Library Importing

In [62]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
#pip install catboost
from catboost import CatBoostRegressor, Pool
from sklearn.model_selection import GridSearchCV
from sklearn.svm import NuSVC, SVR
```

In [2]:

```
def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()
```

Data Loading

In [3]:

```
cancer= pd.read_csv("data.csv")
```

In [4]:

```
cancer
```

Out[4]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	
...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	
565	926682	M	20.13	28.25	131.20	1261.0	
566	926954	M	16.60	28.08	108.30	858.1	
567	927241	M	20.60	29.33	140.10	1265.0	
568	92751	B	7.76	24.54	47.92	181.0	

569 rows × 33 columns

In [5]:

```
cancer.head()
```

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	M	17.99	10.38	122.80	1001.0	0.
1	842517	M	20.57	17.77	132.90	1326.0	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.1
4	84358402	M	20.29	14.34	135.10	1297.0	0.1

5 rows × 33 columns

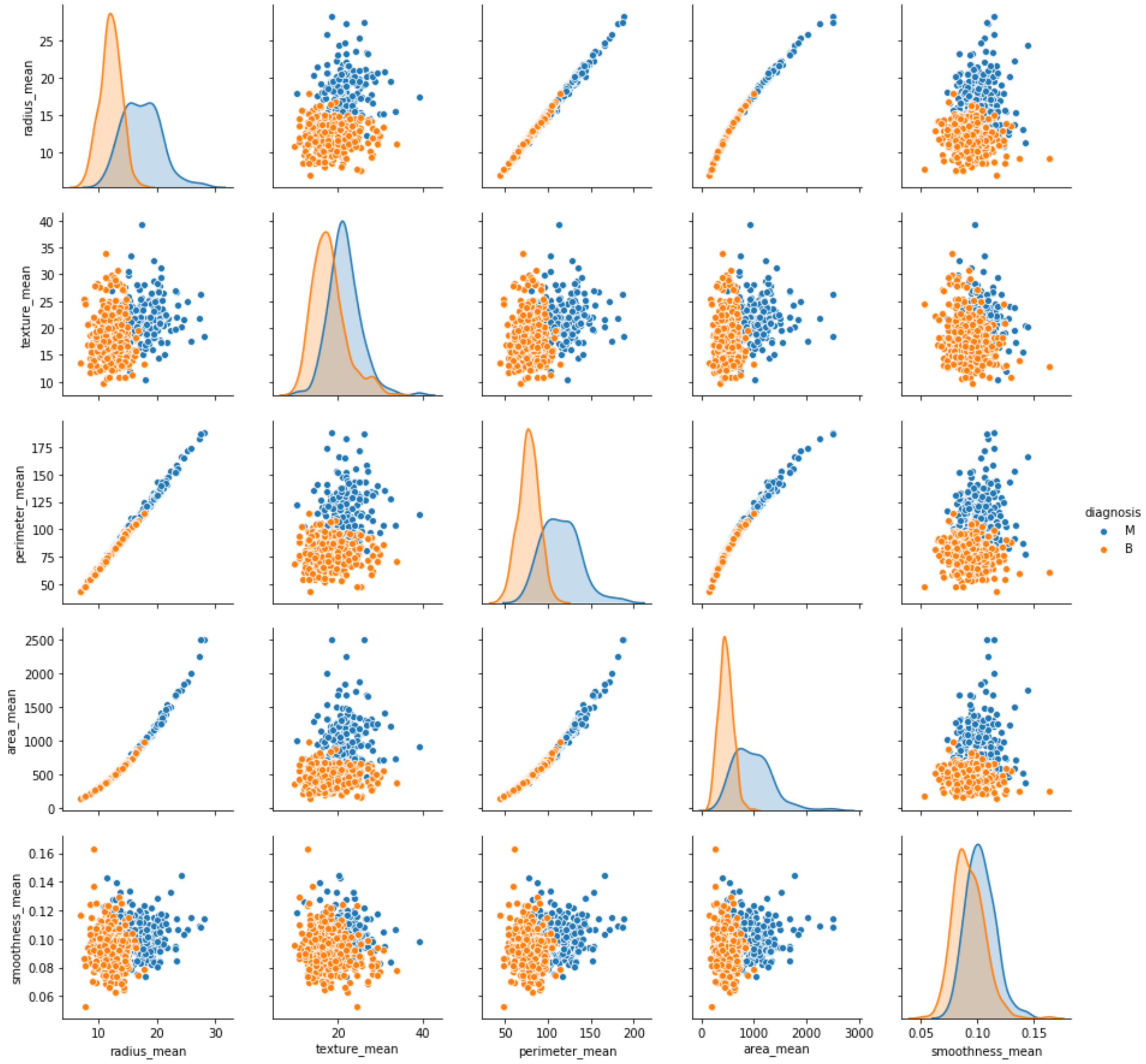
Data preprocessing and visualization

In [6]:

```
sns.pairplot(cancer,hue='diagnosis',vars=['radius_mean','texture_mean','perimeter_mean
```

Out[6]:

```
<seaborn.axisgrid.PairGrid at 0x2335e97ed88>
```

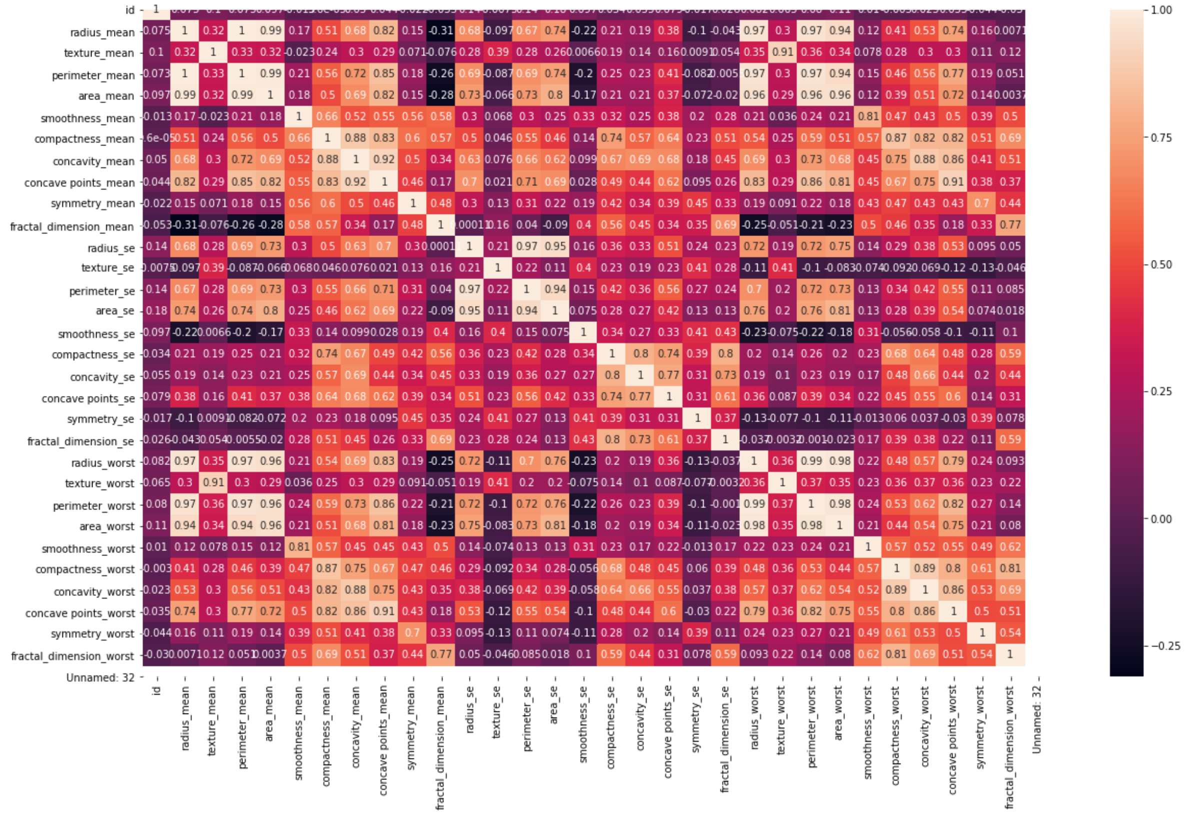


In [7]:

```
plt.figure(figsize=(20,12))
sns.heatmap(cancer.corr(),annot=True)
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2335f5e4248>
```



In [8]:

```
X=cancer.drop(['diagnosis','Unnamed: 32'],axis=1)
X.head()
```

Out[8]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
0	842302	17.99	10.38	122.80	1001.0	0.11840	
1	842517	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	11.42	20.38	77.58	386.1	0.14250	
4	84358402	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 31 columns

In [9]:

```
y= cancer['diagnosis']
y.head()
```

Out[9]:

```
0    M
1    M
2    M
3    M
4    M
Name: diagnosis, dtype: object
```

In [10]:

```

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(y)
print(integer_encoded)
y  = integer_encoded
'''
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
print(onehot_encoded)'''

```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1
 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0
 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1
 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0
 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0
 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0
 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 1
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 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 1 1 1 1 1 1 0]
```

Out[10]:

```
'\nonehot_encoder = OneHotEncoder(sparse=False)\ninteger_encoded = integer_e
ncoded.reshape(len(integer_encoded), 1)\nonehot_encoded = onehot_encoder.fit
_transform(integer_encoded)\nprint(onehot_encoded)'
```

In [11]:

```
x_train,x_test,y_train,y_test =train_test_split(X,y,test_size=0.25,random_state=20)
```

In [12]:

```
print("Size of the training set 'X' (input features) is:",x_train.shape)
print('\n')
print("Size of the testing set 'X' (input features) is:",x_test.shape)
print('\n')
print("Size of the training set 'y' (output features) is:",y_train.shape)
print('\n')
print("Size of the testing set 'y' (output features) is:",y_test.shape)
```

```
Size of the training set 'X' (input features) is: (426, 31)


Size of the testing set 'X' (input features) is: (143, 31)


Size of the training set 'y' (output features) is: (426,)


Size of the testing set 'y' (output features) is: (143,)
```

Satistical Analysis

In [13]:

```
def gen_features(X):
    s = []
    s.append(X.mean())
    s.append(X.std())
    s.append(X.min())
    s.append(X.kurtosis())
    s.append(X.skew())
    s.append(np.quantile(X,0.01))
    s.append(np.quantile(X,0.05))
    s.append(np.quantile(X,0.95))
    s.append(np.quantile(X,0.99))
    s.append(np.abs(X).std())
    s.append(np.abs(X).max())
    s.append(np.abs(X).mean())
    return pd.Series(s)
X_train_stat = pd.DataFrame()
stat = []
for df in x_train:
    #print(cancer[df].head())
    ch = gen_features(cancer[df])
    print(ch)
    #stat.append(ch)
    X_train_stat[df] = ch
    #X_train_stat.append(ch, ignore_index=True)

```

```
0     3.037183e+07
1     1.250206e+08
2     8.670000e+03
3     4.219319e+01
4     6.473752e+00
5     8.621004e+04
6     9.026700e+04
7     9.042446e+07
8     9.010343e+08
9     1.250206e+08
10    9.113205e+08
11    3.037183e+07
dtype: float64
0     14.127292
1      3.524049
2      6.981000
3      0.845522
4      0.942380
5      8.458360
```

In [14]:

```
X_train_stat.describe()
```

Out[14]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
count	1.200000e+01	12.000000	12.000000	12.000000	12.000000	12.0000
mean	1.844791e+08	11.259729	14.950045	73.554708	687.604042	0.1787
std	3.404888e+08	9.159329	12.449062	62.591301	782.626419	0.2427
min	6.473752e+00	0.845522	0.650450	0.972214	1.645732	0.0140
25%	6.682503e+04	3.524049	4.301036	24.298981	197.623000	0.0646
50%	3.037183e+07	8.993780	12.009200	57.161800	351.914129	0.0963
75%	1.250206e+08	15.739469	21.254736	102.931775	818.616828	0.1405
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.8559

8 rows × 31 columns

SVM withSVC

In [15]:

```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [16]:

```
svc_model =SVC(kernel = 'linear', random_state = 0)
```

In [17]:

```
svc_model.fit(x_train,y_train)
```

Out[17]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=0,
    shrinking=True, tol=0.001, verbose=False)
```

In [18]:

```
y_predict = svc_model.predict(x_test)
```

In [19]:

```
from sklearn.metrics import classification_report, confusion_matrix
cm = np.array(confusion_matrix(y_test, y_predict, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['is_cancer', 'is_healthy'],
                         columns=['predicted_cancer','predicted_healthy'])
confusion
```

Out[19]:

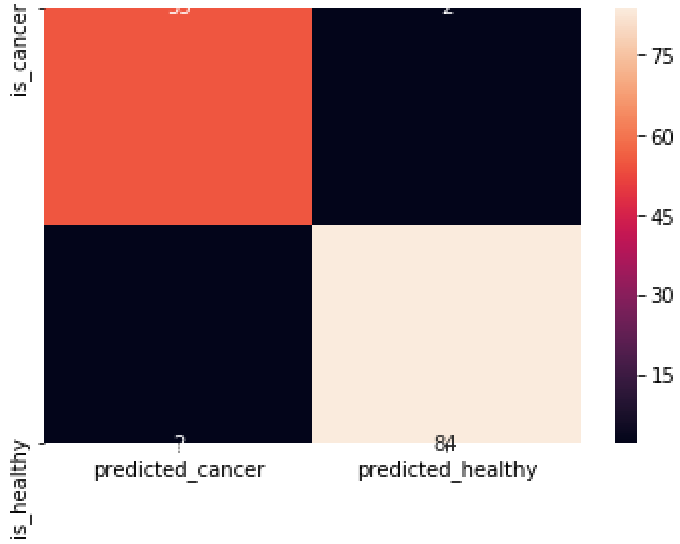
	predicted_cancer	predicted_healthy
is_cancer	55	2
is_healthy	2	84

In [20]:

```
sns.heatmap(confusion, annot=True)
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23361afb748>
```



In [21]:

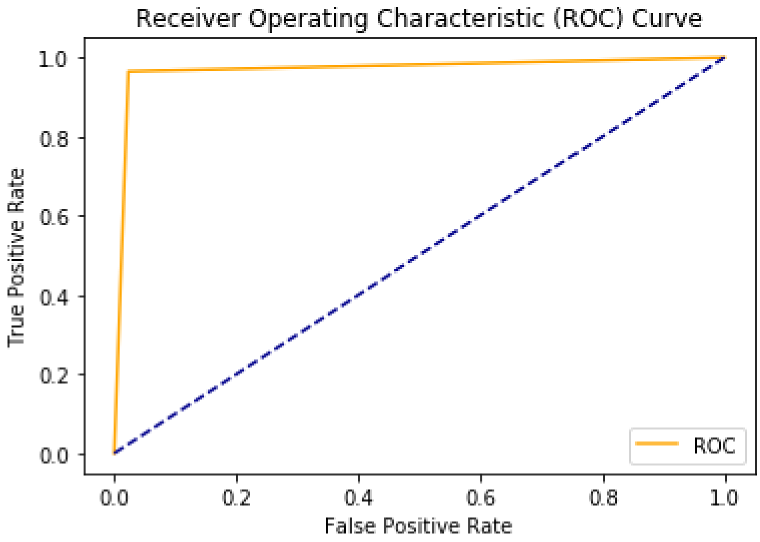
```
print("classification Repot")
all_labels = ['M','B']
print(classification_report(y_test, y_predict,target_names=all_labels))
```

```
classification Repot
              precision    recall  f1-score   support

           M       0.98      0.98      0.98        86
           B       0.96      0.96      0.96        57

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143
```

