In [85]:
```
import pandas as pd
import random
import seaborn as sns
sns.set(color_codes=True)
from sklearn.preprocessing import LabelEncoder
import numpy as np
from pandas_datareader import data
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.datasets import load_boston
from sklearn.model_selection import GridSearchCV
```

In [2]:
```
pip install pandas-datareader
```

```
Requirement already satisfied: pandas-datareader in c:\users\priya\anaconda3\li
b\site-packages (0.10.0)
Requirement already satisfied: requests>=2.19.0 in c:\users\priya\anaconda3\lib
\site-packages (from pandas-datareader) (2.25.1)
Requirement already satisfied: pandas>=0.23 in c:\users\priya\anaconda3\lib\sit
e-packages (from pandas-datareader) (1.2.4)
Requirement already satisfied: lxml in c:\users\priya\anaconda3\lib\site-packag
es (from pandas-datareader) (4.6.3)
Requirement already satisfied: numpy>=1.16.5 in c:\users\priya\anaconda3\lib\si
te-packages (from pandas>=0.23->pandas-datareader) (1.19.5)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\priya\anacond
a3\lib\site-packages (from pandas>=0.23->pandas-datareader) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\priya\anaconda3\lib\sit
e-packages (from pandas>=0.23->pandas-datareader) (2021.1)
Requirement already satisfied: six>=1.5 in c:\users\priya\anaconda3\lib\site-pa
ckages (from python-dateutil>=2.7.3->pandas>=0.23->pandas-datareader) (1.15.0)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\priya\anaconda3\li
b\site-packages (from requests>=2.19.0->pandas-datareader) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\priya\anaconda
3\lib\site-packages (from requests>=2.19.0->pandas-datareader) (1.26.4)
Requirement already satisfied: idna<3,>=2.5 in c:\users\priya\anaconda3\lib\sit
e-packages (from requests>=2.19.0->pandas-datareader) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\priya\anaconda3\l
ib\site-packages (from requests>=2.19.0->pandas-datareader) (2020.12.5)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
df = pd.read_csv('heart-statlog_csv.csv', na_values = ['?'])
```

In [4]:
```
df.head()
```

Out[4]:

	age	sex	chest	resting_blood_pressure	serum_cholestoral	fasting_blood_sugar	resting_electro
0	70	1	4	130	322	0	
1	67	0	3	115	564	0	
2	57	1	2	124	261	0	
3	64	1	4	128	263	0	
4	74	0	2	120	269	0	

In [5]:
```
df.shape
```

Out[5]:
```
(270, 14)
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
age                                0
sex                                0
chest                              0
resting_blood_pressure             0
serum_cholestoral                  0
fasting_blood_sugar                0
resting_electrocardiographic_results    0
maximum_heart_rate_achieved        0
exercise_induced_angina            0
oldpeak                            0
slope                              0
number_of_major_vessels            0
thal                               0
class                              0
dtype: int64
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   age                                   270 non-null    int64  
 1   sex                                   270 non-null    int64  
 2   chest                                 270 non-null    int64  
 3   resting_blood_pressure                270 non-null    int64  
 4   serum_cholestoral                     270 non-null    int64  
 5   fasting_blood_sugar                   270 non-null    int64  
 6   resting_electrocardiographic_results  270 non-null    int64  
 7   maximum_heart_rate_achieved           270 non-null    int64  
 8   exercise_induced_angina               270 non-null    int64  
 9   oldpeak                               270 non-null    float64
 10  slope                                 270 non-null    int64  
 11  number_of_major_vessels               270 non-null    int64  
 12  thal                                  270 non-null    int64  
 13  class                                 270 non-null    object 
dtypes: float64(1), int64(12), object(1)
memory usage: 29.7+ KB
```

In [8]:
```
le=LabelEncoder()
label=le.fit_transform(df["class"])
```

In [9]:
```
le.classes_
```

Out[9]:
```
array(['absent', 'present'], dtype=object)
```

In [10]:
```
Data=df.drop("class",axis='columns')
```

In [11]:
```
Data.head()
```

Out[11]:

	age	sex	chest	resting_blood_pressure	serum_cholestoral	fasting_blood_sugar	resting_electro
0	70	1	4	130	322	0	
1	67	0	3	115	564	0	
2	57	1	2	124	261	0	
3	64	1	4	128	263	0	
4	74	0	2	120	269	0	

In [12]:
```
Data["class"]=label
```

In [13]:
```
Data
```

Out[13]:

g_blood_sugar	resting_electrocardiographic_results	maximum_heart_rate_achieved	exercise_induced_a
0	2	109	
0	2	160	
0	0	141	
0	0	105	
0	2	121	
...	...	...	
1	0	162	
0	0	173	
0	2	153	
0	0	148	
0	2	108	

In [14]:
```
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

In [15]:
```
df['class'].unique()
```

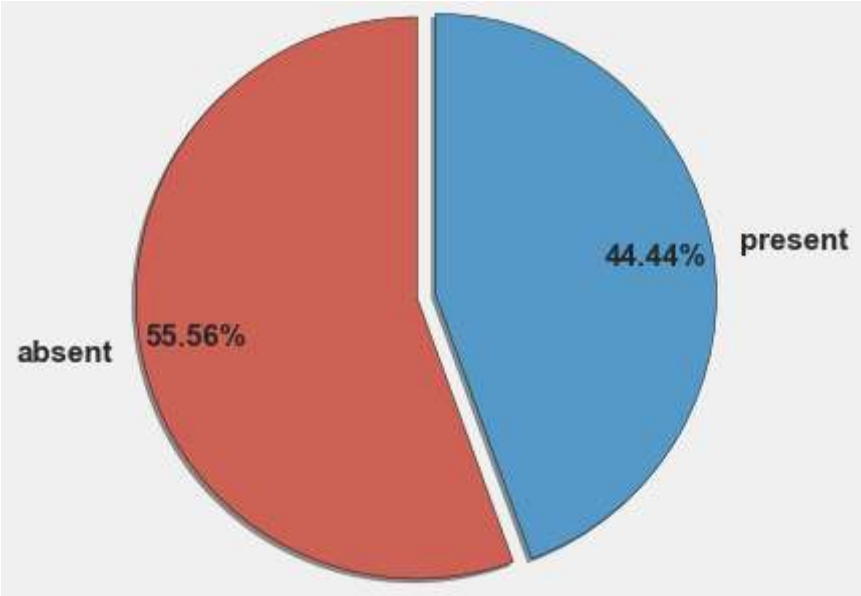
Out[15]:
```
array(['present', 'absent'], dtype=object)
```

In [16]:
```
work_count = df['class'].value_counts().tolist()
```

In [17]:
```
work_label = df['class'].value_counts().index
```

In [18]:
```
colors = ['#CD6155','#5499C7','#AF7AC5','#48C9B0','#52BE80','#F4D03F']
```

In [19]:
```
plt.pie(work_count, labels = work_label,
        autopct = '%1.2f%%',
        colors = colors[:3],
        wedgeprops = {'edgecolor':'k'},
        textprops = {'fontweight':'bold', 'size':15},
        shadow = True,
        explode = [0.1, 0],
        startangle = 90,
        pctdistance = 0.8,
        radius=1.5)
plt.show()
plt.savefig(r'E:\overleaf charts\barchart.png')
```

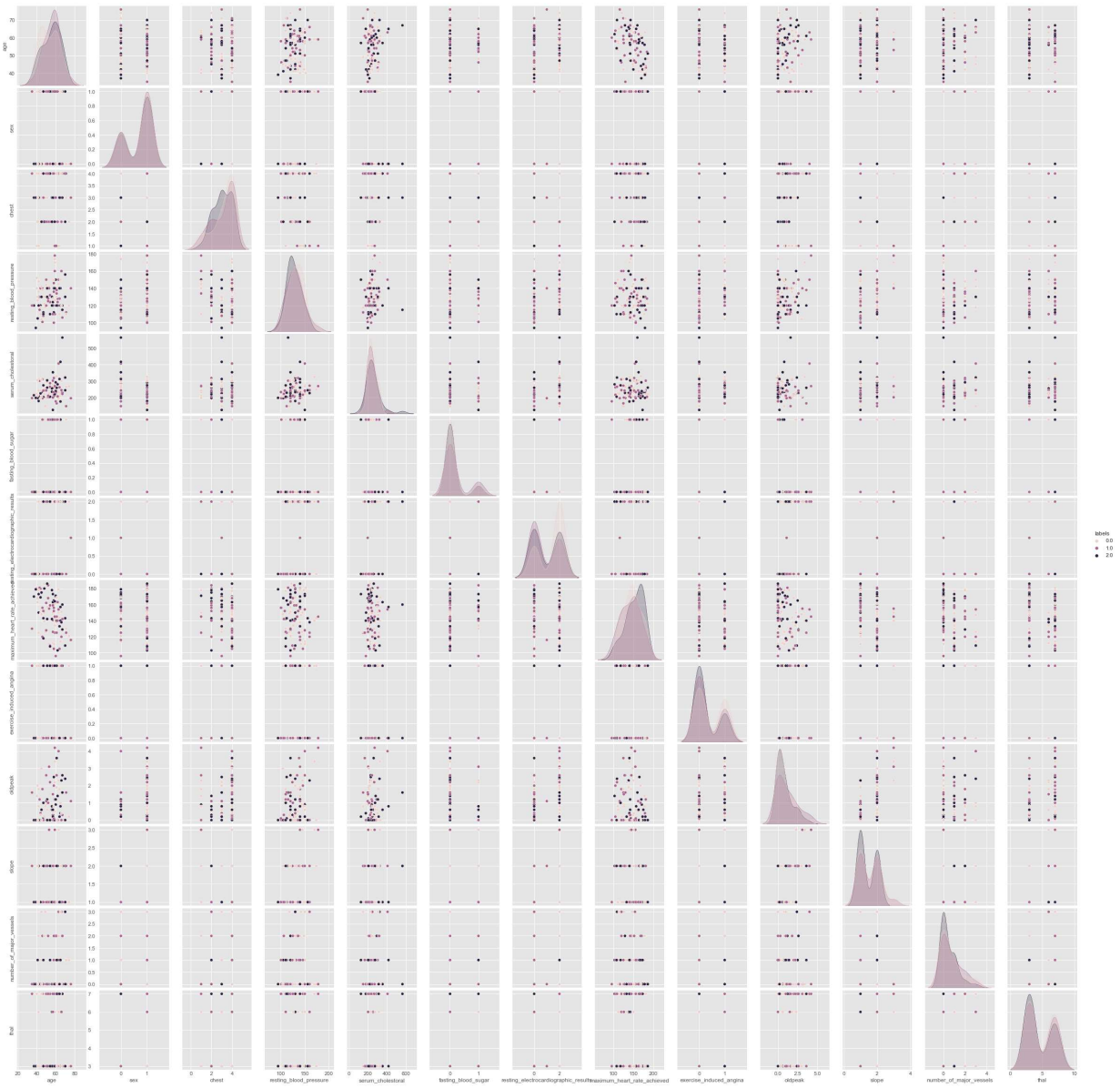



```
<Figure size 432x288 with 0 Axes>
```

PSO

In [68]:
```
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=100, n_features=14, n_classes=3,
                           n_informative=7, n_redundant=1, n_repeated=2,
                           random_state=1)
```

In [69]:
```
df['labels'] = pd.Series(y)

sns.pairplot(df, hue='labels');
```



In [22]:
```
pip install pyswarms
```

```
Requirement already satisfied: pyswarms in c:\users\priya\anaconda3\lib\site-pa
ckages (1.3.0)
Requirement already satisfied: tqdm in c:\users\priya\anaconda3\lib\site-packag
es (from pyswarms) (4.59.0)
Requirement already satisfied: numpy in c:\users\priya\anaconda3\lib\site-packa
ges (from pyswarms) (1.19.5)
Requirement already satisfied: future in c:\users\priya\anaconda3\lib\site-pack
ages (from pyswarms) (0.18.2)
Requirement already satisfied: attrs in c:\users\priya\anaconda3\lib\site-packa
ges (from pyswarms) (20.3.0)
Requirement already satisfied: matplotlib>=1.3.1 in c:\users\priya\anaconda3\li
b\site-packages (from pyswarms) (3.3.4)
Requirement already satisfied: pyyaml in c:\users\priya\anaconda3\lib\site-pack
ages (from pyswarms) (5.4.1)
Requirement already satisfied: scipy in c:\users\priya\anaconda3\lib\site-packa
ges (from pyswarms) (1.6.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\u
sers\priya\anaconda3\lib\site-packages (from matplotlib>=1.3.1->pyswarms) (2.4.
7)
Requirement already satisfied: pillow>=6.2.0 in c:\users\priya\anaconda3\lib\si
te-packages (from matplotlib>=1.3.1->pyswarms) (8.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\priya\anaconda3\lib\sit
e-packages (from matplotlib>=1.3.1->pyswarms) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\priya\anaconda3
\lib\site-packages (from matplotlib>=1.3.1->pyswarms) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\priya\anaconda3\li
b\site-packages (from matplotlib>=1.3.1->pyswarms) (1.3.1)
Requirement already satisfied: six in c:\users\priya\anaconda3\lib\site-package
s (from cycler>=0.10->matplotlib>=1.3.1->pyswarms) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [23]:
```
RANDOM_SEED =1
```

In [60]:
```
random.seed(RANDOM_SEED)
np.random.seed(RANDOM_SEED)

SWARM_SIZE=30
NUM_ITER=1000

NUM_PATIENTS=40
MAX_AGE=max(df['age'])

age=df['age']
```

In [25]:
```
def f_per_particle(m, max_age):
    """Computes for the objective function per particle

    Inputs
    ------
    m : numpy.ndarray
        Binary mask that can be obtained from BinaryPSO, will
        be used to mask features.
    alpha: float (default is 0.5)
        Constant weight for trading-off classifier performance
        and number of features

    Returns
    -------
    numpy.ndarray
        Computed objective function
    """
    total_features = 14
    # Get the subset of the features from the binary mask
    if np.count_nonzero(m) == 0:
        X_subset = X
    else:
        X_subset = X[:,m==1]
    # Perform classification and store performance in P
    classifier.fit(X_subset, y)
    P = (classifier.predict(X_subset) == y).mean()
    # Compute for the objective function
    j = (alpha * (1.0 - P)
        + (1.0 - alpha) * (1 - (X_subset.shape[1] / total_features)))

    return j

def f_per_particle(m, alpha):
    """Computes for the objective function per particle

    Inputs
    ------
    m : numpy.ndarray
        Binary mask that can be obtained from BinaryPSO, will
        be used to mask features.
    alpha: float (default is 0.5)
        Constant weight for trading-off classifier performance
        and number of features

    Returns
    -------
    numpy.ndarray
        Computed objective function
    """
    total_features = 13
    # Get the subset of the features from the binary mask
    if np.count_nonzero(m) == 0:
        X_subset = X
    else:
        X_subset = X[:,m==1]
    # Perform classification and store performance in P
```

```
    classifier.fit(X_subset, y)
    P = (classifier.predict(X_subset) == y).mean()
    # Compute for the objective function
    j = (alpha * (1.0 - P)
        + (1.0 - alpha) * (1 - (X_subset.shape[1] / total_features)))

    return j
```

In [26]:
```
import numpy as np
import seaborn as sns
import pandas as pd

# Import PySwarms
import pyswarms as ps
```

In [27]:
```
from tensorflow.keras.optimizers import SGD
```

In [28]:
```
def f(x,max_age=MAX_AGE):
    n_particles=x.shape[0]
    j=[f_per_particle(x[i],max_age) for i in range(n_particles)]
    return np.array(j)
```

In [29]:
```
def run_model(options):
    optimizer = ps.discrete.BinaryPSO(n_particles=SWARM_SIZE, dimensions=15, opti
    cost, pos = optimizer.optimize(f, iters=1000)
    return cost,pos,optimizer.cost_history
```

In [30]:
```
def plot_history(history):
    plt.style.use('ggplot')
    plt.rcParams['ytick.right']=True
    plt.rcParams['ytick.labelright']=True
    plt.rcParams['ytick.left']=False
    plt.rcParams['ytick.labelleft']=False
    plt.rcParams['font.family']='Arial'

    plt.ylim([min[history]-1000,max[history]+500])
    plt.title('Cost History')
    plt.plot(history)
```

In [31]:
```
checkpoint_state=np.random.get_state()
```

INITIAL MODEL

In [32]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.models import load_model

#empty model
classifier = Sequential()
```

In [33]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

In [34]:
```
inputs = keras.Input(shape=(784,), name="digits")
x = layers.Dense(64, activation="relu", name="dense_1")(inputs)
x = layers.Dense(64, activation="relu", name="dense_2")(x)
outputs = layers.Dense(10, activation="softmax", name="predictions")(x)

model = keras.Model(inputs=inputs, outputs=outputs)
```

In [35]:
```
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

# Preprocess the data (these are NumPy arrays)
x_train = x_train.reshape(60000, 784).astype("float32") / 255
x_test = x_test.reshape(10000, 784).astype("float32") / 255

y_train = y_train.astype("float32")
y_test = y_test.astype("float32")

# Reserve 10,000 samples for validation
x_val = x_train[-10000:]
y_val = y_train[-10000:]
x_train = x_train[:-10000]
y_train = y_train[:-10000]
```

In [36]:
```
model.compile(
    optimizer=keras.optimizers.RMSprop(),  # Optimizer
    # Loss function to minimize
    loss=keras.losses.SparseCategoricalCrossentropy(),
    # List of metrics to monitor
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)
```

In [37]:
```
print("Fit model on training data")
history = model.fit(
    x_train,
    y_train,
    batch_size=64,
    epochs=2,
    # We pass some validation for
    # monitoring validation loss and metrics
    # at the end of each epoch
    validation_data=(x_val, y_val),
)
```

```
Fit model on training data
Epoch 1/2
782/782 [==============================] - 3s 3ms/step - loss: 0.3420 - sparse_
categorical_accuracy: 0.9025 - val_loss: 0.1939 - val_sparse_categorical_accura
cy: 0.9446
Epoch 2/2
782/782 [==============================] - 1s 2ms/step - loss: 0.1616 - sparse_
categorical_accuracy: 0.9521 - val_loss: 0.1493 - val_sparse_categorical_accura
cy: 0.9545
```

In [38]:
```
history.history
```

Out[38]:
```
{'loss': [0.3419865667819977, 0.1616363227367401],
 'sparse_categorical_accuracy': [0.9025200009346008, 0.9521399736404419],
 'val_loss': [0.1939145028591156, 0.14933522045612335],
 'val_sparse_categorical_accuracy': [0.944599986076355, 0.9545000195503235]}
```

In [39]:
```
# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=128)
print("test loss, test acc:", results)

# Generate predictions (probabilities -- the output of the last layer)
# on new data using `predict`
print("Generate predictions for 3 samples")
predictions = model.predict(x_test[:3])
print("predictions shape:", predictions.shape)
```

```
Evaluate on test data
79/79 [==============================] - 0s 1ms/step - loss: 0.1532 - sparse_ca
tegorical_accuracy: 0.9512
test loss, test acc: [0.15317949652671814, 0.951200008392334]
Generate predictions for 3 samples
predictions shape: (3, 10)
```

In [40]:
```
model.compile(
    optimizer=keras.optimizers.RMSprop(learning_rate=1e-3),
    loss=keras.losses.SparseCategoricalCrossentropy(),
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)
```

In [41]:
```
model.compile(
    optimizer="rmsprop",
    loss="sparse_categorical_crossentropy",
    metrics=["sparse_categorical_accuracy"],
)
```

In [ ]:
```
```

In [42]:
```
from sklearn import linear_model

# Create an instance of the classifier
classifier = linear_model.LogisticRegression()
```

In [43]:
```

np.random.set_state(checkpoint_state)
options = {'c1': 0.5, 'c2': 0.5, 'w':0.9, 'k': SWARM_SIZE, 'p':2}
```

final model

In [45]:
```

# Initialize swarm, arbitrary
options = {'c1': 0.5, 'c2': 0.5, 'w':0.9, 'k': SWARM_SIZE, 'p':2}

# Call instance of PSO
dimensions = 15 # dimensions should be the number of features
optimizer = ps.discrete.BinaryPSO(n_particles=30, dimensions=dimensions, options=

# Perform optimization
cost, pos = optimizer.optimize(f, iters=1000, verbose=2)
```

```
2021-09-08 19:28:49,964 - pyswarms.discrete.binary - INFO - Optimize for 1000 i
ters with {'c1': 0.5, 'c2': 0.5, 'w': 0.9, 'k': 30, 'p': 2}
pyswarms.discrete.binary: 100%|
|1000/1000, best_cost=-36.6
2021-09-08 19:31:46,865 - pyswarms.discrete.binary - INFO - Optimization finish
ed | best cost: -36.58769230769231, best pos: [0 0 0 0 0 1 0 0 0 0 0 0 0 1 0]
```

In [54]:
```

# Create two instances of LogisticRegression
classfier = linear_model.LogisticRegression()

# Get the selected features from the final positions
X_selected_features = X[:,pos==1]  # subset

# Perform classification and store performance in P
classifier.fit(X_selected_features, y)
```

Out[54]:
```
LogisticRegression()
```

In [72]:
```
selected_features=df.values[np.where(pos==1)].tolist()
```

In [73]:
```
df1 = pd.DataFrame(selected_features)
df1['labels'] = pd.Series(y)
sns.pairplot(df1,hue='labels');
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWa
rning: Dataset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWa
rning: Dataset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWa
rning: Dataset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWa
rning: Dataset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
```

In [119]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0
```

In [120]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [121]:
```
max(df['age'])
```

Out[121]:
```
77
```

In [ ]:

In [ ]:

In [75]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0
```

In [81]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [87]:
```
RandomForestClassifier,VotingClassifier,AdaBoostClassifier,GradientBoostingClassi
```

DECISION TREE

In [193]:
```
from sklearn.tree import DecisionTreeClassifier
decc = DecisionTreeClassifier()
decc.fit(X_train,y_train)
y_pred_decc = decc.predict(X_test)
```

In [194]:
```
from random import random
import pandas as pd
from matplotlib import pyplot as plt
```

In [195]:
```
from sklearn.metrics import confusion_matrix
```

In [196]:
```
CM=confusion_matrix(y_test,y_pred_decc)
sns.heatmap(CM, annot=True)

TN = CM[0][0]
FN = CM[1][0]
TP = CM[1][1]
FP = CM[0][1]
specificity = TN/(TN+FP)
print(specificity)
```

```
0.4
```

In [197]:
```
from sklearn.metrics import accuracy_score
```

In [198]:
```
acc= accuracy_score(y_test,y_pred_decc)
print(acc)
```

```
0.6
```

BAGGED TREE

In [168]:
```
from sklearn.ensemble import BaggingClassifier
```

In [169]:
```
bt = BaggingClassifier(base_estimator=decc,n_estimators=270)
```

In [170]:
```
bt.fit(X_train,y_train)
```

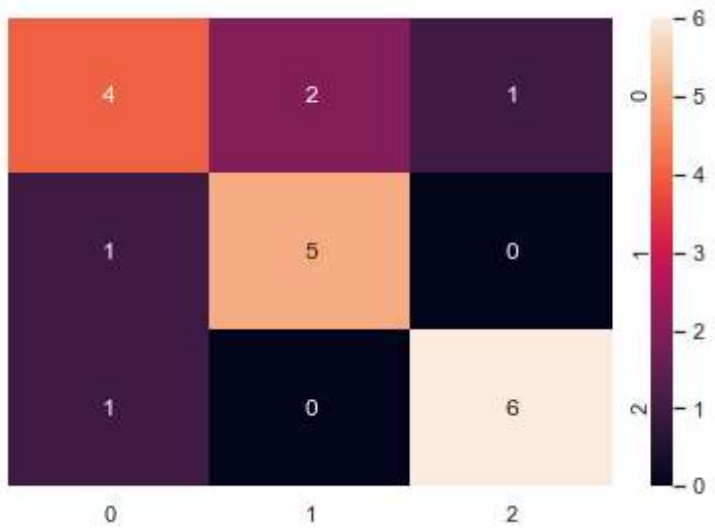
Out[170]:
```
BaggingClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=270)
```

In [171]:
```
y_pred_ada = bt.predict(X_test)
```

In [172]:
```
from sklearn.metrics import log_loss,roc_auc_score,precision_score,f1_score,recal
from sklearn.metrics import classification_report, confusion_matrix,accuracy_scor
from sklearn import metrics
```

In [173]:
```
CM=confusion_matrix(y_test,y_pred_ada)
sns.heatmap(CM, annot=True)

TN = CM[0][0]
FN = CM[1][0]
TP = CM[1][1]
FP = CM[0][1]
specificity = TN/(TN+FP)
print(specificity)
```

```
0.6666666666666666
```



In [174]:
```
acc= accuracy_score(y_test,y_pred_ada)
print(acc)
```

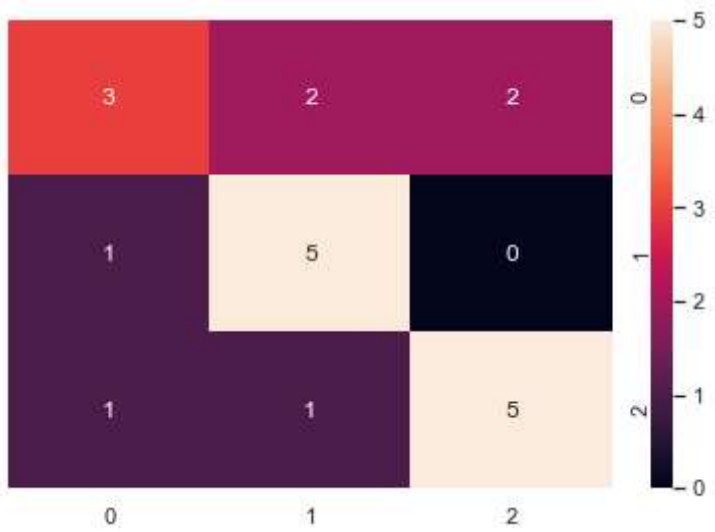
```
0.75
```

RANDOM FOREST

In [230]:
```
rf_ent = RandomForestClassifier(criterion='entropy',n_estimators=100)
rf_ent.fit(X_train, y_train)
y_pred_rfe = rf_ent.predict(X_test)
```

In [231]:
```
CM=confusion_matrix(y_test,y_pred_rfe)
sns.heatmap(CM, annot=True)

TN = CM[0][0]
FN = CM[1][0]
TP = CM[1][1]
FP = CM[0][1]
specificity = TN/(TN+FP)
print(specificity)
```

```
0.6
```



In [232]:
```
acc= accuracy_score(y_test,y_pred_rfe)
print(acc)
```

```
0.65
```

KNN

In [243]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X, y)
```

Out[243]:
```
KNeighborsClassifier()
```

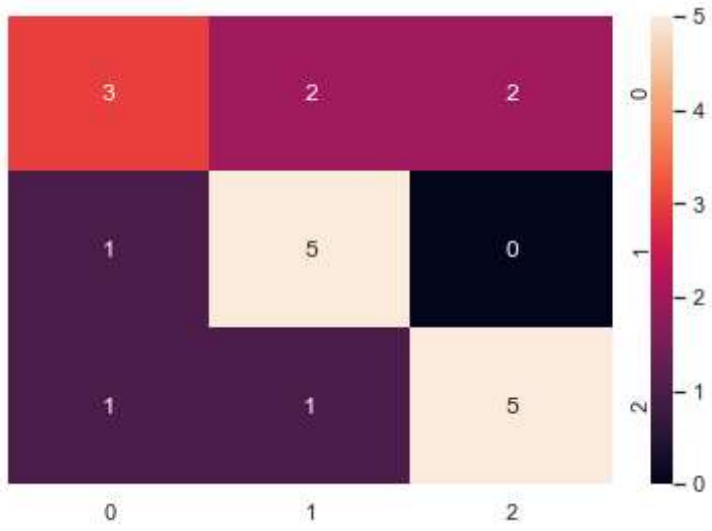
In [244]:
```
y_pred = classifier.predict(X_test)
```

In [245]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[4 1 2]
 [1 5 0]
 [0 1 6]]
              precision    recall  f1-score   support

           0       0.80      0.57      0.67         7
           1       0.71      0.83      0.77         6
           2       0.75      0.86      0.80         7

    accuracy                           0.75        20
   macro avg       0.75      0.75      0.75        20
weighted avg       0.76      0.75      0.74        20
```

In [246]:
```
CM=confusion_matrix(y_test,y_pred_rfe)
sns.heatmap(CM, annot=True)

TN = CM[0][0]
FN = CM[1][0]
TP = CM[1][1]
FP = CM[0][1]
specificity = TN/(TN+FP)
print(specificity)
```

```
0.6
```



In [247]:
```
acc= accuracy_score(y_test,y_pred)
print(acc)
```

```
0.75
```

ADABOOST

In [254]:
```
adaboost = AdaBoostClassifier()
adaboost.fit(X_train,y_train)
y_pred_adaboost = adaboost.predict(X_test)
```

In [255]:
```
acc= accuracy_score(y_test,y_pred_adaboost)
print(acc)
```

```
0.55
```

In [ ]:

In [ ]:

In [20]:
```
df.value_counts()[:20]
```

Out[20]:
```
age  sex  chest  resting_blood_pressure  serum_cholestoral  fasting_blood_sug
ar  resting_electrocardiographic_results  maximum_heart_rate_achieved  exerci
se_induced_angina  oldpeak  slope  number_of_major_vessels  thal  class
29   1    2      130                     204                0
2                                       202                          0
0.0      1      0                       3      absent      1
59   1    4      170                     326                0
2                                       140                          1
3.4      3      0                       7      present     1
58   1    4      128                     259                0
2                                       130                          1
3.0      2      2                       7      present     1
                 146                     218                0
0                                       105                          0
2.0      2      1                       7      present     1
                 150                     270                0
2                                       111                          1
0.8      1      0                       7      present     1
59   0    4      174                     249                0
0                                       143                          1
0.0      2      0                       3      present     1
     1    1      160                     273                0
2                                       125                          0
0.0      1      0                       3      present     1
                 170                     288                0
2                                       159                          0
0.2      2      0                       7      present     1
                 178                     270                0
2                                       145                          0
4.2      3      0                       7      absent      1
          2      140                     221                0
0                                       164                          1
0.0      1      0                       3      absent      1
          3      126                     218                1
0                                       134                          0
2.2      2      1                       6      present     1
                 150                     212                1
0                                       157                          0
1.6      1      0                       3      absent      1
          4      110                     239                0
2                                       142                          1
1.2      2      1                       7      present     1
                 135                     234                0
0                                       161                          0
0.5      2      0                       7      absent      1
                 138                     271                0
2                                       182                          0
0.0      1      0                       3      absent      1
                 140                     177                0
0                                       162                          1
0.0      1      1                       7      present     1
60   0    1      150                     240                0
0                                       171                          0
0.9      1      0                       3      absent      1
```

```
58   1    4      125                      300                 0
2                                    171                          0
0.0      1      2                        7     present     1
60   0    3      102                      318                 0
0                                    160                          0
0.0      1      1                        3     absent      1
                 120                      178                 1
0                                    96                           0
0.0      1      0                        3     absent      1
dtype: int64
```

In [22]:
```
X = df.drop(['class'],axis=1)
y = df['class']
```

In [122]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0
```

In [123]:
```
print('Distribution of traget variable in training set')
print(y_train.value_counts())

print('Distribution of traget variable in test set')
print(y_test.value_counts())
```

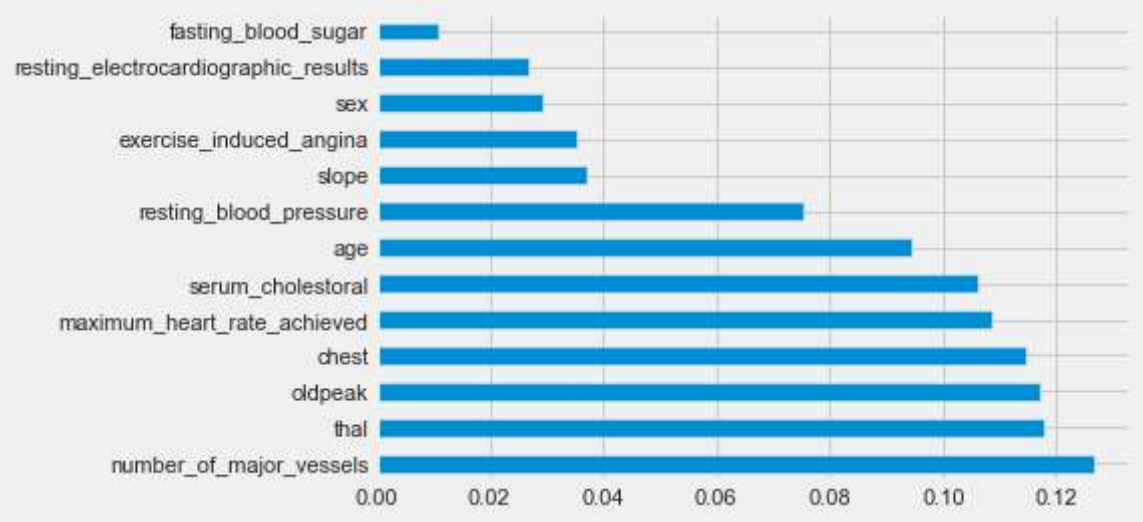
```
Distribution of traget variable in training set

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-123-007fda9da671> in <module>
      1 print('Distribution of traget variable in training set')
----> 2 print(y_train.value_counts())
      3
      4 print('Distribution of traget variable in test set')
      5 print(y_test.value_counts())

AttributeError: 'numpy.ndarray' object has no attribute 'value_counts'
```

In [124]:
```
print('------------Training Set-----------------')
print(X_train.shape)
print(y_train.shape)

print('------------Test Set-----------------')
print(X_test.shape)
print(y_test.shape)
```

```
------------Training Set-----------------
(80, 15)
(80,)
------------Test Set-----------------
(20, 15)
(20,)
```

In [43]:
```
feat_importances = pd.Series(rf_ent.feature_importances_, index=X_train.columns)
feat_importances.nlargest(20).plot(kind='barh')
```

Out[43]: <AxesSubplot:>



In [49]:
```
X = df.iloc[:,0:12]
y = df.iloc[:,12]
```

In [51]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_st
```

In [52]:
```
rt = DecisionTreeRegressor(criterion = 'mse', max_depth=5)
```

In [53]:
```
rt.fit(X_train,y_train)
```

Out[53]: DecisionTreeRegressor(max_depth=5)

In [54]:
```
y_pred = rt.predict(X_test)
r2_score(y_test,y_pred)
```

Out[54]: 0.10740364193337304

In [55]:
```
param_grid = {
    'max_depth':[2,4,8,10,None],
    'criterion':['mse','mae'],
    'max_features':[0.25,0.5,1.0],
    'min_samples_split':[0.25,0.5,1.0]
}
```

In [56]:
```
reg = GridSearchCV(DecisionTreeRegressor(),param_grid=param_grid)
```

In [57]:
```
reg.fit(X_train,y_train)
```

Out[57]:
```
GridSearchCV(estimator=DecisionTreeRegressor(),
             param_grid={'criterion': ['mse', 'mae'],
                         'max_depth': [2, 4, 8, 10, None],
                         'max_features': [0.25, 0.5, 1.0],
                         'min_samples_split': [0.25, 0.5, 1.0]})
```

In [36]:
```
reg.best_score_
```

Out[36]:
```
0.16345729494299172
```

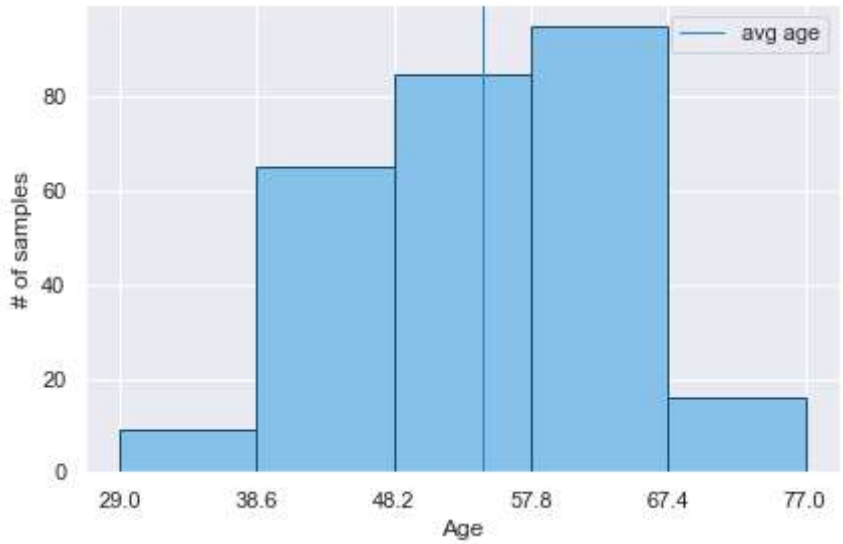
In [38]:
```
reg.best_params_
```

Out[38]:
```
{'criterion': 'mse',
 'max_depth': 2,
 'max_features': 0.25,
 'min_samples_split': 1.0}
```

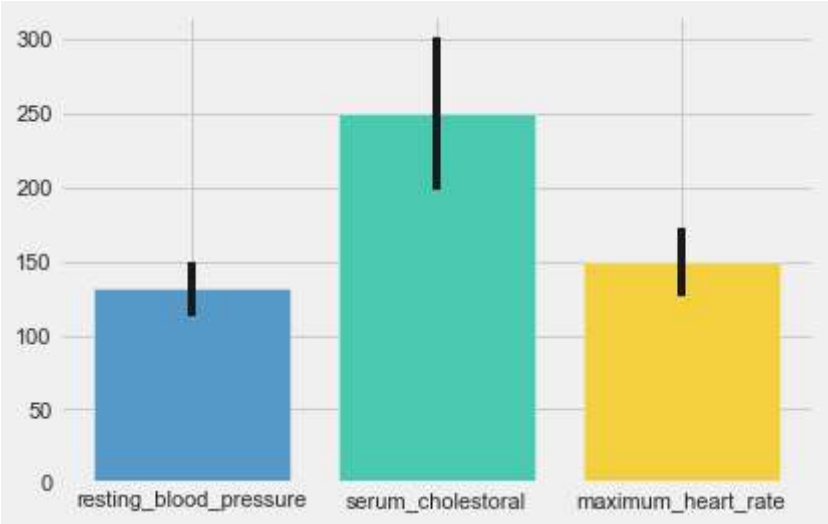
In [25]:
```
count, bin_edges = np.histogram(df['age'], bins = 5)
```

In [ ]:
```

```

In [26]:
```
age_mean = df['age'].mean()
plt.hist(df['age'], color = '#85C1E9', ec = '#1B4F72', bins = 5)
plt.xlabel('Age')
plt.ylabel('# of samples')
plt.xticks(bin_edges)
plt.axvline(age_mean, label = 'avg age',
            color = '#2E86C1', linewidth = 1)
plt.legend()
plt.show()
```



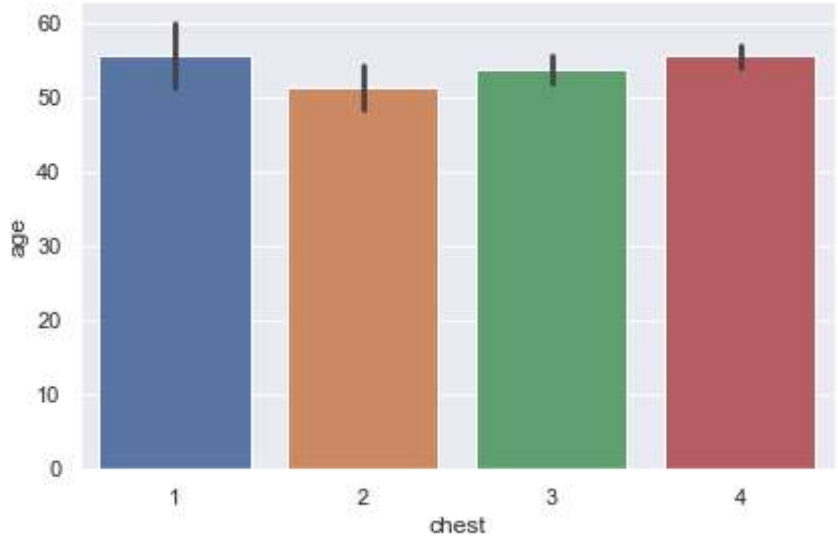
In [20]:
```
plt.bar('resting_blood_pressure', df['resting_blood_pressure'].mean(), yerr = df[
       color = colors[1])
plt.bar('serum_cholestoral', df['serum_cholestoral'].mean(), yerr = df['serum_cho
       color = colors[3])
plt.bar('maximum_heart_rate', df['maximum_heart_rate_achieved'].mean(), yerr = df
       color = colors[5])
plt.show()
```



In [27]:
```
sns.barplot(df['chest'], df['age'])
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

