Lab 1

Mitchell Morrison, Kyle Kolodziej, Brian Pattison

Business Understanding

The *Heart Disease Dataset* from Kaggle (https://www.kaggle.com/sid321axn/heart-statlog-cleveland-hungary-final) is intended to help us draw conclusions about individuals who may be at risk of heart disease (Coronary Heart Disease, Hypertension, and Stroke). As the leading cause of death worldwide, it is important to gain an understanding of the factors that influence heart disease.

Our dataset is an aggregate of 5 datasets used in different research papers. A majority of the data was collected following 2008 from 5 hospitals in Ohio, Virginia, Hungary, Switzerland, and the UK. Each study attempted to collect test data normally linked to heart disease, including but not limited to age, sex, chest pain, cholesterol, blood sugar, electrocardiograms, heart rate, and excercise induced angina. These tests were conducted on individuals who were believed to be at high risk of heart disease, and concluded each test with a diagnosis. The dataset reflects each patients test data and will help us interpret what kinds of test data are related to increased risk of heart disease.

Our goal using this dataset is to classify if an individual is at high risk of heart disease and in need of seeing a cardiologist to review their data or conduct additional testing. There are no resources available for determining the current false positive and false negative rates of heart failure diagnoses. This is because diagnosing heart disease is presenting that an individual is at an increased risk (not a certainty when diagnosed) of having complications, meaning the actual symptoms and effects vary. Our dataset's target feature indicates that the individual is suffering from heart disease.

Since there is insufficient data presented around heart disease diagnoses, we intend on minimizing the potential for false negative classifications. This is because we feel it is more important to suggest an individual see a medical professional about their condition rather than concluding someone is not at risk of heart disease when they actually could be. Other medical diagnoses such as brest cancer screenings (American Cancer Society) and Alzheimers (Alzheimer's Association) have false negatives rates of nearly 20% and 12%, respectively. Using other medical diagnosis false negative rates as a standard, we would like to minimize the percent of false negative classifications to ~10% or less of diagnoses.

With respect to False Positive diagnoses, if a classification algorithm suggests someone is at high risk of heart disease when they are truly healthy, this only poses the risk of making the patient seek medical help and improve their health and heart. Although it is not our goal to classify individuals as at risk if they are not, we expect to over estimate the rate heart diagnoses than from that of the dataset for safety purposes. Having a sensitive

classification algorithm will only raise concern if there is notable features from a patient that are likely at risk.

In [23]:
```
# machine learning first lab

import pandas as pd
import numpy as np
import seaborn as sns
```

In [24]:
```
df = pd.read_csv('heart_statlog_cleveland_hungary_final.csv')
df.head()
```

Out[24]:

	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	resting ecg	max heart rate	exercise angina	oldpeak	ST slope	targe
0	40	1	2	140	289	0	0	172	0	0.0	1	0
1	49	0	3	160	180	0	0	156	0	1.0	2	1
2	37	1	2	130	283	0	1	98	0	0.0	1	0
3	48	0	4	138	214	0	0	108	1	1.5	2	1
4	54	1	3	150	195	0	0	122	0	0.0	1	0

In [25]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1190 entries, 0 to 1189
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   age                  1190 non-null   int64
 1   sex                  1190 non-null   int64
 2   chest pain type      1190 non-null   int64
 3   resting bp s         1190 non-null   int64
 4   cholesterol          1190 non-null   int64
 5   fasting blood sugar  1190 non-null   int64
 6   resting ecg          1190 non-null   int64
 7   max heart rate       1190 non-null   int64
 8   exercise angina      1190 non-null   int64
 9   oldpeak              1190 non-null   float64
 10  ST slope             1190 non-null   int64
 11  target               1190 non-null   int64
dtypes: float64(1), int64(11)
memory usage: 111.7 KB
```

Data Understanding

Using the information above about the data types for our dataset, we can see all of our data is numerical. However, the fields of sex, chest pain type, fasting blood sugar, resting ecg, exercise angina, and ST slope are nominal values. A more detailed representation of the data types are below:

1. Age: Patients Age in years (Numeric)
2. Sex: Gender of patient (Male - 1, Female - 0) (Nominal)

3. Chest Pain Type: Type of chest pain experienced by patient categorized into 1 typical, 2 typical angina, 3 non- anginal pain, 4 asymptomatic (Nominal)
4. resting bp s: Level of blood pressure at resting mode in mm/HG (Numerical)
5. cholestrol: Serum cholestrol in mg/dl (Numeric)
6. fasting blood sugar: Blood sugar levels on fasting > 120 mg/dl represents as 1 in case of true and 0 as false (Nominal)
7. resting ecg: Result of electrocardiogram while at rest are represented in 3 distinct values 0 : Normal 1: Abnormality in ST-T wave 2: Left ventricular hypertrophy (Nominal)
8. max heart rate: Maximum heart rate achieved (Numeric)
9. exercise angina: Angina induced by exercise 0 depicting NO 1 depicting Yes (Nominal)
10. oldpeak: Exercise induced ST-depression in comparison with the state of rest (Numeric)
11. ST slope: ST segment measured in terms of slope during peak exercise 0: Normal 1: Upsloping 2: Flat 3: Downsloping (Nominal)

Source: Dataset description on Kaggle

All of the attributes listed above are necessary to complete a thorough evaluation of each patient. Additionally, resources such as if the patient has a family history of heart disease would be useful for this dataset (but we don't have this unfortunately). Listing characteristics such as ST slope and chest pain type are much easier to work with as nominal data as opposed to categorical data.

Our dataset was missing values in the cholesterol attribute. We imputed data using the K-Nearest Neighbors algorithm implemented in class. There were 172 / 1190 missing cholesterol values that needed to be imputed, and only 1 missing blood pressure value.

In [26]:
```
import pandas as pd
import numpy as np
from sklearn.impute import KNNImputer
import copy
import matplotlib
import matplotlib.pyplot as plt
import warnings

print('Pandas:', pd.__version__)
print('Numpy:',np.__version__)

df = pd.read_csv('heart_statlog_cleveland_hungary_final.csv') # read in the csv

df.head()
```

```
Pandas: 1.0.3
Numpy: 1.18.2
```

Out[26]:

	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	resting ecg	max heart rate	exercise angina	oldpeak	ST slope	target
0	40	1	2	140	289	0	0	172	0	0.0	1	0

	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	resting ecg	max heart rate	exercise angina	oldpeak	ST slope	target
1	49	0	3	160	180	0	0	156	0	1.0	2	1
2	37	1	2	130	283	0	1	98	0	0.0	1	0
3	48	0	4	138	214	0	0	108	1	1.5	2	1
4	54	1	3	150	195	0	0	122	0	0.0	1	0

In [27]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1190 entries, 0 to 1189
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   age                  1190 non-null   int64
 1   sex                  1190 non-null   int64
 2   chest pain type      1190 non-null   int64
 3   resting bp s         1190 non-null   int64
 4   cholesterol          1190 non-null   int64
 5   fasting blood sugar  1190 non-null   int64
 6   resting ecg          1190 non-null   int64
 7   max heart rate       1190 non-null   int64
 8   exercise angina      1190 non-null   int64
 9   oldpeak              1190 non-null   float64
 10  ST slope             1190 non-null   int64
 11  target               1190 non-null   int64
dtypes: float64(1), int64(11)
memory usage: 111.7 KB
```

Based on this dataframe information, it appears that no values are missing in any of the 1190 entries. However, upon closer inspection, there are irregularities.

In [28]:
```
df.describe()
```

Out[28]:

	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	re
count	1190.000000	1190.000000	1190.000000	1190.000000	1190.000000	1190.000000	1190
mean	53.720168	0.763866	3.232773	132.153782	210.363866	0.213445	0
std	9.358203	0.424884	0.935480	18.368823	101.420489	0.409912	0
min	28.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0
25%	47.000000	1.000000	3.000000	120.000000	188.000000	0.000000	0
50%	54.000000	1.000000	4.000000	130.000000	229.000000	0.000000	0
75%	60.000000	1.000000	4.000000	140.000000	269.750000	0.000000	2
max	77.000000	1.000000	4.000000	200.000000	603.000000	1.000000	2

This description of the dataframe shows that there are zero values for cholestorol and for the resting blood pressure. A zero in those columns does not make sense given the context of those columns.

Data Imputation

In [29]:
```
cols = ['resting bp s', 'cholesterol']
df[cols] = df[cols].replace(['0', 0], np.nan)
df.columns = [c.replace(' ', '_') for c in df.columns]
df.describe()
```

Out[29]:

	age	sex	chest_pain_type	resting_bp_s	cholesterol	fasting_blood_s
count	1190.000000	1190.000000	1190.000000	1189.000000	1018.000000	1190.00
mean	53.720168	0.763866	3.232773	132.264929	245.906680	0.21
std	9.358203	0.424884	0.935480	17.971769	57.244599	0.40
min	28.000000	0.000000	1.000000	80.000000	85.000000	0.00
25%	47.000000	1.000000	3.000000	120.000000	209.000000	0.00
50%	54.000000	1.000000	4.000000	130.000000	240.000000	0.00
75%	60.000000	1.000000	4.000000	140.000000	276.000000	0.00
max	77.000000	1.000000	4.000000	200.000000	603.000000	1.00

We went through and replaced any zero in the 'resting bp s' and cholesterol columns with NaN, so they could be imputed.

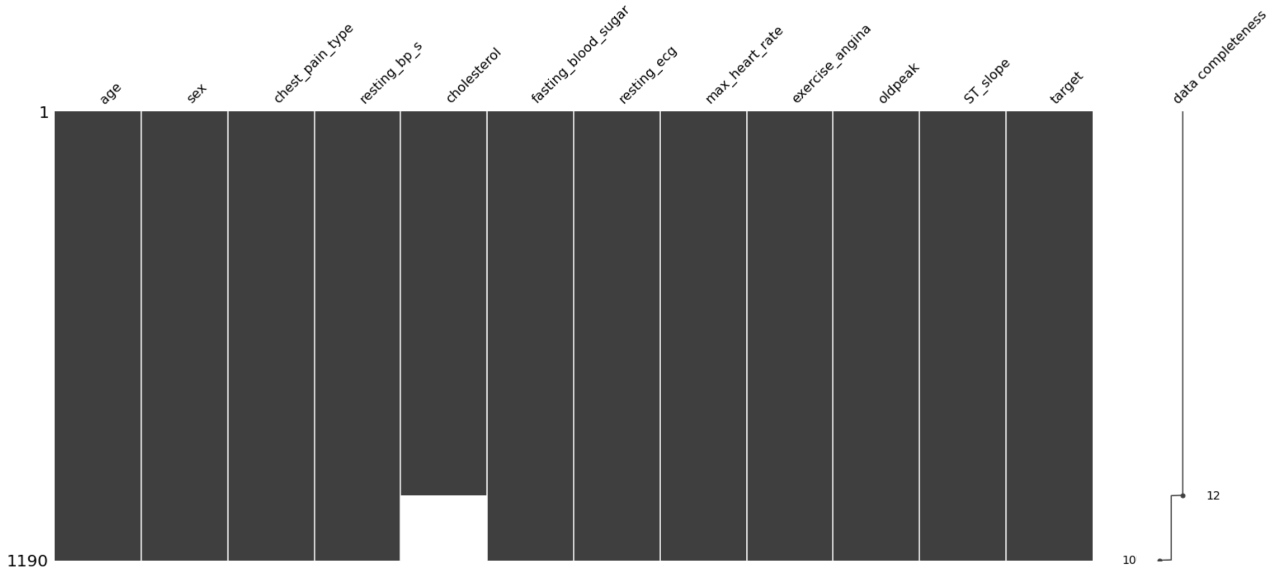
In [30]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1190 entries, 0 to 1189
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   age                  1190 non-null   int64  
 1   sex                  1190 non-null   int64  
 2   chest_pain_type      1190 non-null   int64  
 3   resting_bp_s         1189 non-null   float64
 4   cholesterol          1018 non-null   float64
 5   fasting_blood_sugar  1190 non-null   int64  
 6   resting_ecg          1190 non-null   int64  
 7   max_heart_rate       1190 non-null   int64  
 8   exercise_angina      1190 non-null   int64  
 9   oldpeak              1190 non-null   float64
 10  ST_slope             1190 non-null   int64  
 11  target               1190 non-null   int64  
dtypes: float64(3), int64(9)
memory usage: 111.7 KB
```

Now this dataframe information shows that there is 1 missing value in 'resting bp s' and 127 missing values in the 'cholesterol' columns.

In [31]:
```
# this python magics will allow plot to be embedded into the notebook
warnings.simplefilter('ignore', DeprecationWarning)
%matplotlib inline
import missingno as mn

mn.matrix(df.sort_values(by=["cholesterol","resting_bp_s"]), labels = True)
```

Out[31]:
```
<AxesSubplot:>
```



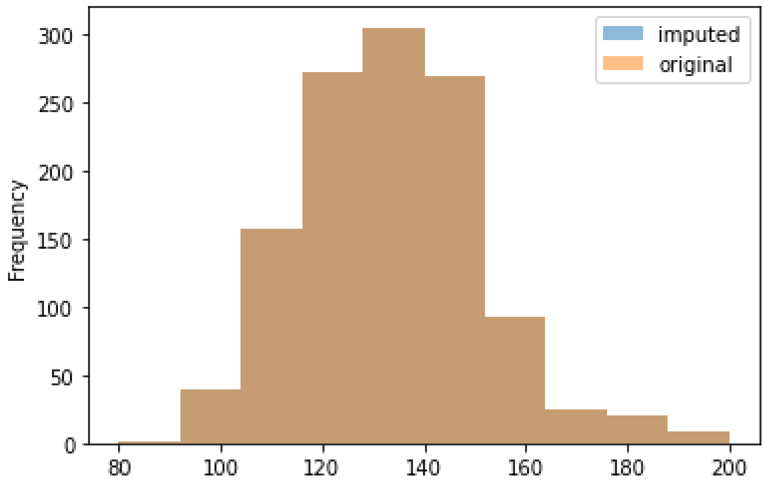
This topical graph shows that the only features missing any data are in cholesterol.

In [32]:
```python
from sklearn.impute import KNNImputer
import copy
#using K closest samples imputation.


# get object for imputation
knn_obj = KNNImputer(n_neighbors=3)

# create a numpy matrix from pandas numeric values to impute
temp = df[['resting_bp_s','cholesterol']].to_numpy()

# use sklearn imputation object
knn_obj.fit(temp)
temp_imputed = knn_obj.transform(temp)
#     could have also done:
# temp_imputed = knn_obj.fit_transform(temp)

df_imputed = copy.deepcopy(df) # not just an alias
df_imputed[['resting_bp_s','cholesterol']] = temp_imputed
df_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1190 entries, 0 to 1189
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   age                  1190 non-null   int64  
 1   sex                  1190 non-null   int64  
 2   chest_pain_type      1190 non-null   int64  
 3   resting_bp_s         1190 non-null   float64
 4   cholesterol          1190 non-null   float64
 5   fasting_blood_sugar  1190 non-null   int64  
 6   resting_ecg          1190 non-null   int64  
 7   max_heart_rate       1190 non-null   int64  
 8   exercise_angina      1190 non-null   int64  
 9   oldpeak              1190 non-null   float64
 10  ST_slope             1190 non-null   int64  
 11  target               1190 non-null   int64  
dtypes: float64(3), int64(9)
memory usage: 111.7 KB
```

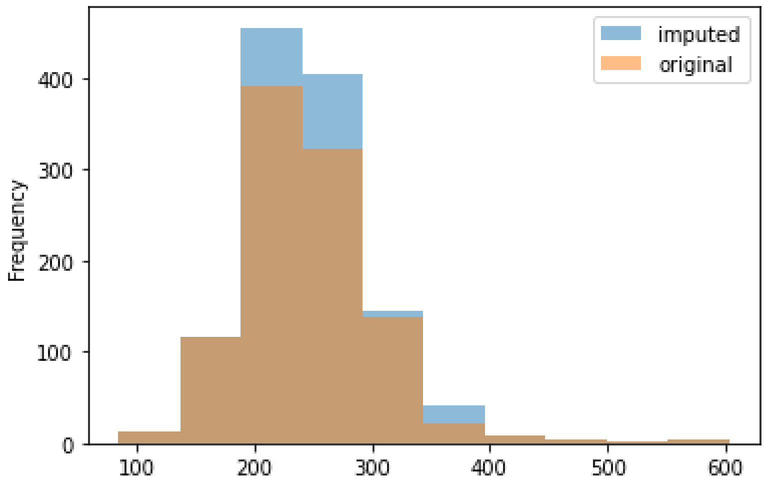
We used a K closest samples imputation inorder to impute the two columns.

In [33]:
```
df_imputed.resting_bp_s.plot(kind='hist', alpha=0.5, label="imputed")
df.resting_bp_s.plot(kind='hist', alpha=0.5, label="original")
plt.legend()
plt.show()
```



This chart shows that the imputation of the missing value in the 'resting_bp_s' column did not change the distribution of the frequencies.

In [34]:
```
df_imputed.cholesterol.plot(kind='hist', alpha=0.5, label="imputed")
df.cholesterol.plot(kind='hist', alpha=0.5, label="original")
plt.legend()
plt.show()
```



This chart shows that the imputation of the missing values in the 'cholesterol' column did not significantly change the distribution of the frequencies. It imputed a lot of values in the 200 to 300 range but those were the most frequent cholesterol values so the imputation did not change much about the data as a whole.

In [35]:
```
df_imputed.describe()
```

Out[35]:

	age	sex	chest_pain_type	resting_bp_s	cholesterol	fasting_blood_s
count	1190.000000	1190.000000	1190.000000	1190.000000	1190.000000	1190.00(

	age	sex	chest_pain_type	resting_bp_s	cholesterol	fasting_blood_s
mean	53.720168	0.763866	3.232773	132.264929	246.986756	0.21
std	9.358203	0.424884	0.935480	17.964210	55.540146	0.40
min	28.000000	0.000000	1.000000	80.000000	85.000000	0.00
25%	47.000000	1.000000	3.000000	120.000000	212.000000	0.00
50%	54.000000	1.000000	4.000000	130.000000	242.000000	0.00
75%	60.000000	1.000000	4.000000	140.000000	275.666667	0.00
max	77.000000	1.000000	4.000000	200.000000	603.000000	1.00

This description of the dataframe with imputed values shows that the means of both 'resting_bp_s' and 'cholesterol' were barely changed by the imputation of the missing values.

Visualizations

In this portion of our lab we will be using visualizations using our dataset to best interpret major factors that influence heart disease diagnoses.

General Correlations

In [36]:
```
df_copy = copy.deepcopy(df_imputed)
result = df_copy.corrwith(df["target"])
print('The values below represent each features correlation directly with the ta
print(result)
```

```
The values below represent each features correlation directly with the target va
riable
age                    0.262029
sex                    0.311267
chest_pain_type        0.460127
resting_bp_s           0.129995
cholesterol            0.116052
fasting_blood_sugar    0.216695
resting_ecg            0.073059
max_heart_rate        -0.413278
exercise_angina        0.481467
oldpeak                0.398385
ST_slope               0.505608
target                 1.000000
dtype: float64
```

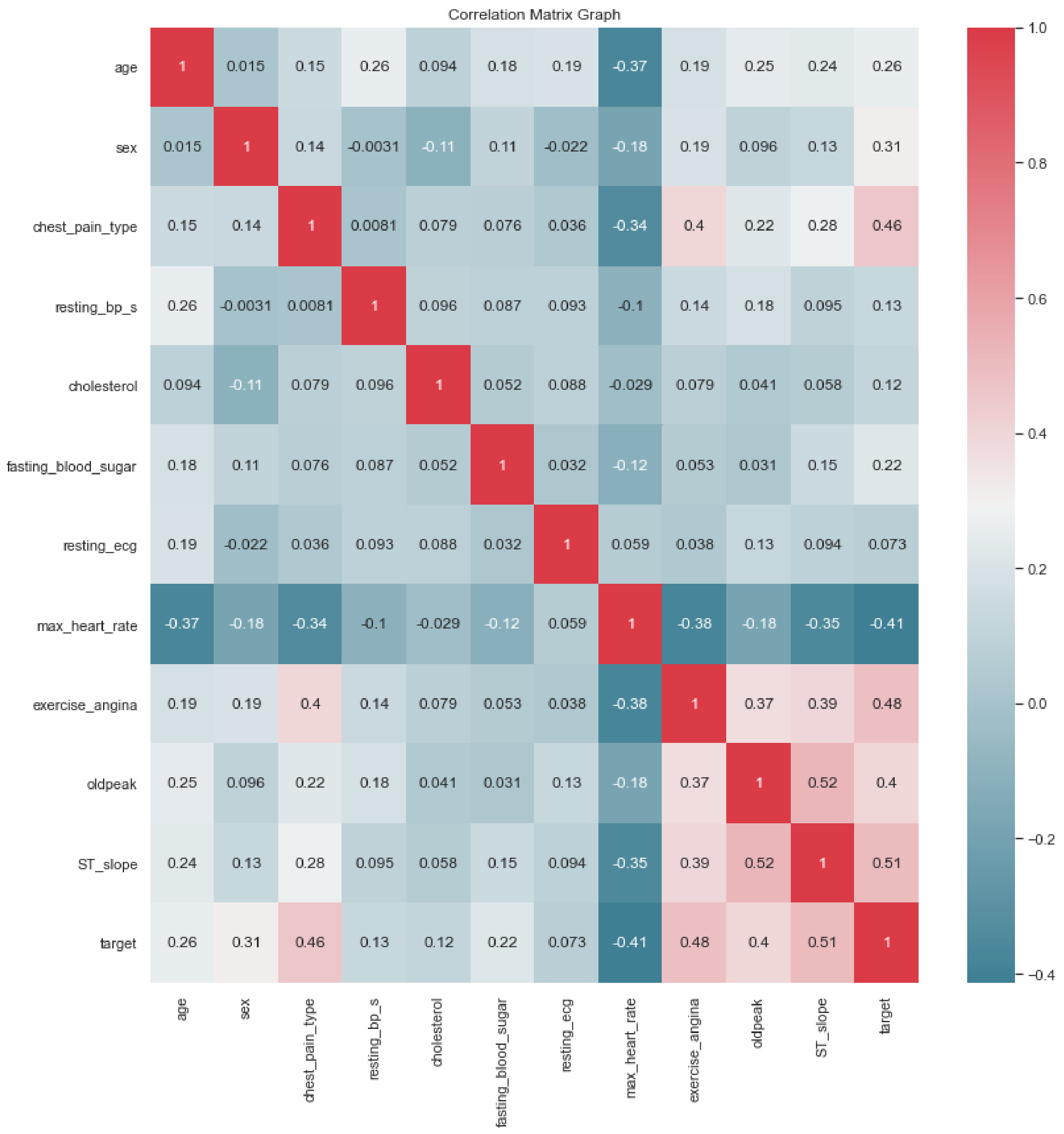
We will use the values above to draw a more clear correlation plot for features that show high relationship or warrant firther investigation.

In [37]:
```
# plot the correlation matrix using seaborn
sns.set(style="darkgrid") # one of the many styles to plot using

f, ax = plt.subplots(figsize=(12, 12))
cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(df_imputed.corr(), cmap=cmap, annot=True)
```

