In [24]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.preprocessing import StandardScaler
```

In [2]:

```
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
df = pd.DataFrame( data['data'], columns=data['feature_names'])
df.head()
```

Out[2]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 30 columns

In [3]:

```
df.isna().sum()
```

Out[3]:

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
dtype: int64
```

In [11]:

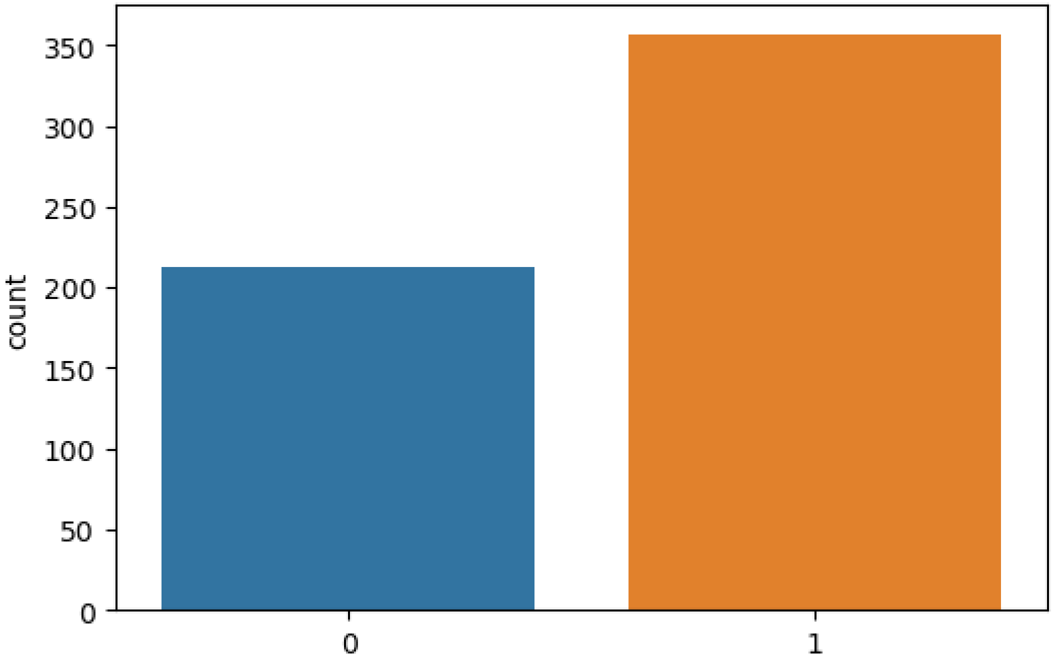
```
target = data['target']
target[:5]
```

Out[11]:

```
array([0, 0, 0, 0, 0])
```

In [61]:

```
plt.figure(dpi=100)
sns.countplot(target)
plt.show()
```



In [15]:

```
features=df
features.head(5)
```

Out[15]:

ıean ıess	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	wo smoothne
7760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.16
7864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0.12
5990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.14
3390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0.20
3280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.13

In [27]:

```
x_train,x_test,y_train,y_test= train_test_split(features,target,test_size=0.2)
```

In [41]:

```
x_train.head()
```

Out[41]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	me symme
539	7.691	25.44	48.34	170.4	0.08668	0.11990	0.09252	0.01364	0.20
407	12.850	21.37	82.63	514.5	0.07551	0.08316	0.06126	0.01867	0.15
417	15.500	21.08	102.90	803.1	0.11200	0.15710	0.15220	0.08481	0.20
318	9.042	18.90	60.07	244.5	0.09968	0.19720	0.19750	0.04908	0.23
0	17.990	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.24

5 rows × 30 columns

In [35]:

```
model1=LogisticRegression()
model1.fit(x_train,y_train)
```

```
F:\RC SLOG\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:940:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[35]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [36]:

```
pred = model1.predict(x_test)
pred[:5]
```

Out[36]:

```
array([1, 0, 1, 0, 1])
```

In [37]:

```
y_test[:5]
```

Out[37]:

```
array([1, 0, 1, 0, 1])
```

In [53]:

```
r2_score(y_test, pred)*100
```

Out[53]:

```
54.293351152689596
```

In [72]:

```
prob = model1.predict_proba(x_test)
prob[:5]
```

Out[72]:

```
array([[1.78737514e-02, 9.82126249e-01],
       [9.99997337e-01, 2.66304652e-06],
       [1.23864036e-04, 9.99876136e-01],
       [9.99932159e-01, 6.78412478e-05],
       [2.72163255e-01, 7.27836745e-01]])
```

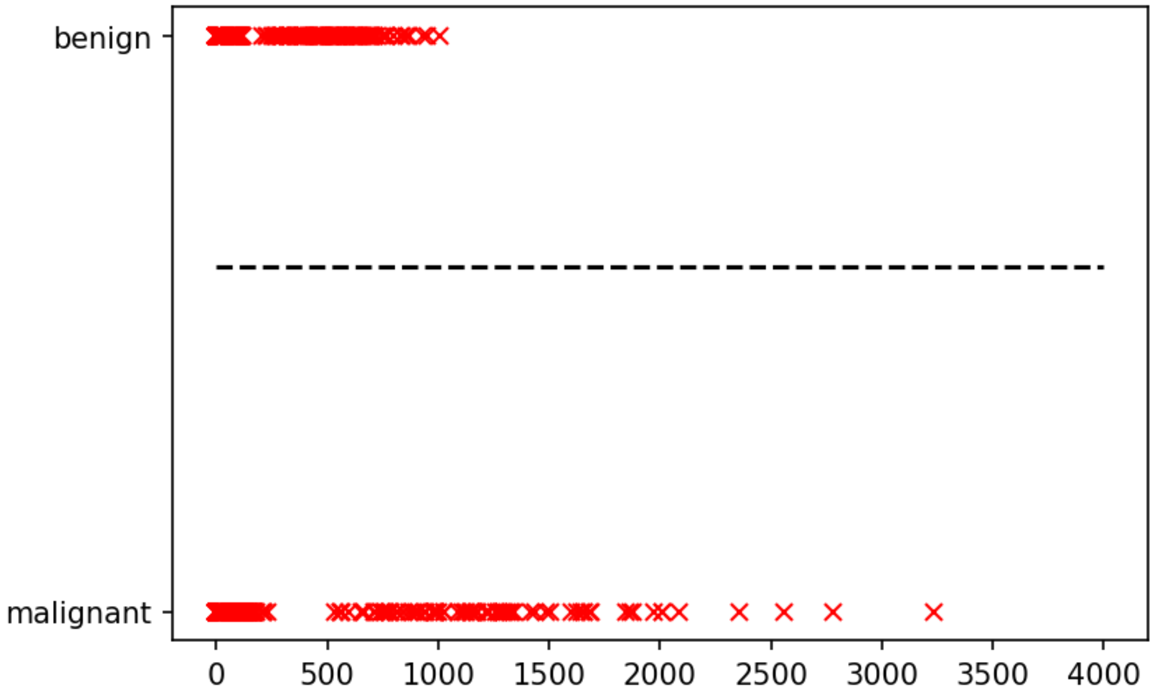
In [73]:

```
p = prob[:, 1]
```

In [89]:

```
thresh_hold = 0.6
pred = model1.predict(x_test)
pred[pred > thresh_hold] = 1

plt.figure(dpi=150)
plt.plot(x_test, y_test, 'xr')
plt.yticks([0, 1], ['malignant', 'benign'], fontsize=10,)
plt.plot([0,4000], [thresh_hold, thresh_hold], 'k--')
plt.show()
```



In [ ]:

```

```