In [18]:

```
#Import libraries:
import pandas as pd
import numpy as np
import xgboost as xgb
from xgboost.sklearn import XGBClassifier
from sklearn import metrics   #Additional scklearn functions
from sklearn.model_selection import cross_validate
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV, train_test_split  #Perforing
grid search
from sklearn.metrics import f1_score

import matplotlib.pylab as plt
%matplotlib inline
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 12, 4
```

In [32]:

```
#import data set
train = pd.read_csv('../data/TrainData.csv')
target = 'NEXT_MONTH_DEFAULT'
IDcol = 'Client_ID'
```

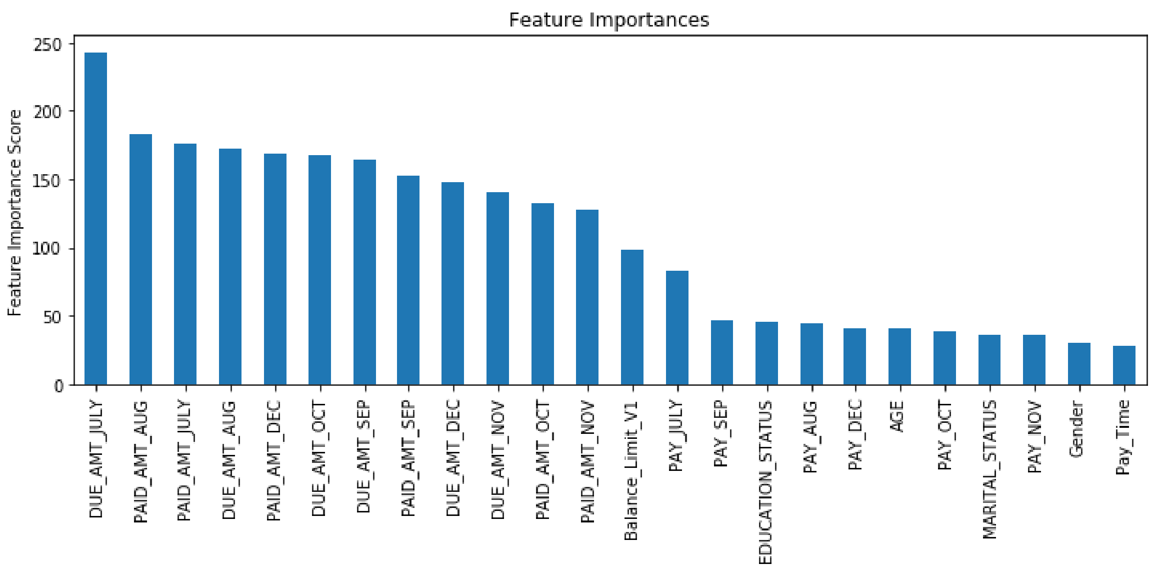
In [26]:

```
#function to fit model, calculate scores and save predictions for test set

def modelfit(alg, dtrain, predictors,useTrainCV=True, cv_folds=5, early_stopping_rounds=50):

    if useTrainCV:
        xgb_param = alg.get_xgb_params()
        xgtrain = xgb.DMatrix(dtrain[predictors].values, label=dtrain[target].values)
        cvresult = xgb.cv(xgb_param, xgtrain, num_boost_round=alg.get_params()['n_estimators'], nfo
ld=cv_folds,
            metrics='auc', early_stopping_rounds=early_stopping_rounds, verbose_eval=None)
        alg.set_params(n_estimators=cvresult.shape[0])

    #Fit the algorithm on the data
    alg.fit(dtrain[predictors], dtrain['NEXT_MONTH_DEFAULT'],eval_metric='auc')

    #Predict training set:
    dtrain_predictions = alg.predict(dtrain[predictors])
    dtrain_predprob = alg.predict_proba(dtrain[predictors])[:,1]

    #Print model report:
    print("\nModel Report")
    print ("Accuracy : %.4g" % metrics.accuracy_score(dtrain['NEXT_MONTH_DEFAULT'].values, dtrain_p
redictions))
    print ("AUC Score (Train): %f" % metrics.roc_auc_score(dtrain['NEXT_MONTH_DEFAULT'], dtrain_pre
dprob))

    feat_imp = pd.Series(alg.get_booster().get_fscore()).sort_values(ascending=False)
    feat_imp.plot(kind='bar', title='Feature Importances')
    plt.ylabel('Feature Importance Score')


    #importing test data
    Test = pd.read_csv('TestData.csv')

    #making copy to work with
    Test_copy = Test.copy()

    Test_copy= Test_copy.drop(['Client_ID'], axis=1)

    #predicting values for test set
    y_pred=alg.predict(Test_copy)

    #Create a  DataFrame with the client ids and our prediction
    submission = pd.DataFrame({'Client_ID':Test['Client_ID'],'NEXT_MONTH_DEFAULT':y_pred})
    #Convert DataFrame to a csv file that can be uploaded
    filename = 'XGBoost Preds 1.csv'

    submission.to_csv(filename,index=False)

    print('Saved file: ' + filename)
```

In [27]:

```
#Choose all predictors except target & IDcols
predictors = [x for x in train.columns if x not in [target, IDcol]]
xgb1 = XGBClassifier(
 learning_rate =0.099,
 max_depth=5,
 min_child_weight=1,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27)
modelfit(xgb1, train, predictors)
```

```
Model Report
Accuracy : 0.8359
AUC Score (Train): 0.844488
Saved file: XGBoost Preds 1.csv
```



Parameter Tuning

In [65]:

```
#tuning for max depth and min child weight

param_test1 = {
 'max_depth':range(3,10,2),
 'min_child_weight':range(1,6,2)
}
gsearch1 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, max_depth=5,
 min_child_weight=1, gamma=0, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1, seed=27),
 param_grid = param_test1, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch1.fit(train[predictors],train[target])
gsearch1.cv_results_, gsearch1.best_params_, gsearch1.best_score_
```

```
C:\ProgramData\Miniconda3\envs\sample1\lib\site-packages\sklearn\model_selection\_search.py:825: F
utureWarning: The parameter 'iid' is deprecated in 0.22 and will be removed in 0.24.
  "removed in 0.24.", FutureWarning
```

Out[65]:

```
({'mean_fit_time': array([1.28496814, 1.26793776, 1.24309297, 2.01788125, 2.00223627,
         1.96888356, 2.92638159, 2.81732163, 2.80824142, 4.04019103,
         3.90401125, 3.4892643 ]),
  'std_fit_time': array([0.01255557, 0.01193091, 0.01764072, 0.02580398, 0.02768251,
         0.05034188, 0.03143239, 0.02544744, 0.03533723, 0.07286999,
         0.07589951, 0.3208887 ]),
  'mean_score_time': array([0.01591697, 0.01562552, 0.01562576, 0.02334609, 0.01631775,
         0.02144179, 0.01905894, 0.03119469, 0.02122045, 0.02499866,
         0.03124719, 0.02187328]),
  'std_score_time': array([3.10022736e-03, 1.60574515e-06, 4.37028474e-07, 6.98917663e-03,
         2.07700090e-03, 5.02415769e-03, 6.86860088e-03, 9.88186048e-03,
         6.93209163e-03, 7.65414932e-03, 8.97163759e-07, 7.65449975e-03]),
  'param_max_depth': masked_array(data=[3, 3, 3, 5, 5, 5, 7, 7, 7, 9, 9, 9],
               mask=[False, False, False, False, False, False, False, False,
                     False, False, False, False],
         fill_value='?',
              dtype=object),
  'param_min_child_weight': masked_array(data=[1, 3, 5, 1, 3, 5, 1, 3, 5, 1, 3, 5],
               mask=[False, False, False, False, False, False, False, False,
                     False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'max_depth': 3, 'min_child_weight': 1},
   {'max_depth': 3, 'min_child_weight': 3},
   {'max_depth': 3, 'min_child_weight': 5},
   {'max_depth': 5, 'min_child_weight': 1},
   {'max_depth': 5, 'min_child_weight': 3},
   {'max_depth': 5, 'min_child_weight': 5},
   {'max_depth': 7, 'min_child_weight': 1},
   {'max_depth': 7, 'min_child_weight': 3},
   {'max_depth': 7, 'min_child_weight': 5},
   {'max_depth': 9, 'min_child_weight': 1},
   {'max_depth': 9, 'min_child_weight': 3},
   {'max_depth': 9, 'min_child_weight': 5}],
  'split0_test_score': array([0.75670726, 0.75497129, 0.75570134, 0.75441545, 0.75404095,
         0.75547023, 0.74477641, 0.74934791, 0.74920832, 0.74365697,
         0.74611417, 0.75175889]),
  'split1_test_score': array([0.75350054, 0.75201974, 0.75238583, 0.7514889 , 0.7525919 ,
         0.75372109, 0.75155246, 0.74777485, 0.75337948, 0.7453904 ,
         0.74571359, 0.74901681]),
  'split2_test_score': array([0.79021156, 0.79110296, 0.7911123 , 0.788858  , 0.78660422,
         0.78775934, 0.78345404, 0.78423833, 0.78208851, 0.78274687,
         0.78223313, 0.78283135]),
  'split3_test_score': array([0.80225249, 0.80292032, 0.80199693, 0.80228157, 0.8019293 ,
         0.80069852, 0.79748991, 0.79761387, 0.79480175, 0.79646599,
         0.78961939, 0.79424972]),
  'split4_test_score': array([0.78889629, 0.78905802, 0.78965334, 0.78925304, 0.78977322,
         0.79020787, 0.78299913, 0.78870048, 0.78868232, 0.78136147,
         0.78224576, 0.78380289]),
  'mean_test_score': array([0.77831363, 0.77801446, 0.77816995, 0.77725939, 0.77698792,
         0.77757141, 0.77205439, 0.77353509, 0.77363208, 0.76992434,
         0.76918521, 0.77233193]),
  'std_test_score': array([0.01954045, 0.02059264, 0.02018295, 0.02044724, 0.01999869,
         0.01926467, 0.02030331, 0.0208468 , 0.01872359, 0.02140828,
         0.01919152, 0.01837964]),
  'rank_test_score': array([ 1,  3,  2,  5,  6,  4, 10,  8,  7, 11, 12,  9])},
 {'max_depth': 3, 'min_child_weight': 1},
 0.778313628289898)
```

ideal values are 3 for max_depth and 1 for min_child_weight. Lets go one step deeper and look for optimum values.

In [67]:

```
param_test2 = {
 'max_depth':[2,3,4],
 'min_child_weight':[0,1,2]
}
gsearch2 = GridSearchCV(estimator = XGBClassifier( learning_rate=0.1, max_depth=5,
 min_child_weight=2, gamma=0, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test2, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch2.fit(train[predictors],train[target])
```

```
gsearch2.cv_results_, gsearch2.best_params_, gsearch2.best_score_
```

```
C:\ProgramData\Miniconda3\envs\sample1\lib\site-packages\sklearn\model_selection\_search.py:825: F
utureWarning: The parameter 'iid' is deprecated in 0.22 and will be removed in 0.24.
  "removed in 0.24.", FutureWarning
```

Out[67]:

```
({'mean_fit_time': array([0.9221745 , 0.96220379, 0.96971192, 1.24334869, 1.24364271,
         1.23045707, 1.60698524, 1.6156889 , 1.40778213]),
  'std_fit_time': array([0.00910323, 0.03466869, 0.03574978, 0.02694165, 0.02639062,
         0.01523675, 0.01800928, 0.01611296, 0.25968451]),
  'mean_score_time': array([0.01562495, 0.01869454, 0.01291971, 0.01562405, 0.01471868,
         0.01669717, 0.02187381, 0.01902194, 0.01249876]),
  'std_score_time': array([1.01601008e-06, 6.28478449e-03, 7.17853592e-03, 9.88128836e-03,
         1.41577649e-03, 2.14493319e-03, 7.65358489e-03, 6.13602868e-03,
         6.24938014e-03]),
  'param_max_depth': masked_array(data=[2, 2, 2, 3, 3, 3, 4, 4, 4],
               mask=[False, False, False, False, False, False, False, False,
                     False],
         fill_value='?',
              dtype=object),
  'param_min_child_weight': masked_array(data=[0, 1, 2, 0, 1, 2, 0, 1, 2],
               mask=[False, False, False, False, False, False, False, False,
                     False],
         fill_value='?',
              dtype=object),
  'params': [{'max_depth': 2, 'min_child_weight': 0},
   {'max_depth': 2, 'min_child_weight': 1},
   {'max_depth': 2, 'min_child_weight': 2},
   {'max_depth': 3, 'min_child_weight': 0},
   {'max_depth': 3, 'min_child_weight': 1},
   {'max_depth': 3, 'min_child_weight': 2},
   {'max_depth': 4, 'min_child_weight': 0},
   {'max_depth': 4, 'min_child_weight': 1},
   {'max_depth': 4, 'min_child_weight': 2}],
  'split0_test_score': array([0.75477363, 0.75461272, 0.75483078, 0.75583592, 0.75670726,
         0.75641163, 0.75346576, 0.75450289, 0.75652356]),
  'split1_test_score': array([0.75148732, 0.75163839, 0.75159983, 0.75359608, 0.75350054,
         0.75370438, 0.75194342, 0.75178866, 0.75249281]),
  'split2_test_score': array([0.79001996, 0.79025116, 0.79011852, 0.79010273, 0.79021156,
         0.79116915, 0.78958373, 0.78869351, 0.78924817]),
  'split3_test_score': array([0.79974316, 0.79973737, 0.79907862, 0.80185718, 0.80225249,
         0.80284004, 0.80321087, 0.80479603, 0.80402635]),
  'split4_test_score': array([0.78545661, 0.78544661, 0.78559083, 0.79010891, 0.78889629,
         0.79019168, 0.79026209, 0.78988402, 0.78923106]),
  'mean_test_score': array([0.77629614, 0.77633725, 0.77624372, 0.77830016, 0.77831363,
         0.77886338, 0.77769317, 0.77793302, 0.77830439]),
  'std_test_score': array([0.01949726, 0.01952502, 0.01932432, 0.0197414 , 0.01954045,
         0.01995845, 0.02097872, 0.0210368 , 0.02020604]),
  'rank_test_score': array([8, 7, 9, 4, 2, 1, 6, 5, 3])},
 {'max_depth': 3, 'min_child_weight': 2},
 0.7788633789230923)
```

Tuning for gamma

'max_depth': 3, 'min_child_weight': 2

In [68]:

```
param_test3 = {
 'gamma':[i/10.0 for i in range(0,5)]
}
gsearch3 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, max_depth=3,
 min_child_weight=2, gamma=0, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test3, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch3.fit(train[predictors],train[target])
gsearch3.cv_results_, gsearch3.best_params_, gsearch3.best_score_
```

```
C:\ProgramData\Miniconda3\envs\sample1\lib\site-packages\sklearn\model_selection\_search.py:825: F
utureWarning: The parameter 'iid' is deprecated in 0.22 and will be removed in 0.24.
```

```
 "removed in 0.24.", FutureWarning
```

Out[68]:

```
({'mean_fit_time': array([1.27708449, 1.27315269, 1.25151448, 1.25634866, 1.09734697]),
  'std_fit_time': array([0.00919516, 0.0148938 , 0.02154317, 0.0126496 , 0.27614099]),
  'mean_score_time': array([0.01670103, 0.01364598, 0.01562519, 0.01250114, 0.01562381]),
  'std_score_time': array([2.14324488e-03, 2.26672978e-03, 9.88151466e-03, 6.25057264e-03,
         6.57274664e-07]),
  'param_gamma': masked_array(data=[0.0, 0.1, 0.2, 0.3, 0.4],
               mask=[False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'gamma': 0.0},
   {'gamma': 0.1},
   {'gamma': 0.2},
   {'gamma': 0.3},
   {'gamma': 0.4}],
  'split0_test_score': array([0.75641163, 0.75638345, 0.75636555, 0.75636555, 0.75632709]),
  'split1_test_score': array([0.75370438, 0.75370438, 0.75370438, 0.75368859, 0.75286996]),
  'split2_test_score': array([0.79116915, 0.79117441, 0.79078043, 0.79078043, 0.79062475]),
  'split3_test_score': array([0.80284004, 0.80285057, 0.8028661 , 0.80289939, 0.80289939]),
  'split4_test_score': array([0.79019168, 0.79019168, 0.79019168, 0.79019168, 0.79019168]),
  'mean_test_score': array([0.77886338, 0.7788609 , 0.77878163, 0.77878513, 0.77858258]),
  'std_test_score': array([0.01995845, 0.01996797, 0.01992773, 0.01993975, 0.02013722]),
  'rank_test_score': array([1, 2, 4, 3, 5])},
 {'gamma': 0.0},
 0.7788633789230923)
```

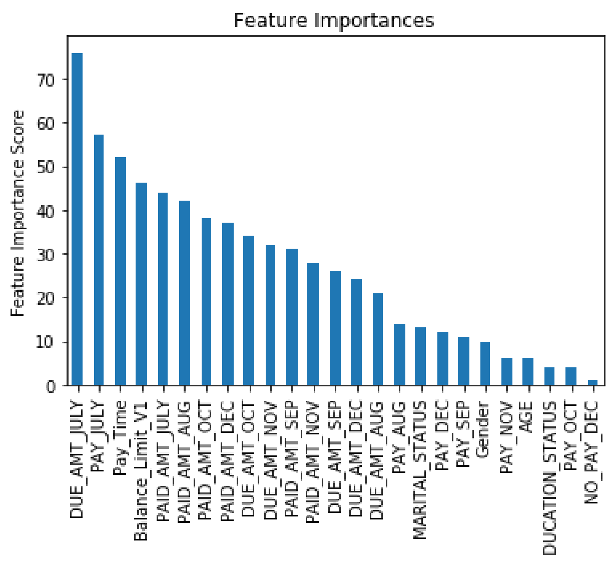
Tuned Model

'max_depth': 3 'min_child_weight': 2 'gamma': 0.0

In [69]:

```
xgb2 = XGBClassifier(
 learning_rate =0.1,
 max_depth=3,
 min_child_weight=2,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27)
modelfit(xgb2, train, predictors)
```

```
Model Report
Accuracy : 0.8239
AUC Score (Train): 0.800786
```



Tune subsample and colsample_bytree

'max_depth': 3 'min_child_weight': 2 'gamma': 0.0

In [71]:

```
param_test4 = {
 'subsample':[i/10.0 for i in range(6,10)],
 'colsample_bytree':[i/10.0 for i in range(6,10)]
}
gsearch4 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, max_depth=3,
 min_child_weight=2, gamma=0, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test4, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch4.fit(train[predictors],train[target])
gsearch4.cv_results_, gsearch4.best_params_, gsearch4.best_score_
```

```
C:\ProgramData\Miniconda3\envs\sample1\lib\site-packages\sklearn\model_selection\_search.py:825: F
utureWarning: The parameter 'iid' is deprecated in 0.22 and will be removed in 0.24.
  "removed in 0.24.", FutureWarning
```

Out[71]:

```
({'mean_fit_time': array([1.05600457, 1.04098873, 1.04065905, 1.04367232, 1.10166154,
         1.11554193, 1.15702281, 1.18223667, 1.26903825, 1.22745914,
         1.23242068, 1.23999891, 1.31568103, 1.32155886, 1.31240091,
         1.26636705]),
  'std_fit_time': array([0.01214149, 0.01258038, 0.0125552 , 0.0116911 , 0.02101644,
         0.00765425, 0.03111199, 0.01626618, 0.03105859, 0.0180564 ,
         0.01070505, 0.02151924, 0.01028265, 0.02986074, 0.01397601,
         0.04138351]),
  'mean_score_time': array([0.01562376, 0.01874914, 0.01249862, 0.01562462, 0.01874866,
         0.01562495, 0.01874905, 0.01517134, 0.01531725, 0.01562409,
         0.01969485, 0.01562457, 0.01517062, 0.01249886, 0.01902661,
         0.01249781]),
  'std_score_time': array([3.81469727e-07, 6.24935671e-03, 6.24930861e-03, 2.41074079e-06,
         6.24911791e-03, 1.41773880e-06, 6.24809268e-03, 6.38839091e-04,
         7.56957300e-04, 4.62310777e-07, 5.87262723e-03, 9.22159176e-07,
         1.09964849e-03, 6.24942798e-03, 6.80537230e-03, 6.24890332e-03]),
  'param_colsample_bytree': masked_array(data=[0.6, 0.6, 0.6, 0.6, 0.7, 0.7, 0.7, 0.7, 0.8, 0.8, 0
.8,
                     0.8, 0.9, 0.9, 0.9, 0.9],
               mask=[False, False, False, False, False, False, False, False,
                     False, False, False, False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'param_subsample': masked_array(data=[0.6, 0.7, 0.8, 0.9, 0.6, 0.7, 0.8, 0.9, 0.6, 0.7, 0.8,
                     0.9, 0.6, 0.7, 0.8, 0.9],
               mask=[False, False, False, False, False, False, False, False,
                     False, False, False, False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'colsample_bytree': 0.6, 'subsample': 0.6},
   {'colsample_bytree': 0.6, 'subsample': 0.7},
   {'colsample_bytree': 0.6, 'subsample': 0.8},
   {'colsample_bytree': 0.6, 'subsample': 0.9},
   {'colsample_bytree': 0.7, 'subsample': 0.6},
   {'colsample_bytree': 0.7, 'subsample': 0.7},
   {'colsample_bytree': 0.7, 'subsample': 0.8},
   {'colsample_bytree': 0.7, 'subsample': 0.9},
   {'colsample_bytree': 0.8, 'subsample': 0.6},
   {'colsample_bytree': 0.8, 'subsample': 0.7},
   {'colsample_bytree': 0.8, 'subsample': 0.8},
   {'colsample_bytree': 0.8, 'subsample': 0.9},
   {'colsample_bytree': 0.9, 'subsample': 0.6},
   {'colsample_bytree': 0.9, 'subsample': 0.7},
   {'colsample_bytree': 0.9, 'subsample': 0.8},
   {'colsample_bytree': 0.9, 'subsample': 0.9}],
  'split0_test_score': array([0.75609323, 0.75692322, 0.75550289, 0.7560183 , 0.75403055,
         0.75536778, 0.75473452, 0.75665814, 0.7566908 , 0.75578127,
         0.75641163, 0.75698827, 0.75573531, 0.75583315, 0.75526468,
         0.75559349]),
  'split1_test_score': array([0.7533554 , 0.7516018 , 0.75097437, 0.75212725, 0.75359213,
         0.75115913, 0.75261243, 0.75270454, 0.7529797 , 0.75123927,
         0.75370438, 0.75286851, 0.75232885, 0.75287509, 0.75306326,
         0.75116373]),
  'split2_test_score': array([0.78980257, 0.78893182, 0.78962163, 0.78925133, 0.79111954,
         0.78986718, 0.79080359, 0.7886964 , 0.79114704, 0.7907645 ,
         0.79116915, 0.78980704, 0.79102361, 0.78912566, 0.78952886,
         0.78940858]),
  'split3_test_score': array([0.80210195, 0.8031131 , 0.80219853, 0.8010575 , 0.8028557 ,
         0.80200944, 0.80149004, 0.80230749, 0.80159018, 0.80360512,
         0.80284004, 0.80105474, 0.80295835, 0.80295137, 0.80265239,
         0.80063114]),
  'split4_test_score': array([0.78760143, 0.78843085, 0.78847454, 0.79123074, 0.78695584,
         0.78742444, 0.78989244, 0.79047645, 0.78693728, 0.7894503 ,
         0.79019168, 0.78926028, 0.78768512, 0.78804831, 0.78929581,
         0.78925488]),
  'mean_test_score': array([0.77779091, 0.77780016, 0.77735439, 0.77793702, 0.77771075,
         0.77716559, 0.77790661, 0.77816861, 0.777869  , 0.77816809,
         0.77886338, 0.77799577, 0.77794625, 0.77776672, 0.777961  ,
         0.77721037]),
  'std_test_score': array([0.0194907 , 0.01999935, 0.02032126, 0.01992921, 0.02019899,
         0.02017559, 0.02021342, 0.01977939, 0.01943825, 0.02078139,
         0.01995845, 0.01934324, 0.02020418, 0.01984781, 0.02003486,
         0.01994049]),
  'rank_test_score': array([11, 10, 14,  7, 13, 16,  8,  2,  9,  3,  1,  4,  6, 12,  5, 15])},
 {'colsample_bytree': 0.8, 'subsample': 0.8},
 0.7788633789230923)
```

