In [9]:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.preprocessing import Imputer
from sklearn.linear_model import ElasticNet
import pandas as pd

df = pd.read_csv('project_data_021.csv')


for i in range(df.shape[0]):
  df.iloc[i, 11] = 1 if df.iloc[i, 11] < np.mean(df.iloc[:, 11]) else  0


y = df.iloc[:, 11].values

X = df.iloc[:, 0:11].values


# Setup the pipeline
steps = [('scaler', StandardScaler()),
         ('SVM', SVC())]

pipeline = Pipeline(steps)

# Specify the hyperparameter space
parameters = {'SVM__C':[1, 10, 100],
              'SVM__gamma':[0.1, 0.01]}

# Create train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=21)

# Instantiate the GridSearchCV object: cv
cv = GridSearchCV(pipeline, parameters, cv=3)

# Fit to the training set
cv.fit(X_train, y_train)

# Predict the labels of the test set: y_pred
y_pred = cv.predict(X_test)

# Compute and print metrics
print("Accuracy: {}".format(cv.score(X_test, y_test)))
print("\nClassification Report \n\n", classification_report(y_test, y_pred))
print("Tuned Model Parameters: {}".format(cv.best_params_))



# Setup the pipeline steps: steps
steps = [('imputation', Imputer(missing_values='NaN', strategy='mean', axis=0)),
         ('scaler', StandardScaler()),
         ('elasticnet', ElasticNet())]

# Create the pipeline: pipeline
pipeline = Pipeline(steps)

# Specify the hyperparameter space
parameters = {'elasticnet__l1_ratio':np.linspace(0,1,30)}

# Create train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)
```

```
# Create the GridSearchCV object: gm_cv
gm_cv = GridSearchCV(pipeline, parameters, cv=3)

# Fit to the training set
gm_cv.fit(X_train, y_train)

# Compute and print the metrics
r2 = gm_cv.score(X_test, y_test)
print("\nTuned ElasticNet Alpha: {}".format(gm_cv.best_params_))
print("\nTuned ElasticNet R squared: {}".format(r2))
```

```
Accuracy: 0.9428571428571428

Classification Report

              precision    recall  f1-score   support

           0       0.95      0.99      0.97       916
           1       0.70      0.22      0.33        64

   micro avg       0.94      0.94      0.94       980
   macro avg       0.82      0.61      0.65       980
weighted avg       0.93      0.94      0.93       980

Tuned Model Parameters: {'SVM__C': 10, 'SVM__gamma': 0.1}

Tuned ElasticNet Alpha: {'elasticnet__l1_ratio': 0.0}

Tuned ElasticNet R squared: 0.043924644721776485
```

In [ ]: