```
import pandas as pd

# read the dataset into a pandas dataframe object
df = pd.read_csv('/Users/aghazarian/wdbc.data', header=None)

#print some data to make sure the data was properly read into the dataframe
print(df.head(n=3))
rows,columns = df.shape

# get a sense of the size of the dataset
print('# of rows:', rows)
print('# of columns:', columns)
# first column is just an ID number, second column is the diagnostics - M for Malignant and B for Benign
# 3rd to 32nd column include the 30 features of the dataset

# assign the 30 features of the dataset and the target variable to separate Numpy arrays
# same result in the next 2 lines if we replace iloc with loc
X = df.iloc[:, 2:].values
Y = df.iloc[:, 1].values

# print X and Y by typing the variable names in console to make sure you properly extrated the
# independent and dependent variables. You can also directly look at the contents of these variables
# using the variabe explorer on the right

# transform the class labels from their original string representation (M and B) to integers

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
Y = le.fit_transform(Y)
# now Malignant is 1 and Benign is 0

# devide the dataset into 80% training and 20% separate test data
from sklearn.cross_validation import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=1)

# create a pipeline with the following steps chained to each other
#1. for optimal performance transform all feature values into the same scale - standardize the columns
#   before feeding them to the classifier
#2. Compress the initial 30 dimensional data into a lower 2 dimensional space using PCA
#3. Apply the logistic regression algorithm

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

# Define the steps of the pipeline, each step is a tuple:  a name and either a transformer or estimator object
classification_pipeline = Pipeline([('standard_scaler', StandardScaler()),
                                    ('pca', PCA(n_components=2)),
                                    ('classifier',LogisticRegression(random_state=1))])
# fit the training data into the model
classification_pipeline.fit(X_train,Y_train)

# compute the accuracy of the model on test data
accuracy = classification_pipeline.score(X_test, Y_test)
print('Test Accuracy: %.3f' % accuracy)
```

```
# instead of just a single test on test data, let's do a stratified k-fold cross validation on training data
from sklearn.cross_validation import cross_val_score
scores = cross_val_score(estimator=classification_pipeline, X=X_train, y=Y_train, cv=10, n_jobs=1)
import numpy as np
print('CV accuracy %.3f +/- %.3f' % (np.mean(scores), np.std(scores)))
```