In [73]:

```
param_test5 = {
 'subsample':[i/100.0 for i in range(75,90,5)],
 'colsample_bytree':[i/100.0 for i in range(75,90,5)]
}
gsearch4 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, max_depth=3,
 min_child_weight=2, gamma=0, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test4, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch4.fit(train[predictors],train[target])
gsearch4.cv_results_, gsearch4.best_params_, gsearch4.best_score_
```

```
C:\ProgramData\Miniconda3\envs\sample1\lib\site-packages\sklearn\model_selection\_search.py:825: F
utureWarning: The parameter 'iid' is deprecated in 0.22 and will be removed in 0.24.
  "removed in 0.24.", FutureWarning
```

Out[73]:

```
({'mean_fit_time': array([1.07346168, 1.07633109, 1.18824096, 1.2212347 , 1.19405141,
         1.17316036, 1.21589541, 1.12609882, 1.38432808, 1.33024788,
         1.24376478, 1.36324363, 1.60126972, 1.51236339, 1.43849831,
         1.34049726]),
  'std_fit_time': array([0.02004305, 0.00743798, 0.09771839, 0.03791808, 0.04608522,
         0.06405776, 0.05596584, 0.01682662, 0.00933869, 0.07896747,
         0.01218416, 0.07186976, 0.03036713, 0.13886486, 0.01646725,
         0.04444728]),
  'mean_score_time': array([0.01509781, 0.01399322, 0.01658978, 0.02138762, 0.01371899,
         0.01250043, 0.01391802, 0.01809969, 0.01562476, 0.01856837,
         0.01664438, 0.02035875, 0.01929178, 0.01694584, 0.01299272,
         0.01255956]),
  'std_score_time': array([1.05295235e-03, 2.09610208e-03, 2.41464645e-03, 8.39955488e-03,
         1.85797502e-03, 6.25021463e-03, 7.46048337e-03, 4.77434778e-03,
         1.20631319e-06, 5.23421725e-03, 3.22921807e-03, 7.64509012e-03,
         5.61143836e-03, 7.81564343e-03, 6.65987273e-03, 4.77428340e-03]),
  'param_colsample_bytree': masked_array(data=[0.6, 0.6, 0.6, 0.6, 0.7, 0.7, 0.7, 0.7, 0.8, 0.8, 0
.8,
                    0.8, 0.9, 0.9, 0.9, 0.9],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False],
         fill_value='?',
             dtype=object),
  'param_subsample': masked_array(data=[0.6, 0.7, 0.8, 0.9, 0.6, 0.7, 0.8, 0.9, 0.6, 0.7, 0.8,
                    0.9, 0.6, 0.7, 0.8, 0.9],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'colsample_bytree': 0.6, 'subsample': 0.6},
   {'colsample_bytree': 0.6, 'subsample': 0.7},
   {'colsample_bytree': 0.6, 'subsample': 0.8},
   {'colsample_bytree': 0.6, 'subsample': 0.9},
   {'colsample_bytree': 0.7, 'subsample': 0.6},
   {'colsample_bytree': 0.7, 'subsample': 0.7},
   {'colsample_bytree': 0.7, 'subsample': 0.8},
   {'colsample_bytree': 0.7, 'subsample': 0.9},
   {'colsample_bytree': 0.8, 'subsample': 0.6},
   {'colsample_bytree': 0.8, 'subsample': 0.7},
   {'colsample_bytree': 0.8, 'subsample': 0.8},
   {'colsample_bytree': 0.8, 'subsample': 0.9},
   {'colsample_bytree': 0.9, 'subsample': 0.6},
   {'colsample_bytree': 0.9, 'subsample': 0.7},
   {'colsample_bytree': 0.9, 'subsample': 0.8},
   {'colsample_bytree': 0.9, 'subsample': 0.9}],
  'split0_test_score': array([0.75609323, 0.75692322, 0.75550289, 0.7560183 , 0.75403055,
         0.75536778, 0.75473452, 0.75665814, 0.7566908 , 0.75578127,
         0.75641163, 0.75698827, 0.75573531, 0.75583315, 0.75526468,
         0.75559349]),
  'split1_test_score': array([0.7533554 , 0.7516018 , 0.75097437, 0.75212725, 0.75359213,
         0.75115913, 0.75261243, 0.75270454, 0.7529797 , 0.75123927,
         0.75370438, 0.75286851, 0.75232885, 0.75287509, 0.75306326,
         0.75116373]),
  'split2_test_score': array([0.78980257, 0.78893182, 0.78962163, 0.78925133, 0.79111954,
         0.78986718, 0.79080359, 0.7886964 , 0.79114704, 0.7907645 ,
         0.79116915, 0.78980704, 0.79102361, 0.78912566, 0.78952886,
         0.78940858]),
  'split3_test_score': array([0.80210195, 0.8031131 , 0.80219853, 0.8010575 , 0.8028557 ,
         0.80200944, 0.80149004, 0.80230749, 0.80159018, 0.80360512,
         0.80284004, 0.80105474, 0.80295835, 0.80295137, 0.80265239,
         0.80063114]),
  'split4_test_score': array([0.78760143, 0.78843085, 0.78847454, 0.79123074, 0.78695584,
         0.78742444, 0.78989244, 0.79047645, 0.78693728, 0.7894503 ,
         0.79019168, 0.78926028, 0.78768512, 0.78804831, 0.78929581,
         0.78925488]),
  'mean_test_score': array([0.77779091, 0.77780016, 0.77735439, 0.77793702, 0.77771075,
         0.77716559, 0.77790661, 0.77816861, 0.777869  , 0.77816809,
         0.77886338, 0.77799577, 0.77794625, 0.77776672, 0.777961  ,
         0.77721037]),
  'std_test_score': array([0.0194907 , 0.01999935, 0.02032126, 0.01992921, 0.02019899,
         0.02017559, 0.02021342, 0.01977939, 0.01943825, 0.02078139,
         0.01995845, 0.01934324, 0.02020418, 0.01984781, 0.02003486,
         0.01994049]),
  'rank_test_score': array([11, 10, 14,  7, 13, 16,  8,  2,  9,  3,  1,  4,  6, 12,  5, 15])},
 {'colsample_bytree': 0.8, 'subsample': 0.8},
 0.7788633789230923)
```

