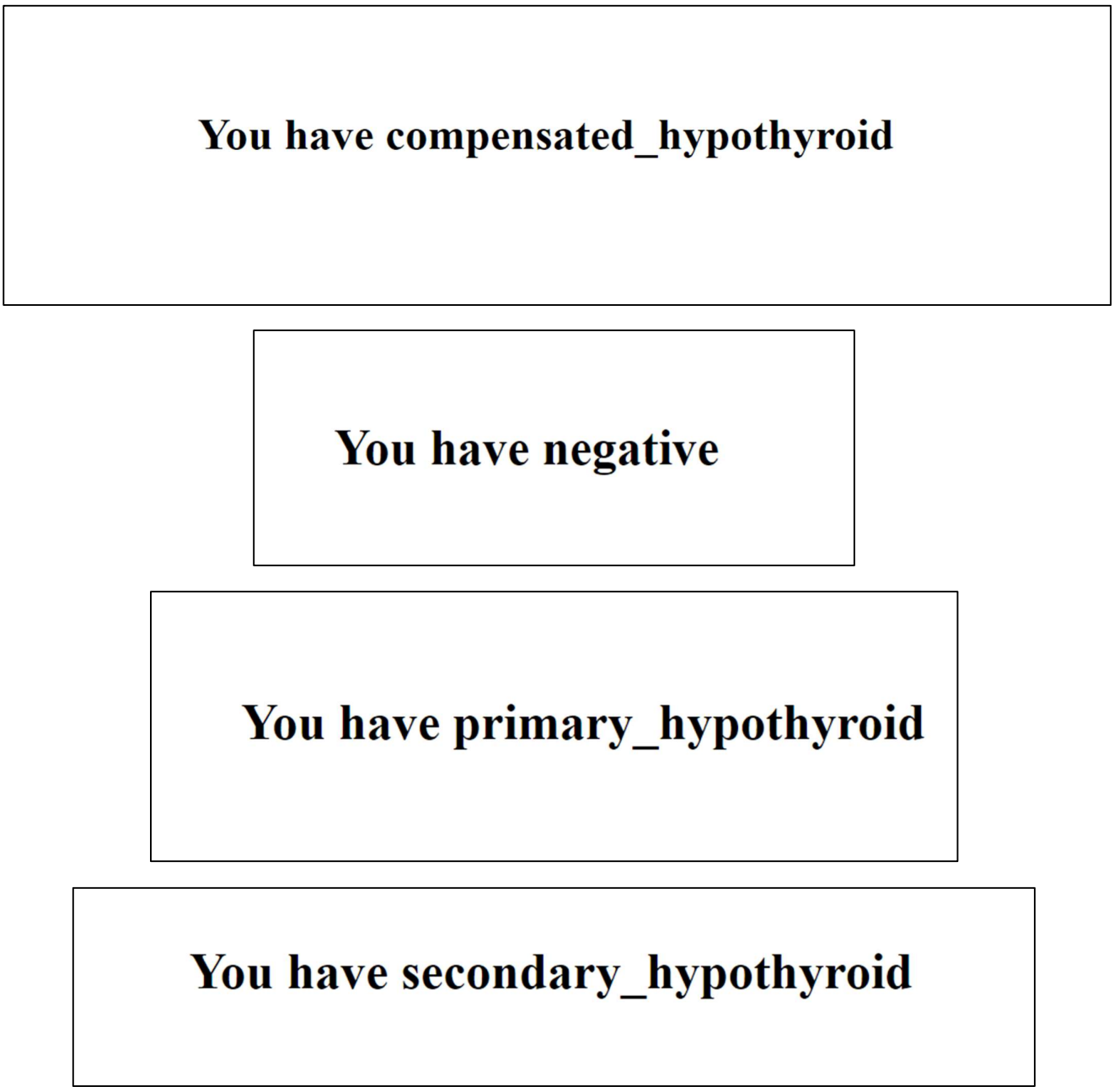
```
            <option value="false" selected>No</option>
        </select><br>


        <label for="query_hypothyroid">Have you been diagnosed with hypothyroidism or
hyperthyroidism?</label>
        <select name="query_hypothyroid">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>


        <label for="lithium">Are you on any lithium medication?</label>
        <select name="lithium">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>


        <label for="hypopituitary">Have you been diagnosed with hypopituitarism?</label>
        <select name="hypopituitary">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>

        <label for="goitre">Have you ever been diagnosed with a goiter (enlarged thyroid
gland)?</label>
        <select name="goitre">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>


        <label for="tumor">Have you ever been diagnosed with a thyroid tumor?</label>
        <select name="tumor">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>


        <label for="psych">Have you ever been diagnosed with a psychiatric
disorder?</label>
        <select name="psych">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>



        <label for="sick">Have you been feeling unwell lately?</label>
        <select name="sick">
            <option value="true">Yes</option>
            <option value="false" selected>No</option>
        </select><br>
```

```
        <label for="T3">How much is your T3 (thyroid hormone level)?</label>
        <input type="text" name="T3"><br>

        <label for="TT4">How much is your TT4 (thyroid hormone level)?</label>
        <input type="text" name="TT4"><br>

        <label for="T4U">How much is your T4U (thyroid hormone level)?</label>
        <input type="text" name="T4U"><br>

        <label for="FTI">How much is your FTI?</label>
        <input type="text" name="FTI"><br>









        <input type="submit" value="Submit">
    </form>
</div>
</body>
</html>
```

Frontend


What is your age?
Male or female?
Male
Are you currently taking levothyroxine (thyroxine)?
No
Do you have query related to thyroxine treatment?
No
Are you on your antithyroid medication?
No
Are you currently pregnant?
No
Have you undergone thyroid related surgery?
No
Have you undergone treatment with radioactive iodine (I131)?
No
Have you been diagnosed with hypothyroidism or hyperthyroidism?
No
Are you on any lithium medication?
No
Have you been diagnosed with hypopituitarism?
No
Have you ever been diagnosed with a goiter (enlarged thyroid gland)?
No
Have you ever been diagnosed with a thyroid tumor?
No
Have you ever been diagnosed with a psychiatric disorder?
No
Have you been feeling unwell lately?
No
How much is your T3 (thyroid hormone level)?
How much is your TT4 (thyroid hormone level)?
How much is your T4U (thyroid hormone level)?
How much is your FTI?
Submit

Output


You have compensated_hypothyroid
You have negative
You have primary_hypothyroid
You have secondary_hypothyroid