In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('data.csv')
print("The size of initial dataset: ", df.shape)
```

```
The size of initial dataset:  (569, 33)
```

In [3]:
```
df.isna().sum()
```

Out[3]:
```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32                569
dtype: int64
```

In [4]:
```
df_1 = df.dropna(axis=1)
print("The size of final dataset: ", df_1.shape)
```

```
The size of final dataset:  (569, 32)
```

In [5]:
```
y = df_1.diagnosis #feature labels
na_list = ['id','diagnosis']
X = df_1.drop(na_list,axis = 1 ) #data matrix
X.head()
```

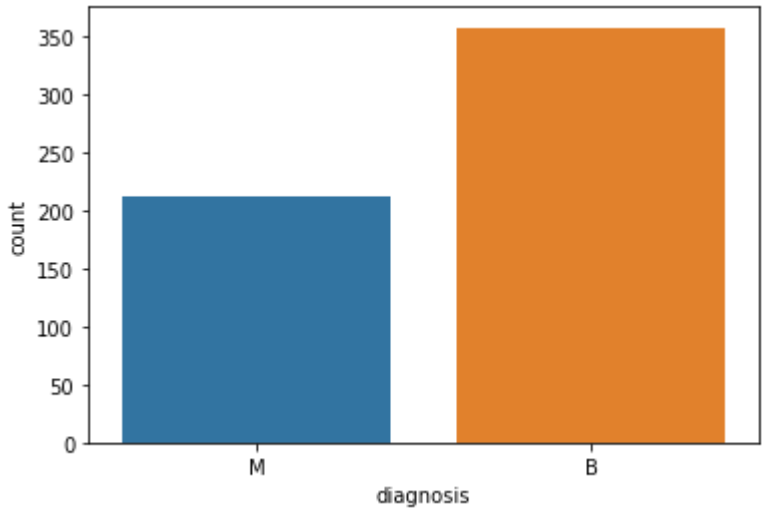
Out[5]:

| | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean | compactness_mean |
|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 |

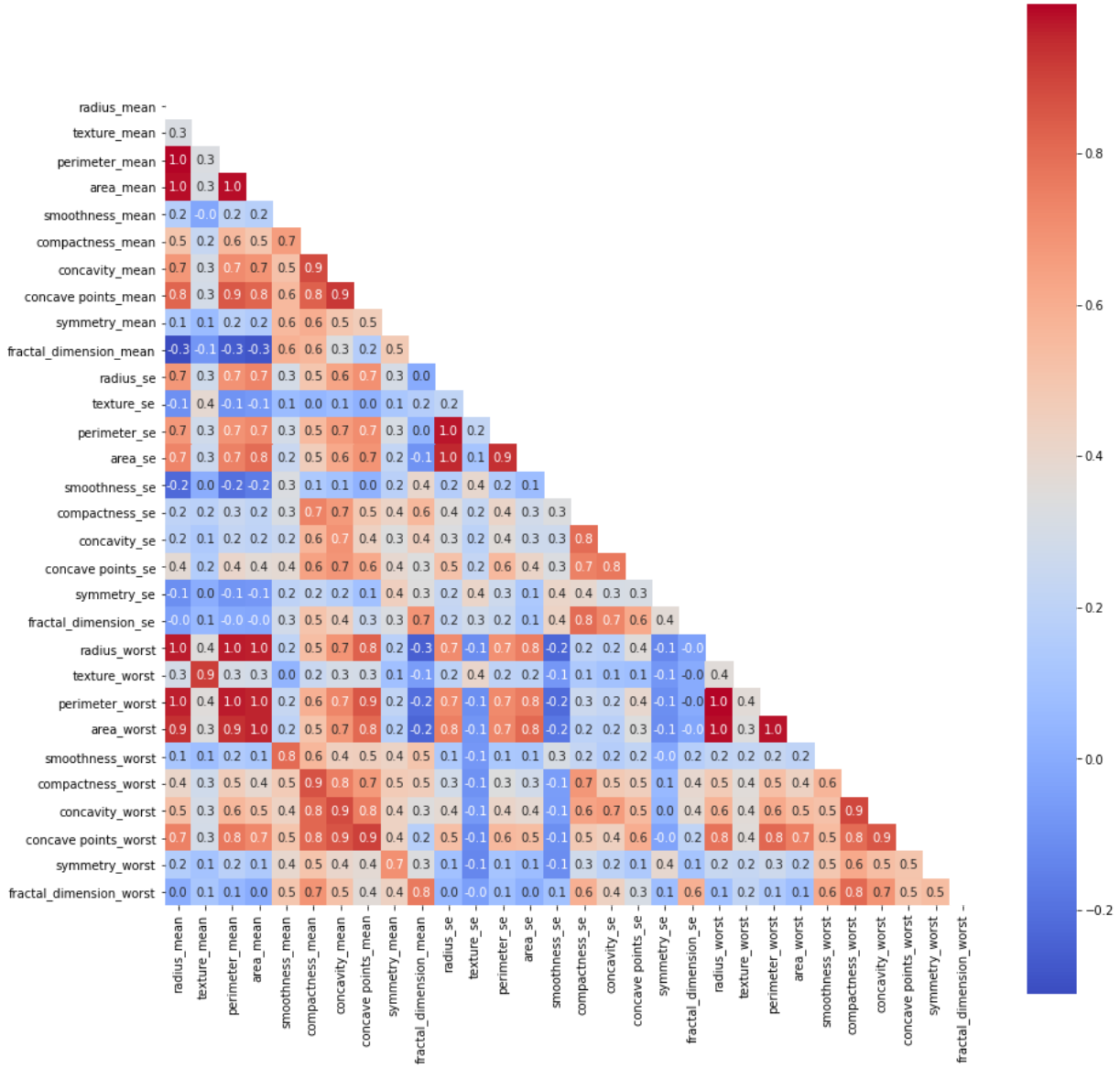
5 rows × 30 columns

In [6]:
```
#histograms showing the count of M and B
sns.countplot(x = df_1['diagnosis'], label="Count")
```

Out[6]: <AxesSubplot:xlabel='diagnosis', ylabel='count'>



In [7]:
```
plt.figure(figsize=(15,15))
upper_traing_matrix = np.triu(X.corr())
sns.heatmap(X.corr(), annot=True, square=True, fmt= '.1f', cmap='coolwarm',
plt.show()
```



In [8]:
```
# features which have correlation greater than 0.95 can be dropped
correlation_matrix = X.corr().abs()
mask = np.triu(np.ones(correlation_matrix.shape), k=1)
upper_triangular_matrix = correlation_matrix.where(mask.astype(np.bool))
na_features = [c for c in upper_triangular_matrix.columns if any(upper_trian
print("The features which can be dropped because of high correlation are")
print(na_features)
```

```
The features which can be dropped because of high correlation are
['perimeter_mean', 'area_mean', 'perimeter_se', 'area_se', 'radius_worst', 'per
imeter_worst', 'area_worst']
```

In [9]:
```
# feature selection
X = X.drop(X[na_features], axis=1)
X.head()
```

Out[9]:

| | radius_mean | texture_mean | smoothness_mean | compactness_mean | concavity_mean | concav points_mea |
|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 0.11840 | 0.27760 | 0.3001 | 0.147 |
| **1** | 20.57 | 17.77 | 0.08474 | 0.07864 | 0.0869 | 0.070 |
| **2** | 19.69 | 21.25 | 0.10960 | 0.15990 | 0.1974 | 0.127 |
| **3** | 11.42 | 20.38 | 0.14250 | 0.28390 | 0.2414 | 0.105 |
| **4** | 20.29 | 14.34 | 0.10030 | 0.13280 | 0.1980 | 0.104 |

5 rows × 23 columns

In [10]:
```
print("The original values for y are: ")
print(y)
from sklearn.preprocessing import LabelEncoder
Y_new = LabelEncoder()
y= Y_new.fit_transform(y)
print("The new values for y are: ")
print(y)
```

```
The original values for y are:
0      M
1      M
2      M
3      M
4      M
      ..
564    M
565    M
566    M
567    M
568    B
Name: diagnosis, Length: 569, dtype: object
The new values for y are:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1
 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0
 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1
 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0
 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0
 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0
 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 1
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0]
```

In [11]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40,
print("The shape of X_train is ", X_train.shape)
print("The shape of y_train is ", y_train.shape)
print("The shape of X_test is ", X_test.shape)
print("The shape of y_test is ", y_test.shape)
```

```
The shape of X_train is  (341, 23)
The shape of y_train is  (341,)
The shape of X_test is  (228, 23)
The shape of y_test is  (228,)
```

