Import libraries

```
%matplotlib inline
from sklearn.utils.validation import column_or_1d
from sklearn import tree
from sklearn.linear_model import LogisticRegression
from sklearn import neighbors, datasets
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn import neighbors, linear_model, metrics
from sklearn.metrics import classification_report
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import matthews_corrcoef
from sklearn.model_selection import learning_curve
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
```

Data importing and processing

```
wdbc = pd.read_csv("wdbc.data", header = None, names = ['id', 'diagnosis',

 'radius_mean','texture_mean','perimeter_mean','area_mean','smoothness_mean','
compactness_mean','concavity_mean','concave_points_mean','symmetry_mean','frac
tal_dimension_mean',

 'radius_se','texture_se','perimeter_se','area_se','smoothness_se','compactnes
s_se','concavity_se','concave_points_se','symmetry_se','fractal_dimension_se',

 'radius_worst','texture_worst','perimeter_worst','area_worst','smoothness_wor
st','compactness_worst','concavity_worst','concave_points_worst','symmetry_wor
st','fractal_dimension_worst'])
wdbc = wdbc.replace({'M': 0, 'B': 1})
```

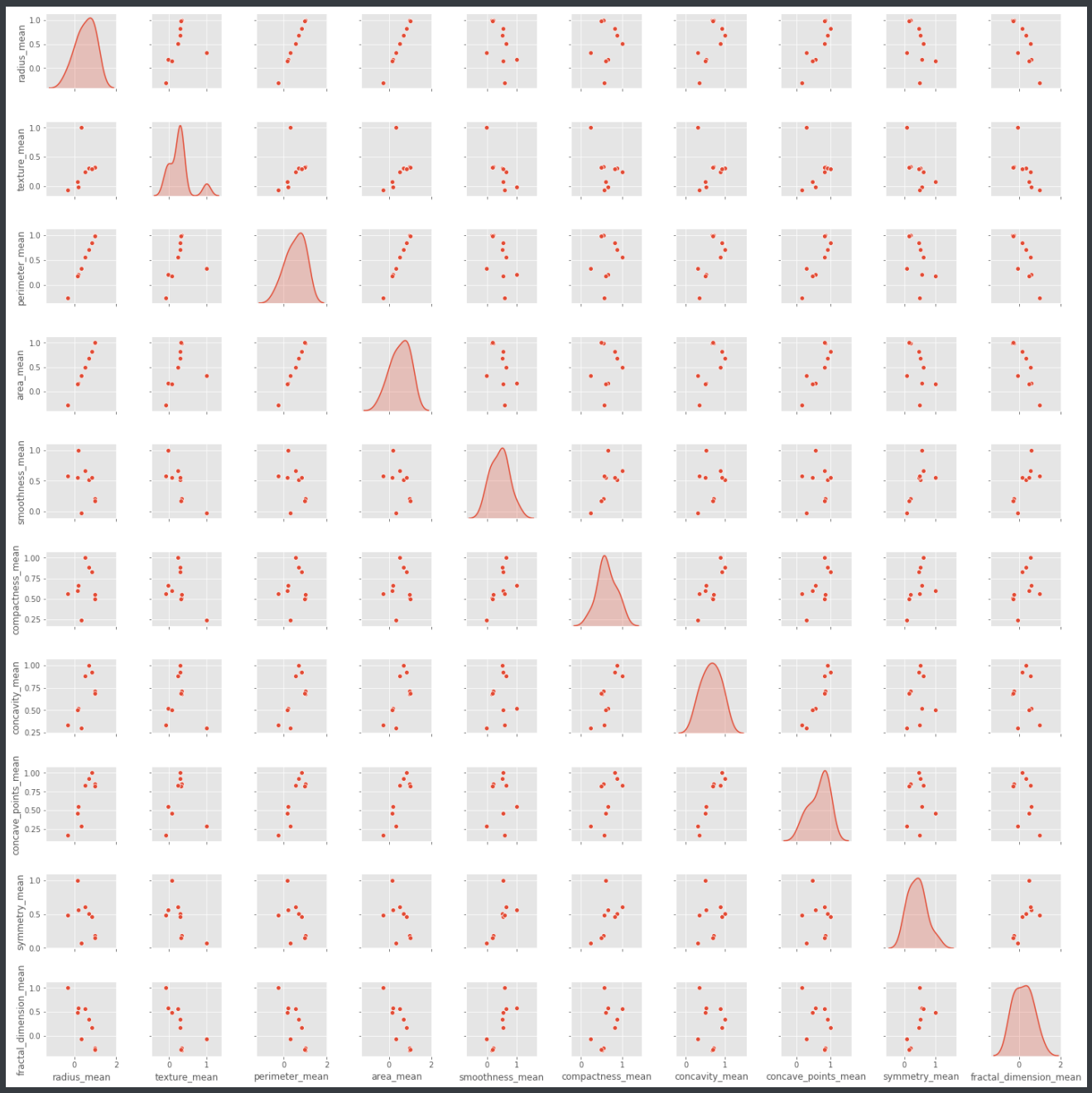
Explore dataset

```
n_samples, n_features = wdbc.shape
print ('The dimensions of the data set are', n_samples, 'by', (n_features-2))
```