subsample: 0.8 colsample_bytree: 0.8 'max_depth': 3 'min_child_weight': 2 'gamma': 0.0 Tuning Regularization Parameters

In [75]:

```
param_test6 = {
 'reg_alpha':[1e-5, 1e-2, 0.1, 1, 100]
}
gsearch6 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, max_depth=3,
 min_child_weight=2, gamma=0.0, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test6, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch6.fit(train[predictors],train[target])
gsearch6.cv_results_, gsearch6.best_params_, gsearch6.best_score_
```

```
C:\ProgramData\Miniconda3\envs\sample1\lib\site-packages\sklearn\model_selection\_search.py:825: F
utureWarning: The parameter 'iid' is deprecated in 0.22 and will be removed in 0.24.
  "removed in 0.24.", FutureWarning
```

Out[75]:

```
({'mean_fit_time': array([1.41009712, 1.43547306, 1.3456162 , 1.34926867, 1.01481066]),
  'std_fit_time': array([0.01353788, 0.01852879, 0.08596336, 0.03144356, 0.26751479]),
  'mean_score_time': array([0.02161946, 0.01464472, 0.01376343, 0.01362324, 0.01219325]),
  'std_score_time': array([0.00814284, 0.00256712, 0.00685548, 0.00491524, 0.00292427]),
  'param_reg_alpha': masked_array(data=[1e-05, 0.01, 0.1, 1, 100],
  'param_reg_alpha': masked_array(data=[1e-05, 0.01, 0.1, 1, 100],
               mask=[False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'reg_alpha': 1e-05},
   {'reg_alpha': 0.01},
   {'reg_alpha': 0.1},
   {'reg_alpha': 1},
   {'reg_alpha': 100}],
  'split0_test_score': array([0.7564119 , 0.75596418, 0.75570516, 0.75597102, 0.75622939]),
  'split1_test_score': array([0.75370464, 0.75316499, 0.751713  , 0.75334079, 0.74696713]),
  'split2_test_score': array([0.79116915, 0.79104677, 0.79069818, 0.79033578, 0.78879023]),
  'split3_test_score': array([0.80284004, 0.80284162, 0.80234631, 0.80455943, 0.7980122 ]),
  'split4_test_score': array([0.79019168, 0.79014734, 0.78947556, 0.79087925, 0.78388593]),
  'mean_test_score': array([0.77886348, 0.77863298, 0.77798764, 0.77901725, 0.77477697]),
  'std_test_score': array([0.01995833, 0.0201754 , 0.02036537, 0.02055056, 0.01968055]),
  'rank_test_score': array([2, 3, 4, 1, 5])},
 {'reg_alpha': 1},
 0.7790172545071166)
```

In [78]:

```
param_test7 = {
 'reg_alpha':[0.9, 0.95, 1, 1.05, 1.1]
}
gsearch7 = GridSearchCV(estimator = XGBClassifier( learning_rate =0.1, max_depth=3,
 min_child_weight=2, gamma=0.0, subsample=0.8, colsample_bytree=0.8,
 objective= 'binary:logistic', nthread=4, scale_pos_weight=1,seed=27),
 param_grid = param_test7, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch7.fit(train[predictors],train[target])
gsearch7.cv_results_, gsearch7.best_params_, gsearch7.best_score_
```

```
C:\ProgramData\Miniconda3\envs\sample1\lib\site-packages\sklearn\model_selection\_search.py:825: F
utureWarning: The parameter 'iid' is deprecated in 0.22 and will be removed in 0.24.
  "removed in 0.24.", FutureWarning
```

Out[78]:

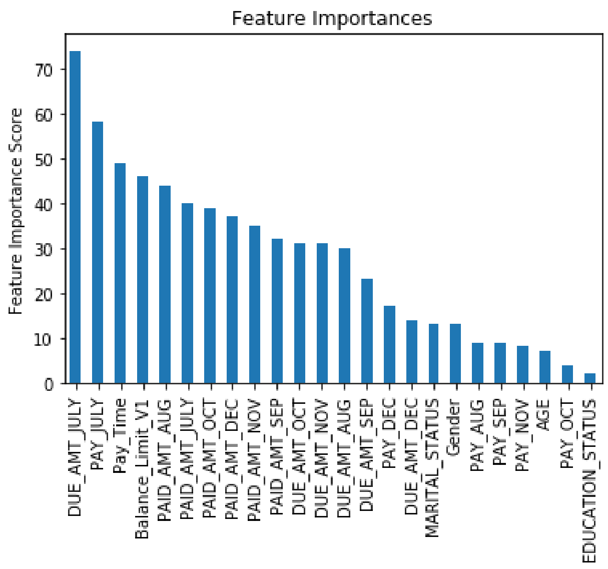
```
({'mean_fit_time': array([1.29074273, 1.29587889, 1.28163171, 1.28201852, 1.11942749]),
  'std_fit_time': array([0.01356018, 0.0071943 , 0.01513304, 0.01673532, 0.3006235 ]),
  'mean_score_time': array([0.01599026, 0.01399226, 0.01497111, 0.01471968, 0.0127933 ]),
  'std_score_time': array([0.00352007, 0.00227904, 0.0007988 , 0.00244771, 0.00256046]),
  'param_reg_alpha': masked_array(data=[0.9, 0.95, 1, 1.05, 1.1],
               mask=[False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'reg_alpha': 0.9},
   {'reg_alpha': 0.95},
   {'reg_alpha': 1},
   {'reg_alpha': 1.05},
   {'reg_alpha': 1.1}],
  'split0_test_score': array([0.75576863, 0.7555237 , 0.75597102, 0.75610494, 0.75630537]),
  'split1_test_score': array([0.75330618, 0.75350476, 0.75334079, 0.75275468, 0.75290101]),
  'split2_test_score': array([0.79134509, 0.7905179 , 0.79033578, 0.7910444 , 0.7911877 ]),
  'split3_test_score': array([0.80321864, 0.80305125, 0.80455943, 0.8033597 , 0.80351274]),
  'split4_test_score': array([0.79064805, 0.79059475, 0.79087925, 0.79109717, 0.79088096]),
  'mean_test_score': array([0.77885732, 0.77863847, 0.77901725, 0.77887218, 0.77895756]),
  'std_test_score': array([0.02036854, 0.02022896, 0.02055056, 0.02048279, 0.02042923]),
  'rank_test_score': array([4, 5, 1, 3, 2])},
 {'reg_alpha': 1},
 0.7790172545071166)
```

subsample: 0.8 colsample_bytree: 0.8 'max_depth': 3 'min_child_weight': 2 'gamma': 0.0 'reg_alpha': 1`

In [79]:

```
xgb3 = XGBClassifier(
 learning_rate =0.1,
 max_depth=3,
 min_child_weight=2,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 reg_alpha=1,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27)
modelfit(xgb3, train, predictors)
```

```
Model Report
Accuracy : 0.8229
AUC Score (Train): 0.799757
```



Tuning the learning rate

In [105]:

```
param_test7 = {
 'learning_rate':[i/100.0 for i in range(0,21,1)]
}
gsearch7 = GridSearchCV(estimator = XGBClassifier(
 learning_rate =0.09,
 max_depth=5,
 min_child_weight=1,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27),
 param_grid = param_test7, scoring='roc_auc',n_jobs=4,iid=False, cv=5)
gsearch7.fit(train[predictors],train[target])
gsearch7.cv_results_, gsearch7.best_params_, gsearch7.best_score_
```

```
C:\ProgramData\Miniconda3\envs\sample1\lib\site-packages\sklearn\model_selection\_search.py:825: F
utureWarning: The parameter 'iid' is deprecated in 0.22 and will be removed in 0.24.
  "removed in 0.24.", FutureWarning
```

Out[105]:

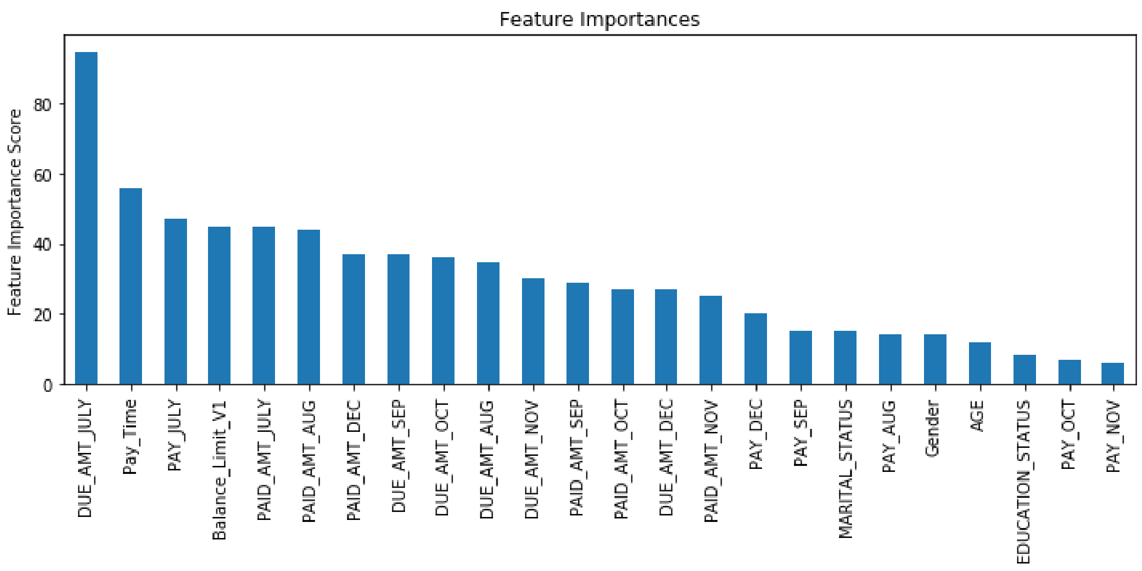
```
({'mean_fit_time': array([2.05833726, 2.11104541, 2.12478261, 2.11444783, 2.10995879,
         2.20771437, 2.14755921, 2.12234855, 2.16665301, 2.12224808,
         2.10000682, 2.08544731, 2.12231994, 2.13821383, 2.194168  ,
         2.25241017, 2.42616043, 2.35872297, 2.29966793, 2.29581928,
         2.04607978]),
  'std_fit_time': array([0.01739834, 0.01384469, 0.0021233 , 0.00801809, 0.03433244,
         0.01600891, 0.01169129, 0.02975344, 0.01520996, 0.04044578,
         0.02337563, 0.01511646, 0.01838071, 0.02294383, 0.02895548,
         0.05588503, 0.04295863, 0.02547597, 0.07204354, 0.03881566,
         0.52245861]),
  'mean_score_time': array([0.01297245, 0.01644402, 0.01874833, 0.02189975, 0.02209635,
         0.01569824, 0.01874943, 0.01874981, 0.01909633, 0.01874914,
         0.01874967, 0.01874948, 0.01874971, 0.01874986, 0.01874838,
         0.01931429, 0.01874981, 0.01562371, 0.01562448, 0.02501769,
         0.01851592]),
  'std_score_time': array([6.55058634e-03, 6.69406438e-04, 6.24892715e-03, 7.68664063e-03,
         7.93285612e-03, 1.46275843e-04, 6.24969006e-03, 6.24902276e-03,
         6.94217694e-03, 6.24840266e-03, 6.24921373e-03, 6.24907023e-03,
         6.24918947e-03, 6.24923708e-03, 6.24878407e-03, 4.32934783e-03,
         6.24926100e-03, 9.88136375e-03, 1.30063836e-06, 5.84916177e-03,
         3.97801114e-03]),
  'param_learning_rate': masked_array(data=[0.0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08,
                     0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17,
                     0.18, 0.19, 0.2],
               mask=[False, False, False, False, False, False, False, False,
                     False, False, False, False, False, False, False, False,
                     False, False, False, False, False],
         fill_value='?',
              dtype=object),
  'params': [{'learning_rate': 0.0},
   {'learning_rate': 0.01},
   {'learning_rate': 0.02},
   {'learning_rate': 0.03},
   {'learning_rate': 0.04},
   {'learning_rate': 0.05},
   {'learning_rate': 0.06},
   {'learning_rate': 0.07},
   {'learning_rate': 0.08},
   {'learning_rate': 0.09},
   {'learning_rate': 0.1},
   {'learning_rate': 0.11},
   {'learning_rate': 0.12},
   {'learning_rate': 0.13},
   {'learning_rate': 0.14},
   {'learning_rate': 0.15},
   {'learning_rate': 0.16},
   {'learning_rate': 0.17},
   {'learning_rate': 0.18},
   {'learning_rate': 0.19},
   {'learning_rate': 0.2}],
  'split0_test_score': array([0.5       , 0.75440566, 0.75869099, 0.75943689, 0.75849302,
         0.75794924, 0.75739327, 0.7569589 , 0.75604907, 0.75508009,
         0.75304865, 0.75497294, 0.75142271, 0.74935598, 0.75014433,
         0.74891909, 0.74844439, 0.74820799, 0.74786041, 0.74282831,
         0.74457661]),
  'split1_test_score': array([0.5       , 0.75004685, 0.75265624, 0.75424224, 0.75470965,
         0.7548531 , 0.75454258, 0.75311184, 0.75413572, 0.75400357,
         0.75063963, 0.75128001, 0.74999334, 0.75423433, 0.75410683,
         0.75102024, 0.75332362, 0.74958334, 0.74974312, 0.74784396,
         0.74567473]),
  'split2_test_score': array([0.5       , 0.79110873, 0.79222531, 0.79251925, 0.79342317,
         0.79323437, 0.79377439, 0.79326364, 0.79257251, 0.78998386,
         0.78830678, 0.78854005, 0.78455506, 0.78872294, 0.78836494,
         0.78564904, 0.78346059, 0.78662443, 0.7819909 , 0.77675217,
         0.77881048]),
  'split3_test_score': array([0.5       , 0.79625753, 0.79943393, 0.80184108, 0.80268947,
         0.80258572, 0.80334581, 0.80425186, 0.80223638, 0.80300301,
         0.8026018 , 0.79897544, 0.79801022, 0.79675446, 0.79809752,
         0.79557243, 0.79703433, 0.79499134, 0.7921296 , 0.79639168,
         0.79612526]),
  'split4_test_score': array([0.5       , 0.78735124, 0.79254186, 0.79490705, 0.79651981,
         0.79578962, 0.79503065, 0.79419356, 0.79443612, 0.79234452,
         0.79270202, 0.79127203, 0.79033256, 0.7909193 , 0.7879842 ,
         0.78838102, 0.78800883, 0.78377312, 0.7803897 , 0.78442229,
         0.77676925]),
  'mean_test_score': array([0.5       , 0.775834  , 0.77910967, 0.7805893 , 0.78116703,
         0.78088241, 0.78081734, 0.78035596, 0.77988596, 0.77888301,
         0.77745978, 0.77700809, 0.77486278, 0.7759974 , 0.77573957,
         0.77390836, 0.77405435, 0.77263604, 0.77042275, 0.76964768,
         0.76839126]),
  'std_test_score': array([0.        , 0.01953067, 0.0194022 , 0.01970052, 0.02031374,
         0.02024492, 0.02057418, 0.02106545, 0.02051082, 0.02035574,
         0.02143515, 0.01983176, 0.02018408, 0.01999427, 0.01965859,
         0.01982404, 0.01947777, 0.01973639, 0.0181166 , 0.02087432,
         0.02015492]),
  'rank_test_score': array([21, 12,  7,  4,  1,  2,  3,  5,  6,  8,  9, 10, 14, 11, 13, 16, 15,
         17, 18, 19, 20])},
 {'learning_rate': 0.04},
 0.7811670255881003)
```

Tuned Model

In [99]:

```
xgb4 = XGBClassifier(
 learning_rate =0.1,
 max_depth=3,
 n_estimators=5000,
 min_child_weight=2,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 reg_alpha=1,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27)
modelfit(xgb4, train, predictors)
```

```
Model Report
Accuracy : 0.8244
AUC Score (Train): 0.803234
Saved file: XGBoost 1.csv
```


Feature Importances
Feature Importance Score
DUE_AMT_JULY
Pay_Time
PAY_JULY
Balance_Limit_V1
PAID_AMT_JULY
PAID_AMT_AUG
PAID_AMT_DEC
DUE_AMT_SEP
DUE_AMT_OCT
DUE_AMT_AUG
DUE_AMT_NOV
PAID_AMT_SEP
PAID_AMT_OCT
DUE_AMT_DEC
PAID_AMT_NOV
PAY_DEC
PAY_SEP
MARITAL_STATUS
PAY_AUG
Gender
AGE
EDUCATION_STATUS
PAY_OCT
PAY_NOV