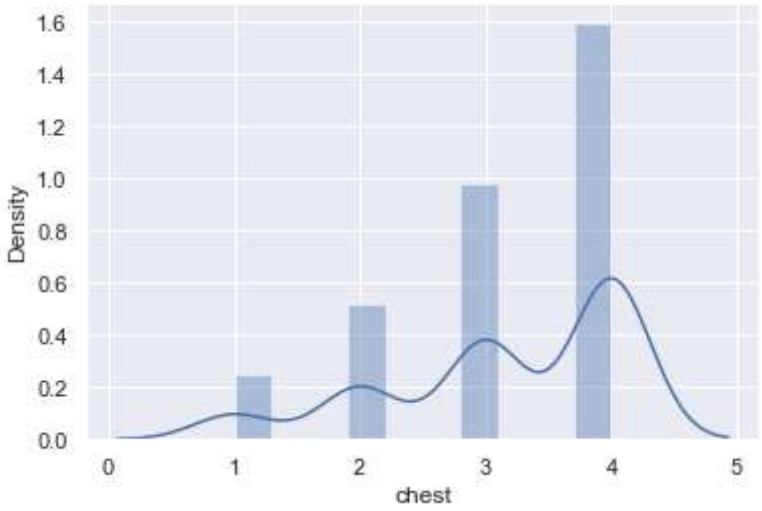
Out[27]: <AxesSubplot:xlabel='chest', ylabel='age'>



In [36]:
```
sns.distplot(df['chest'])
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

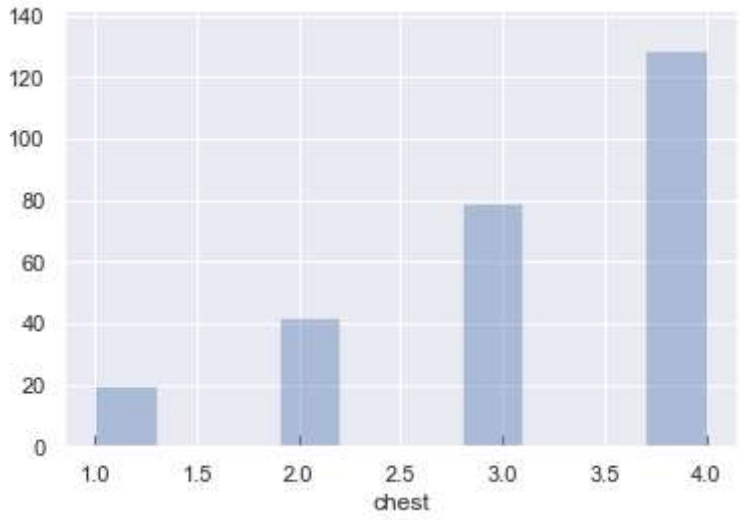
Out[36]: <AxesSubplot:xlabel='chest', ylabel='Density'>



In [37]:
```
sns.distplot(df['chest'], kde=False, rug=True)
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\distributions.py:2056: Futur
eWarning: The `axis` variable is no longer used and will be removed. Instead, a
ssign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```

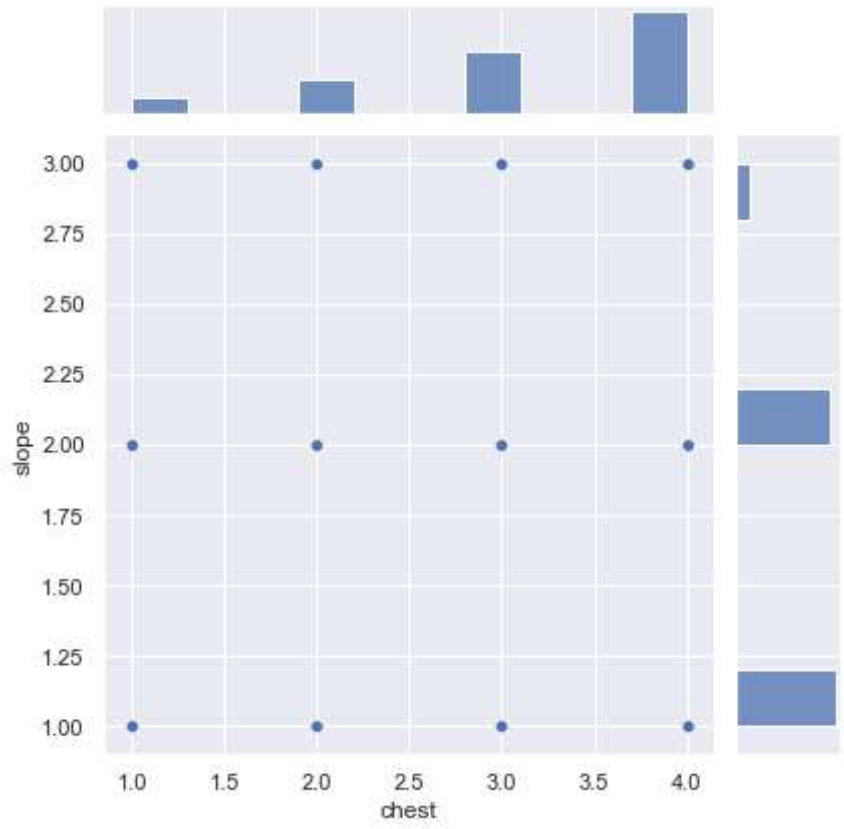
Out[37]: <AxesSubplot:xlabel='chest'>



In [38]:
```
sns.jointplot(df['chest'], df['slope'])
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

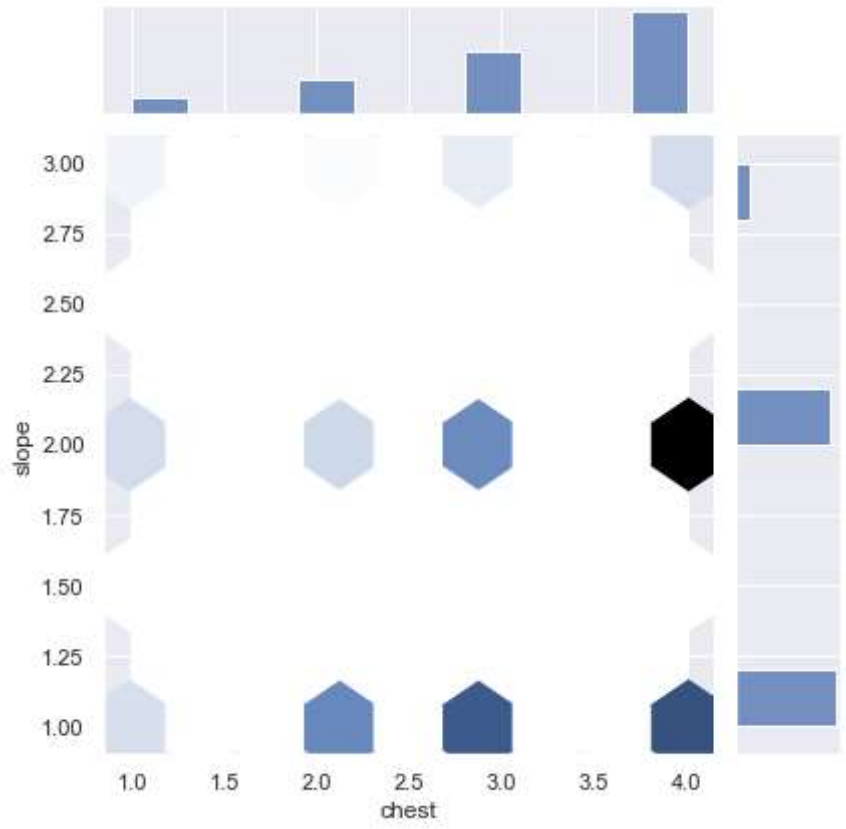
Out[38]: <seaborn.axisgrid.JointGrid at 0x1d86d103160>



In [39]:
```
sns.jointplot(df['chest'], df['slope'], kind="hex")
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

