This repository Search

Pull requests Issues Gist

sermakarevich / **kaggle-homesite**

Watch 0 Star 3 Fork 0

Code Issues 0 Pull requests 0 Wiki Pulse Graphs

Branch: **master** kaggle-homesite / **homesite_55-th.ipynb**

Find file Copy path

sermakarevich homesite solution 115b8a6 6 days ago

1 contributor

```
228 lines (227 sloc) | 7.74 KB
```

Raw Blame History

In [1]:

```
import pandas as pd
%matplotlib inline
from matplotlib import pyplot as plt
import seaborn, gc
import numpy as np
from sklearn import cross_validation, linear_model, metrics, ensemble, preprocessing, svm, naive_bayes
from sklearn import neighbors, feature_selection
from sklearn import grid_search
import xgboost as xgb
from scipy import stats
```

```
/Users/sermakarevich/anaconda/lib/python2.7/site-packages/matplotlib/__init__.py:872: UserWarning: axes.co
lor_cycle is deprecated and replaced with axes.prop_cycle; please use the latter.
  warnings.warn(self.msg_depr % (key, alt_key))
```

In [8]:

```
def fix_field10(df):
    df["Field10"] = df["Field10"].apply(lambda x: x.replace(",", "")).astype(int)

    return df

def get_lovvar_cols(df, threshold):
    selector = feature_selection.VarianceThreshold(threshold=threshold)
    selector.fit(df)

    cols = df.columns[~selector.get_support()]
    cols = [col for col in cols if col not in ["Field9", "Field8", 'Field11', 'Field12']]

    return cols

def count_less_0(df):
    df["Below0"] = np.sum(df<0, axis = 1)

    return df
```

In [9]:

```
train = pd.read_csv("/Users/sermakarevich/data/homesite/train.csv")
test = pd.read_csv("/Users/sermakarevich/data/homesite/test.csv")
test.index = test["QuoteNumber"]
test["QuoteConversion_Flag"] = 1
train.index = train["QuoteNumber"]
y = train["QuoteConversion_Flag"]
df = pd.concat([train, test], axis = 0)
df_ind = df.index

df = df.replace(' ', np.nan, regex=True)

golden_feature=[("CoverageField1B","PropertyField21B"),
                ("GeographicField6A","GeographicField8A"),
                ("GeographicField6A","GeographicField13A"),
                ("GeographicField8A","GeographicField13A"),
                ("GeographicField11A","GeographicField13A"),
                ("GeographicField8A","GeographicField11A"),
                ("CoverageField1A", "PropertyField21A"),
                ("CoverageField2B", "PropertyField21B")]

df = df.fillna(-1)
df = fix_field10(df)

df["Date"] = pd.to_datetime(df["Original_Quote_Date"])
df['DayOfWeek'] = df["Date"].map(lambda x: x.dayofweek)
df['Year'] = df["Date"].map(lambda x: x.year)
df['Month'] = df["Date"].map(lambda x: x.month)
df["Season"] = df["Date"].map(lambda x: x.quarter)
del df["Date"], df["QuoteNumber"], df["Original_Quote_Date"], df["QuoteConversion_Flag"]

df = count_less_0(df)

for featureA,featureB in golden_feature:
    df["_".join([featureA,featureB,"diff"])]= df[featureA] - df[featureB]

for i in df.dtypes[df.dtypes==object].index:
    one_hot = pd.DataFrame(pd.get_dummies(df[i], prefix=i), index = df.index)
    df = pd.concat([df, one_hot], axis = 1)
    del df[i]

df.index.name = "QuoteNumber"
```

```
train = df.ix[train.index].copy()
test = df.ix[test.index].copy()
```

In [ ]:

```
filt_cols = ["Original_Quote_Date", "QuoteConversion_Flag", "QuoteNumber", "Date", "MonthDay"] + \
    get_lovvar_cols(train, 0.0001)
cols = [col for col in train.columns if col not in filt_cols]

skf = cross_validation.StratifiedKFold(y, n_folds=10, random_state=0, shuffle=True)

predictions_test_list = []
predictions_train_list = []
auc = []

dtest_real = xgb.DMatrix(test[cols])

for train_index, test_index in skf:
    dtrain = xgb.DMatrix(train.iloc[train_index][cols],
             y.iloc[train_index])
    dtest = xgb.DMatrix(train.iloc[test_index][cols],
                        y.iloc[test_index])

    watchlist = [(dtest, 'test'), (dtrain, 'train')]

    params = {}
    params["objective"] = "binary:logistic"
    params["eta"] = 0.02
    params["subsample"] = 0.8
    params["colsample_bytree"] = 0.6
    params["max_depth"] = 6
    params["eval_metric"] = "auc"
    params["nthread"] = 4
    params["min_child_weight"] = 4

    plst = list(params.items())
    num_rounds = 3000

    gbm = xgb.train(plst, dtrain, num_rounds, evals=watchlist, early_stopping_rounds=30)
    prediction = gbm.predict(dtest)
    result = metrics.roc_auc_score(y.ix[train.index[test_index]], prediction)
    print result
    auc.append(result)

    prediction = pd.Series(prediction, index = train.index[test_index])
    predictions_train_list.append(prediction)

    prediction = pd.Series(gbm.predict(dtest_real), index = test.index)
    predictions_test_list.append(prediction)

num_rounds = 3400
dtrain = xgb.DMatrix(train.loc[:, cols], y.ix[train.index])
gbm = xgb.train(plst, dtrain, num_rounds)
prediction = pd.Series(gbm.predict(dtest_real), index = test.index)
predictions_test_list.append(prediction)

prediction = np.mean(pd.concat(predictions_test_list, axis = 1), axis = 1)
prediction.name = "QuoteConversion_Flag"
prediction = pd.DataFrame(prediction)
prediction.to_csv(\
"/Users/sermakarevich/Dropbox/Machine_learning/Kaggle/homesite/predictions/xgboost_logbin_3000.csv")

prediction = pd.concat(predictions_train_list, axis = 0)
prediction.name = "xgboost"
prediction = pd.DataFrame(prediction)
prediction.to_csv(\
"/Users/sermakarevich/Dropbox/Machine_learning/Kaggle/homesite/validation_pred/xgboost_validation_3000.cs
v")
```

In [11]:

```
print auc, np.mean(auc)
```

```
[0.96770087975294095, 0.97001840934814965, 0.96718125389359832, 0.96694673547778298, 0.96911860561136165,
0.96763167331274635, 0.96594224696793307, 0.9678801325612878, 0.96730752257368358, 0.96786600216570906] 0.
967759346167
```