Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Import Data

```
dataset = pd.read_csv('patient_data.csv')
# dataset.isnull().values.any() no missing data
print('dataset size: %d\n' % dataset.size)
print('Overview of data \n', dataset.head())
print('Class types:\n',dataset['Class'].value_counts())
# we don't need sample code number
x = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

```
     dataset size: 7513

     Overview of data
         Sample code number  Clump Thickness  ...  Mitoses  Class
     0             1000025                5  ...        1      2
     1             1002945                5  ...        1      2
     2             1015425                3  ...        1      2
     3             1016277                6  ...        1      2
     4             1017023                4  ...        1      2

     [5 rows x 11 columns]
     Class types:
      2    444
     4    239
     Name: Class, dtype: int64
```

We are going to assume that class 2 is normal and class 4 is cancerous

Split into train and test sets

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

Feature scaling

```
# although the variance of features in x is similar so it may not be required
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

Model creation

```
# Logistic regression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression().fit(x_train, y_train)
# KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2).fit(x_train, y_trair
# SVM
from sklearn.svm import SVC
svc = SVC(kernel='rbf').fit(x_train, y_train)
# Naive Bayes
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB().fit(x_train, y_train)
# Random forest
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(criterion='entropy').fit(x_train, y_train)
```

Confustion matrix and accuracy

```
from sklearn.metrics import confusion_matrix, accuracy_score
# logistic regression
print("Logistic Regression performance")
y_pred_lr = lr.predict(x_test)
print(confusion_matrix(y_test, y_pred_lr))
print(accuracy_score(y_test, y_pred_lr))

# knn
print("KNN performance")
y_pred_knn = knn.predict(x_test)
print(confusion_matrix(y_test, y_pred_knn))
print(accuracy_score(y_test, y_pred_knn))

# SVM
print("SVM performance")
y_pred_svc = svc.predict(x_test)
print(confusion_matrix(y_test, y_pred_svc))
print(accuracy_score(y_test, y_pred_svc))

# Naive Bayes
print("Naive Bayes performance")
y_pred_nb = nb.predict(x_test)
print(confusion_matrix(y_test, y_pred_nb))
print(accuracy_score(y_test, y_pred_nb))

# Random Forest
print("Random Forest performance")
y_pred_rf = rf.predict(x_test)
print(confusion_matrix(y_test, y_pred_rf))
print(accuracy_score(y_test, y_pred_rf))
```

```
     Logistic Regression performance
     [[84  3]
      [ 3 47]]
     0.9562043795620438
     KNN performance
     [[82  5]
      [ 2 48]]
     0.948905109489051
     SVM performance
     [[82  5]
      [ 1 49]]
     0.9562043795620438
     Naive Bayes performance
     [[80  7]
      [ 1 49]]
     0.9416058394160584
     Random Forest performance
     [[84  3]
      [ 1 49]]
     0.9708029197080292
```