In [3]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn import ensemble
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

import numpy as np
from sklearn import datasets
```

In [4]:
```
# Load data
cal = datasets.fetch_california_housing()
data = cal['data']
targets = cal['target']
```

In [6]:
```

```

Out[6]:
```
array([ 8.30140000e+00,  2.10000000e+01,  6.23813708e+00,  9.71880492e-01,
        2.40100000e+03,  2.10984183e+00,  3.78600000e+01, -1.22220000e+02])
```

Task 1

In [8]:
```
# 1a
clf = LinearRegression()
scores = cross_val_score(clf, data, targets, cv=5)
print('Linear Regression R^2 Scores', scores)
print('Linear Regression Mean R^2 Score', np.mean(scores))
print()

clf = ensemble.GradientBoostingRegressor()
scores = cross_val_score(clf, data, targets, cv=5)
print('Boosting R^2 Scores', scores)
print('Boosting Scores Mean R^2 Score', np.mean(scores))
```

```
Linear Regression R^2 Scores [0.54866323 0.46820691 0.55078434 0.53698703 0.6
6051406]
Linear Regression Mean R^2 Score 0.5530311140279233

Boosting R^2 Scores [0.60256286 0.69877396 0.7180343  0.65023363 0.67979733]
Boosting Scores Mean R^2 Score 0.6698804157532645
```

In [4]:
```
# 1b
tuned_parameters = [{'max_depth': [3, 10],
                     'n_estimators': [50, 100],
                     'learning_rate': [0.01, 0.1]}]
clf = ensemble.GradientBoostingRegressor()
clf = GridSearchCV(clf, tuned_parameters)
clf.fit(data, targets)

print("Scores for parameter grid search:")
print()
means = clf.cv_results_['mean_test_score']
stds = clf.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, clf.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))
```

```
Scores for parameter grid search:

0.280 (+/-0.105) for {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators':
50}
0.435 (+/-0.121) for {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators':
100}
0.374 (+/-0.053) for {'learning_rate': 0.01, 'max_depth': 10, 'n_estimators':
50}
0.542 (+/-0.077) for {'learning_rate': 0.01, 'max_depth': 10, 'n_estimators':
100}
0.633 (+/-0.094) for {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 5
0}
0.670 (+/-0.081) for {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 1
00}
0.652 (+/-0.139) for {'learning_rate': 0.1, 'max_depth': 10, 'n_estimators':
50}
0.659 (+/-0.129) for {'learning_rate': 0.1, 'max_depth': 10, 'n_estimators':
100}
```

1c) Briefly discuss the performance and summarize your findings.

When running linear regression and gradient boosting using 5-fold cross validation, we got the following R^2 Score results:

```
Linear Regression Mean Score = 0.5530311140279233
Boosting Scores Mean Score = 0.6698681752149087
```

Overall the scores are not great. R^2 measures the goodness-of-fit of the model on the data. R^2 closer to 1 is better. The boosting score is significantly greater than the linear regressor which makes sense.

For 1b) we tested all of the following possible permutations:

```
'max_depth': [3, 10],
'n_estimators': [50, 100],
'learning_rate': [0.01, 0.1]
```

Our parameter grid search yielded interesting results. None of the parameter combinations were convincingly better than the default parameters. In fact, increasing the max_depth and decreasing the learning rate were very detrimental to the R^2 score.

Task 2

In [10]:
```
new_targets = np.array([x>2 for x in targets])
```

In [12]:

```python
# 2a
clf = LogisticRegression()
scores = cross_val_score(clf, data, new_targets, cv=5)
print('Logistic Regression Accuracy', scores)
print('Logistic Regression Mean Accuracy', np.mean(scores))
print()

clf = ensemble.GradientBoostingClassifier()
scores = cross_val_score(clf, data, new_targets, cv=5)
print('Boosting Classifier Accuracy', scores)
print('Boosting Classifier Mean Accuracy', np.mean(scores))
```

```
/Users/David/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/l
ogistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/David/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/l
ogistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/David/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/l
ogistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/David/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/l
ogistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/David/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/l
ogistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)

Logistic Regression Accuracy [0.80988133 0.79796512 0.77616279 0.74612403 0.8
2481221]
Logistic Regression Mean Accuracy 0.7909890954886174

Boosting Classifier Accuracy [0.79099055 0.75436047 0.80741279 0.75314922 0.8
2650836]
Boosting Classifier Mean Accuracy 0.786484278963419
```

In [7]:
```
# 2b
tuned_parameters = [{'max_depth': [3, 5],
                     'n_estimators': [100, 200],
                     'learning_rate': [0.1, 0.5]}]
clf = ensemble.GradientBoostingClassifier()
clf = GridSearchCV(clf, tuned_parameters)
clf.fit(data, new_targets)

print("Scores for parameter grid search:")
print()
means = clf.cv_results_['mean_test_score']
stds = clf.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, clf.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))
```

```
Scores for parameter grid search:

0.788 (+/-0.059) for {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 1
00}
0.774 (+/-0.095) for {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 2
00}
0.765 (+/-0.100) for {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 1
00}
0.752 (+/-0.125) for {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 2
00}
0.751 (+/-0.130) for {'learning_rate': 0.5, 'max_depth': 3, 'n_estimators': 1
00}
0.748 (+/-0.139) for {'learning_rate': 0.5, 'max_depth': 3, 'n_estimators': 2
00}
0.741 (+/-0.125) for {'learning_rate': 0.5, 'max_depth': 5, 'n_estimators': 1
00}
0.742 (+/-0.136) for {'learning_rate': 0.5, 'max_depth': 5, 'n_estimators': 2
00}
```

In [14]:
```python
# 2c
clf = LogisticRegression()
scores = cross_val_score(clf, data, new_targets, cv=5, scoring='roc_auc')
print('Logistic Regression ROC AUC scores', scores)
print('Logistic Regression ROC AUC Mean score', np.mean(scores))
print()

clf = ensemble.GradientBoostingClassifier()
scores = cross_val_score(clf, data, new_targets, cv=5, scoring='roc_auc')
print('Boosting Classifier ROC AUC scores', scores)
print('Boosting Classifier ROC AUC Mean score', np.mean(scores))
```

```
/Users/David/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/l
ogistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/David/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/l
ogistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/David/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/l
ogistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/David/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/l
ogistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/David/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/l
ogistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)

Logistic Regression ROC AUC scores [0.88418515 0.88251405 0.86366318 0.819350
2  0.90110991]
Logistic Regression ROC AUC Mean score 0.8701644975389575

Boosting Classifier ROC AUC scores [0.88490914 0.84740781 0.90563496 0.897960
67 0.91313523]
Boosting Classifier ROC AUC Mean score 0.8898095631008707
```

2d) Briefly discuss the performance and summarize your findings. Are they good classifiers? Compare the result with a trivial classifier. Compare the results when using accuracy and ROC_AUC.

The performance of the logistic regression and gradient boosting clasifiers was pretty good. The mean accuracy of the models on 5-fold cross validation is as follows:

```
Logistic Regression Mean Accuracy = 0.7909890954886174
Boosting Classifier Mean Accuracy = 0.786484278963419
```

The accuracies were roughly the same with the logistic regression having a slightly better mean accuracy.

The results of ROC_AUC were:

```
Logistic Regression ROC AUC Mean score 0.8701644975389575
Boosting Classifier ROC AUC Mean score 0.8898095631008707
```

ROC_AUC measures the ability between [0,1] of a classifier to discriminate between classes of data. Values closer to 1 mean the model is good at discriminating classes. Given the results above, we can say that our model can predict the class of data relatively well. The two classifiers performed similarly. The boosting classifier has a slightly greater ROC AUC mean score.

In [ ]:

In [ ]: