In [153]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
%matplotlib inline
```

In [154]:

```
df = pd.read_csv('wdbc.data', header=None)
df.shape
```

Out[154]:

```
(569, 32)
```

In [155]:

```
df.describe()
```

Out[155]:

	0	2	3	4	5	6	
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.00000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.10434
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.05281
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.01938
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.06492
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.09263
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.13040
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.34540

8 rows × 31 columns

In [156]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       569 non-null    int64  
 1   1       569 non-null    object 
 2   2       569 non-null    float64
 3   3       569 non-null    float64
 4   4       569 non-null    float64
 5   5       569 non-null    float64
 6   6       569 non-null    float64
 7   7       569 non-null    float64
 8   8       569 non-null    float64
 9   9       569 non-null    float64
 10  10      569 non-null    float64
 11  11      569 non-null    float64
 12  12      569 non-null    float64
 13  13      569 non-null    float64
 14  14      569 non-null    float64
 15  15      569 non-null    float64
 16  16      569 non-null    float64
 17  17      569 non-null    float64
 18  18      569 non-null    float64
 19  19      569 non-null    float64
 20  20      569 non-null    float64
 21  21      569 non-null    float64
 22  22      569 non-null    float64
 23  23      569 non-null    float64
 24  24      569 non-null    float64
 25  25      569 non-null    float64
 26  26      569 non-null    float64
 27  27      569 non-null    float64
 28  28      569 non-null    float64
 29  29      569 non-null    float64
 30  30      569 non-null    float64
 31  31      569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

In [157]:

```
df[1].replace(to_replace = ['M','B'], value = [1,0], inplace=True)
y = df[1]
y.shape
```

Out[157]:

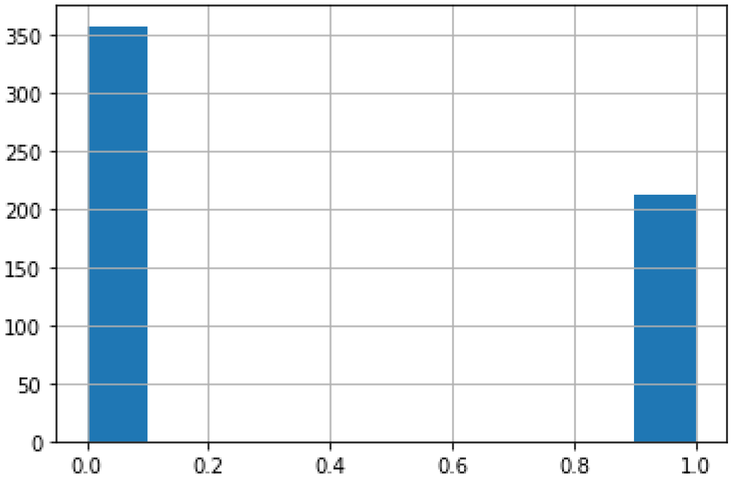
```
(569,)
```

In [158]:

```
y.hist()
```

Out[158]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feabd050760>
```



In [159]:

```
X = df.drop([0,1],axis=1)
X.shape
```

Out[159]:

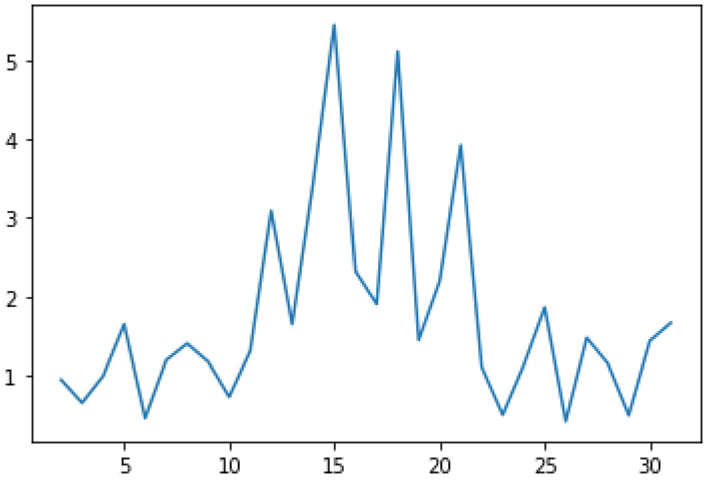
```
(569, 30)
```

In [160]:

```
X.skew().plot()
```

Out[160]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feabb5d9fd0>
```



Part (a)

In [161]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random
_state=42, stratify=y)
X_train.shape, X_test.shape,  y_train.shape,  y_test.shape
```

Out[161]:

```
((455, 30), (114, 30), (455,), (114,))
```

In [162]:

```
parameters = {'max_depth':range(1,8)}
dtree_model = GridSearchCV(DecisionTreeClassifier(random_state=42), parameters,
cv=5)
dtree_model.fit(X_train, y_train)
```

Out[162]:

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(random_state=4
2),
             param_grid={'max_depth': range(1, 8)})
```

In [163]:

```
dtree_model.score(X_train,y_train)
```

Out[163]:

```
0.9956043956043956
```

In [164]:

```
accuracy_DecisionTree = dtree_model.score(X_test,y_test)
accuracy_DecisionTree
```

Out[164]:

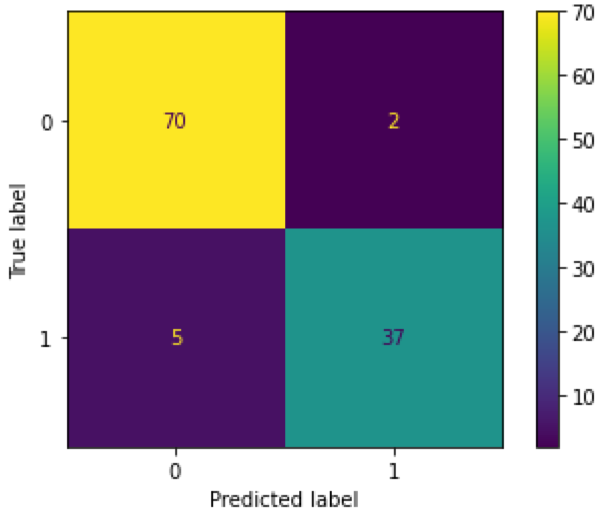
```
0.9385964912280702
```

In [165]:

```
plot_confusion_matrix(dtree_model,X_test, y_test)
```

Out[165]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x
7feabbe4b340>
```



In [166]:

```
rfc=RandomForestClassifier(random_state=42)
```

In [167]:

```
param_grid = {
    'n_estimators': [200, 500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [1,2,3,4,5,6,7,8,10],
    'criterion' :['gini', 'entropy']
}
CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5, n_jobs=-1)
CV_rfc.fit(X_train, y_train)
```

Out[167]:

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=4
2), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 10],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [200, 500]})
```

In [168]:

```
CV_rfc.best_params_
```

Out[168]:

```
{'criterion': 'entropy',
 'max_depth': 7,
 'max_features': 'auto',
 'n_estimators': 200}
```

In [169]:

```
CV_rfc.score(X_test, y_test)
```

Out[169]:

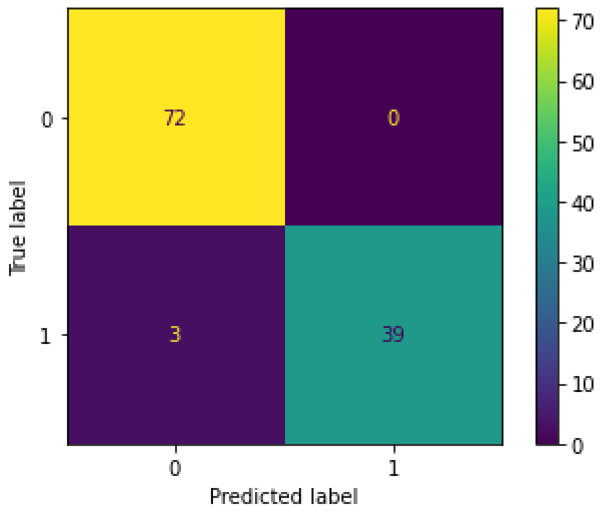
```
0.9736842105263158
```

In [170]:

