Importing Libraries

In [1]:

```
# !pip install xgboost
```

In [2]:

```
# !pip install graphviz
```

In [3]:

```
# !pip install lightgbm
```

In [1]:

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

from sklearn.ensemble import GradientBoostingClassifier
import xgboost as xgb
import lightgbm as lgb

import graphviz

from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, roc_curve, roc_auc_score
from sklearn.feature_selection import mutual_info_classif
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
```

Importing The Dataset

In [2]:

```
lung_cancer = pd.read_csv('lung_cancer.csv')
lung_cancer
```

Out[2]:

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING
0	M	69	1	2	2	1	1	2	1	2	2
1	M	74	2	1	1	1	2	2	2	1	1
2	F	59	1	1	1	2	1	2	1	2	1
3	M	63	2	2	2	1	1	1	1	1	2
4	F	63	1	2	1	1	1	1	1	2	1
...	...	...	...	...	...	...	...	...	...	...	...
304	F	56	1	1	1	2	2	2	1	1	2
305	M	70	2	1	1	1	1	2	2	2	2
306	M	58	2	1	1	1	1	1	2	2	2
307	M	67	2	1	2	1	1	2	2	1	2
308	M	62	1	1	1	2	1	2	2	2	2

309 rows × 16 columns

In [3]:

```
lung_cancer.isnull().sum()
```

Out[3]:

```
GENDER                   0
AGE                      0
SMOKING                  0
YELLOW_FINGERS           0
ANXIETY                  0
PEER_PRESSURE            0
CHRONIC DISEASE          0
FATIGUE                  0
ALLERGY                  0
WHEEZING                 0
ALCOHOL CONSUMING        0
COUGHING                 0
SHORTNESS OF BREATH      0
SWALLOWING DIFFICULTY    0
CHEST PAIN               0
LUNG_CANCER              0
dtype: int64
```

In [4]:

```
le = LabelEncoder()
```

In [5]:

```
lung_cancer.GENDER = le.fit_transform(lung_cancer['GENDER'])
lung_cancer.LUNG_CANCER = le.fit_transform(lung_cancer['LUNG_CANCER'])
```

In [6]:

```
x = lung_cancer.drop(['LUNG_CANCER'], axis = 1)
y = lung_cancer['LUNG_CANCER']
mutual_info = mutual_info_classif(x,y)
mutual_info = pd.Series(mutual_info)
mutual_info.index = x.columns
mutual_info.sort_values(ascending=False)
```

Out[6]:

```
ALLERGY                  0.071304
WHEEZING                 0.051149
SWALLOWING DIFFICULTY    0.045092
AGE                      0.034302
COUGHING                 0.033011
ALCOHOL CONSUMING        0.027062
YELLOW_FINGERS           0.016004
SMOKING                  0.012691
CHEST PAIN               0.009974
PEER_PRESSURE            0.006295
GENDER                   0.000000
ANXIETY                  0.000000
CHRONIC DISEASE          0.000000
FATIGUE                  0.000000
SHORTNESS OF BREATH      0.000000
dtype: float64
```

In [7]:

```
lung_cancer.mean()
```

Out[7]:

```
GENDER                    0.524272
AGE                      62.673139
SMOKING                   1.563107
YELLOW_FINGERS            1.569579
ANXIETY                   1.498382
PEER_PRESSURE             1.501618
CHRONIC DISEASE           1.504854
FATIGUE                   1.673139
ALLERGY                   1.556634
WHEEZING                  1.556634
ALCOHOL CONSUMING         1.556634
COUGHING                  1.579288
SHORTNESS OF BREATH       1.640777
SWALLOWING DIFFICULTY     1.469256
CHEST PAIN                1.556634
LUNG_CANCER               0.873786
dtype: float64
```

In [8]:

```
lung_cancer.min()
```

Out[8]:

```
GENDER                    0
AGE                      21
SMOKING                   1
YELLOW_FINGERS            1
ANXIETY                   1
PEER_PRESSURE             1
CHRONIC DISEASE           1
FATIGUE                   1
ALLERGY                   1
WHEEZING                  1
ALCOHOL CONSUMING         1
COUGHING                  1
SHORTNESS OF BREATH       1
SWALLOWING DIFFICULTY     1
CHEST PAIN                1
LUNG_CANCER               0
dtype: int64
```

In [9]:

```
lung_cancer.max()
```

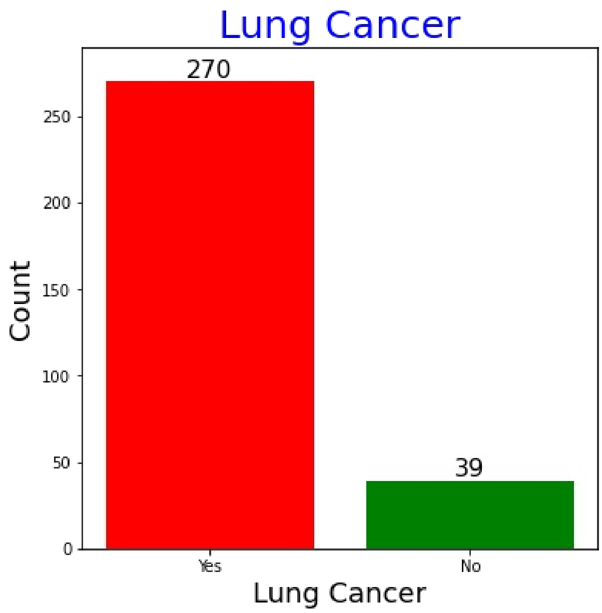
```
Out[9]: GENDER                      1
        AGE                        87
        SMOKING                     2
        YELLOW_FINGERS              2
        ANXIETY                     2
        PEER_PRESSURE               2
        CHRONIC DISEASE             2
        FATIGUE                     2
        ALLERGY                     2
        WHEEZING                    2
        ALCOHOL CONSUMING           2
        COUGHING                    2
        SHORTNESS OF BREATH         2
        SWALLOWING DIFFICULTY       2
        CHEST PAIN                  2
        LUNG_CANCER                 1
        dtype: int64
```

```
In [10]: lung_cancer.std()
```

```
Out[10]: GENDER                   0.500221
         AGE                      8.210301
         SMOKING                  0.496806
         YELLOW_FINGERS           0.495938
         ANXIETY                  0.500808
         PEER_PRESSURE            0.500808
         CHRONIC DISEASE          0.500787
         FATIGUE                  0.469827
         ALLERGY                  0.497588
         WHEEZING                 0.497588
         ALCOHOL CONSUMING        0.497588
         COUGHING                 0.494474
         SHORTNESS OF BREATH      0.480551
         SWALLOWING DIFFICULTY    0.499863
         CHEST PAIN               0.497588
         LUNG_CANCER              0.332629
         dtype: float64
```

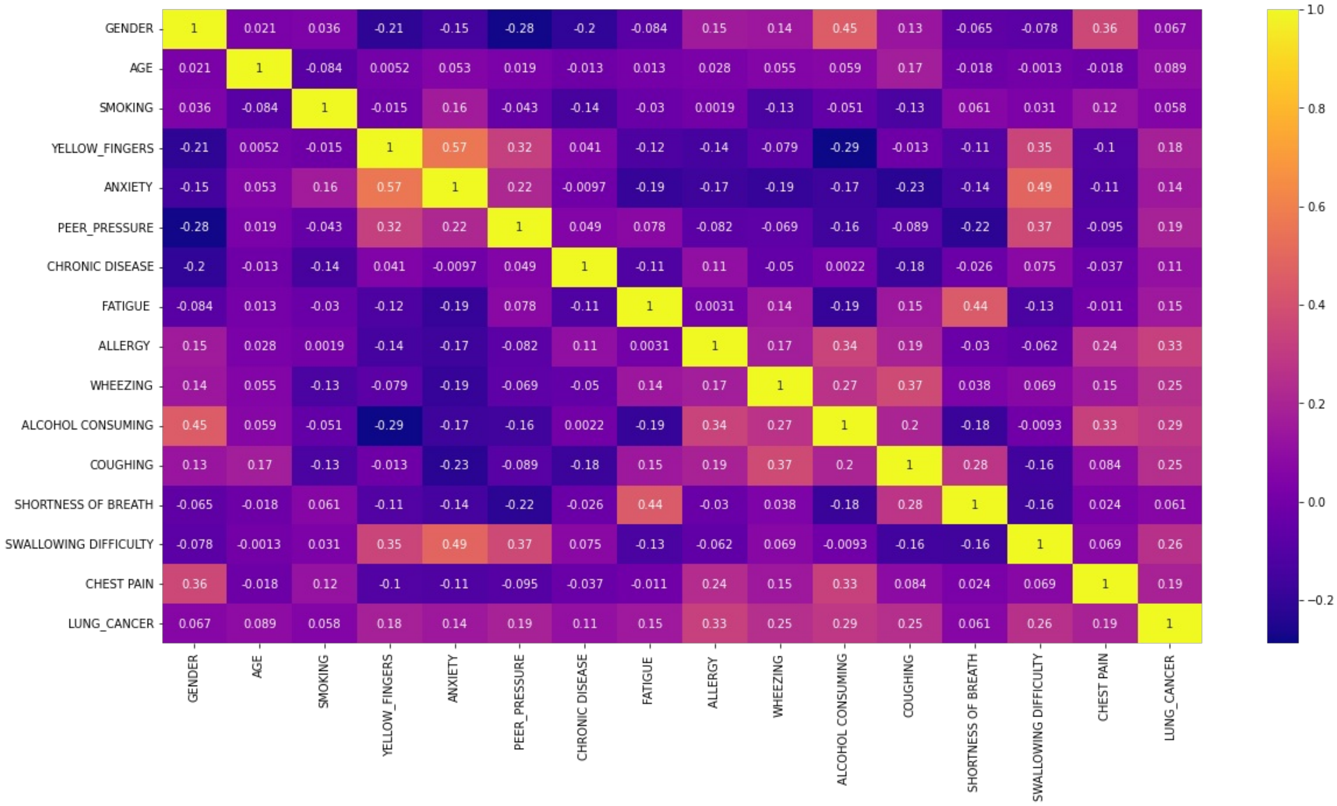
Visualizing The Data

```
In [11]: plt.figure(figsize=(6,6))
         names = ["Yes", "No"]
         count = [(lung_cancer.LUNG_CANCER.values == 1).sum(), (lung_cancer.LUNG_CANCER.values == 0).sum()]
         plt.bar(names, count, color = ["Red", "Green"])
         plt.title('Lung Cancer', color = 'blue', fontsize= 25)
         plt.xlabel('Lung Cancer', fontsize= 18)
         plt.ylabel('Count', fontsize= 18)
         plt.ylim(0,290)
         for i in range(len(names)):
             plt.text(i, count[i], ha='center', va='bottom', fontsize=16)
         plt.show()
```



```
In [12]: plt.figure(figsize = (20, 10))
         sns.heatmap(lung_cancer.corr(), annot = True, cmap="plasma")
```

```
Out[12]: <AxesSubplot:>
```



	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTNESS OF BREATH	SWALLOWING DIFFICULTY	CHEST PAIN	LUNG_CANCER
GENDER	1	0.021	0.036	-0.21	-0.15	-0.28	-0.2	-0.084	0.15	0.14	0.45	0.13	-0.065	-0.078	0.36	0.067
AGE	0.021	1	-0.084	0.0052	0.053	0.019	-0.013	0.013	0.028	0.055	0.059	0.17	-0.018	-0.0013	-0.018	0.089
SMOKING	0.036	-0.084	1	-0.015	0.16	-0.043	-0.14	-0.03	0.0019	-0.13	-0.051	-0.13	0.061	0.031	0.12	0.058
YELLOW_FINGERS	-0.21	0.0052	-0.015	1	0.57	0.32	0.041	-0.12	-0.14	-0.079	-0.29	-0.013	-0.11	0.35	-0.1	0.18
ANXIETY	-0.15	0.053	0.16	0.57	1	0.22	-0.0097	-0.19	-0.17	-0.19	-0.17	-0.23	-0.14	0.49	-0.11	0.14
PEER_PRESSURE	-0.28	0.019	-0.043	0.32	0.22	1	0.049	0.078	-0.082	-0.069	-0.16	-0.089	-0.22	0.37	-0.095	0.19
CHRONIC DISEASE	-0.2	-0.013	-0.14	0.041	-0.0097	0.049	1	-0.11	0.11	-0.05	0.0022	-0.18	-0.026	0.075	-0.037	0.11
FATIGUE	-0.084	0.013	-0.03	-0.12	-0.19	0.078	-0.11	1	0.0031	0.14	-0.19	0.15	0.44	-0.13	-0.011	0.15
ALLERGY	0.15	0.028	0.0019	-0.14	-0.17	-0.082	0.11	0.0031	1	0.17	0.34	0.19	-0.03	-0.062	0.24	0.33
WHEEZING	0.14	0.055	-0.13	-0.079	-0.19	-0.069	-0.05	0.14	0.17	1	0.27	0.37	0.038	0.069	0.15	0.25
ALCOHOL CONSUMING	0.45	0.059	-0.051	-0.29	-0.17	-0.16	0.0022	-0.19	0.34	0.27	1	0.2	-0.18	-0.0093	0.33	0.29
COUGHING	0.13	0.17	-0.13	-0.013	-0.23	-0.089	-0.18	0.15	0.19	0.37	0.2	1	0.28	-0.16	0.084	0.25
SHORTNESS OF BREATH	-0.065	-0.018	0.061	-0.11	-0.14	-0.22	-0.026	0.44	-0.03	0.038	-0.18	0.28	1	-0.16	0.024	0.061
SWALLOWING DIFFICULTY	-0.078	-0.0013	0.031	0.35	0.49	0.37	0.075	-0.13	-0.062	0.069	-0.0093	-0.16	-0.16	1	0.069	0.26
CHEST PAIN	0.36	-0.018	0.12	-0.1	-0.11	-0.095	-0.037	-0.011	0.24	0.15	0.33	0.084	0.024	0.069	1	0.19
LUNG_CANCER	0.067	0.089	0.058	0.18	0.14	0.19	0.11	0.15	0.33	0.25	0.29	0.25	0.061	0.26	0.19	1

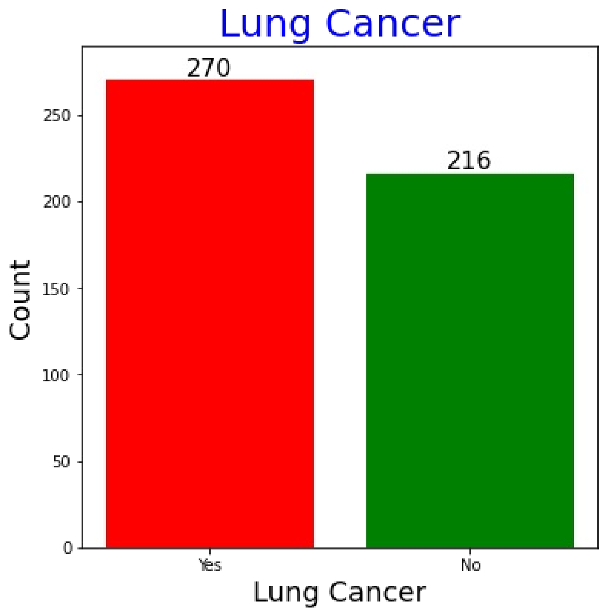
In [13]:

```
os = RandomOverSampler(0.7)
x_os, y_os = os.fit_resample(x,y)
print("Before fit {}".format(Counter(y)))
print("After fit {}".format(Counter(y_os)))
```

```
Before fit Counter({1: 270, 0: 39})
After fit Counter({1: 270, 0: 189})
C:\Users\nitis\anaconda3\lib\site-packages\imblearn\utils\_validation.py:586: FutureWarning: Pass sampling_stra
tegy=0.7 as keyword args. From version 0.9 passing these as positional arguments will result in an error
  warnings.warn(
```

In [14]:

```
plt.figure(figsize=(6,6))
names = ["Yes", "No"]
count = [270, 216]
plt.bar(names, count, color = ["Red", "Green"])
plt.title('Lung Cancer', color = 'blue', fontsize= 25)
plt.xlabel('Lung Cancer', fontsize= 18)
plt.ylabel('Count', fontsize= 18)
plt.ylim(0,290)
for i in range(len(names)):
    plt.text(i, count[i], count[i], ha='center', va='bottom', fontsize=16)
plt.show()
```



In [15]:

```
scaler = StandardScaler()
```

```
scaler.fit(x_os)
scaled_x = scaler.transform(x_os)
```

In [16]:

```
pca = PCA(n_components=7)
pca.fit(scaled_x)
pca_x = pca.transform(scaled_x)
pca_x.shape
```

Out[16]:

```
(459, 7)
```

Models Training

In [17]:

```
x_train, x_test, y_train, y_test = train_test_split(pca_x,y_os,test_size = 0.35)

GBM = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1, random_state=45).fit(x_train

XGB = xgb.XGBClassifier(objective="binary:logistic", random_state=45, eval_metric="auc", n_estimators=50).fit(x

LGBM = lgb.LGBMClassifier(num_leaves=31, learning_rate=1, n_estimators=100, random_state=45).fit(x_train, y_tra
```

Evaluating Models

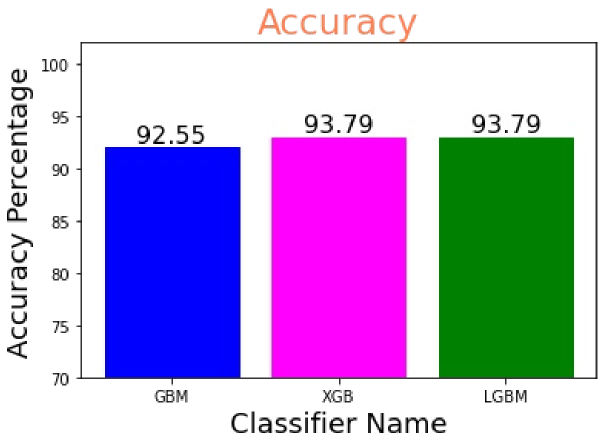
In [18]:

```
GBM_pre = GBM.predict(x_test)
GBM_sc = GBM.score(x_test, y_test) * 100
GBM_sc = "{:.2f}".format(GBM_sc)

XGB_pre = XGB.predict(x_test)
XGB_sc = XGB.score(x_test, y_test) * 100
XGB_sc = "{:.2f}".format(XGB_sc)

