In the model the building part, you can use the cancer dataset, which is a very famous multi-class classificati
image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei prese

The dataset comprises 30 features (mean radius, mean texture, mean perimeter, mean area, mean smoothne
concave points, mean symmetry, mean fractal dimension, radius error, texture error, perimeter error, area erro
concave points error, symmetry error, fractal dimension error, worst radius, worst texture, worst perimeter, wo
concavity, worst concave points, worst symmetry, and worst fractal dimension) and a target (type of cancer).
(harmful) and benign (not harmful). Here, you can build a model to classify the type of cancer. The dataset is
download it from the UCI Machine Learning Library.

```
import numpy as np #linear algebra library of Python
from sklearn import datasets
```

```
cancer = datasets.load_breast_cancer()
```

```
cancer.data.shape
```

```
print(cancer.data[0:5])
```

```
from sklearn.model_selection import train_test_split #method to split training and tes
X_train, X_test, y_train, y_test=train_test_split(cancer.data, cancer.target, test_siz
```

```
from sklearn.model_selection import GridSearchCV
from sklearn import svm
from sklearn.svm import SVC
```

```
params_grid = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],'C': [1, 10, 100, 1000]}, {'k
```

```
model = GridSearchCV(SVC(), params_grid, cv=5)
model.fit(X_train, y_train)
```

```
print('Best score for training data:',model.best_score_,"\n")

# View the best parameters for the model found using grid search
print('Best C:',model.best_estimator_.C,"\n")
print('Best Kernel:',model.best_estimator_.kernel,"\n")
print('Best Gamma:',model.best_estimator_.gamma,"\n")
```

```
model = SVC(C=100, kernel='linear')
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
```

```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
from sklearn.metrics import confusion_matrix
y_pred=model.predict(X_test)
confusion_matrix(y_test,y_pred)
```

Classifier not so good: true positives=104, true negatives=62, false positives=1 and false negatives=4. Recall TP/(TP+FP)=99%