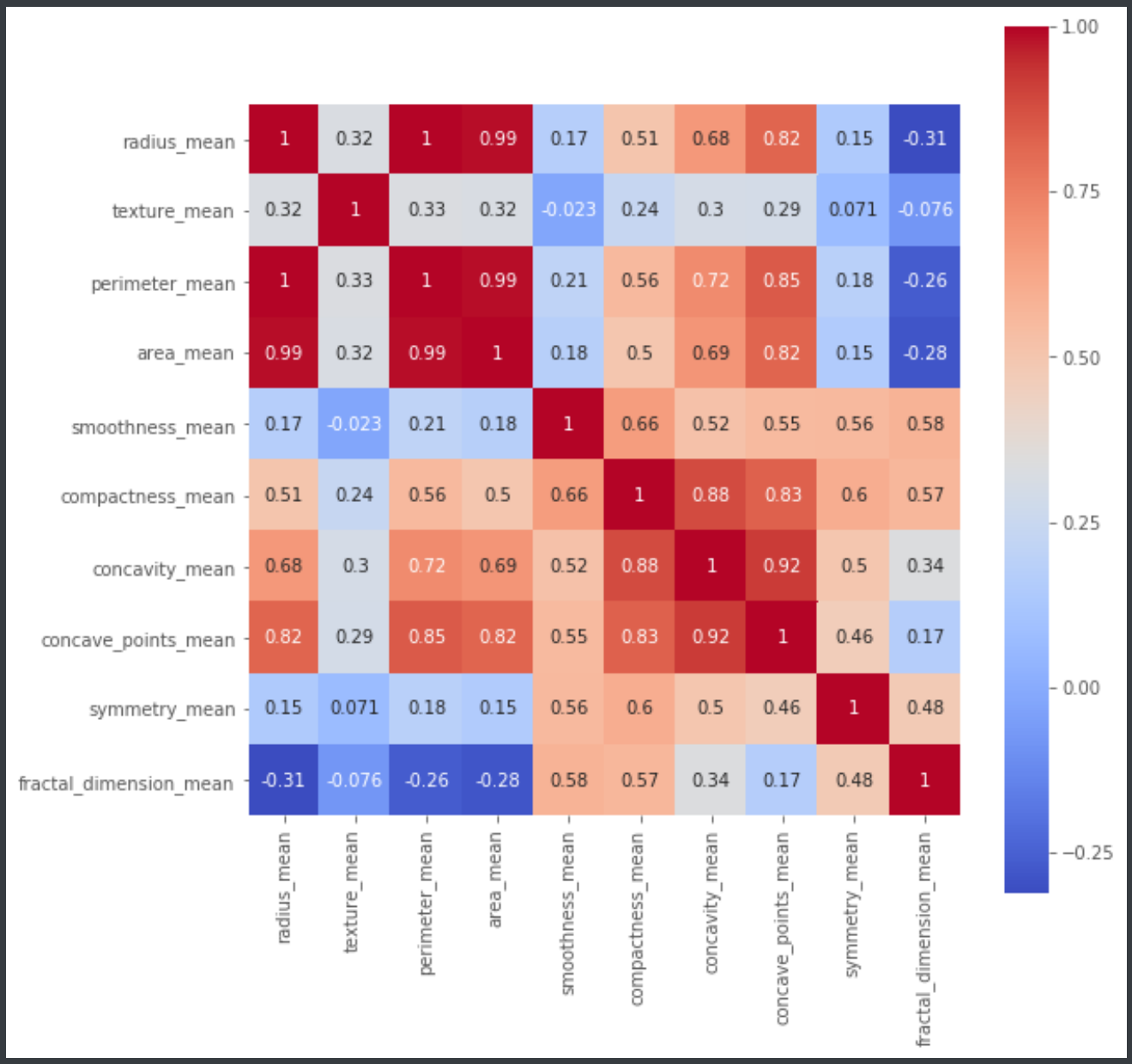
```
The dimensions of the data set are 569 by 30
```

Observe the correlation matrix and create heat map

```
featureMeans = list(wdbc.columns[2:12])

plt.style.use('ggplot')
correlationData = wdbc[featureMeans].corr()
sns.pairplot(wdbc[featureMeans].corr(), diag_kind='kde', height=2);
```




```
plt.figure(figsize=(9,9))
sns.heatmap(wdbc[featureMeans].corr(), annot=True, square=True,
cmap='coolwarm')
plt.show()
```



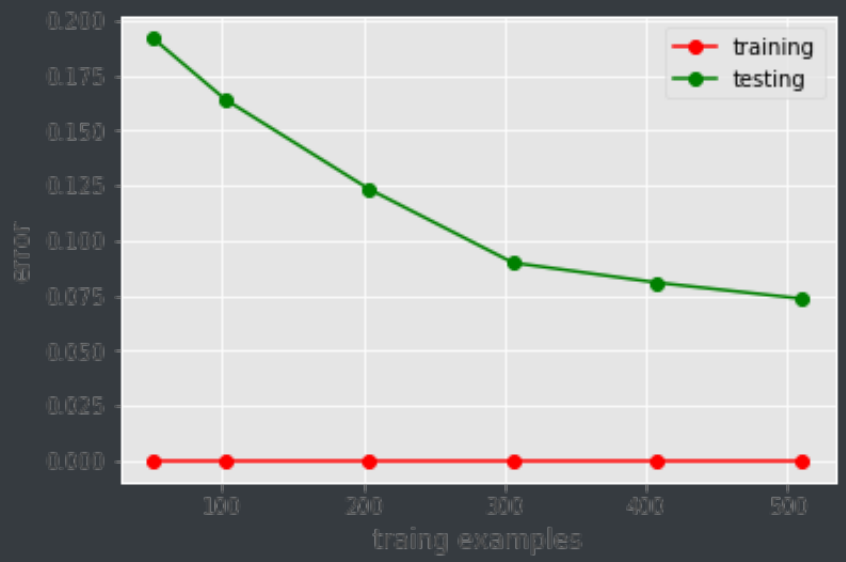
Select features X and Y

```
X = wdbc.iloc[:, 2:].as_matrix()
Y = wdbc['diagnosis'].as_matrix()
Xf = pd.DataFrame(X)
Yf = pd.DataFrame(Y)
X = Xf.as_matrix().astype(int)
Y = Yf.as_matrix().astype(int)
```

1. Decision tree model

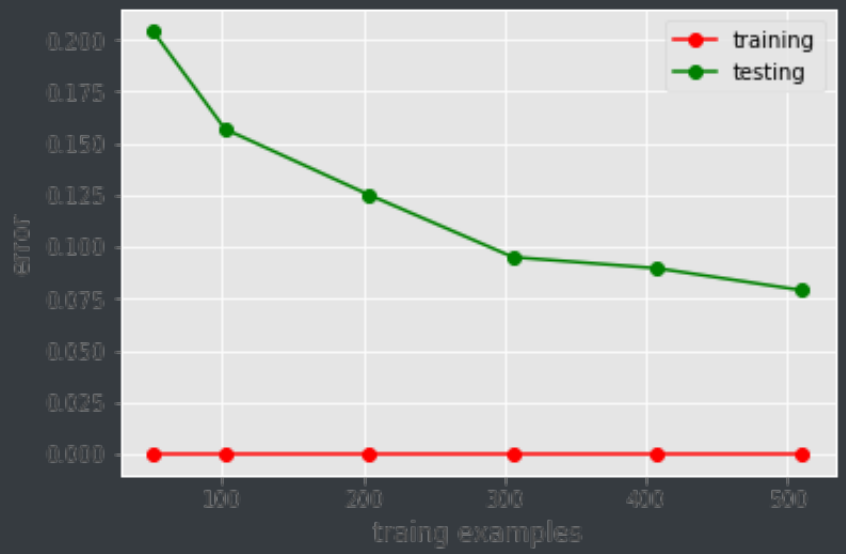
```
for m in range(4,12):
    clf = tree.DecisionTreeClassifier(max_depth = m)
    clf = clf.fit(X, Y)
    # cross validation
    scores = cross_val_score(clf, X, Y, cv=10)
    print("When max_depth =", m)
    print ('\t')
    print("The cross validation result: ", scores)
    print ('\t')
    print("Confusion matrix:\n%s" % metrics.confusion_matrix(Y_test, Y_pred))
    print ('\t')
    print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
    print ('\t')
    print("The kappa stats is: ", cohen_kappa_score(Y_true, Y_pred))
    print ('\t')
    print("The MCC stats is: ", matthews_corrcoef(Y_true, Y_pred))
    print ('\t')
    #Splitting data into training set (70%) and testing set (30%)
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
    Y_true, Y_pred = Y_test, clf.predict(X_test)
    # Classification report
    print(classification_report(Y_true, Y_pred))
    train_sizes,train_score,test_score =
learning_curve(tree.DecisionTreeClassifier(),X, Y, train_sizes=
[0.1,0.2,0.4,0.6,0.8,1], cv=10, scoring='accuracy')
    train_error = 1- np.mean(train_score,axis=1)
    test_error = 1- np.mean(test_score,axis=1)
    plt.plot(train_sizes,train_error,'o-',color = 'r',label = 'training')
    plt.plot(train_sizes,test_error,'o-',color = 'g',label = 'testing')
    plt.legend(loc='best')
    plt.xlabel('traing examples')
    plt.ylabel('error')
    plt.show()
    print ('\n\n\n')
```

```
When max_depth = 4

The cross validation result:  [0.87931034 0.9137931  0.92982456 0.89473684
0.94736842 0.94736842
 0.94736842 0.89285714 0.91071429 0.89285714]

Confusion matrix:
[[73  1]
 [ 0 97]]

Accuracy: 0.92 (+/- 0.05)

The kappa stats is:  0.9880694899881393

The MCC stats is:  0.9881398169534863

              precision    recall  f1-score   support

           0       0.94      0.96      0.95        70
           1       0.97      0.96      0.97       101

   micro avg       0.96      0.96      0.96       171
   macro avg       0.96      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171
```



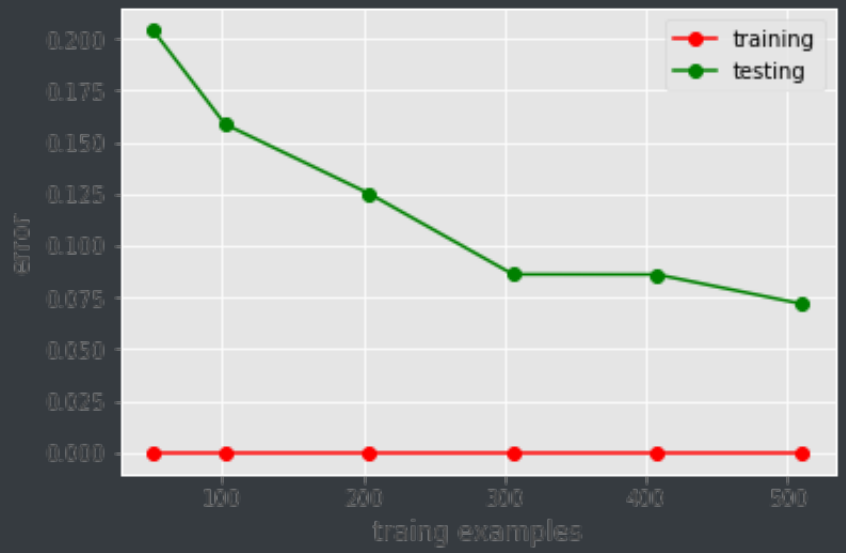

```
When max_depth = 5

The cross validation result:  [0.87931034 0.93103448 0.92982456 0.9122807
0.94736842 0.96491228
 0.94736842 0.91071429 0.94642857 0.98214286]

Confusion matrix:
[[67  3]
 [ 4 97]]

Accuracy: 0.94 (+/- 0.06)

The kappa stats is:  0.9155317197092654

The MCC stats is:  0.9155983822797661

              precision    recall  f1-score   support

           0       0.96      0.93      0.94        70
           1       0.95      0.97      0.96       101

   micro avg       0.95      0.95      0.95       171
   macro avg       0.95      0.95      0.95       171
```

```
weighted avg       0.95      0.95      0.95       171
```



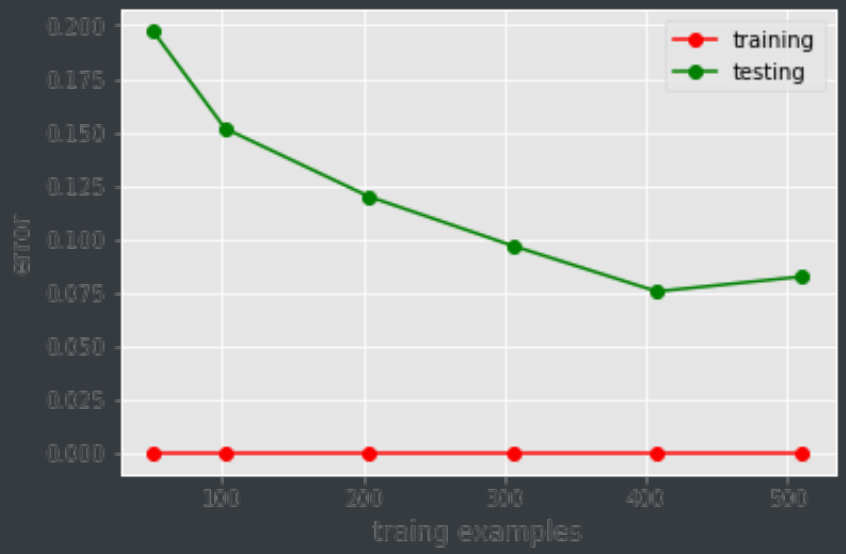

```
When max_depth = 6

The cross validation result:  [0.89655172 0.89655172 0.94736842 0.89473684
0.94736842 0.94736842
 0.9122807  0.92857143 0.94642857 0.98214286]

Confusion matrix:
[[65  5]
 [ 3 98]]

Accuracy: 0.93 (+/- 0.06)

The kappa stats is:  0.9028271061230289

The MCC stats is:  0.9030936306394655

              precision    recall  f1-score   support

           0       0.98      0.98      0.98        62
```

```
           1       0.99      0.99      0.99       109

   micro avg       0.99      0.99      0.99       171
   macro avg       0.99      0.99      0.99       171
weighted avg       0.99      0.99      0.99       171
```



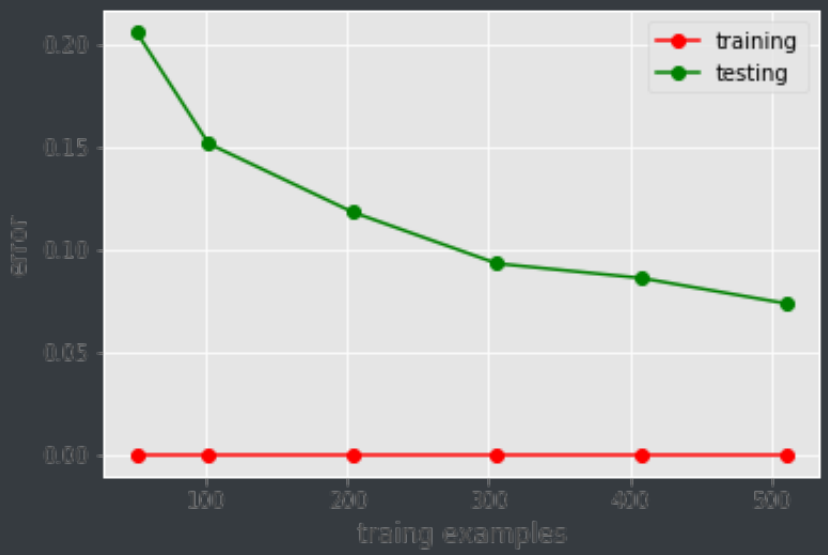

```
When max_depth = 7

The cross validation result:  [0.89655172 0.87931034 0.9122807  0.9122807
0.9122807  0.94736842
 0.92982456 0.91071429 0.94642857 0.96428571]

Confusion matrix:
[[ 61   1]
 [  1 108]]

Accuracy: 0.92 (+/- 0.05)

The kappa stats is:  0.9746966558153299

The MCC stats is:  0.9746966558153299
```

```
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        53
           1       0.97      1.00      0.98       118

   micro avg       0.98      0.98      0.98       171
   macro avg       0.98      0.96      0.97       171
weighted avg       0.98      0.98      0.98       171
```



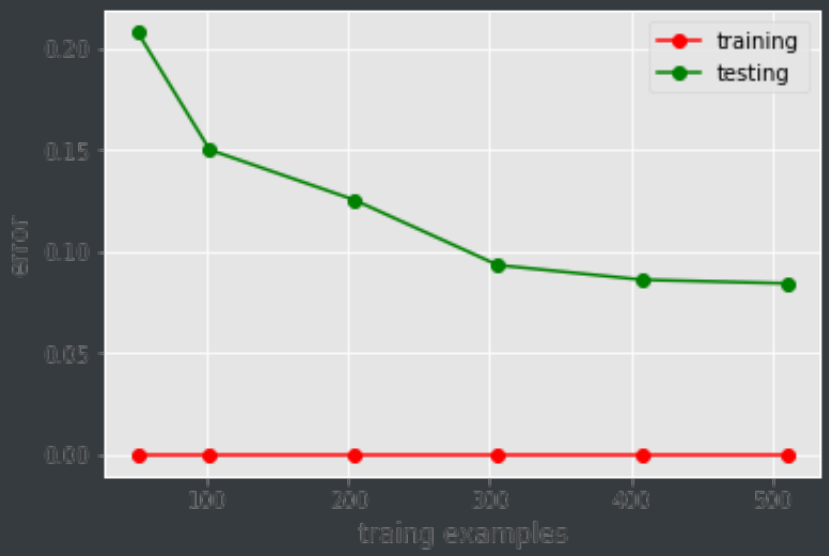

```
When max_depth = 8

The cross validation result:  [0.87931034 0.87931034 0.9122807  0.9122807
0.9122807  0.96491228
 0.92982456 0.94642857 0.94642857 0.94642857]

Confusion matrix:
[[ 49   4]
 [  0 118]]

Accuracy: 0.92 (+/- 0.06)
```

```
The kappa stats is:  0.9441541476159373

The MCC stats is:  0.9456298951208352

              precision    recall  f1-score   support

           0       0.98      1.00      0.99        53
           1       1.00      0.99      1.00       118

   micro avg       0.99      0.99      0.99       171
   macro avg       0.99      1.00      0.99       171
weighted avg       0.99      0.99      0.99       171
```




```
When max_depth = 9

The cross validation result:  [0.87931034 0.93103448 0.9122807  0.9122807
0.94736842 0.96491228
 0.92982456 0.92857143 0.94642857 0.92857143]

Confusion matrix:
```

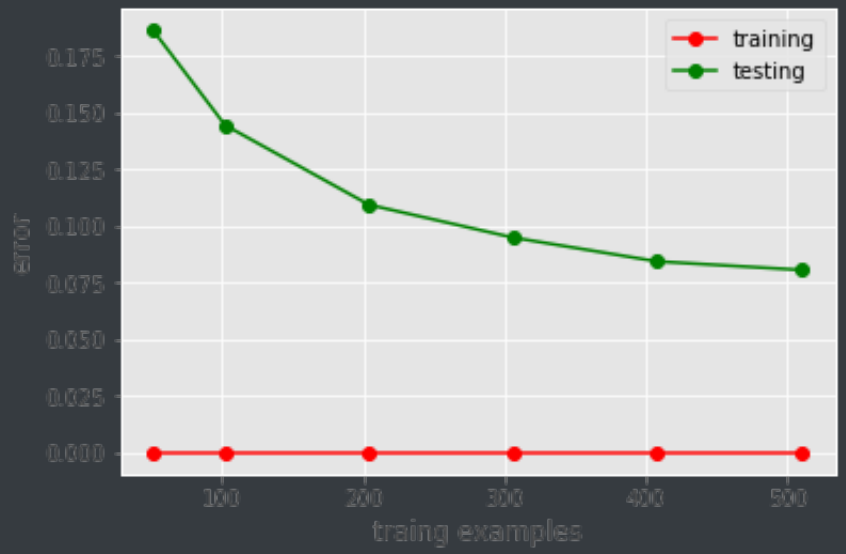
```
[[ 53   0]
 [  1 117]]

Accuracy: 0.93 (+/- 0.04)

The kappa stats is:  0.9863994273443092

The MCC stats is:  0.9864906699041353

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        59
           1       1.00      1.00      1.00       112

   micro avg       1.00      1.00      1.00       171
   macro avg       1.00      1.00      1.00       171
weighted avg       1.00      1.00      1.00       171
```




```
When max_depth = 10
```

```
The cross validation result:  [0.9137931  0.9137931  0.9122807  0.94736842
0.92982456 0.94736842
 0.92982456 0.89285714 0.92857143 0.98214286]

Confusion matrix:
[[ 59   0]
 [  0 112]]

Accuracy: 0.93 (+/- 0.05)

The kappa stats is:  1.0

The MCC stats is:  1.0

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        66
           1       1.00      1.00      1.00       105

   micro avg       1.00      1.00      1.00       171
   macro avg       1.00      1.00      1.00       171
weighted avg       1.00      1.00      1.00       171
```

```
When max_depth = 11

The cross validation result:  [0.87931034 0.89655172 0.9122807  0.9122807
0.89473684 0.94736842
 0.92982456 0.92857143 0.94642857 0.92857143]

Confusion matrix:
[[ 66   0]
 [  0 105]]

Accuracy: 0.92 (+/- 0.04)

The kappa stats is:  1.0

The MCC stats is:  1.0

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        58
           1       1.00      1.00      1.00       113

   micro avg       1.00      1.00      1.00       171
   macro avg       1.00      1.00      1.00       171
weighted avg       1.00      1.00      1.00       171
```



In order to set the model that provides the best predictive performance, I tested different possible value for the hyperparameter: max_depth. The evaluation metrics used in this case include The 'Accuracy rate', 'Kappa statistic', 'MCC statistic', 'Percision', 'Recall', 'F-measure'.

Generally speaking, Specifically, overfitting occurs if the model or algorithm shows low bias but high variance. Overfitting is often a result of an excessively complicated model. Underfitting occurs if the model or algorithm shows low variance but high bias. Underfitting is often a result of an excessively simple model. In this case, there is no obvious underfitting occurs. However, when the max_depth reaching to 6 and even larger, there do show a overfitting problem acorrding to the learning curves.

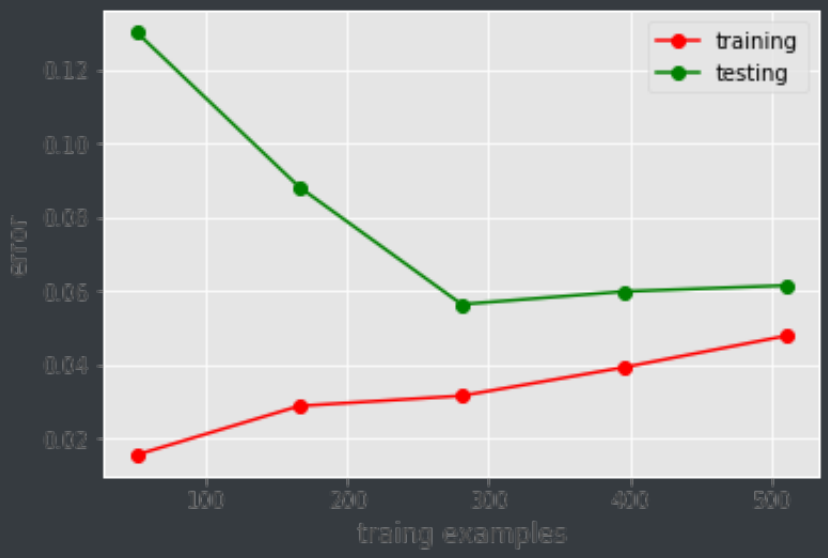
Take all the evaluation matrix into consideration, I think the model has a best performance when max_depth = 5.

2. Logistic regression model

```
lr = LogisticRegression(C=1e5,multi_class='multinomial',solver ='newton-cg')
lr = lr.fit(X, Y)

# cross validation
scores = cross_val_score(lr, X, Y, cv=10)
print("The cross validation result: ", scores)
```

```
print("Confusion matrix:\n%s" % metrics.confusion_matrix(Y_test, Y_pred))
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
print("The kappa stats is: ", cohen_kappa_score(Y_true, Y_pred))
print("The MCC stats is: ", matthews_corrcoef(Y_true, Y_pred))
#Splitting data into training set (70%) and testing set (30%)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
Y_true, Y_pred = Y_test, lr.predict(X_test)
# Classification report
print(classification_report(Y_true, Y_pred))
train_sizes,train_score,test_score = learning_curve(LogisticRegression(),X, Y,
cv=10, scoring='accuracy')
train_error = 1- np.mean(train_score,axis=1)
test_error = 1- np.mean(test_score,axis=1)
plt.plot(train_sizes,train_error,'o-',color = 'r',label = 'training')
plt.plot(train_sizes,test_error,'o-',color = 'g',label = 'testing')
plt.legend(loc='best')
plt.xlabel('traing examples')
plt.ylabel('error')
plt.show()
print ('\n\n\n')
```

```
The cross validation result:  [0.96551724 0.96551724 0.94736842 0.94736842
0.98245614 0.96491228
 0.94736842 0.92857143 0.94642857 0.94642857]
Confusion matrix:
[[ 48   6]
 [  3 114]]
Accuracy: 0.95 (+/- 0.03)
The kappa stats is:  0.8763557483731019
The MCC stats is:  0.877101011293492
              precision    recall  f1-score   support

           0       0.95      0.94      0.94        63
           1       0.96      0.97      0.97       108

   micro avg       0.96      0.96      0.96       171
   macro avg       0.96      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171
```



