# Data Exploration & Analysis

In [2]:

*#Importing the basic libraries for analysis*

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.preprocessing **import** LabelEncoder

**import** warnings

In [3]:

df **=** pd.read\_csv("C:\\Users\\GPT BANTWAL\\survey lung cancer.csv") df.head()

Out[3]:

**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

In [4]:

print("Data shape:") print(df.shape)

print("\n")

print("Data summary:") print(df.info())

print("\n")

Data shape:

(309, 16)

Data summary:

<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

None

In [5]:

df.describe()

Out[5]:

**AGE SMOKING YELLOW\_FINGERS ANXIETY PEER\_PRESSURE CHRONIC**

**DISEASE**

**FATIGUE ALLERGY WHEEZING ALC**

**CONSU**

**count** 309.000000 309.000000 309.000000 309.000000 309.000000 309.000000 309.000000 309.000000 309.000000 309.0

**mean** 62.673139 1.563107 1.569579 1.498382 1.501618 1.504854 1.673139 1.556634 1.556634 1.5

**std** 8.210301 0.496806 0.495938 0.500808 0.500808 0.500787 0.469827 0.497588 0.497588 0.4

**min** 21.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.0

**25%** 57.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.0

**50%** 62.000000 2.000000 2.000000 1.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.0

**75%** 69.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.0

**max** 87.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.0

|  |  |  |
| --- | --- | --- |
| In [6]: | df.isnull().sum() |  |
| Out[6]: | 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 [7]:

print(df.duplicated().sum())

df.drop\_duplicates(inplace**=True**) print

33

Out[7]: <function print(\*args, sep=' ', end='\n', file=None, flush=False)>

In [8]:

print("Data shape:") print(df.shape)

print("\n")

print("Data summary:") print(df.info())

print("\n")

Data shape:

(276, 16)

Data summary:

<class 'pandas.core.frame.DataFrame'> Int64Index: 276 entries, 0 to 283

Data columns (total 16 columns):

# Column Non-Null Count Dtype

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 |  | GENDER | 276 | non-null |  | object |
| 1 |  | AGE | 276 | non-null |  | int64 |
| 2 |  | SMOKING | 276 | non-null |  | int64 |
| 3 |  | YELLOW\_FINGERS | 276 | non-null |  | int64 |
| 4 |  | ANXIETY | 276 | non-null |  | int64 |
| 5 |  | PEER\_PRESSURE | 276 | non-null |  | int64 |
| 6 |  | CHRONIC DISEASE | 276 | non-null |  | int64 |
| 7 |  | FATIGUE | 276 | non-null |  | int64 |
| 8 |  | ALLERGY | 276 | non-null |  | int64 |
| 9 |  | WHEEZING | 276 | non-null |  | int64 |
| 10 |  | ALCOHOL CONSUMING | 276 | non-null |  | int64 |
| 11 |  | COUGHING | 276 | non-null |  | int64 |
| 12 |  | SHORTNESS OF BREATH | 276 | non-null |  | int64 |
| 13 |  | SWALLOWING DIFFICULTY | 276 | non-null |  | int64 |
| 14 |  | CHEST PAIN | 276 | non-null |  | int64 |
| 15 |  | LUNG\_CANCER | 276 | non-null |  | object |

dtypes: int64(14), object(2) memory usage: 36.7+ KB

None

In [9]:

df.describe()

Out[9]:

**AGE SMOKING YELLOW\_FINGERS ANXIETY PEER\_PRESSURE CHRONIC**

**DISEASE**

**FATIGUE ALLERGY WHEEZING ALC**

**CONSU**

**count** 276.000000 276.000000 276.000000 276.000000 276.000000 276.000000 276.000000 276.000000 276.000000 276.0

**mean** 62.909420 1.543478 1.576087 1.496377 1.507246 1.521739 1.663043 1.547101 1.547101 1.5

**std** 8.379355 0.499011 0.495075 0.500895 0.500856 0.500435 0.473529 0.498681 0.498681 0.4

**min** 21.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.0

**25%** 57.750000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.0

**50%** 62.500000 2.000000 2.000000 1.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.0

**75%** 69.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.0

**max** 87.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.0

In [10]:

encoder **=** LabelEncoder()

df['LUNG\_CANCER']**=**encoder.fit\_transform(df['LUNG\_CANCER']) df['GENDER']**=**encoder.fit\_transform(df['GENDER'])

df.head()

Out[10]:

**GENDER AGE SMOKING YELLOW\_FINGERS ANXIETY PEER\_PRESSURE CHRONIC**

**DISEASE**

**FATIGUE ALLERGY WHEEZING ALCOHOL**

**CONSUMING**

**0** 1 69 1 2 2 1 1 2 1 2 2

**1** 1 74 2 1 1 1 2 2 2 1 1

**2** 0 59 1 1 1 2 1 2 1 2 1

**3** 1 63 2 2 2 1 1 1 1 1 2

**4** 0 63 1 2 1 1 1 1 1 2 1

In [11]:

sum **=** 0

count **=** 0

**for** age **in** df['AGE']: sum **+=** age

count **+=**1 print(count)

avg **=** sum**/**count print(avg)

276

62.90942028985507

In [12]:

con\_col **=** ['AGE'] cat\_col**=**[]

**for** i **in** df.columns:

**if** i**!=**'AGE':

cat\_col.append(i)

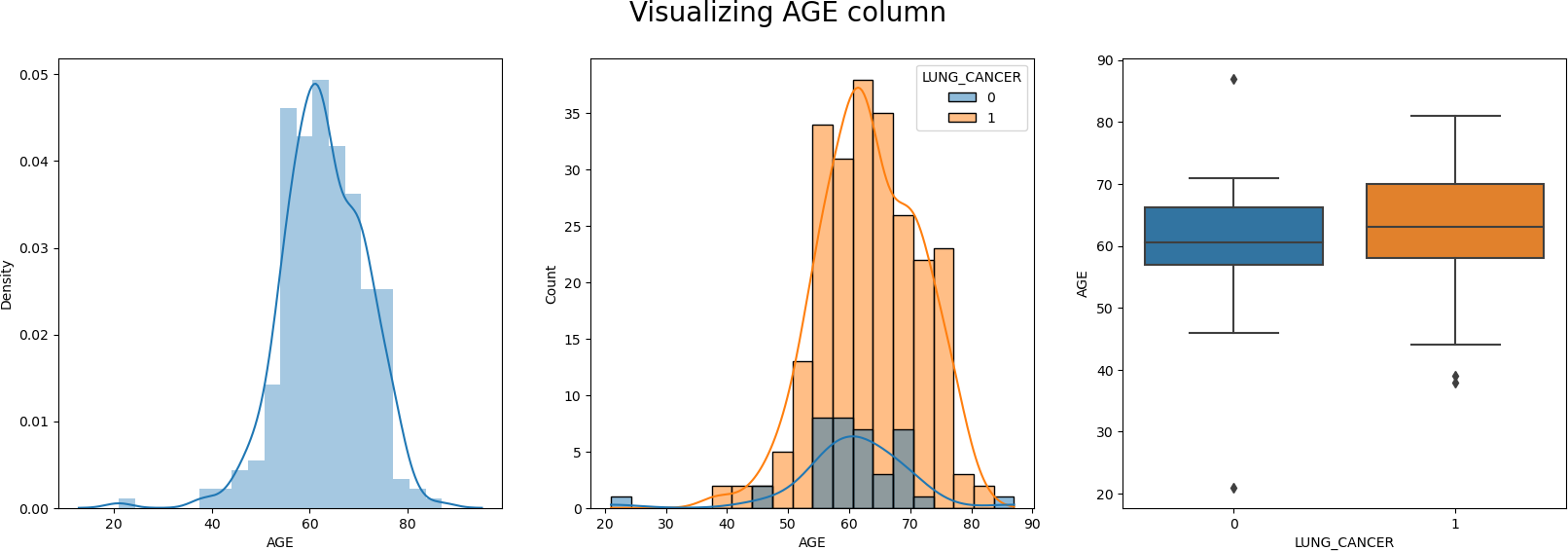
In [13]:

warnings.filterwarnings('ignore')

fig,ax **=** plt.subplots(1,3,figsize**=**(20,6)) sns.distplot(df['AGE'],ax**=**ax[0])

sns.histplot(data **=**df,x**=**'AGE',ax**=**ax[1],hue**=**'LUNG\_CANCER',kde**=True**) sns.boxplot(x**=**df['LUNG\_CANCER'],y**=**df['AGE'],ax**=**ax[2])

plt.suptitle("Visualizing AGE column",size**=**20) plt.show()