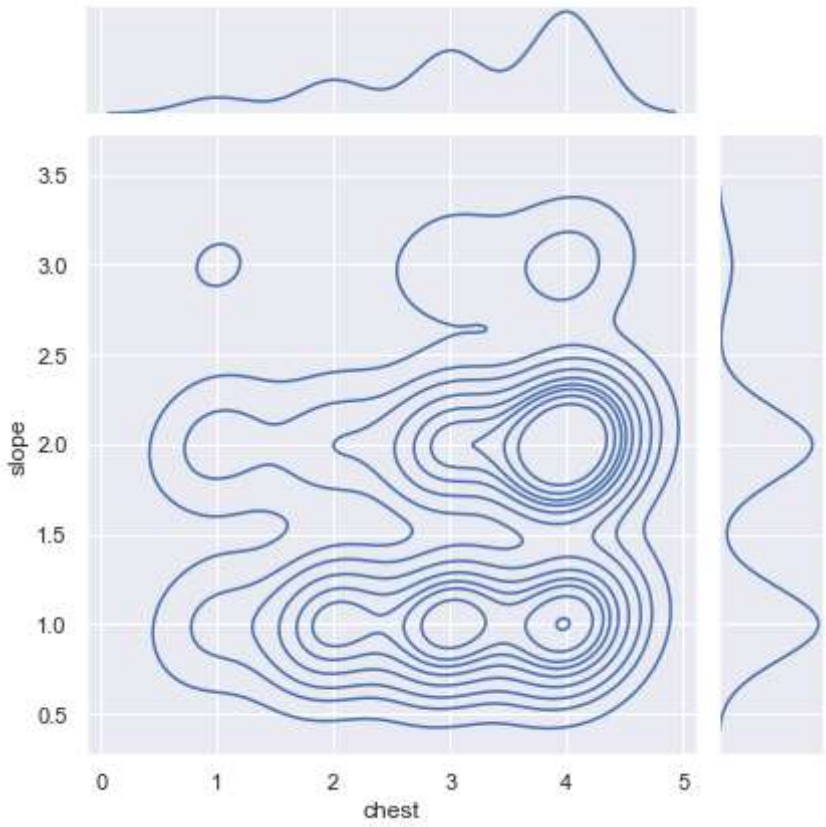
Out[39]:
```
<seaborn.axisgrid.JointGrid at 0x1d86d0223d0>
```



In [40]:
```
sns.jointplot(df['chest'], df['slope'], kind="kde")
```

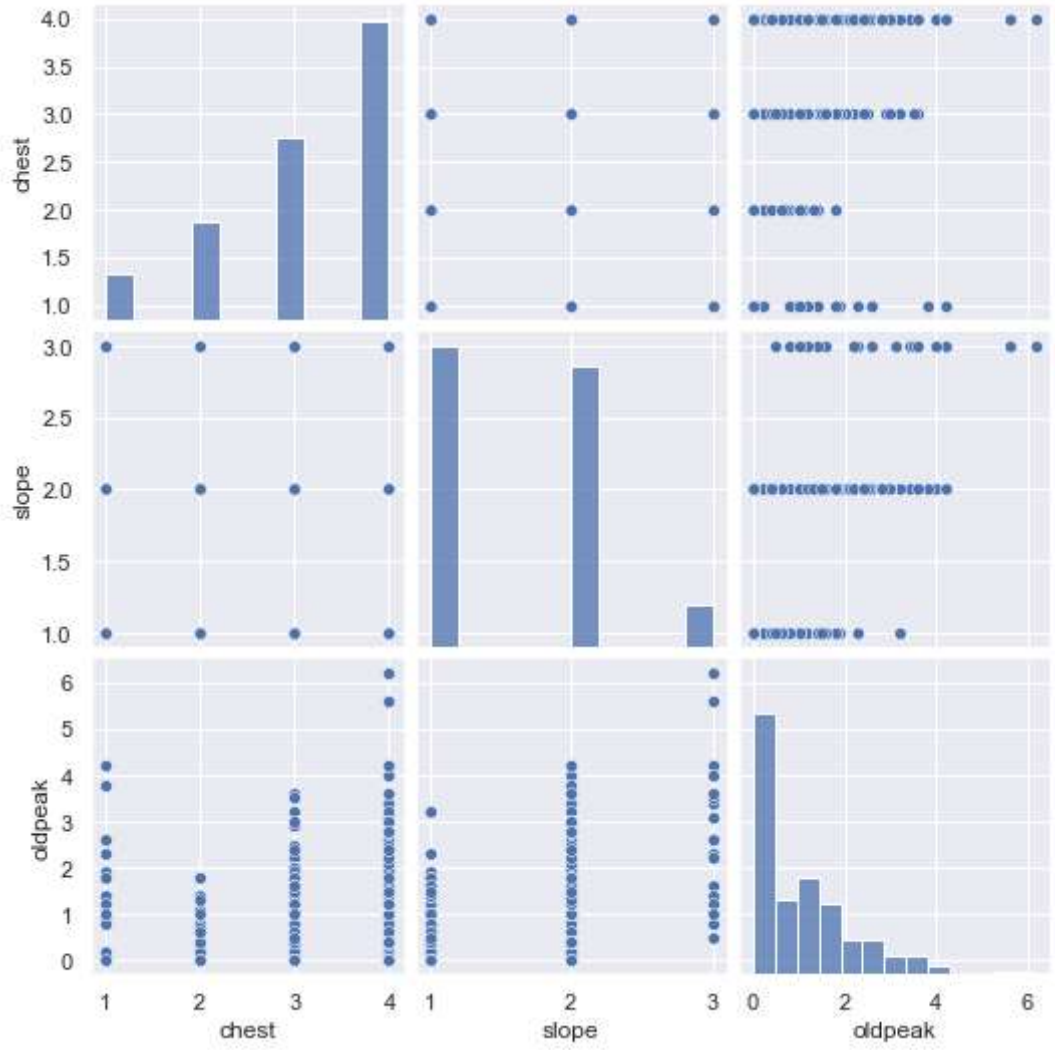
```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[40]: <seaborn.axisgrid.JointGrid at 0x1d86d14aa90>



In [41]:
```
sns.pairplot(df[['chest', 'slope', 'oldpeak']])
```

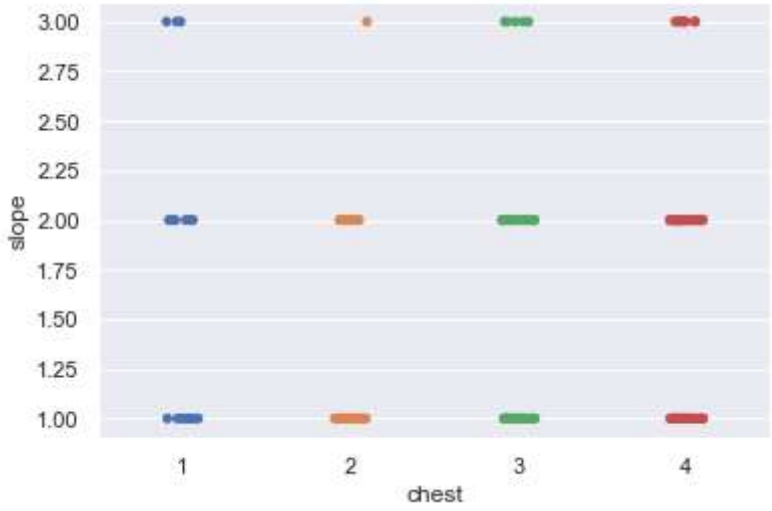
Out[41]: <seaborn.axisgrid.PairGrid at 0x1d86e3d14f0>



In [42]:
```
sns.stripplot(df['chest'], df['slope'])
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

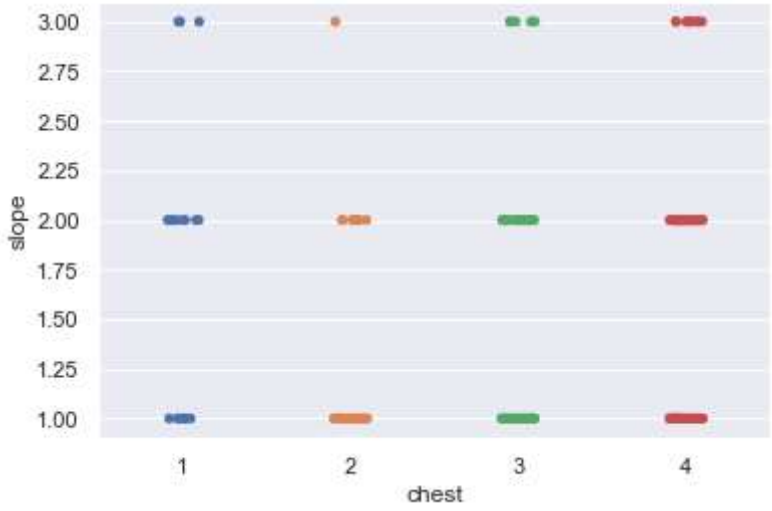
Out[42]: <AxesSubplot:xlabel='chest', ylabel='slope'>



In [43]:
```
sns.stripplot(df['chest'], df['slope'], jitter = True)
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

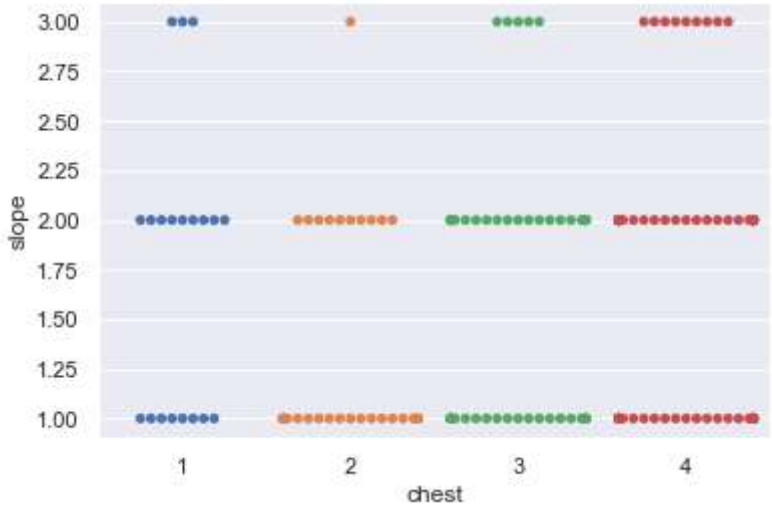
Out[43]: <AxesSubplot:xlabel='chest', ylabel='slope'>



In [44]:
```
sns.swarmplot(df['chest'], df['slope'])
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWar
ning: 42.9% of the points cannot be placed; you may want to decrease the size o
f the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWar
ning: 60.8% of the points cannot be placed; you may want to decrease the size o
f the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWar
ning: 72.9% of the points cannot be placed; you may want to decrease the size o
f the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

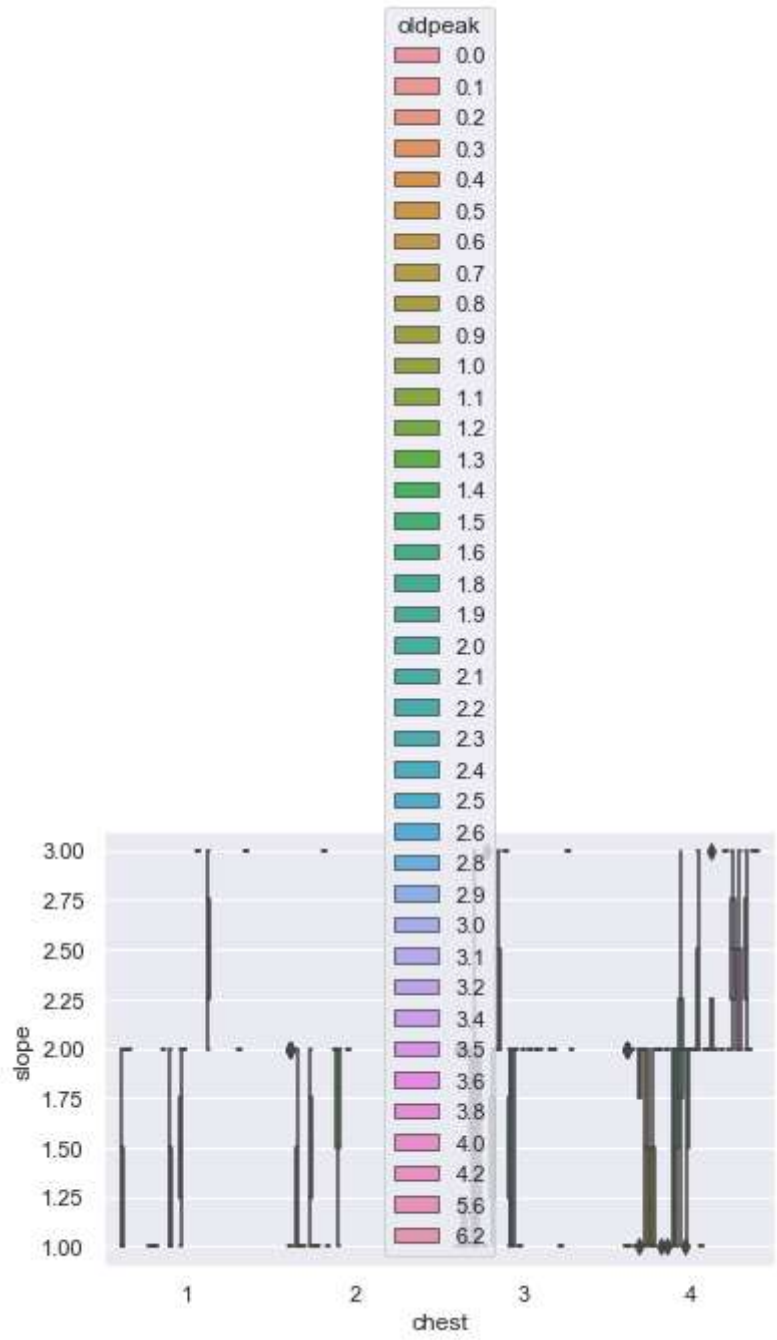
Out[44]: <AxesSubplot:xlabel='chest', ylabel='slope'>



In [45]:
```
sns.boxplot(df['chest'], df['slope'], hue=df['oldpeak'])
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

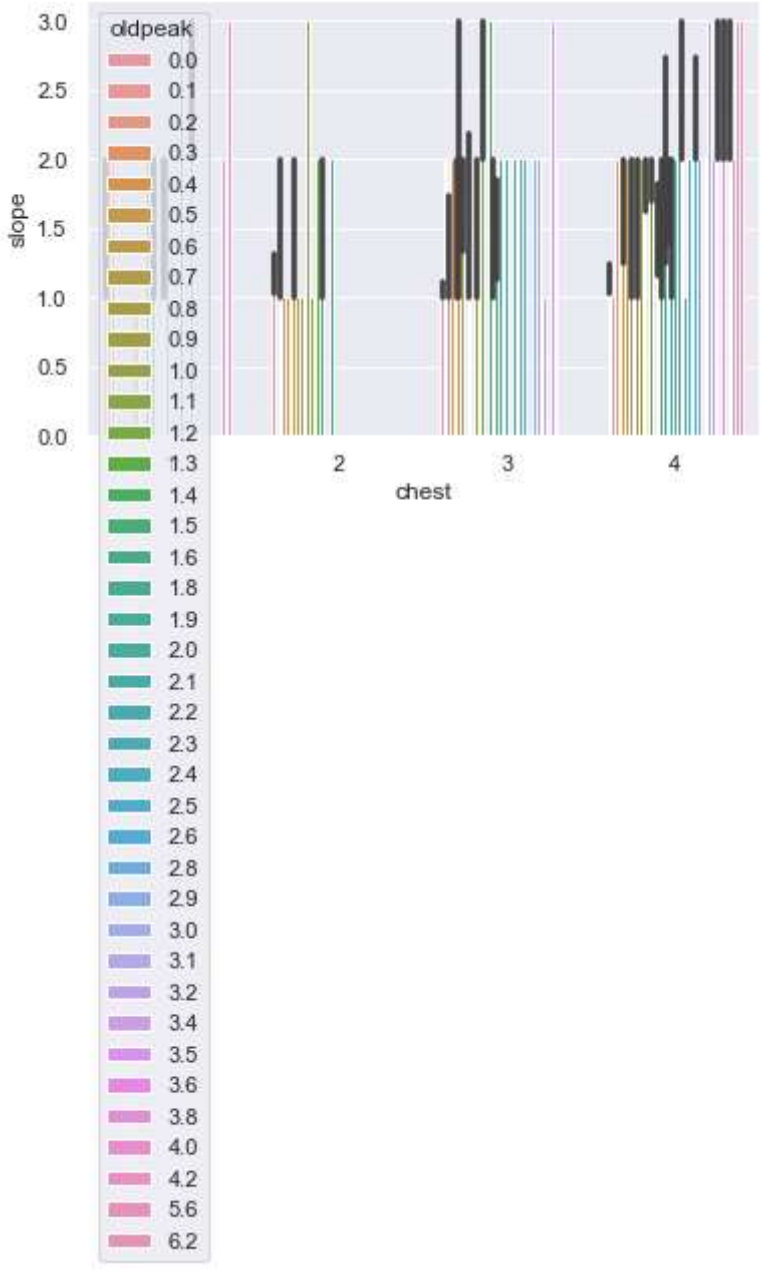
Out[45]: <AxesSubplot:xlabel='chest', ylabel='slope'>



In [46]:
```
sns.barplot(df['chest'], df['slope'], hue=df['oldpeak'])
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

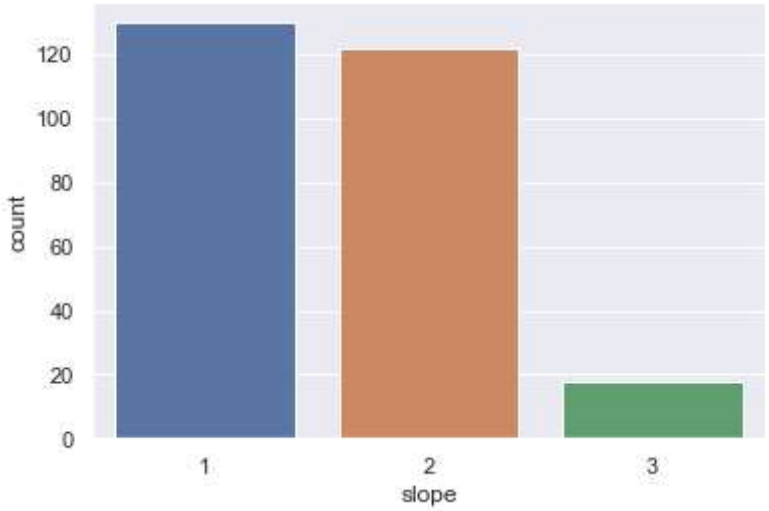
Out[46]: <AxesSubplot:xlabel='chest', ylabel='slope'>



In [47]:
```
sns.countplot(df['slope'])
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

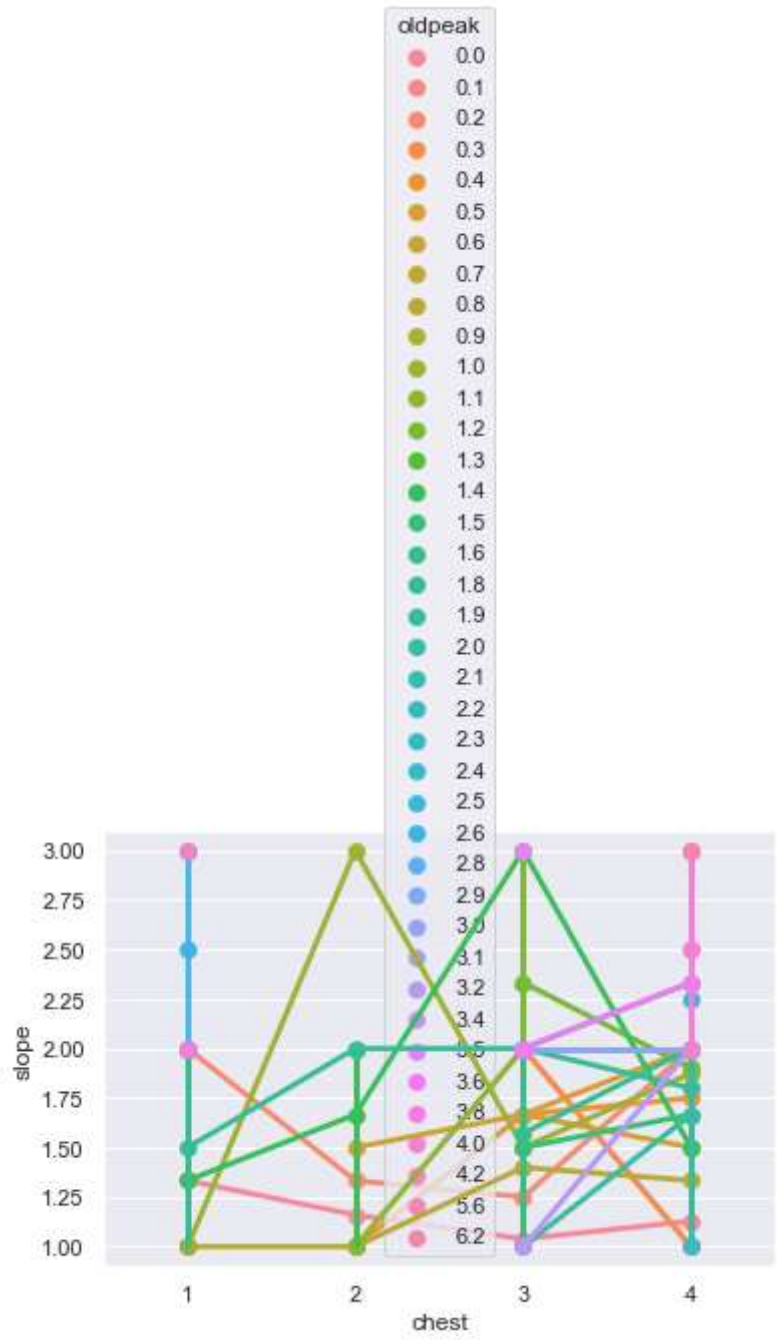
Out[47]:
```
<AxesSubplot:xlabel='slope', ylabel='count'>
```



In [48]:
```
sns.pointplot(df['chest'], df['slope'], hue=df['oldpeak'])
```

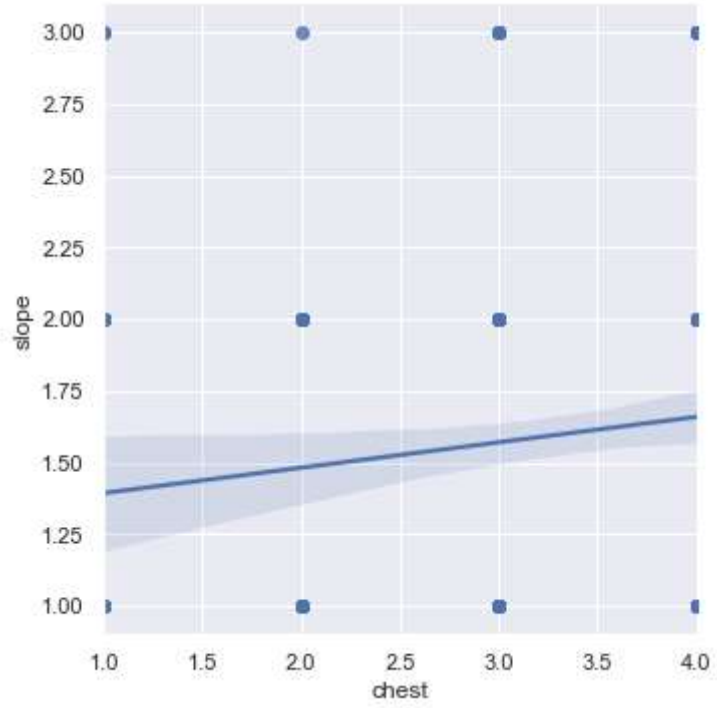
```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[48]: <AxesSubplot:xlabel='chest', ylabel='slope'>



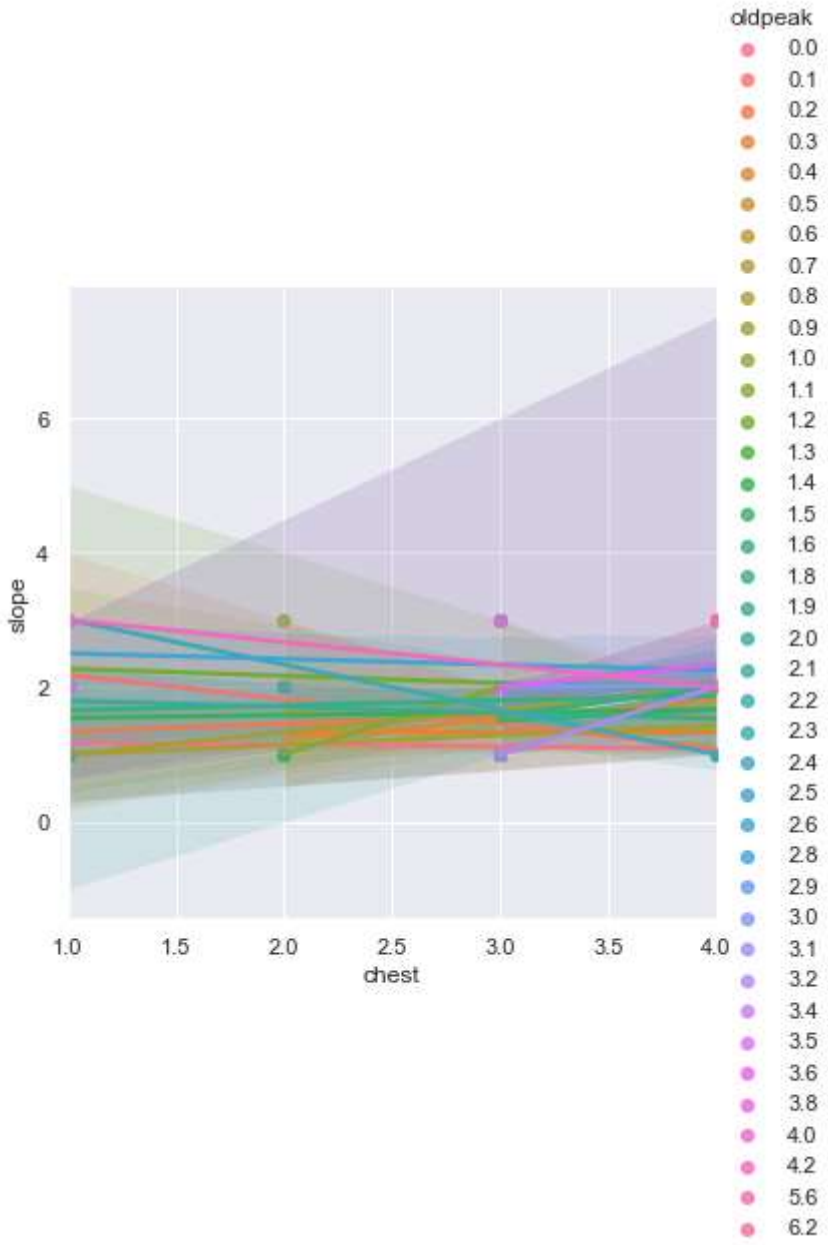
In [49]:
```
sns.lmplot(x="chest", y="slope", data=df)
```

Out[49]:
```
<seaborn.axisgrid.FacetGrid at 0x1d86faca5b0>
```



In [50]:
```
sns.lmplot(x="chest", y="slope", hue="oldpeak", data=df)
```

Out[50]:
```
<seaborn.axisgrid.FacetGrid at 0x1d86f783e50>
```



In [ ]:

In [ ]: