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\__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 [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]:

```
0.8860398860398860
```

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, err
```

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_j
```

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

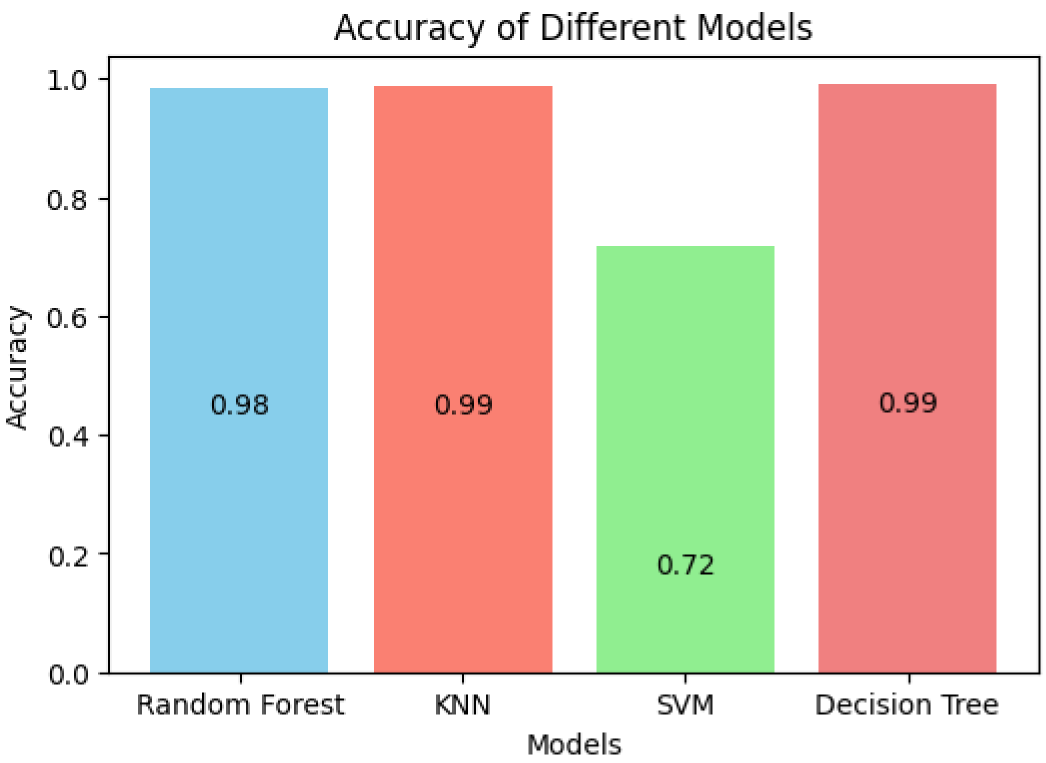
Cluster number 2

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

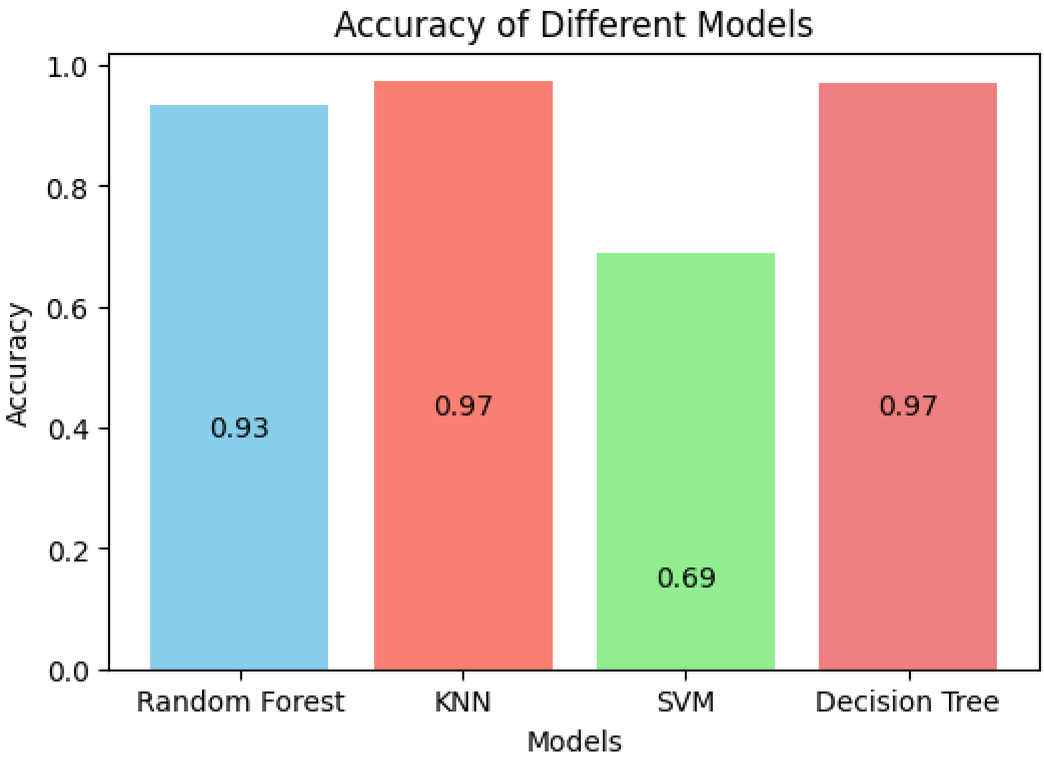
In [91]:
```
with open("dt.pkl", 'wb') as f:
    pickle.dump(dt,f)
```

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)
```