In [14]:

fig,ax **=** plt.subplots(15,2,figsize**=**(30,90))

**for** index,i **in** enumerate(cat\_col):

sns.countplot(data**=**df,x**=**i,ax**=**ax[index,0])

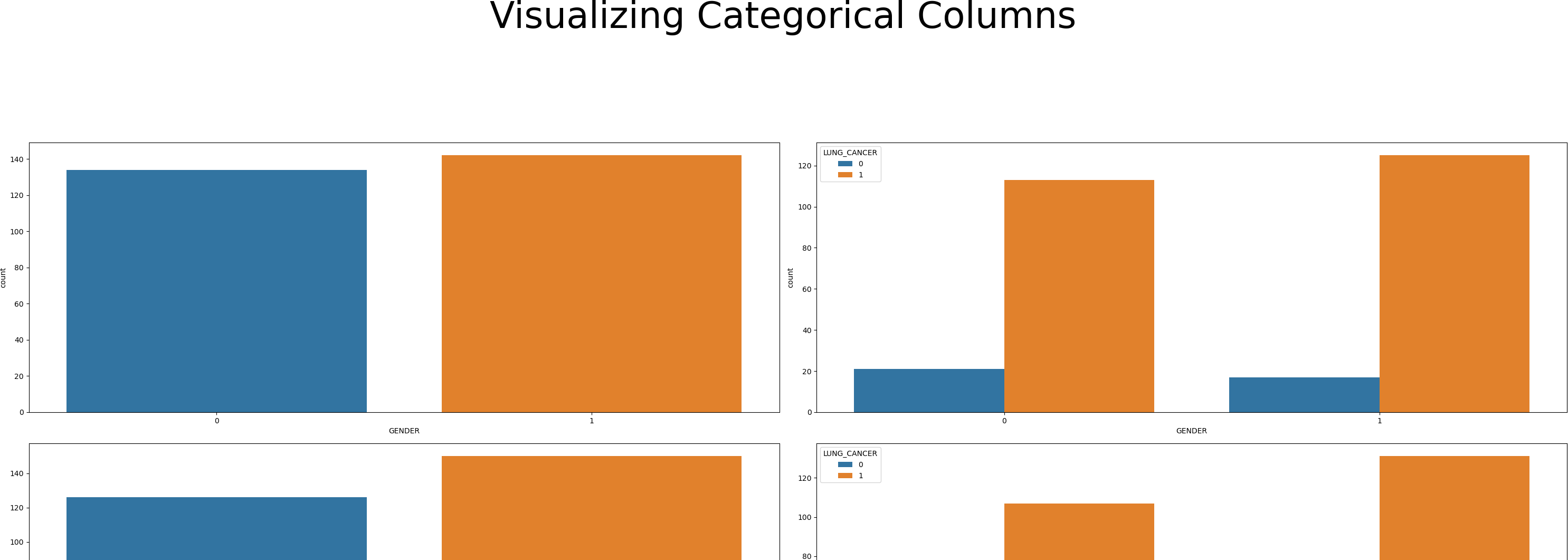
sns.countplot(data**=**df,x**=**i,ax**=**ax[index,1],hue**=**'LUNG\_CANCER') fig.tight\_layout()

fig.subplots\_adjust(top**=**0.95)

plt.suptitle("Visualizing Categorical Columns",fontsize**=**50)

Out[14]: Text(0.5, 0.98, 'Visualizing Categorical Columns')





In [15]:

fig,ax **=** plt.subplots(15,3,figsize**=**(30,90))

**for** index,i **in** enumerate(cat\_col):

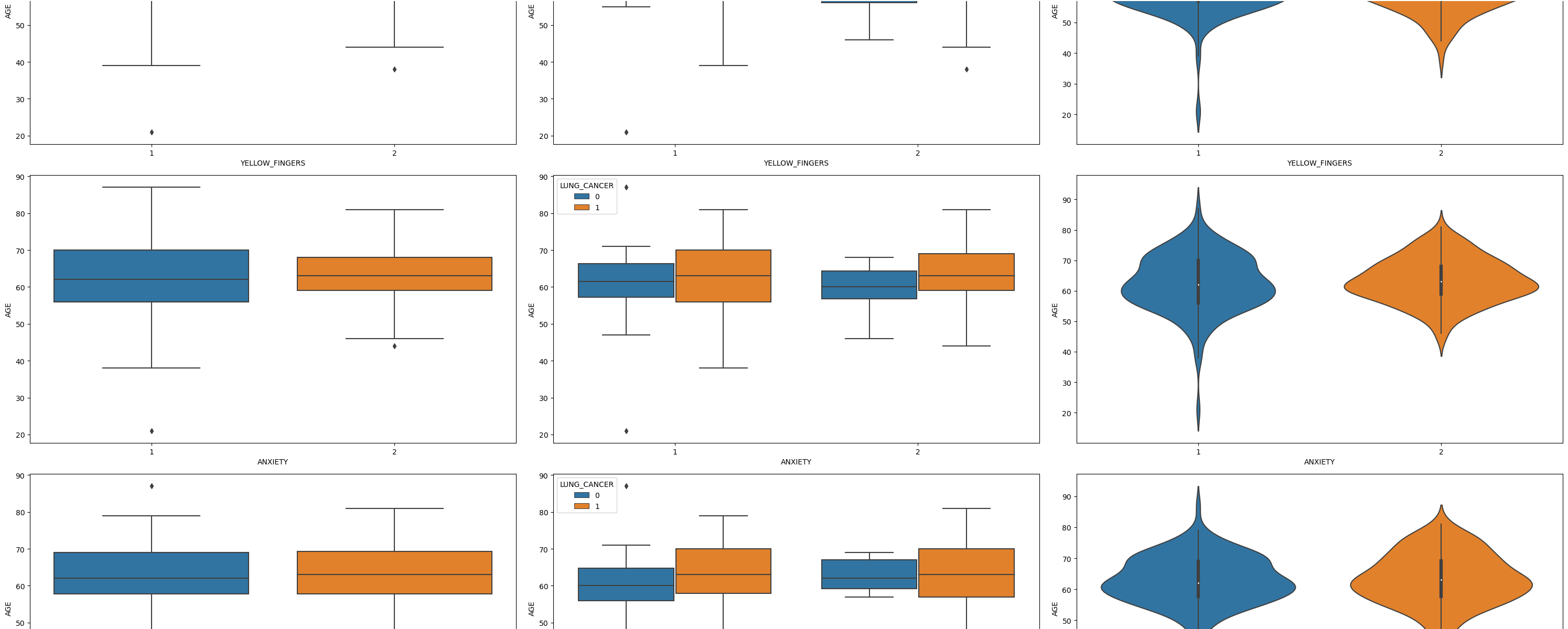
sns.boxplot(x**=**df[i],y**=**df['AGE'],ax**=**ax[index,0])

sns.boxplot(x**=**df[i],y**=**df['AGE'],ax**=**ax[index,1],hue**=**df['LUNG\_CANCER']) sns.violinplot(x**=**df[i],y**=**df['AGE'],ax**=**ax[index,2])

fig.tight\_layout()

fig.subplots\_adjust(top**=**0.95)

plt.suptitle("Visualizing AGE vs Categorical Columns",fontsize**=**50)



plt.figure(figsize=(15,15)) sns.heatmap(df.corr(),annot=True,linewidth=0.5,fmt='0.2f')

In [16]:

X**=**df.drop(['LUNG\_CANCER'],axis**=**1) y**=**df['LUNG\_CANCER']

# linearRegression

In [17]:

*# Import libraries* **import** pandas **as** pd **import** numpy **as** np

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.metrics **import** mean\_squared\_error, r2\_score

**from** sklearn.model\_selection **import** train\_test\_split

*# Load the dataset*

*# Split the dataset into features and target # Split the data into train and test sets*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.98, random\_state**=**42)

*# Create a linear regression model*

model **=** LinearRegression()

*# Fit the model on the train set*

model.fit(X\_train, y\_train)

*# Predict on the test set*

y\_pred **=** model.predict(X\_test) model.score(X\_train, y\_train)

Out[17]: 1.0

# LogisticRegression

In [18]:

*# import the necessary libraries*

**from** sklearn.datasets **import** load\_breast\_cancer

**from** sklearn.linear\_model **import** LogisticRegression **from** sklearn.model\_selection **import** train\_test\_split **from** sklearn.metrics **import** accuracy\_score

*# split the train and test dataset*

X\_train, X\_test,y\_train, y\_test **=** train\_test\_split(X, y,test\_size**=**0.20,random\_state**=**45)

*# LogisticRegression*

clf **=** LogisticRegression(random\_state**=**45) clf.fit(X\_train, y\_train)

*# Prediction*

y\_pred **=** clf.predict(X\_test)

acc **=** accuracy\_score(y\_test, y\_pred)

print("Logistic Regression model accuracy (in %):", acc**\***100)

Logistic Regression model accuracy (in %): 94.64285714285714

# RandomForestClassifier

In [23]:

**from** sklearn.ensemble **import** RandomForestClassifier rf **=** RandomForestClassifier(n\_estimators**=**100)

rf.fit(X\_train, y\_train)

ypred **=** rf.predict(X\_test)

print("Score the X-train with Y-train is : ", rf.score(X\_train,y\_train)) print("Score the X-test with Y-test is : ", rf.score(X\_test,y\_test))

print("Accuracy Score :",accuracy\_score(y\_test,ypred)**\***100)

Score the X-train with Y-train is : 1.0

Score the X-test with Y-test is : 0.9036144578313253 Accuracy Score : 90.36144578313254

# ANN

In [160]:

**from** sklearn.model\_selection **import** train\_test\_split

X\_train,X\_test,y\_train,y\_test**=**train\_test\_split(X,y,test\_size**=**0.50,random\_state**=**0) print(X\_train.shape)

print(X\_test.shape) print(y\_train.shape) print(y\_test.shape)

(138, 15)

(138, 15)

(138,)

(138,)

In [161]:

**from** sklearn.preprocessing **import** StandardScaler scaler**=**StandardScaler()

X\_train**=**scaler.fit\_transform(X\_train) X\_test**=**scaler.transform(X\_test)

In [162]:

**import** warnings

warnings.filterwarnings('ignore')

**from** sklearn.neural\_network **import** MLPRegressor

ann\_model**=**MLPRegressor(hidden\_layer\_sizes**=**(128,64,32),activation**=**'relu',solver**=**'lbfgs') ann\_model.fit(X\_train,y\_train)

Out[162]:

▾

MLPRegressor

MLPRegressor(hidden\_layer\_sizes=(128, 64, 32), solver='lbfgs')

In [163]:

y\_pred**=**ann\_model.predict(X\_test) y\_pred

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Out[163]: | array([-0.27046645, | 1.11420306, | 1.08119798, | 0.99855865, | 0.85089268, |
|  | 1.03020246, | 1.10064246, | 0.99658885, | 1.03396644, | -0.35122544, |
|  | 0.9355521 , | 1.0812097 , | 1.02217053, | 1.00651045, | 0.60816984, |
|  | 0.72060743, | 1.09897109, | 0.09602785, | 0.93502529, | -0.1795292 , |
|  | 1.00088269, | 1.12379087, | 1.03445763, | 1.11736311, | -0.02374211, |
|  | 1.12034558, | 1.05507871, | 0.99283378, | 0.98878657, | 0.99784774, |
|  | 0.93312314, | 0.97423255, | 1.08226353, | 1.00558863, | 1.00996518, |
|  | 0.74354272, | -0.04148026, | 1.11505494, | 0.76466434, | 0.99053938, |
|  | 1.04484394, | 0.95998932, | 0.74473401, | 0.53927351, | 0.96967064, |
|  | 1.12121031, | -0.12568266, | 0.88086758, | 0.58331154, | 0.99763951, |
|  | 0.9925576 , | -0.27234935, | 0.89640104, | 0.52923437, | 1.01365338, |
|  | 0.08252559, | 1.0415732 , | 1.12555965, | 0.00620682, | 0.34468117, |
|  | 1.012124 , | 0.98080112, | 1.00419668, | 1.01818306, | 0.99279325, |
|  | 0.99407564, | 0.99415995, | 0.95270087, | 0.52614255, | 0.97924742, |
|  | 0.0252738 , | 0.38481667, | 0.90413418, | 0.44521408, | 0.98998584, |
|  | 0.55718558, | 1.05580083, | 0.99225173, | 1.00142334, | 1.02567507, |
|  | -0.03524893, | 0.10791324, | 1.02547129, | 0.9518621 , | 0.88645931, |
|  | 0.01715826, | 0.32812365, | 1.02467144, | 1.06840844, | 0.9903359 , |
|  | 1.38624109, | 0.9553072 , | 1.20999602, | 1.00292861, | 0.89077127, |
|  | 1.24003028, | 0.99587639, | 0.99408538, | -0.02719458, | -0.03850283, |
|  | 1.01230475, | 1.00774796, | 0.00277839, | 1.05886746, | 1.07943519, |
|  | 1.07667188, | 0.90798554, | 0.74995355, | 1.03025105, | -0.12231914, |
|  | 0.99654681, | 0.9947503 , | 1.07403895, | 0.98672129, | 0.83938453, |
|  | 1.04902887, | 0.9823371 , | 0.50050329, | 0.55416735, | 1.04616607, |
|  | 0.01509583, | 0.97688152, | 0.68568851, | 0.9523753 , | 1.02306403, |
|  | 0.81576317, | 1.00948095, | 0.84185211, | -0.03077194, | 0.92916801, |
|  | 0.10791324, | 1.01271946, | -0.08388515, | 1.25519857, | 0.63832129, |
|  | 1.00640523, | 1.04678753, | 0.40416379]) |  |  |

In [164]:

**from** sklearn.metrics **import** r2\_score r2\_score**=**r2\_score(y\_test,y\_pred)

print('R-square score:',r2\_score)

R-square score: 0.09410233952613589

In [165]:

train\_accuracy**=**ann\_model.score(X\_train,y\_train) print('train\_accuracy:',train\_accuracy)

test\_accuracy**=**ann\_model.score(X\_test,y\_test) print('test\_accuracy:',test\_accuracy)

train\_accuracy: 0.9998859603548554

test\_accuracy: 0.09410233952613589

In [166]:

**import** math

**from** sklearn **import** metrics

print('Mean Absolute Error:',metrics.mean\_absolute\_error(y\_test,y\_pred)) print('Mean Squared Error:',metrics.mean\_squared\_error(y\_test,y\_pred))

print('Root Mean Squared Error:',math.sqrt(metrics.mean\_squared\_error(y\_test,y\_pred)))

Mean Absolute Error: 0.1785092389238528 Mean Squared Error: 0.11226204992219699

Root Mean Squared Error: 0.3350552938280441

In [169]:

**import** matplotlib.pyplot **as** plt

logistic\_regression\_accuracy **=**0.9464285714285714

random\_forest\_accuracy **=**0.9036144578313254 ANN\_model**=**0.9998859603548554

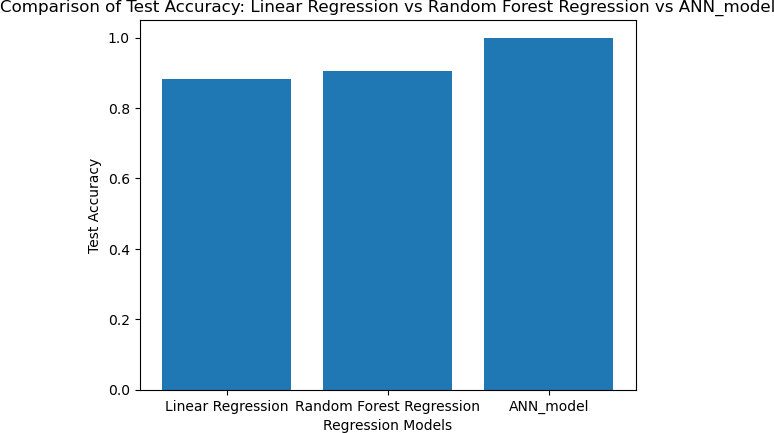
accuracy\_scores **=** [linear\_regression\_accuracy, random\_forest\_accuracy,ANN\_model] model\_names **=** ['Linear Regression', 'Random Forest Regression','ANN\_model']

plt.bar(model\_names, accuracy\_scores) plt.xlabel('Regression Models')

plt.ylabel('Test Accuracy')

plt.title('Comparison of Test Accuracy: Linear Regression vs Random Forest Regression vs ANN\_model')

plt.show()



In [ ]: