## Importing the libraries

In [72]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

## Importing the dataset

In [73]:
```
df = pd.read_csv('/Users/ashleshad/Downloads/breast_cancer.csv')
```

## EDA

In [74]:
```
df.head()
```

Out[74]:

| | Clump Thickness | Uniformity of Cell Size | Uniformity of Cell Shape | Marginal Adhesion | Single Epithelial Cell Size | Bare Nuclei | Bland Chromatin | Normal Nucleoli | Mitoses | Class |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 5 | 1 | 1 | 1 | 2 | 1 | 3 | 1 | 1 | 2 |
| **1** | 5 | 4 | 4 | 5 | 7 | 10 | 3 | 2 | 1 | 2 |
| **2** | 3 | 1 | 1 | 1 | 2 | 2 | 3 | 1 | 1 | 2 |
| **3** | 6 | 8 | 8 | 1 | 3 | 4 | 3 | 7 | 1 | 2 |
| **4** | 4 | 1 | 1 | 3 | 2 | 1 | 3 | 1 | 1 | 2 |

In [75]:
```
df.describe()
```

Out[75]:

| | Clump Thickness | Uniformity of Cell Size | Uniformity of Cell Shape | Marginal Adhesion | Single Epithelial Cell Size | Bare Nuclei | Bland Chromatin | Normal Nucleoli | Mitoses | Class |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 683.000000 | 683.000000 | 683.000000 | 683.000000 | 683.000000 | 683.000000 | 683.000000 | 683.000000 | 683.000000 | 683.000000 |
| **mean** | 4.442167 | 3.150805 | 3.215227 | 2.830161 | 3.234261 | 3.544656 | 3.445095 | 2.869693 | 1.603221 | 2.699854 |
| **std** | 2.820761 | 3.065145 | 2.988581 | 2.864562 | 2.223085 | 3.643857 | 2.449697 | 3.052666 | 1.732674 | 0.954592 |
| **min** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 2.000000 |
| **25%** | 2.000000 | 1.000000 | 1.000000 | 1.000000 | 2.000000 | 1.000000 | 2.000000 | 1.000000 | 1.000000 | 2.000000 |
| **50%** | 4.000000 | 1.000000 | 1.000000 | 1.000000 | 2.000000 | 1.000000 | 3.000000 | 1.000000 | 1.000000 | 2.000000 |
| **75%** | 6.000000 | 5.000000 | 5.000000 | 4.000000 | 4.000000 | 6.000000 | 5.000000 | 4.000000 | 1.000000 | 4.000000 |
| **max** | 10.000000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 | 4.000000 |

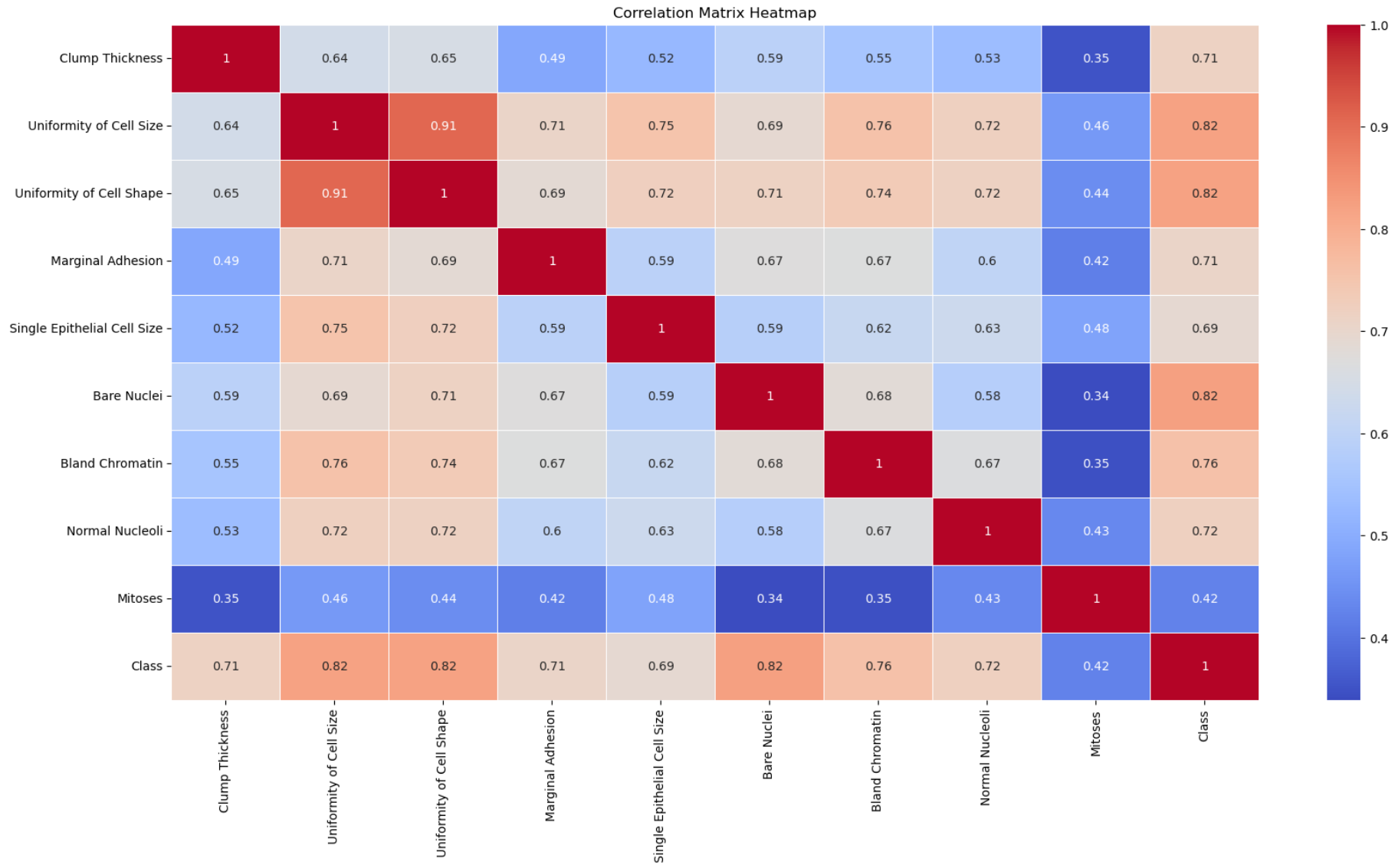
In [76]:
```
correlation_matrix = df.corr()

plt.figure(figsize=(20, 10))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=.5)

# Adding title
plt.title('Correlation Matrix Heatmap')

# Show the plot
plt.show()
```




In [77]:
```
df.isnull().sum()
```

Out[77]:
```
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
```

## Splitting into training and test set

In [78]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

In [79]:
```
X = df.drop('Class',1)
```

```
/var/folders/_j/tzw6wdvd1fv_1s_66tcw6my40000gn/T/ipykernel_3153/87674357.py:1: FutureWarning: In a future v
ersion of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
  X = df.drop('Class',1)
```

In [80]:
```
X
```

Out[80]:

| | Clump Thickness | Uniformity of Cell Size | Uniformity of Cell Shape | Marginal Adhesion | Single Epithelial Cell Size | Bare Nuclei | Bland Chromatin | Normal Nucleoli | Mitoses |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 5 | 1 | 1 | 1 | 2 | 1 | 3 | 1 | 1 |
| **1** | 5 | 4 | 4 | 5 | 7 | 10 | 3 | 2 | 1 |
| **2** | 3 | 1 | 1 | 1 | 2 | 2 | 3 | 1 | 1 |
| **3** | 6 | 8 | 8 | 1 | 3 | 4 | 3 | 7 | 1 |
| **4** | 4 | 1 | 1 | 3 | 2 | 1 | 3 | 1 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **678** | 3 | 1 | 1 | 1 | 3 | 2 | 1 | 1 | 1 |
| **679** | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 |
| **680** | 5 | 10 | 10 | 3 | 7 | 3 | 8 | 10 | 2 |
| **681** | 4 | 8 | 6 | 4 | 3 | 4 | 10 | 6 | 1 |
| **682** | 4 | 8 | 8 | 5 | 4 | 5 | 10 | 4 | 1 |

683 rows × 9 columns

In [81]:
```
Y = df['Class']
```

In [82]:
```
Y
```

Out[82]:
```
0      2
1      2
2      2
3      2
4      2
      ..
678    2
679    2
680    4
681    4
682    4
Name: Class, Length: 683, dtype: int64
```

## Performing by Logistic Regression

In [83]:
```
LR = LogisticRegression()
```

In [84]:
```
from sklearn.model_selection import train_test_split
```

In [105]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.20, random_state=4)
```

In [106]:
```
LR.fit(X_train, y_train)
```

Out[106]:
```
LogisticRegression
LogisticRegression()
```

In [107]:
```
model_predict=LR.predict(X_test)
```

In [108]:
```
LR.score(X_train,y_train)
```

Out[108]:
```
0.9761904761904762
```

In [109]:
```
LR.score(X_test,y_test)
```

Out[109]:
```
0.9708029197080292
```

In [110]:
```
print(classification_report(y_test, model_predict))
```

```
              precision    recall  f1-score   support

           2       1.00      0.96      0.98        90
           4       0.92      1.00      0.96        47

    accuracy                           0.97       137
   macro avg       0.96      0.98      0.97       137
weighted avg       0.97      0.97      0.97       137
```

## Performing by Decision Tree Classifier

In [111]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [112]:
```
DTC = DecisionTreeClassifier(criterion='entropy', max_depth=2)
```

In [113]:
```
DTC.fit(X_train,y_train)
```

Out[113]:
```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=2)
```

In [114]:
```
model_pred=DTC.predict(X_test)
```

In [115]:
```
model_pred
```

Out[115]:
```
array([2, 4, 2, 4, 2, 2, 4, 2, 4, 2, 2, 4, 4, 4, 4, 2, 2, 2, 2, 2, 4, 2,
       4, 4, 4, 4, 2, 2, 4, 4, 4, 4, 4, 2, 2, 4, 4, 2, 2, 2, 4, 4, 2, 4,
       4, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 4, 2, 2, 2, 2, 4, 4, 2, 4, 4,
       4, 2, 4, 2, 4, 4, 2, 2, 2, 4, 2, 2, 4, 4, 2, 2, 2, 2, 4, 4, 2, 4,
       2, 2, 2, 4, 2, 2, 2, 4, 2, 2, 2, 4, 2, 4, 2, 2, 4, 2, 4, 2, 2, 4,
       2, 2, 4, 2, 4, 2, 2, 2, 4, 4, 4, 4, 4, 2, 2, 4, 4, 2, 4, 4, 4, 2,
       2, 2, 2, 2, 4])
```

In [116]:
```
print(classification_report(y_test,model_pred))
```

```
              precision    recall  f1-score   support

           2       0.99      0.84      0.91        90
           4       0.77      0.98      0.86        47

    accuracy                           0.89       137
   macro avg       0.88      0.91      0.88       137
weighted avg       0.91      0.89      0.89       137
```

## Performing by Random Forest Classifier

In [117]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [118]:
```
RFC=RandomForestClassifier(n_estimators=100, random_state=4)
```

In [119]:
```
RFC.fit(X_train,y_train)
```

Out[119]:
```
RandomForestClassifier
RandomForestClassifier(random_state=4)
```

In [120]:
```
y_pred = RFC.predict(X_test)
```

In [121]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, classification_report
```

In [122]:
```
confusion_matrix(y_test,y_pred)
```

Out[122]:
```
array([[86,  4],
       [ 0, 47]])
```

In [123]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           2       1.00      0.96      0.98        90
           4       0.92      1.00      0.96        47

    accuracy                           0.97       137
   macro avg       0.96      0.98      0.97       137
weighted avg       0.97      0.97      0.97       137
```

**Logistic Regression performed better than the other two.**