In [22]:

```
auc = roc_auc_score(y_test, y_predict)
print('AUC: %.2f' % auc)
fpr, tpr, thresholds = roc_curve(y_test, y_predict)
plot_roc_curve(fpr, tpr)
```

```
AUC: 0.97
```



SVM with RBF

In [23]:

```
svc_model = SVC(kernel = 'rbf', random_state = 0)
svc_model.fit(x_train, y_train)
y_predict = svc_model.predict(x_test)

cm = np.array(confusion_matrix(y_test, y_predict, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['is_cancer', 'is_healthy'],
                         columns=['predicted_cancer','predicted_healthy'])
confusion

```

Out[23]:

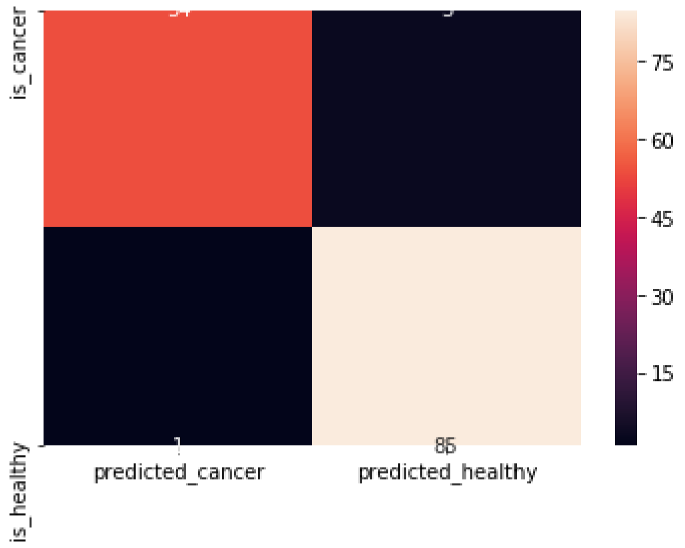
	predicted_cancer	predicted_healthy
is_cancer	54	3
is_healthy	1	85

In [24]:

```python
sns.heatmap(confusion, annot=True)
```

Out[24]:

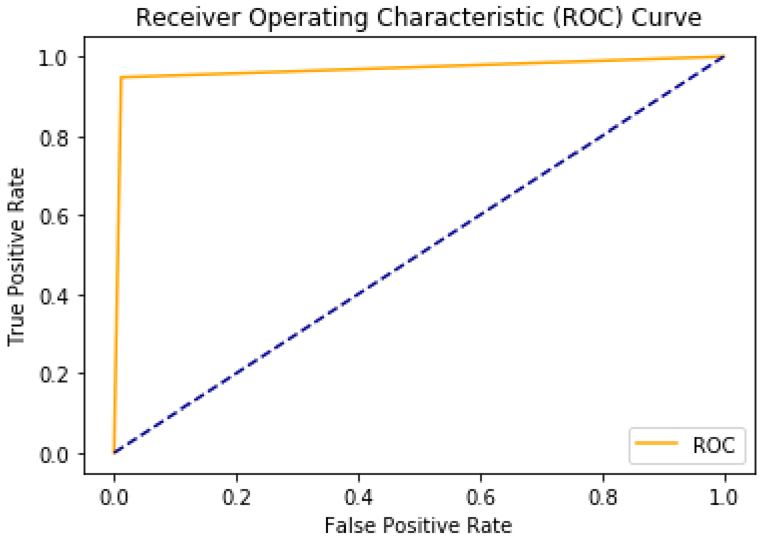
```
<matplotlib.axes._subplots.AxesSubplot at 0x23360151648>
```



In [25]:

```python
auc = roc_auc_score(y_test, y_predict)
print('AUC: %.2f' % auc)
fpr, tpr, thresholds = roc_curve(y_test, y_predict)
plot_roc_curve(fpr, tpr)
```

```
AUC: 0.97
```



Result compare with other machine learning algorithm

RandomForestClassifier

In [26]:

```
model_r = RandomForestClassifier()
model_r.fit(x_train, y_train)
y_predict_r = model_r.predict_proba(x_test)
```

```
C:\Users\Debanik Roy\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:
245: FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

In [27]:

```
cm = np.array(confusion_matrix(y_test, np.argmax(y_predict_r,axis=1), labels=[1,0]))
confusion = pd.DataFrame(cm, index=['is_cancer', 'is_healthy'],
                         columns=['predicted_cancer','predicted_healthy'])
confusion
```

Out[27]:

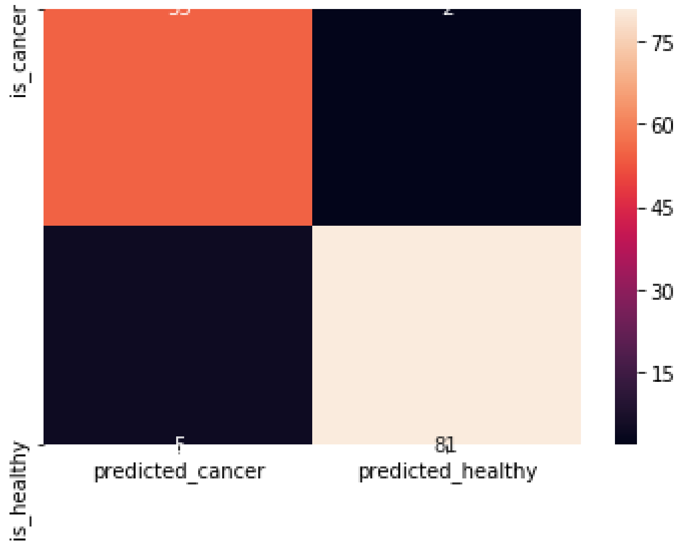
	predicted_cancer	predicted_healthy
is_cancer	55	2
is_healthy	5	81

In [28]:

```
sns.heatmap(confusion, annot=True)
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2335e585b48>
```



In [29]:

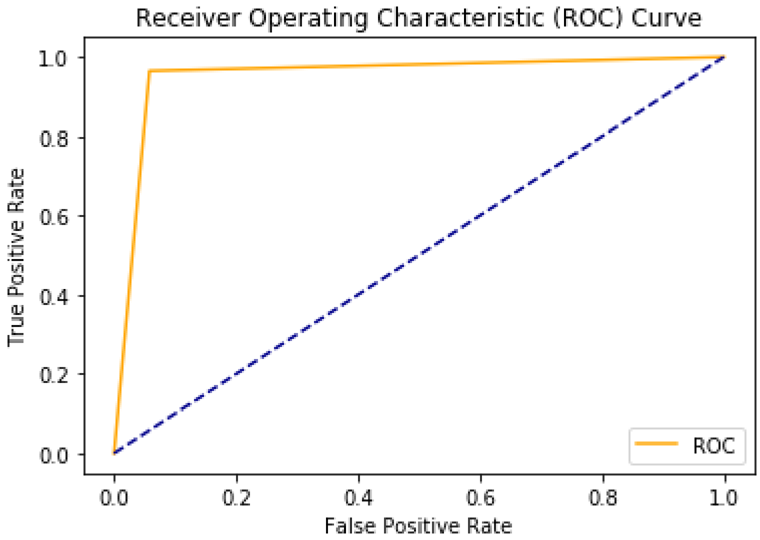
```
print("classification Repot")
all_labels = ['M','B']
print(classification_report(y_test, np.argmax(y_predict_r,axis=1),target_names=all_lab
```

```
classification Repot
              precision    recall  f1-score   support

           M       0.98      0.94      0.96        86
           B       0.92      0.96      0.94        57

    accuracy                           0.95       143
   macro avg       0.95      0.95      0.95       143
weighted avg       0.95      0.95      0.95       143
```

In [30]:

```
auc = roc_auc_score(y_test, np.argmax(y_predict_r,axis=1))
print('AUC: %.2f' % auc)
fpr, tpr, thresholds = roc_curve(y_test, np.argmax(y_predict_r,axis=1))
plot_roc_curve(fpr, tpr)
```

```
AUC: 0.95
```



KNeighborsClassifier

In [37]:

```
model_k = KNeighborsClassifier()
model_k.fit(x_train, y_train)
```

Out[37]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [38]:

```
y_predict_k = model_r.predict_proba(x_test)
```

In [39]:

```
cm = np.array(confusion_matrix(y_test, np.argmax(y_predict_k,axis=1), labels=[1,0]))
confusion = pd.DataFrame(cm, index=['is_cancer', 'is_healthy'],
                                 columns=['predicted_cancer','predicted_healthy'])
confusion

```

Out[39]:

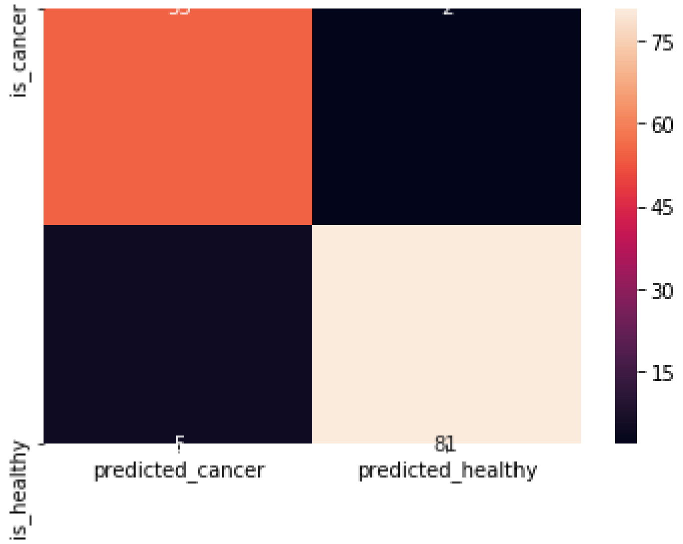
	predicted_cancer	predicted_healthy
is_cancer	55	2
is_healthy	5	81

In [40]:

```
sns.heatmap(confusion, annot=True)
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x233604aa908>
```



In [41]:

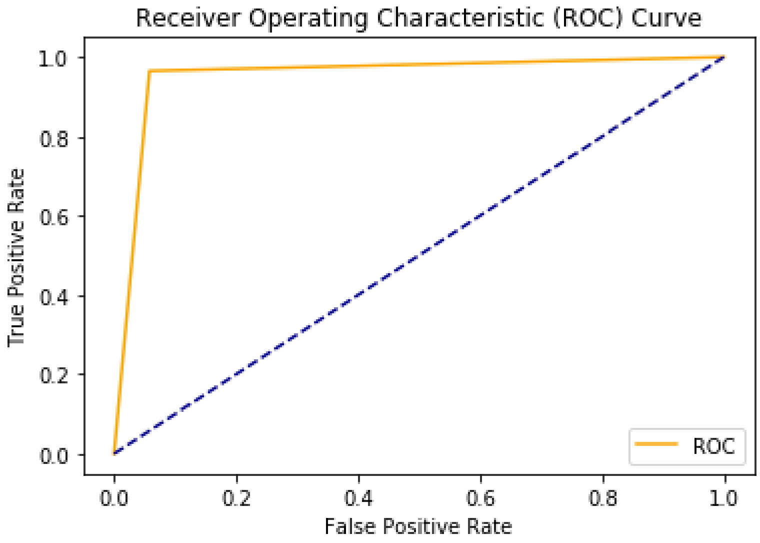
```
print("classification Repot")
all_labels = ['M','B']
print(classification_report(y_test, np.argmax(y_predict_r,axis=1),target_names=all_labe
```

```
classification Repot
              precision    recall  f1-score   support

           M       0.98      0.94      0.96        86
           B       0.92      0.96      0.94        57

    accuracy                           0.95       143
   macro avg       0.95      0.95      0.95       143
weighted avg       0.95      0.95      0.95       143
```

In [42]:

```
auc = roc_auc_score(y_test, np.argmax(y_predict_k,axis=1))
print('AUC: %.2f' % auc)
fpr, tpr, thresholds = roc_curve(y_test, np.argmax(y_predict_k,axis=1))
plot_roc_curve(fpr, tpr)
```

```
AUC: 0.95
```



LogisticRegression

In [43]:

```
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train, y_train)
```

```
C:\Users\Debanik Roy\Anaconda3\lib\site-packages\sklearn\linear_model\logist
ic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22.
Specify a solver to silence this warning.
  FutureWarning)
```

Out[43]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [44]:

```
y_predict_l = classifier.predict_proba(x_test)
cm = np.array(confusion_matrix(y_test, np.argmax(y_predict_l,axis=1), labels=[1,0]))
confusion = pd.DataFrame(cm, index=['is_cancer', 'is_healthy'],
                         columns=['predicted_cancer','predicted_healthy'])
confusion

```

Out[44]:

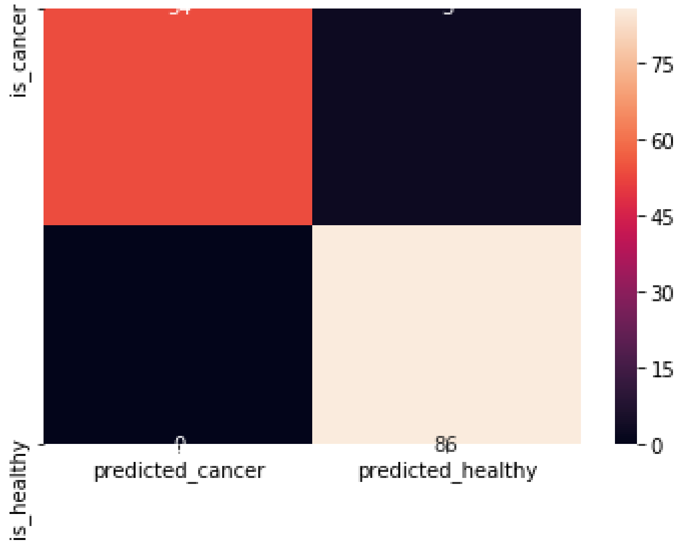
	predicted_cancer	predicted_healthy
is_cancer	54	3
is_healthy	0	86

In [45]:

```
sns.heatmap(confusion, annot=True)
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23361b915c8>
```



In [46]:

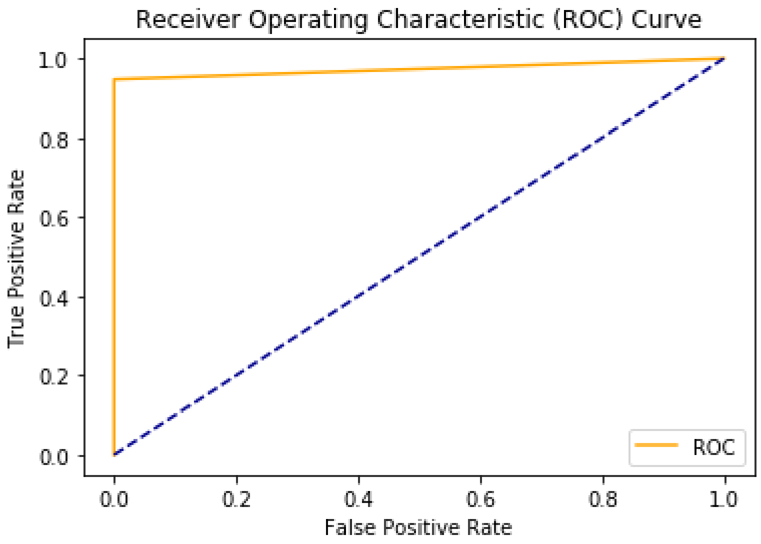
```
print("classification Repot")
all_labels = ['M','B']
print(classification_report(y_test, np.argmax(y_predict_l,axis=1),target_names=all_lab
```

```
classification Repot
              precision    recall  f1-score   support

           M       0.97      1.00      0.98        86
           B       1.00      0.95      0.97        57

    accuracy                           0.98       143
   macro avg       0.98      0.97      0.98       143
weighted avg       0.98      0.98      0.98       143
```

In [47]:

```
auc = roc_auc_score(y_test, np.argmax(y_predict_l,axis=1))
print('AUC: %.2f' % auc)
fpr, tpr, thresholds = roc_curve(y_test, np.argmax(y_predict_l,axis=1))
plot_roc_curve(fpr, tpr)
```

```
AUC: 0.97
```



GaussianNB (Naïve Bayes)

In [48]:

```
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

Out[48]:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [49]:

```
y_predict_G = classifier.predict_proba(x_test)
cm = np.array(confusion_matrix(y_test, np.argmax(y_predict_G,axis=1), labels=[1,0]))
confusion = pd.DataFrame(cm, index=['is_cancer', 'is_healthy'],
                              columns=['predicted_cancer','predicted_healthy'])
confusion

```

Out[49]:

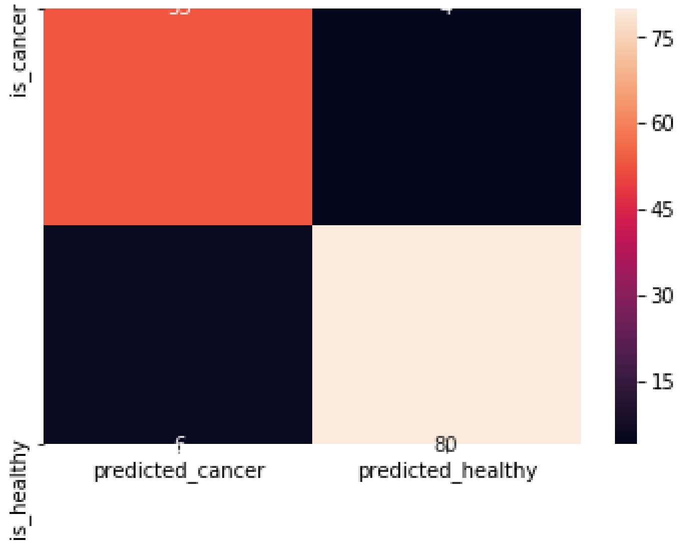
	predicted_cancer	predicted_healthy
is_cancer	53	4
is_healthy	6	80

In [50]:

```
sns.heatmap(confusion, annot=True)
```

Out[50]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23362071c88>
```



In [51]:

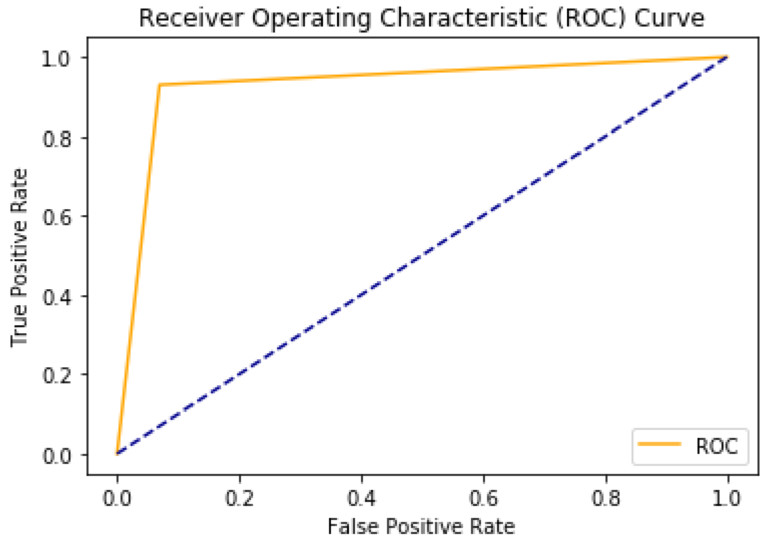
```
print("classification Repot")
all_labels = ['M','B']
print(classification_report(y_test, np.argmax(y_predict_G,axis=1),target_names=all_lab
```

```
classification Repot
              precision    recall  f1-score   support

           M       0.95      0.93      0.94        86
           B       0.90      0.93      0.91        57

    accuracy                           0.93       143
   macro avg       0.93      0.93      0.93       143
weighted avg       0.93      0.93      0.93       143
```

In [52]:

```
auc = roc_auc_score(y_test, np.argmax(y_predict_G,axis=1))
print('AUC: %.2f' % auc)
fpr, tpr, thresholds = roc_curve(y_test, np.argmax(y_predict_G,axis=1))
plot_roc_curve(fpr, tpr)
```

```
AUC: 0.93
```



Decision Tree Algorithm

In [53]:

```
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(x_train, y_train)
```

Out[53]:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=Non
e,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

In [54]:

```
y_predict_D = classifier.predict_proba(x_test)
cm = np.array(confusion_matrix(y_test, np.argmax(y_predict_D,axis=1), labels=[1,0]))
confusion = pd.DataFrame(cm, index=['is_cancer', 'is_healthy'],
                         columns=['predicted_cancer','predicted_healthy'])
confusion

```

Out[54]:

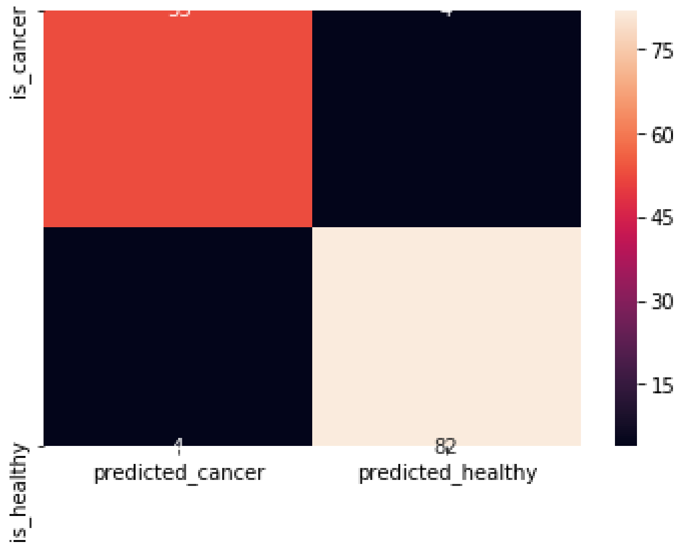
	predicted_cancer	predicted_healthy
is_cancer	53	4
is_healthy	4	82

In [55]:

```
sns.heatmap(confusion, annot=True)
```

Out[55]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23362183308>
```



In [56]:

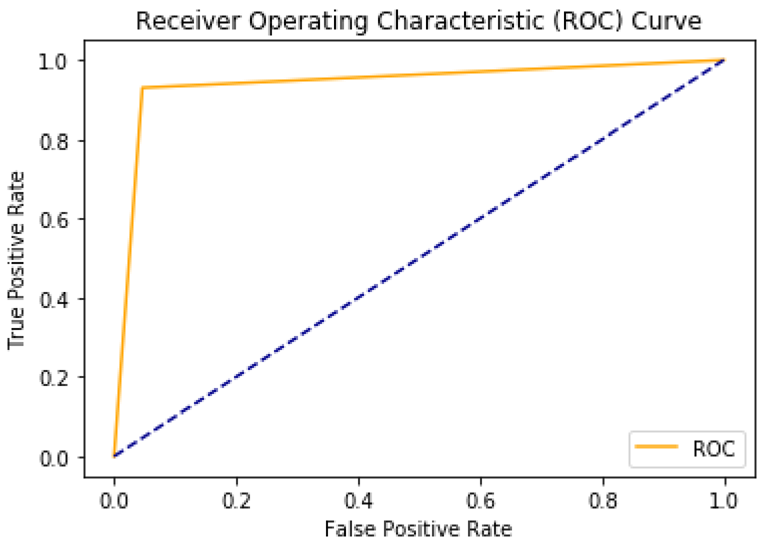
```
print("classification Repot")
all_labels = ['M','B']
print(classification_report(y_test, np.argmax(y_predict_D,axis=1),target_names=all_lab
```

```
classification Repot
              precision    recall  f1-score   support

           M       0.95      0.95      0.95        86
           B       0.93      0.93      0.93        57

    accuracy                           0.94       143
   macro avg       0.94      0.94      0.94       143
weighted avg       0.94      0.94      0.94       143
```

In [57]:

```
auc = roc_auc_score(y_test, np.argmax(y_predict_D,axis=1))
print('AUC: %.2f' % auc)
fpr, tpr, thresholds = roc_curve(y_test, np.argmax(y_predict_D,axis=1))
plot_roc_curve(fpr, tpr)
```

```
AUC: 0.94
```



catboost

In [58]:

```
train_pool = Pool(x_train,y_train)
m = CatBoostRegressor(iterations=1000, loss_function="MAE", boosting_type="Ordered")
m.fit(x_train,y_train, silent=True)
m.best_score_
```

Out[58]:

```
{'learn': {'MAE': 0.02804645084439716}}
```

In [59]:

```
y_pred_c = m.predict(np.argmax(x_test,axis=1))
```

In [60]:

```
y_pred_c
```

Out[60]:

```
0.9406923110069474
```

GridSearchCV

In [63]:

```
parameters = [{'gamma': [0.001, 0.005, 0.01, 0.02, 0.05, 0.1],
               'C': [0.1, 0.2, 0.25, 0.5, 1, 1.5, 2]}]
               #'nu': [0.75, 0.8, 0.85, 0.9, 0.95, 0.97]}]
reg1 = GridSearchCV(SVR(kernel='rbf', tol=0.01), parameters, cv=5, scoring='neg_mean_al
reg1.fit(x_train, y_train.flatten())
y_pred1 = reg1.predict(x_train)

print("Best CV score: {:.4f}".format(reg1.best_score_))
print(reg1.best_params_)
#print(y_pred1)
```

```
Best CV score: -0.1374
{'C': 2, 'gamma': 0.02}
```

In [ ]:

```

```

In [ ]:

```

```