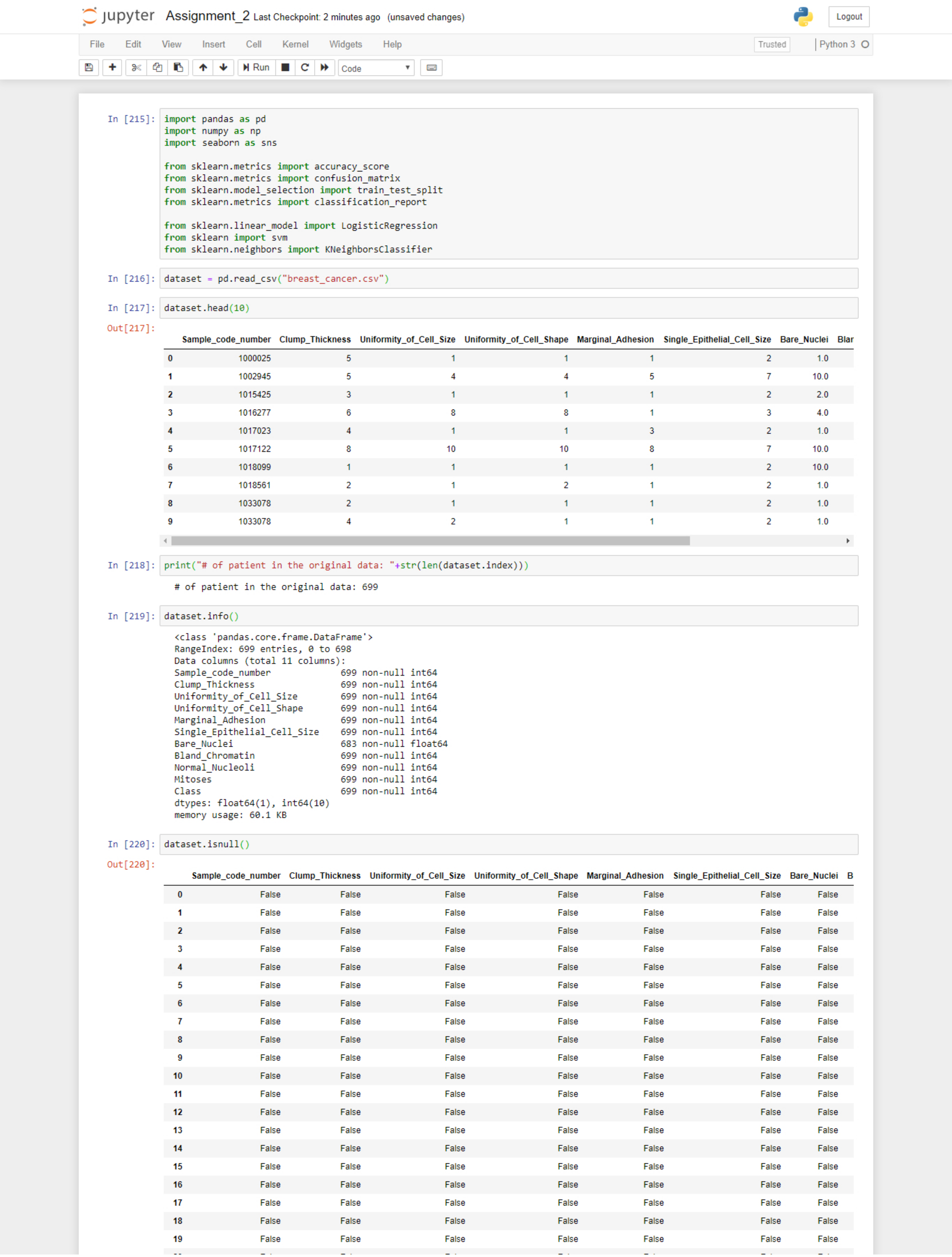


In [215]:
```
import pandas as pd
import numpy as np
import seaborn as sns

from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
```

In [216]:
```
dataset = pd.read_csv("breast_cancer.csv")
```

In [217]:
```
dataset.head(10)
```

Out[217]:

| | Sample_code_number | Clump_Thickness | Uniformity_of_Cell_Size | Uniformity_of_Cell_Shape | Marginal_Adhesion | Single_Epithelial_Cell_Size | Bare_Nuclei | Blar |
|---|---|---|---|---|---|---|---|---|
| **0** | 1000025 | 5 | 1 | 1 | 1 | 2 | 1.0 | |
| **1** | 1002945 | 5 | 4 | 4 | 5 | 7 | 10.0 | |
| **2** | 1015425 | 3 | 1 | 1 | 1 | 2 | 2.0 | |
| **3** | 1016277 | 6 | 8 | 8 | 1 | 3 | 4.0 | |
| **4** | 1017023 | 4 | 1 | 1 | 3 | 2 | 1.0 | |
| **5** | 1017122 | 8 | 10 | 10 | 8 | 7 | 10.0 | |
| **6** | 1018099 | 1 | 1 | 1 | 1 | 2 | 10.0 | |
| **7** | 1018561 | 2 | 1 | 2 | 1 | 2 | 1.0 | |
| **8** | 1033078 | 2 | 1 | 1 | 1 | 2 | 1.0 | |
| **9** | 1033078 | 4 | 2 | 1 | 1 | 2 | 1.0 | |

In [218]:
```
print("# of patient in the original data: "+str(len(dataset.index)))
```

```
# of patient in the original data: 699
```

In [219]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 11 columns):
Sample_code_number             699 non-null int64
Clump_Thickness                699 non-null int64
Uniformity_of_Cell_Size        699 non-null int64
Uniformity_of_Cell_Shape       699 non-null int64
Marginal_Adhesion              699 non-null int64
Single_Epithelial_Cell_Size    699 non-null int64
Bare_Nuclei                    683 non-null float64
Bland_Chromatin                699 non-null int64
Normal_Nucleoli                699 non-null int64
Mitoses                        699 non-null int64
Class                          699 non-null int64
dtypes: float64(1), int64(10)
memory usage: 60.1 KB
```

In [220]:
```
dataset.isnull()
```

Out[220]:

| | Sample_code_number | Clump_Thickness | Uniformity_of_Cell_Size | Uniformity_of_Cell_Shape | Marginal_Adhesion | Single_Epithelial_Cell_Size | Bare_Nuclei | B |
|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | |
| **1** | False | False | False | False | False | False | False | |
| **2** | False | False | False | False | False | False | False | |
| **3** | False | False | False | False | False | False | False | |
| **4** | False | False | False | False | False | False | False | |
| **5** | False | False | False | False | False | False | False | |
| **6** | False | False | False | False | False | False | False | |
| **7** | False | False | False | False | False | False | False | |
| **8** | False | False | False | False | False | False | False | |
| **9** | False | False | False | False | False | False | False | |
| **10** | False | False | False | False | False | False | False | |
| **11** | False | False | False | False | False | False | False | |
| **12** | False | False | False | False | False | False | False | |
| **13** | False | False | False | False | False | False | False | |
| **14** | False | False | False | False | False | False | False | |
| **15** | False | False | False | False | False | False | False | |
| **16** | False | False | False | False | False | False | False | |
| **17** | False | False | False | False | False | False | False | |
| **18** | False | False | False | False | False | False | False | |
| **19** | False | False | False | False | False | False | False | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 20 | False | False | False | False | False | False | False |
| 21 | False | False | False | False | False | False | False |
| 22 | False | False | False | False | False | False | False |
| 23 | False | False | False | False | False | False | True |
| 24 | False | False | False | False | False | False | False |
| 25 | False | False | False | False | False | False | False |
| 26 | False | False | False | False | False | False | False |
| 27 | False | False | False | False | False | False | False |
| 28 | False | False | False | False | False | False | False |
| 29 | False | False | False | False | False | False | False |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 669 | False | False | False | False | False | False | False |
| 670 | False | False | False | False | False | False | False |
| 671 | False | False | False | False | False | False | False |
| 672 | False | False | False | False | False | False | False |
| 673 | False | False | False | False | False | False | False |
| 674 | False | False | False | False | False | False | False |
| 675 | False | False | False | False | False | False | False |
| 676 | False | False | False | False | False | False | False |
| 677 | False | False | False | False | False | False | False |
| 678 | False | False | False | False | False | False | False |
| 679 | False | False | False | False | False | False | False |
| 680 | False | False | False | False | False | False | False |
| 681 | False | False | False | False | False | False | False |
| 682 | False | False | False | False | False | False | False |
| 683 | False | False | False | False | False | False | False |
| 684 | False | False | False | False | False | False | False |
| 685 | False | False | False | False | False | False | False |
| 686 | False | False | False | False | False | False | False |
| 687 | False | False | False | False | False | False | False |
| 688 | False | False | False | False | False | False | False |
| 689 | False | False | False | False | False | False | False |
| 690 | False | False | False | False | False | False | False |
| 691 | False | False | False | False | False | False | False |
| 692 | False | False | False | False | False | False | False |
| 693 | False | False | False | False | False | False | False |
| 694 | False | False | False | False | False | False | False |
| 695 | False | False | False | False | False | False | False |
| 696 | False | False | False | False | False | False | False |
| 697 | False | False | False | False | False | False | False |
| 698 | False | False | False | False | False | False | False |

699 rows × 11 columns

In [221]:
```
dataset.isnull().sum()
```

Out[221]:
```
Sample_code_number             0
Clump_Thickness                0
Uniformity_of_Cell_Size        0
Uniformity_of_Cell_Shape       0
Marginal_Adhesion              0
Single_Epithelial_Cell_Size    0
Bare_Nuclei                   16
Bland_Chromatin                0
Normal_Nucleoli                0
Mitoses                        0
Class                          0
dtype: int64
```

In [222]:
```
dataset.dropna(inplace=True)
```

In [223]:
```
dataset.isnull().sum()
```

Out[223]:
```
Sample_code_number             0
Clump_Thickness                0
Uniformity_of_Cell_Size        0
Uniformity_of_Cell_Shape       0
Marginal_Adhesion              0
Single_Epithelial_Cell_Size    0
Bare_Nuclei                    0
Bland_Chromatin                0
Normal_Nucleoli                0
Mitoses                        0
Class                          0
dtype: int64
```

In [224]:
```
dataset.drop("Sample_code_number", axis=1, inplace=True)
```

In [225]:
```
dataset.head(5)
```

Out[225]:

| Clump_Thickness | Uniformity_of_Cell_Size | Uniformity_of_Cell_Shape | Marginal_Adhesion | Single_Epithelial_Cell_Size | Bare_Nuclei | Bland_Chromatin | Normal_N |
|---|---|---|---|---|---|---|---|

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| **0** | 5 | 1 | 1 | 1 | 2 | 1.0 | 3 |
| **1** | 5 | 4 | 4 | 5 | 7 | 10.0 | 3 |
| **2** | 3 | 1 | 1 | 1 | 2 | 2.0 | 3 |
| **3** | 6 | 8 | 8 | 1 | 3 | 4.0 | 3 |
| **4** | 4 | 1 | 1 | 3 | 2 | 1.0 | 3 |

In [226]:

```
X = dataset.drop("Class", axis=1)
y = dataset["Class"]
```

In [227]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=4)
```

In [228]:

```
modelLRC = LogisticRegression()
modelSVMC  = svm.SVC(kernel='linear')
modelKNNC = KNeighborsClassifier(n_neighbors=3)
```

In [229]:

```
modelLRC.fit(X_train, y_train)
```

Out[229]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [230]:

```
modelSVMC.fit(X_train, y_train)
```

Out[230]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [231]:

```
modelKNNC.fit(X_train, y_train)
```

Out[231]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='uniform')
```

In [232]:

```
predictionLRC = modelLRC.predict(X_test)
predictionSVMC = modelSVMC.predict(X_test)
predictionKNNC = modelKNNC.predict(X_test)
```

In [233]:

```
print("Confusion Matrix for Logistic Regression: ")
confusion_matrix(y_test,predictionLRC)
```

```
Confusion Matrix for Logistic Regression:
```

Out[233]:

```
array([[125,   7],
       [  3,  70]], dtype=int64)
```

In [234]:

```
print("Confusion Matrix for Support Vector Machine: ")
confusion_matrix(y_test,predictionSVMC)
```

```
Confusion Matrix for Support Vector Machine:
```

Out[234]:

```
array([[122,  10],
       [  3,  70]], dtype=int64)
```

In [235]:

```
print("Confusion Matrix for KNN where (N=3): ")
confusion_matrix(y_test,predictionKNNC)
```

```
Confusion Matrix for KNN where (N=3):
```

Out[235]:

```
array([[126,   6],
       [  2,  71]], dtype=int64)
```

In [236]:

```
LR = accuracy_score(y_test,predictionLRC)*100
SVM = accuracy_score(y_test,predictionSVMC)*100
KNN = accuracy_score(y_test,predictionKNNC)*100
```

In [237]:

```
#SVM = accuracy_score(y_test,predictionSVMC)*100
```

In [238]:

```
#KNN = accuracy_score(y_test,predictionKNNC)*100
```

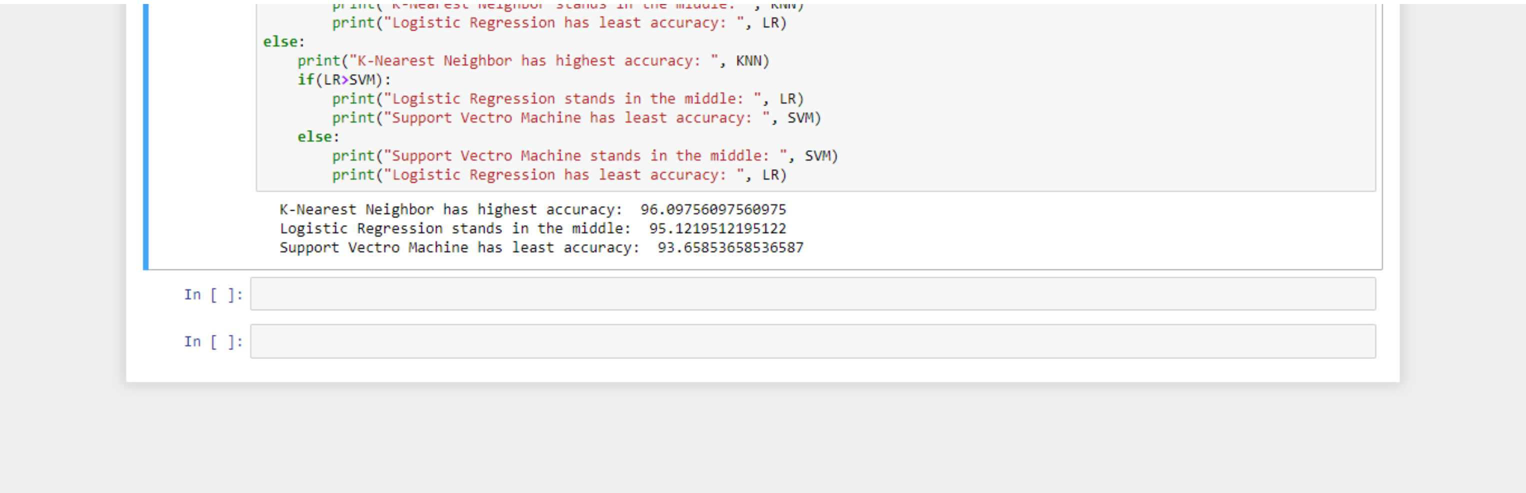
In [239]:

```
print("Accuracy Chart: ")
print("Logistic Regression: ",LR)
print("Support Vector Machine: ",SVM)
print("K-Nearest Neighbors (where k=3): ",KNN)
#sorted([LR, SVM, KNN], reverse=True)
```

```
Accuracy Chart:
Logistic Regression:  95.1219512195122
Support Vector Machine:  93.65853658536587
K-Nearest Neighbors (where k=3):  96.09756097560975
```

In [240]:

```
if((LR>SVM) and (LR>KNN)) :
    print("Logistic Regression has highest accuracy: ", LR)
    if(SVM>KNN):
        print("Support Vectro Machine stands in the middle: ", SVM)
        print("K-Nearest Neighbor has least accuracy: ", KNN)
    else:
        print("K-Nearest Neighbor stands in the middle: ", KNN)
        print("Support Vectro Machine has least accuracy: ", SVM)
elif((SVM>LR) and (SVM>KNN)):
    print("Support Vectro Machine has highest accuracy: ", SVM)
    if(LR>KNN):
        print("Logistic Regression stands in the middle: ", LR)
        print("K-Nearest Neighbor has least accuracy: ", KNN)
    else:
        print("K-Nearest Neighbor stands in the middle: ", KNN)
```



```
        print("K-Nearest Neighbor stands in the middle: ", KNN)
        print("Logistic Regression has least accuracy: ", LR)
else:
    print("K-Nearest Neighbor has highest accuracy: ", KNN)
    if(LR>SVM):
        print("Logistic Regression stands in the middle: ", LR)
        print("Support Vectro Machine has least accuracy: ", SVM)
    else:
        print("Support Vectro Machine stands in the middle: ", SVM)
        print("Logistic Regression has least accuracy: ", LR)
```

```
K-Nearest Neighbor has highest accuracy:  96.09756097560975
Logistic Regression stands in the middle:  95.1219512195122
Support Vectro Machine has least accuracy:  93.65853658536587
```