lung_cancer

June 8, 2022

[ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

[ ]:

```
#  read the dataset or import the data using pandas
#  1 -- > NO
# 2 -- > YES

df = pd.read_csv('C:/Users/91861/Downloads/survey lung cancer.csv')
df.head()
```

[ ]:

```
  GENDER  AGE  SMOKING  YELLOW_FINGERS  ANXIETY  PEER_PRESSURE  \
0      M   69        1               2        2              1
1      M   74        2               1        1              1
2      F   59        1               1        1              2
3      M   63        2               2        2              1
4      F   63        1               2        1              1

   CHRONIC DISEASE  FATIGUE   ALLERGY   WHEEZING  ALCOHOL CONSUMING  COUGHING  \
0                1         2         1         2                  2         2
1                2         2         2         1                  1         1
2                1         2         1         2                  1         2
3                1         1         1         1                  2         1
4                1         1         1         2                  1         2

   SHORTNESS OF BREATH  SWALLOWING DIFFICULTY  CHEST PAIN LUNG_CANCER  \
0                    2                      2           2         YES
1                    2                      2           2         YES
2                    2                      1           2          NO
3                    1                      2           2          NO
4                    2                      1           1          NO

   Unnamed: 16
```

```
0          NaN
1          NaN
2          NaN
3          NaN
4          NaN
```

```
[ ]: df.drop(columns=df.columns[-1],
             axis=1,
             inplace=True)
```

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

```
[ ]:   GENDER  AGE  SMOKING  YELLOW_FINGERS  ANXIETY  PEER_PRESSURE  \
     0      M   69        1               2        2              1
     1      M   74        2               1        1              1
     2      F   59        1               1        1              2
     3      M   63        2               2        2              1
     4      F   63        1               2        1              1

        CHRONIC DISEASE  FATIGUE   ALLERGY   WHEEZING  ALCOHOL CONSUMING  COUGHING  \
     0                1        2         1          2                  2         2
     1                2        2         2          1                  1         1
     2                1        2         1          2                  1         2
     3                1        1         1          1                  2         1
     4                1        1         1          2                  1         2

        SHORTNESS OF BREATH  SWALLOWING DIFFICULTY  CHEST PAIN LUNG_CANCER
     0                    2                      2            2         YES
     1                    2                      2            2         YES
     2                    2                      1            2          NO
     3                    1                      2            2          NO
     4                    2                      1            1          NO
```

```
[ ]: # Exploratory Data Analysis
     # Now, let's see the size of the dataset

     df.shape
```

```
[ ]: (309, 16)
```

```
[ ]: df.info()

     # Out of 16 features, we have 14 int types and only two with the string data␣
      ↪types.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
```

```
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   GENDER                 309 non-null    object
 1   AGE                    309 non-null    int64 
 2   SMOKING                309 non-null    int64 
 3   YELLOW_FINGERS         309 non-null    int64 
 4   ANXIETY                309 non-null    int64 
 5   PEER_PRESSURE          309 non-null    int64 
 6   CHRONIC DISEASE        309 non-null    int64 
 7   FATIGUE                309 non-null    int64 
 8   ALLERGY                309 non-null    int64 
 9   WHEEZING               309 non-null    int64 
 10  ALCOHOL CONSUMING      309 non-null    int64 
 11  COUGHING               309 non-null    int64 
 12  SHORTNESS OF BREATH    309 non-null    int64 
 13  SWALLOWING DIFFICULTY  309 non-null    int64 
 14  CHEST PAIN             309 non-null    int64 
 15  LUNG_CANCER            309 non-null    object
dtypes: int64(14), object(2)
memory usage: 38.8+ KB
```

[ ]:

```
df.isnull().sum()

# Woah! Fortunately, this dataset doesn't hold any missing values.
```

[ ]:

```
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
```

[ ]:

```
df.describe()
```

```
# As we are getting some information from each feature so let's see how␣
 ↪statistically the dataset is spread
```

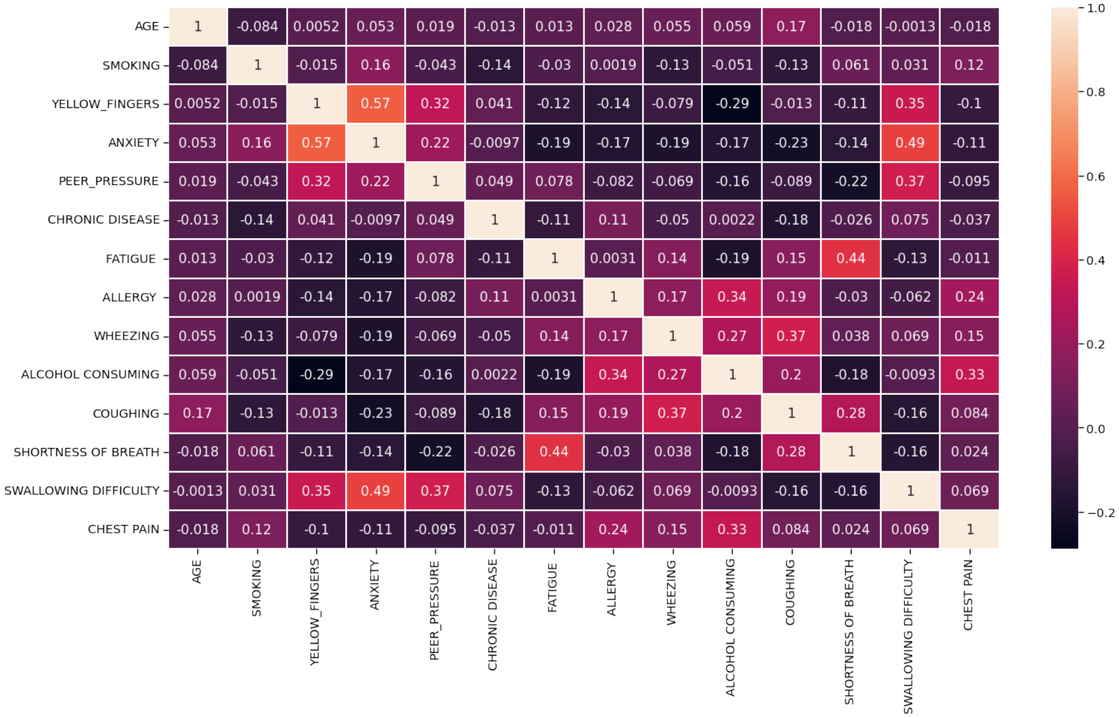
[ ]:
```
              AGE     SMOKING  YELLOW_FINGERS     ANXIETY  PEER_PRESSURE  \
count  309.000000  309.000000      309.000000  309.000000     309.000000
mean    62.673139    1.563107        1.569579    1.498382       1.501618
std      8.210301    0.496806        0.495938    0.500808       0.500808
min     21.000000    1.000000        1.000000    1.000000       1.000000
25%     57.000000    1.000000        1.000000    1.000000       1.000000
50%     62.000000    2.000000        2.000000    1.000000       2.000000
75%     69.000000    2.000000        2.000000    2.000000       2.000000
max     87.000000    2.000000        2.000000    2.000000       2.000000

       CHRONIC DISEASE     FATIGUE     ALLERGY     WHEEZING  ALCOHOL CONSUMING  \
count       309.000000  309.000000  309.000000  309.000000         309.000000
mean          1.504854    1.673139    1.556634    1.556634           1.556634
std           0.500787    0.469827    0.497588    0.497588           0.497588
min           1.000000    1.000000    1.000000    1.000000           1.000000
25%           1.000000    1.000000    1.000000    1.000000           1.000000
50%           2.000000    2.000000    2.000000    2.000000           2.000000
75%           2.000000    2.000000    2.000000    2.000000           2.000000
max           2.000000    2.000000    2.000000    2.000000           2.000000

         COUGHING  SHORTNESS OF BREATH  SWALLOWING DIFFICULTY  CHEST PAIN
count  309.000000           309.000000             309.000000  309.000000
mean     1.579288             1.640777               1.469256    1.556634
std      0.494474             0.480551               0.499863    0.497588
min      1.000000             1.000000               1.000000    1.000000
25%      1.000000             1.000000               1.000000    1.000000
50%      2.000000             2.000000               1.000000    2.000000
75%      2.000000             2.000000               2.000000    2.000000
max      2.000000             2.000000               2.000000    2.000000
```

[ ]:
```
df.corr().head()

# It is always better to check the correlation between the features,
# so that we can analyze that which feature is negatively correlated and which␣
 ↪is positively correlated
# Let's check the correlation between various features.
```

[ ]:
```
                     AGE   SMOKING  YELLOW_FINGERS   ANXIETY  PEER_PRESSURE  \
AGE             1.000000 -0.084475        0.005205  0.053170       0.018685
SMOKING        -0.084475  1.000000       -0.014585  0.160267      -0.042822
YELLOW_FINGERS  0.005205 -0.014585        1.000000  0.565829       0.323083
ANXIETY         0.053170  0.160267        0.565829  1.000000       0.216841
PEER_PRESSURE   0.018685 -0.042822        0.323083  0.216841       1.000000
```

```
                CHRONIC DISEASE  FATIGUE   ALLERGY   WHEEZING  \
AGE                   -0.012642  0.012614  0.027990  0.055011
SMOKING               -0.141522 -0.029575  0.001913 -0.129426
YELLOW_FINGERS         0.041122 -0.118058 -0.144300 -0.078515
ANXIETY               -0.009678 -0.188538 -0.165750 -0.191807
PEER_PRESSURE          0.048515  0.078148 -0.081800 -0.068771

                ALCOHOL CONSUMING  COUGHING  SHORTNESS OF BREATH  \
AGE                      0.058985  0.169950            -0.017513
SMOKING                 -0.050623 -0.129471             0.061264
YELLOW_FINGERS          -0.289025 -0.012640            -0.105944
ANXIETY                 -0.165750 -0.225644            -0.144077
PEER_PRESSURE           -0.159973 -0.089019            -0.220175

                SWALLOWING DIFFICULTY  CHEST PAIN
AGE                         -0.001270   -0.018104
SMOKING                      0.030718    0.120117
YELLOW_FINGERS               0.345904   -0.104829
ANXIETY                      0.489403   -0.113634
PEER_PRESSURE                0.366590   -0.094828
```

```
[ ]: plt.figure(figsize=(20,12))
     sns.set_context('notebook',font_scale = 1.3)
     sns.heatmap(df.corr(),annot=True,linewidth =2)
     plt.tight_layout()
```

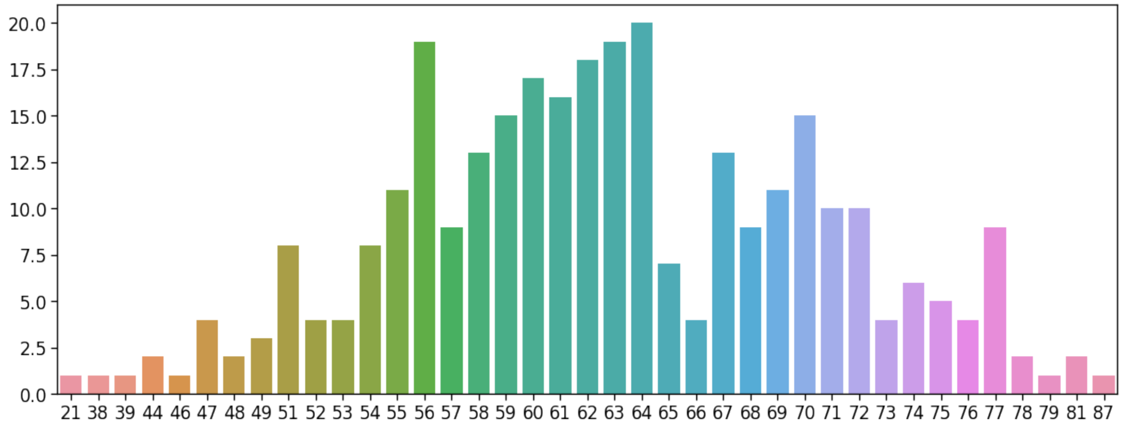


[ ]:

```
# Getting Started

# Title : Lung Cancer Prediction

# Lung Cancer Status :

# 0 -- > Yes

# 1 -- > NO

df.replace({"LUNG_CANCER":{'YES':0,'NO':1}},inplace=True)
```

[ ]:

```
# Age Analysis

plt.figure(figsize=(15,6))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=df.AGE.value_counts().index,y=df.AGE.value_counts().values)
plt.tight_layout()
```



[ ]:

```
df['AGE'].value_counts()
```

[ ]:

```
64     20
63     19
56     19
62     18
60     17
61     16
59     15
70     15
67     13
```

```
58    13
69    11
55    11
72    10
71    10
68     9
57     9
77     9
51     8
54     8
65     7
74     6
75     5
76     4
52     4
53     4
73     4
47     4
66     4
49     3
81     2
78     2
44     2
48     2
21     1
79     1
38     1
39     1
87     1
46     1
Name: AGE, dtype: int64
```

[ ]:

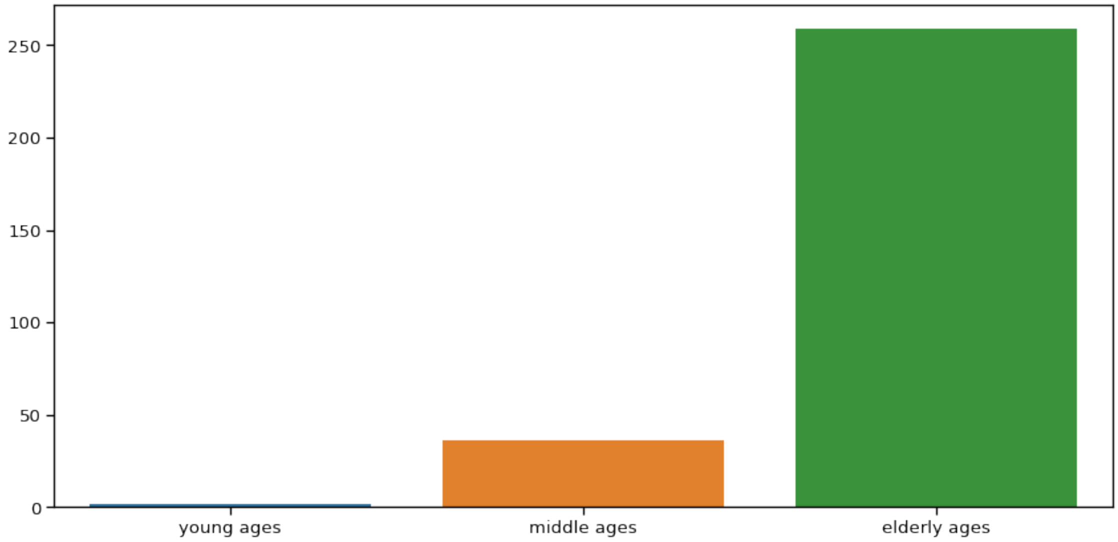
```
minAge=min(df.AGE)
maxAge=max(df.AGE)
meanAge=df.AGE.mean()
print('Min Age :',minAge)
print('Max Age :',maxAge)
print('Mean Age :',meanAge)
print()
```

```
Min Age : 21
Max Age : 87
Mean Age : 62.67313915857605
```

[ ]:

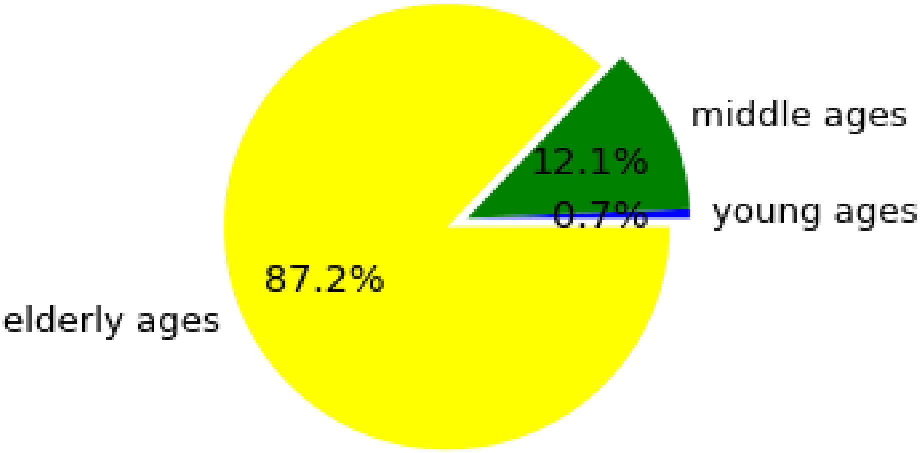
```
# We should divide the Age feature into three parts - "Young", "Middle" and "Elder"
```

```
Young = df[(df.AGE>=29)&(df.AGE<40)]
Middle = df[(df.AGE>=40)&(df.AGE<55)]
Elder = df[(df.AGE>55)]

plt.figure(figsize=(12,6))
sns.set_context('notebook',font_scale = 1.2)
sns.barplot(x=['young ages','middle ages','elderly ages'],y=[len(Young),len(Middle),len(Elder)])
plt.tight_layout()
```



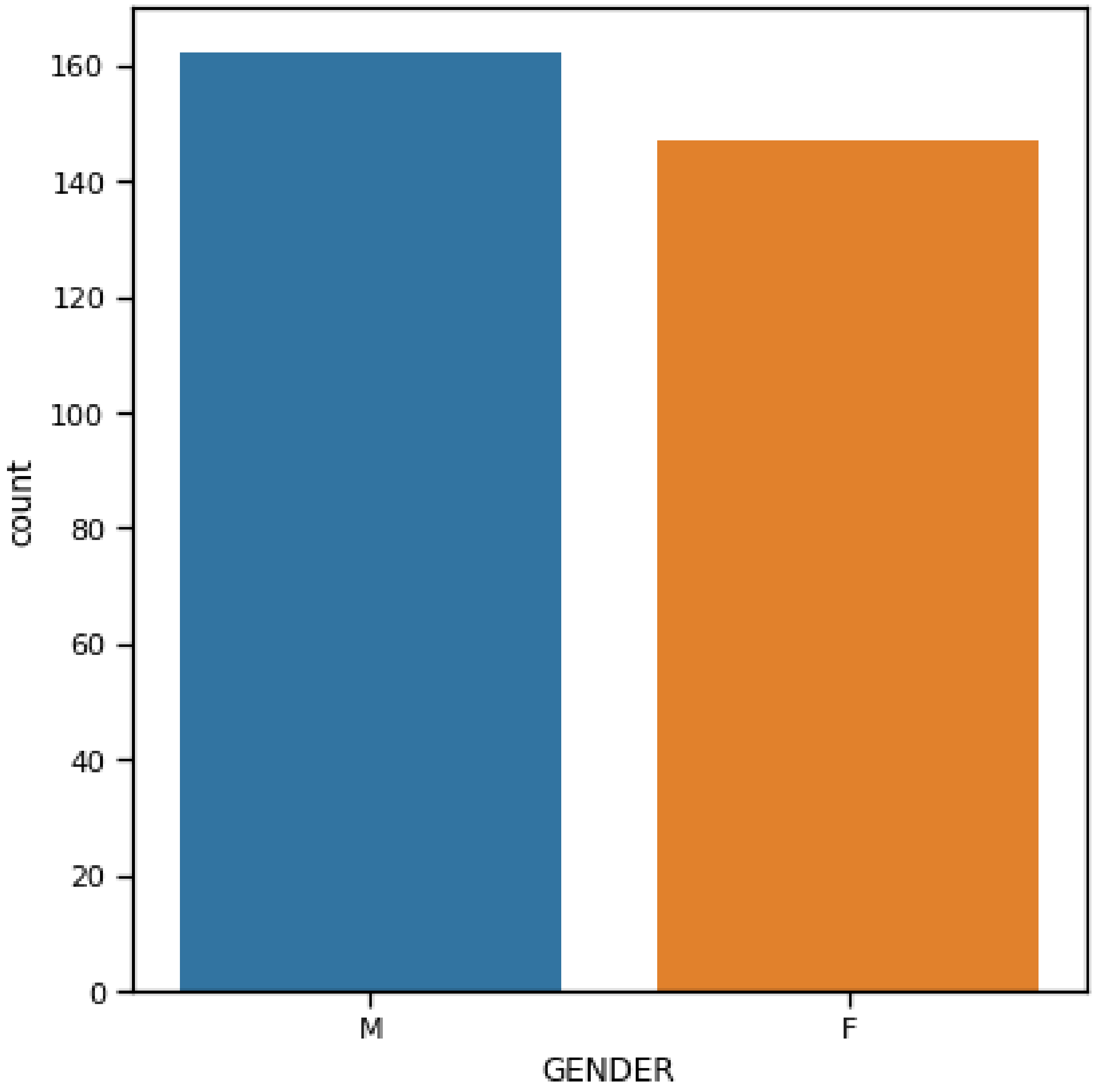
[ ]:

```
# Inference: Here we can see that elder people are the most affected by heart disease and young ones are the least affected.

# To prove the above inference we will plot the pie chart.

colors = ['blue','green','yellow']
explode = [0,0,0.1]
plt.figure(figsize=(5,5))
sns.set_context('notebook',font_scale = 1.2)
plt.pie([len(Young),len(Middle),len(Elder)],labels=['young ages','middle ages','elderly ages'],explode=explode,colors=colors, autopct='%1.1f%%')
plt.tight_layout()
```



