In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
df = pd.read_csv('data.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
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

5 rows × 33 columns

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
id                         569 non-null int64
diagnosis                  569 non-null object
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
Unnamed: 32                0 non-null float64
dtypes: float64(31), int64(1), object(1)
memory usage: 144.5+ KB
```

In [5]:

```
del df['id']
```

In [6]:

```
df.head()
```

Out[6]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	M	17.99	10.38	122.80	1001.0	0.11840
1	M	20.57	17.77	132.90	1326.0	0.08474
2	M	19.69	21.25	130.00	1203.0	0.10960
3	M	11.42	20.38	77.58	386.1	0.14250
4	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 32 columns

In [7]:

```
del df['Unnamed: 32']
```

In [8]:

```
df.head()
```

Out[8]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	M	17.99	10.38	122.80	1001.0	0.11840
1	M	20.57	17.77	132.90	1326.0	0.08474
2	M	19.69	21.25	130.00	1203.0	0.10960
3	M	11.42	20.38	77.58	386.1	0.14250
4	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 31 columns

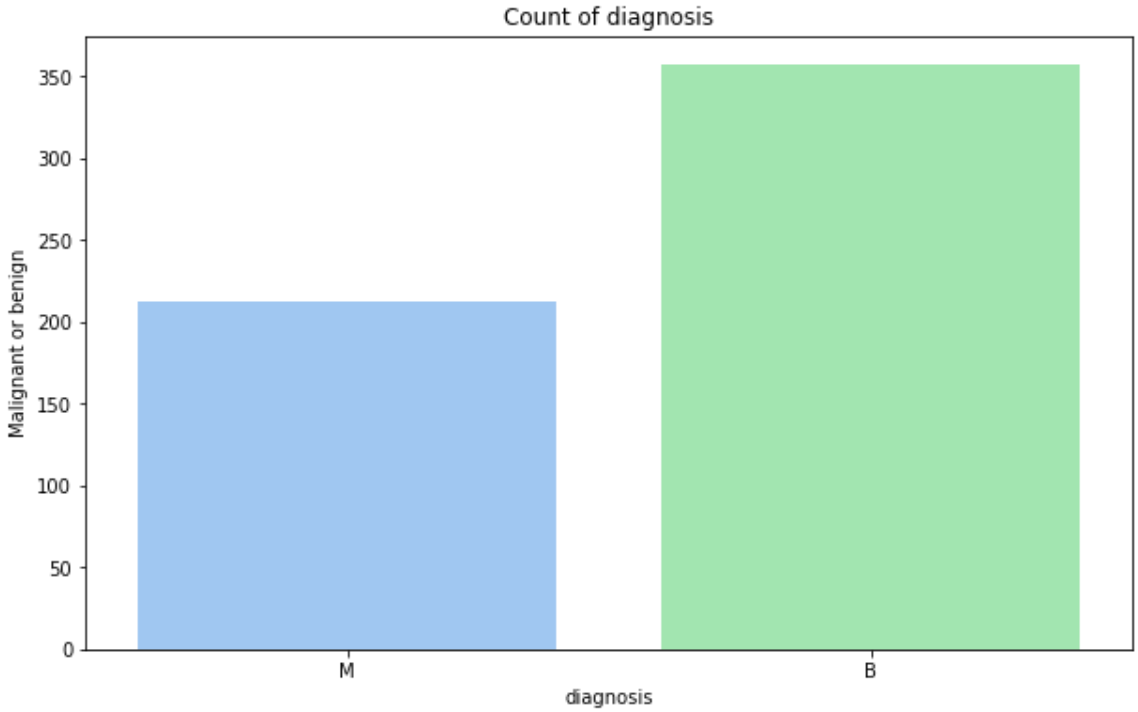
The diagnosis of breast tissues (M = malignant, B = benign)

In [9]:

```
plt.figure(figsize=(10, 6))
sns.countplot('diagnosis', data=df, palette='pastel')
plt.ylabel('Malignant or benign')
plt.title('Count of diagnosis')
```

Out[9]:

```
Text(0.5,1,'Count of diagnosis')
```



In [10]:

```
x = df.drop('diagnosis', axis=1)
```

In [11]:

```
y = df['diagnosis']
```

In [12]:

```
from sklearn.model_selection import train_test_split
```

In [13]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4)
```

In [14]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [15]:

```
dtrain = DecisionTreeClassifier()
```

In [16]:

```
dtrain.fit(x_train, y_train)
```

Out[16]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=Non
e,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=Non
e,
            splitter='best')
```

In [17]:

```
prd = dtrain.predict(x_test)
```

In [18]:

```
from sklearn.metrics import classification_report, confusion_matrix
```

In [19]:

```
print(classification_report(y_test, prd))
print('\n')
print(confusion_matrix(y_test, prd))
```

```
             precision    recall  f1-score   support

          B       0.93      0.94      0.94       143
          M       0.90      0.88      0.89        85

avg / total       0.92      0.92      0.92       228



[[135   8]
 [ 10  75]]
```

Now, lets try random forest

In [20]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [21]:

```
rfc = RandomForestClassifier(n_estimators=1000)
```

In [22]:

```
rfc.fit(x_train, y_train)
```

Out[22]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gin
i',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=1000, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [23]:

```
rfc_pred = rfc.predict(x_test)
```

In [24]:

```
print(classification_report(y_test, rfc_pred))
print('\n')
print(confusion_matrix(y_test, rfc_pred))
```

```
             precision    recall  f1-score   support

          B       0.94      0.99      0.97       143
          M       0.99      0.89      0.94        85

avg / total       0.96      0.96      0.96       228



[[142   1]
 [  9  76]]
```

Random forest is the best model in that case

I can get 96% of precision in that case, the most important diagnosis is Malignant, and for that is 99%. Is a good model.