LGBM_pre = LGBM.predict(x_test)
LGBM_sc = LGBM.score(x_test, y_test) * 100
LGBM_sc = "{:.2f}".format(LGBM_sc)
```

Accuracy

In [19]:

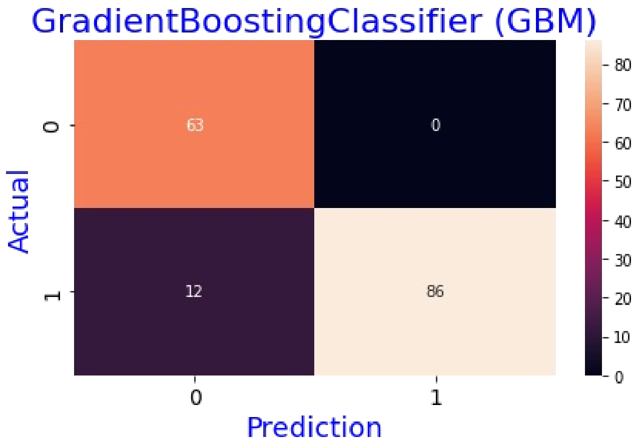
```
algo = ['GBM', 'XGB', 'LGBM']
acc = [math.floor(int(float(GBM_sc))),math.floor(int(float(XGB_sc))), math.floor(int(float(LGBM_sc)))]
pert = [GBM_sc, XGB_sc, LGBM_sc]
colors = ['blue','magenta', 'green']
plt.bar(algo, acc,color=colors, edgecolor=colors)
plt.title('Accuracy', color = 'coral', fontsize= 23)
plt.xlabel('Classifier Name', fontsize= 18)
plt.ylabel('Accuracy Percentage', fontsize= 18)
plt.ylim(70,102)
for i in range(len(algo)):
    plt.text(i, acc[i], pert[i], ha='center', va='bottom', fontsize=16)
plt.show()
```



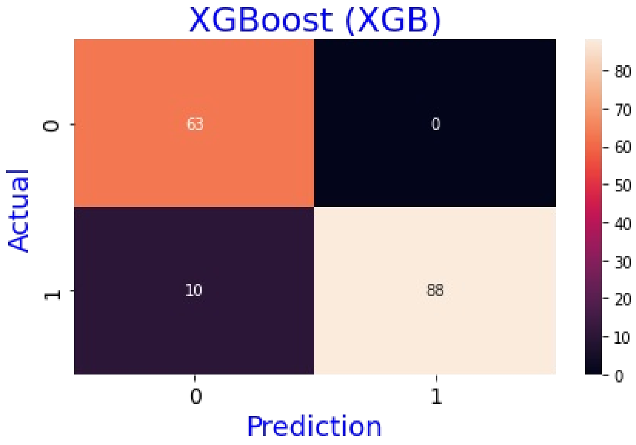
In [20]:

```
def print_confusion_matrix(confusion_matrix, class_names, figsize = (7,4), fontsize=14, title = "Confusion Matr
    df_cm = pd.DataFrame(
        confusion_matrix, index=class_names, columns=class_names,
    )
    fig = plt.figure(figsize=figsize)
    try:
        heatmap = sns.heatmap(df_cm, annot=True, fmt="d")
    except ValueError:
        raise ValueError("Confusion matrix values must be integers.")
    heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=90, ha='right', fontsize=fontsize)
    heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=0, ha='center', fontsize=fontsize)
    plt.ylabel('Actual', fontsize=18, color='blue')
    plt.xlabel('Prediction',  fontsize=18, color='blue')
    plt.title(title, fontsize=22, color='blue')
```

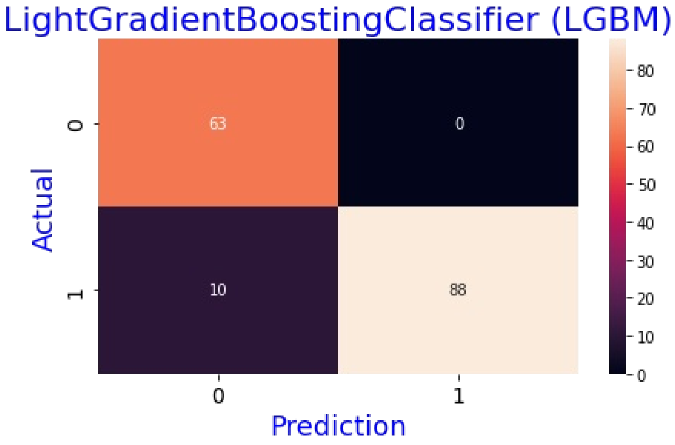
```
In [21]: cm = confusion_matrix(y_test, GBM_pre)
         print_confusion_matrix(cm,["0", "1"], title="GradientBoostingClassifier (GBM)")
```



```
In [22]: cm = confusion_matrix(y_test, XGB_pre)
         print_confusion_matrix(cm,["0", "1"], title="XGBoost (XGB)")
```



```
In [23]: cm = confusion_matrix(y_test, LGBM_pre)
         print_confusion_matrix(cm,["0", "1"], title="LightGradientBoostingClassifier (LGBM)")
```



```
In [24]: pd.DataFrame(classification_report(y_test, GBM_pre, output_dict=True))
```

Out[24]:

	0	1	accuracy	macro avg	weighted avg
precision	0.840000	1.000000	0.925466	0.920000	0.937391
recall	1.000000	0.877551	0.925466	0.938776	0.925466
f1-score	0.913043	0.934783	0.925466	0.923913	0.926276
support	63.000000	98.000000	0.925466	161.000000	161.000000

```
In [25]: pd.DataFrame(classification_report(y_test, XGB_pre, output_dict=True))
```

Out[25]:

	0	1	accuracy	macro avg	weighted avg
precision	0.863014	1.000000	0.937888	0.931507	0.946397
recall	1.000000	0.897959	0.937888	0.948980	0.937888
f1-score	0.926471	0.946237	0.937888	0.936354	0.938502
support	63.000000	98.000000	0.937888	161.000000	161.000000

In [26]:

