-- Breast Cancer --

In [54]:
```
import pandas as pd
```

In [55]:
```
import numpy as np
```

In [56]:
```
import matplotlib.pyplot as plt
```

In [57]:
```
import seaborn as sns
```

In [59]:
```
# Import Dataset
df=pd.read_csv("brestcancer.csv")
df.head()
```

Out[59]:

	Unnamed: 0	id	clump_thickness	uniform_cell_size	uniform_cell_shape	marginal_adhesion
0	0	1000025	5	1	1	1
1	1	1002945	5	4	4	5
2	2	1015425	3	1	1	1
3	3	1016277	6	8	8	1
4	4	1017023	4	1	1	3

In [60]:
```
# Check for null values
pd.isnull(df).sum()
```

Out[60]:
```
Unnamed: 0                0
id                        0
clump_thickness           0
uniform_cell_size         0
uniform_cell_shape        0
marginal_adhesion         0
single_epithelial_size    0
bare_nuclei               0
bland_chromatin           0
normal_nucleoli           0
mitoses                   0
class                     0
dtype: int64
```

In [61]:
```
# Outcome Feature (Class) values
df['class'].unique()
```

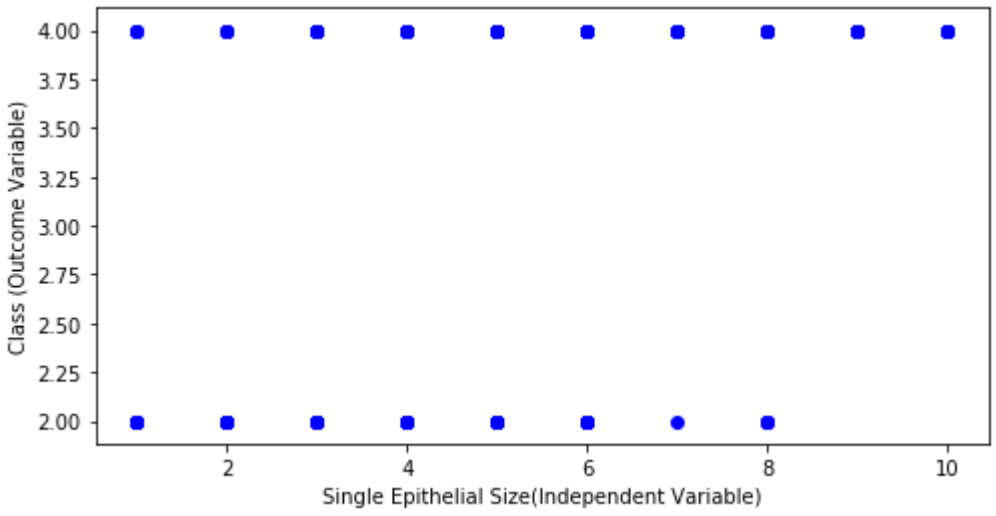
Out[61]:
```
array([2, 4], dtype=int64)
```

In [62]:
```
# Data Description
df.describe()
```

Out[62]:

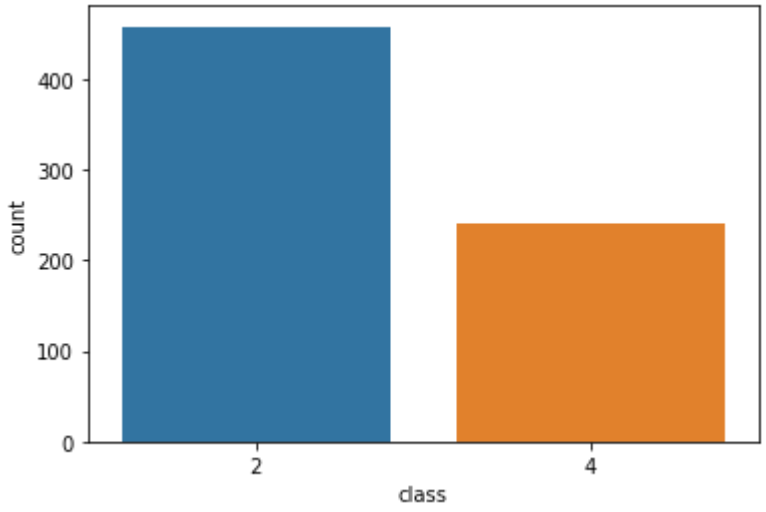
	Unnamed: 0	id	clump_thickness	uniform_cell_size	uniform_cell_shape	marginal_
count	699.000000	6.990000e+02	699.000000	699.000000	699.000000	6
mean	349.000000	1.071704e+06	4.417740	3.134478	3.207439	
std	201.928205	6.170957e+05	2.815741	3.051459	2.971913	
min	0.000000	6.163400e+04	1.000000	1.000000	1.000000	
25%	174.500000	8.706885e+05	2.000000	1.000000	1.000000	
50%	349.000000	1.171710e+06	4.000000	1.000000	1.000000	
75%	523.500000	1.238298e+06	6.000000	5.000000	5.000000	
max	698.000000	1.345435e+07	10.000000	10.000000	10.000000	

In [73]:
```
# Scatter Plot
X=df["clump_thickness"]
Y=df["class"]
plt.figure(figsize=(8,4))
plt.scatter(X,Y, marker='o', color='blue')
plt.xlabel("Single Epithelial Size(Independent Variable)")
plt.ylabel("Class (Outcome Variable)")
plt.show()
```



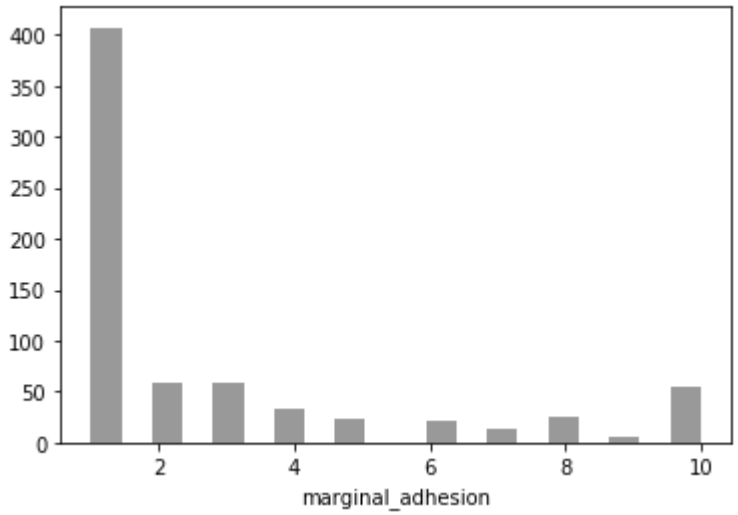
In [30]:
```
# Count Plot
sns.countplot(x="class",data=df)
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x82c63696d8>



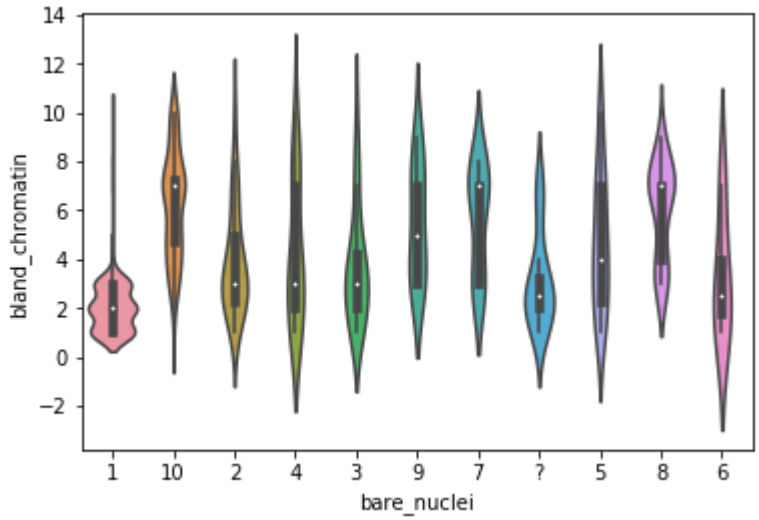
In [34]:
```
sns.distplot(df['marginal_adhesion'],color='black',bins=20,kde=False)
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x82c5edeef0>



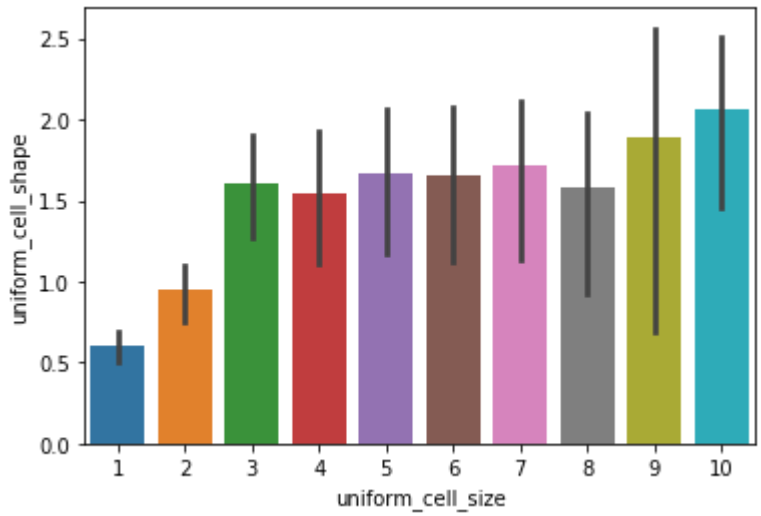
In [33]:
```
# Violin Plot
sns.violinplot(x='bare_nuclei',y='bland_chromatin',data=df)
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x82bf6a4eb8>



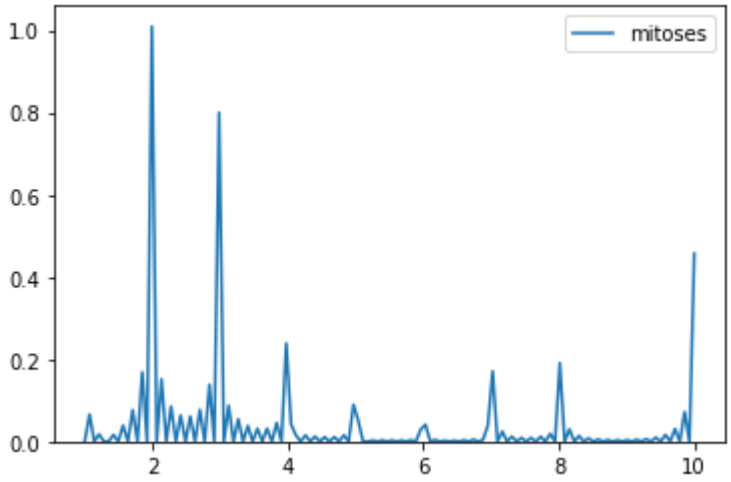
In [35]:
```
# Barplot
sns.barplot(x='uniform_cell_size',y='uniform_cell_shape',data=df,estimator=np.st
```

Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x82c5f684a8>



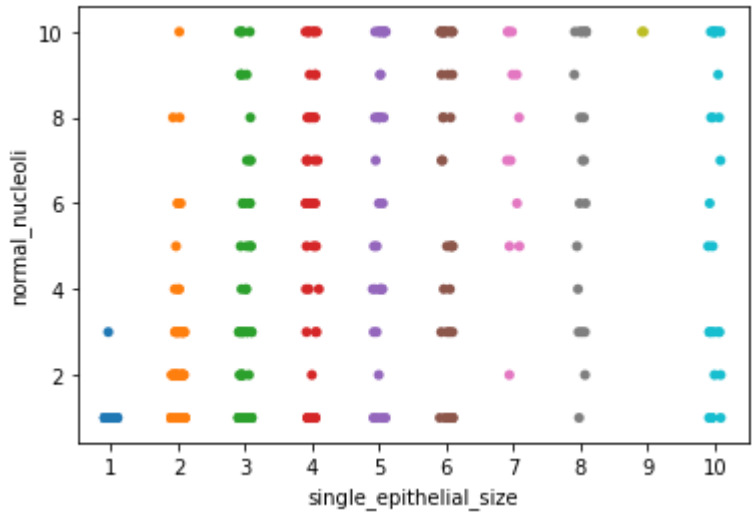
In [43]:
```
sns.kdeplot(df['mitoses'])
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x82c648c9e8>



In [42]:
```
# Strip Plot
sns.stripplot(x='single_epithelial_size',y='normal_nucleoli',data=df,jitter=True
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x82c60df048>



In [16]:
```
from sklearn.model_selection import train_test_split
```

In [17]:
```
# Separating the Dependent and Independent Features
x=df[["clump_thickness","uniform_cell_size","uniform_cell_shape","marginal_adhes
      "single_epithelial_size","bland_chromatin","normal_nucleoli","mitoses"]]
y=df["class"]
```

In [18]:
```
# x_train & y_train for Train the model
# x_test & y_test for Test/ Predict model
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.8,random_state
```

In [21]:
```
# Check train and test data shapes
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(559, 8)
(140, 8)
(559,)
(140,)
```

In [22]:
```
from sklearn import svm
```

In [23]:
```
# Build SVM Classifier Model (Support Vector Classifier)
svc = svm.SVC(kernel= 'linear')
```

In [24]:
```
# Train Model
svc.fit(x_train,y_train)
```

Out[24]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [25]:
```
# Predicting Outcome Class
y_pred = svc.predict(x_test)
```

In [26]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
```

In [27]:
```
# Confusion Matrix from predicted and actual class values
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix :")
print(cm)
```

```
Confusion Matrix :
[[82  3]
 [ 1 54]]
```

In [39]:
```
# Accuracy Score of Class prediction
acc=accuracy_score(y_test, y_pred)*100
print("Accuracy = ",acc,"%")
```

```
Accuracy =  97.14285714285714 %
```

In [29]:
```
# Comparision of Real and Predicted Class values
out = pd.DataFrame({'Real_class': y_test, 'Predicted_class': y_pred})
out.head(10)
```

Out[29]:

	Real_class	Predicted_class
476	2	2
531	2	2
40	2	4
432	2	2
14	4	4
157	2	2
266	4	4
31	2	2
251	4	4
103	4	4

In [ ]:
```

```