In [12]:
```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [13]:
```
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score
#Decision Tree
clf1 = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
scores_1 = cross_val_score(clf1, X, y, cv=5)
clf1.fit(X_train, y_train)
print('Training Accuracy for decision tree is ', clf1.score(X_train, y_train
print("CV accuracy is %0.2f and standard deviation is %0.2f" % (scores_1.mea
#Random Forest
clf2 = RandomForestClassifier(n_estimators = 8, criterion = 'entropy', rando
scores_2 = cross_val_score(clf2, X, y, cv=5)
clf2.fit(X_train, y_train)
print('Training Accuracy for random forest is ', clf2.score(X_train, y_train
print("CV accuracy is %0.2f and standard deviation is %0.2f" % (scores_2.mea
#Naive Bayes
clf3 = GaussianNB()
scores_3 = cross_val_score(clf3, X, y, cv=5)
clf3.fit(X_train, y_train)
print('Training Accuracy for naive bayes classifier is ', clf3.score(X_train
print("CV accuracy is %0.2f and standard deviation is %0.2f" % (scores_3.mea
```

```
Training Accuracy for decision tree is  1.0
CV accuracy is 0.93 and standard deviation is 0.01
Training Accuracy for random forest is  0.9970674486803519
CV accuracy is 0.95 and standard deviation is 0.01
Training Accuracy for naive bayes classifier is  0.9237536656891495
CV accuracy is 0.92 and standard deviation is 0.01
```

In [14]:
```
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
y_pred_1 = clf1.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred_1)
print("The confusion matrix for Decision Tree is: ")
print(conf_matrix)
print("Testing Accuracy for decision tree is ", accuracy_score(y_test, y_pre
```

```
The confusion matrix for Decision Tree is:
[[140   8]
 [ 10  70]]
Testing Accuracy for decision tree is  0.9210526315789473
```

In [15]:
```
y_pred_2 = clf2.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred_2)
print("The confusion matrix for Random Forest is: ")
print(conf_matrix)
print("Testing Accuracy for random forest is ", accuracy_score(y_test, y_pre
```

```
The confusion matrix for Random Forest is:
[[143   5]
 [ 11  69]]
Testing Accuracy for random forest is  0.9298245614035088
```

In [16]:
```
y_pred_3 = clf3.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred_3)
print("The confusion matrix for naive bayes classifier is: ")
print(conf_matrix)
print("Testing Accuracy for naive bayes classifier is ", accuracy_score(y_te
```

```
The confusion matrix for naive bayes classifier is:
[[141   7]
 [ 10  70]]
Testing Accuracy for naive bayes classifier is  0.9254385964912281
```

In [17]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
print("The shape of X_train is ", X_train.shape)
print("The shape of y_train is ", y_train.shape)
print("The shape of X_test is ", X_test.shape)
print("The shape of y_test is ", y_test.shape)
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
#Decision Tree
clf1 = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
scores_1 = cross_val_score(clf1, X, y, cv=5)
clf1.fit(X_train, y_train)
print('Training Accuracy for decision tree is ', clf1.score(X_train, y_train
print("CV accuracy is %0.2f and standard deviation is %0.2f" % (scores_1.mea
#Random Forest
clf2 = RandomForestClassifier(n_estimators = 8, criterion = 'entropy', rando
scores_2 = cross_val_score(clf2, X, y, cv=5)
clf2.fit(X_train, y_train)
print('Training Accuracy for random forest is ', clf2.score(X_train, y_train
print("CV accuracy is %0.2f and standard deviation is %0.2f" % (scores_2.mea
#Naive Bayes
clf3 = GaussianNB()
scores_3 = cross_val_score(clf3, X, y, cv=5)
clf3.fit(X_train, y_train)
print('Training Accuracy for naive bayes classifier is ', clf3.score(X_train
print("CV accuracy is %0.2f and standard deviation is %0.2f" % (scores_3.mea
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
y_pred_1 = clf1.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred_1)
print("The confusion matrix for Decision Tree is: ")
print(conf_matrix)
print("Testing Accuracy for decision tree is ", accuracy_score(y_test, y_pre
y_pred_2 = clf2.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred_2)
print("The confusion matrix for Random Forest is: ")
print(conf_matrix)
print("Testing Accuracy for random forest is ", accuracy_score(y_test, y_pre
y_pred_3 = clf3.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred_3)
print("The confusion matrix for naive bayes classifier is: ")
print(conf_matrix)
print("Testing Accuracy for naive bayes classifier is ", accuracy_score(y_te
```

```
The shape of X_train is  (426, 23)
The shape of y_train is  (426,)
The shape of X_test is  (143, 23)
The shape of y_test is  (143,)
Training Accuracy for decision tree is  1.0
CV accuracy is 0.93 and standard deviation is 0.01
Training Accuracy for random forest is  0.9929577464788732
```

```
CV accuracy is 0.95 and standard deviation is 0.01
Training Accuracy for naive bayes classifier is  0.9131455399061033
CV accuracy is 0.92 and standard deviation is 0.01
The confusion matrix for Decision Tree is:
[[84  5]
 [ 4 50]]
Testing Accuracy for decision tree is  0.9370629370629371
The confusion matrix for Random Forest is:
[[86  3]
 [ 4 50]]
Testing Accuracy for random forest is  0.951048951048951
The confusion matrix for naive bayes classifier is:
[[85  4]
 [ 4 50]]
Testing Accuracy for naive bayes classifier is  0.9440559440559441
```

In [18]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.80,
print("The shape of X_train is ", X_train.shape)
print("The shape of y_train is ", y_train.shape)
print("The shape of X_test is ", X_test.shape)
print("The shape of y_test is ", y_test.shape)
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
#Decision Tree
clf1 = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
scores_1 = cross_val_score(clf1, X, y, cv=5)
clf1.fit(X_train, y_train)
print('Training Accuracy for decision tree is ', clf1.score(X_train, y_train
print("CV accuracy is %0.2f and standard deviation is %0.2f" % (scores_1.mea
#Random Forest
clf2 = RandomForestClassifier(n_estimators = 8, criterion = 'entropy', rando
scores_2 = cross_val_score(clf2, X, y, cv=5)
clf2.fit(X_train, y_train)
print('Training Accuracy for random forest is ', clf2.score(X_train, y_train
print("CV accuracy is %0.2f and standard deviation is %0.2f" % (scores_2.mea
#Naive Bayes
clf3 = GaussianNB()
scores_3 = cross_val_score(clf3, X, y, cv=5)
clf3.fit(X_train, y_train)
print('Training Accuracy for naive bayes classifier is ', clf3.score(X_train
print("CV accuracy is %0.2f and standard deviation is %0.2f" % (scores_3.mea
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
y_pred_1 = clf1.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred_1)
print("The confusion matrix for Decision Tree is: ")
print(conf_matrix)
print("Testing Accuracy for decision tree is ", accuracy_score(y_test, y_pre
y_pred_2 = clf2.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred_2)
print("The confusion matrix for Random Forest is: ")
print(conf_matrix)
print("Testing Accuracy for random forest is ", accuracy_score(y_test, y_pre
y_pred_3 = clf3.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred_3)
print("The confusion matrix for naive bayes classifier is: ")
print(conf_matrix)
print("Testing Accuracy for naive bayes classifier is ", accuracy_score(y_te
```

```
The shape of X_train is  (113, 23)
The shape of y_train is  (113,)
The shape of X_test is  (456, 23)
The shape of y_test is  (456,)
Training Accuracy for decision tree is  1.0
CV accuracy is 0.93 and standard deviation is 0.01
```

```
Training Accuracy for random forest is  0.9911504424778761
CV accuracy is 0.95 and standard deviation is 0.01
Training Accuracy for naive bayes classifier is  0.911504424778761
CV accuracy is 0.92 and standard deviation is 0.01
The confusion matrix for Decision Tree is:
[[255  35]
 [ 18 148]]
Testing Accuracy for decision tree is  0.8837719298245614
The confusion matrix for Random Forest is:
[[274  16]
 [ 18 148]]
Testing Accuracy for random forest is  0.9254385964912281
The confusion matrix for naive bayes classifier is:
[[268  22]
 [ 17 149]]
Testing Accuracy for naive bayes classifier is  0.9144736842105263
```

In [19]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
print("For the 2nd experiment, the shape of X_train is ", X_train.shape)
print("For the 2nd experiment, the shape of y_train is ", y_train.shape)
print("For the 2nd experiment, the shape of X_test is ", X_test.shape)
print("For the 2nd experiment, the shape of y_test is ", y_test.shape)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.80,
print("For the 3rd experiment, the shape of X_train is ", X_train.shape)
print("For the 3rd experiment, the shape of y_train is ", y_train.shape)
print("For the 3rd experiment, the shape of X_test is ", X_test.shape)
print("For the 3rd experiment, the shape of y_test is ", y_test.shape)
```

```
For the 2nd experiment, the shape of X_train is  (426, 23)
For the 2nd experiment, the shape of y_train is  (426,)
For the 2nd experiment, the shape of X_test is  (143, 23)
For the 2nd experiment, the shape of y_test is  (143,)
For the 3rd experiment, the shape of X_train is  (113, 23)
For the 3rd experiment, the shape of y_train is  (113,)
For the 3rd experiment, the shape of X_test is  (456, 23)
For the 3rd experiment, the shape of y_test is  (456,)
```