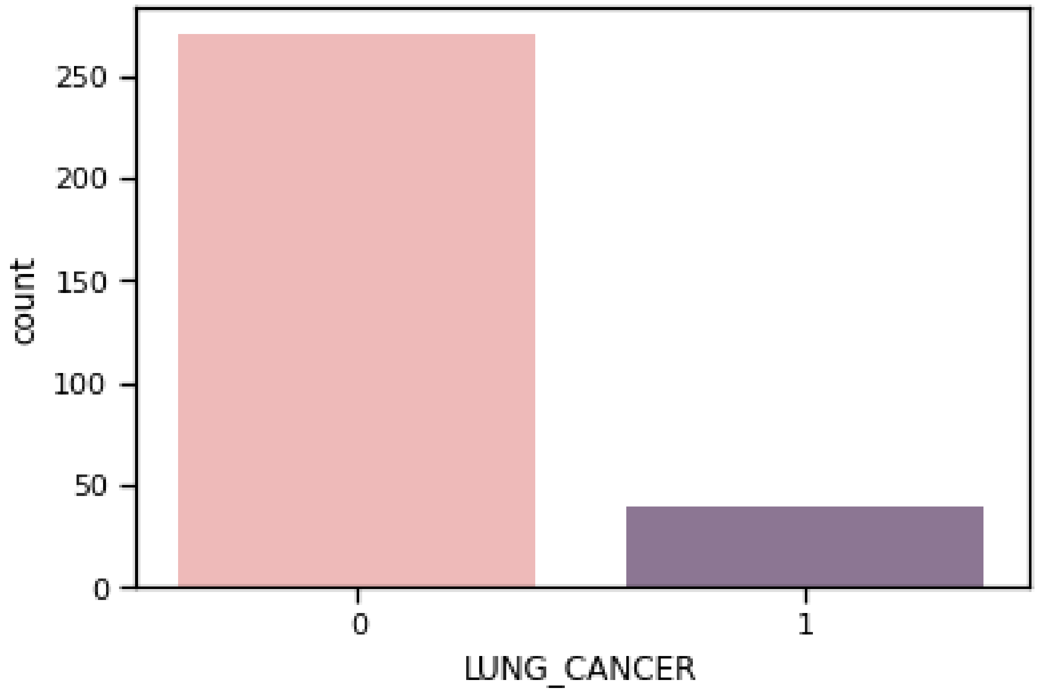
[ ]:

```
# Sex - Feature Analysis

plt.figure(figsize=(6,6))
sns.set_context('notebook',font_scale = 1)
sns.countplot(df['GENDER'])
plt.tight_layout()

# Inference: Here it is clearly visible that, Ratio of Male to Female almost equal but male is higher.
```



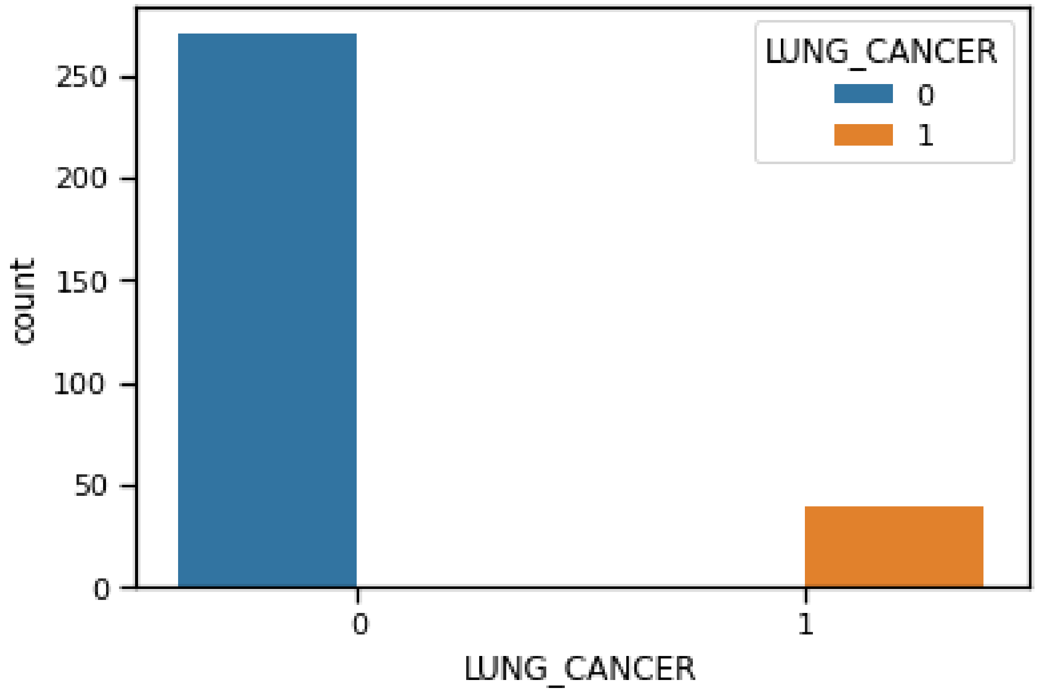
```
[ ]: # Lung cancer - Feature Analysis
     # Most affected cancer

     colours=["#f7b2b0","#8f7198", "#003f5c"]
     sns.countplot(data= df, x="LUNG_CANCER",palette=colours)
```

```
[ ]: <AxesSubplot:xlabel='LUNG_CANCER', ylabel='count'>
```



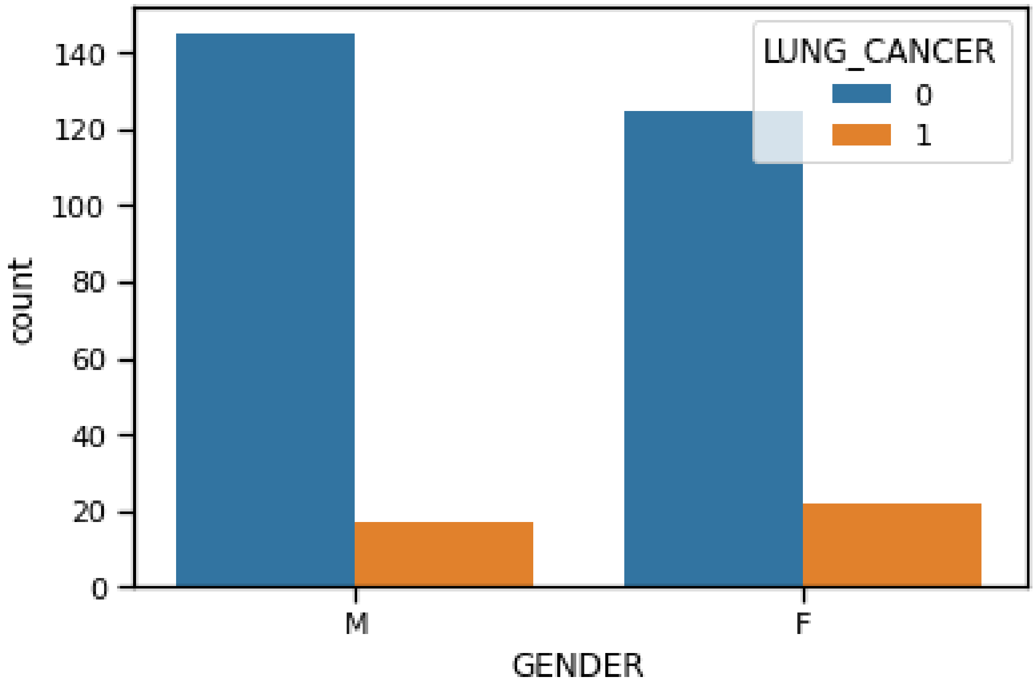
```
[ ]: sns.countplot(x='LUNG_CANCER',hue='LUNG_CANCER',data=df)
```

```
[ ]: <AxesSubplot:xlabel='LUNG_CANCER', ylabel='count'>
```



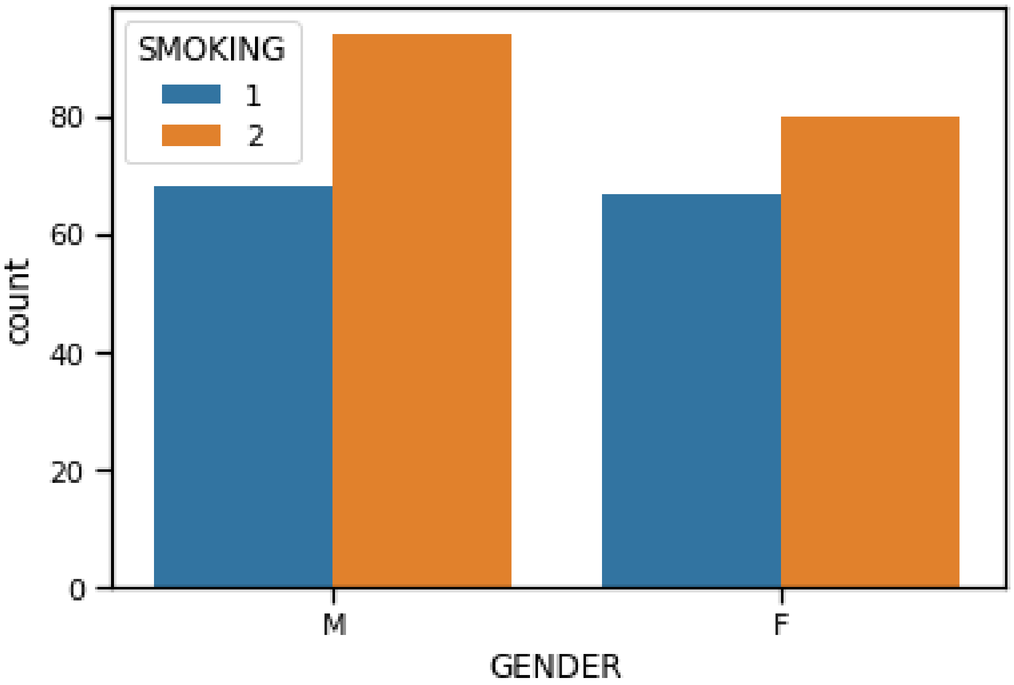
[ ]:
```
sns.countplot(x='GENDER',hue='LUNG_CANCER',data=df)
```

[ ]:
```
<AxesSubplot:xlabel='GENDER', ylabel='count'>
```



[ ]:
```
sns.countplot(x='GENDER',hue='SMOKING',data=df)

df['SMOKING'].value_counts()

# MALE SMOKING MORE THAN FEMALE
```

[ ]:
```
2    174
1    135
Name: SMOKING, dtype: int64
```



[ ]:
```
df['LUNG_CANCER'].value_counts()
```

[ ]:
```
0    270
1     39
Name: LUNG_CANCER, dtype: int64
```

[ ]:
```
#  encode the Gender - Feature Analysis

df.replace({"GENDER":{'M':0,'F':1}},inplace=True)
df.head(5)
```

[ ]:
```
   GENDER  AGE  SMOKING  YELLOW_FINGERS  ANXIETY  PEER_PRESSURE  \
0       0   69        1               2        2              1
1       0   74        2               1        1              1
2       1   59        1               1        1              2
3       0   63        2               2        2              1
4       1   63        1               2        1              1

   CHRONIC DISEASE  FATIGUE   ALLERGY   WHEEZING  ALCOHOL CONSUMING  COUGHING  \
0                1        2         1         2                  2         2
1                2        2         2         1                  1         1
2                1        2         1         2                  1         2
3                1        1         1         1                  2         1
4                1        1         1         2                  1         2
```

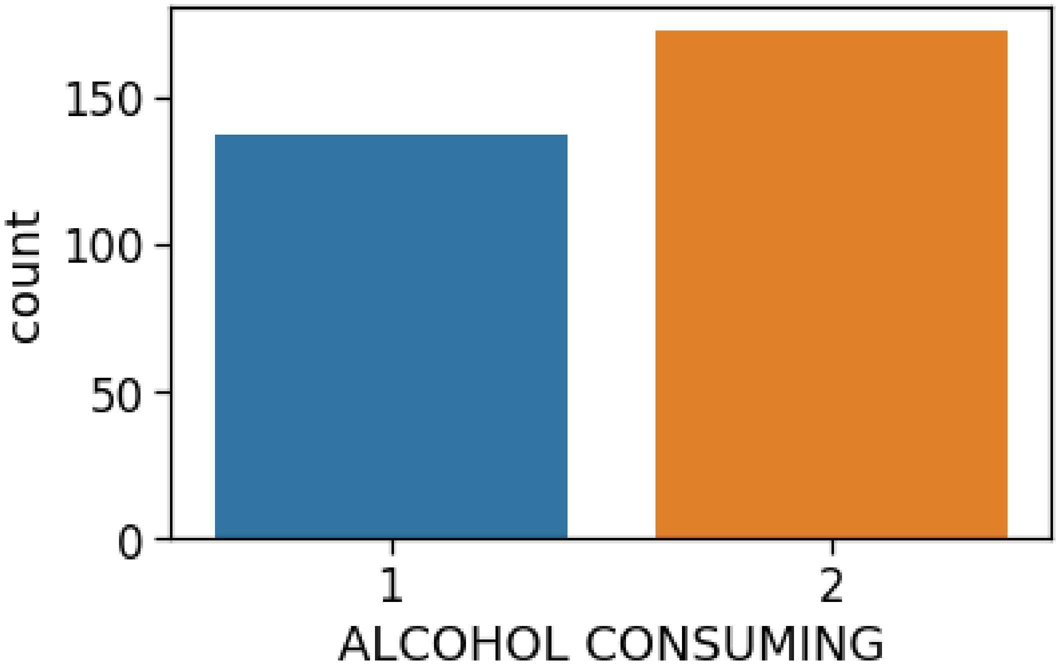
```
   SHORTNESS OF BREATH  SWALLOWING DIFFICULTY  CHEST PAIN  LUNG_CANCER
0                    2                      2           2            0
1                    2                      2           2            0
2                    2                      1           2            1
3                    1                      2           2            1
4                    2                      1           1            1
```

[ ]:

```
df['GENDER']=df['GENDER'].astype(np.int64)
```

[ ]:

```
# Alcohol consuming - Feature Analysis

sns.set_context('notebook',font_scale = 1.5)
sns.countplot(df['ALCOHOL CONSUMING'])
plt.tight_layout()
```



[ ]:

```
# TARGET VARIABLE IS LUNG CANCER

X = df.drop(columns=['LUNG_CANCER'],axis=1)
y = df['LUNG_CANCER']
```

[ ]:

```
print("The shape of X is " , X.shape)
print("The shape of Y is " , y.shape)
```

```
The shape of X is  (309, 15)
The shape of Y is  (309,)
```

[ ]:

```
from sklearn.model_selection import train_test_split

# separating into train and testing

X_train, X_test,Y_train,Y_test = train_test_split(X,y,test_size=0.
 ↪2,stratify=y,random_state=42)
print("Shape of X_train is " ,X_train.shape)
print("Shape of X_test  is " ,X_test.shape)
print("Shape of Y_train is " ,Y_train.shape)
print("Shape of Y_test  is " ,Y_test.shape)
```

```
Shape of X_train is  (247, 15)
Shape of X_test  is  (62, 15)
Shape of Y_train is  (247,)
Shape of Y_test  is  (62,)
```

[ ]:

```
print(Y_train.value_counts())
print(Y_test.value_counts())
```

```
0    216
1     31
Name: LUNG_CANCER, dtype: int64
0    54
1     8
Name: LUNG_CANCER, dtype: int64
```

[ ]:

```
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.metrics import precision_score, recall_score, confusion_matrix,␣
 ↪classification_report, accuracy_score, f1_score
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
```

[ ]:

```
#pipelines of models( it is short was to fit and pred)

pipeline_lr=Pipeline([('lr_classifier',LogisticRegression(random_state=42))])
```

```
pipeline_dt=Pipeline([␣
 ↪('dt_classifier',DecisionTreeClassifier(random_state=42))])

pipeline_rf=Pipeline([('rf_classifier',RandomForestClassifier())])

pipeline_svc=Pipeline([('sv_classifier',SVC())])

pipeline_knn=Pipeline([('KNN_Classifier',KNeighborsClassifier())])

# List of all the pipelines

pipelines = [pipeline_lr, pipeline_dt, pipeline_rf, pipeline_svc, pipeline_knn]

# Dictionary of pipelines and classifier types for ease of reference

pipe_dict = {0: 'Logistic Regression', 1: 'Decision Tree', 2: 'RandomForest', 3:
 ↪ "SVC", 4:'Knn Neighbours'}


# # Fit the pipelines

# for pipe in pipelines:
#     pipe.fit(X_train, Y_train)
```

[ ]:
```
df['GENDER'].unique()
```

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

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309 entries, 0 to 308
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   GENDER                  309 non-null    int64
 1   AGE                     309 non-null    int64
 2   SMOKING                 309 non-null    int64
 3   YELLOW_FINGERS          309 non-null    int64
 4   ANXIETY                 309 non-null    int64
 5   PEER_PRESSURE           309 non-null    int64
 6   CHRONIC DISEASE         309 non-null    int64
 7   FATIGUE                 309 non-null    int64
 8   ALLERGY                 309 non-null    int64
 9   WHEEZING                309 non-null    int64
 10  ALCOHOL CONSUMING       309 non-null    int64
 11  COUGHING                309 non-null    int64
 12  SHORTNESS OF BREATH    309 non-null    int64
 13  SWALLOWING DIFFICULTY  309 non-null    int64
 14  CHEST PAIN             309 non-null    int64
 15  LUNG_CANCER            309 non-null    int64
dtypes: int64(16)
memory usage: 38.8 KB
```

[ ]:

```
# Fit the pipelines

for pipe in pipelines:
    pipe.fit(X_train, Y_train)
```

[ ]:

```
#cross validation on accuracy

cv_results_accuracy = []
for i, model in enumerate(pipelines):
    cv_score = cross_val_score(model, X_train,Y_train, cv=10 )
    cv_results_accuracy.append(cv_score)
    print("%s: %f " % (pipe_dict[i], cv_score.mean()))
```

```
Logistic Regression: 0.914833
Decision Tree: 0.858500
RandomForest: 0.906833
SVC: 0.874500
Knn Neighbours: 0.874500
```

[ ]:

```
# LogisticRegression gave more accuracy score so we choose LR

lr = LogisticRegression()
lr.fit(X_train, Y_train)
y_pred = lr.predict(X_test)

lr_train_acc = accuracy_score(Y_train, lr.predict(X_train))
lr_test_acc = accuracy_score(Y_test, y_pred)
cm = confusion_matrix(Y_test, y_pred)

print(f"Training Accuracy of Logistic Regression Model is {lr_train_acc}")
print(f"Test Accuracy of Logistic Regression Model is {lr_test_acc}")

print(f"confusion_matrix is\n {cm}")
```

```
Training Accuracy of Logistic Regression Model is 0.9392712550607287
Test Accuracy of Logistic Regression Model is 0.9193548387096774
confusion_matrix is
 [[53  1]
 [ 4  4]]
```

[ ]:

```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.98      0.95        54
           1       0.80      0.50      0.62         8

    accuracy                           0.92        62
   macro avg       0.86      0.74      0.79        62
weighted avg       0.91      0.92      0.91        62
```