```
f.tight_layout()
plt.title('Correlation Matrix Graph')
```

Out[37]:

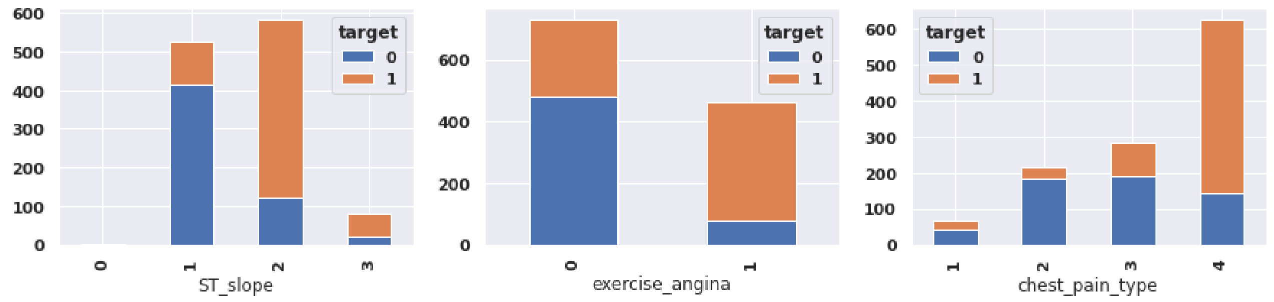
```
Text(0.5, 1.0, 'Correlation Matrix Graph')
```



The correlation matrix above helps us determine what factors are closely related in our dataset. It also helps us pick out features to explore that are highly related to the target variable, such as ST_slope, exercise angina, and chest pain type. Although attributes such as age and sex seem to have little relationship to each other, we intend on exploring these attributes as well to learn more about their role in heart diagnosis from our dataset.

In [38]:

```
font = {'family' : 'DejaVu Sans',
        'weight' : 'bold',
        'size'   : 10}

plt.rc('font', **font)
plt.figure(figsize=(15,3))
ax1 = plt.subplot(1,3,1)
ax2 = plt.subplot(1,3,2)
ax3 = plt.subplot(1,3,3)

pd.crosstab([df_imputed['ST_slope']], # categories to cross tabulate
            df_imputed.target).plot(kind='bar', stacked=True, ax=ax1)

pd.crosstab([df_imputed['exercise_angina']], # categories to cross tabulate
            df_imputed.target).plot(kind='bar', stacked=True, ax=ax2)

pd.crosstab([df_imputed['chest_pain_type']], # categories to cross tabulate
            df_imputed.target).plot(kind='bar', stacked=True, ax=ax3)

plt.show()
```



The stacked bar charts above show single attributes correlation to the target diagnosis variable. In the ST_slope subplot, we can clearly tell that patients with types 2 (flat) and 3 (downsloping) are much more likely to be diagnosed than those with type 1 (upsloping). Similarly we see much higher diagnosis rates for patients with exercise angina of type 1 (chest pain while exercising) and chest pain type 4 (asymptomatic). These charts warrant further investigation between the relationships of chest pain and exercise angina. Similarly, we should look further into ST_slope and its role in diagnoses.

Chest Pain and Exercise Induced Angina

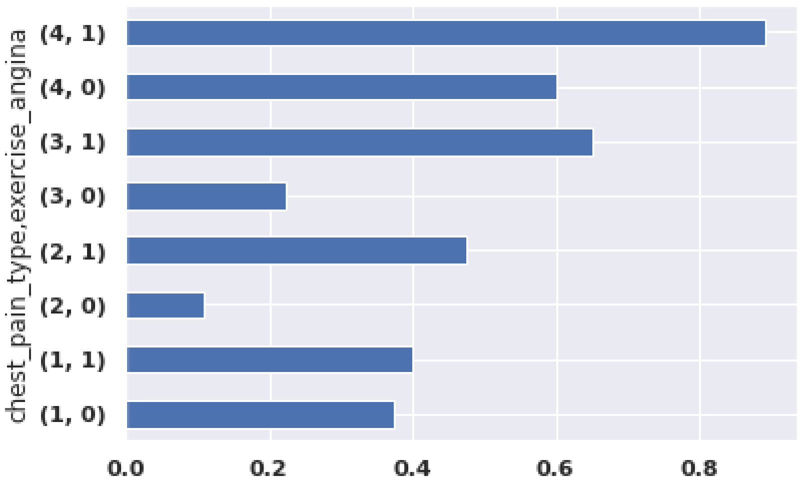
Question 1

Our correlation chart shows a significant relationship between patients chest pain type and exercise induced angina. How does the combination of these factors influence diagnosis rates, and where do we see similarities between types of chest pain and if or if not the patient has exercise angina?

Chest Pain Type: Type of chest pain experienced by patient categorized into 1 typical, 2 typical angina, 3 non- anginal pain, 4 asymptomatic (Nominal)
Exercise Angina: Angina induced by exercise 0 depicting NO, 1 depicting Yes (Nominal)

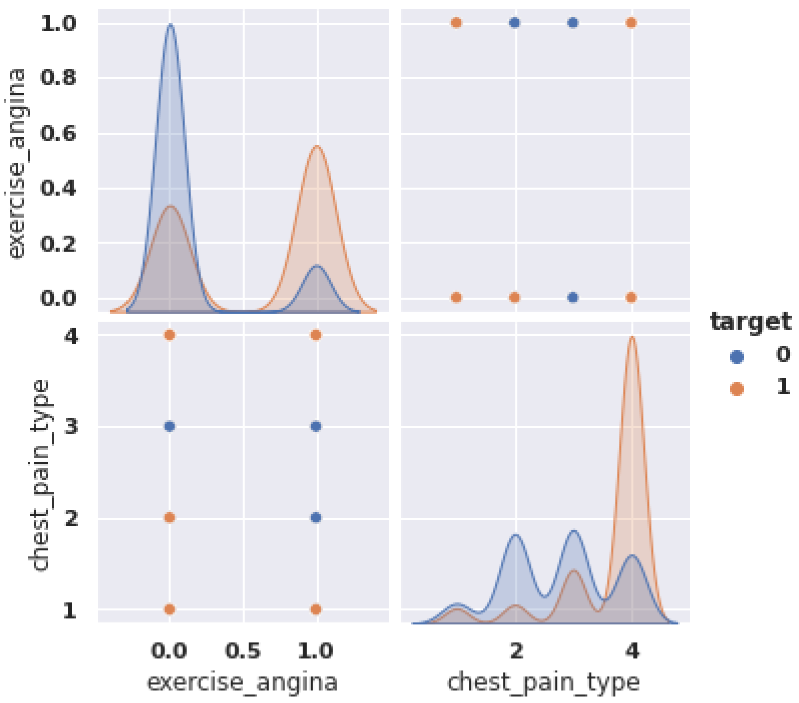
In [39]:
```
font = {'family' : 'DejaVu Sans',
        'weight' : 'bold',
        'size'   : 10}

plt.rc('font', **font)
```

```
df_grouped = df_imputed.groupby(by=['chest_pain_type', 'exercise_angina'])
diagnosed_rate = df_grouped.target.sum() / df_grouped.target.count()

ax = diagnosed_rate.plot(kind='barh')
plt.show()
```



In [40]:
```
font = {'family' : 'DejaVu Sans',
        'weight' : 'bold',
        'size'   : 10}

plt.rc('font', **font)

features = ['exercise_angina', 'chest_pain_type', 'target']

sns.pairplot(df_imputed[features], hue = "target", diag_kind = 'kde')
```

Out[40]:
```
<seaborn.axisgrid.PairGrid at 0x7f85e2e124d0>
```



This chart shows that overall, and exercise angina of 1 increases the chance of diagnosis. It also reaffirms the conclusion we reached earlier which is that a chest pain of type 4

significantly increases the diagnosis rate. It is interesting to note that a chest pain of type 4 combined with an exercise angina value of 1 has a diagnosis rate of over 90%. This chart also shows though, that somebody with chest paint type 3 and an exercise angina value of 1 has a higher diagnosis rate than chest pain type 4 with an exercise angina value of 0. The importance of exercise angina is most prevelent with chest pain type 2 and doesn't play much a factor in patients with chest pain type 4.

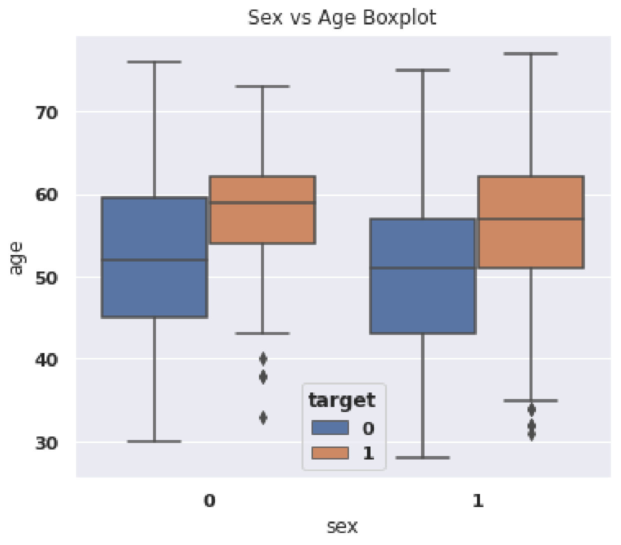
Age and Sex

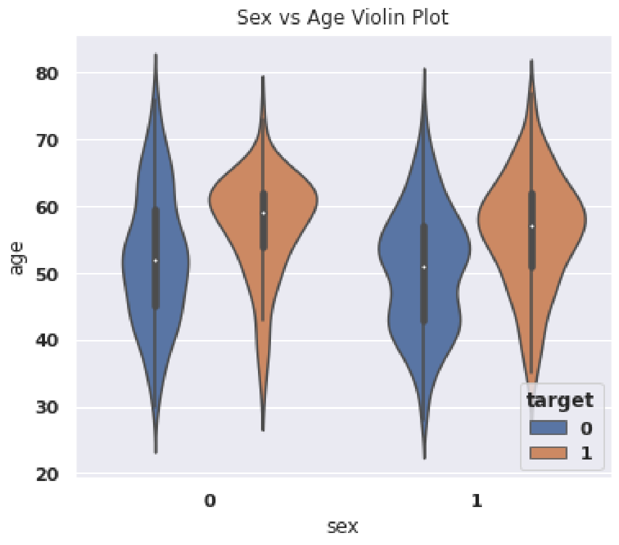
Question 2

What roles do sex and age play in diagnosis of heart disease? Although our correlation chart shows little relationship between the 2 factors, sex and age have been regarded as 2 of the most prevalent factors to influence heart disease in the US.

Source:
https://www.ahajournals.org/doi/full/10.1161/01.CIR.99.9.1165#:~:text=There%20is%20a%2

In [41]:
```
font = {'family' : 'DejaVu Sans',
        'weight' : 'bold',
        'size'   : 10}

plt.rc('font', **font)

plt.subplots(figsize=(20, 5))

plt.subplot(1,3,1)
sns.boxplot(x="sex", y="age", hue="target", data=df_imputed)
plt.title('Sex vs Age Boxplot')

plt.subplot(1,3,2)
sns.violinplot(x="sex", y="age", hue="target", data=df_imputed)
plt.title('Sex vs Age Violin Plot')

plt.show()
```





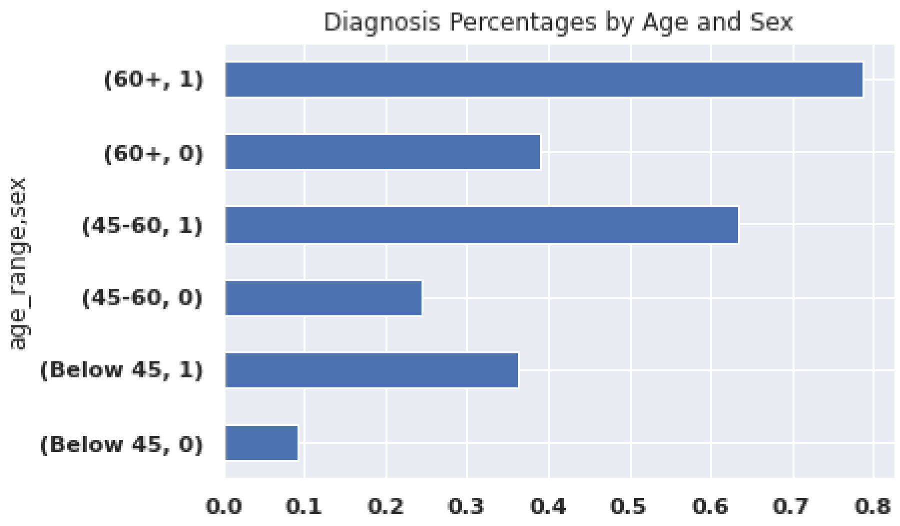
The "Sex vs Age" Box and Violin Plots subtly point to differences in diagnosis rates between men and women. However, since these charts are not varied enough for us to draw menaingful conclusions, we must investigate further. We will be using a bar plot with Age Ranges broken down as seen in class with the titanic dataset so we can see detailed depictions of how age and sex are directly related to developing heart disease.

In [42]:
```
# Breakdown categories for age to make comparisons
df_imputed['age_range'] = pd.cut(df_imputed['age'],[0,45,60,1e5],
                                 labels=['Below 45','45-60','60+']) # this creat

print(df_imputed.groupby(['sex']).size())
print(df_imputed.groupby(["age_range", "sex"]).size())
```

```
sex
0    281
1    909
dtype: int64
age_range  sex
Below 45   0       65
           1      187
45-60      0      139
           1      506
60+        0       77
           1      216
dtype: int64
```

This breakdown of men (1) and women (0) in each defined age range is necessary to keep in mind when drawing conclusions because we see a significantly larger sample of men than women in this dataset.

In [43]:
```
df_grouped = df_imputed.groupby(by=['age_range', 'sex'])
diagnosed_rate = df_grouped.target.sum() / df_grouped.target.count()

ax = diagnosed_rate.plot(kind='barh')
plt.title('Diagnosis Percentages by Age and Sex')
plt.show()
```



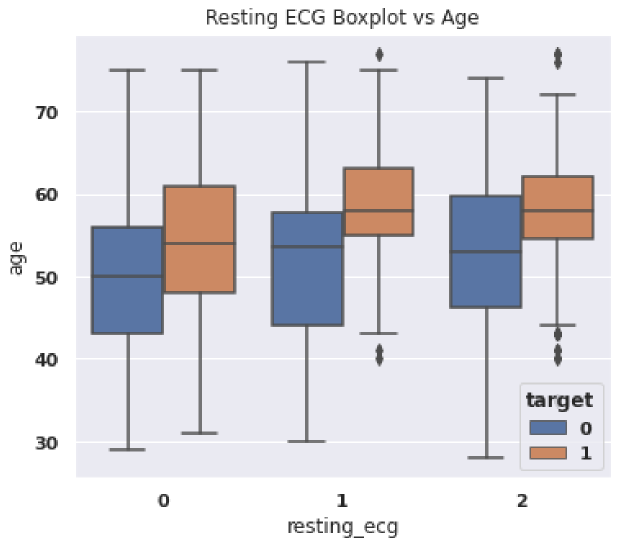
Sex: The bar plot above clearly denotes the differences in diagnosis rates between men and women of the same age range. The largest differences are in men and women over the age of 60, as we see men above 60 have diagnosis rates of nearly 80%, whereas women in the same age category are below 40%. This chart helps us make meaningful conclusions about the relationship between male and female diagnoses of heart disease.

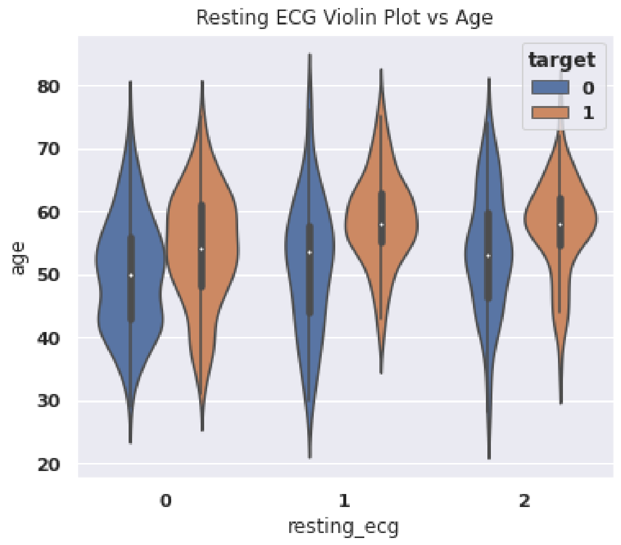
Age: This plot also shows the role age plays in an increasing chance of being diagnosed. Both men and women as they age are more likely to be diagnosed.

Resting ECG and Age

Result of electrocardiogram while at rest are represented in 3 distinct values 0 : Normal 1: Abnormality in ST-T wave 2: Left ventricular hypertrophy (Nominal)

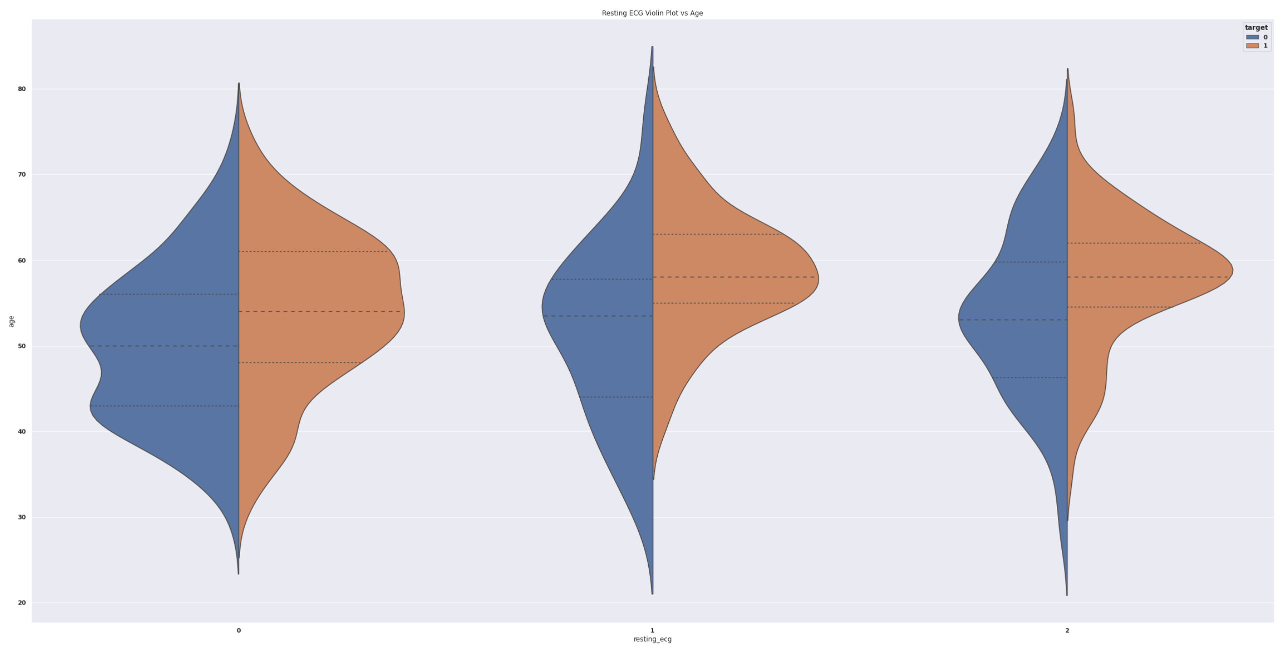
In [44]:
```
font = {'family' : 'DejaVu Sans',
        'weight' : 'bold',
        'size'   : 10}

plt.rc('font', **font)

plt.subplots(figsize=(20, 5))

plt.subplot(1,3,1)
sns.boxplot(x="resting_ecg", y="age", hue="target", data=df_imputed)
plt.title('Resting ECG Boxplot vs Age')

plt.subplot(1,3,2)
sns.violinplot(x="resting_ecg", y="age", hue="target", data=df_imputed)
plt.title('Resting ECG Violin Plot vs Age')

plt.show()
```



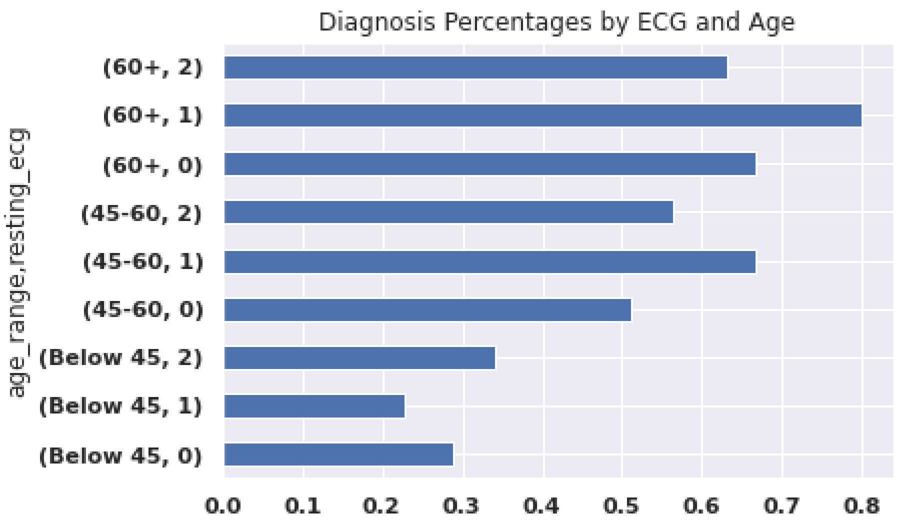


These plots showing resting ecg and age compared to having heart disease show that these two factors have a large impact. People over the age 50 who have a resting ecg value of 1, meaning they have abnormality in the ST-T wave, are much more likely to have heart disease. This conclusion holds true to people with a resting ecg value of 2, meaning they have left ventrical hypertrophy.

In [45]:
```python
font = {'family' : 'DejaVu Sans',
        'weight' : 'bold',
        'size'   : 22}

plt.rc('font', **font)

plt.subplots(figsize=(40, 20))

sns.violinplot(x="resting_ecg", y="age", hue="target", data=df_imputed,
                  split=True, inner="quart")
plt.title('Resting ECG Violin Plot vs Age')

plt.show()
```



In [46]:
```
font = {'family' : 'DejaVu Sans',
        'weight' : 'bold',
        'size'   : 10}

plt.rc('font', **font)

df_grouped = df_imputed.groupby(by=['age_range', 'resting_ecg'])
diagnosed_rate = df_grouped.target.sum() / df_grouped.target.count()

ax = diagnosed_rate.plot(kind='barh')
plt.title('Diagnosis Percentages by ECG and Age')
plt.show()
```



We can see some relationships between resting ECG and age. These graphs are much more difficult to interpret than the age and sex comparisons. This is because we know age has a significant impact on diagnosis rates. However, looking specifically at resting ecg levels we can see that type 1 (abnormal ST-T wave) in people over 60. This is not a solid conclusion, but merely a relation we have noticed.

Cholesterol

Question 3

Cholesterol is dubbed as one of the most influential factors in heart health. So, can cholesterol be easily controlled or are cholesterol values influenced by sex and age? How do sex and age play a role in cholesterol levels, and what relationship do each of these factors play in heart disease diagnosis?

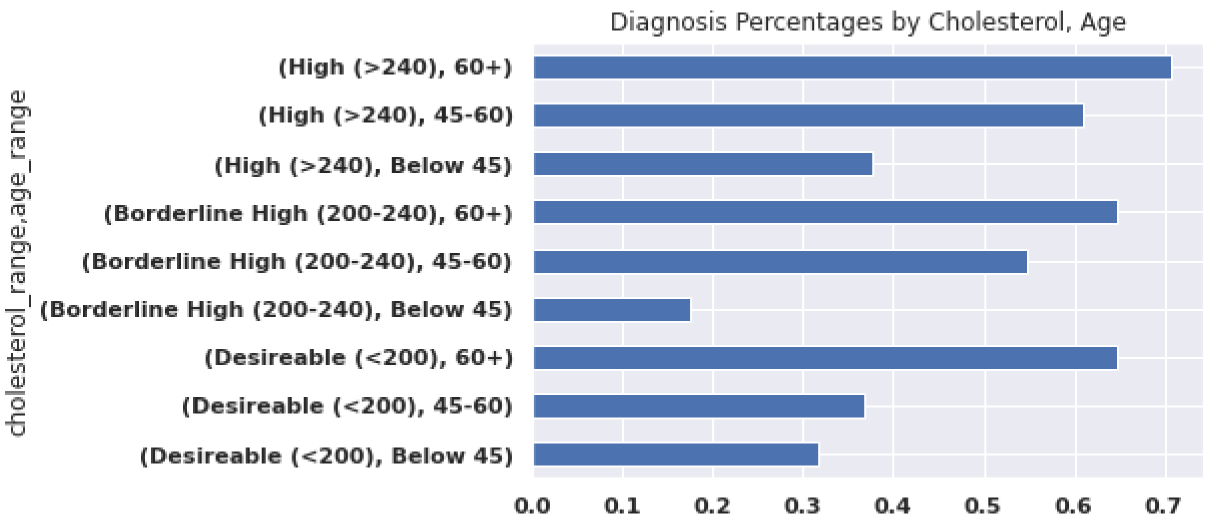
Source: https://www.sphealth.org/stories-news/stories/health-matters-cholesterol-major-contributor-heart-disease

In [47]:
```
df_imputed['cholesterol_range'] = pd.cut(df_imputed['cholesterol'],[0,200,240,1e
                                  labels=['Desireable (<200)','Borderline High (2

print(df_imputed.groupby(["cholesterol_range", "sex", "age_range"]).size())
```

```
cholesterol_range              sex   age_range
```

```
Desireable (<200)          0    Below 45      19
                                45-60         16
                                60+           10
                           1    Below 45      41
                                45-60         87
                                60+           27
Borderline High (200-240)  0    Below 45      25
                                45-60         35
                                60+           15
                           1    Below 45      61
                                45-60        179
                                60+           70
High (>240)                0    Below 45      21
                                45-60         88
                                60+           52
                           1    Below 45      85
                                45-60        240
                                60+          119
dtype: int64
```

In [48]:
```python
font = {'family' : 'DejaVu Sans',
        'weight' : 'bold',
        'size'   : 10}
plt.rc('font', **font)

df_grouped = df_imputed.groupby(by=['cholesterol_range', 'age_range'])
diagnosed_rate = df_grouped.target.sum() / df_grouped.target.count()

ax = diagnosed_rate.plot(kind='barh')
plt.title('Diagnosis Percentages by Cholesterol, Age')
plt.show()
```



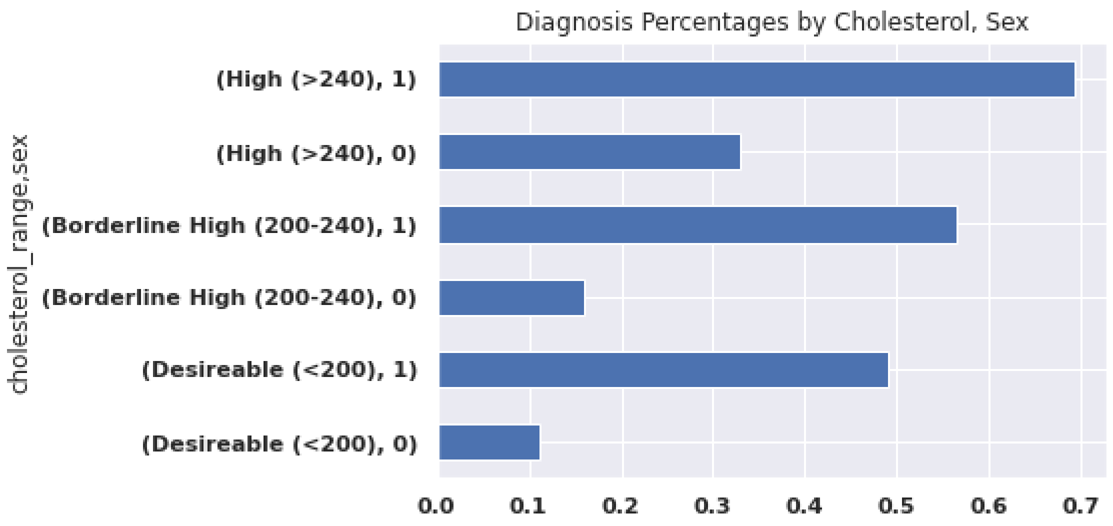
This bar chart indicates that individuals under the age of 45 from this dataset are not able to draw conclusions about diagnoses because even individuals with desireable cholesterol have a higher rate of diagnoses than those with borderline high cholesterol levels. Although we know the importance of cholesterol, the age breakdown only interferes with the interpretation of this chart. The following calculations of the percent diagnosed explicitly by cholesterol indicate that cholesterol values are of importance as there is a relationship between cholesterol levels and heart disease diagnosis.

In [49]:
```python
df_grouped = df_imputed.groupby(by=['cholesterol_range'])
diagnosed_rate = df_grouped.target.sum() / df_grouped.target.count()
```

```
print(diagnosed_rate)
```

```
cholesterol_range
Desireable (<200)           0.405000
Borderline High (200-240)   0.485714
High (>240)                 0.596694
Name: target, dtype: float64
```

In [50]:
```
font = {'family' : 'DejaVu Sans',
        'weight' : 'bold',
        'size'   : 10}
plt.rc('font', **font)

df_grouped = df_imputed.groupby(by=['cholesterol_range', 'sex'])
diagnosed_rate = df_grouped.target.sum() / df_grouped.target.count()

ax = diagnosed_rate.plot(kind='barh')
plt.title('Diagnosis Percentages by Cholesterol, Sex')
plt.show()
```



Finally, although cholesterol data suggests that there is a higher chance of diagnoses with higher levels of cholesterol, we again see the relationship between sex and diagnoses. Clearly from the plot above, we can tell that sex plays a large role in heart disease diagnoses, even as individuals have extremely high cholesterol levels.

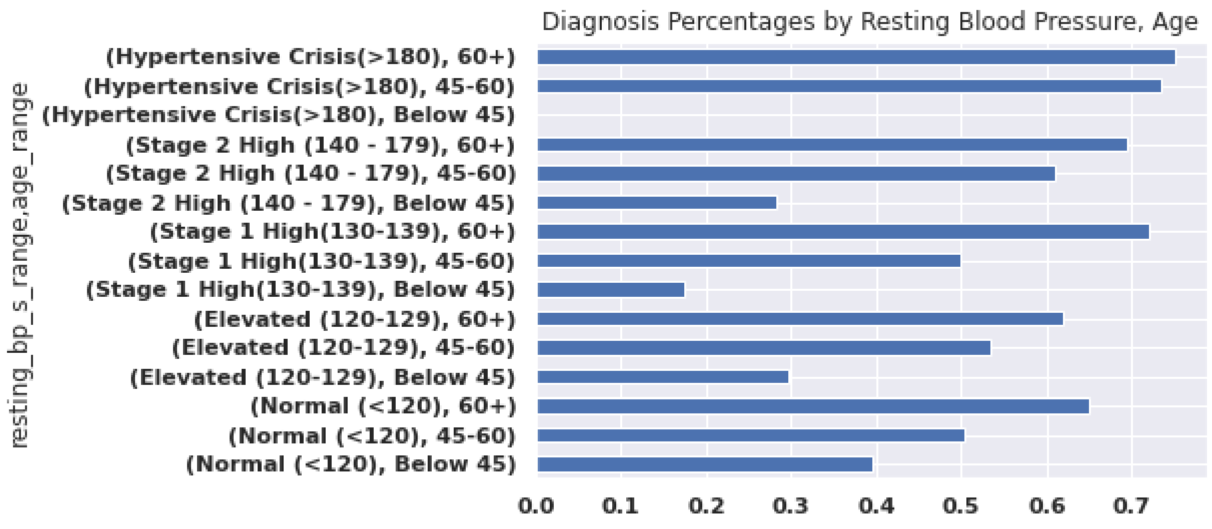
Resting Blood Pressure

In [51]:
```
df_imputed['resting_bp_s_range'] = pd.cut(df_imputed['resting_bp_s'],[0,119,129,
                                          labels=['Normal (<120)','Elevated (120
                                         'Stage 2 High (140 - 179)', 'Hypertensi

print(df_imputed.groupby(["resting_bp_s_range", "sex", "age_range"]).size())
```

```
resting_bp_s_range          sex  age_range
Normal (<120)               0    Below 45      25
                                 45-60         20
                                 60+           11
                            1    Below 45      51
                                 45-60         85
                                 60+           26
Elevated (120-129)          0    Below 45      17
                                 45-60         33
                                 60+            8
```

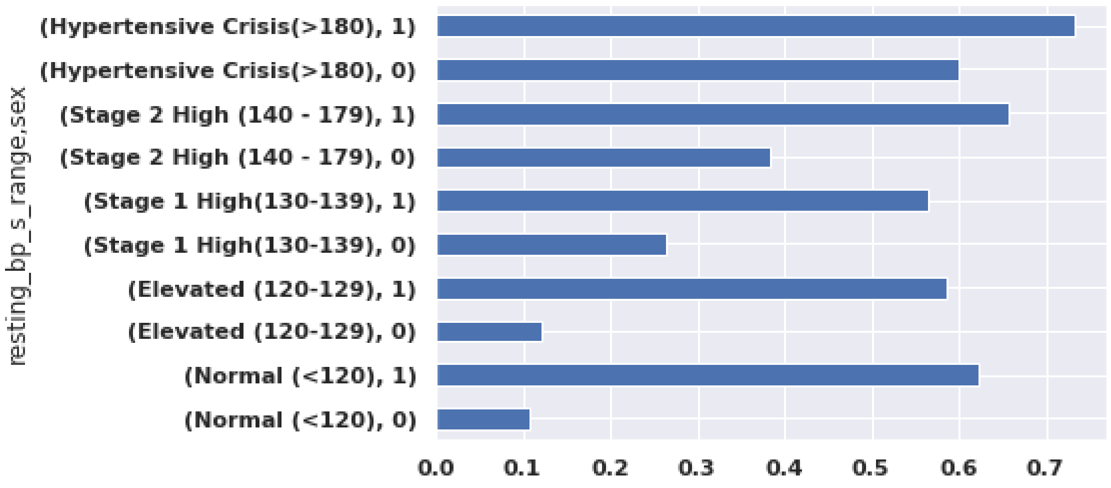
```
                              1     Below 45      54
                                    45-60        126
                                    60+           42
Stage 1 High(130-139)         0     Below 45      17
                                    45-60         47
                                    60+           12
                              1     Below 45      40
                                    45-60        111
                                    60+           49
Stage 2 High (140 - 179)      0     Below 45       5
                                    45-60         33
                                    60+           43
                              1     Below 45      41
                                    45-60        175
                                    60+           94
Hypertensive Crisis(>180)     0     Below 45       1
                                    45-60          6
                                    60+            3
                              1     Below 45       1
                                    45-60          9
                                    60+            5
dtype: int64
```

In [52]:
```
font = {'family' : 'DejaVu Sans',
        'weight' : 'bold',
        'size'   : 10}
plt.rc('font', **font)


df_grouped = df_imputed.groupby(by=['resting_bp_s_range', 'age_range'])
diagnosed_rate = df_grouped.target.sum() / df_grouped.target.count()

ax = diagnosed_rate.plot(kind='barh')
plt.title('Diagnosis Percentages by Resting Blood Pressure, Age')
plt.show()
```



This chart reaffirms that age is an important factor in diagnosis rates. However, we can conclude that a blood pressure of over 180 (hypertensive crisis) signifcantly increases the chances of diagnosis. This data is inconclusive for people below 45 though because it shows that people below 45 with a normal blood pressure have a higher rate of diagnosis than those with elevated, stage 1 high and stage 2 high blood pressures. Overall, this chart shows that blood pressure has some effect, but age is the more important factor here. It is important to remember that since this chart is broken into many small categories, some may contain significantly less data than others.

In [53]:
```
font = {'family' : 'DejaVu Sans',
        'weight' : 'bold',
        'size'   : 10}
plt.rc('font', **font)

df_grouped = df_imputed.groupby(by=['resting_bp_s_range', 'sex'])
diagnosed_rate = df_grouped.target.sum() / df_grouped.target.count()

ax = diagnosed_rate.plot(kind='barh')
plt.show()
```



This chart comparing blood pressure to sex is more interesting. First of all, women with hypertensive crisis have less of a chance of diagnosis than men with a healthy blood pressure. Another interesting feature is that as blood pressure decreases for women, their chance of diagnosis also decreases. For men, the chance of diagnosis is not correlated with their blood pressure. The data shows that men with a blood pressure of over 140 (Stage 2 High) are less likely to be diagnosed than men with a healthy blood pressure, though this probably means there is a lack of data for that category as seen in the breakdown of sex, age, and cholesterol above.

Dimensionality Reduction: Principal Component Analysis

Final section - Undergrads

In [54]:
```
from sklearn.decomposition import PCA

df_data = df_imputed.copy()
df_data.head() #copying over the table to run PCA
```

Out[54]:

	age	sex	chest_pain_type	resting_bp_s	cholesterol	fasting_blood_sugar	resting_ecg	max_
0	40	1	2	140.0	289.0	0	0	

	age	sex	chest_pain_type	resting_bp_s	cholesterol	fasting_blood_sugar	resting_ecg	max_
1	49	0	3	160.0	180.0	0	0	
2	37	1	2	130.0	283.0	0	1	
3	48	0	4	138.0	214.0	0	0	
4	54	1	3	150.0	195.0	0	0	

In [62]:
```
for col in ['target', 'age_range', 'cholesterol_range', 'resting_bp_s_range']:
    if col in df_data:
        del df_data[col]

df_data.head() #want to delete the range and target from this one
```

Out[62]:

	age	sex	chest_pain_type	resting_bp_s	cholesterol	fasting_blood_sugar	resting_ecg	max_
0	40	1	2	140.0	289.0	0	0	
1	49	0	3	160.0	180.0	0	0	
2	37	1	2	130.0	283.0	0	1	
3	48	0	4	138.0	214.0	0	0	
4	54	1	3	150.0	195.0	0	0	

Principal Component Analysis: all features

Below, a PCA will be used on all 11 original features of the imputed data set.

In [63]:
```
y = df['target'].copy()
y.head() #now have a variable with the target columndd
```

Out[63]:
```
0    0
1    1
2    0
3    1
4    0
Name: target, dtype: int64
```

In [64]:
```
pca = PCA(n_components=2)
pca.fit(df_data) # fit data and then transform it
X_pca = pca.transform(df_data)

# print the components

print ('pca:', pca.components_)
```

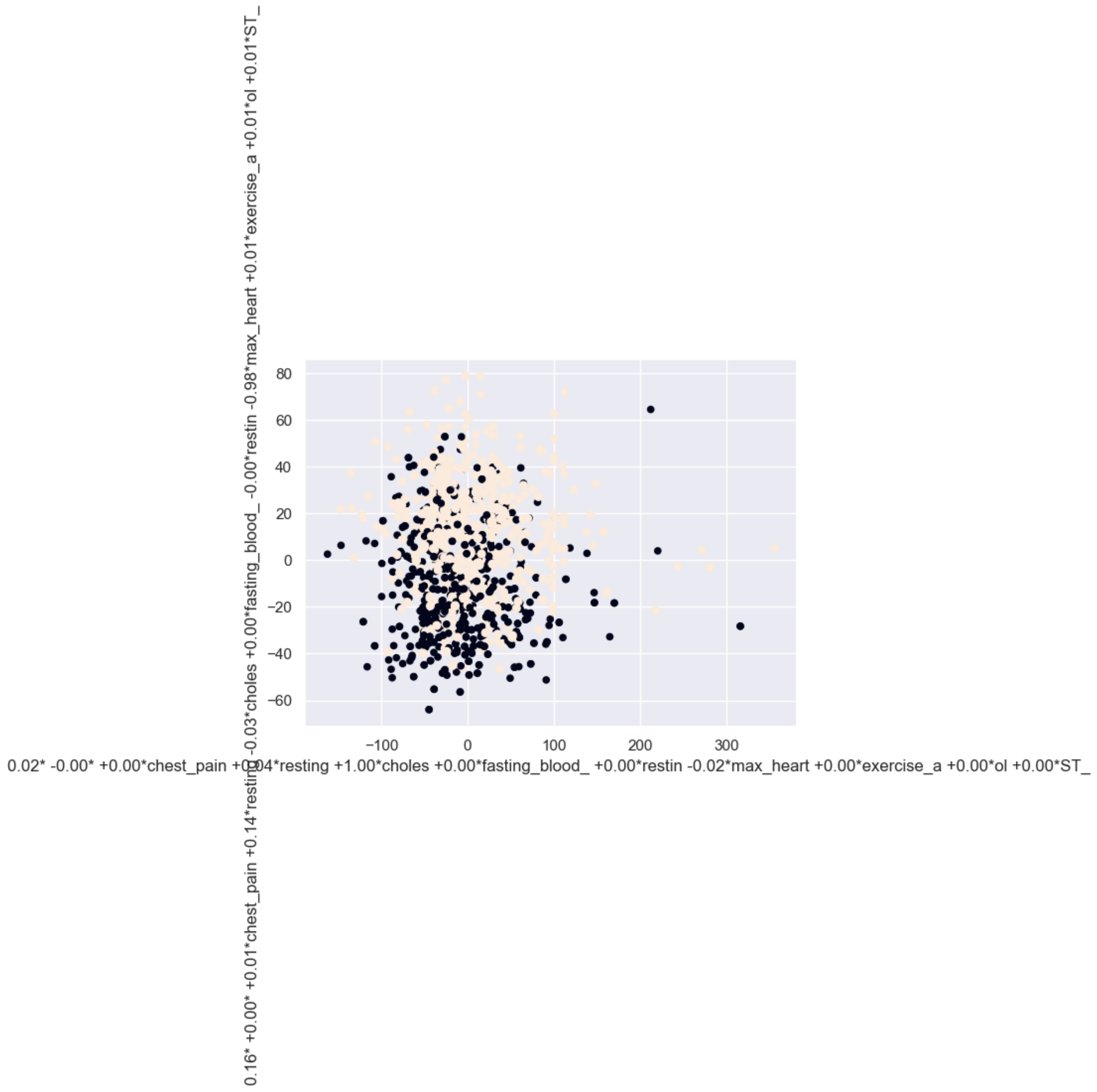
```
pca: [[ 1.73965788e-02 -8.14010679e-04  1.38929103e-03  3.52726145e-02
   9.99055886e-01  4.01030379e-04  1.39844699e-03 -1.82768030e-02
   7.39762472e-04  8.77178648e-04  6.86029825e-04]
 [ 1.56149312e-01  2.98029885e-03  1.19284846e-02  1.41506455e-01
 -2.56206528e-02  2.05919356e-03 -1.38794772e-03 -9.77026560e-01
  7.22688265e-03  8.68105706e-03  8.43046138e-03]]
```

In [65]:

```
def get_feature_names_from_weights(weights, names):
    tmp_array = []
    for comp in weights:
        tmp_string = ''
        for fidx,f in enumerate(names):
            if fidx>0 and comp[fidx]>=0:
                tmp_string+='+'
            tmp_string += '%.2f*%s ' % (comp[fidx],f[:-5])
        tmp_array.append(tmp_string)
    return tmp_array

from matplotlib import pyplot as plt
plt.style.use('default')
cmap = sns.set(style="darkgrid")
# now let's get to the Data Analytics!
data_cols = list(df_data.columns)

pca_weight_strings = get_feature_names_from_weights(pca.components_, data_cols)

# create some pandas dataframes from the transformed outputs
df_pca = pd.DataFrame(X_pca,columns=[pca_weight_strings])

from matplotlib.pyplot import scatter

# scatter plot the output, with the names created from the weights
ax = scatter(X_pca[:,0], X_pca[:,1], c=y, s=(y+2)*10, cmap=cmap)
plt.xlabel(pca_weight_strings[0])
plt.ylabel(pca_weight_strings[1])
```

Out[65]:

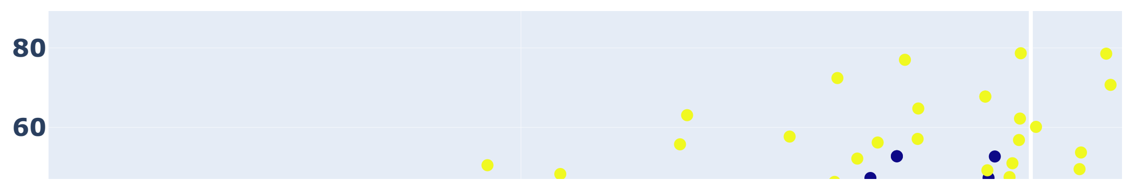
```
Text(0, 0.5, '0.16* +0.00* +0.01*chest_pain +0.14*resting -0.03*choles +0.00*fas
ting_blood_ -0.00*restin -0.98*max_heart +0.01*exercise_a +0.01*ol +0.01*ST_ ')
```



It is unclear in this scatter plot whether there are any groupings of data points that can be used to reduce the dimensionality of the data set. Additional graphs will be drawn below to further investigate a possible grouping of features that could reduce the dimensionality.

In [69]:
```
# Manipulated example from https://github.com/teddyroland/python-biplot/blob/mas
def biplot(pca, dat, title=''):

    import plotly
    from plotly.graph_objs import Bar, Line
    from plotly.graph_objs import Scatter, Layout
    from plotly.graph_objs.scatter import Marker
    from plotly.graph_objs.layout import XAxis, YAxis
    plotly.offline.init_notebook_mode() # run at the start of every notebook

    # 0,1 denote PC1 and PC2; change values for other PCs
    xvector = pca.components_[0]
    yvector = pca.components_[1]
    tmp = pca.transform(dat.values)
    xs = tmp[:,0]
    ys = tmp[:,1]

    annotations = [Scatter(x=xs, y=ys, mode ='markers',
                           marker=Marker(color=y), #Changed this to y (the targe
                           name='PCA Trans. Data')]
    for i in range(len(xvector)):
        txt = list(dat.columns.values)[i]
        annotations.append(
                Scatter(
                    x=[0, xvector[i]*max(xs)],
                    y=[0, yvector[i]*max(ys)],
                    mode='lines+text',
                    text=['', txt],
                    name=txt,
                ))

    plotly.offline.iplot({
        "data": annotations,
        "layout": Layout(xaxis=XAxis(title='Principal Component One'),
                          yaxis=YAxis(title='Principal Component Two'),
                         title=title)
    })


    plt.show()

#X = iris.data don't need this
pca = PCA(n_components=11) # Number of components for the # of features that I'm
pca.fit(df_data)
biplot(pca,pd.DataFrame(df_data,columns=data_cols),'Heart Biplot')
```

Heart Biplot



In looking at this Biplot, it appears that there are no clear correlations between the PCA's and features when investigating all 11 features of the data set. However, it appears there may be a grouping of features that could use further investigation to determine whether their dimensionality can be reduced. Specificially, the features old peak and exercise angina appear to be pointing in the same direction (nearly on top of each other). Close to their direction is ST slope, chest pain type, and age. Below, we will focus on looking at old peak, exercise angina, ST slope, and chest pain type to determine if there is a combination of these features that could lead us to reduce the dimensionality of this data set. Age is not going to be included in this subset of features as we assume that age will not be able to combine with one of these specific health features to be able to reduce the dimensionality.

In [70]:
```
def plot_explained_variance(pca):
    import plotly
    from plotly.graph_objs import Bar, Line
    from plotly.graph_objs import Scatter, Layout
    from plotly.graph_objs.scatter import Marker
    from plotly.graph_objs.layout import XAxis, YAxis
    plotly.offline.init_notebook_mode() # run at the start of every notebook

    explained_var = pca.explained_variance_ratio_
    cum_var_exp = np.cumsum(explained_var)

    plotly.offline.iplot({
        "data": [Bar(y=explained_var, name='individual explained variance'),
                 Scatter(y=cum_var_exp, name='cumulative explained variance')
            ],
        "layout": Layout(xaxis=XAxis(title='Principal components'), yaxis=YAxis(
    })

pca = PCA(n_components=8)
X_pca = pca.fit(df_data)
plot_explained_variance(pca)
```

Here is a plot demonstrating the variance that the Principal Components cover. In looking at this plot, it looks like there is a naturally "knee" after the 4th Principal Component. Respectively, these four Principal Components covers ~99.9% of the original 11 features (after imputation) of the data set. The 4th Principal Component may not technically be necessary due to the fact the first three components cover ~98.3% of the features.

Principal Component Analysis: old peak, exercise angina, ST slope, and chest pain type

As stated above, the features old peak, exercise angina, ST slope, and chest pain type appear they may be correlated such that some sort of dimensionality reduction could be applicable. A PCA will be drawn below with a comparison of these four features.

In [71]:
```
df_data2 = df_data.copy()
for col in ['target', 'age_range', 'cholesterol_range', 'age', 'sex', 'resting_b
            'cholesterol', 'fasting_blood_sugar', 'resting_ecg', 'max_heart_rate
    if col in df_data2:
        del df_data2[col]

df_data2.head() #want to delete the range and target from this one
```

Out[71]:

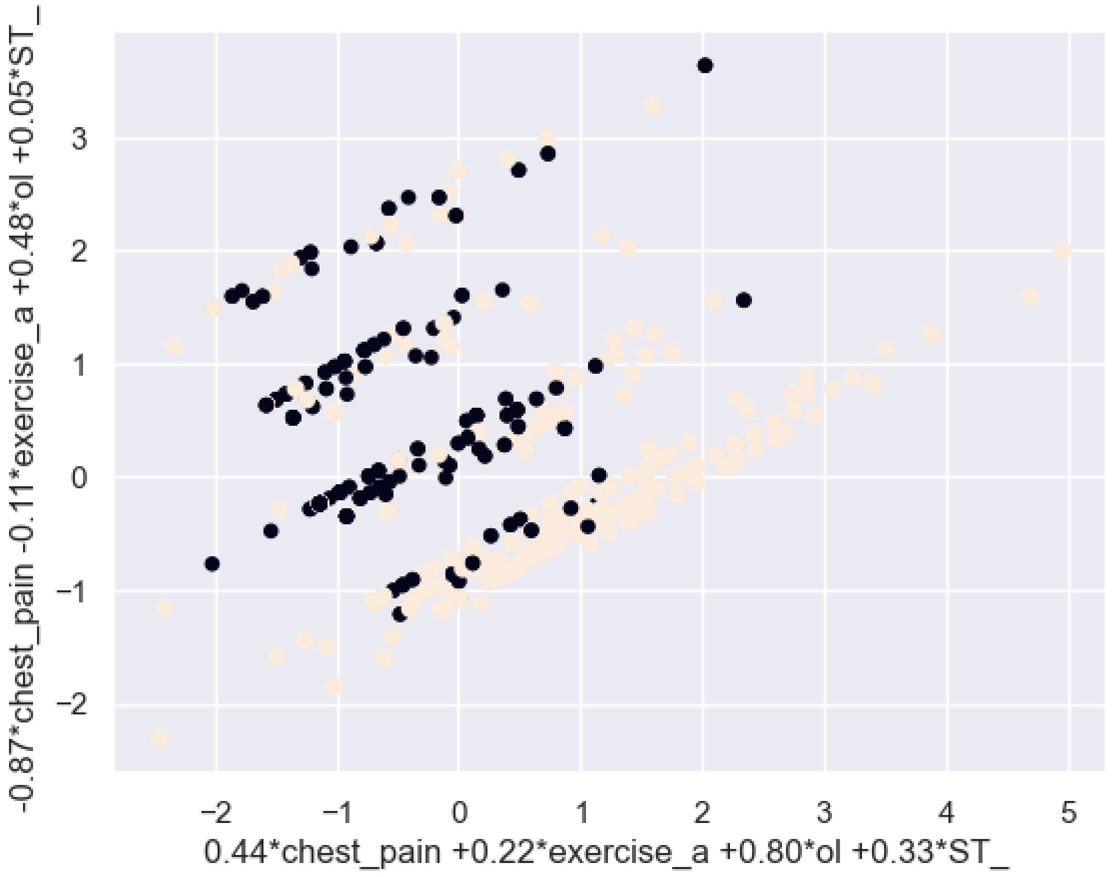
	chest_pain_type	exercise_angina	oldpeak	ST_slope
0	2	0	0.0	1
1	3	0	1.0	2
2	2	0	0.0	1
3	4	1	1.5	2
4	3	0	0.0	1

In [72]:
```
pca = PCA(n_components=2)
pca.fit(df_data2) # fit data and then transform it
X_pca = pca.transform(df_data2)

# print the components

print ('pca:', pca.components_)
```

```
pca: [[ 0.4404993   0.21970716  0.80453627  0.33228077]
 [-0.86612635 -0.11145744  0.48488059  0.0478873 ]]
```

In [73]:
```
data_cols = list(df_data2.columns)

pca_weight_strings = get_feature_names_from_weights(pca.components_, data_cols)

# create some pandas dataframes from the transformed outputs
df_pca = pd.DataFrame(X_pca,columns=[pca_weight_strings])

from matplotlib.pyplot import scatter

# scatter plot the output, with the names created from the weights
ax = scatter(X_pca[:,0], X_pca[:,1], c=y, s=(y+2)*10, cmap=cmap)
plt.xlabel(pca_weight_strings[0])
plt.ylabel(pca_weight_strings[1])
```

Out[73]:
```
Text(0, 0.5, '-0.87*chest_pain -0.11*exercise_a +0.48*ol +0.05*ST_ ')
```



This appears to be 4 distinct groups of data points such that there is no clear correlation between the 4 features selected. If this is the case, it would not make sense to reduce the dimensionality of these features. A Biplot will be shown below to confirm that these 4 features are in fact not correlated to PCA 1 and PCA 2.

In [74]:
```
pca = PCA(n_components=4) # Number of components for the # of features that I'm
pca.fit(df_data2)
biplot(pca,pd.DataFrame(df_data2,columns=data_cols),'Heart Biplot')
```

Similar to the scatter plot, the biplot above also indicates that there is no clear correlation between the four features above where dimensionality reduction may be applicable. In the first PCA of all 11 features, old peak and exercise angina were the two features that were on top of each other. To see if there is some sort of correlation to allow dimensionality reduction, two more PCA's will be used with a combination of these four features. Both PCA's will include old peak and exercise angina as those features were most closely related in the first PCA. Chest pain type will be added on to these two features in one of the PCA's and ST slope will be added to the other PCA.

Principal Component Analysis: old peak, exercise angina, and chest pain type

In [75]:
```
df_data3 = df_data2.copy() #Going to use df_data2 so not as much data needs to b
for col in ['ST_slope']: #Along with not copying over as much data, we only need
    if col in df_data3:
        del df_data3[col]

df_data3.head() #want to delete the range and target from this one
```

Out[75]:

	chest_pain_type	exercise_angina	oldpeak
0	2	0	0.0
1	3	0	1.0

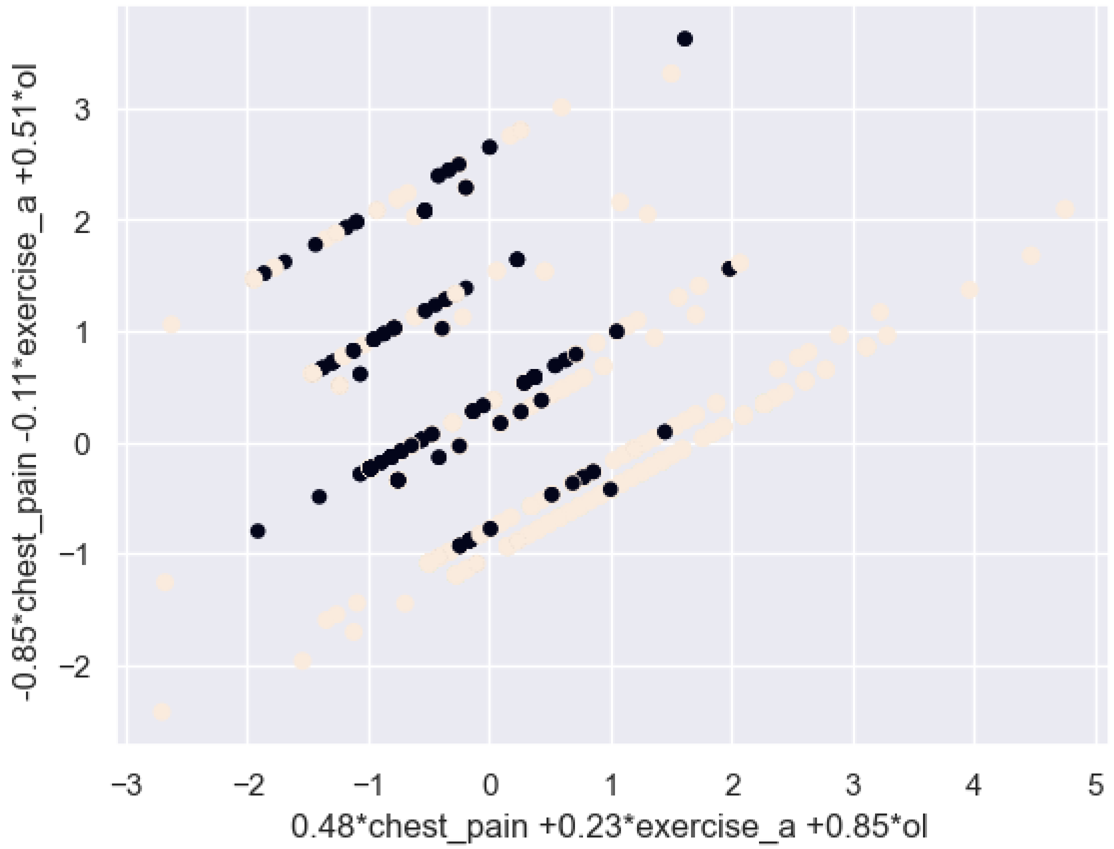
	chest_pain_type	exercise_angina	oldpeak
2	2	0	0.0
3	4	1	1.5
4	3	0	0.0

In [76]:
```
pca = PCA(n_components=2)
pca.fit(df_data3) # fit data and then transform it
X_pca = pca.transform(df_data3)

# print the components

print ('pca:', pca.components_)
```

```
pca: [[ 0.48147311  0.22652691  0.84668129]
 [-0.85173644 -0.10690253  0.5129492 ]]
```

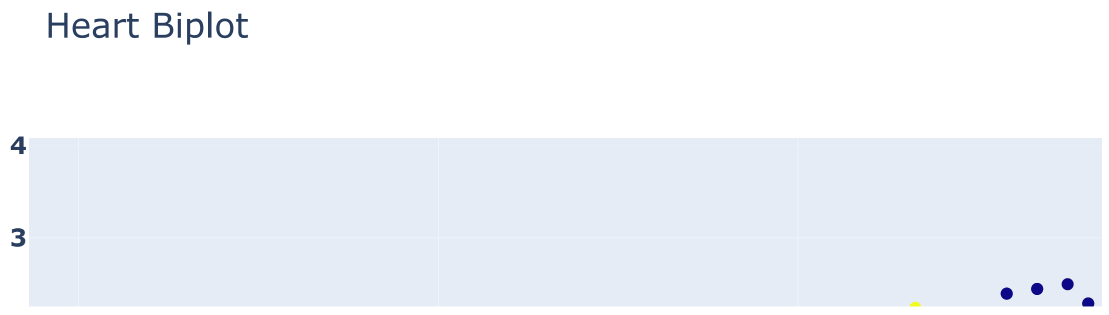
In [77]:
```
data_cols = list(df_data3.columns)

pca_weight_strings = get_feature_names_from_weights(pca.components_, data_cols)

# create some pandas dataframes from the transformed outputs
df_pca = pd.DataFrame(X_pca,columns=[pca_weight_strings])

from matplotlib.pyplot import scatter

# scatter plot the output, with the names created from the weights
ax = scatter(X_pca[:,0], X_pca[:,1], c=y, s=(y+2)*10, cmap=cmap)
plt.xlabel(pca_weight_strings[0])
plt.ylabel(pca_weight_strings[1])
```

Out[77]:
```
Text(0, 0.5, '-0.85*chest_pain -0.11*exercise_a +0.51*ol ')
```



Again, it appears that there are no distinct groupings in this scatter plot. If anything, it looks like the three features are now broken down into four seperate groups.

In [78]:
```
pca = PCA(n_components=3) # Number of components for the # of features that I'm
pca.fit(df_data3)
biplot(pca,pd.DataFrame(df_data3,columns=data_cols),'Heart Biplot')
```



Similar to the scatter plot, the biplot of old peak, exercise angina, and chest pain type indicated that there is no clear correlation between the features that would allow dimensionality reduction. Next up, we will look at old peak, exercise angina, and ST slope to determine if there is any applicable dimensionality reduction between these features.

Principal Component Analysis: old peak, exercise angina, and ST slope

In [79]:
```
df_data4 = df_data2.copy() #Going to use df_data2 so not as much data needs to b
for col in ['chest_pain_type']: #Along with not copying over as much data, we on
    if col in df_data4:
        del df_data4[col]

df_data4.head() #want to delete the range and target from this one
```

Out[79]:

	exercise_angina	oldpeak	ST_slope
0	0	0.0	1
1	0	1.0	2
2	0	0.0	1
3	1	1.5	2
4	0	0.0	1

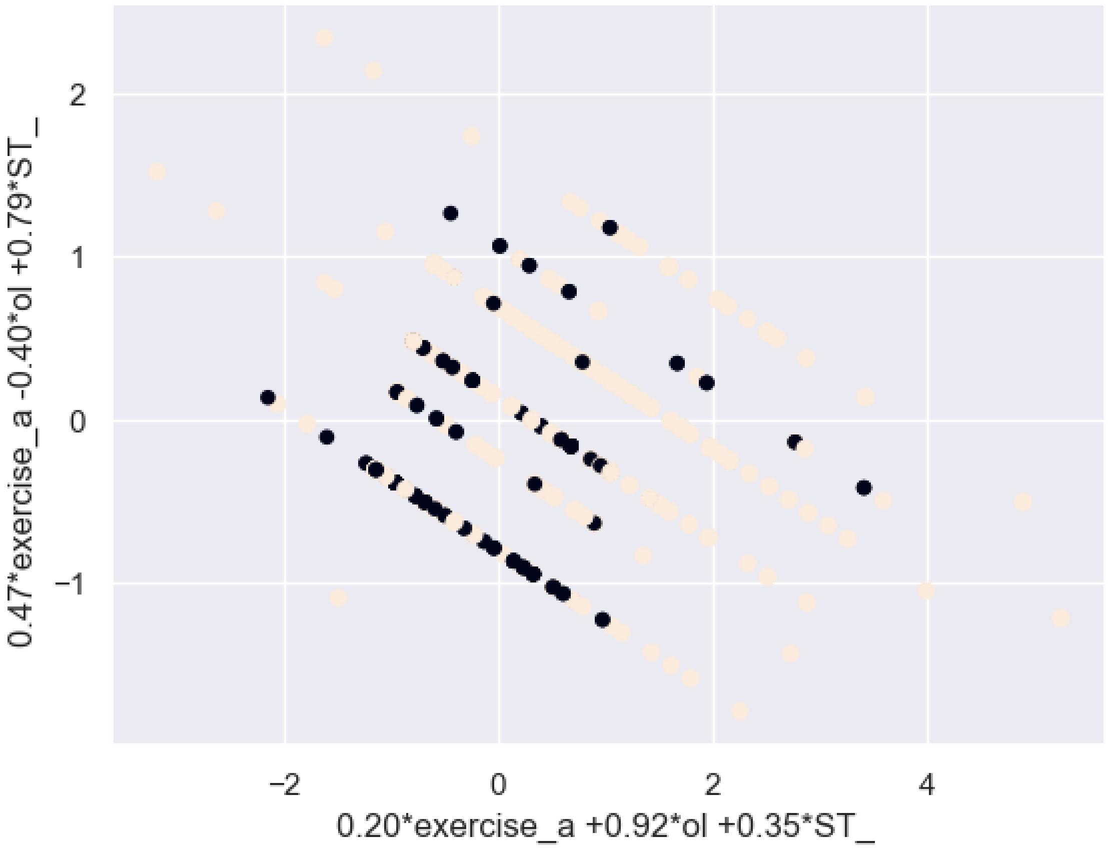
In [80]:
```
pca = PCA(n_components=2)
pca.fit(df_data4) # fit data and then transform it
X_pca = pca.transform(df_data4)

# print the components

print ('pca:', pca.components_)
```

```
pca: [[ 0.19750765  0.91637131  0.34821595]
 [ 0.47249516 -0.40021801  0.78522218]]
```

In [81]:
```
data_cols = list(df_data4.columns)

pca_weight_strings = get_feature_names_from_weights(pca.components_, data_cols)

# create some pandas dataframes from the transformed outputs
df_pca = pd.DataFrame(X_pca,columns=[pca_weight_strings])

from matplotlib.pyplot import scatter

# scatter plot the output, with the names created from the weights
ax = scatter(X_pca[:,0], X_pca[:,1], c=y, s=(y+2)*10, cmap=cmap)
```

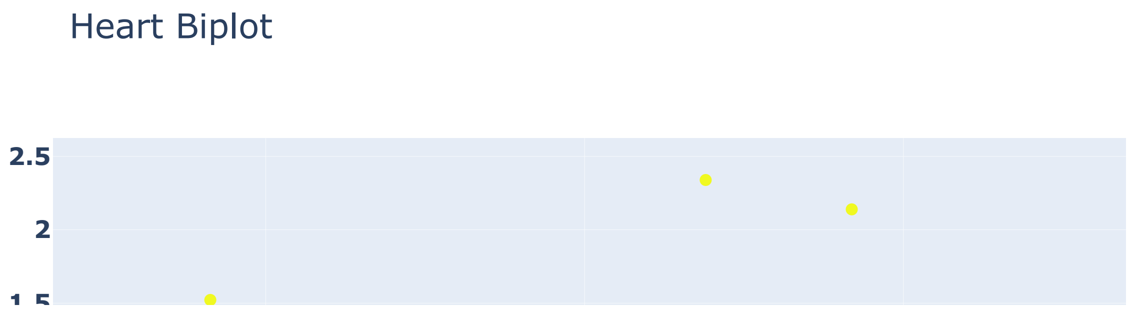
```
plt.xlabel(pca_weight_strings[0])
plt.ylabel(pca_weight_strings[1])
```

Out[81]:
```
Text(0, 0.5, '0.47*exercise_a -0.40*ol +0.79*ST_ ')
```



This scatter plot does not appear to have any distinct groupings which would reduce the dimensionality. A biplot will be drawn below to confirm this.

In [82]:
```
pca = PCA(n_components=3) # Number of components for the # of features that I'm
pca.fit(df_data4)
biplot(pca,pd.DataFrame(df_data4,columns=data_cols),'Heart Biplot')
```



In this biplot it looks like ST slope and exercise angina point in similar enough directions to be able to combine these features and reduce the dimensionality. The fact that ST slope and exercise angina are not pointing in the exact direction coupled with the previous PCA's not showing a direct correlation of these two features to the principal components leads us to believe that it may be too quick to assume that dimensionality reduction should be applied between these features. However, the correlation matrices drawn earlier demonstrated that ST slope and exercise angina were closely related to each other (0.39 correlation). It is worth looking further into these two features to see if dimensionality reduction can and should be applied.

```
In [ ]:
```