In [1]:
```
#Importing necessary liberies needed to analyze the data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#At this point we are loading the already downloaded dataset to a dataframe
data = pd.read_csv(r"C:\Users\hp pc\Desktop\ik\new job\mrs temitope\breast_cance
```

In [3]:
```
data.shape
```

Out[3]: (699, 11)

In [4]:
```
data.head()
```

Out[4]:

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitose
0	1000025	5	1	1	1	2	1	3	1	
1	1002945	5	4	4	5	7	10	3	2	
2	1015425	3	1	1	1	2	2	3	1	
3	1016277	6	8	8	1	3	4	3	7	
4	1017023	4	1	1	3	2	1	3	1	

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Sample code number           699 non-null    int64
 1   Clump Thickness              699 non-null    int64
 2   Uniformity of Cell Size      699 non-null    int64
 3   Uniformity of Cell Shape     699 non-null    int64
 4   Marginal Adhesion            699 non-null    int64
 5   Single Epithelial Cell Size  699 non-null    int64
 6   Bare Nuclei                  699 non-null    object
 7   Bland Chromatin              699 non-null    int64
 8   Normal Nucleoli              699 non-null    int64
 9   Mitoses                      699 non-null    int64
 10   Class                       699 non-null    int64
dtypes: int64(10), object(1)
memory usage: 60.2+ KB
```

In [6]:
```
# change the dataset of the column "Bare Nuclei" to int64
data['Bare Nuclei'] = pd.to_numeric(data['Bare Nuclei'], errors='coerce')
```

In [7]:
```
1 data.isnull().sum()
```

Out[7]:
```
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 [8]:
```
1 mean = data['Bare Nuclei'].mean()
2 data['Bare Nuclei'].fillna(mean, inplace=True)
```

In [9]:
```
1 data.isnull().sum()
```

Out[9]:
```
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 [10]:
```
1 data.describe()
```

Out[10]:

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Blar Chromat
count	6.990000e+02	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.0000(
mean	1.071704e+06	4.417740	3.134478	3.207439	2.806867	3.216023	3.544656	3.4377(
std	6.170957e+05	2.815741	3.051459	2.971913	2.855379	2.214300	3.601852	2.4383(
min	6.163400e+04	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000(
25%	8.706885e+05	2.000000	1.000000	1.000000	1.000000	2.000000	1.000000	2.0000(
50%	1.171710e+06	4.000000	1.000000	1.000000	1.000000	2.000000	1.000000	3.0000(
75%	1.238298e+06	6.000000	5.000000	5.000000	4.000000	4.000000	5.000000	5.0000(
max	1.345435e+07	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.0000(

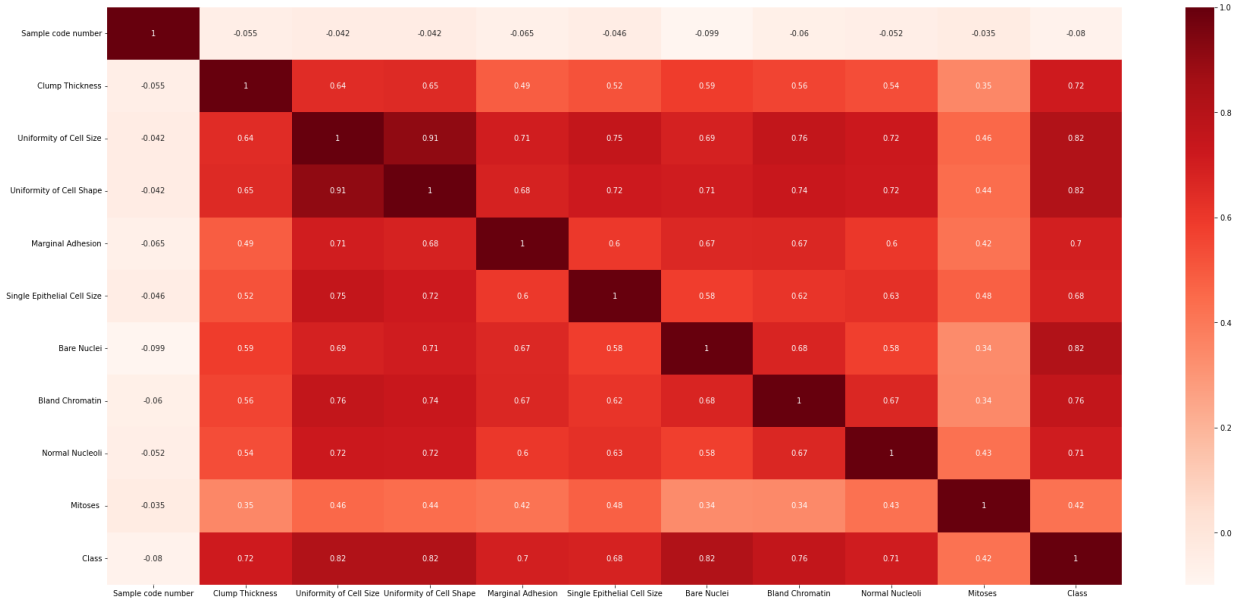
In [11]:
```
1 data.corr()
```

Out[11]:

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	N
Sample code number	1.000000	-0.055308	-0.041603	-0.041576	-0.064878	-0.045528	-0.098668	-0.060051	-0.
Clump Thickness	-0.055308	1.000000	0.644913	0.654589	0.486356	0.521816	0.587300	0.558428	0.
Uniformity of Cell Size	-0.041603	0.644913	1.000000	0.906882	0.705582	0.751799	0.686801	0.755721	0.
Uniformity of Cell Shape	-0.041576	0.654589	0.906882	1.000000	0.683079	0.719668	0.709606	0.735948	0.
Marginal Adhesion	-0.064878	0.486356	0.705582	0.683079	1.000000	0.599599	0.665049	0.666715	0.
Single Epithelial Cell Size	-0.045528	0.521816	0.751799	0.719668	0.599599	1.000000	0.581261	0.616102	0.
Bare Nuclei	-0.098668	0.587300	0.686801	0.709606	0.665049	0.581261	1.000000	0.675896	0.
Bland Chromatin	-0.060051	0.558428	0.755721	0.735948	0.666715	0.616102	0.675896	1.000000	0.
Normal Nucleoli	-0.052072	0.535835	0.722865	0.719446	0.603352	0.628881	0.577362	0.665878	1.
Mitoses	-0.034901	0.350034	0.458693	0.438911	0.417633	0.479101	0.338740	0.344169	0.
Class	-0.080226	0.716001	0.817904	0.818934	0.696800	0.682785	0.816050	0.756616	0.

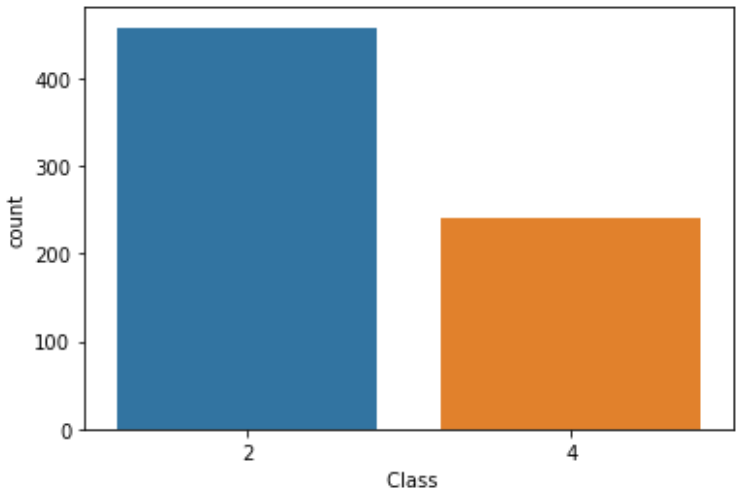
In [12]:
```
1 plt.figure(figsize=(30,14))
2 sns.heatmap(data.corr(),annot=True,cmap='Reds')
```

Out[12]: <AxesSubplot:>



In [13]:
```
1 sns.countplot(data=data, x=' Class')
```

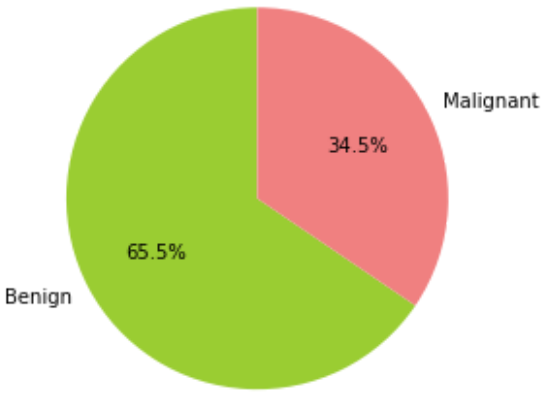
Out[13]:
```
<AxesSubplot:xlabel=' Class', ylabel='count'>
```



In [14]:
```
1 data.groupby(' Class').count()['Sample code number']
```

Out[14]:
```
 Class
2    458
4    241
Name: Sample code number, dtype: int64
```

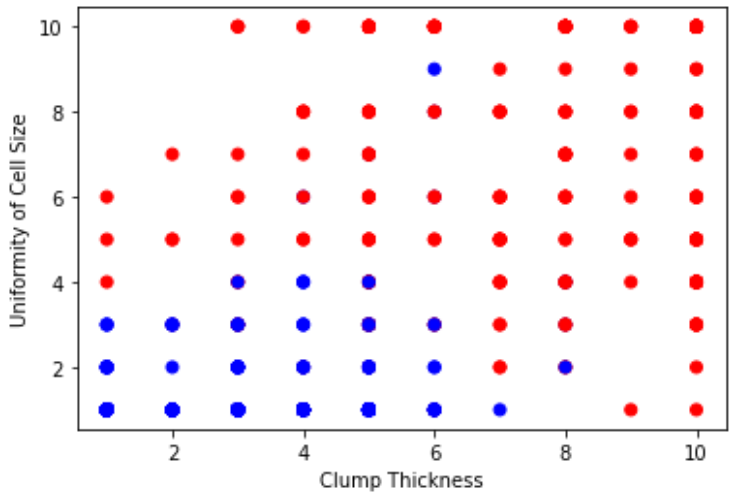
In [15]:
```
1 labels = ['Benign', 'Malignant']
2 sizes = [len(data[data[' Class'] == 2]), len(data[data[' Class'] == 4])]
3 colors = ['yellowgreen', 'lightcoral']
4
5 plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)
6 plt.axis('equal')
7 plt.show()
```



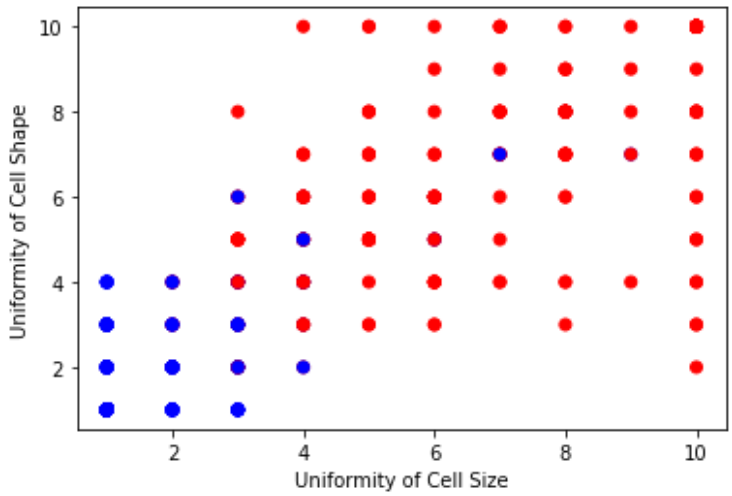
In [16]:
```
1 data.columns
```

Out[16]:
```
Index(['Sample code number', 'Clump Thickness', 'Uniformity of Cell Size',
       'Uniformity of Cell Shape', 'Marginal Adhesion',
       'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
       'Normal Nucleoli', 'Mitoses ', ' Class'],
      dtype='object')
```

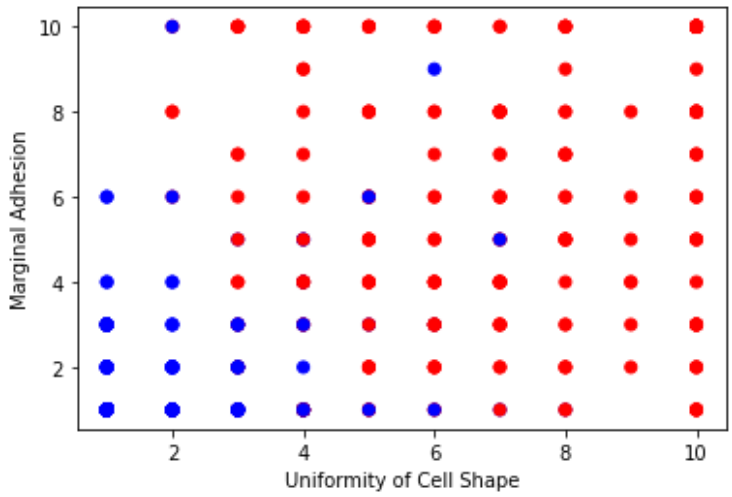
In [17]:
```
colors = {2:'blue', 4:'red'}
plt.scatter(data['Clump Thickness'], data['Uniformity of Cell Size'], c=data[' 
plt.xlabel('Clump Thickness')
plt.ylabel('Uniformity of Cell Size')
plt.show()
```



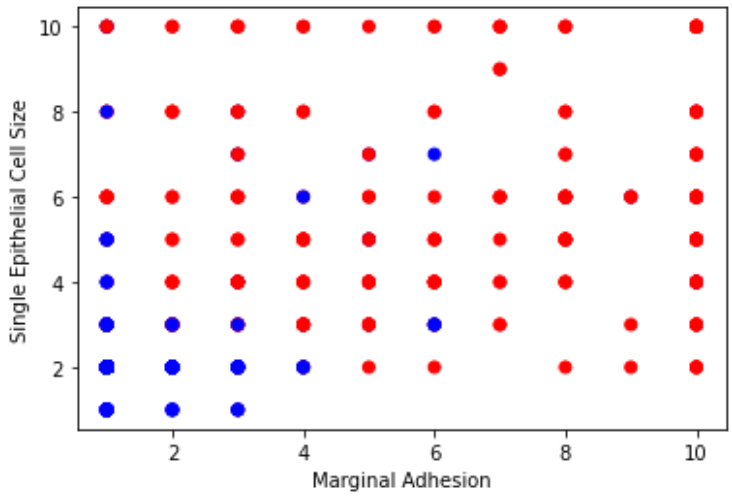
In [18]:
```
colors = {2:'blue', 4:'red'}
plt.scatter(data['Uniformity of Cell Size'], data['Uniformity of Cell Shape'], 
plt.xlabel('Uniformity of Cell Size')
plt.ylabel('Uniformity of Cell Shape')
plt.show()
```



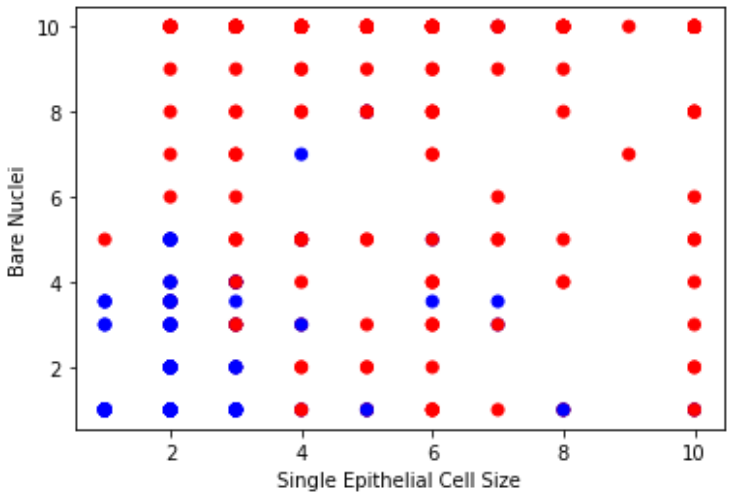
In [19]:
```
colors = {2:'blue', 4:'red'}
plt.scatter(data['Uniformity of Cell Shape'], data['Marginal Adhesion'], c=data
plt.xlabel('Uniformity of Cell Shape')
plt.ylabel('Marginal Adhesion')
plt.show()
```



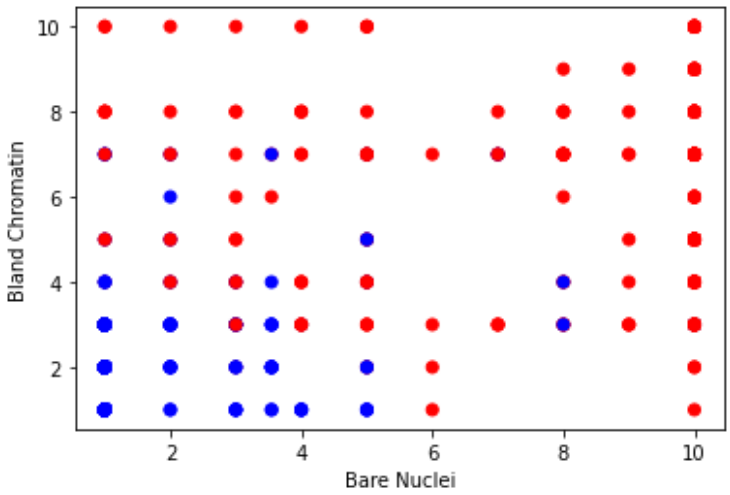
In [20]:
```
colors = {2:'blue', 4:'red'}
plt.scatter(data['Marginal Adhesion'], data['Single Epithelial Cell Size'], c=da
plt.xlabel('Marginal Adhesion')
plt.ylabel('Single Epithelial Cell Size')
plt.show()
```



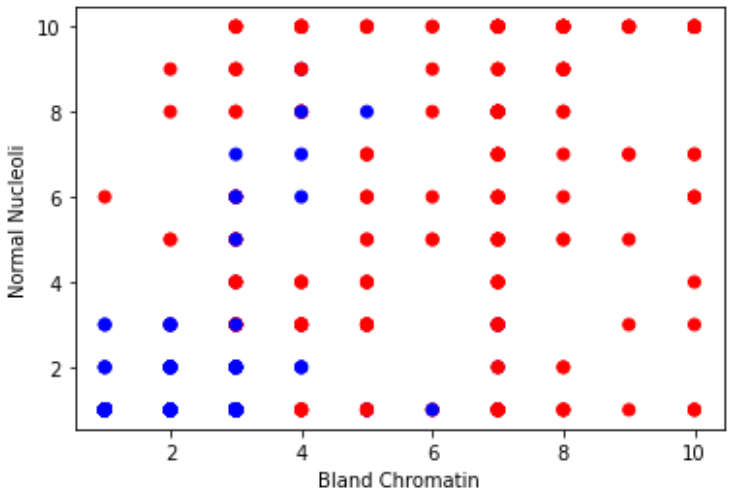
In [21]:
```
colors = {2:'blue', 4:'red'}
plt.scatter(data['Single Epithelial Cell Size'], data['Bare Nuclei'], c=data[' C
plt.xlabel('Single Epithelial Cell Size')
plt.ylabel('Bare Nuclei')
plt.show()
```



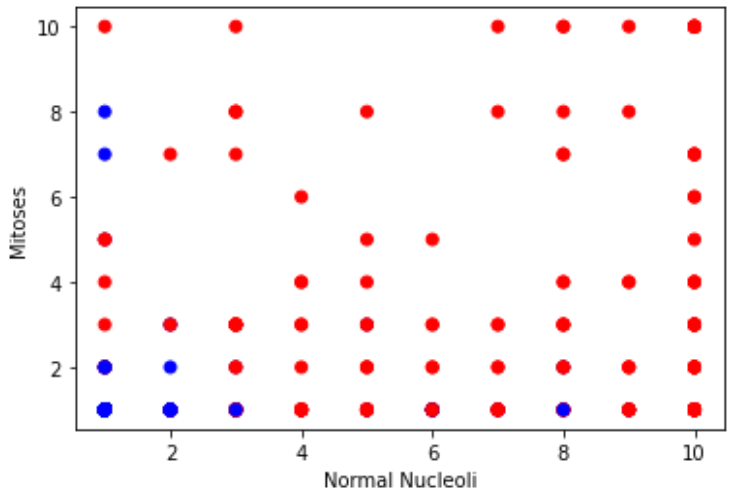
In [22]:
```
colors = {2:'blue', 4:'red'}
plt.scatter(data['Bare Nuclei'], data['Bland Chromatin'], c=data[' Class'].map(c
plt.xlabel('Bare Nuclei')
plt.ylabel('Bland Chromatin')
plt.show()
```



In [23]:
```
colors = {2:'blue', 4:'red'}
plt.scatter(data['Bland Chromatin'], data['Normal Nucleoli'], c=data[' Class'].
plt.xlabel('Bland Chromatin')
plt.ylabel('Normal Nucleoli')
plt.show()
```



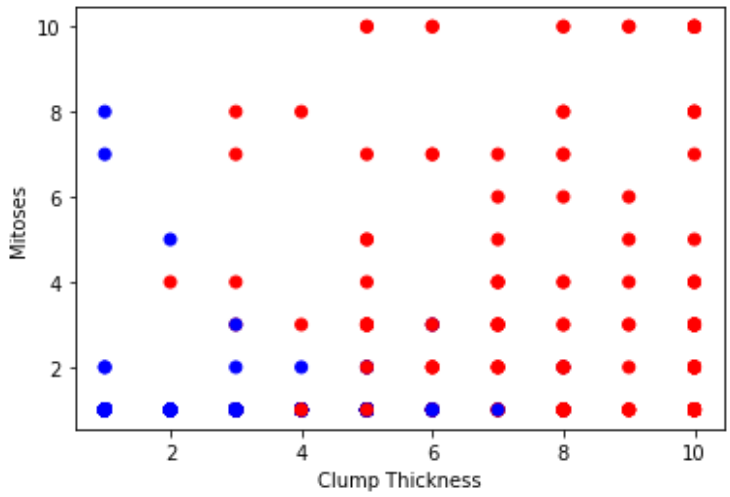
In [24]:
```
colors = {2:'blue', 4:'red'}
plt.scatter(data['Normal Nucleoli'], data['Mitoses '], c=data[' Class'].map(col
plt.xlabel('Normal Nucleoli')
plt.ylabel('Mitoses')
plt.show()
```



In [25]:
```
data.columns
```

Out[25]:
```
Index(['Sample code number', 'Clump Thickness', 'Uniformity of Cell Size',
       'Uniformity of Cell Shape', 'Marginal Adhesion',
       'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
       'Normal Nucleoli', 'Mitoses ', ' Class'],
      dtype='object')
```

In [26]:
```
colors = {2:'blue', 4:'red'}
plt.scatter(data['Clump Thickness'], data['Mitoses '], c=data[' Class'].map(col
plt.xlabel('Clump Thickness')
plt.ylabel('Mitoses')
plt.show()
```

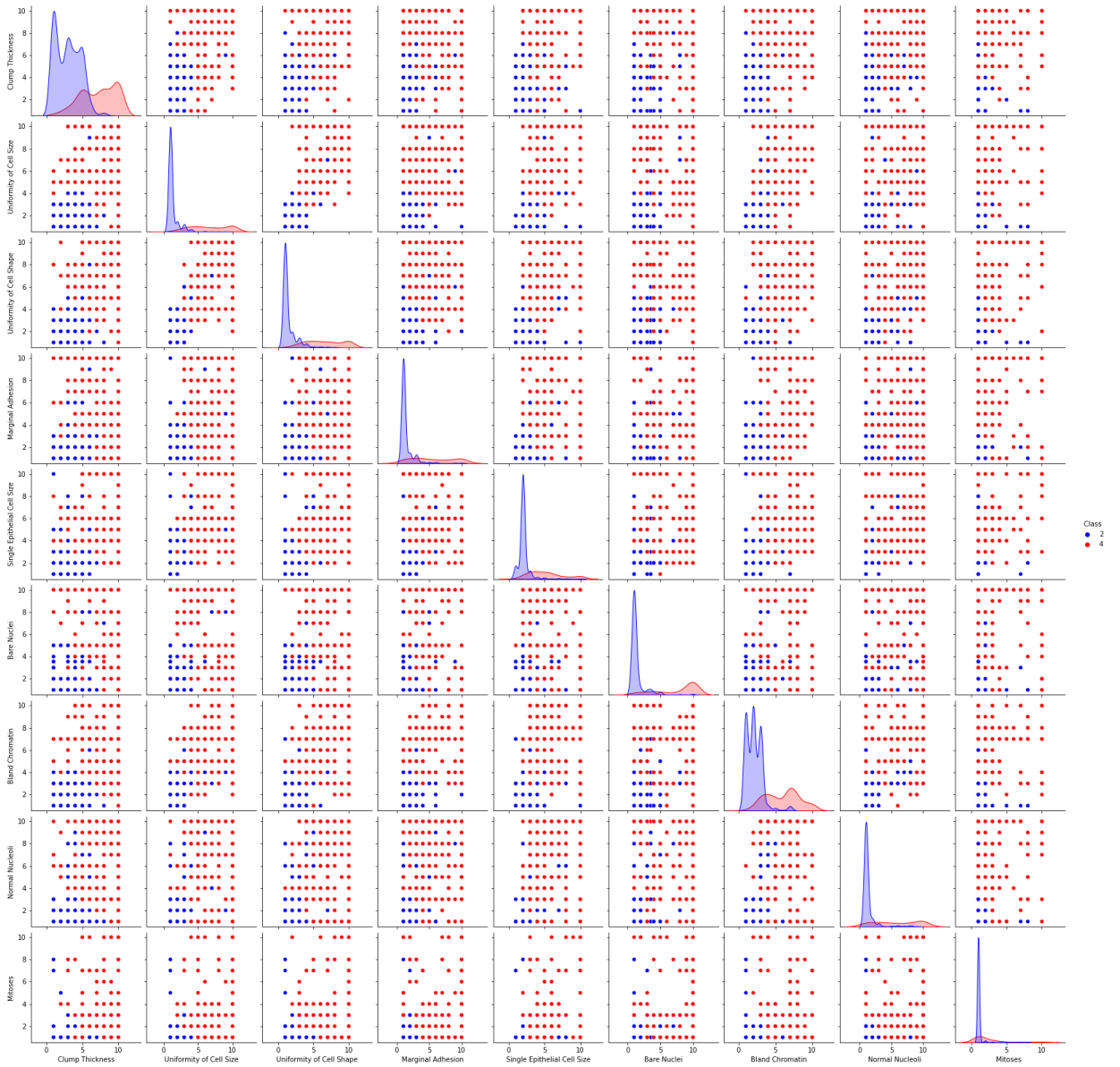


In [27]:
```
data_copy= data.copy()
data_copy.drop('Sample code number',axis=1, inplace=True)
data_copy.columns

```

Out[27]:
```
Index(['Clump Thickness', 'Uniformity of Cell Size',
       'Uniformity of Cell Shape', 'Marginal Adhesion',
       'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
       'Normal Nucleoli', 'Mitoses ', ' Class'],
      dtype='object')
```

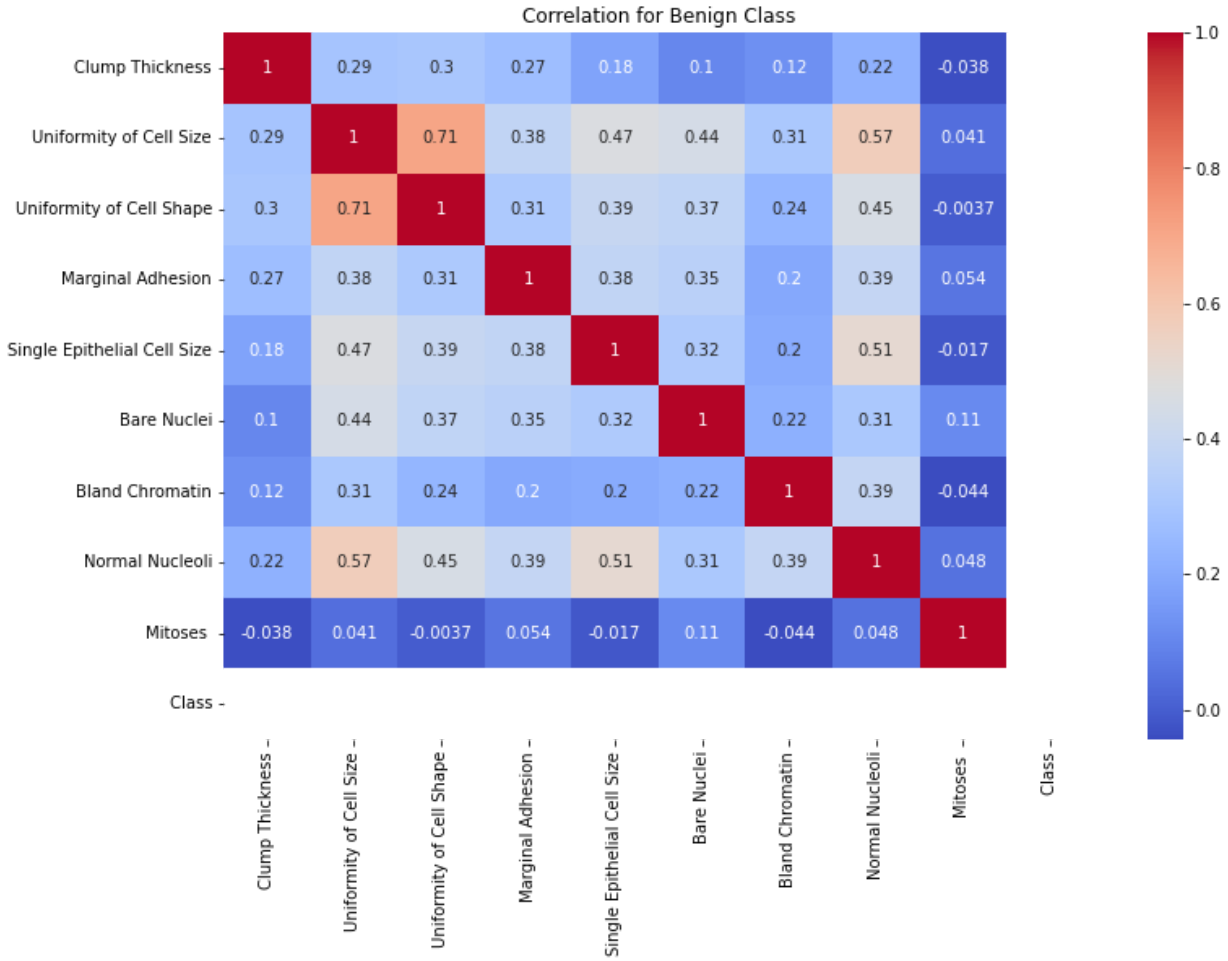
In [28]:

```
import seaborn as sns
my_palette = {2: 'blue', 4: 'red'}

# Create the pairplot
sns.pairplot(data_copy, hue=' Class', palette=my_palette)

# Show the plot
plt.show()

```

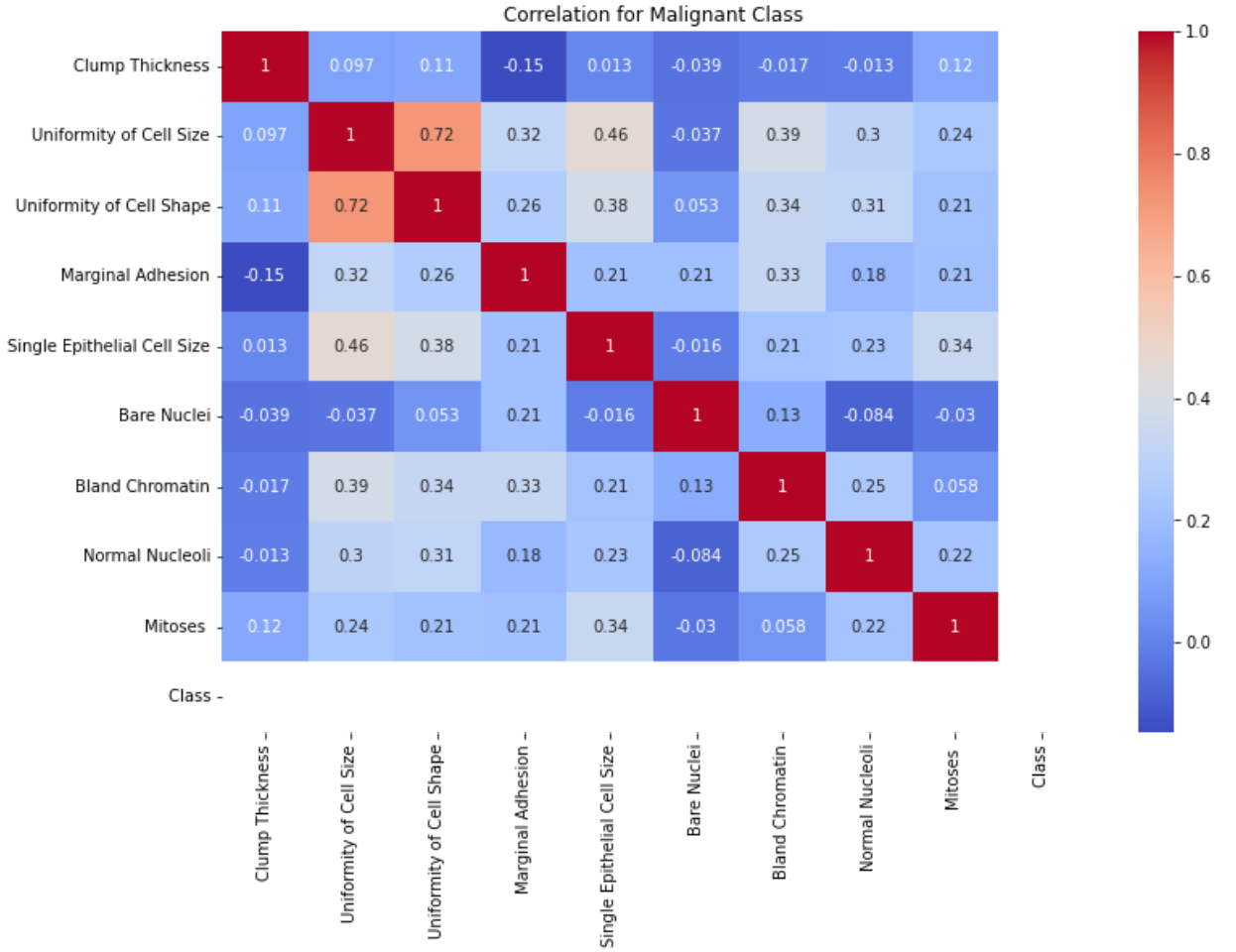


In [29]:

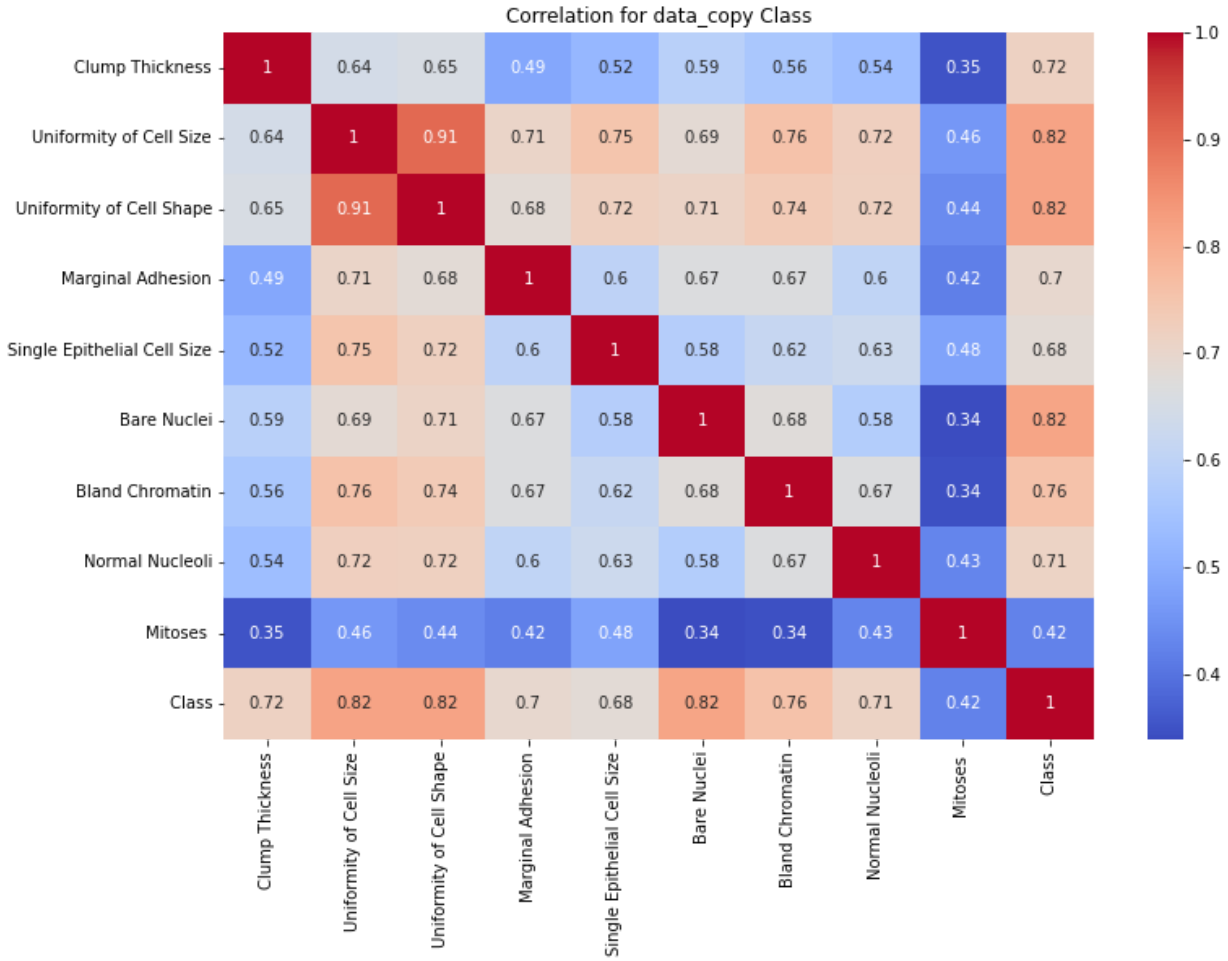
```
from scipy.spatial.distance import pdist, squareform

# Split the data into benign and malignant classes
benign_data = data_copy[data_copy[' Class'] == 2]
malignant_data = data_copy[data_copy[' Class'] == 4]

# Create a heatmap for benign data
plt.figure(figsize=(12,8))
sns.heatmap(benign_data.corr(),annot=True,cmap='coolwarm')
plt.title('Correlation for Benign Class')
plt.show()

```



In [30]:
```
# Compute the correlation matrix
corr_matrix = malignant_data.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, cmap='coolwarm', annot=True)
plt.title('Correlation for Malignant Class')
plt.show()
```



In [31]:
```
corr_matrix = data_copy.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, cmap='coolwarm', annot=True)
plt.title('Correlation for data_copy Class')
plt.show()
```



pre-processing steps

In [ ]:
```

```

In [32]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

from sklearn.svm import SVC


from sklearn.feature_selection import RFE
```

In [33]:
```
# Split the data into features (X) and target (y)
X = data_copy.drop(' Class', axis=1)
y = data_copy[' Class']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_
```

In [34]:
```
# Initialize SVM and Random Forest algorithms
svm = SVC(kernel='linear')
rf = RandomForestClassifier(n_estimators=50, random_state=0)

# Perform RFE feature selection with SVM
svm_rfe = RFE(estimator=svm, n_features_to_select=9, step=1)
svm_rfe.fit(X_train, y_train)

# Perform RFE feature selection with Random Forest
rf_rfe = RFE(estimator=rf, n_features_to_select=9, step=1)
rf_rfe.fit(X_train, y_train)

# Print the selected feature rankings
print('Feature Rankings:', svm_rfe.ranking_)
print('Feature Rankings:', rf_rfe.ranking_)
print(" ")

# Print the selected features
selected_features = X_train.columns[svm_rfe.support_]
print('Selected Features for SVM:', selected_features)
print(" ")
selected_features = X_train.columns[rf_rfe.support_]
print('Selected Features for Random Forest:', selected_features)
```

```
Feature Rankings: [1 1 1 1 1 1 1 1 1]
Feature Rankings: [1 1 1 1 1 1 1 1 1]

Selected Features for SVM: Index(['Clump Thickness', 'Uniformity of Cell Size',
       'Uniformity of Cell Shape', 'Marginal Adhesion',
       'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
       'Normal Nucleoli', 'Mitoses '],
      dtype='object')

Selected Features for Random Forest: Index(['Clump Thickness', 'Uniformity of Cell
Size',
       'Uniformity of Cell Shape', 'Marginal Adhesion',
       'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
       'Normal Nucleoli', 'Mitoses '],
      dtype='object')
```

In [35]:
```
# Transform training and test sets to include only selected features
X_train_svm = svm_rfe.transform(X_train)
X_test_svm = svm_rfe.transform(X_test)

X_train_rf = rf_rfe.transform(X_train)
X_test_rf = rf_rfe.transform(X_test)
```

In [36]:
```
# Train SVM and Random Forest using selected features
svm.fit(X_train_svm, y_train)
rf.fit(X_train_rf, y_train)
```

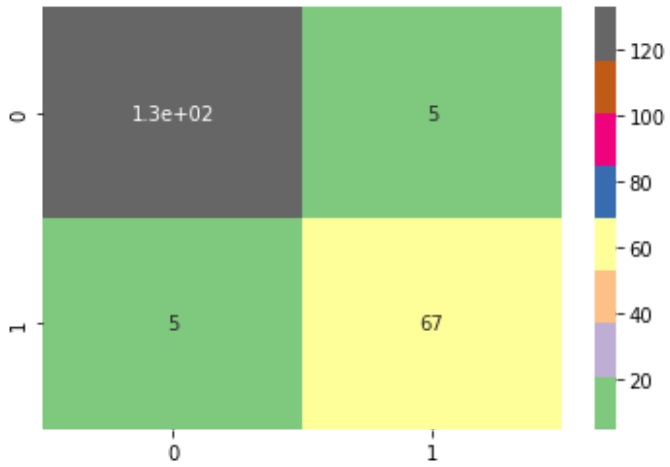
Out[36]:
```
RandomForestClassifier(n_estimators=50, random_state=0)
```

Model Evaluation

In [37]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_scor
```

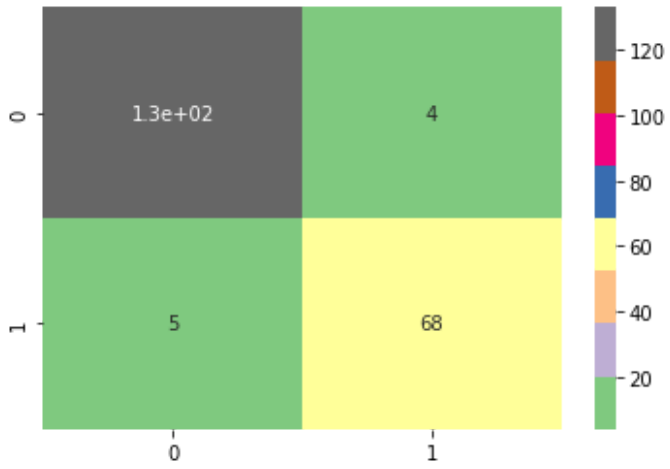
In [38]:
```
# Predict on test set

svm_pred = svm.predict(X_test_svm)

print("Support Vector Machine(SVM) ")
print("accuracy is ",accuracy_score(svm_pred,y_test))
print ("precision is ",precision_score(svm_pred,y_test,pos_label=4))
print("recall is ",recall_score(svm_pred,y_test,pos_label=4))
print("f1 is ",f1_score(svm_pred,y_test,pos_label=4))
print(sns.heatmap(confusion_matrix(svm_pred, y_test),cmap='Accent', annot=True)
```

```
Support Vector Machine(SVM)
accuracy is  0.9523809523809523
precision is  0.9305555555555556
recall is  0.9305555555555556
f1 is  0.9305555555555556
AxesSubplot(0.125,0.125;0.62x0.755)
```



In [39]:
```
# random forest
rf_pred = rf.predict(X_test_rf)
print("Random Forest ")
print("accuracy is ",accuracy_score(rf_pred,y_test))
print ("precision is ",precision_score(rf_pred,y_test,pos_label=4))
print("recall is ",recall_score(rf_pred,y_test,pos_label=4))
print("f1 is ",f1_score(rf_pred,y_test,pos_label=4))
print(sns.heatmap(confusion_matrix(rf_pred, y_test),cmap='Accent', annot=True))
```

```
Random Forest
accuracy is  0.9571428571428572
precision is  0.9444444444444444
recall is  0.9315068493150684
f1 is  0.9379310344827586
AxesSubplot(0.125,0.125;0.62x0.755)
```



In [ ]:
```
1
```