In [1]:
```
#ML PROJECT
#BREAST CANCER CLASSIFICATION
```

In [ ]:
```

```

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import sklearn as sk
```

In [3]:
```
df = pd.read_csv(r'C:\Users\zeesh\OneDrive\Desktop\data.csv')
df
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smooth
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	
...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	
565	926682	M	20.13	28.25	131.20	1261.0	
566	926954	M	16.60	28.08	108.30	858.1	
567	927241	M	20.60	29.33	140.10	1265.0	
568	92751	B	7.76	24.54	47.92	181.0	

569 rows × 15 columns

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

Out[4]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothne
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   id                      569 non-null    int64  
 1   diagnosis               569 non-null    object 
 2   radius_mean             569 non-null    float64
 3   texture_mean            569 non-null    float64
 4   perimeter_mean          569 non-null    float64
 5   area_mean               569 non-null    float64
 6   smoothness_mean         569 non-null    float64
 7   compactness_mean        569 non-null    float64
 8   concavity_mean          569 non-null    float64
 9   concave points_mean     569 non-null    float64
 10  symmetry_mean           569 non-null    float64
 11  fractal_dimension_mean  569 non-null    float64
 12  radius_se               569 non-null    float64
 13  fractal_dimension_worst  569 non-null    float64
 14  compactness_se          569 non-null    float64
dtypes: float64(13), int64(1), object(1)
memory usage: 66.8+ KB
```

In [6]:
```
df.shape
```

Out[6]:
```
(569, 15)
```

In [7]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
```

In [8]:
```
le
```

Out[8]:
```
LabelEncoder()
```

In [9]:
```
df.diagnosis = le.fit_transform(df.diagnosis)
```

In [10]:
```
df.head()
```

Out[10]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothne
0	842302	1	17.99	10.38	122.80	1001.0	
1	842517	1	20.57	17.77	132.90	1326.0	
2	84300903	1	19.69	21.25	130.00	1203.0	
3	84348301	1	11.42	20.38	77.58	386.1	
4	84358402	1	20.29	14.34	135.10	1297.0	

In [11]:
```
df.duplicated().sum()
```

Out[11]:
```
0
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
fractal_dimension_worst    0
compactness_se             0
dtype: int64
```

In [13]:
```
X = X = df.drop('diagnosis', axis = 1)
```

In [14]:
```
Y = df['diagnosis']
Y
```

Out[14]:
```
0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: diagnosis, Length: 569, dtype: int32
```

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

In [ ]:
```
#KNN
```

In [17]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [18]:
```
knnmodel=KNeighborsClassifier(n_neighbors=5)
```

In [19]:
```
knnmodel.fit(X_train, Y_train)
```

Out[19]:
```
KNeighborsClassifier()
```

In [20]:
```
y_predict = knnmodel.predict(X_test)
```

In [21]:
```
y_predict
```

Out[21]:
```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0])
```

In [22]:
```
from sklearn import metrics
print("Accuracy:", metrics.accuracy_score(Y_test, y_predict))
```

```
Accuracy: 0.631578947368421
```

In [ ]:
```
#NAIVE BAYES
```

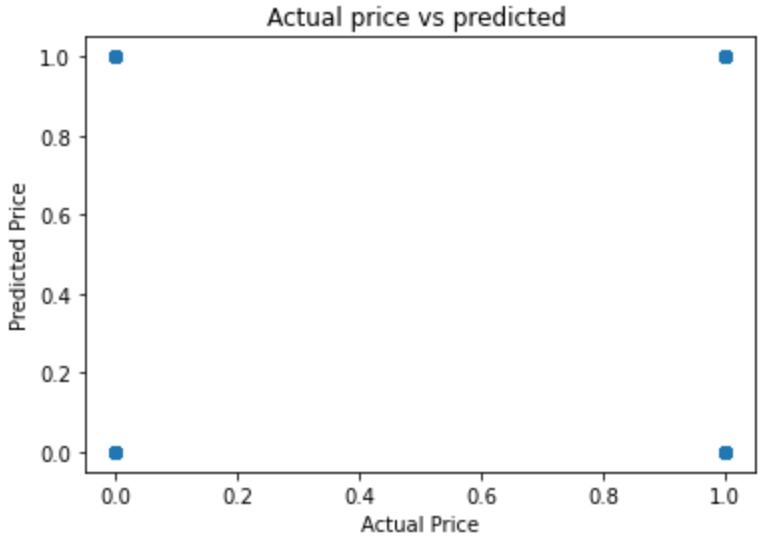
In [23]:
```
from sklearn.naive_bayes import MultinomialNB # as the dataset is discrete
Model= MultinomialNB()
```

In [24]:
```
Model.fit(X_train, Y_train)
#predict response for test dataset
y_predict = Model.predict(X_test)
```

In [25]:
```
from sklearn import metrics
print("Accuracy:", metrics.accuracy_score(Y_test, y_predict))
```

```
Accuracy: 0.4473684210526316
```

In [49]:
```
plt.scatter(Y_test, y_predict)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual price vs predicted ")
plt.show()
```



In [ ]:
```
#LOGISTIC REGRESSION
```

In [27]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train, Y_train)
```

Out[27]:
```
LogisticRegression()
```

In [28]:
```
y_pred = classifier.predict(X_test)
```

In [29]:
```
from sklearn.metrics import confusion_matrix
```

In [30]:
```
cm = confusion_matrix(Y_test, y_pred)
```

In [31]:
```
cm
```

Out[31]:
```
array([[67,  0],
       [47,  0]], dtype=int64)
```

In [ ]:
```
#LINEAR REGRESSION
```

In [32]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
```

In [33]:
```
regressor.fit(X_train, Y_train)
```

Out[33]:
```
LinearRegression()
```

In [56]:
```
y_pred = regressor.predict(X_test)
```

In [57]:
```
from sklearn import metrics
```

In [58]:
```
print("Accuracy:", metrics.r2_score(Y_test, y_pred))
```

```
Accuracy: 0.6648441423965659
```

In [59]:
```
#DECISION TREE
```

In [60]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [38]:
```
classifier = DecisionTreeClassifier(criterion= 'entropy', random_state=0)
```

In [39]:
```
classifier.fit(X_train, Y_train)
```

Out[39]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [40]:
```
y_pred = classifier.predict(X_test)
```

In [42]:
```
from sklearn.metrics import confusion_matrix
```

In [44]:
```
cm = confusion_matrix(Y_test, y_pred)
cm
```

Out[44]:
```
array([[65,  2],
       [ 7, 40]], dtype=int64)
```

In [ ]:
```
#RANDOM FOREST
```

In [45]:
```
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 5, criterion="entropy")
classifier.fit(X_train, Y_train)
```

Out[45]:
```
RandomForestClassifier(criterion='entropy', n_estimators=5)
```

In [46]:
```
y_pred = classifier.predict(X_test)
```

In [47]:
```
from sklearn.metrics import confusion_matrix
```

In [48]:
```
cm = confusion_matrix(Y_test, y_pred)
cm
```

Out[48]:
```
array([[65,  2],
       [ 3, 44]], dtype=int64)
```

In [ ]:
```

```