Question 1

According to ESL 9.2 (Beezer et al.), both the Gini index and cross-entropy are preferable when growing the tree since they can represent the purity better than error rate. However, it is better to use error rate when pruning the tree because it will be more efficient.

Question 2

Load packages

```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import ElasticNet, LogisticRegressionCV
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn import metrics
```

Load data

```
train = pd.read_csv('gss_train.csv')
test = pd.read_csv('gss_test.csv')
X_train = train.drop('colrac', axis = 1)
y_train = train['colrac']
X_test = test.drop('colrac', axis = 1)
y_test = test['colrac']
```

Logistic regression

```
lgs = LogisticRegressionCV(cv = 10)
model_1 = lgs.fit(X_train, y_train)
```

Elastic net

```
parameters = {'alpha': np.arange(0.0, 1.0, 0.01),
             'l1_ratio': np.arange(0.0, 10.0, 0.1)}
eln = ElasticNet()
model_2 = GridSearchCV(estimator = eln, param_grid = parameters, cv = 10)
model_2.fit(X_train, y_train)
model_2.get_params(deep = False)['estimator']
```

```
ElasticNet(alpha=1.0, copy_X=True, fit_intercept=True, l1_ratio=0.5,
      max_iter=1000, normalize=False, positive=False, precompute=False,
      random_state=None, selection='cyclic', tol=0.0001, warm_start=False)
```

CART

```
parameters = {'min_samples_split': range(10,500,20), 'max_depth':
range(1,20,2)}
dt = DecisionTreeClassifier()
model_3 = GridSearchCV(dt, parameters, cv = 10, scoring='accuracy')
model_3.fit(X_train, y_train)
model_3.get_params(deep = False)['estimator']
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            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=None,
            splitter='best')
```

Naive bayes

```
parameters = {}
nb = GaussianNB()
model_4 = GridSearchCV(nb, parameters, cv = 10, scoring='accuracy')
model_4.fit(X_train, y_train)
model_4.get_params(deep = False)['estimator']
```

```
GaussianNB(priors=None)
```

Random Forest

```
parameters = {'n_estimators': range(10, 100, 10), 'min_samples_leaf': range(2, 
10, 2),
              'max_features': range(2, 54, 3), 'max_leaf_nodes': range(2, 30, 
3)}
rf = RandomForestClassifier()
model_5 = GridSearchCV(rf, parameters, cv = 10, scoring='accuracy')
model_5.fit(X_train, y_train)
model_5.get_params(deep = False)['estimator']
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            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=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

Bagging

```
parameters = {'n_estimators': range(5, 50, 5)}
bg = BaggingClassifier(base_estimator = DecisionTreeClassifier())
model_6 = GridSearchCV(bg, parameters, cv = 10, scoring='accuracy')
model_6.fit(X_train, y_train)
model_6.get_params(deep = False)['estimator']
```

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(class_weight=None,
criterion='gini', max_depth=None,
            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=None,
            splitter='best'),
         bootstrap=True, bootstrap_features=False, max_features=1.0,
         max_samples=1.0, n_estimators=10, n_jobs=1, oob_score=False,
         random_state=None, verbose=0, warm_start=False)
```

Boosting

```
parameters = {'learning_rate': np.arange(0.1, 0.9, 0.1), 'n_estimators':
range(10, 500, 50), 'max_depth': range(2, 8)}
bs = GradientBoostingClassifier()
model_7 = GridSearchCV(bs, parameters, cv = 10, scoring='accuracy')
model_7.fit(X_train, y_train)
model_7.get_params(deep = False)['estimator']
```

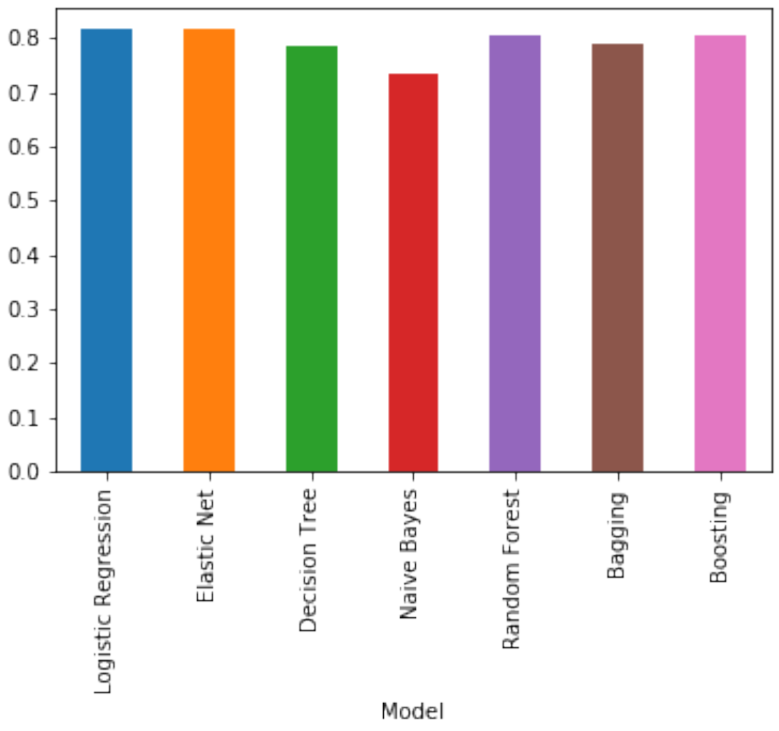
```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              presort='auto', random_state=None, subsample=1.0, verbose=0,
              warm_start=False)
```

Question 3

CV Score

```
model_2_pre =  model_2.predict(X_train)
model_2_pre[model_2_pre >= 0.5] = 1
model_2_pre[model_2_pre <= 0.5] = 0
scores = pd.DataFrame({'Model': ['Logistic Regression','Elastic Net',
'Decision Tree', 'Naive Bayes', 'Random Forest',
                               'Bagging', 'Boosting'],
                     'Score': [accuracy_score(y_train,
model_1.predict(X_train)), accuracy_score(y_train, model_2_pre),
model_3.best_score_,
                              model_4.best_score_, model_5.best_score_,
model_6.best_score_,
                              model_7.best_score_]})
scores.plot(kind = 'bar', x = 'Model', y = 'Score', legend=False)
scores
```

	Model	Score
0	Logistic Regression	0.816396
1	Elastic Net	0.817073
2	Decision Tree	0.785908
3	Naive Bayes	0.734417
4	Random Forest	0.805556
5	Bagging	0.787940
6	Boosting	0.804878



ROC/AUC

```
fpr, tpr, thresholds = metrics.roc_curve(y_train, model_1.predict(X_train),
pos_label=1)
print('The AUC of Logistic Regression is {}'.format(metrics.auc(fpr, tpr)))

fpr, tpr, thresholds = metrics.roc_curve(y_train, model_2.predict(X_train),
pos_label=1)
print('The AUC of Elastic Net is {}'.format(metrics.auc(fpr, tpr)))

fpr, tpr, thresholds = metrics.roc_curve(y_train, model_3.predict(X_train),
pos_label=1)
print('The AUC of Decision Tree is {}'.format(metrics.auc(fpr, tpr)))

fpr, tpr, thresholds = metrics.roc_curve(y_train, model_4.predict(X_train),
pos_label=1)
print('The AUC of Naive Bayes is {}'.format(metrics.auc(fpr, tpr)))

fpr, tpr, thresholds = metrics.roc_curve(y_train, model_5.predict(X_train),
pos_label=1)
print('The AUC of Random Forest is {}'.format(metrics.auc(fpr, tpr)))

fpr, tpr, thresholds = metrics.roc_curve(y_train, model_6.predict(X_train),
pos_label=1)
print('The AUC of Bagging is {}'.format(metrics.auc(fpr, tpr)))

fpr, tpr, thresholds = metrics.roc_curve(y_train, model_7.predict(X_train),
pos_label=1)
print('The AUC of Boosting is {}'.format(metrics.auc(fpr, tpr)))
```

```
The AUC of Logistic Regression is 0.8160351571487735
The AUC of Elastic Net is 0.8910809442731581
The AUC of Decision Tree is 0.7820993051401224
The AUC of Naive Bayes is 0.7431245685886521
The AUC of Random Forest is 0.8634043532281073
The AUC of Bagging is 0.9993548387096773
The AUC of Boosting is 1.0
```

Question 4

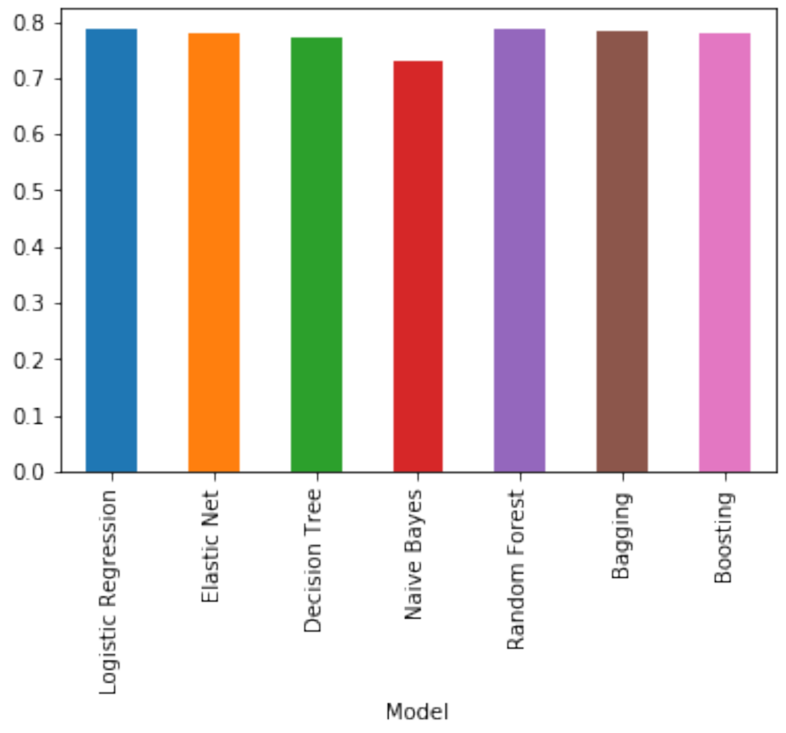
From the cross validation error rate, I find that Elastic Net and Logistic Regression have the best performance. However, the AUC score shows that Bagging and Boosting are the best models. As for the best model, I think Elastic Net shows both high accuracy and AUC score so I want to pick it.

Question 5

Error Rate

```
model_2_pre =  model_2.predict(X_test)
model_2_pre[model_2_pre >= 0.5] = 1
model_2_pre[model_2_pre <= 0.5] = 0
scores = pd.DataFrame({'Model': ['Logistic Regression','Elastic Net',
'Decision Tree', 'Naive Bayes', 'Random Forest',
                               'Bagging', 'Boosting'],
                     'Score': [accuracy_score(y_test,
model_1.predict(X_test)), accuracy_score(y_test, model_2_pre),
                              accuracy_score(y_test, model_3.predict(X_test)),
accuracy_score(y_test, model_4.predict(X_test)),
                              accuracy_score(y_test, model_5.predict(X_test)),
accuracy_score(y_test, model_6.predict(X_test)),
                              accuracy_score(y_test,
model_7.predict(X_test))]})
scores.plot(kind = 'bar', x = 'Model', y = 'Score', legend=False)
scores
```

	Model	Score
0	Logistic Regression	0.787018
1	Elastic Net	0.778905
2	Decision Tree	0.772819
3	Naive Bayes	0.728195
4	Random Forest	0.784990
5	Bagging	0.782961
6	Boosting	0.780933



ROC/AUC

```
fpr, tpr, thresholds = metrics.roc_curve(y_test, model_1.predict(X_test), pos_label=1)
print('The AUC of Logistic Regression is {}'.format(metrics.auc(fpr, tpr)))

fpr, tpr, thresholds = metrics.roc_curve(y_test, model_2.predict(X_test), pos_label=1)
print('The AUC of Elastic Net is {}'.format(metrics.auc(fpr, tpr)))

fpr, tpr, thresholds = metrics.roc_curve(y_test, model_3.predict(X_test), pos_label=1)
print('The AUC of Decision Tree is {}'.format(metrics.auc(fpr, tpr)))

fpr, tpr, thresholds = metrics.roc_curve(y_test, model_4.predict(X_test), pos_label=1)
print('The AUC of Naive Bayes is {}'.format(metrics.auc(fpr, tpr)))
```

```
fpr, tpr, thresholds = metrics.roc_curve(y_test, model_5.predict(X_test), pos_label=1)
print('The AUC of Random Forest is {}'.format(metrics.auc(fpr, tpr)))

fpr, tpr, thresholds = metrics.roc_curve(y_test, model_6.predict(X_test), pos_label=1)
print('The AUC of Bagging is {}'.format(metrics.auc(fpr, tpr)))

fpr, tpr, thresholds = metrics.roc_curve(y_test, model_7.predict(X_test), pos_label=1)
print('The AUC of Boosting is {}'.format(metrics.auc(fpr, tpr)))
```

```
The AUC of Logistic Regression is 0.7822906322409798
The AUC of Elastic Net is 0.8599139357828534
The AUC of Decision Tree is 0.764490235021516
The AUC of Naive Bayes is 0.726961271102284
The AUC of Random Forest is 0.7773419397550481
The AUC of Bagging is 0.7760675273088382
The AUC of Boosting is 0.7750993048659385
```

From the prediction result of the test set, I find Logistic Regression still maintains high accuracy compared with other models. Thus, Logistic Regression has good generalization ability and it should be the best model. However, Elastic Net does not keep a high accuracy, which means its generalization ability is bad. Plus, the AUC score of Bagging and Boosting become lower, which is close to 1 in the training set. Therefore, I can conclude Bagging and Boosting also show low generalization ability.