Heart Disease Assignment

In [34]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

heart_data = pd.read_csv('E:\!DataScience_DSPP_JNTU\Assignments\ML_AL\JNTUH_ML_DL_assignment_3\heart_disease_uci.csv')
```

In [2]:
```
heart_data
```

Out[2]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal	2
2	3	67	Male	Cleveland	asymptomatic	120.0	229.0	False	lv hypertrophy	129.0	True	2.6	flat	2.0	reversable defect	1
3	4	37	Male	Cleveland	non-anginal	130.0	250.0	False	normal	187.0	False	3.5	downsloping	0.0	normal	0
4	5	41	Female	Cleveland	atypical angina	130.0	204.0	False	lv hypertrophy	172.0	False	1.4	upsloping	0.0	normal	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
915	916	54	Female	VA Long Beach	asymptomatic	127.0	333.0	True	st-t abnormality	154.0	False	0.0	NaN	NaN	NaN	1
916	917	62	Male	VA Long Beach	typical angina	NaN	139.0	False	st-t abnormality	NaN	NaN	NaN	NaN	NaN	NaN	0
917	918	55	Male	VA Long Beach	asymptomatic	122.0	223.0	True	st-t abnormality	100.0	False	0.0	NaN	NaN	fixed defect	2
918	919	58	Male	VA Long Beach	asymptomatic	NaN	385.0	True	lv hypertrophy	NaN	NaN	NaN	NaN	NaN	NaN	0
919	920	62	Male	VA Long Beach	atypical angina	120.0	254.0	False	lv hypertrophy	93.0	True	0.0	NaN	NaN	NaN	1

920 rows × 16 columns

Data Preprocessing

In [3]:
```
heart_data.drop(['id','dataset'], axis=1, inplace=True)
heart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       920 non-null    int64  
 1   sex       920 non-null    object 
 2   cp        920 non-null    object 
 3   trestbps  861 non-null    float64
 4   chol      890 non-null    float64
 5   fbs       830 non-null    object 
 6   restecg   918 non-null    object 
 7   thalch    865 non-null    float64
 8   exang     865 non-null    object 
 9   oldpeak   858 non-null    float64
 10  slope     611 non-null    object 
 11  ca        309 non-null    float64
 12  thal      434 non-null    object 
 13  num       920 non-null    int64  
dtypes: float64(5), int64(2), object(7)
memory usage: 100.8+ KB
```

In [4]:
```
heart_data.describe()
```

Out[4]:

	age	trestbps	chol	thalch	oldpeak	ca	num
count	920.000000	861.000000	890.000000	865.000000	858.000000	309.000000	920.000000
mean	53.510870	132.132404	199.130337	137.545665	0.878788	0.676375	0.995652
std	9.424685	19.066070	110.780810	25.926276	1.091226	0.935653	1.142693
min	28.000000	0.000000	0.000000	60.000000	-2.600000	0.000000	0.000000
25%	47.000000	120.000000	175.000000	120.000000	0.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	140.000000	0.500000	0.000000	1.000000
75%	60.000000	140.000000	268.000000	157.000000	1.500000	1.000000	2.000000
max	77.000000	200.000000	603.000000	202.000000	6.200000	3.000000	4.000000

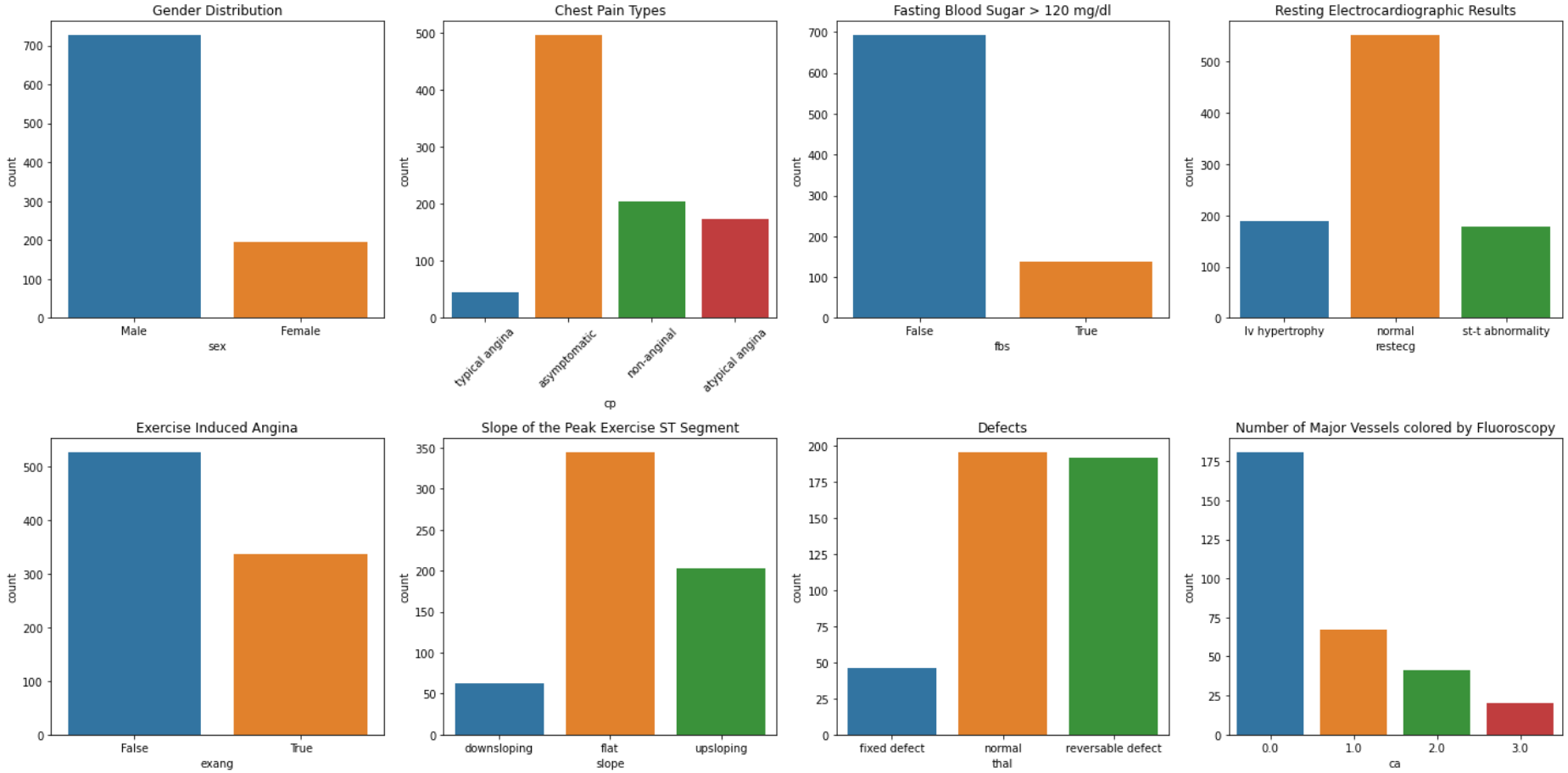
In [5]:
```
# Separate numeric and categoriecal variables for visualization purpose
CATEGORICAL_COLS = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'thal', 'ca']
NUMERICAL_COLS = ['age', 'trestbps', 'chol', 'thalch', 'oldpeak']
```

```
heart_cat = heart_data[CATEGORICAL_COLS]
heart_num = heart_data[NUMERICAL_COLS]

heart_cat.nunique()
```

Out[5]:
```
sex        2
cp         4
fbs        2
restecg    3
exang      2
slope      3
thal       3
ca         4
dtype: int64
```

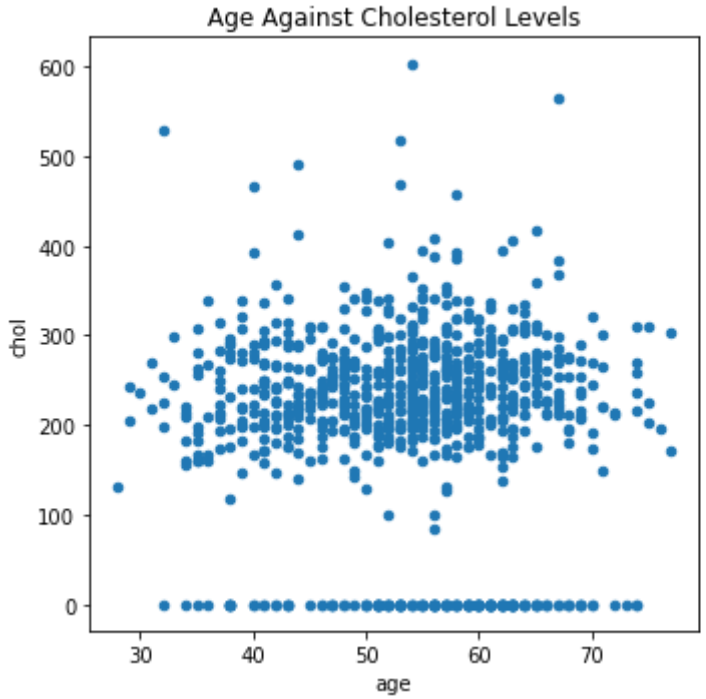
In [6]:
```
fig, axes = plt.subplots(2, 4, figsize=(20,10))

sns.countplot(x='sex', data=heart_cat, ax=axes[0,0])
axes[0,0].set_title('Gender Distribution')

sns.countplot(x='cp', data=heart_cat, ax=axes[0,1])
axes[0,1].tick_params(axis='x', rotation=45)
axes[0,1].set_title('Chest Pain Types')

sns.countplot(x='fbs', data=heart_cat, ax=axes[0,2])
axes[0,2].set_title('Fasting Blood Sugar > 120 mg/dl')

sns.countplot(x='restecg', data=heart_cat, ax=axes[0,3])
axes[0,3].set_title('Resting Electrocardiographic Results')

sns.countplot(x='exang', data=heart_cat, ax=axes[1,0])
axes[1,0].set_title('Exercise Induced Angina')

sns.countplot(x='slope', data=heart_cat, ax=axes[1,1])
axes[1,1].set_title('Slope of the Peak Exercise ST Segment')

sns.countplot(x='thal', data=heart_cat, ax=axes[1,2])
axes[1,2].set_title('Defects')

sns.countplot(x='ca', data=heart_cat, ax=axes[1,3])
axes[1,3].set_title('Number of Major Vessels colored by Fluoroscopy')
```

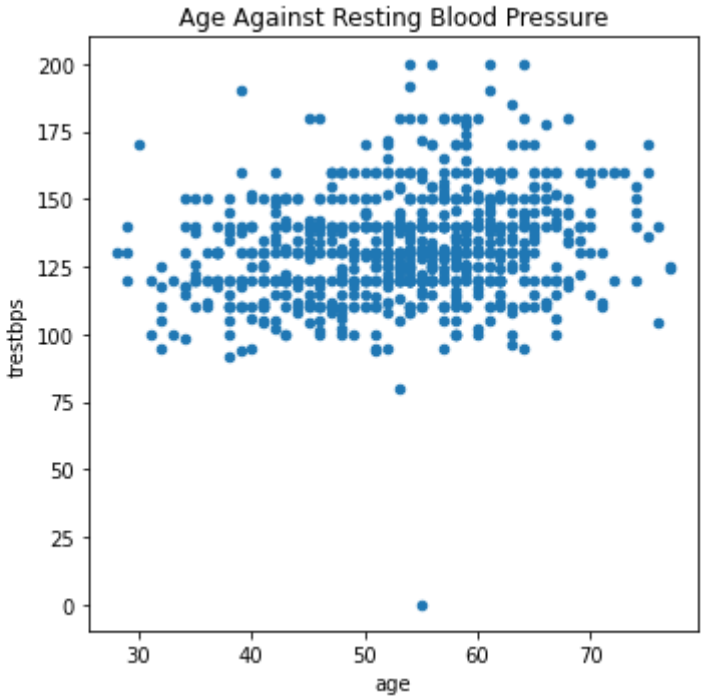
```
plt.tight_layout()
plt.show()
```

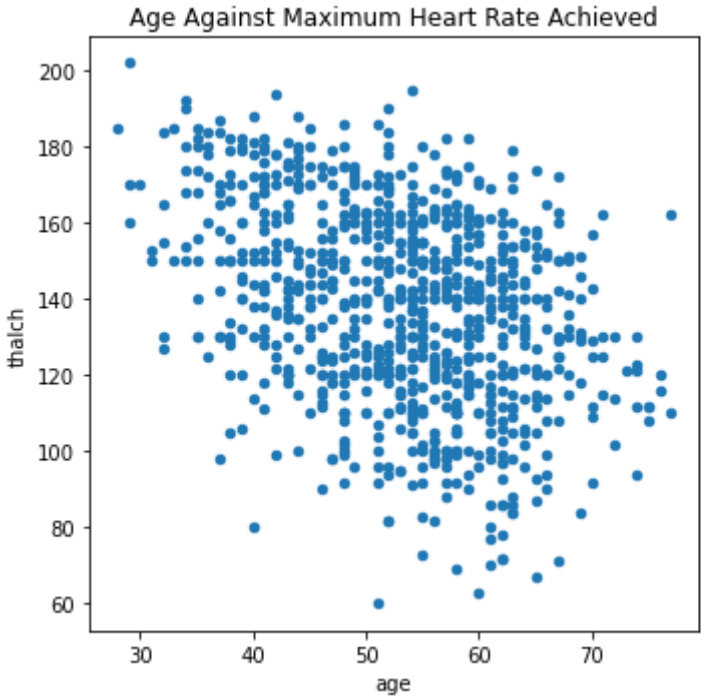


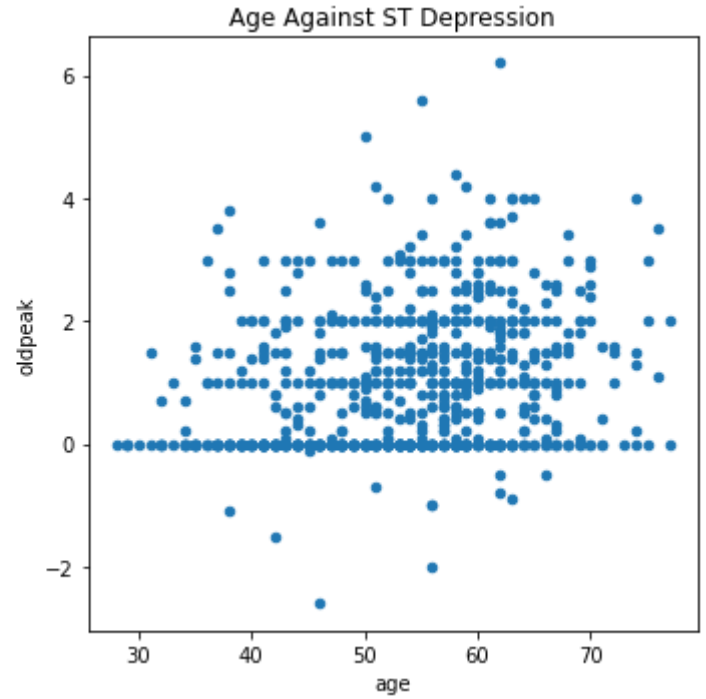
In [7]:
```
# Use scatterplots: To visualize key relationships in numerical data

fig, axes = plt.subplots(2, 2, figsize=(10,10))

heart_num.plot('age', 'chol', kind='scatter', ax=axes[0,0])
axes[0,0].set_title('Age Against Cholesterol Levels')

heart_num.plot('age', 'trestbps', kind='scatter', ax=axes[0,1])
axes[0,1].set_title('Age Against Resting Blood Pressure')

heart_num.plot('age', 'thalch', kind='scatter', ax=axes[1,0])
```

```
axes[1,0].set_title('Age Against Maximum Heart Rate Achieved')

heart_num.plot('age', 'oldpeak', kind='scatter', ax=axes[1,1])
axes[1,1].set_title('Age Against ST Depression')