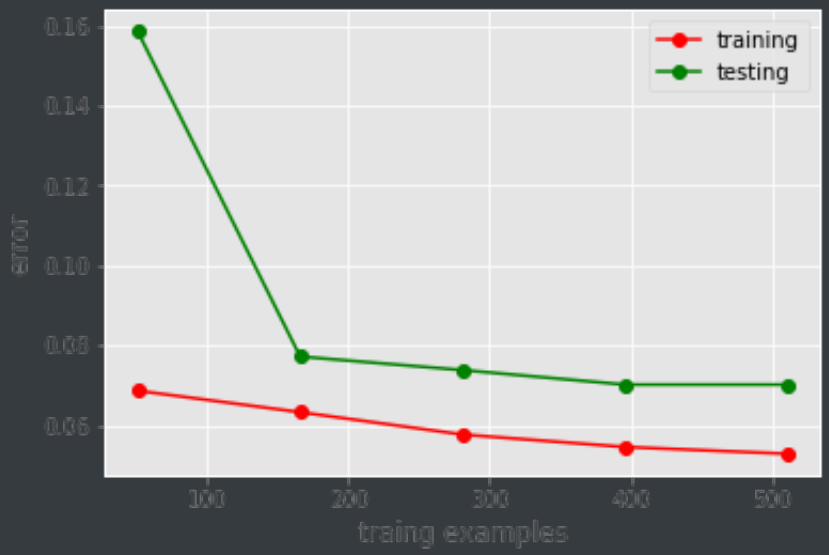
In the Logistic Regression, the learning graph shows there it is neither overfitting and underfitting in this case. And since this default set of hyperparameter already presents an accurate and good result, it is safe to say that lr = LogisticRegression(C=1e5,multi_class='multinomial',solver ='newton-cg') perform well.

3. KNN model

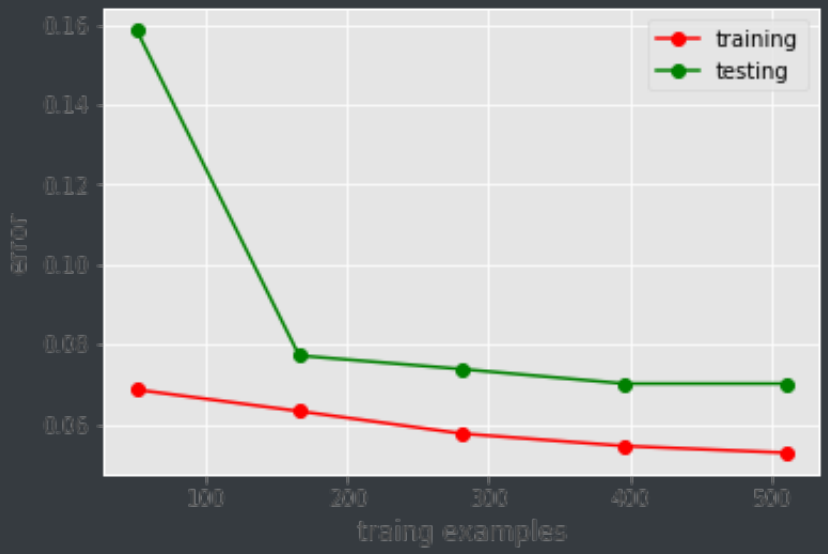
```
for n_neighbors in range(3,11):
    knn = neighbors.KNeighborsClassifier(n_neighbors)
    knn = knn.fit(X, Y)
    # cross validation
    scores = cross_val_score(knn, X, Y, cv=10)
    print("When neighbors =", n_neighbors)
    print("The cross validation result: ", scores)
    print("Confusion matrix:\n%s" % metrics.confusion_matrix(Y_test, Y_pred))
    print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
    print("The kappa stats is: ", cohen_kappa_score(Y_true, Y_pred))
```

```
    print("The MCC stats is: ", matthews_corrcoef(Y_true, Y_pred))
    #Splitting data into training set (70%) and testing set (30%)
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
    Y_true, Y_pred = Y_test, knn.predict(X_test)
    # Classification report
    print(classification_report(Y_true, Y_pred))
    train_sizes,train_score,test_score =
learning_curve(neighbors.KNeighborsClassifier(),X, Y, cv=10,
scoring='accuracy')
    train_error = 1- np.mean(train_score,axis=1)
    test_error = 1- np.mean(test_score,axis=1)
    plt.plot(train_sizes,train_error,'o-',color = 'r',label = 'training')
    plt.plot(train_sizes,test_error,'o-',color = 'g',label = 'testing')
    plt.legend(loc='best')
    plt.xlabel('traing examples')
    plt.ylabel('error')
    plt.show()
    print ('\n\n\n')
```

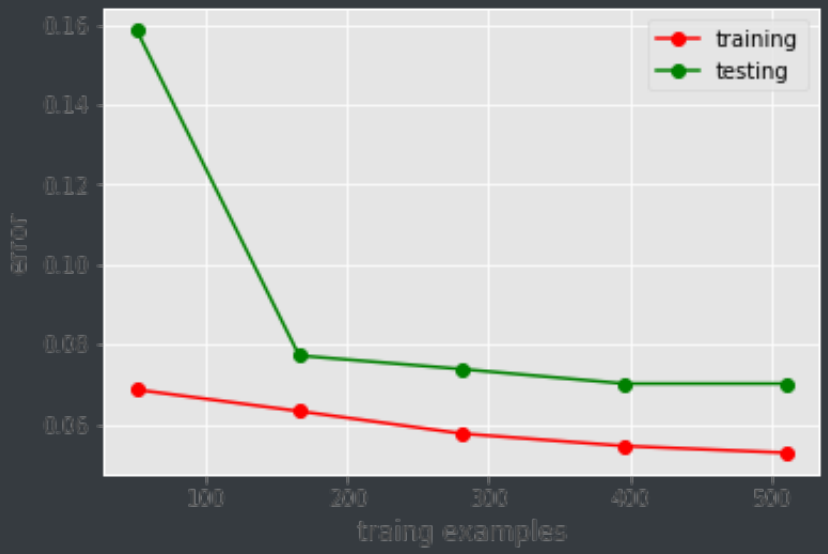
```
When neighbors = 3
The cross validation result:  [0.9137931  0.86206897 0.89473684 0.94736842
0.94736842 0.94736842
 0.96491228 0.94642857 0.89285714 0.92857143]
Confusion matrix:
[[ 60   4]
 [  1 106]]
Accuracy: 0.92 (+/- 0.06)
The kappa stats is:  0.9369794353947077
The MCC stats is:  0.9376499884661174
              precision    recall  f1-score   support

           0       0.95      0.82      0.88        66
           1       0.89      0.97      0.93       105

   micro avg       0.91      0.91      0.91       171
   macro avg       0.92      0.89      0.90       171
weighted avg       0.92      0.91      0.91       171
```



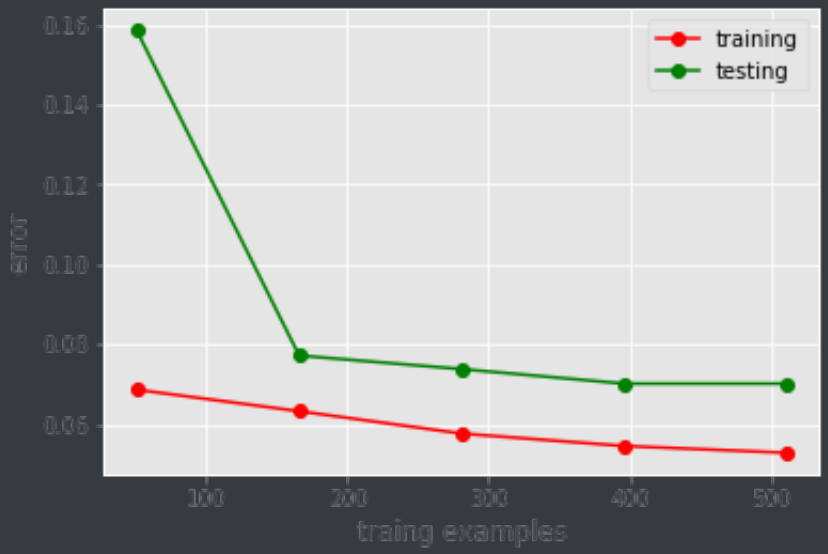

```
When neighbors = 4
The cross validation result:  [0.9137931  0.87931034 0.89473684 0.94736842
0.94736842 0.94736842
 0.96491228 0.91071429 0.875      0.92857143]
Confusion matrix:
[[ 54  12]
 [  3 102]]
Accuracy: 0.92 (+/- 0.06)
The kappa stats is:  0.810126582278481
The MCC stats is:  0.8154355063002009
              precision    recall  f1-score   support

           0       0.91      0.97      0.94        71
           1       0.98      0.93      0.95       100

   micro avg       0.95      0.95      0.95       171
   macro avg       0.94      0.95      0.95       171
weighted avg       0.95      0.95      0.95       171
```



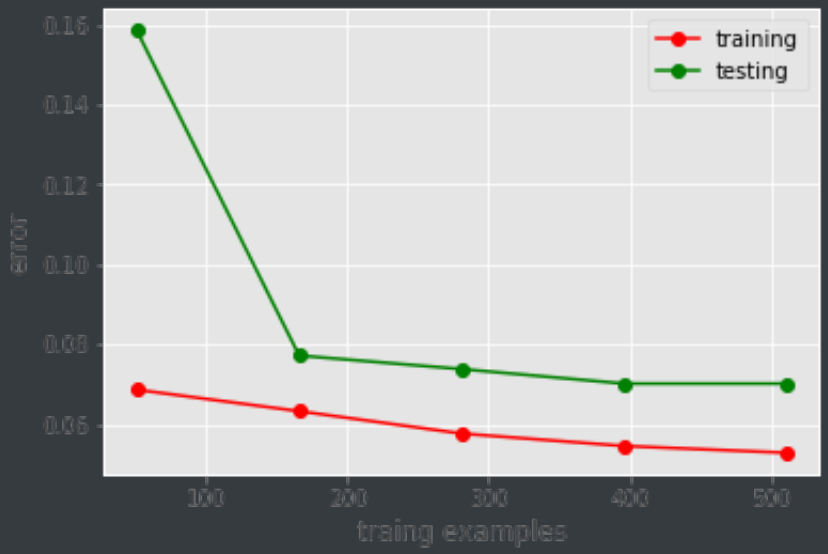

```
When neighbors = 5
The cross validation result:  [0.9137931  0.87931034 0.89473684 0.96491228
0.94736842 0.92982456
 0.96491228 0.92857143 0.91071429 0.96428571]
Confusion matrix:
[[69  2]
 [ 7 93]]
Accuracy: 0.93 (+/- 0.06)
The kappa stats is:  0.8927152317880794
The MCC stats is:  0.8943051437517539
              precision    recall  f1-score   support

           0       0.96      0.91      0.93        55
           1       0.96      0.98      0.97       116

   micro avg       0.96      0.96      0.96       171
   macro avg       0.96      0.95      0.95       171
weighted avg       0.96      0.96      0.96       171
```



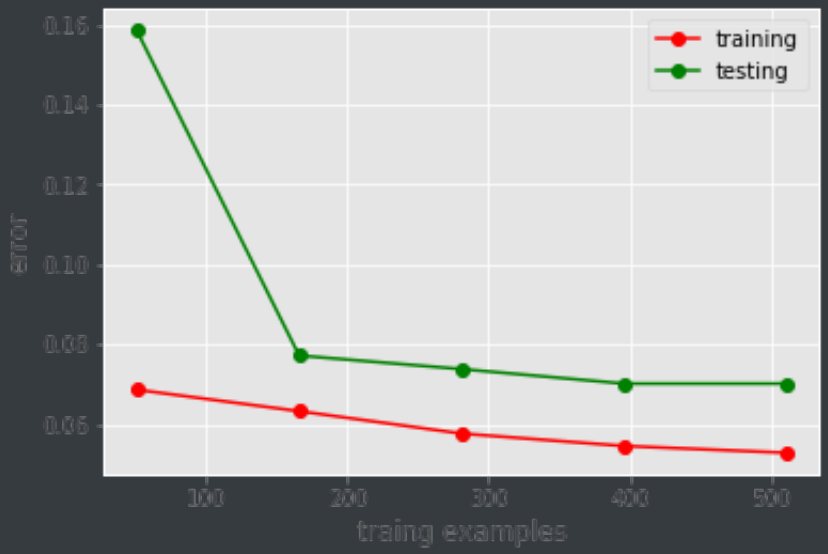

```
When neighbors = 6
The cross validation result:  [0.9137931  0.87931034 0.89473684 0.96491228
0.94736842 0.92982456
 0.96491228 0.92857143 0.89285714 0.94642857]
Confusion matrix:
[[ 50   5]
 [  2 114]]
Accuracy: 0.93 (+/- 0.06)
The kappa stats is:  0.9048262701757176
The MCC stats is:  0.9055799001029633
              precision    recall  f1-score   support

           0       0.96      0.91      0.94        57
           1       0.96      0.98      0.97       114

   micro avg       0.96      0.96      0.96       171
   macro avg       0.96      0.95      0.95       171
weighted avg       0.96      0.96      0.96       171
```