```
pd.DataFrame(classification_report(y_test, LGBM_pre, output_dict=True))
```

Out[26]:

	0	1	accuracy	macro avg	weighted avg
precision	0.863014	1.000000	0.937888	0.931507	0.946397
recall	1.000000	0.897959	0.937888	0.948980	0.937888
f1-score	0.926471	0.946237	0.937888	0.936354	0.938502
support	63.000000	98.000000	0.937888	161.000000	161.000000

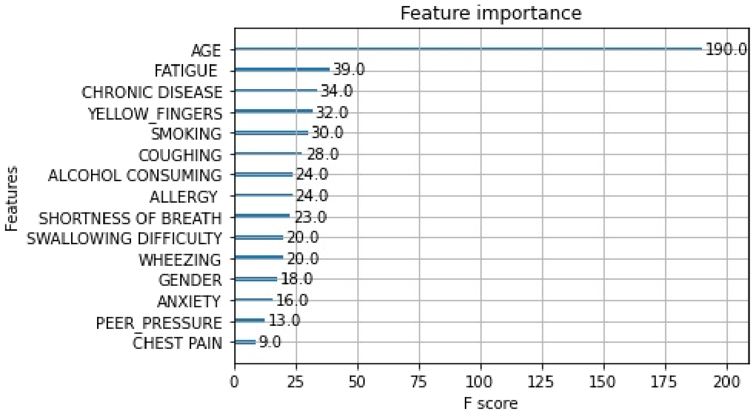
In [46]:

```
x_train, x_test, y_train, y_test = train_test_split(x_os,y_os,test_size = 0.35)

GBM = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1, random_state=45).fit(x_train

XGB = xgb.XGBClassifier(objective="binary:logistic", random_state=45, eval_metric="auc", n_estimators=50).fit(x_

LGBM = lgb.LGBMClassifier(num_leaves=31, learning_rate=1, n_estimators=100, random_state=45).fit(x_train, y_tra
```

In [47]:

```
plt.figure(figsize=(10,9))
xgb.plot_importance(XGB)
plt.show()
```

```
<Figure size 720x648 with 0 Axes>
```



In [48]:

```
# !pip install shap
```

In [49]:

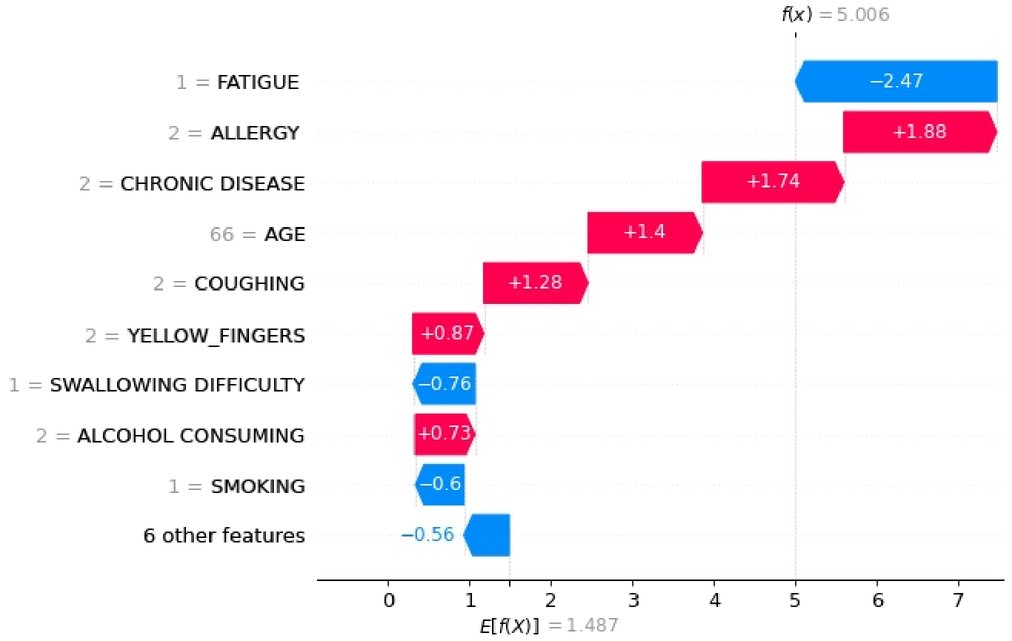
```
import shap
```

In [50]:

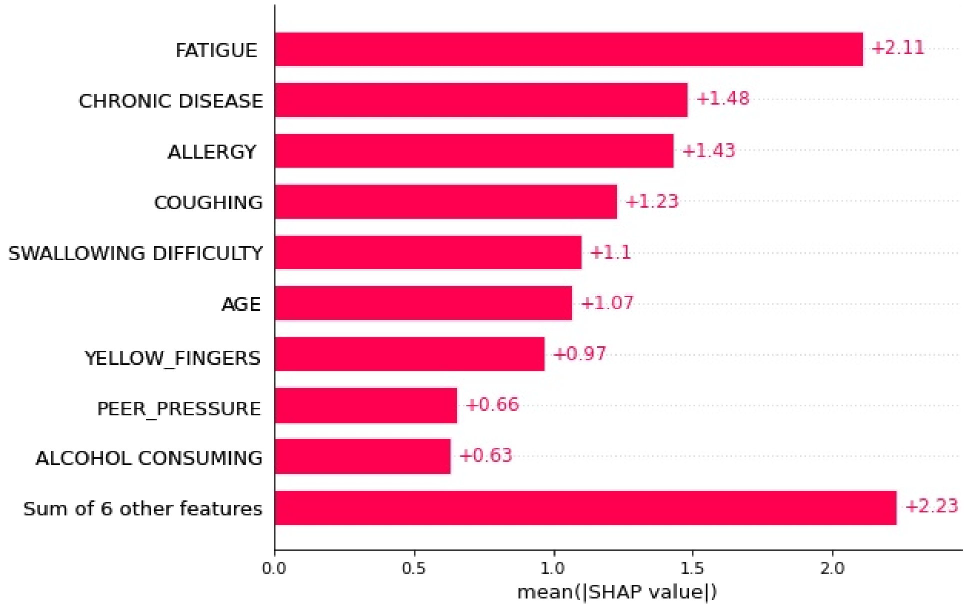
```
explainer = shap.Explainer(GBM, x_train)
shap_values = explainer(x_train)
```

In [51]:

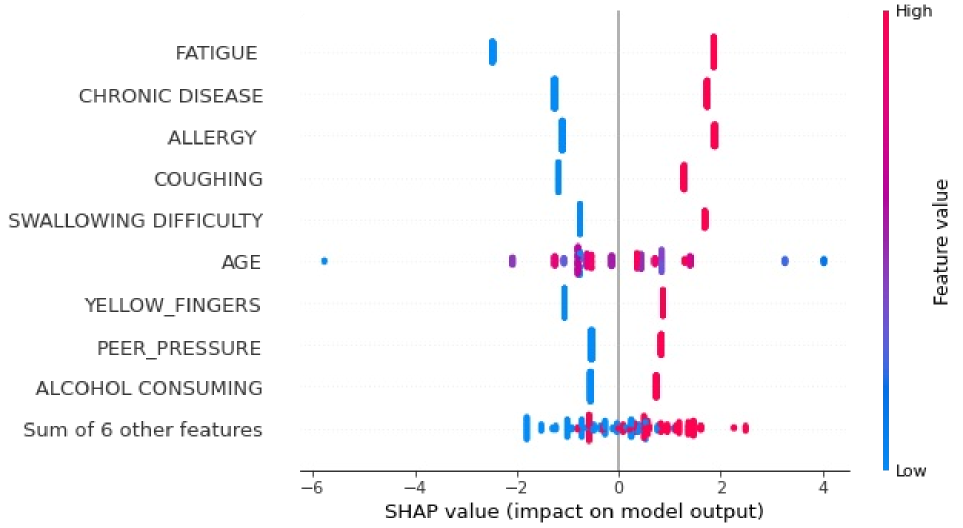
```
shap.plots.waterfall(shap_values[0])
```



```
In [52]: shap.plots.bar(shap_values)
```

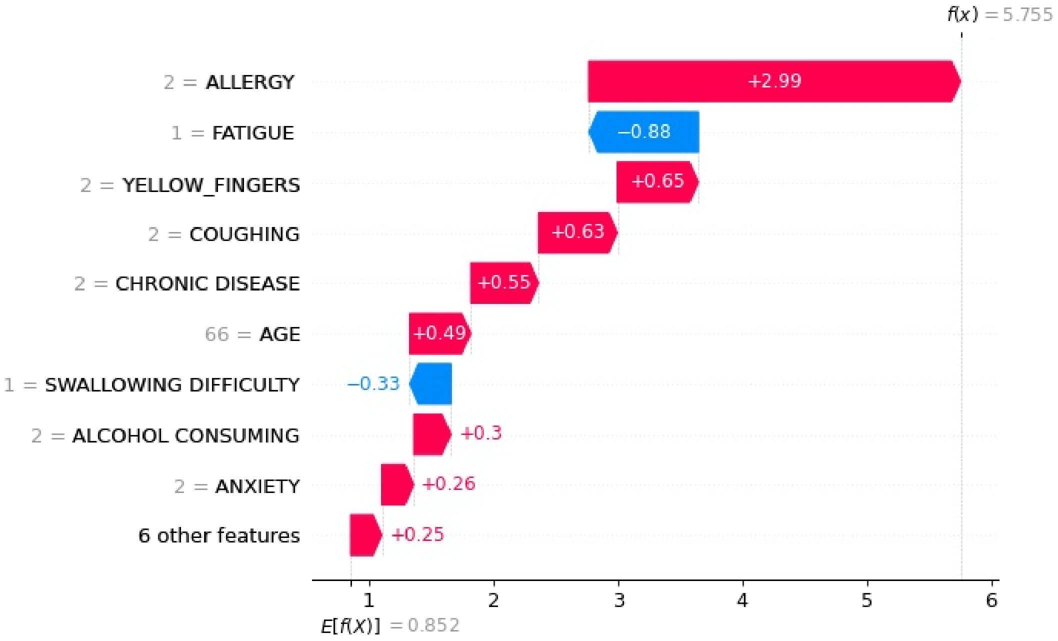


```
In [53]: shap.plots.beeswarm(shap_values)
```

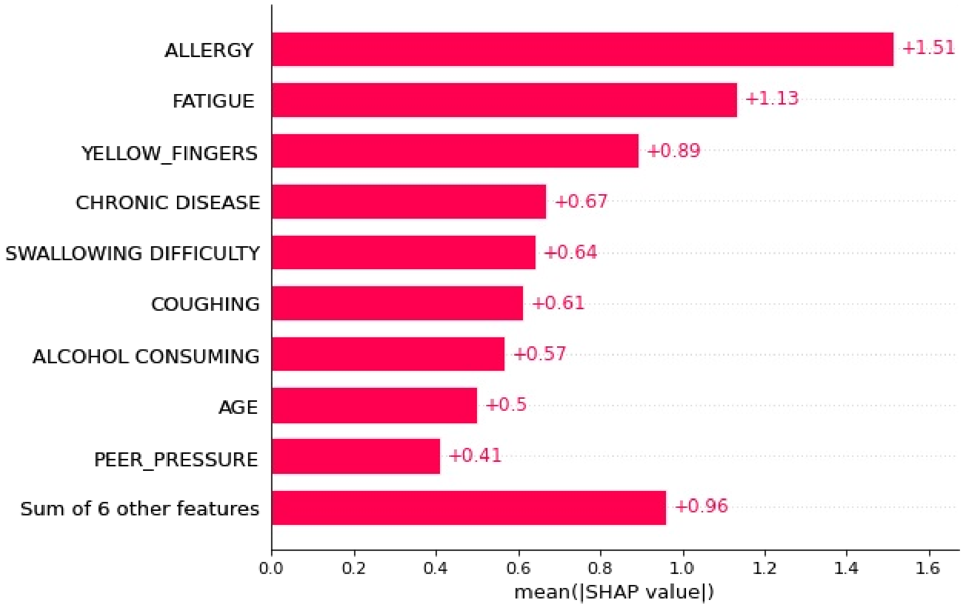


```
In [54]: explainer = shap.Explainer(XGB, x_train)
         shap_values = explainer(x_train)
```

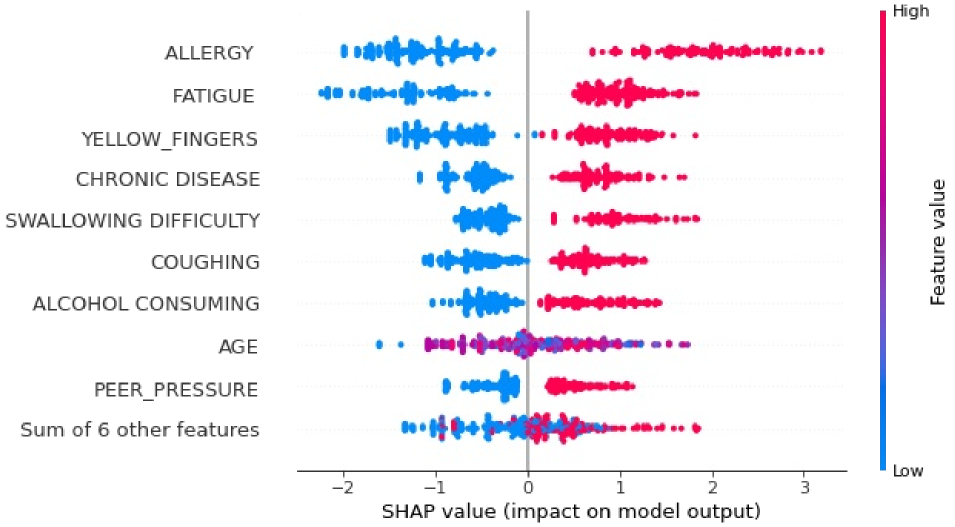
```
In [55]: shap.plots.waterfall(shap_values[0])
```



In [56]:
```
shap.plots.bar(shap_values)
```

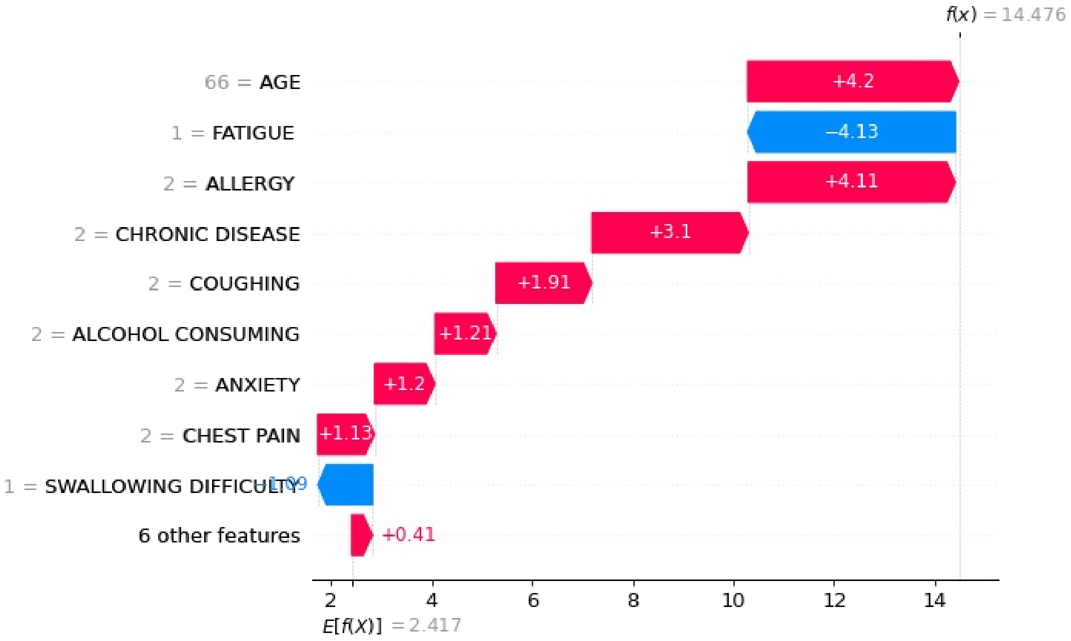


In [57]:
```
shap.plots.beeswarm(shap_values)
```



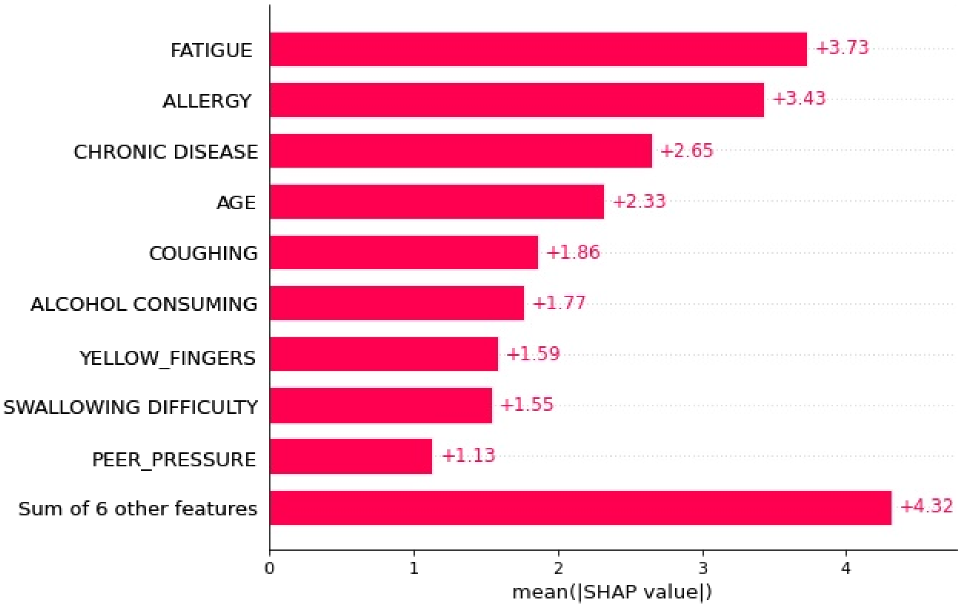
In [58]:
```
explainer = shap.Explainer(LGBM, x_train)
shap_values = explainer(x_train)
```

In [59]:
```
shap.plots.waterfall(shap_values[0])
```



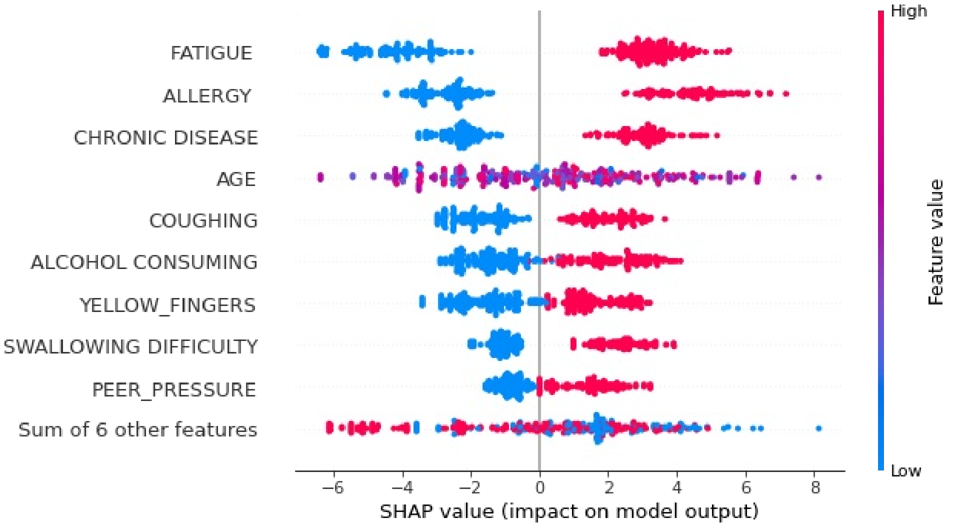
In [60]:

```
shap.plots.bar(shap_values)
```



In [61]:

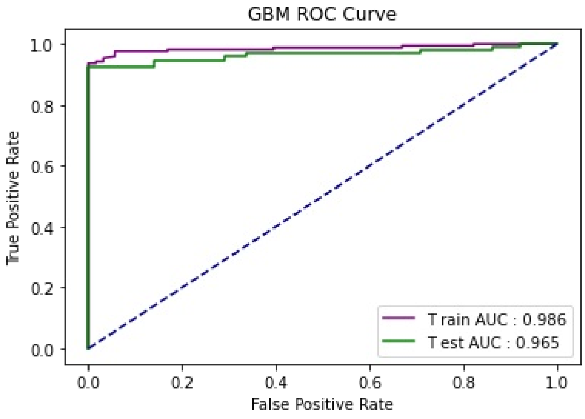
```
shap.plots.beeswarm(shap_values)
```



In [62]:

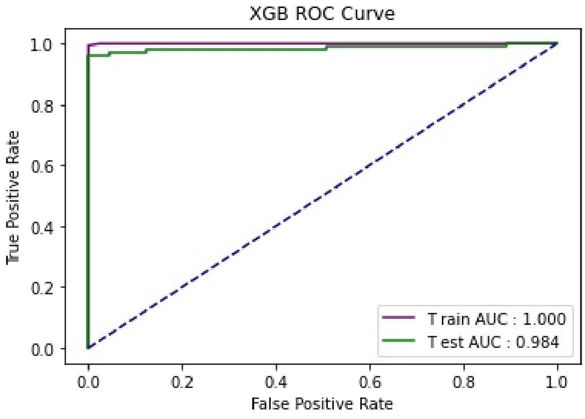
```
GBM_probs_tr = GBM.predict_proba(x_train)[:,1]
auc_tr = roc_auc_score(y_train, GBM_probs_tr)
fpr_tr, tpr_tr, thresholds = roc_curve(y_train, GBM_probs_tr)

GBM_probs_ts = GBM.predict_proba(x_test)[:,1]
auc_ts = roc_auc_score(y_test, GBM_probs_ts)
fpr_ts, tpr_ts, thresholds = roc_curve(y_test, GBM_probs_ts)
```

```
plt.plot(fpr_tr,tpr_tr, color='purple', label="T rain AUC : {:.3f}".format(auc_tr))
plt.plot(fpr_ts,tpr_ts, color='green', label="T est AUC : {:.3f}".format(auc_ts))
plt.plot([0,1], [0,1], color='darkblue', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('GBM ROC Curve')
plt.legend()
plt.show()
```



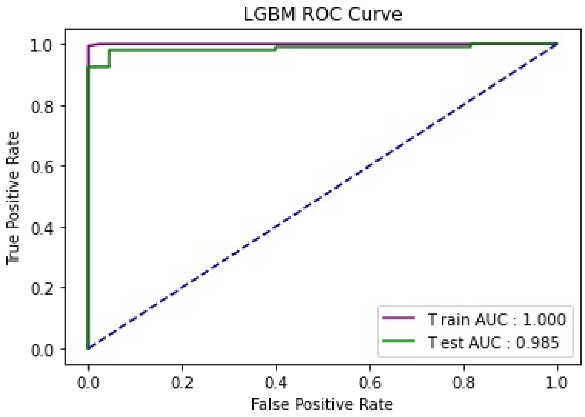
In [63]:

```
XGB_probs_tr = XGB.predict_proba(x_train)[:,1]
auc_tr = roc_auc_score(y_train, XGB_probs_tr)
fpr_tr, tpr_tr, thresholds = roc_curve(y_train, XGB_probs_tr)

XGB_probs_ts = XGB.predict_proba(x_test)[:,1]
auc_ts = roc_auc_score(y_test, XGB_probs_ts)
fpr_ts, tpr_ts, thresholds = roc_curve(y_test, XGB_probs_ts)

plt.plot(fpr_tr,tpr_tr, color='purple', label="T rain AUC : {:.3f}".format(auc_tr))
plt.plot(fpr_ts,tpr_ts, color='green', label="T est AUC : {:.3f}".format(auc_ts))
plt.plot([0,1], [0,1], color='darkblue', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('XGB ROC Curve')
plt.legend()
plt.show()
```



In [64]:

```
LGBM_probs_tr = LGBM.predict_proba(x_train)[:,1]
auc_tr = roc_auc_score(y_train, LGBM_probs_tr)
fpr_tr, tpr_tr, thresholds = roc_curve(y_train, LGBM_probs_tr)

LGBM_probs_ts = LGBM.predict_proba(x_test)[:,1]
auc_ts = roc_auc_score(y_test, LGBM_probs_ts)
fpr_ts, tpr_ts, thresholds = roc_curve(y_test, LGBM_probs_ts)

plt.plot(fpr_tr,tpr_tr, color='purple', label="T rain AUC : {:.3f}".format(auc_tr))
plt.plot(fpr_ts,tpr_ts, color='green', label="T est AUC : {:.3f}".format(auc_ts))
plt.plot([0,1], [0,1], color='darkblue', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('LGBM ROC Curve')
plt.legend()
plt.show()
```



In [65]:

```
# import pickle
# pickle.dump(GBM, open("GBM_lung_cancer_model.pkl", "wb"))
```

In [ ]:

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