plt.tight_layout()
plt.show()
```

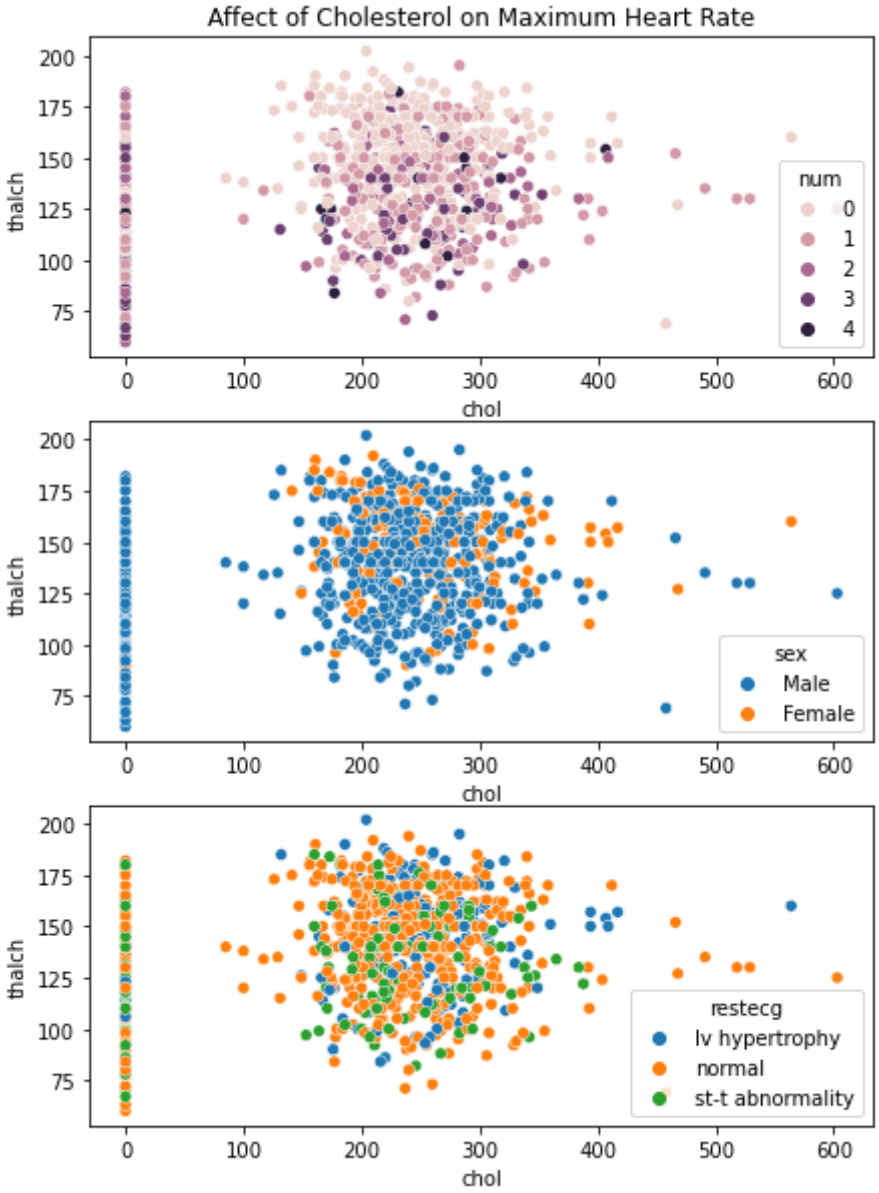

Age Against Cholesterol Levels
chol
age
0
100
200
300
400
500
600
30
40
50
60
70


Age Against Resting Blood Pressure
trestbps
age
0
25
50
75
100
125
150
175
200
30
40
50
60
70

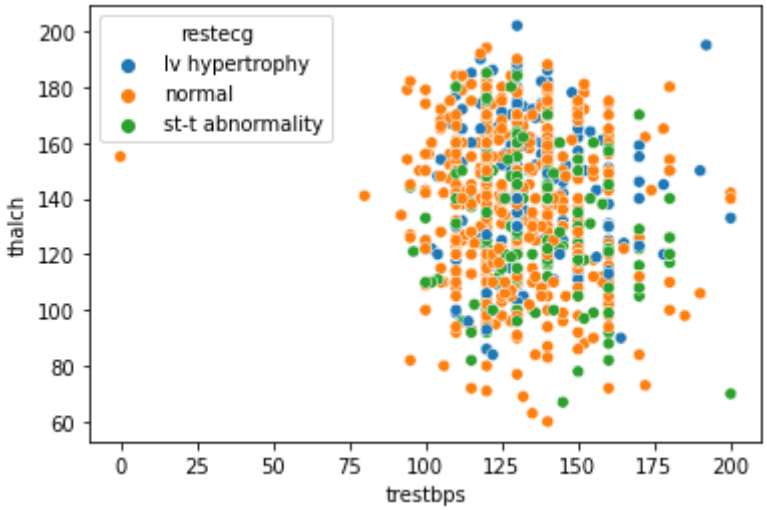

Age Against Maximum Heart Rate Achieved
thalch
age
60
80
100
120
140
160
180
200
30
40
50
60
70


Age Against ST Depression
oldpeak
age
−2
0
2
4
6
30
40
50
60
70

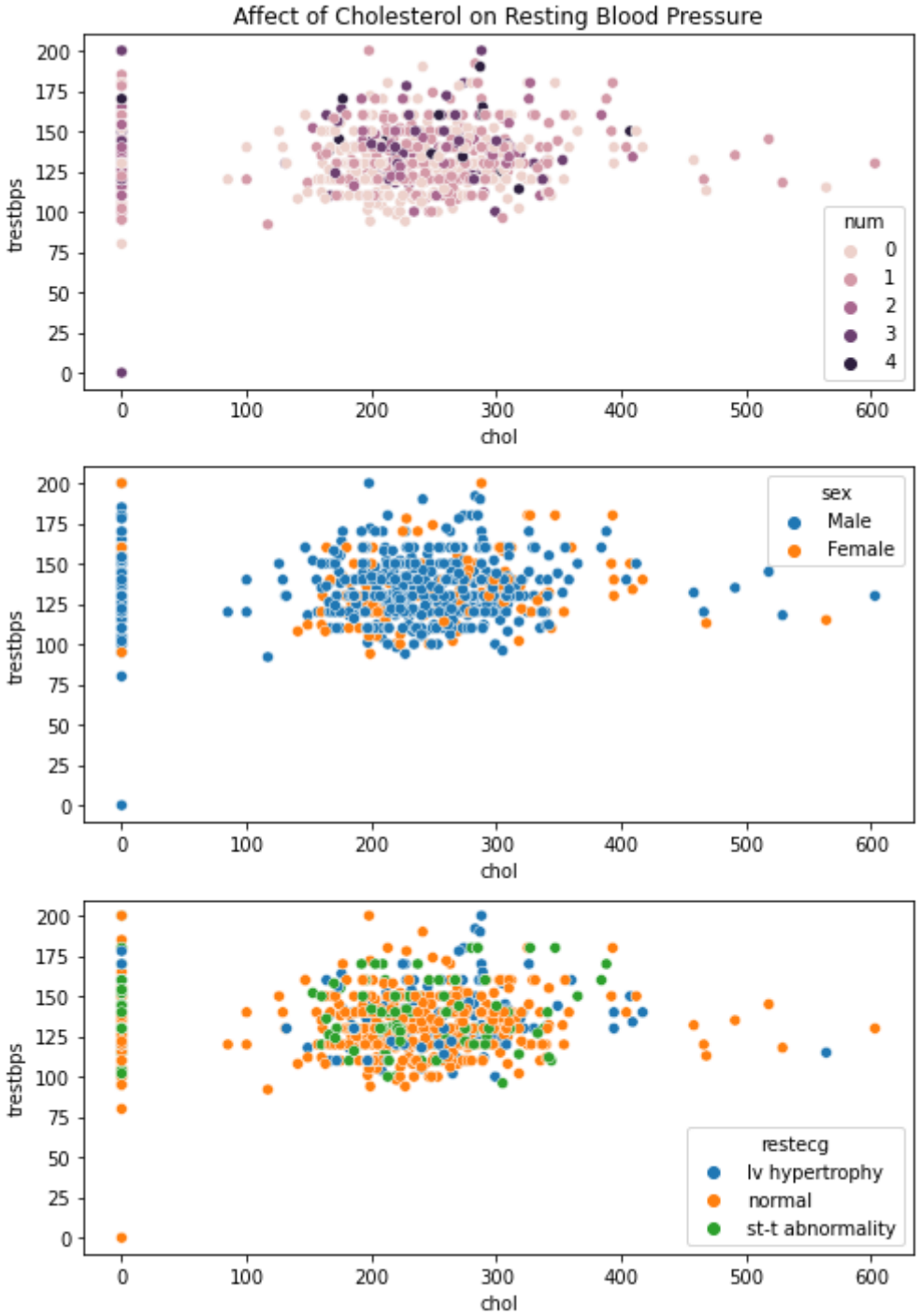
In [8]:
```
fig, axes = plt.subplots(3, figsize=(7,10))

sns.scatterplot(x='chol', y='thalch', hue='num', data=heart_data, ax=axes[0])
axes[0].set_title('Affect of Cholesterol on Maximum Heart Rate')

sns.scatterplot(x='chol', y='thalch', hue='sex', data=heart_data, ax=axes[1])

sns.scatterplot(x='chol', y='thalch', hue='restecg', data=heart_data, ax=axes[2])
plt.show()
```



In [9]:
```
sns.scatterplot(x='trestbps', y='thalch', hue='restecg', data=heart_data)
plt.show()
```



In [10]:
```
fig, axes = plt.subplots(3, figsize=(7,10))

axes[0].set_title('Affect of Cholesterol on Resting Blood Pressure')
sns.scatterplot(x='chol', y='trestbps', hue='num', data=heart_data, ax=axes[0])
sns.scatterplot(x='chol', y='trestbps', hue='sex', data=heart_data, ax=axes[1])
sns.scatterplot(x='chol', y='trestbps', hue='restecg', data=heart_data, ax=axes[2])

plt.tight_layout()
plt.show()
```


Affect of Cholesterol on Resting Blood Pressure
trestbps
chol
num
0
1
2
3
4
sex
Male
Female
restecg
lv hypertrophy
normal
st-t abnormality

In [11]:
```
heart_data.groupby('num').mean()
```

Out[11]:

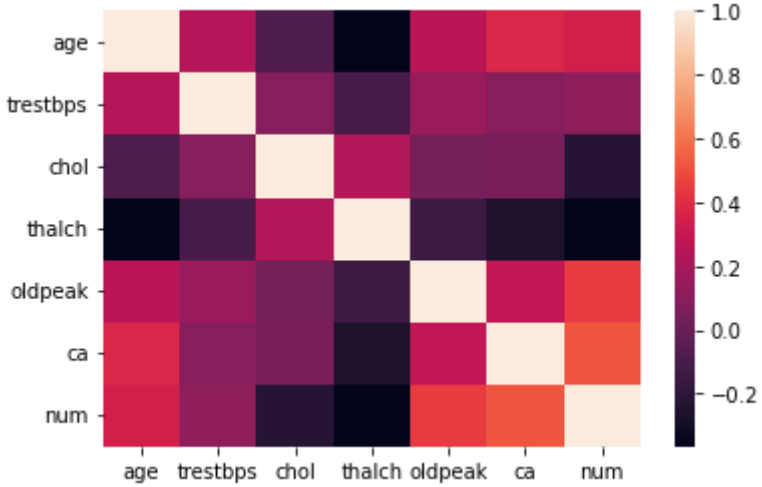
num	age	trestbps	chol	thalch	oldpeak	ca
0	50.547445	129.913043	227.905612	148.800512	0.418205	0.278788
1	53.528302	132.861111	195.255814	131.035714	1.001200	0.741379
2	57.577982	133.613861	143.859813	128.666667	1.353465	1.222222
3	59.214953	136.152174	159.716981	120.500000	1.581319	1.459459
4	59.214286	138.720000	192.148148	127.846154	2.307692	1.692308

In [12]:
```
print('Average Cholesterol Level Based on Target Variable and Chest Pain Type')
print(pd.crosstab(index=heart_data.num, columns=heart_data.cp, values=heart_data.chol, aggfunc=np.mean))
print('\n')