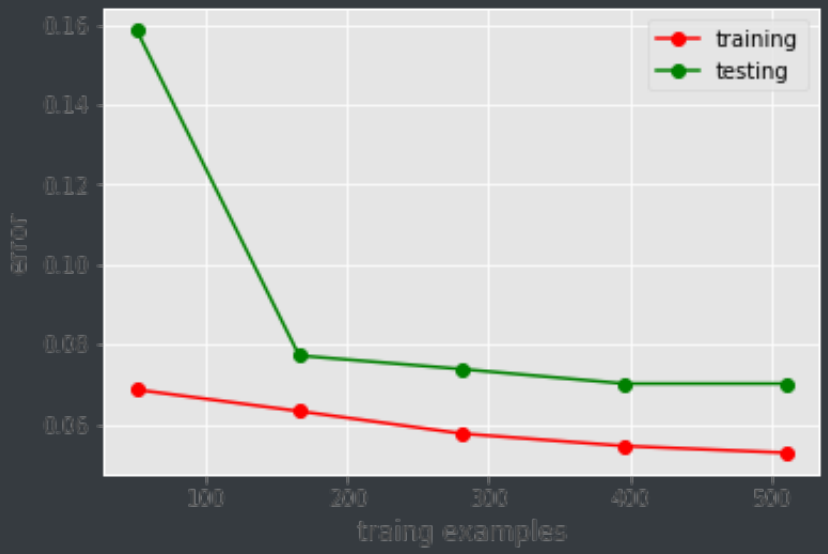

```
When neighbors = 7
The cross validation result:  [0.93103448 0.86206897 0.9122807  0.96491228
0.92982456 0.92982456
 0.96491228 0.92857143 0.91071429 0.94642857]
Confusion matrix:
[[ 52   5]
 [  2 112]]
Accuracy: 0.93 (+/- 0.06)
The kappa stats is:  0.9066666666666666
The MCC stats is:  0.9073928715621604
              precision    recall  f1-score   support

           0       0.95      0.90      0.92        61
           1       0.95      0.97      0.96       110

   micro avg       0.95      0.95      0.95       171
   macro avg       0.95      0.94      0.94       171
weighted avg       0.95      0.95      0.95       171
```



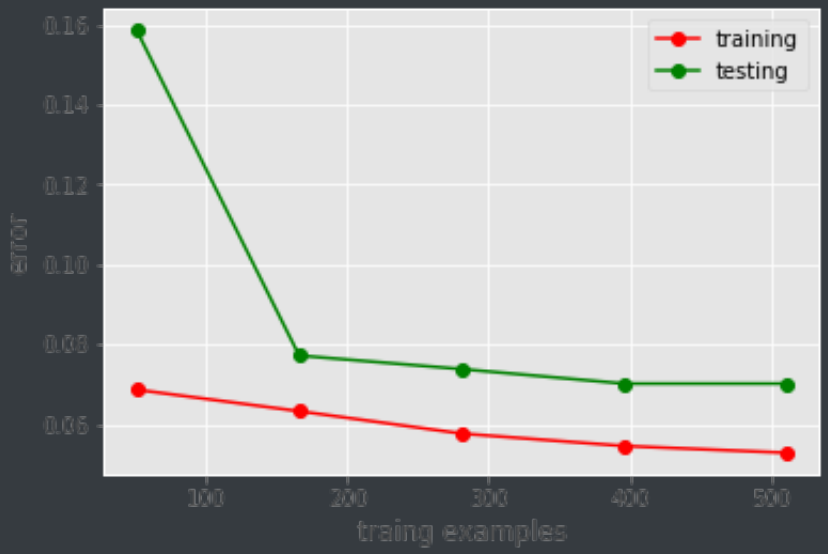

```
When neighbors = 8
The cross validation result:  [0.93103448 0.86206897 0.9122807  0.96491228
0.94736842 0.92982456
 0.96491228 0.92857143 0.89285714 0.96428571]
Confusion matrix:
[[ 55   6]
 [  3 107]]
Accuracy: 0.93 (+/- 0.06)
The kappa stats is:  0.8840503277329917
The MCC stats is:  0.8847113718356191
              precision    recall  f1-score   support

           0       0.94      0.92      0.93        63
           1       0.95      0.96      0.96       108

   micro avg       0.95      0.95      0.95       171
   macro avg       0.94      0.94      0.94       171
weighted avg       0.95      0.95      0.95       171
```




```
When neighbors = 9
The cross validation result:  [0.89655172 0.86206897 0.9122807  0.96491228
0.94736842 0.92982456
 0.96491228 0.92857143 0.91071429 0.96428571]
Confusion matrix:
[[ 58   5]
 [  4 104]]
Accuracy: 0.93 (+/- 0.06)
The kappa stats is:  0.8865295288652952
The MCC stats is:  0.8865999974723524
              precision    recall  f1-score   support

           0       0.92      0.86      0.89        65
           1       0.92      0.95      0.94       106

   micro avg       0.92      0.92      0.92       171
   macro avg       0.92      0.91      0.91       171
weighted avg       0.92      0.92      0.92       171
```




```
When neighbors = 10
The cross validation result:  [0.9137931  0.86206897 0.9122807  0.96491228
0.96491228 0.92982456
 0.96491228 0.94642857 0.89285714 0.96428571]
Confusion matrix:
[[ 56   9]
 [  5 101]]
Accuracy: 0.93 (+/- 0.07)
The kappa stats is:  0.824177438307873
The MCC stats is:  0.8252193400671107
              precision    recall  f1-score   support

           0       0.95      0.88      0.92        68
           1       0.93      0.97      0.95       103

   micro avg       0.94      0.94      0.94       171
   macro avg       0.94      0.93      0.93       171
weighted avg       0.94      0.94      0.94       171
```



In the KNN models, no matter how I change the hyperparameter n_neighbors, the learning curve show that the model do not have the overfitting and underfitting problem with in these different models. According to the accuracy rate , kappa, mcc and F-measure, the KNN model has the best result when the number of neighbors = 6.