```
plot_confusion_matrix(CV_rfc,X_test, y_test)
```

Out[170]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x
7feabd3720a0>
```



Part (b)

In [171]:

```
rfc=RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [10, 20, 100, 200, 300, 400, 500, 600, 700]
}
CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5, n_jobs=-1)
CV_rfc.fit(X, y)
```

Out[171]:

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=4
2), n_jobs=-1,
             param_grid={'n_estimators': [10, 20, 100, 200, 300, 40
0, 500, 600,
                                          700]})
```

In [172]:

```
CV_rfc.cv_results_
```

Out[172]:

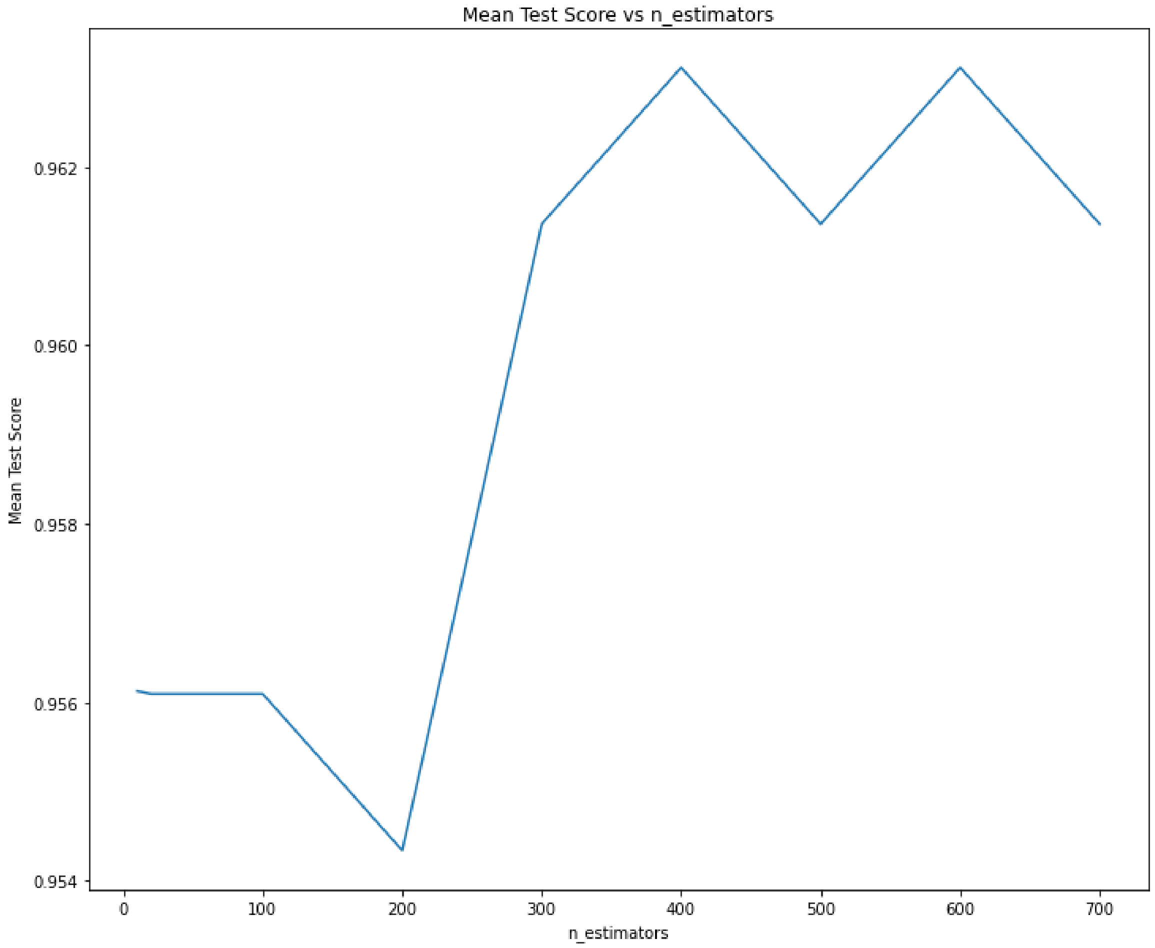
```
{'mean_fit_time': array([0.03571968, 0.08496003, 0.28642054, 0.57570
558, 0.85590138,
        1.13937521, 1.40751786, 1.66839108, 1.52594824]),
 'std_fit_time': array([0.00373215, 0.00598535, 0.00818676, 0.012939
29, 0.00877402,
        0.00999438, 0.02215886, 0.01370128, 0.19436753]),
 'mean_score_time': array([0.00652199, 0.00686426, 0.01773381, 0.032
46331, 0.04336243,
        0.05755959, 0.07233667, 0.08863335, 0.05705395]),
 'std_score_time': array([0.00145194, 0.00108925, 0.00246044, 0.0024
6199, 0.00092247,
        0.0010968 , 0.00119552, 0.01265821, 0.00509207]),
 'param_n_estimators': masked_array(data=[10, 20, 100, 200, 300, 40
0, 500, 600, 700],
              mask=[False, False, False, False, False, False, False,
False,
                    False],
        fill_value='?',
             dtype=object),
 'params': [{'n_estimators': 10},
  {'n_estimators': 20},
  {'n_estimators': 100},
  {'n_estimators': 200},
  {'n_estimators': 300},
  {'n_estimators': 400},
  {'n_estimators': 500},
  {'n_estimators': 600},
  {'n_estimators': 700}],
 'split0_test_score': array([0.9122807 , 0.92982456, 0.92105263, 0.9
2105263, 0.92982456,
        0.92982456, 0.92982456, 0.92982456, 0.92982456]),
 'split1_test_score': array([0.92105263, 0.92105263, 0.93859649, 0.9
3859649, 0.93859649,
        0.94736842, 0.94736842, 0.94736842, 0.94736842]),
 'split2_test_score': array([0.99122807, 0.98245614, 0.98245614, 0.9
8245614, 0.99122807,
        0.99122807, 0.99122807, 0.99122807, 0.98245614]),
 'split3_test_score': array([0.96491228, 0.97368421, 0.96491228, 0.9
5614035, 0.97368421,
        0.97368421, 0.96491228, 0.97368421, 0.97368421]),
 'split4_test_score': array([0.99115044, 0.97345133, 0.97345133, 0.9
7345133, 0.97345133,
        0.97345133, 0.97345133, 0.97345133, 0.97345133]),
 'mean_test_score': array([0.95612483, 0.95609377, 0.95609377, 0.954
33939, 0.96135693,
        0.96311132, 0.96135693, 0.96311132, 0.96135693]),
 'std_test_score': array([0.03373016, 0.02539148, 0.02283883, 0.0224
2725, 0.02325016,
        0.02174903, 0.02117153, 0.02174903, 0.01966409]),
 'rank_test_score': array([6, 7, 7, 9, 3, 1, 3, 1, 3], dtype=int32)}
```

In [173]:

```
y = np.array(CV_rfc.cv_results_['mean_test_score'])
x = np.array(param_grid['n_estimators'])
```

In [174]:

```
%matplotlib inline
plt.figure(figsize=(12, 10))
plt.plot(x, y)
plt.xlabel('n_estimators')
plt.ylabel('Mean Test Score')
plt.title('Mean Test Score vs n_estimators')
plt.show()
```



Part (c)

In [175]:

```
y = df[1]
y.shape
rfc=RandomForestClassifier(random_state=42)
param_grid = {
    'max_features': [i for i in range(1,31)]
}
CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5, n_jobs=-1)
CV_rfc.fit(X, y)
```

Out[175]:

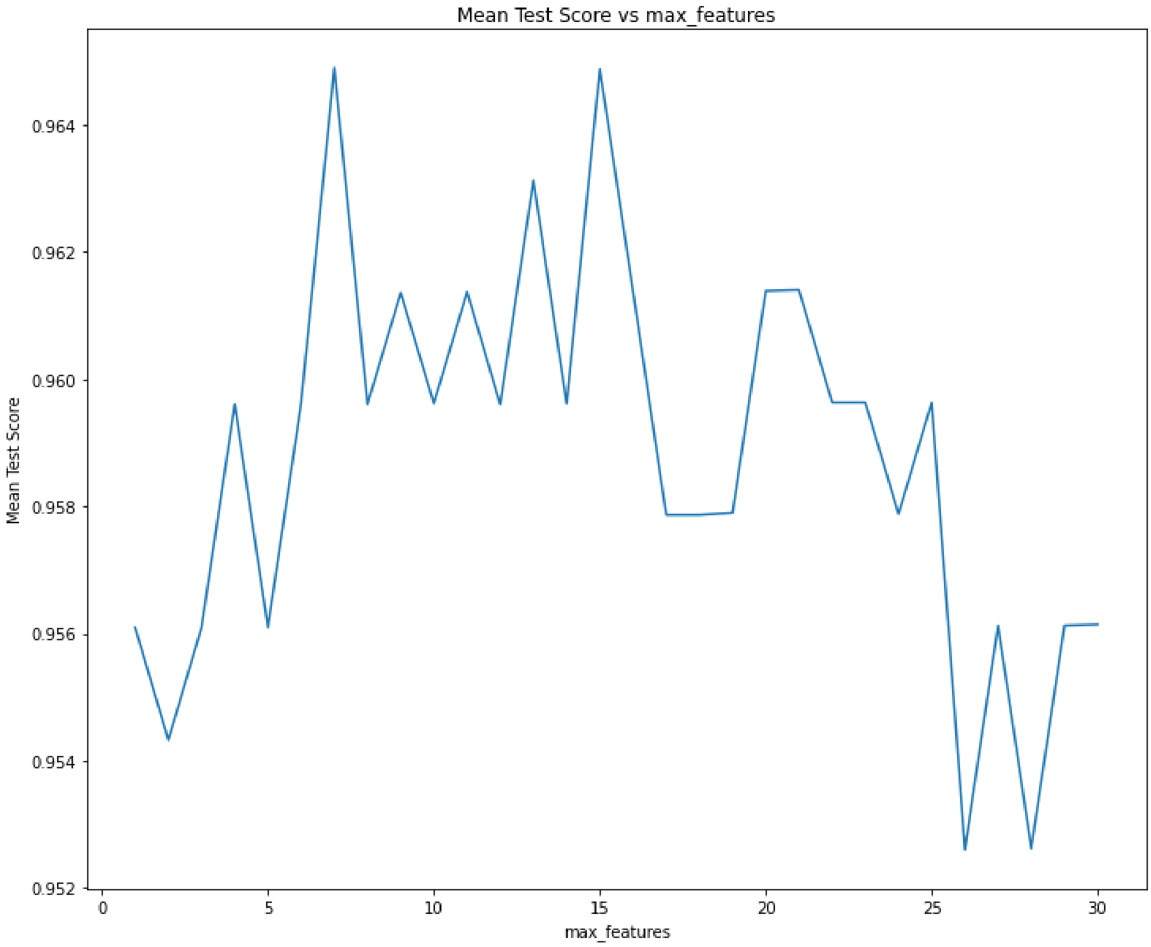
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=4
2), n_jobs=-1,
             param_grid={'max_features': [1, 2, 3, 4, 5, 6, 7, 8, 9,
10, 11, 12,
                                          13, 14, 15, 16, 17, 18, 1
9, 20, 21,
                                          22, 23, 24, 25, 26, 27, 2
8, 29, 30]})
```

In [176]:

```
out = np.array(CV_rfc.cv_results_['mean_test_score'])
inp = np.array(param_grid['max_features'])
```

In [177]:

```
%matplotlib inline
plt.figure(figsize=(12, 10))
plt.plot(inp, out)
plt.xlabel('max_features')
plt.ylabel('Mean Test Score')
plt.title('Mean Test Score vs max_features')
plt.show()
```


Mean Test Score vs max_features
Mean Test Score
0.952
0.954
0.956
0.958
0.960
0.962
0.964
max_features
0
5
10
15
20
25
30

In [ ]:

```

```