print('Average Cholesterol Level Based on Target Variable and Patient Gender')
print(pd.crosstab(index=heart_data.num, columns=heart_data.sex, values=heart_data.chol, aggfunc=np.mean))
print('\n')

print('Average Cholesterol Level Based on Target Variable and Cardiographic Results')
```

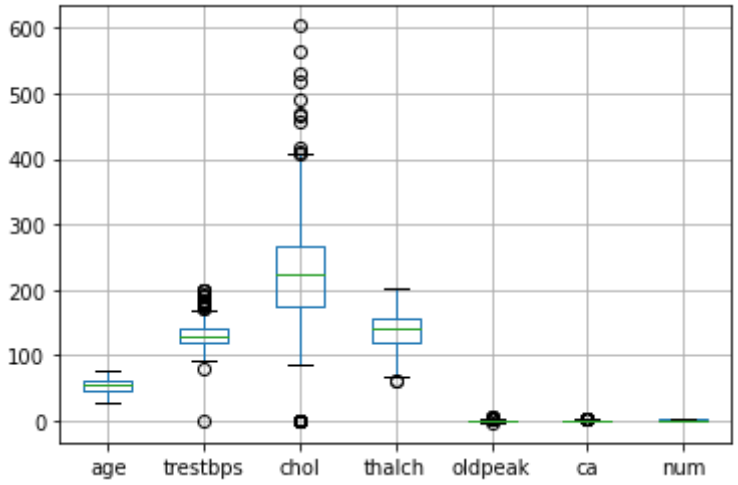
```
Average Cholesterol Level Based on Target Variable and Chest Pain Type
cp   asymptomatic  atypical angina  non-anginal  typical angina
num
0      227.843137       233.957143   222.209677      222.730769
1      193.273684       250.157895   170.756757      215.250000
2      152.321839       123.000000   118.642857       58.500000
3      157.219512       200.000000   152.888889      228.666667
4      196.478261              NaN   146.000000      231.000000


Average Cholesterol Level Based on Target Variable and Patient Gender
sex      Female        Male
num
0    248.102190  217.054902
1    221.366667  191.820175
2    216.400000  136.381443
3    216.250000  155.102041
4    316.000000  182.240000


Average Cholesterol Level Based on Target Variable and Cardiographic Results
```

In [13]:
```
# Display correlation matrix and heatmap
corr = heart_data.corr()
print(corr)

sns.heatmap(corr)
plt.show()
```

```
              age  trestbps      chol    thalch   oldpeak        ca       num
age       1.000000  0.244253 -0.086234 -0.365778  0.258243  0.370416  0.339596
trestbps  0.244253  1.000000  0.092853 -0.104899  0.161908  0.093705  0.122291
chol     -0.086234  0.092853  1.000000  0.236121  0.047734  0.051606 -0.231547
thalch   -0.365778 -0.104899  0.236121  1.000000 -0.151174 -0.264094 -0.366265
oldpeak   0.258243  0.161908  0.047734 -0.151174  1.000000  0.281817  0.443084
ca        0.370416  0.093705  0.051606 -0.264094  0.281817  1.000000  0.516216
num       0.339596  0.122291 -0.231547 -0.366265  0.443084  0.516216  1.000000
```



In [14]:
```
# Display boxplot to visualize outliers in the data

heart_data.boxplot()
plt.show()
```



In [15]:
```
heart_data.loc[heart_data['chol']==0,:]
```

Out[15]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
597	32	Male	typical angina	95.0	0.0	NaN	normal	127.0	False	0.7	upsloping	NaN	NaN	1
598	34	Male	asymptomatic	115.0	0.0	NaN	NaN	154.0	False	0.2	upsloping	NaN	NaN	1
599	35	Male	asymptomatic	NaN	0.0	NaN	normal	130.0	True	NaN	NaN	NaN	reversable defect	3
600	36	Male	asymptomatic	110.0	0.0	NaN	normal	125.0	True	1.0	flat	NaN	fixed defect	1
601	38	Female	asymptomatic	105.0	0.0	NaN	normal	166.0	False	2.8	upsloping	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
818	43	Male	asymptomatic	122.0	0.0	False	normal	120.0	False	0.5	upsloping	NaN	NaN	1
819	63	Male	non-anginal	130.0	0.0	True	st-t abnormality	160.0	False	3.0	flat	NaN	NaN	0
822	48	Male	non-anginal	102.0	0.0	NaN	st-t abnormality	110.0	True	1.0	downsloping	NaN	NaN	1
839	56	Male	asymptomatic	NaN	0.0	False	lv hypertrophy	NaN	NaN	NaN	NaN	NaN	NaN	1
840	62	Male	non-anginal	NaN	0.0	True	st-t abnormality	NaN	NaN	NaN	NaN	NaN	NaN	2

172 rows × 14 columns

Data Analysis

In [16]:
```
# Cholesterol Levels

median_chol = heart_data.loc[heart_data['chol']!=0, 'chol'].median()
heart_df = heart_data.fillna(value={'chol': median_chol})
heart_df.loc[heart_df['chol']==0, 'chol'] = median_chol
```

In [17]:
```
# Resting Blood Pressure

mean_bp = heart_df.loc[heart_df['trestbps']!=0,'trestbps'].mean()
heart_df = heart_df.fillna(value={'trestbps': mean_bp})
heart_df.loc[heart_df['trestbps']==0, 'trestbps'] = mean_bp
```

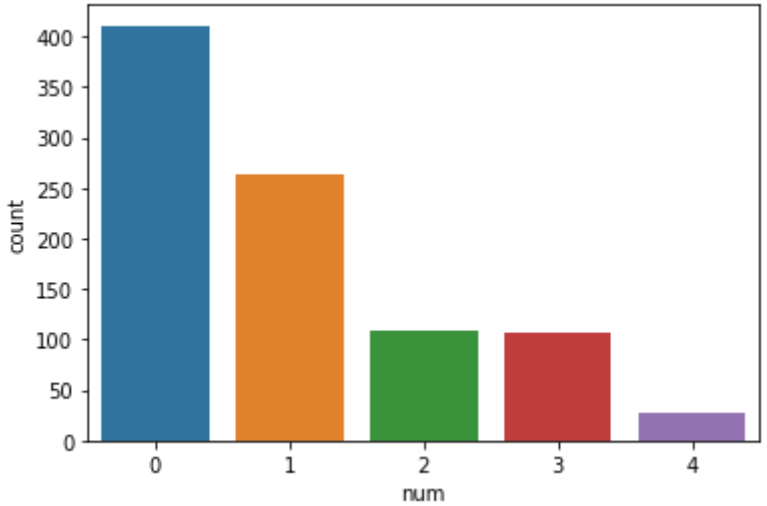
In [18]:
```
# Maximum Heart Rate
```

```
mean_hr = heart_df.loc[heart_df['thalch']!=0,'thalch'].mean()
heart_df = heart_df.fillna(value={'thalch': mean_hr})
heart_df.loc[heart_df['thalch']==0, 'thalch'] = mean_hr
```

In [19]:
```
# Old Peak

mean_peak = heart_df.oldpeak.mean()
heart_df = heart_df.fillna(value={'oldpeak': mean_peak})
heart_df.loc[heart_df['oldpeak']==0, 'oldpeak'] = mean_peak
```

In [20]:
```
# Drop the columns with missing values and reassign datatypes

heart_df.drop(labels=['ca','thal','slope'], axis=1, inplace=True)
heart_df = heart_df.astype({'sex':'category', 'cp':'category', 'fbs':'bool', 'restecg':'category', 'exang':'bool'})

# Drop remaining rows with missing values and display distribution for target variables

heart_df.dropna(inplace=True)
sns.countplot('num', data=heart_df)
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyw
ord will result in an error or misinterpretation.
  warnings.warn(
```



Preparing the Data - Model Training

In [21]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

In [22]:
```
# One hot encode the categorical variables and split the target and independent variables
heart_onehot = pd.get_dummies(heart_df, columns=['sex','cp', 'fbs', 'restecg', 'exang'])

X = heart_onehot.drop('num', axis=1)
y = heart_onehot.num

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

y_train.value_counts()
```

Out[22]:
```
0    340
1    204
3     86
2     82
4     22
Name: num, dtype: int64
```

In [23]:
```
heart_onehot.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 918 entries, 0 to 919
Data columns (total 19 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       918 non-null    int64
 1   trestbps                  918 non-null    float64
 2   chol                      918 non-null    float64
 3   thalch                    918 non-null    float64
 4   oldpeak                   918 non-null    float64
 5   num                       918 non-null    int64
 6   sex_Female                918 non-null    uint8
 7   sex_Male                  918 non-null    uint8
 8   cp_asymptomatic           918 non-null    uint8
 9   cp_atypical angina        918 non-null    uint8
 10  cp_non-anginal            918 non-null    uint8
 11  cp_typical angina         918 non-null    uint8
 12  fbs_False                 918 non-null    uint8
 13  fbs_True                  918 non-null    uint8
 14  restecg_lv hypertrophy    918 non-null    uint8
 15  restecg_normal            918 non-null    uint8
 16  restecg_st-t abnormality  918 non-null    uint8
 17  exang_False               918 non-null    uint8
 18  exang_True                918 non-null    uint8
dtypes: float64(4), int64(2), uint8(13)
memory usage: 61.9 KB
```

Decision Tree Classifier

In [24]:
```
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
```

In [25]:
```
weights = {0:1, 1:0.5, 2:0.5, 3:0.5, 4:0.5}

clf = DecisionTreeClassifier(criterion='entropy', max_depth=5)
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.62      0.92      0.74        71
           1       0.40      0.32      0.36        59
           2       0.40      0.15      0.22        27
           3       0.31      0.19      0.24        21
           4       0.22      0.33      0.27         6

    accuracy                           0.51       184
   macro avg       0.39      0.38      0.36       184
weighted avg       0.47      0.51      0.47       184
```

In [26]:
```
# Perform Decision Tree model with class weighting
weights = {0:1, 1:0.5, 2:0.5, 3:0.5, 4:0.5}

clf = DecisionTreeClassifier(criterion='entropy', max_depth=5, class_weight='balanced')
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.68      0.71        71
           1       0.45      0.37      0.41        59
           2       0.29      0.37      0.33        27
           3       0.30      0.33      0.32        21
           4       0.14      0.33      0.20         6

    accuracy                           0.48       184
   macro avg       0.39      0.42      0.39       184
weighted avg       0.52      0.48      0.50       184
```

Gradient Boosting

In [27]:
```
gradient_booster = GradientBoostingClassifier(learning_rate=0.02, max_depth=3, n_estimators=150)
gradient_booster.fit(X_train, y_train)
y_pred = gradient_booster.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.62      0.89      0.73        71
           1       0.44      0.46      0.45        59
           2       0.33      0.07      0.12        27
           3       0.45      0.24      0.31        21
           4       0.50      0.33      0.40         6

    accuracy                           0.54       184
   macro avg       0.47      0.40      0.40       184
weighted avg       0.50      0.54      0.49       184
```

Random Forest Classifier

In [28]:
```
clf = RandomForestClassifier(n_estimators=150)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.61      0.87      0.72        71
           1       0.43      0.36      0.39        59
           2       0.25      0.15      0.19        27
           3       0.29      0.24      0.26        21
           4       0.00      0.00      0.00         6

    accuracy                           0.50       184
   macro avg       0.32      0.32      0.31       184
weighted avg       0.44      0.50      0.46       184
```

In [29]:
```
clf = RandomForestClassifier(n_estimators=150, class_weight='balanced_subsample')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.63      0.89      0.74        71
           1       0.47      0.41      0.44        59
           2       0.36      0.15      0.21        27
           3       0.32      0.33      0.33        21
           4       0.00      0.00      0.00         6

    accuracy                           0.53       184
   macro avg       0.36      0.36      0.34       184
weighted avg       0.48      0.53      0.49       184
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-scor
e are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavi
or.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-scor
e are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavi
or.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-scor
e are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavi
or.
  _warn_prf(average, modifier, msg_start, len(result))
```

Decision Tree With SMOTE

In [47]:

```
!pip install -U imbalanced-learn
!conda install -c glemaitre imbalanced-learn
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting imbalanced-learn
  Downloading imbalanced_learn-0.10.1-py3-none-any.whl (226 kB)
Requirement already satisfied: numpy>=1.17.3 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn) (1.21.5)
Collecting joblib>=1.1.1
  Downloading joblib-1.2.0-py3-none-any.whl (297 kB)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn) (2.2.
0)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn) (1.0.2)
Requirement already satisfied: scipy>=1.3.2 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn) (1.7.3)
Installing collected packages: joblib, imbalanced-learn
Successfully installed imbalanced-learn-0.10.1 joblib-1.2.0
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... failed with initial frozen solve. Retrying with flexible solve.
Solving environment: ...working... failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): ...working... done
Solving environment: ...working... failed with initial frozen solve. Retrying with flexible solve.
Solving environment: ...working...
Found conflicts! Looking for incompatible packages.
This can take several minutes.  Press CTRL-C to abort.
failed
```

```
Building graph of deps:   0%|          | 0/4 [00:00<?, ?it/s]
Examining python=3.9:   0%|          | 0/4 [00:00<?, ?it/s]
Examining @/win-64::__archspec==1=x86_64:  25%|##5       | 1/4 [00:00<00:01,  1.92it/s]
Examining @/win-64::__archspec==1=x86_64:  50%|#####     | 2/4 [00:00<00:00,  3.85it/s]
Examining @/win-64::__win==0=0:  50%|#####     | 2/4 [00:00<00:00,  3.85it/s]
Examining imbalanced-learn:  75%|#######5  | 3/4 [00:00<00:00,  3.85it/s]


Determining conflicts:   0%|          | 0/4 [00:00<?, ?it/s]
Examining conflict for python imbalanced-learn:   0%|          | 0/4 [00:00<?, ?it/s]


UnsatisfiableError: The following specifications were found
to be incompatible with the existing python installation in your environment:

Specifications:

  - imbalanced-learn -> python[version='2.7.*|3.4.*|3.5.*|3.6.*']

Your python: python=3.9

If python is on the left-most side of the chain, that's the version you've asked for.
When python appears to the right, that indicates that the thing on the left is somehow
not available for the python version you are constrained to. Note that conda will not
change your python version to a different minor version unless you explicitly specify
that.
```

In [59]:
```
pip install imblearn==0.0
```

```
Note: you may need to restart the kernel to use updated packages.
Defaulting to user installation because normal site-packages is not writeable
Collecting imblearn==0.0
  Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Requirement already satisfied: imbalanced-learn in c:\users\raju\appdata\roaming\python\python39\site-packages (from imblearn==
0.0) (0.10.1)
Requirement already satisfied: numpy>=1.17.3 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblearn==0.
0) (1.21.5)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblea
rn==0.0) (2.2.0)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblear
n==0.0) (1.0.2)
Requirement already satisfied: scipy>=1.3.2 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblearn==0.0)
(1.7.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\raju\appdata\roaming\python\python39\site-packages (from imbalanced-lea
rn->imblearn==0.0) (1.2.0)
Installing collected packages: imblearn
Successfully installed imblearn-0.0
```

In [30]:
```
from imblearn.over_sampling import SMOTE
```

In [31]:
```
smt = SMOTE(sampling_strategy='not majority')

print('Before', y_train.value_counts())

X_train_SM, y_train_SM = smt.fit_resample(X_train, y_train)

val, counter = np.unique(y_train_SM, return_counts=True)
print('After', (val, counter))
```

```
Before 0    340
1    204
3     86
2     82
4     22
Name: num, dtype: int64
After (array([0, 1, 2, 3, 4], dtype=int64), array([340, 340, 340, 340, 340], dtype=int64))
```

In [32]:
```
clf = DecisionTreeClassifier(criterion='entropy', max_depth=6)
clf.fit(X_train_SM, y_train_SM)
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.66	0.82	0.73	71
1	0.51	0.36	0.42	59
2	0.33	0.11	0.17	27
3	0.24	0.29	0.26	21
4	0.14	0.50	0.22	6
accuracy			0.49	184
macro avg	0.38	0.41	0.36	184
weighted avg	0.50	0.49	0.48	184