In [166]:

```
#Load packages:
import numpy as np
import pandas as pd
import os
from sklearn.preprocessing import LabelEncoder

from scipy.stats.mstats import mode
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import cross_val_score

from sklearn import linear_model
from sklearn.model_selection import train_test_split, KFold
from sklearn.metrics import confusion_matrix, accuracy_score
import seaborn as sns; sns.set()
from IPython.display import Image, display

# matplotlib and seaborn for plotting
import matplotlib.pyplot as plt
plt.style.use('ggplot')
%matplotlib inline

# Suppress warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
print(os.listdir("../WorkSample/"))
```

```
['Untitled.ipynb', 'Data Scientist - Work Sample.docx', 'device_failure_worksample.csv',
'.ipynb_checkpoints']
```

Data Wrangling

In [3]:

```
# Training data
df_all = pd.read_csv('../WorkSample/device_failure_worksample.csv')
print('All data shape: ', df_all.shape)
df_all.head()
```

```
All data shape:  (124494, 12)
```

Out[3]:

	date	device	attribute1	attribute2	attribute3	attribute4	attribute5	attribute6	attribute7	attribute8	attribute9	fai
0	15001	S1F01085	215630672	56	0	52	6	407438	0	0	7	0
1	15001	S1F0166B	61370680	0	3	0	6	403174	0	0	0	0
2	15001	S1F01E6Y	173295968	0	0	0	12	237394	0	0	0	0
3	15001	S1F01JE0	79694024	0	0	0	6	410186	0	0	0	0
4	15001	S1F01R2B	135970480	0	0	0	15	313173	0	0	3	0

In [4]:

```
# Examine the Distribution of the Target Column
df_all['failure'].value_counts()
```

Out[4]:

```
0    124388
1       106
```

```
Name: failure, dtype: int64
```

Groupby device ID

In [5]:

```
#group by device ID, count the number of 'date' on a certain device
date_cnt = df_all.groupby('device')['date'].count()
#date_cnt.value_counts(dropna=False)
```

In [6]:

```
#aggregate variables
df_all = df_all.groupby('device').agg({'failure' : 'sum', 'attribute6' : 'sum',
                                'attribute1':'max', 'attribute9': 'max',
                                'attribute2':'max', 'attribute3': 'max', 'attribute4' : 'max',
                                'attribute5':'max', 'attribute7': 'max'})
print('unique devices shape: ', df_all.shape)
```

```
unique devices shape:  (1168, 9)
```

In [7]:

```
#merge a new dataframe,  date: the number of date
df_merge = pd.concat([df_all, date_cnt], axis=1, ignore_index=False)
print('new merged df shape: ', df_merge.shape)
```

```
new merged df shape:  (1168, 10)
```

Exploratory Data Analysis:

In [8]:

```
df_merge.head(10)
```

Out[8]:

	failure	attribute6	attribute1	attribute9	attribute2	attribute3	attribute4	attribute5	attribute7	date
device										
S1F01085	0	2447271	215630672	7	56	0	52	6	0	6
S1F013BB	0	4134126	243346080	0	0	0	0	5	0	6
S1F0166B	0	2421295	224339296	0	0	3	0	6	0	6
S1F01E6Y	0	12236477	240257968	0	0	0	0	12	0	48
S1F01JE0	0	2463785	235562856	0	0	0	0	6	0	6
S1F01R2B	0	73859055	243500200	3	0	0	0	19	0	223
S1F01TD5	0	2483446	236835240	1	0	0	41	6	0	6
S1F01XDJ	0	44678613	240833760	0	0	0	0	8	0	106
S1F023H2	1	9589318	243825496	3	0	0	1	19	16	19
S1F02A0J	0	75471330	244123040	0	0	1	0	16	0	227

In [113]:

```
#df_merge.columns
featurs = [ 'attribute6', 'attribute1', 'attribute9', 'attribute2',
       'attribute3', 'attribute4', 'attribute5', 'attribute7', 'date']
```

In [9]:

```
df_merge['failure'].value_counts(dropna=False)
#failure: 106, normal: 1062
```

Out[9]:

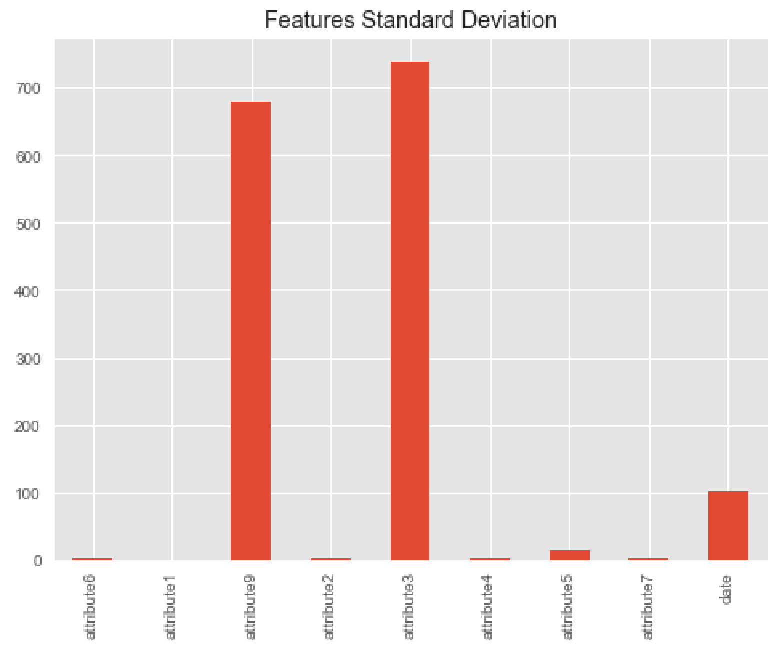
```
0    1062
1     106
Name: failure, dtype: int64
```

In [115]:

```
#plot and compare the standard deviation of input features:
df_merge[featurs].std().plot(kind='bar', figsize=(8,6), title="Features Standard Deviation")
```

Out[115]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a17dbc470>
```

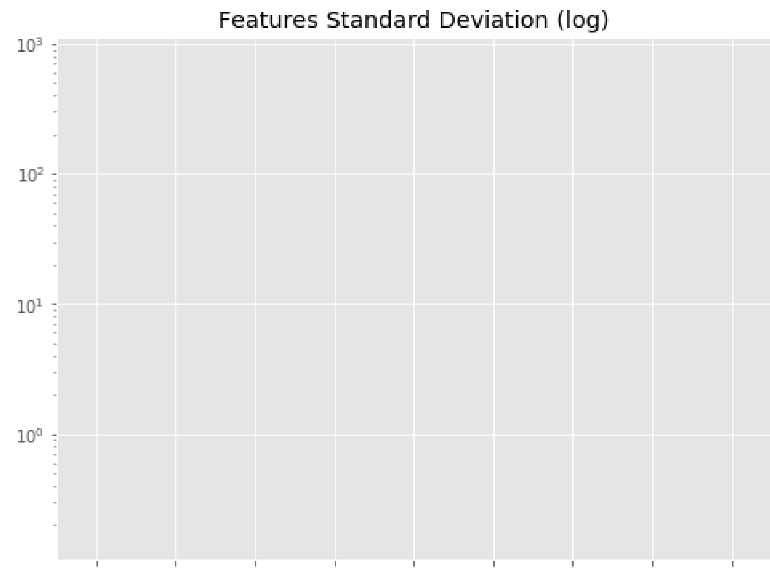


In [159]:

```
# plot and compare the log standard deviation of input features:
df_merge[featurs].std().plot(kind='bar', figsize=(8,6), logy=True,title="Features Standard
Deviation (log)")
```

Out[159]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a26bbeeb8>
```



attribute6 attribute1 attribute9 attribute2 attribute3 attribute4 attribute5 attribute7 date

In [120]:

```
# get ordered list of top variance features:
featurs_top_var = df_merge[featurs].std().sort_values(ascending=False)
featurs_top_var
```

Out[120]:

```
attribute3    738.567697501974294
attribute9    678.406910698268234
date          102.651042776436441
attribute5     12.370542404057760
attribute6      2.521804158298231
attribute2      2.394329752260311
attribute4      1.127853354564793
attribute7      0.873606727302364
attribute1      0.164961378125472
dtype: float64
```

In [10]:

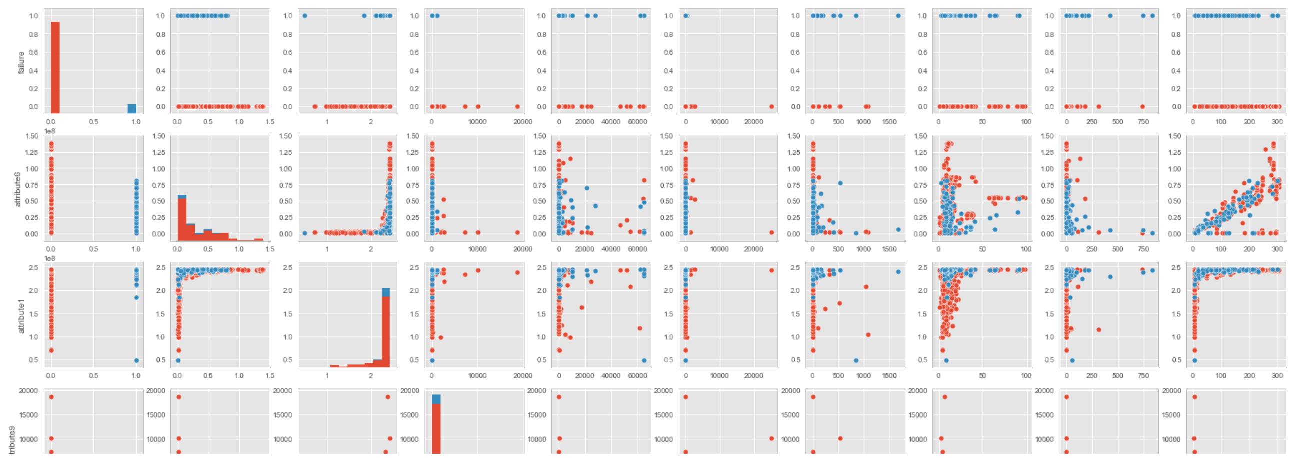
```
#variables correlation
df_merge.corr()
```

Out[10]:

	failure	attribute6	attribute1	attribute9	attribute2	attribute3	attribute4	attribute5	attribute7	date
failure	1.000000	-0.027355	0.099725	-0.012201	0.178851	-0.011711	0.181233	0.077348	0.204515	-0.017000
attribute6	-0.027355	1.000000	0.411288	-0.046693	-0.022909	-0.018768	-0.057795	0.150072	-0.050576	0.879975
attribute1	0.099725	0.411288	1.000000	0.008648	-0.071950	0.016221	-0.113202	0.162886	-0.007493	0.474314
attribute9	-0.012201	-0.046693	0.008648	1.000000	-0.006273	0.447703	0.078266	-0.028133	0.015573	-0.056289
attribute2	0.178851	-0.022909	-0.071950	-0.006273	1.000000	-0.003510	0.347504	-0.006053	0.081082	-0.017311
attribute3	-0.011711	-0.018768	0.016221	0.447703	-0.003510	1.000000	0.189068	-0.023523	-0.004162	-0.022751
attribute4	0.181233	-0.057795	-0.113202	0.078266	0.347504	0.189068	1.000000	-0.006778	0.060772	-0.070330
attribute5	0.077348	0.150072	0.162886	-0.028133	-0.006053	-0.023523	-0.006778	1.000000	0.000141	0.182373
attribute7	0.204515	-0.050576	-0.007493	0.015573	0.081082	-0.004162	0.060772	0.000141	1.000000	0.000559
date	-0.017000	0.879975	0.474314	-0.056289	-0.017311	-0.022751	-0.070330	0.182373	0.000559	1.000000

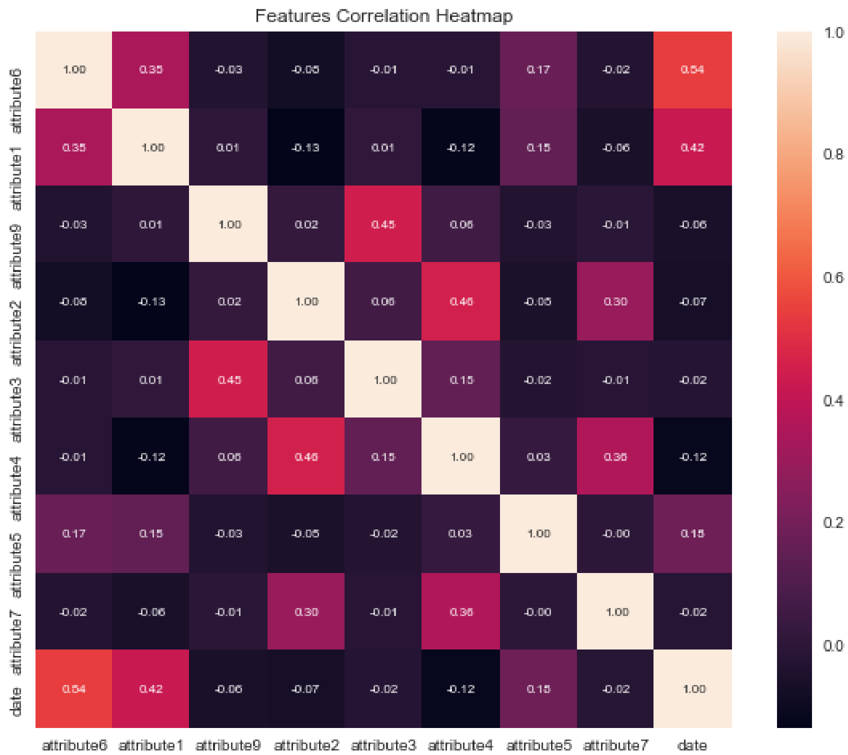
In [15]:

```
#visualization of correlations with pairplot
sns.pairplot(df_merge, hue='failure')
plt.show()
```



In [123]:

```
# plot a heatmap to display +ve and -ve correlation among features and label
cm = np.corrcoef(df_merge[featurs].values.T)
sns.set(font_scale=1.0)
fig = plt.figure(figsize=(10, 8))
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 8}, yticklab
els=featurs, xticklabels=featurs)
plt.title('Features Correlation Heatmap')
plt.show()
```



In [124]:

```
#reset matplotlib original theme
```

```
sns.reset_orig()
```

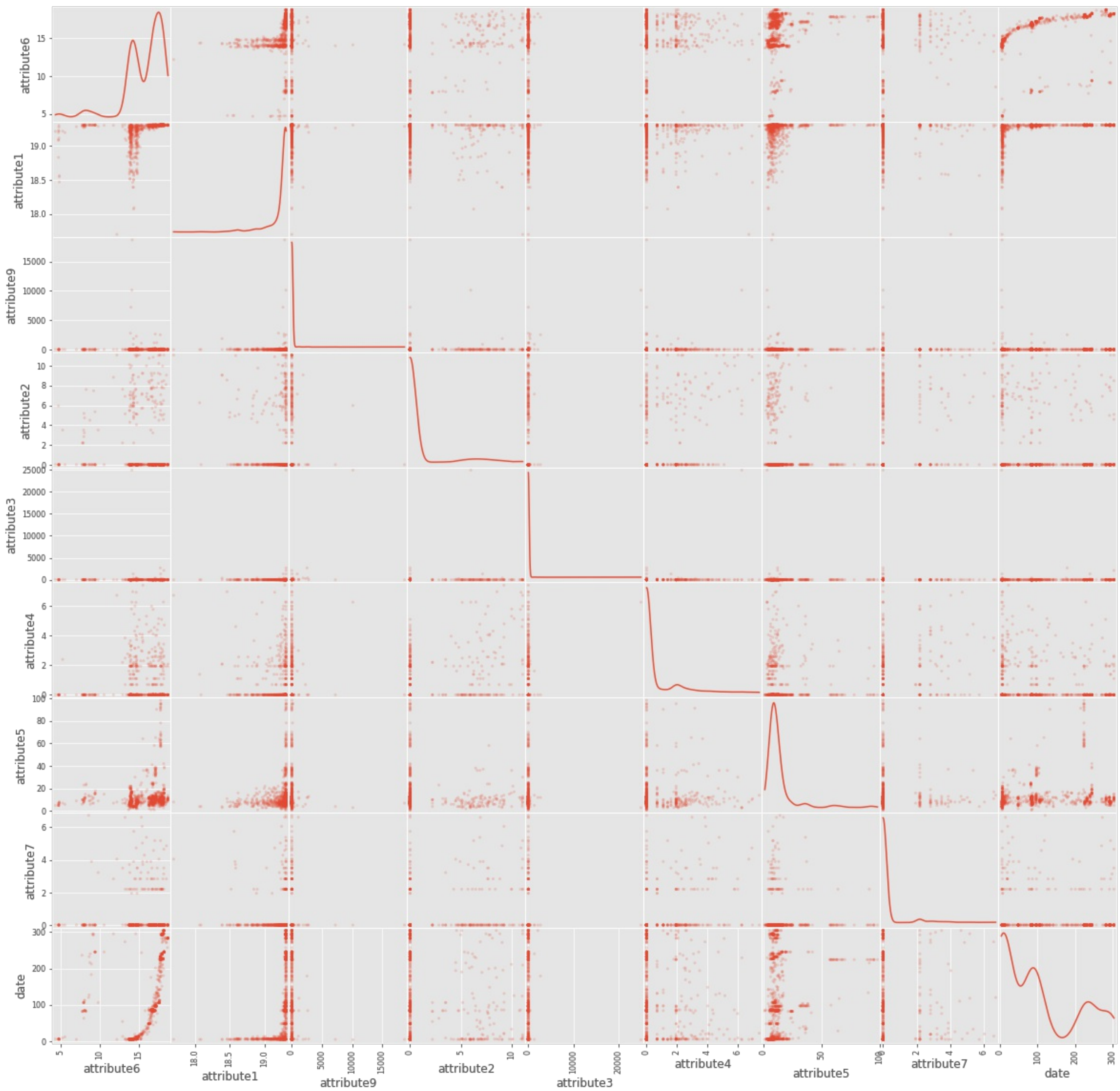
In [126]:

```
#create scatter matrix to disply relatiohships and distribution among features and regression labe
l
from pandas.tools.plotting import scatter_matrix
scatter_matrix(df_merge[featurs], alpha=0.2, figsize=(20, 20), diagonal='kde')
```

Out[126]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1a17d80ef0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1b85d9b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1cc782e8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17e180b8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a170ec4a8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a170ec5f8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17fd6e80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17073080>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1806e160>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a18054c18>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a16eeaf60>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a16fd8080>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1701e2b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a170dc470>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a170a9a20>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17123da0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17171ef0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a171a4ef0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a171967b8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1726d588>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17267da0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17455f60>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1748b0f0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17635780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a177ebb00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17954c50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17a2ec50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a179494e0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17ee4588>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1b5fb6a0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17a4e780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17a97d30>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17b37390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17b8c400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17c134e0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17cc7a90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a17cd4908>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17d032b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17d6e390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17e3f940>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17efc940>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17f84fd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a181f90f0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a182d2a58>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a183bf6a0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a183d2208>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1b5a8470>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1b959550>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1b897b00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1b8b0cf8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1bae02b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1bc14390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a18136940>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1816f940>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a16f46fd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a16f83fd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a172a3a20>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17341080>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a16f98748>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a15181ef0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a260f6470>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a26130550>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x103c505c0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a16271d30>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a175052b0>,
```

```
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a175f1390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17786390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a178ba0f0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a178f2240>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1792f320>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a179a6400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17983860>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a17ad2160>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17b0b240>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17c46320>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17e7d860>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a17a5dcc0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a181dc0f0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a182971d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1830f2b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a182aba58>]],
      dtype=object)
```



Exploration of each feature invidually

In [131]:

```
# Plot 2 main graphs for a single feature.
# - plot1: histogram;
# - plot2: boxplot
def explore_feature(s, e):
```

```
def explore_feature(s, e):
    fig = plt.figure(figsize=(10, 8))

    sub1 = fig.add_subplot(221)
    sub1.set_title(s +' histogram')
    sub1.hist(df_merge[s])

    sub2 = fig.add_subplot(222)
    sub2.set_title(s +' boxplot')
    sub2.boxplot(df_merge[s])

    if e > 100 or e <= 0:
        select_engines = list(pd.unique(df_merge.id))
    else:
        select_engines = np.random.choice(range(1,101), e, replace=False)

    plt.tight_layout()
    plt.show()
```

In [133]:

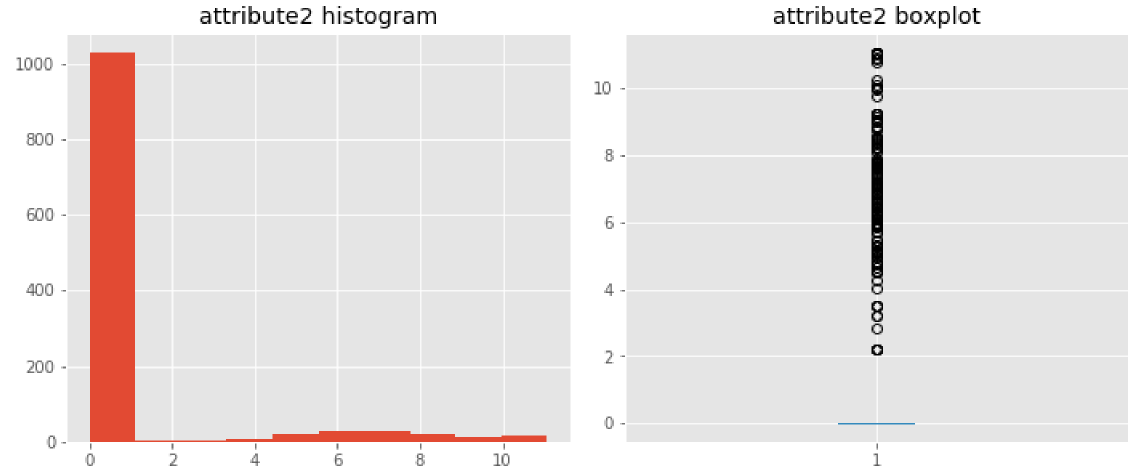
```
df_merge.columns
```

Out[133]:

```
Index(['failure', 'attribute6', 'attribute1', 'attribute9', 'attribute2',
       'attribute3', 'attribute4', 'attribute5', 'attribute7', 'date'],
      dtype='object')
```

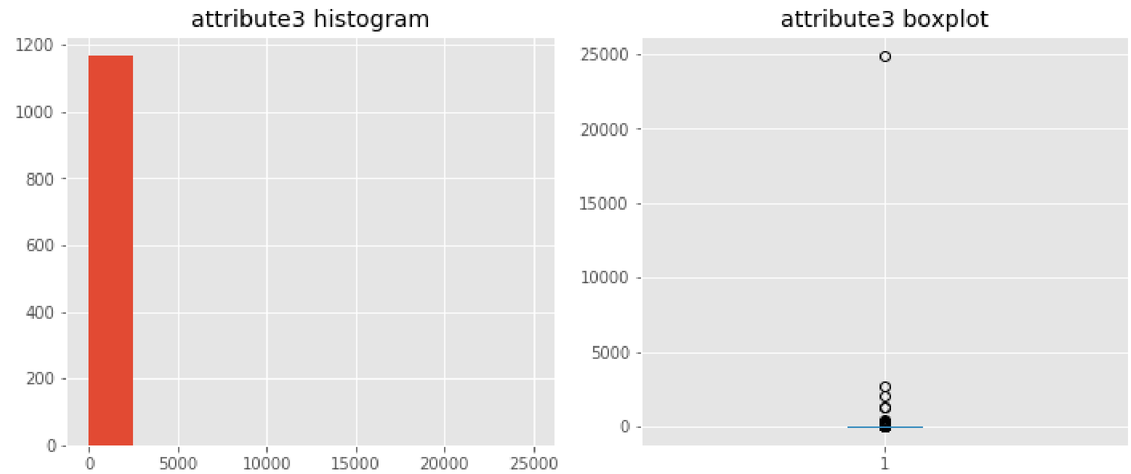
In [134]:

```
explore_feature("attribute2", 10)
```



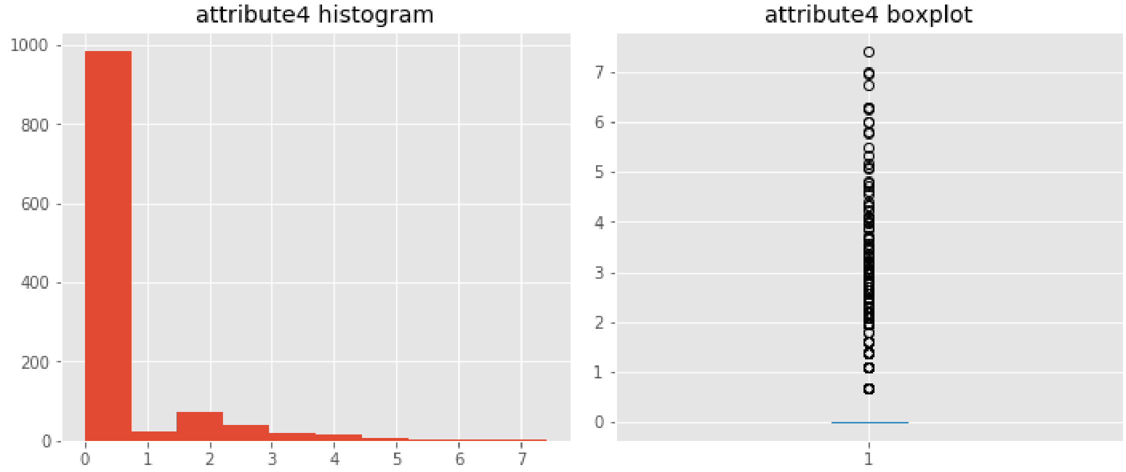
In [135]:

```
explore_feature("attribute3", 10)
```



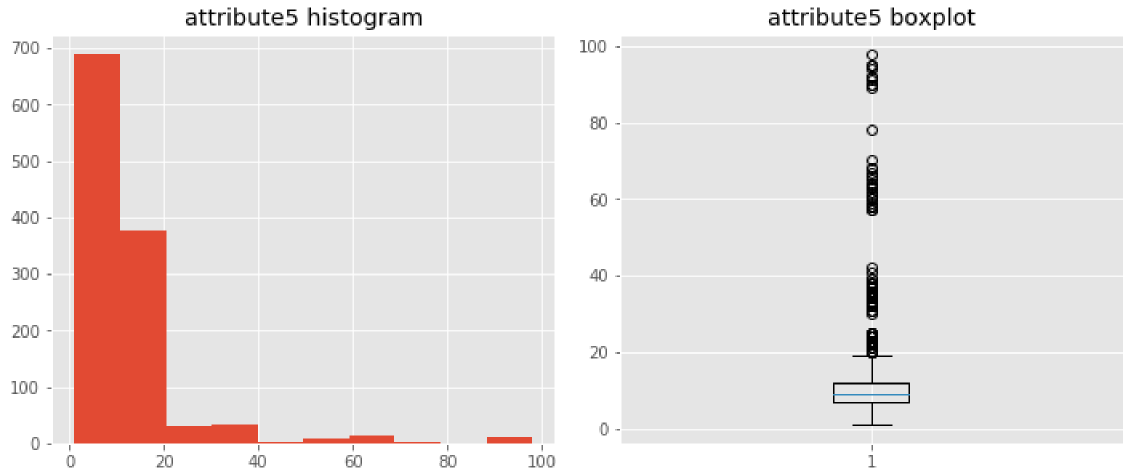
In [136]:

```
explore_feature("attribute4", 10)
```



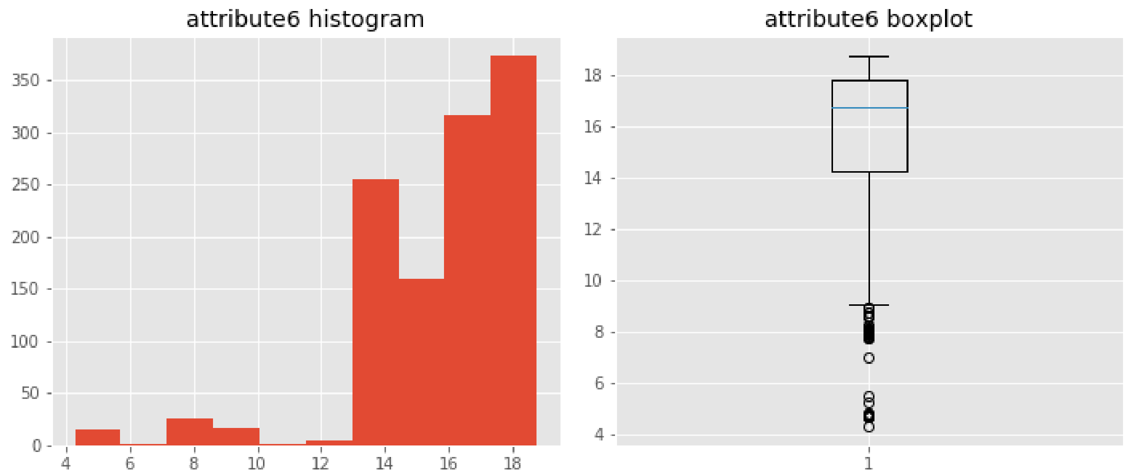
In [137]:

```
explore_feature("attribute5", 10)
```



In [138]:

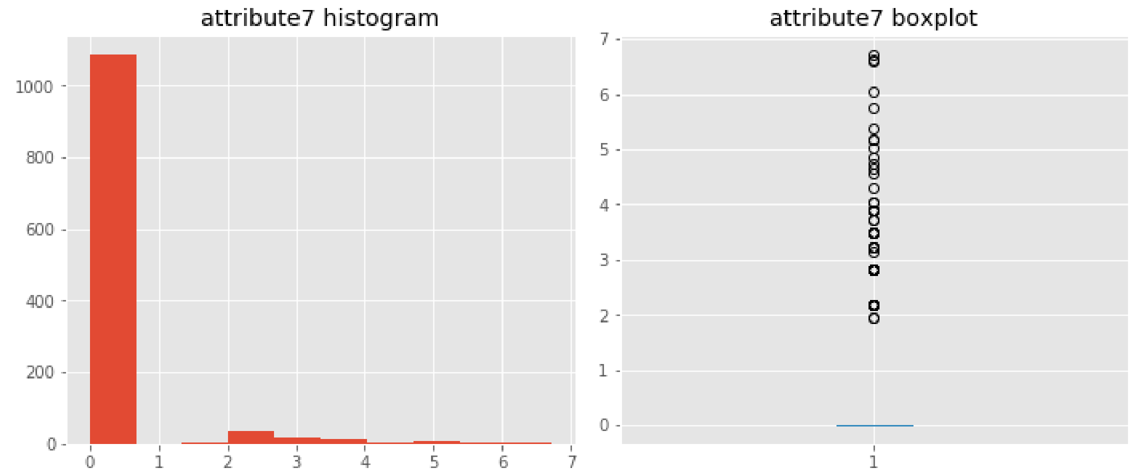
```
explore_feature("attribute6", 10)
```



In [139]:

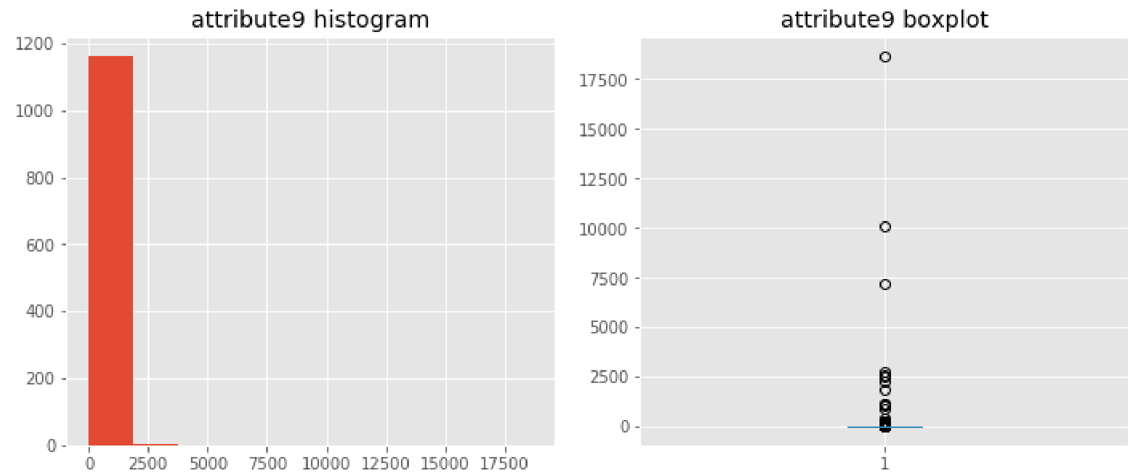
```
explore_feature("attribute7", 10)
```

```
explore_feature("attribute7", 10)
```



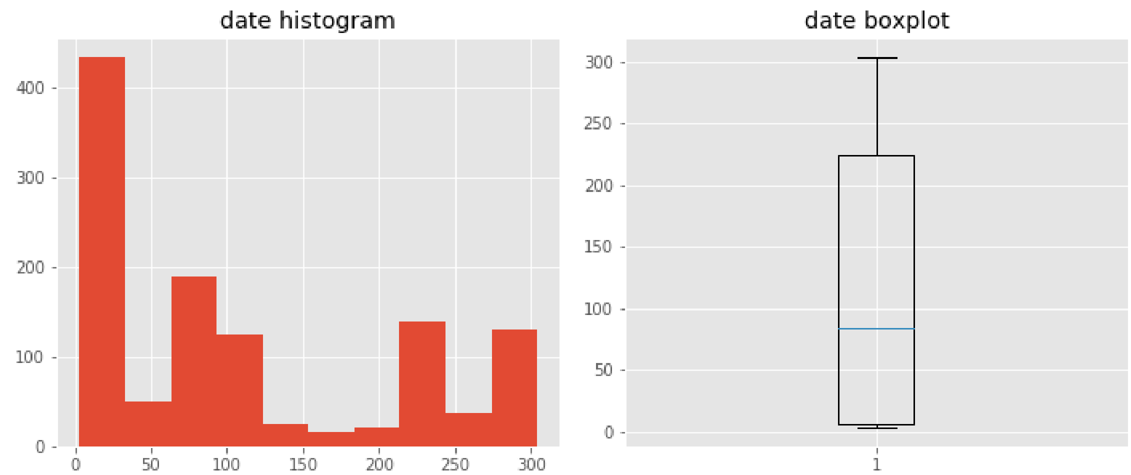
In [141]:

```
explore_feature("attribute9", 10)
```



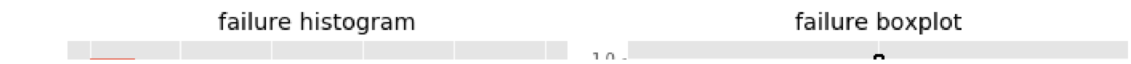
In [142]:

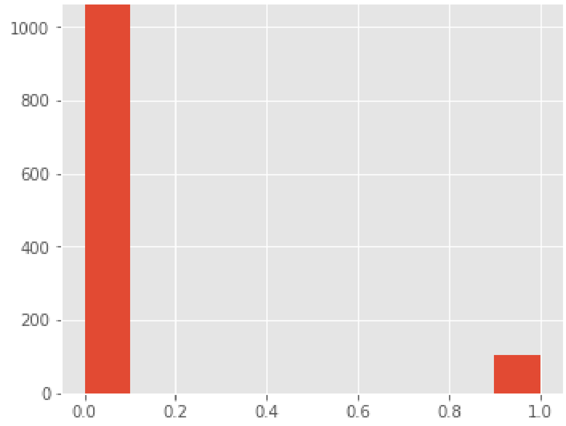
```
explore_feature("date", 10)
```

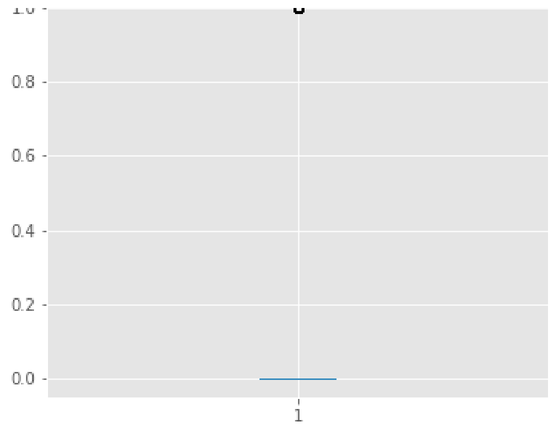


In [143]:

```
explore_feature("failure", 10)
```







EDA Summary:

- All attributes are of integer data type;
- Attribute7 and 8 are exactly the same. Attribute8 is dropped;
- Data level: total 124,494 records, only 106 failures, very imbalanced dataset. Aggregate to the device level: 1,062 majority cases and 106 minority cases (roughly 10%);
- Attribute4,7,9 have limited number of distictive values, they are likely to be categorical variable;
- Attribute 1 and 6 are likely to be continuous variables;
- very high correlation between Attribute3 and 9, it may hurt the performance, should be dropped;
- numpy log(1 + attribute) is used to transform some attributes;

Split the data

In [24]:

```
#Only attribute1,2,4,5,6,7 are used in the training
X = df_merge.drop(['failure','attribute3','attribute9'], axis=1)
y = df_merge['failure']
```

In [25]:

```
# Split the data, specify 80/20 for training set and test set
RANDOM_STATE = 110
X_train, X_test, y_train, y_test = train_test_split( X, y, train_size=0.8,
random_state=RANDOM_STATE, stratify=df_all['failure'])
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[25]:

```
((934, 7), (934,), (234, 7), (234,))
```

In [83]:

```
X_train.columns
```

Out[83]:

```
Index(['attribute6', 'attribute1', 'attribute2', 'attribute4', 'attribute5',
       'attribute7', 'date'],
      dtype='object')
```

Modeling on Binary Classifcation

Modules:

- Logistic Regression;
- Decision Tree;
- Random Forest;
- Light GBM

- SVC;
- SVC Linear;
- KNN;
- Gaussian NB

In [89]:

```
from sklearn import metrics
from sklearn import model_selection

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
import lightgbm as lgb
```

Perform Grid Search hyper parameter tuning

In [164]:

```
# Perform Grid Search hyper parameter tuning
def bin_classify(model, clf, features, params=None, score=None):
    grid_search = model_selection.GridSearchCV(estimator=clf, param_grid=params, cv=10, scoring=sco
re, n_jobs=-1)

    grid_search.fit(X_train, y_train)
    y_pred = grid_search.predict(X_test)

    if hasattr(grid_search, 'predict_proba'):
        y_score = grid_search.predict_proba(X_test)[:,1]
    elif hasattr(grid_search, 'decision_function'):
        y_score = grid_search.decision_function(X_test)
    else:
        y_score = y_pred

    predictions = {'y_pred' : y_pred, 'y_score' : y_score}
    df_predictions = pd.DataFrame.from_dict(predictions)

    return grid_search.best_estimator_, df_predictions
```

Calculate and plot metrics

In [160]:

```
# Calculate main binary classifcation metrics, plot AUC ROC and Precision-Recall curves.
def bin_metrics(model, y_test, y_pred, y_score, print_out=True, plot_out=True):
    binclass_metrics = {
                        'Accuracy' : metrics.accuracy_score(y_test, y_pred),
                        'Precision' : metrics.precision_score(y_test, y_pred),
                        'Recall' : metrics.recall_score(y_test, y_pred),
                        'F1 Score' : metrics.f1_score(y_test, y_pred),
                        'ROC AUC' : metrics.roc_auc_score(y_test, y_score)
                       }

    df_metrics = pd.DataFrame.from_dict(binclass_metrics, orient='index')
    df_metrics.columns = [model]

    fpr, tpr, thresh_roc = metrics.roc_curve(y_test, y_score)
    roc_auc = metrics.auc(fpr, tpr)

    roc_que = []
    for thr in thresh_roc:
        roc_que.append((y_score >= thr).mean())

    roc_que = np.array(roc_que)

    roc_thresh = {
                    'Threshold' : thresh_roc,
                    'TPR' : tpr,
                    'FPR' : fpr,
```

```
                'Que' : roc_que
               }

    df_roc_thresh = pd.DataFrame.from_dict(roc_thresh)

    #calculate other classification metrics: TP, FP, TN, FN, TNR, FNR
    df_roc_thresh['TP'] = (25*df_roc_thresh.TPR).astype(int)
    df_roc_thresh['FP'] = (25 - (25*df_roc_thresh.TPR)).astype(int)
    df_roc_thresh['TN'] = (75*(1 - df_roc_thresh.FPR)).astype(int)
    df_roc_thresh['FN'] = (75 - (75*(1 - df_roc_thresh.FPR))).astype(int)

    df_roc_thresh['TNR'] = df_roc_thresh['TN']/(df_roc_thresh['TN'] + df_roc_thresh['FN'])
    df_roc_thresh['FNR'] = df_roc_thresh['TN']/(df_roc_thresh['TN'] + df_roc_thresh['FP'])

    df_roc_thresh['Model'] = model

    precision, recall, thresh_prc = metrics.precision_recall_curve(y_test, y_score)

    thresh_prc = np.append(thresh_prc,1)

    devices_prc = []
    for thr in thresh_prc:
        devices_prc.append((y_score >= thr).mean())

    devices_prc = np.array(devices_prc)

    prc_thresh = {
                'Threshold' : thresh_prc,
                'Precision' : precision,
                'Recall' : recall,
                'Que' : devices_prc
               }

    df_prc_thresh = pd.DataFrame.from_dict(prc_thresh)

    if print_out:
        print('------------------------------------------------------------')
        print(model, '\n')
        print('Confusion Matrix:')
        print(metrics.confusion_matrix(y_test, y_pred))
        print('\nClassification Report:')
        print(metrics.classification_report(y_test, y_pred))
        print('\nMetrics:')
        print(df_metrics)

        print('\nROC Thresholds:\n')
        print(df_roc_thresh[['Threshold', 'TP', 'FP', 'TN', 'FN', 'TPR', 'FPR', 'TNR','FNR', 'Que']
])

        print('\nPrecision-Recall Thresholds:\n')
        print(df_prc_thresh[['Threshold', 'Precision', 'Recall', 'Que']])

    if plot_out:
        fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2, sharex=False, sharey=False )
        fig.set_size_inches(10,10)

        ax1.plot(fpr, tpr, color='darkorange', lw=2, label='AUC = %0.2f'% roc_auc)
        ax1.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
        ax1.set_xlim([-0.05, 1.0])
        ax1.set_ylim([0.0, 1.05])
        ax1.set_xlabel('False Positive Rate')
        ax1.set_ylabel('True Positive Rate')
        ax1.legend(loc="lower right", fontsize='small')

        ax2.plot(recall, precision, color='blue', lw=2, label='Precision-Recall curve')
        ax2.set_xlim([0.0, 1.0])
        ax2.set_ylim([0.0, 1.05])
        ax2.set_xlabel('Recall')
        ax2.set_ylabel('Precision')
        ax2.legend(loc="lower left", fontsize='small')

        ax3.plot(thresh_roc, fpr, color='red', lw=2, label='FPR')
        ax3.plot(thresh_roc, tpr, color='green',label='TPR')
        ax3.plot(thresh_roc, roc_que, color='blue',label='Devices')
        ax3.set_ylim([0.0, 1.05])
        ax3.set_xlabel('Threshold')
        ax3.set_ylabel('%')
```

```
        ax3.legend(loc='upper right', fontsize='small')

        ax4.plot(thresh_prc, precision, color='red', lw=2, label='Precision')
        ax4.plot(thresh_prc, recall, color='green',label='Recall')
        ax4.plot(thresh_prc, devices_prc, color='blue',label='Devices')
        ax4.set_ylim([0.0, 1.05])
        ax4.set_xlabel('Threshold')
        ax4.set_ylabel('%')
        ax4.legend(loc='lower left', fontsize='small')

    return  df_metrics, df_roc_thresh, df_prc_thresh
```

3.1 Logistic Regression

X_train, X_test, y_train, y_test

In [90]:

```
model = 'Logistic Regression B'
from sklearn.linear_model import LogisticRegression
clf_lgrb = LogisticRegression(random_state=RANDOM_STATE)
gs_params = {'C': [.01, 0.1, 1.0, 10], 'solver': ['liblinear', 'lbfgs']}
gs_score = 'roc_auc'

clf_lgrb, pred_lgrb = bin_classify(model, clf_lgrb, X_train.columns, params=gs_params, score=gs_sco
re)
print('\nBest Parameters:\n',clf_lgrb)

metrics_lgrb, roc_lgrb, prc_lgrb = bin_metrics(model, y_test, pred_lgrb.y_pred, pred_lgrb.y_score,
print_out=True, plot_out=True)
```

```
Best Parameters:
 LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=110, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
-----------------------------------------------------------
Logistic Regression B

Confusion Matrix:
[[211   2]
 [ 14   7]]

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.96       213
           1       0.78      0.33      0.47        21

    accuracy                           0.93       234
   macro avg       0.86      0.66      0.72       234
weighted avg       0.92      0.93      0.92       234


Metrics:
           Logistic Regression B
Accuracy       0.931623931623932
Precision      0.777777777777778
Recall         0.333333333333333
F1 Score       0.466666666666667
ROC AUC        0.776883523362397

ROC Thresholds:

            Threshold  TP  FP  TN  FN               TPR               FPR  \
0   1.990933366407582   0  25  75   0  0.000000000000000  0.000000000000000
1   0.990933366407582   0  25  74   0  0.000000000000000  0.004694835680751
2   0.570805899982113   7  17  74   0  0.285714285714286  0.004694835680751
3   0.546557874028760   7  17  74   0  0.285714285714286  0.009389671361502
4   0.527724551622573   8  16  74   0  0.333333333333333  0.009389671361502
5   0.482770692644816   8  16  73   1  0.333333333333333  0.014084507042254
6   0.420177951005166   9  15  73   1  0.380952380952381  0.014084507042254
7   0.411251246678592   9  15  73   1  0.380952380952381  0.018779342723005
```

8	0.299410065943001	13	11	73	1	0.523809523809524	0.018779342723005
9	0.243394603458738	13	11	71	3	0.523809523809524	0.046948356807512
10	0.224815973932152	14	10	71	3	0.571428571428571	0.046948356807512
11	0.167740303441073	14	10	69	5	0.571428571428571	0.075117370892019
12	0.163148501452955	15	9	69	5	0.619047619047619	0.075117370892019
13	0.065622210476688	15	9	58	16	0.619047619047619	0.215962441314554
14	0.063436142723140	16	8	58	16	0.666666666666667	0.215962441314554
15	0.052886015896892	16	8	56	18	0.666666666666667	0.248826291079812
16	0.052664658742359	17	7	56	18	0.714285714285714	0.248826291079812
17	0.045295383725943	17	7	44	30	0.714285714285714	0.403755868544601
18	0.045132590243583	19	5	44	30	0.761904761904762	0.403755868544601
19	0.038339155196203	19	5	32	42	0.761904761904762	0.563380281690141
20	0.037772280018154	20	4	32	42	0.809523809523810	0.563380281690141
21	0.036447671252431	20	4	29	45	0.809523809523810	0.605633802816901
22	0.036141398557367	21	3	29	45	0.857142857142857	0.605633802816901
23	0.033173730614800	21	3	22	52	0.857142857142857	0.704225352112676
24	0.033075061946045	22	2	22	52	0.904761904761905	0.704225352112676
25	0.028777858119225	22	2	11	63	0.904761904761905	0.845070422535211
26	0.028609236525732	23	1	11	63	0.952380952380952	0.845070422535211
27	0.027650287566671	23	1	9	65	0.952380952380952	0.868544600938967
28	0.027567238203976	25	0	9	65	1.000000000000000	0.868544600938967
29	0.005421878363778	25	0	0	75	1.000000000000000	1.000000000000000

	TNR	FNR	Que
0	1.000000000000000	0.750000000000000	0.000000000000000
1	1.000000000000000	0.747474747474748	0.004273504273504
2	1.000000000000000	0.813186813186813	0.029914529914530
3	1.000000000000000	0.813186813186813	0.034188034188034
4	1.000000000000000	0.822222222222222	0.038461538461538
5	0.986486486486487	0.820224719101124	0.042735042735043
6	0.986486486486487	0.829545454545455	0.047008547008547
7	0.986486486486487	0.829545454545455	0.051282051282051
8	0.986486486486487	0.869047619047619	0.064102564102564
9	0.959459459459459	0.865853658536585	0.089743589743590
10	0.959459459459459	0.876543209876543	0.094017094017094
11	0.932432432432432	0.873417721518987	0.119658119658120
12	0.932432432432432	0.884615384615385	0.123931623931624
13	0.783783783783784	0.865671641791045	0.252136752136752
14	0.783783783783784	0.878787878787879	0.256410256410256
15	0.756756756756757	0.875000000000000	0.286324786324786
16	0.756756756756757	0.888888888888889	0.290598290598291
17	0.594594594594595	0.862745098039216	0.431623931623932
18	0.594594594594595	0.897959183673469	0.435897435897436
19	0.432432432432432	0.864864864864865	0.581196581196581
20	0.432432432432432	0.888888888888889	0.585470085470085
21	0.391891891891892	0.878787878787879	0.623931623931624
22	0.391891891891892	0.906250000000000	0.628205128205128
23	0.297297297297297	0.880000000000000	0.717948717948718
24	0.297297297297297	0.916666666666667	0.722222222222222
25	0.148648648648649	0.846153846153846	0.850427350427350
26	0.148648648648649	0.916666666666667	0.854700854700855
27	0.121621621621622	0.900000000000000	0.876068376068376
28	0.121621621621622	1.000000000000000	0.880341880341880
29	0.000000000000000	NaN	1.000000000000000

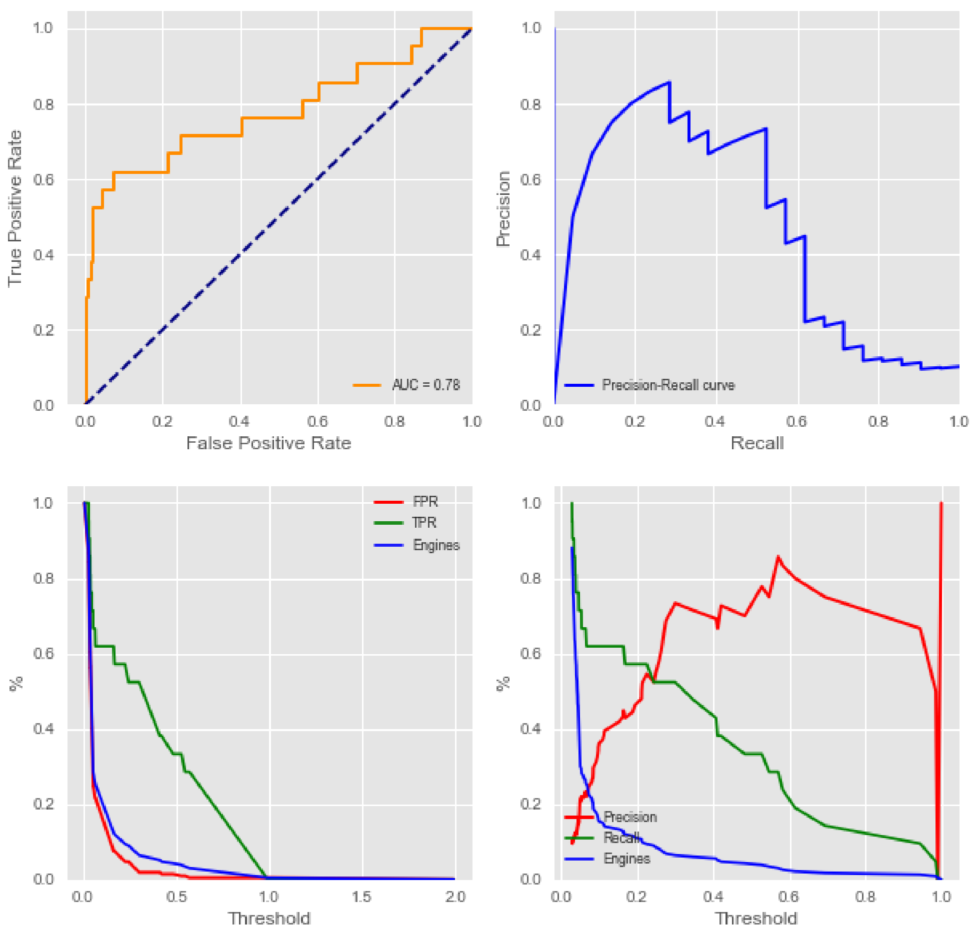
Precision-Recall Thresholds:

	Threshold	Precision	Recall \
0	0.027567238203976	0.101941747572816	1.000000000000000
1	0.027650287566671	0.097560975609756	0.952380952380952
2	0.028077837944239	0.098039215686275	0.952380952380952
3	0.028165149879066	0.098522167487685	0.952380952380952
4	0.028291588781657	0.099009900990099	0.952380952380952
5	0.028525513731767	0.099502487562189	0.952380952380952
6	0.028609236525732	0.100000000000000	0.952380952380952
7	0.028777858119225	0.095477386934673	0.904761904761905
8	0.028849606516354	0.095959595959596	0.904761904761905
9	0.028880406278028	0.096446700507614	0.904761904761905
10	0.028927427922907	0.096938775510204	0.904761904761905
11	0.028929500342682	0.097435897435897	0.904761904761905
12	0.029208194833608	0.097938144329897	0.904761904761905
13	0.029434049763566	0.098445595854922	0.904761904761905
14	0.029692227377960	0.098958333333333	0.904761904761905
15	0.029759021505965	0.099476439790576	0.904761904761905
16	0.029798453720001	0.100000000000000	0.904761904761905
17	0.029816259302062	0.100529100529101	0.904761904761905
18	0.030292815316540	0.101063829787234	0.904761904761905

19	0.030363085997867	0.101604278074866	0.904761904761905
20	0.030442102507605	0.102150537634409	0.904761904761905
21	0.030455980628689	0.102702702702703	0.904761904761905
22	0.030579816609160	0.103260869565217	0.904761904761905
23	0.030582411127082	0.103825136612022	0.904761904761905
24	0.030602750978929	0.104395604395604	0.904761904761905
25	0.030652579081067	0.104972375690608	0.904761904761905
26	0.030839900516405	0.105555555555556	0.904761904761905
27	0.030881576035184	0.106145251396648	0.904761904761905
28	0.030953110131438	0.106741573033708	0.904761904761905
29	0.030981278263886	0.107344632768362	0.904761904761905
..	...	...	...
177	0.163148501452955	0.448275862068966	0.619047619047619
178	0.167740303441073	0.428571428571429	0.571428571428571
179	0.187519866904086	0.444444444444444	0.571428571428571
180	0.193046043699411	0.461538461538462	0.571428571428571
181	0.210914441875792	0.480000000000000	0.571428571428571
182	0.211203722490775	0.500000000000000	0.571428571428571
183	0.212718914914697	0.521739130434783	0.571428571428571
184	0.224815973932152	0.545454545454545	0.571428571428571
185	0.243394603458738	0.523809523809524	0.523809523809524
186	0.248736366656993	0.550000000000000	0.523809523809524
187	0.256918072030372	0.578947368421053	0.523809523809524
188	0.263623500293050	0.611111111111111	0.523809523809524
189	0.268879172784122	0.647058823529412	0.523809523809524
190	0.275429285075221	0.687500000000000	0.523809523809524
191	0.299410065943001	0.733333333333333	0.523809523809524
192	0.348123042290649	0.714285714285714	0.476190476190476
193	0.407360637524883	0.692307692307692	0.428571428571429
194	0.411251246678592	0.666666666666667	0.380952380952381
195	0.420177951005166	0.727272727272727	0.380952380952381
196	0.482770692644816	0.700000000000000	0.333333333333333
197	0.527724551622573	0.777777777777778	0.333333333333333
198	0.546557874028760	0.750000000000000	0.285714285714286
199	0.570805899982113	0.857142857142857	0.285714285714286
200	0.582490911633483	0.833333333333333	0.238095238095238
201	0.614992605264941	0.800000000000000	0.190476190476190
202	0.693507794832522	0.750000000000000	0.142857142857143
203	0.944683887423743	0.666666666666667	0.095238095238095
204	0.985697108772448	0.500000000000000	0.047619047619048
205	0.990933366407582	0.000000000000000	0.000000000000000
206	1.000000000000000	1.000000000000000	0.000000000000000

	Que
0	0.880341880341880
1	0.876068376068376
2	0.871794871794872
3	0.867521367521368
4	0.863247863247863
5	0.858974358974359
6	0.854700854700855
7	0.850427350427350
8	0.846153846153846
9	0.841880341880342
10	0.837606837606838
11	0.833333333333333
12	0.829059829059829
13	0.824786324786325
14	0.820512820512820
15	0.816239316239316
16	0.811965811965812
17	0.807692307692308
18	0.803418803418803
19	0.799145299145299
20	0.794871794871795
21	0.790598290598291
22	0.786324786324786
23	0.782051282051282
24	0.777777777777778
25	0.773504273504274
26	0.769230769230769
27	0.764957264957265
28	0.760683760683761
29	0.756410256410256
..	...
177	0.123931623931624
178	0.119658119658120

```
179   0.115384615384615
180   0.111111111111111
181   0.106837606837607
182   0.102564102564103
183   0.098290598290598
184   0.094017094017094
185   0.089743589743590
186   0.085470085470085
187   0.081196581196581
188   0.076923076923077
189   0.072649572649573
190   0.068376068376068
191   0.064102564102564
192   0.059829059829060
193   0.055555555555556
194   0.051282051282051
195   0.047008547008547
196   0.042735042735043
197   0.038461538461538
198   0.034188034188034
199   0.029914529914530
200   0.025641025641026
201   0.021367521367521
202   0.017094017094017
203   0.012820512820513
204   0.008547008547009
205   0.004273504273504
206   0.000000000000000

[207 rows x 4 columns]
```



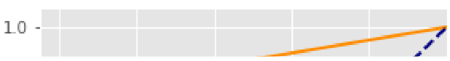
3.2 Decision Tree

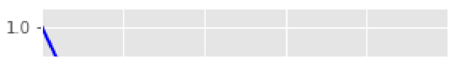
In [161]:

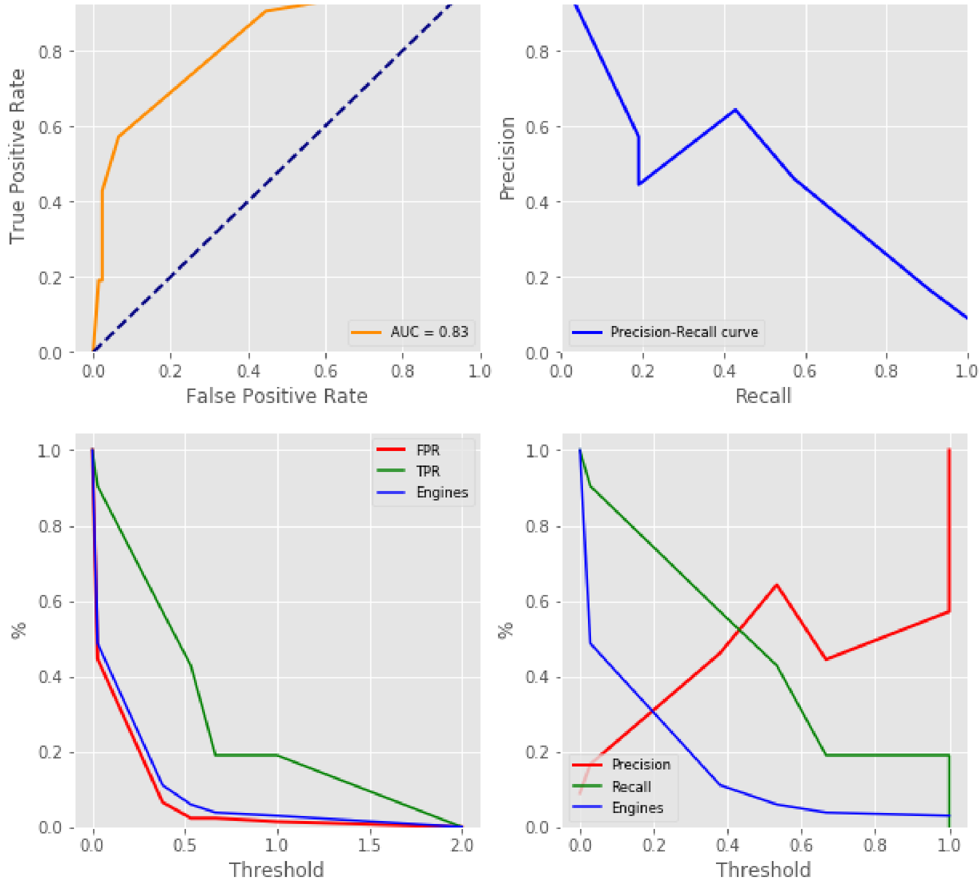
```
model = 'Decision Tree B'
clf_dtrb = DecisionTreeClassifier(random_state=RANDOM_STATE)
gs_params = {'max_depth': [2, 3, 4, 5, 6], 'criterion': ['gini', 'entropy']}
gs_score = 'roc_auc'
```

```python
gs_score = 'roc_auc'
features_orig = X_train.columns

clf_dtrb, pred_dtrb = bin_classify(model, clf_dtrb, features_orig, params=gs_params, score=gs_score
)
print('\nBest Parameters:\n',clf_dtrb)

metrics_dtrb, roc_dtrb, prc_dtrb = bin_metrics(model, y_test, pred_dtrb.y_pred, pred_dtrb.y_score,
print_out=True, plot_out=True)
```

```
Best Parameters:
 DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=5,
                       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=110, splitter='best')
-----------------------------------------------------------
Decision Tree B

Confusion Matrix:
[[208   5]
 [ 12   9]]

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       213
           1       0.64      0.43      0.51        21

    accuracy                           0.93       234
   macro avg       0.79      0.70      0.74       234
weighted avg       0.92      0.93      0.92       234


Metrics:
             Decision Tree B
Accuracy   0.927350427350427
Precision  0.642857142857143
Recall     0.428571428571429
F1 Score   0.514285714285714
ROC AUC    0.832550860719875

ROC Thresholds:

           Threshold  TP  FP  TN  FN               TPR               FPR  \
0  2.000000000000000   0  25  75   0  0.000000000000000  0.000000000000000
1  1.000000000000000   4  20  73   1  0.190476190476190  0.014084507042254
2  0.666666666666667   4  20  73   1  0.190476190476190  0.023474178403756
3  0.533333333333333  10  14  73   1  0.428571428571429  0.023474178403756
4  0.380000000000000  14  10  70   4  0.571428571428571  0.065727699530516
5  0.028391167192429  22   2  41  33  0.904761904761905  0.446009389671362
6  0.000000000000000  25   0   0  75  1.000000000000000  1.000000000000000

                 TNR               FNR               Que
0  1.000000000000000  0.750000000000000  0.000000000000000
1  0.986486486486487  0.784946236559140  0.029914529914530
2  0.986486486486487  0.784946236559140  0.038461538461538
3  0.986486486486487  0.839080459770115  0.059829059829060
4  0.945945945945946  0.875000000000000  0.111111111111111
5  0.554054054054054  0.953488372093023  0.487179487179487
6  0.000000000000000                NaN  1.000000000000000

Precision-Recall Thresholds:

           Threshold          Precision             Recall               Que
0  0.000000000000000  0.089743589743590  1.000000000000000  1.000000000000000
1  0.028391167192429  0.166666666666667  0.904761904761905  0.487179487179487
2  0.380000000000000  0.461538461538462  0.571428571428571  0.111111111111111
3  0.533333333333333  0.642857142857143  0.428571428571429  0.059829059829060
4  0.666666666666667  0.444444444444444  0.190476190476190  0.038461538461538
5  1.000000000000000  0.571428571428571  0.190476190476190  0.029914529914530
6  1.000000000000000  1.000000000000000  0.000000000000000  0.029914529914530
```







3.3 Random Forest

In [94]:

```
model = 'Random Forest B'
clf_rfcb = RandomForestClassifier(n_estimators=50, random_state=123)
gs_params = {'max_depth': [4, 5, 6, 7, 8], 'criterion': ['gini', 'entropy']}
gs_score = 'roc_auc'
features_orig = X_train.columns

clf_rfcb, pred_rfcb = bin_classify(model, clf_rfcb, features_orig, params=gs_params, score=gs_score
)
print('\nBest Parameters:\n',clf_rfcb)

metrics_rfcb, roc_rfcb, prc_rfcb = bin_metrics(model, y_test, pred_rfcb.y_pred, pred_rfcb.y_score,
print_out=True, plot_out=True)
```

```
Best Parameters:
 RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=6, 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=50,
                       n_jobs=None, oob_score=False, random_state=123,
                       verbose=0, warm_start=False)
-----------------------------------------------------------
Random Forest B

Confusion Matrix:
[[212   1]
 [ 13   8]]

Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       213
           1       0.89      0.38      0.53        21

    accuracy                           0.94       234
   macro avg       0.92      0.69      0.75       234
weighted avg       0.94      0.94      0.93       234
```

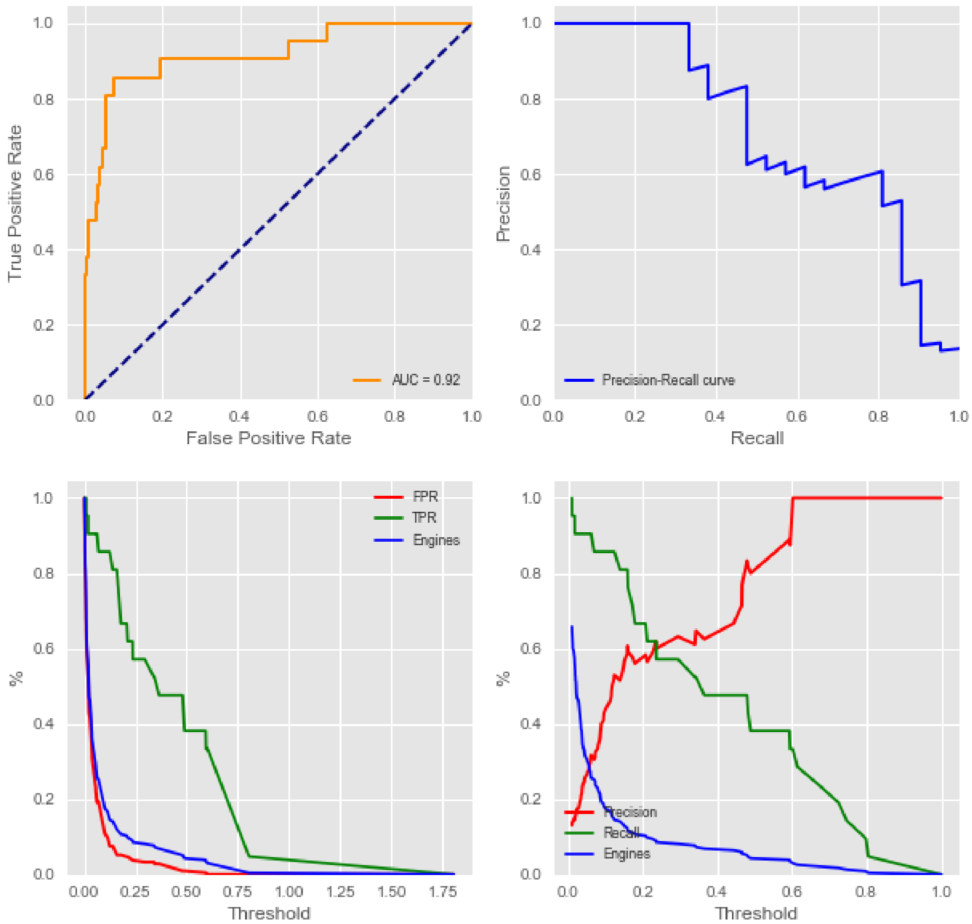
```
weighted avg       0.94      0.94      0.93       234


Metrics:
             Random Forest B
Accuracy   0.940170940170940
Precision  0.888888888888889
Recall     0.380952380952381
F1 Score   0.533333333333333
ROC AUC    0.917057902973396

ROC Thresholds:

            Threshold  TP  FP  TN  FN                TPR                FPR  \
0   1.804410532114944   0  25  75   0  0.000000000000000  0.000000000000000
1   0.804410532114944   1  23  75   0  0.047619047619048  0.000000000000000
2   0.602213899382518   8  16  75   0  0.333333333333333  0.000000000000000
3   0.594872438804659   8  16  74   0  0.333333333333333  0.004694835680751
4   0.593260073260073   9  15  74   0  0.380952380952381  0.004694835680751
5   0.489403677970333   9  15  74   0  0.380952380952381  0.009389671361502
6   0.479744215022564  11  13  74   0  0.476190476190476  0.009389671361502
7   0.364787985629807  11  13  72   2  0.476190476190476  0.028169014084507
8   0.342591304347826  13  11  72   2  0.523809523809524  0.028169014084507
9   0.339967273134745  13  11  72   2  0.523809523809524  0.032863849765258
10  0.294332513339110  14  10  72   2  0.571428571428571  0.032863849765258
11  0.236585712982126  14  10  72   2  0.571428571428571  0.037558685446009
12  0.235617403995122  15   9  72   2  0.619047619047619  0.037558685446009
13  0.212130254131098  15   9  71   3  0.619047619047619  0.046948356807512
14  0.207838694028796  16   8  71   3  0.666666666666667  0.046948356807512
15  0.178924806376419  16   8  71   3  0.666666666666667  0.051643192488263
16  0.159151172937632  20   4  71   3  0.809523809523810  0.051643192488263
17  0.138901162567151  20   4  69   5  0.809523809523810  0.075117370892019
18  0.123866000833548  21   3  69   5  0.857142857142857  0.075117370892019
19  0.113247739721307  21   3  67   7  0.857142857142857  0.098591549295775
20  0.100278430131696  21   3  66   8  0.857142857142857  0.107981220657277
21  0.069295631210124  21   3  60  14  0.857142857142857  0.192488262910798
22  0.061583354709069  22   2  60  14  0.904761904761905  0.192488262910798
23  0.056555939517372  22   2  58  16  0.904761904761905  0.220657276995305
24  0.055787726739894  22   2  57  17  0.904761904761905  0.230046948356808
25  0.036882411590128  22   2  51  23  0.904761904761905  0.309859154929577
26  0.036880856849515  22   2  51  23  0.904761904761905  0.319248826291080
27  0.034435671652704  22   2  48  26  0.904761904761905  0.347417840375587
28  0.033461659657629  22   2  47  27  0.904761904761905  0.361502347417840
29  0.031179471330827  22   2  46  28  0.904761904761905  0.384976525821596
..                ...  ..  ..  ..  ..                ...                ...
41  0.012430923627616  23   1  31  43  0.952380952380952  0.577464788732394
42  0.011891891891892  23   1  30  44  0.952380952380952  0.586854460093897
43  0.010935193553279  23   1  29  45  0.952380952380952  0.610328638497653
44  0.010292106736558  23   1  28  46  0.952380952380952  0.624413145539906
45  0.010130264451371  25   0  28  46  1.000000000000000  0.624413145539906
46  0.009884325095467  25   0  27  47  1.000000000000000  0.629107981220657
47  0.009465593214362  25   0  26  48  1.000000000000000  0.643192488262911
48  0.008913848145614  25   0  26  48  1.000000000000000  0.647887323943662
49  0.008901833169578  25   0  25  49  1.000000000000000  0.661971830985915
50  0.008264852400873  25   0  23  51  1.000000000000000  0.680751173708920
51  0.008213767024400  25   0  23  51  1.000000000000000  0.690140845070423
52  0.007901026180746  25   0  21  53  1.000000000000000  0.708920187793427
53  0.007387320743408  25   0  20  54  1.000000000000000  0.723004694835681
54  0.006777443448234  25   0  19  55  1.000000000000000  0.746478873239437
55  0.006690851901993  25   0  18  56  1.000000000000000  0.755868544600939
56  0.005977443448234  25   0  17  57  1.000000000000000  0.760563380281690
57  0.005708359976105  25   0  16  58  1.000000000000000  0.784037558685446
58  0.004246336524795  25   0  14  60  1.000000000000000  0.802816901408451
59  0.004000000000000  25   0  14  60  1.000000000000000  0.812206572769953
60  0.003784777243593  25   0  13  61  1.000000000000000  0.821596244131455
61  0.003337824258929  25   0  12  62  1.000000000000000  0.830985915492958
62  0.003141690426873  25   0  11  63  1.000000000000000  0.840375586854460
63  0.003024794325036  25   0  11  63  1.000000000000000  0.845070422535211
64  0.002602901772399  25   0  10  64  1.000000000000000  0.854460093896714
65  0.002381707508316  25   0  10  64  1.000000000000000  0.863849765258216
66  0.002347112184625  25   0   7  67  1.000000000000000  0.901408450704225
67  0.001511465796975  25   0   5  69  1.000000000000000  0.929577464788732
68  0.001449153000952  25   0   4  70  1.000000000000000  0.938967136150235
69  0.000073800738007  25   0   1  73  1.000000000000000  0.981220657276995
70  0.000000000000000  25   0   0  75  1.000000000000000  1.000000000000000

                  TNR                FNR                Que
0   1.000000000000000  0.750000000000000  0.000000000000000
```

```
0    1.000000000000000  0.750000000000000  0.000000000000000
1    1.000000000000000  0.765306122448980  0.004273504273504
2    1.000000000000000  0.824175824175824  0.029914529914530
3    1.000000000000000  0.822222222222222  0.034188034188034
4    1.000000000000000  0.831460674157303  0.038461538461538
5    1.000000000000000  0.831460674157303  0.042735042735043
6    1.000000000000000  0.850574712643678  0.051282051282051
7    0.972972972972973  0.847058823529412  0.068376068376068
8    0.972972972972973  0.867469879518072  0.072649572649573
9    0.972972972972973  0.867469879518072  0.076923076923077
10   0.972972972972973  0.878048780487805  0.081196581196581
11   0.972972972972973  0.878048780487805  0.085470085470085
12   0.972972972972973  0.888888888888889  0.089743589743590
13   0.959459459459459  0.887500000000000  0.098290598290598
14   0.959459459459459  0.898734177215190  0.102564102564103
15   0.959459459459459  0.898734177215190  0.106837606837607
16   0.959459459459459  0.946666666666667  0.119658119658120
17   0.932432432432432  0.945205479452055  0.141025641025641
18   0.932432432432432  0.958333333333333  0.145299145299145
19   0.905405405405405  0.957142857142857  0.166666666666667
20   0.891891891891892  0.956521739130435  0.175213675213675
21   0.810810810810811  0.952380952380952  0.252136752136752
22   0.810810810810811  0.967741935483871  0.256410256410256
23   0.783783783783784  0.966666666666667  0.282051282051282
24   0.770270270270270  0.966101694915254  0.290598290598291
25   0.689189189189189  0.962264150943396  0.363247863247863
26   0.689189189189189  0.962264150943396  0.371794871794872
27   0.648648648648649  0.960000000000000  0.397435897435897
28   0.635135135135135  0.959183673469388  0.410256410256410
29   0.621621621621622  0.958333333333333  0.431623931623932
..                 ...                ...                ...
41   0.418918918918919  0.968750000000000  0.611111111111111
42   0.405405405405405  0.967741935483871  0.619658119658120
43   0.391891891891892  0.966666666666667  0.641025641025641
44   0.378378378378378  0.965517241379310  0.653846153846154
45   0.378378378378378  1.000000000000000  0.658119658119658
46   0.364864864864865  1.000000000000000  0.662393162393162
47   0.351351351351351  1.000000000000000  0.675213675213675
48   0.351351351351351  1.000000000000000  0.679487179487180
49   0.337837837837838  1.000000000000000  0.692307692307692
50   0.310810810810811  1.000000000000000  0.709401709401709
51   0.310810810810811  1.000000000000000  0.717948717948718
52   0.283783783783784  1.000000000000000  0.735042735042735
53   0.270270270270270  1.000000000000000  0.747863247863248
54   0.256756756756757  1.000000000000000  0.769230769230769
55   0.243243243243243  1.000000000000000  0.777777777777778
56   0.229729729729730  1.000000000000000  0.782051282051282
57   0.216216216216216  1.000000000000000  0.803418803418803
58   0.189189189189189  1.000000000000000  0.820512820512820
59   0.189189189189189  1.000000000000000  0.829059829059829
60   0.175675675675676  1.000000000000000  0.837606837606838
61   0.162162162162162  1.000000000000000  0.846153846153846
62   0.148648648648649  1.000000000000000  0.854700854700855
63   0.148648648648649  1.000000000000000  0.858974358974359
64   0.135135135135135  1.000000000000000  0.867521367521368
65   0.135135135135135  1.000000000000000  0.876068376068376
66   0.094594594594595  1.000000000000000  0.910256410256410
67   0.067567567567568  1.000000000000000  0.935897435897436
68   0.054054054054054  1.000000000000000  0.944444444444444
69   0.013513513513514  1.000000000000000  0.982905982905983
70   0.000000000000000                NaN  1.000000000000000

[71 rows x 10 columns]
```

```
Precision-Recall Thresholds:

             Threshold          Precision             Recall  \
0    0.010130264451371  0.136363636363636  1.000000000000000
1    0.010292106736558  0.130718954248366  0.952380952380952
2    0.010347112184625  0.131578947368421  0.952380952380952
3    0.010883062175831  0.132450331125828  0.952380952380952
4    0.010935193553279  0.133333333333333  0.952380952380952
5    0.011891891891892  0.137931034482759  0.952380952380952
6    0.012430923627616  0.139860139860140  0.952380952380952
7    0.012438496421083  0.140845070422535  0.952380952380952
8    0.012440764611330  0.141843971631206  0.952380952380952
9    0.012987041646855  0.142857142857143  0.952380952380952
10   0.015087453155515  0.143884892086331  0.952380952380952
```

```
10   0.015087453155515  0.143884892086331  0.952380952380952
11   0.015228278924631  0.144927536231884  0.952380952380952
12   0.016146949483551  0.145985401459854  0.952380952380952
13   0.016482084434247  0.147058823529412  0.952380952380952
14   0.017313774739489  0.148148148148148  0.952380952380952
15   0.017393278558310  0.149253731343284  0.952380952380952
16   0.017647290301027  0.150375939849624  0.952380952380952
17   0.017944481478975  0.151515151515152  0.952380952380952
18   0.018232936094431  0.145038167938931  0.904761904761905
19   0.018935193553279  0.146153846153846  0.904761904761905
20   0.019062081297812  0.148437500000000  0.904761904761905
21   0.019148497939736  0.149606299212598  0.904761904761905
22   0.019326251068330  0.150793650793651  0.904761904761905
23   0.019365883176411  0.152000000000000  0.904761904761905
24   0.019379624739870  0.153225806451613  0.904761904761905
25   0.020000000000000  0.154471544715447  0.904761904761905
26   0.020044573223700  0.155737704918033  0.904761904761905
27   0.020123671497585  0.157024793388430  0.904761904761905
28   0.020987679801871  0.158333333333333  0.904761904761905
29   0.021040609546860  0.159663865546218  0.904761904761905
..                 ...                ...                ...
110  0.158048782428298  0.586206896551724  0.809523809523810
111  0.159151172937632  0.607142857142857  0.809523809523810
112  0.160292542517549  0.592592592592593  0.761904761904762
113  0.172421459258621  0.576923076923077  0.714285714285714
114  0.178924806376419  0.560000000000000  0.666666666666667
115  0.207838694028796  0.583333333333333  0.666666666666667
116  0.212130254131098  0.565217391304348  0.619047619047619
117  0.227469989253648  0.590909090909091  0.619047619047619
118  0.235617403995122  0.619047619047619  0.619047619047619
119  0.236585712982126  0.600000000000000  0.571428571428571
120  0.294332513339110  0.631578947368421  0.571428571428571
121  0.339967273134745  0.611111111111111  0.523809523809524
122  0.342591304347826  0.647058823529412  0.523809523809524
123  0.364787985629807  0.625000000000000  0.476190476190476
124  0.443625980855935  0.666666666666667  0.476190476190476
125  0.465400462147041  0.714285714285714  0.476190476190476
126  0.466095238095238  0.769230769230769  0.476190476190476
127  0.479744215022564  0.833333333333333  0.476190476190476
128  0.482639047025554  0.818181818181818  0.428571428571429
129  0.489403677970333  0.800000000000000  0.380952380952381
130  0.593260073260073  0.888888888888889  0.380952380952381
131  0.594872438804659  0.875000000000000  0.333333333333333
132  0.602213899382518  1.000000000000000  0.333333333333333
133  0.614592377763853  1.000000000000000  0.285714285714286
134  0.671729346104346  1.000000000000000  0.238095238095238
135  0.725003663003663  1.000000000000000  0.190476190476190
136  0.746746126567324  1.000000000000000  0.142857142857143
137  0.798209364470426  1.000000000000000  0.095238095238095
138  0.804410532114944  1.000000000000000  0.047619047619048
139  1.000000000000000  1.000000000000000  0.000000000000000

                   Que
0    0.658119658119658
1    0.653846153846154
2    0.649572649572650
3    0.645299145299145
4    0.641025641025641
5    0.619658119658120
6    0.611111111111111
7    0.606837606837607
8    0.602564102564103
9    0.598290598290598
10   0.594017094017094
11   0.589743589743590
12   0.585470085470085
13   0.581196581196581
14   0.576923076923077
15   0.572649572649573
16   0.568376068376068
17   0.564102564102564
18   0.559829059829060
19   0.555555555555556
20   0.547008547008547
21   0.542735042735043
22   0.538461538461538
23   0.534188034188034
24   0.529914529914530
```

```
24    0.529914529914530
25    0.525641025641026
26    0.521367521367521
27    0.517094017094017
28    0.512820512820513
29    0.508547008547009
..                  ...
110   0.123931623931624
111   0.119658119658120
112   0.115384615384615
113   0.111111111111111
114   0.106837606837607
115   0.102564102564103
116   0.098290598290598
117   0.094017094017094
118   0.089743589743590
119   0.085470085470085
120   0.081196581196581
121   0.076923076923077
122   0.072649572649573
123   0.068376068376068
124   0.064102564102564
125   0.059829059829060
126   0.055555555555556
127   0.051282051282051
128   0.047008547008547
129   0.042735042735043
130   0.038461538461538
131   0.034188034188034
132   0.029914529914530
133   0.025641025641026
134   0.021367521367521
135   0.017094017094017
136   0.012820512820513
137   0.008547008547009
138   0.004273504273504
139   0.000000000000000

[140 rows x 4 columns]
```



3.34 light GBM

In [148]:

```
import sys
!conda install --yes --prefix {sys.prefix} -c conda-forge lightgbm
```

```
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.5.11
  latest version: 4.6.14

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: /Users/simon/anaconda3

  added / updated specs:
    - lightgbm


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    xz-5.2.4                   |      h1de35cc_1001         268 KB  conda-forge
    curl-7.64.0                |         heae2a1f_2         138 KB  conda-forge
    python-3.6.7               |      h8dc6b48_1004        20.5 MB  conda-forge
    requests-2.22.0            |           py36_0          84 KB  conda-forge
    readline-7.0               |      hcfe32e1_1001         393 KB  conda-forge
    libcurl-7.64.0             |         he376013_2         533 KB  conda-forge
    libedit-3.1.20170329       |      hcfe32e1_1001         152 KB  conda-forge
    numpy-1.14.2               |   py36ha9ae307_0         3.9 MB
    openssl-1.1.1b             |         h01d97ff_2         3.5 MB  conda-forge
    llvmdev-4.0.0              |          default_0       100.9 MB  conda-forge
    openmp-4.0.0               |                1         195 KB  conda-forge
    conda-4.6.14               |           py36_0         2.1 MB  conda-forge
    cyrus-sasl-2.1.27          |         h5d77f49_0         211 KB  conda-forge
    ca-certificates-2019.3.9   |         hecc5488_0         146 KB  conda-forge
    cryptography-2.6.1         |   py36h212c5bf_0         564 KB  conda-forge
    ncurses-6.1                |      h0a44026_1002         1.3 MB  conda-forge
    sqlite-3.28.0              |         h9721f7c_0         2.4 MB  conda-forge
    clangdev-4.0.0             |          default_0        62.8 MB  conda-forge
    krb5-1.16.3                |      hcfa6398_1001         1.1 MB  conda-forge
    tk-8.6.9                   |      h2573ce8_1002         3.2 MB  conda-forge
    certifi-2019.3.9           |           py36_0         149 KB  conda-forge
    pycurl-7.43.0.2            |   py36ha12b0ac_0         185 KB
    lightgbm-2.2.3             |   py36h0a44026_0         656 KB  conda-forge
    ------------------------------------------------------------
                                           Total:       205.2 MB

The following NEW packages will be INSTALLED:

    clangdev:        4.0.0-default_0       conda-forge
    libcurl:         7.64.0-he376013_2     conda-forge
    lightgbm:        2.2.3-py36h0a44026_0  conda-forge
    llvmdev:         4.0.0-default_0       conda-forge
    openmp:          4.0.0-1               conda-forge

The following packages will be UPDATED:

    ca-certificates: 2018.03.07-0                            --> 2019.3.9-hecc5488_0        conda-forge
    certifi:         2018.8.24-py36_1                        --> 2019.3.9-py36_0            conda-forge
    conda:           4.5.11-py36_0                           --> 4.6.14-py36_0              conda-forge
    cryptography:    2.0.3-py36h22d4226_1                    --> 2.6.1-py36h212c5bf_0       conda-forge
    curl:            7.55.1-h7601780_3                       --> 7.64.0-heae2a1f_2          conda-forge
    cyrus-sasl:      2.1.26-ha054001_1                       --> 2.1.27-h5d77f49_0          conda-forge
    krb5:            1.14.2-hc0fd8ed_4                       --> 1.16.3-hcfa6398_1001       conda-forge
```

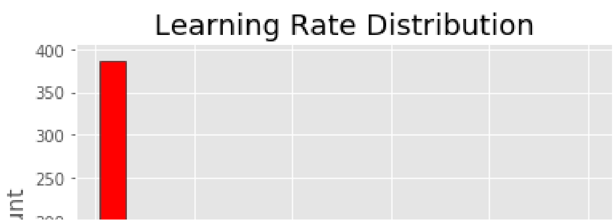
```
    libedit:          3.1-hb4e282d_0                   --> 3.1.20170329-hcfe32e1_1001 conda-forge
    ncurses:          6.0-ha932d30_1                   --> 6.1-h0a44026_1002          conda-forge
    openssl:          1.0.2p-h1de35cc_0                --> 1.1.1b-h01d97ff_2          conda-forge
    pycurl:           7.43.0-py36hdb90038_3            --> 7.43.0.2-py36ha12b0ac_0
    python:           3.6.3-h6804ab2_0                 --> 3.6.7-h8dc6b48_1004        conda-forge
    readline:         7.0-h81b24a6_3                   --> 7.0-hcfe32e1_1001          conda-forge
    requests:         2.14.2-py36_0                    --> 2.22.0-py36_0              conda-forge
    sqlite:           3.20.1-h900c3b0_1                --> 3.28.0-h9721f7c_0          conda-forge
    tk:               8.6.7-hcdce994_1                 --> 8.6.9-h2573ce8_1002        conda-forge
    xz:               5.2.3-ha24016e_1                 --> 5.2.4-h1de35cc_1001        conda-forge

The following packages will be DOWNGRADED:

    numpy:            1.14.3-py36he6379a5_1            --> 1.14.2-py36ha9ae307_0


Downloading and Extracting Packages
xz-5.2.4             | 268 KB    | ##################################### | 100%
curl-7.64.0          | 138 KB    | ##################################### | 100%
python-3.6.7         | 20.5 MB   | ##################################### | 100%
requests-2.22.0      | 84 KB     | ##################################### | 100%
readline-7.0         | 393 KB    | ##################################### | 100%
libcurl-7.64.0       | 533 KB    | ##################################### | 100%
libedit-3.1.20170329 | 152 KB    | ##################################### | 100%
numpy-1.14.2         | 3.9 MB    | ##################################### | 100%
openssl-1.1.1b       | 3.5 MB    | ##################################### | 100%
llvmdev-4.0.0        | 100.9 MB  | ##################################### | 100%
openmp-4.0.0         | 195 KB    | ##################################### | 100%
conda-4.6.14         | 2.1 MB    | ##################################### | 100%
cyrus-sasl-2.1.27    | 211 KB    | ##################################### | 100%
ca-certificates-2019 | 146 KB    | ##################################### | 100%
cryptography-2.6.1   | 564 KB    | ##################################### | 100%
ncurses-6.1          | 1.3 MB    | ##################################### | 100%
sqlite-3.28.0        | 2.4 MB    | ##################################### | 100%
clangdev-4.0.0       | 62.8 MB   | ##################################### | 100%
krb5-1.16.3          | 1.1 MB    | ##################################### | 100%
tk-8.6.9             | 3.2 MB    | ##################################### | 100%
certifi-2019.3.9     | 149 KB    | ##################################### | 100%
pycurl-7.43.0.2      | 185 KB    | ##################################### | 100%
lightgbm-2.2.3       | 656 KB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

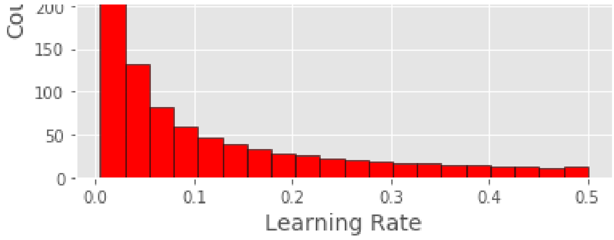
In [146]:

```
# Hyperparameter grid
param_grid = {
    'boosting_type': ['gbdt', 'goss', 'dart'],
    'num_leaves': list(range(20, 150)),
    'learning_rate': list(np.logspace(np.log10(0.005), np.log10(0.5), base = 10, num = 1000)),
    'subsample_for_bin': list(range(20000, 300000, 20000)),
    'min_child_samples': list(range(20, 500, 5)),
    'reg_alpha': list(np.linspace(0, 1)),
    'reg_lambda': list(np.linspace(0, 1)),
    'colsample_bytree': list(np.linspace(0.6, 1, 10)),
    'subsample': list(np.linspace(0.5, 1, 100)),
    'is_unbalance': [True, False]
}
```

In [150]:

```
# Learning rate histogram
plt.hist(gs_params['learning_rate'], bins = 20, color = 'r', edgecolor = 'k');
plt.xlabel('Learning Rate', size = 14); plt.ylabel('Count', size = 14); plt.title('Learning Rate Di
stribution', size = 18);
```





In [165]:

```
import lightgbm as lgb
model = 'LGBM B'
clf_lgbb = lgb.LGBMClassifier()
gs_params = {
    'boosting_type': ['gbdt', 'goss', 'dart'],
    'num_leaves': list(range(20, 150)),
    'learning_rate': list(np.logspace(np.log10(0.005), np.log10(0.5), base = 10, num = 1000)),
    'subsample_for_bin': list(range(20000, 300000, 20000)),
    'min_child_samples': list(range(20, 500, 5)),
    'reg_alpha': list(np.linspace(0, 1)),
    'reg_lambda': list(np.linspace(0, 1)),
    'colsample_bytree': list(np.linspace(0.6, 1, 10)),
    'subsample': list(np.linspace(0.5, 1, 100)),
    'is_unbalance': [True, False]
}
gs_score = 'roc_auc'
features_orig = X_train.columns

clf_lgbb, pred_lgbb = bin_classify(model, clf_lgbb, features_orig, params=gs_params, score=gs_score
)
print('\nBest Parameters:\n',clf_lgbb)

metrics_svcb, roc_svcb, prc_svcb = bin_metricsbin_metricsbin_class_metrics(model, y_test, pred_svcb
.y_pred, pred_svcb.y_score, print_out=True, plot_out=True)
```

3.4 SVC

In [95]:

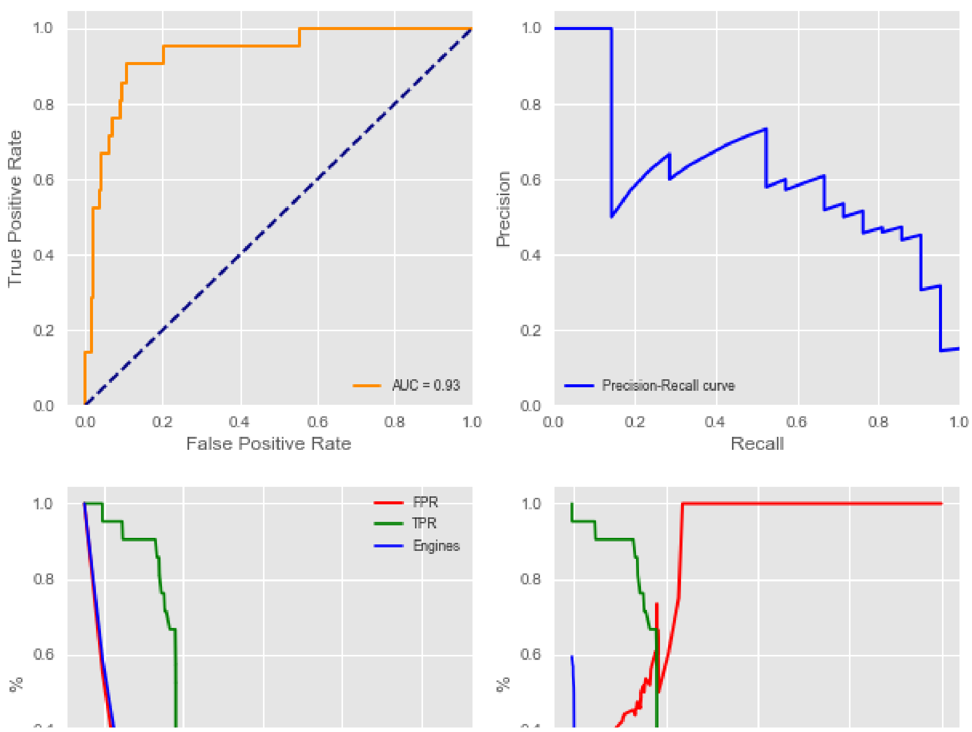
```
model = 'SVC B'
clf_svcb = SVC(kernel='rbf', random_state=123)
gs_params = {'C': [1.0]}
gs_score = 'roc_auc'
features_orig = X_train.columns

clf_svcb, pred_svcb = bin_classify(model, clf_svcb, features_orig, params=gs_params, score=gs_score
)
print('\nBest Parameters:\n',clf_svcb)

metrics_svcb, roc_svcb, prc_svcb = bin_metrics(model, y_test, pred_svcb.y_pred, pred_svcb.y_score,
print_out=True, plot_out=True)
```

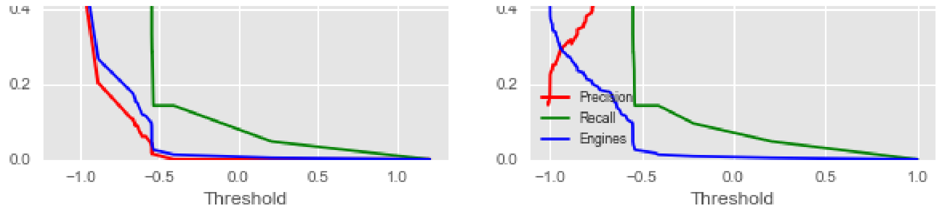
```
Best Parameters:
 SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=123,
    shrinking=True, tol=0.001, verbose=False)
-----------------------------------------------------------
SVC B

Confusion Matrix:
[[213   0]
 [ 20   1]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      1.00      0.96       213
           1       1.00      0.05      0.09        21

    accuracy                           0.91       234
```

```
   macro avg       0.96      0.52      0.52       234
weighted avg       0.92      0.91      0.88       234


Metrics:
                      SVC B
Accuracy   0.914529914529915
Precision  1.000000000000000
Recall     0.047619047619048
F1 Score   0.090909090909091
ROC AUC    0.931589537223340

ROC Thresholds:

            Threshold  TP  FP  TN  FN                TPR                FPR  \
0   1.207600186321636   0  25  75   0  0.000000000000000  0.000000000000000
1   0.207600186321636   1  23  75   0  0.047619047619048  0.000000000000000
2  -0.409482448826507   3  21  75   0  0.142857142857143  0.000000000000000
3  -0.538942849628813   3  21  73   1  0.142857142857143  0.014084507042254
4  -0.545972235720294   7  17  73   1  0.285714285714286  0.014084507042254
5  -0.546989856574733   7  17  73   1  0.285714285714286  0.018779342723005
6  -0.548684246152674   8  16  73   1  0.333333333333333  0.018779342723005
7  -0.548685432879787  10  14  73   1  0.428571428571429  0.018779342723005
8  -0.548685432879977  13  11  73   1  0.523809523809524  0.018779342723005
9  -0.548935464899727  13  11  72   2  0.523809523809524  0.037558685446009
10 -0.549233154802397  14  10  72   2  0.571428571428571  0.037558685446009
11 -0.549314724463178  14  10  71   3  0.571428571428571  0.042253521126761
12 -0.552326387520426  16   8  71   3  0.666666666666667  0.042253521126761
13 -0.586985480144720  16   8  70   4  0.666666666666667  0.061032863849765
14 -0.608763046737940  17   7  70   4  0.714285714285714  0.061032863849765
15 -0.617834858281970  17   7  69   5  0.714285714285714  0.070422535211268
16 -0.622765956767957  19   5  69   5  0.761904761904762  0.070422535211268
17 -0.640131867960532  19   5  68   6  0.761904761904762  0.089201877934272
18 -0.652199618900712  20   4  68   6  0.809523809523810  0.089201877934272
19 -0.652488025867222  20   4  67   7  0.809523809523810  0.093896713615023
20 -0.655216015145488  21   3  67   7  0.857142857142857  0.093896713615023
21 -0.667075331650436  21   3  66   8  0.857142857142857  0.107981220657277
22 -0.677894635045362  22   2  66   8  0.904761904761905  0.107981220657277
23 -0.883306276901718  22   2  59  15  0.904761904761905  0.201877934272300
24 -0.887413192730288  23   1  59  15  0.952380952380952  0.201877934272300
25 -1.011618986207688  23   1  33  41  0.952380952380952  0.553990610328638
26 -1.011835179194099  25   0  33  41  1.000000000000000  0.553990610328638
27 -1.126338283502760  25   0   0  75  1.000000000000000  1.000000000000000

                  TNR                FNR                Que
0   1.000000000000000  0.750000000000000  0.000000000000000
1   1.000000000000000  0.765306122448980  0.004273504273504
2   1.000000000000000  0.781250000000000  0.012820512820513
3   0.986486486486487  0.776595744680851  0.025641025641026
4   0.986486486486487  0.811111111111111  0.038461538461538
5   0.986486486486487  0.811111111111111  0.042735042735043
6   0.986486486486487  0.820224719101124  0.047008547008547
7   0.986486486486487  0.839080459770115  0.055555555555556
8   0.986486486486487  0.869047619047619  0.064102564102564
9   0.972972972972973  0.867469879518072  0.081196581196581
10  0.972972972972973  0.878048780487805  0.085470085470085
11  0.959459459459459  0.876543209876543  0.089743589743590
12  0.959459459459459  0.898734177215190  0.098290598290598
13  0.945945945945946  0.897435897435897  0.115384615384615
14  0.945945945945946  0.909090909090909  0.119658119658120
15  0.932432432432432  0.907894736842105  0.128205128205128
16  0.932432432432432  0.932432432432432  0.132478632478632
17  0.918918918918919  0.931506849315068  0.149572649572650
18  0.918918918918919  0.944444444444444  0.153846153846154
19  0.905405405405405  0.943661971830986  0.158119658119658
20  0.905405405405405  0.957142857142857  0.162393162393162
21  0.891891891891892  0.956521739130435  0.175213675213675
22  0.891891891891892  0.970588235294118  0.179487179487179
23  0.797297297297297  0.967213114754098  0.264957264957265
24  0.797297297297297  0.983333333333333  0.269230769230769
25  0.445945945945946  0.970588235294118  0.589743589743590
26  0.445945945945946  1.000000000000000  0.594017094017094
27  0.000000000000000                NaN  1.000000000000000

Precision-Recall Thresholds:

             Threshold          Precision             Recall  \
```

```
0   -1.011835179194099  0.151079136690647  1.000000000000000
1   -1.011618986207688  0.144927536231884  0.952380952380952
2   -1.010041401604754  0.145985401459854  0.952380952380952
3   -1.009175705666313  0.147058823529412  0.952380952380952
4   -1.008409536519667  0.148148148148148  0.952380952380952
5   -1.007893909864815  0.149253731343284  0.952380952380952
6   -1.005543930839356  0.150375939849624  0.952380952380952
7   -1.005540784852887  0.151515151515152  0.952380952380952
8   -1.005525496651432  0.152671755725191  0.952380952380952
9   -1.005475746422546  0.153846153846154  0.952380952380952
10  -1.005242882544239  0.155038759689922  0.952380952380952
11  -1.005097998521505  0.156250000000000  0.952380952380952
12  -1.004241251997249  0.157480314960630  0.952380952380952
13  -1.004236996620331  0.158730158730159  0.952380952380952
14  -1.003909002237109  0.160000000000000  0.952380952380952
15  -1.003263332995280  0.161290322580645  0.952380952380952
16  -1.002756000186888  0.162601626016260  0.952380952380952
17  -1.001946171965282  0.163934426229508  0.952380952380952
18  -1.001291308570789  0.165289256198347  0.952380952380952
19  -1.001194999678763  0.166666666666667  0.952380952380952
20  -1.000785720915056  0.168067226890756  0.952380952380952
21  -1.000741035388376  0.169491525423729  0.952380952380952
22  -1.000623424455871  0.170940170940171  0.952380952380952
23  -1.000487760498060  0.172413793103448  0.952380952380952
24  -1.000356737906916  0.173913043478261  0.952380952380952
25  -1.000254597786343  0.175438596491228  0.952380952380952
26  -1.000251642663400  0.176991150442478  0.952380952380952
27  -1.000230613531878  0.178571428571429  0.952380952380952
28  -1.000230243306456  0.180180180180180  0.952380952380952
29  -1.000191771919807  0.181818181818182  0.952380952380952
..                 ...                ...                ...
109 -0.617834858281970  0.500000000000000  0.714285714285714
110 -0.614722339173753  0.517241379310345  0.714285714285714
111 -0.608763046737940  0.535714285714286  0.714285714285714
112 -0.586985480144720  0.518518518518518  0.666666666666667
113 -0.583937115309137  0.538461538461538  0.666666666666667
114 -0.581635529498971  0.560000000000000  0.666666666666667
115 -0.569371847342028  0.583333333333333  0.666666666666667
116 -0.552326387520426  0.608695652173913  0.666666666666667
117 -0.549894758420741  0.590909090909091  0.619047619047619
118 -0.549314724463178  0.571428571428571  0.571428571428571
119 -0.549233154802397  0.600000000000000  0.571428571428571
120 -0.548935464899727  0.578947368421053  0.523809523809524
121 -0.548706893733511  0.611111111111111  0.523809523809524
122 -0.548697445759779  0.647058823529412  0.523809523809524
123 -0.548685653781806  0.687500000000000  0.523809523809524
124 -0.548685432879977  0.733333333333333  0.523809523809524
125 -0.548685432879800  0.714285714285714  0.476190476190476
126 -0.548685432879787  0.692307692307692  0.428571428571429
127 -0.548684246152674  0.636363636363636  0.333333333333333
128 -0.546989856574733  0.600000000000000  0.285714285714286
129 -0.545972235720294  0.666666666666667  0.285714285714286
130 -0.542135455936135  0.625000000000000  0.238095238095238
131 -0.541094638872936  0.571428571428571  0.190476190476190
132 -0.538942849628813  0.500000000000000  0.142857142857143
133 -0.487024081449297  0.600000000000000  0.142857142857143
134 -0.430367879352205  0.750000000000000  0.142857142857143
135 -0.409482448826507  1.000000000000000  0.142857142857143
136 -0.217020726794531  1.000000000000000  0.095238095238095
137  0.207600186321636  1.000000000000000  0.047619047619048
138  1.000000000000000  1.000000000000000  0.000000000000000

                   Que
0    0.594017094017094
1    0.589743589743590
2    0.585470085470085
3    0.581196581196581
4    0.576923076923077
5    0.572649572649573
6    0.568376068376068
7    0.564102564102564
8    0.559829059829060
9    0.555555555555556
10   0.551282051282051
11   0.547008547008547
12   0.542735042735043
13   0.538461538461538
```

```
14    0.534188034188034
15    0.529914529914530
16    0.525641025641026
17    0.521367521367521
18    0.517094017094017
19    0.512820512820513
20    0.508547008547009
21    0.504273504273504
22    0.500000000000000
23    0.495726495726496
24    0.491452991452991
25    0.487179487179487
26    0.482905982905983
27    0.478632478632479
28    0.474358974358974
29    0.470085470085470
..                  ...
109   0.128205128205128
110   0.123931623931624
111   0.119658119658120
112   0.115384615384615
113   0.111111111111111
114   0.106837606837607
115   0.102564102564103
116   0.098290598290598
117   0.094017094017094
118   0.089743589743590
119   0.085470085470085
120   0.081196581196581
121   0.076923076923077
122   0.072649572649573
123   0.068376068376068
124   0.064102564102564
125   0.059829059829060
126   0.055555555555556
127   0.047008547008547
128   0.042735042735043
129   0.038461538461538
130   0.034188034188034
131   0.029914529914530
132   0.025641025641026
133   0.021367521367521
134   0.017094017094017
135   0.012820512820513
136   0.008547008547009
137   0.004273504273504
138   0.000000000000000

[139 rows x 4 columns]
```





3.5 SVC Linear

In [96]:

```
model = 'SVC Linear B'
clf_svlb = LinearSVC(random_state=123)
gs_params = {'C': [.01 ,.1 ,1.0]}
gs_score = 'roc_auc'
features_orig = X_train.columns

clf_svlb, pred_svlb = bin_classify(model, clf_svlb, features_orig, params=gs_params, score=gs_score
)
print('\nBest Parameters:\n',clf_svlb)

metrics_svlb, roc_svlb, prc_svlb = bin_metricsbin_metricsbin_class_metrics(model, y_test, pred_svlb
.y_pred, pred_svlb.y_score, print_out=True, plot_out=True)
```

```
Best Parameters:
 LinearSVC(C=0.01, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=123, tol=0.0001,
          verbose=0)
-----------------------------------------------------------
SVC Linear B

Confusion Matrix:
[[211   2]
 [ 15   6]]

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.99      0.96       213
           1       0.75      0.29      0.41        21

    accuracy                           0.93       234
   macro avg       0.84      0.64      0.69       234
weighted avg       0.92      0.93      0.91       234


Metrics:
                SVC Linear B
Accuracy    0.927350427350427
Precision   0.750000000000000
Recall      0.285714285714286
F1 Score    0.413793103448276
ROC AUC     0.769953051643192

ROC Thresholds:

            Threshold  TP  FP  TN  FN               TPR               FPR  \
0    2.233379175281394   0  25  75   0  0.000000000000000  0.000000000000000
1    1.233379175281394   0  25  74   0  0.000000000000000  0.004694835680751
2    0.610478118974451   2  22  74   0  0.095238095238095  0.004694835680751
3    0.250303840996567   2  22  74   0  0.095238095238095  0.009389671361502
4    0.024984692422877   7  17  74   0  0.285714285714286  0.009389671361502
5   -0.052830812988921   7  17  73   1  0.285714285714286  0.018779342723005
6   -0.072521900957506   9  15  73   1  0.380952380952381  0.018779342723005
7   -0.249321943268304   9  15  72   2  0.380952380952381  0.028169014084507
8   -0.266769328376530  11  13  72   2  0.476190476190476  0.028169014084507
9   -0.306363586125689  11  13  72   2  0.476190476190476  0.037558685446009
10  -0.307907028001210  13  11  72   2  0.523809523809524  0.037558685446009
11  -0.486785863590762  13  11  69   5  0.523809523809524  0.075117370892019
12  -0.491301890999518  14  10  69   5  0.571428571428571  0.075117370892019
13  -0.545050492973962  14  10  68   6  0.571428571428571  0.089201877934272
```

14	-0.565057018636403	15	9	68	6	0.619047619047619	0.089201877934272
15	-0.860355939665443	15	9	54	20	0.619047619047619	0.272300469483568
16	-0.865864519040516	16	8	54	20	0.666666666666667	0.272300469483568
17	-0.877695057681444	16	8	53	21	0.666666666666667	0.281690140845070
18	-0.886836001633786	17	7	53	21	0.714285714285714	0.281690140845070
19	-0.920627830339723	17	7	42	32	0.714285714285714	0.427230046948357
20	-0.920927923684569	19	5	42	32	0.761904761904762	0.427230046948357
21	-0.936894930198681	19	5	31	43	0.761904761904762	0.577464788732394
22	-0.937712307249889	22	2	31	43	0.904761904761905	0.577464788732394
23	-0.971855007150347	22	2	10	64	0.904761904761905	0.859154929577465
24	-0.974006469995448	23	1	10	64	0.952380952380952	0.859154929577465
25	-0.989100589040038	23	1	6	68	0.952380952380952	0.915492957746479
26	-0.990259648636021	25	0	6	68	1.000000000000000	0.915492957746479
27	-1.257821958337774	25	0	0	75	1.000000000000000	1.000000000000000

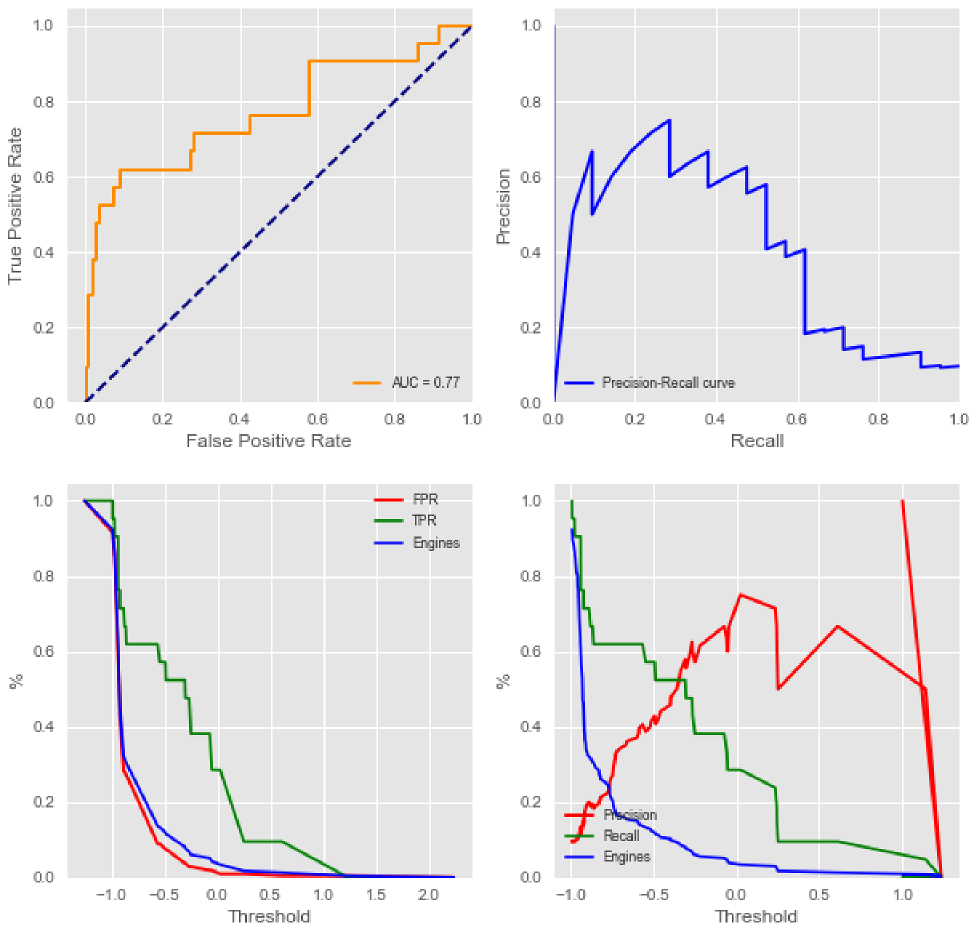
	TNR	FNR	Que
0	1.000000000000000	0.750000000000000	0.000000000000000
1	1.000000000000000	0.747474747474748	0.004273504273504
2	1.000000000000000	0.770833333333333	0.012820512820513
3	1.000000000000000	0.770833333333333	0.017094017094017
4	1.000000000000000	0.813186813186813	0.034188034188034
5	0.986486486486487	0.811111111111111	0.042735042735043
6	0.986486486486487	0.829545454545455	0.051282051282051
7	0.972972972972973	0.827586206896552	0.059829059829060
8	0.972972972972973	0.847058823529412	0.068376068376068
9	0.972972972972973	0.847058823529412	0.076923076923077
10	0.972972972972973	0.867469879518072	0.081196581196581
11	0.932432432432432	0.862500000000000	0.115384615384615
12	0.932432432432432	0.873417721518987	0.119658119658120
13	0.918918918918919	0.871794871794872	0.132478632478632
14	0.918918918918919	0.883116883116883	0.136752136752137
15	0.729729729729730	0.857142857142857	0.303418803418803
16	0.729729729729730	0.870967741935484	0.307692307692308
17	0.716216216216216	0.868852459016393	0.316239316239316
18	0.716216216216216	0.883333333333333	0.320512820512821
19	0.567567567567568	0.857142857142857	0.452991452991453
20	0.567567567567568	0.893617021276596	0.457264957264957
21	0.418918918918919	0.861111111111111	0.594017094017094
22	0.418918918918919	0.939393939393939	0.606837606837607
23	0.135135135135135	0.833333333333333	0.863247863247863
24	0.135135135135135	0.909090909090909	0.867521367521368
25	0.081081081081081	0.857142857142857	0.918803418803419
26	0.081081081081081	1.000000000000000	0.923076923076923
27	0.000000000000000	NaN	1.000000000000000

Precision-Recall Thresholds:

	Threshold	Precision	Recall \
0	-0.990259648636021	0.097222222222222	1.000000000000000
1	-0.989100589040038	0.093023255813953	0.952380952380952
2	-0.988921103475870	0.093457943925234	0.952380952380952
3	-0.987959485857574	0.093896713615023	0.952380952380952
4	-0.987591910194283	0.094339622641509	0.952380952380952
5	-0.987095359278244	0.094786729857820	0.952380952380952
6	-0.986748123738784	0.095238095238095	0.952380952380952
7	-0.984501861938983	0.095693779904306	0.952380952380952
8	-0.983630205664407	0.096153846153846	0.952380952380952
9	-0.979562507273874	0.096618357487923	0.952380952380952
10	-0.977342323017176	0.097087378640777	0.952380952380952
11	-0.975971746694092	0.097560975609756	0.952380952380952
12	-0.975248534230794	0.098039215686275	0.952380952380952
13	-0.974006469995448	0.098522167487685	0.952380952380952
14	-0.971855007150347	0.094059405940594	0.904761904761905
15	-0.971487758535501	0.094527363184080	0.904761904761905
16	-0.971391671685810	0.095000000000000	0.904761904761905
17	-0.970087161786531	0.095477386934673	0.904761904761905
18	-0.968318026975688	0.095959595959596	0.904761904761905
19	-0.968082030265615	0.096446700507614	0.904761904761905
20	-0.967810307389531	0.096938775510204	0.904761904761905
21	-0.966553555727352	0.097435897435897	0.904761904761905
22	-0.965867002389827	0.097938144329897	0.904761904761905
23	-0.965169886761571	0.098445595854922	0.904761904761905
24	-0.964803195368797	0.098958333333333	0.904761904761905
25	-0.964013548144485	0.099476439790576	0.904761904761905
26	-0.962688107085534	0.100000000000000	0.904761904761905
27	-0.960326452219773	0.100529100529101	0.904761904761905
28	-0.958368594538991	0.101063829787234	0.904761904761905

```
29  -0.954988302054025  0.101604278074866  0.904761904761905
..                 ...                ...                ...
187 -0.514003826868127  0.413793103448276  0.571428571428571
188 -0.491301890999518  0.428571428571429  0.571428571428571
189 -0.486785863590762  0.407407407407407  0.523809523809524
190 -0.466470365180505  0.423076923076923  0.523809523809524
191 -0.457567148750824  0.440000000000000  0.523809523809524
192 -0.397421771378681  0.458333333333333  0.523809523809524
193 -0.395410263057828  0.478260869565217  0.523809523809524
194 -0.359226453725414  0.500000000000000  0.523809523809524
195 -0.345451500606185  0.523809523809524  0.523809523809524
196 -0.335178311912623  0.550000000000000  0.523809523809524
197 -0.307907028001210  0.578947368421053  0.523809523809524
198 -0.303638586125689  0.555555555555556  0.476190476190476
199 -0.282927343657583  0.588235294117647  0.476190476190476
200 -0.266769328376530  0.625000000000000  0.476190476190476
201 -0.263130600404759  0.600000000000000  0.428571428571429
202 -0.249321943268304  0.571428571428571  0.380952380952381
203 -0.218077166805714  0.615384615384615  0.380952380952381
204 -0.072521900957506  0.666666666666667  0.380952380952381
205 -0.057016176948133  0.636363636363636  0.333333333333333
206 -0.052830812988921  0.600000000000000  0.285714285714286
207 -0.044740088749604  0.666666666666667  0.285714285714286
208  0.024984692422877  0.750000000000000  0.285714285714286
209  0.233239536019213  0.714285714285714  0.238095238095238
210  0.242547106497457  0.666666666666667  0.190476190476190
211  0.244637947814359  0.600000000000000  0.142857142857143
212  0.250303840996567  0.500000000000000  0.095238095238095
213  0.610478118974451  0.666666666666667  0.095238095238095
214  1.139400123679144  0.500000000000000  0.047619047619048
215  1.233379175281394  0.000000000000000  0.000000000000000
216  1.000000000000000  1.000000000000000  0.000000000000000

                   Que
0    0.923076923076923
1    0.918803418803419
2    0.914529914529915
3    0.910256410256410
4    0.905982905982906
5    0.901709401709402
6    0.897435897435897
7    0.893162393162393
8    0.888888888888889
9    0.884615384615385
10   0.880341880341880
11   0.876068376068376
12   0.871794871794872
13   0.867521367521368
14   0.863247863247863
15   0.858974358974359
16   0.854700854700855
17   0.850427350427350
18   0.846153846153846
19   0.841880341880342
20   0.837606837606838
21   0.833333333333333
22   0.829059829059829
23   0.824786324786325
24   0.820512820512820
25   0.816239316239316
26   0.811965811965812
27   0.807692307692308
28   0.803418803418803
29   0.799145299145299
..                 ...
187  0.123931623931624
188  0.119658119658120
189  0.115384615384615
190  0.111111111111111
191  0.106837606837607
192  0.102564102564103
193  0.098290598290598
194  0.094017094017094
195  0.089743589743590
196  0.085470085470085
197  0.081196581196581
198  0.076923076923077
```

```
199   0.072649572649573
200   0.068376068376068
201   0.064102564102564
202   0.059829059829060
203   0.055555555555556
204   0.051282051282051
205   0.047008547008547
206   0.042735042735043
207   0.038461538461538
208   0.034188034188034
209   0.029914529914530
210   0.025641025641026
211   0.021367521367521
212   0.017094017094017
213   0.012820512820513
214   0.008547008547009
215   0.004273504273504
216   0.008547008547009

[217 rows x 4 columns]
```



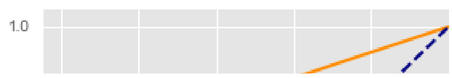
3.6 KNN

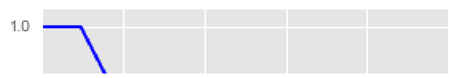
In [97]:

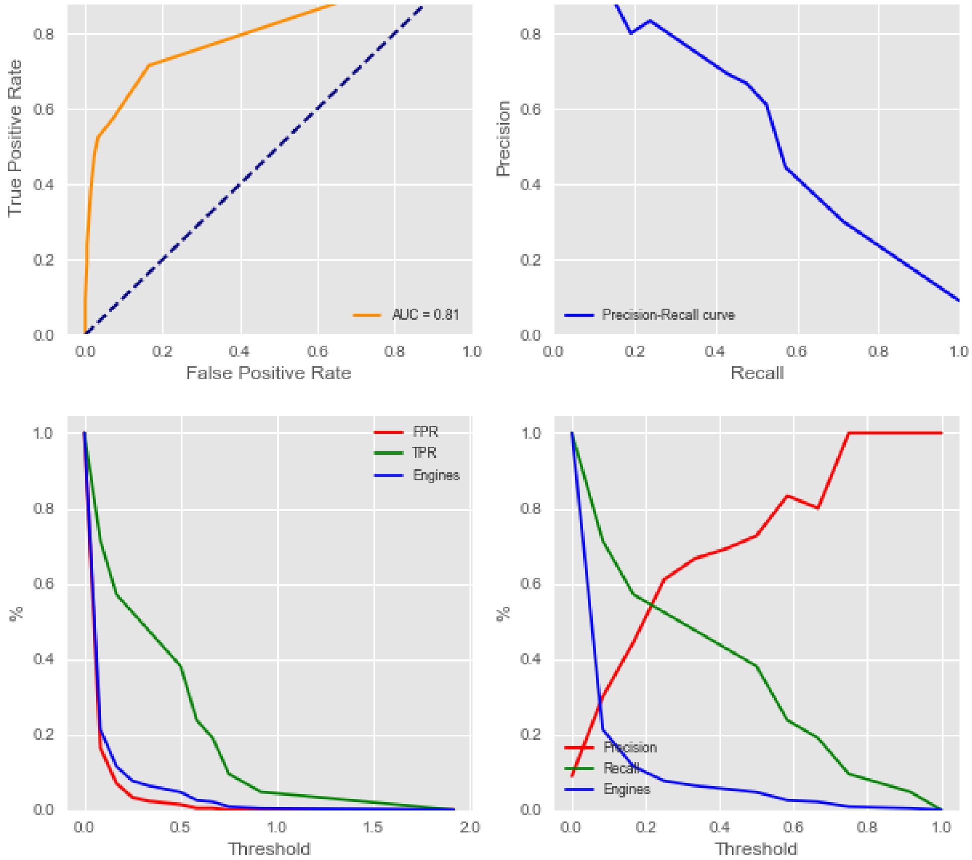
```
model = 'KNN B'
clf_knnb = KNeighborsClassifier(n_jobs=-1)
gs_params = {'n_neighbors': [9, 10, 11, 12, 13]}
gs_score = 'roc_auc'
features_orig = X_train.columns

clf_knnb, pred_knnb = bin_classify(model, clf_knnb, features_orig, params=gs_params, score=gs_score
)
print('\nBest Parameters:\n',clf_knnb)

metrics_knnb, roc_knnb, prc_knnb = bin_metricsbin_metricsbin_class_metrics(model, y_test, pred_knnb
.y_pred, pred_knnb.y_score, print_out=True, plot_out=True)
```

```
Best Parameters:
```

```
Best Parameters:
 KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                      metric_params=None, n_jobs=-1, n_neighbors=12, p=2,
                      weights='uniform')
-----------------------------------------------------------
KNN B

Confusion Matrix:
[[212   1]
 [ 16   5]]

Classification Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96       213
           1       0.83      0.24      0.37        21

    accuracy                           0.93       234
   macro avg       0.88      0.62      0.67       234
weighted avg       0.92      0.93      0.91       234


Metrics:
                      KNN B
Accuracy   0.927350427350427
Precision  0.833333333333333
Recall     0.238095238095238
F1 Score   0.370370370370370
ROC AUC    0.809523809523810

ROC Thresholds:

            Threshold  TP  FP  TN  FN               TPR               FPR  \
0   1.916666666666667   0  25  75   0 0.000000000000000 0.000000000000000
1   0.916666666666667   1  23  75   0 0.047619047619048 0.000000000000000
2   0.750000000000000   2  22  75   0 0.095238095238095 0.000000000000000
3   0.666666666666667   4  20  74   0 0.190476190476190 0.004694835680751
4   0.583333333333333   5  19  74   0 0.238095238095238 0.004694835680751
5   0.500000000000000   9  15  73   1 0.380952380952381 0.014084507042254
6   0.333333333333333  11  13  73   1 0.476190476190476 0.023474178403756
7   0.250000000000000  13  11  72   2 0.523809523809524 0.032863849765258
8   0.166666666666667  14  10  69   5 0.571428571428571 0.070422535211268
9   0.083333333333333  17   7  62  12 0.714285714285714 0.164319248826291
10  0.000000000000000  25   0   0  75 1.000000000000000 1.000000000000000

                  TNR               FNR               Que
0   1.000000000000000 0.750000000000000 0.000000000000000
1   1.000000000000000 0.765306122448980 0.004273504273504
2   1.000000000000000 0.773195876288660 0.008547008547009
3   1.000000000000000 0.787234042553192 0.021367521367521
4   1.000000000000000 0.795698924731183 0.025641025641026
5   0.986486486486487 0.829545454545455 0.047008547008547
6   0.986486486486487 0.848837209302326 0.064102564102564
7   0.972972972972973 0.867469879518072 0.076923076923077
8   0.932432432432432 0.873417721518987 0.115384615384615
9   0.837837837837838 0.898550724637681 0.213675213675214
10  0.000000000000000               NaN 1.000000000000000

Precision-Recall Thresholds:

            Threshold         Precision            Recall               Que
0   0.000000000000000 0.089743589743590 1.000000000000000 1.000000000000000
1   0.083333333333333 0.300000000000000 0.714285714285714 0.213675213675214
2   0.166666666666667 0.444444444444444 0.571428571428571 0.115384615384615
3   0.250000000000000 0.611111111111111 0.523809523809524 0.076923076923077
4   0.333333333333333 0.666666666666667 0.476190476190476 0.064102564102564
5   0.416666666666667 0.692307692307692 0.428571428571429 0.055555555555556
6   0.500000000000000 0.727272727272727 0.380952380952381 0.047008547008547
7   0.583333333333333 0.833333333333333 0.238095238095238 0.025641025641026
8   0.666666666666667 0.800000000000000 0.190476190476190 0.021367521367521
9   0.750000000000000 1.000000000000000 0.095238095238095 0.008547008547009
10  0.916666666666667 1.000000000000000 0.047619047619048 0.004273504273504
11  1.000000000000000 1.000000000000000 0.000000000000000 0.000000000000000
```







3.7 Gaussian NB

In [98]:

```
model = 'Gaussian NB B'
clf_gnbb = GaussianNB()
gs_params = {}
gs_score = 'roc_auc'
features_orig = X_train.columns

clf_gnbb, pred_gnbb = bin_classify(model, clf_gnbb, features_orig, params=gs_params, score=gs_score
)
print('\nBest Parameters:\n',clf_gnbb)

metrics_gnbb, roc_gnbb, prc_gnbb = bin_metricsbin_metricsbin_metricsbin_class_metrics(model,
y_test, pred_gnbb.y_pred, pred_gnbb.y_score, print_out=True, plot_out=True)
```

```
Best Parameters:
 GaussianNB(priors=None, var_smoothing=1e-09)
-----------------------------------------------------------
Gaussian NB B

Confusion Matrix:
[[195  18]
 [  8  13]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.92      0.94       213
           1       0.42      0.62      0.50        21

    accuracy                           0.89       234
   macro avg       0.69      0.77      0.72       234
weighted avg       0.91      0.89      0.90       234


Metrics:
              Gaussian NB B
Accuracy   0.888888888888889
Precision  0.419354838709677
Recall     0.619047619047619
```

Recall 0.619047619047619
F1 Score 0.500000000000000
ROC AUC 0.780907668231612

ROC Thresholds:

	Threshold	TP	FP	TN	FN	TPR	FPR \
0	2.000000000000000	0	25	75	0	0.000000000000000	0.000000000000000
1	1.000000000000000	2	22	74	0	0.095238095238095	0.009389671361502
2	0.999999886310323	7	17	74	0	0.285714285714286	0.009389671361502
3	0.999999809801445	7	17	73	1	0.285714285714286	0.014084507042254
4	0.999678994415328	9	15	73	1	0.380952380952381	0.014084507042254
5	0.997447209111883	9	15	72	2	0.380952380952381	0.037558685446009
6	0.996537257240349	10	14	72	2	0.428571428571429	0.037558685446009
7	0.994374217388783	10	14	71	3	0.428571428571429	0.042253521126761
8	0.993552308852015	11	13	71	3	0.476190476190476	0.042253521126761
9	0.982999915277094	11	13	71	3	0.476190476190476	0.046948356807512
10	0.959657561022827	13	11	71	3	0.523809523809524	0.046948356807512
11	0.837673830530873	13	11	69	5	0.523809523809524	0.070422535211268
12	0.741428580712281	14	10	69	5	0.571428571428571	0.070422535211268
13	0.628715078523703	14	10	68	6	0.571428571428571	0.084507042253521
14	0.604470982062120	15	9	68	6	0.619047619047619	0.084507042253521
15	0.006022752288135	15	9	55	19	0.619047619047619	0.262910798122066
16	0.005363260857376	16	8	55	19	0.666666666666667	0.262910798122066
17	0.003993822792110	16	8	54	20	0.666666666666667	0.267605633802817
18	0.003904651628226	17	7	54	20	0.714285714285714	0.267605633802817
19	0.002969328042199	17	7	49	25	0.714285714285714	0.338028169014085
20	0.002962093566395	19	5	49	25	0.761904761904762	0.338028169014085
21	0.002446669437712	19	5	36	38	0.761904761904762	0.516431924882629
22	0.002417909882507	20	4	36	38	0.809523809523810	0.516431924882629
23	0.002010300801961	20	4	34	40	0.809523809523810	0.539906103286385
24	0.002008832352734	21	3	34	40	0.857142857142857	0.539906103286385
25	0.001590129770694	21	3	26	48	0.857142857142857	0.647887323943662
26	0.001568114493252	22	2	26	48	0.904761904761905	0.647887323943662
27	0.001282489048514	22	2	15	59	0.904761904761905	0.793427230046948
28	0.001275029152261	23	1	15	59	0.952380952380952	0.793427230046948
29	0.001055357431536	23	1	9	65	0.952380952380952	0.877934272300469
30	0.001033987166330	25	0	9	65	1.000000000000000	0.877934272300469
31	0.000023841352473	25	0	0	75	1.000000000000000	1.000000000000000

	TNR	FNR	Que
0	1.000000000000000	0.750000000000000	0.000000000000000
1	1.000000000000000	0.770833333333333	0.017094017094017
2	1.000000000000000	0.813186813186813	0.034188034188034
3	0.986486486486487	0.811111111111111	0.038461538461538
4	0.986486486486487	0.829545454545455	0.047008547008547
5	0.972972972972973	0.827586206896552	0.068376068376068
6	0.972972972972973	0.837209302325581	0.072649572649573
7	0.959459459459459	0.835294117647059	0.076923076923077
8	0.959459459459459	0.845238095238095	0.081196581196581
9	0.959459459459459	0.845238095238095	0.085470085470085
10	0.959459459459459	0.865853658536585	0.089743589743590
11	0.932432432432432	0.862500000000000	0.111111111111111
12	0.932432432432432	0.873417721518987	0.115384615384615
13	0.918918918918919	0.871794871794872	0.128205128205128
14	0.918918918918919	0.883116883116883	0.132478632478632
15	0.743243243243243	0.859375000000000	0.294871794871795
16	0.743243243243243	0.873015873015873	0.299145299145299
17	0.729729729729730	0.870967741935484	0.303418803418803
18	0.729729729729730	0.885245901639344	0.307692307692308
19	0.662162162162162	0.875000000000000	0.371794871794872
20	0.662162162162162	0.907407407407407	0.376068376068376
21	0.486486486486487	0.878048780487805	0.538461538461538
22	0.486486486486487	0.900000000000000	0.542735042735043
23	0.459459459459459	0.894736842105263	0.564102564102564
24	0.459459459459459	0.918918918918919	0.568376068376068
25	0.351351351351351	0.896551724137931	0.666666666666667
26	0.351351351351351	0.928571428571429	0.670940170940171
27	0.202702702702703	0.882352941176471	0.803418803418803
28	0.202702702702703	0.937500000000000	0.807692307692308
29	0.121621621621622	0.900000000000000	0.884615384615385
30	0.121621621621622	1.000000000000000	0.888888888888889
31	0.000000000000000	NaN	1.000000000000000

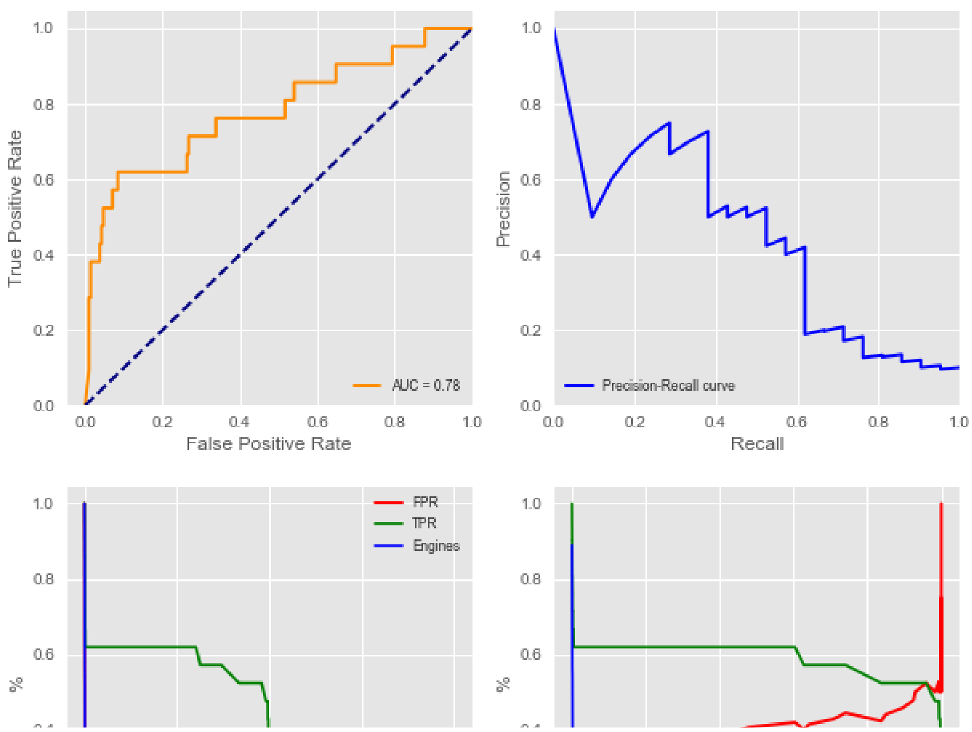
Precision-Recall Thresholds:

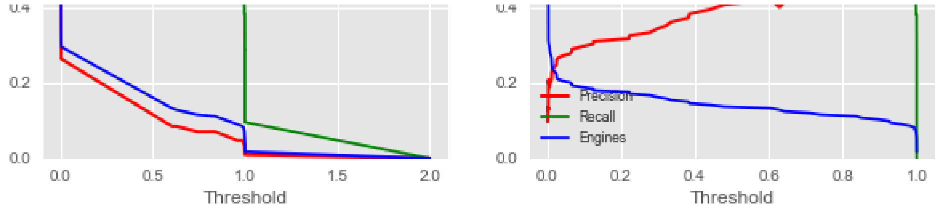
	Threshold	Precision	Recall \
0	0.001033987166330	0.100961538461538	1.000000000000000

0	0.001033987166330	0.100961538461538	1.000000000000000
1	0.001055357431536	0.096618357487923	0.952380952380952
2	0.001055730642298	0.097087378640777	0.952380952380952
3	0.001065142193021	0.097560975609756	0.952380952380952
4	0.001082856643380	0.098039215686275	0.952380952380952
5	0.001086467498413	0.098522167487685	0.952380952380952
6	0.001091587244596	0.099009900990099	0.952380952380952
7	0.001107140721764	0.099502487562189	0.952380952380952
8	0.001123611707384	0.100000000000000	0.952380952380952
9	0.001147683777625	0.100502512562814	0.952380952380952
10	0.001155366330886	0.101010101010101	0.952380952380952
11	0.001166111695849	0.101522842639594	0.952380952380952
12	0.001220708864870	0.102040816326531	0.952380952380952
13	0.001228580833628	0.102564102564103	0.952380952380952
14	0.001236505532478	0.103092783505155	0.952380952380952
15	0.001241487968030	0.103626943005181	0.952380952380952
16	0.001242073366907	0.104166666666667	0.952380952380952
17	0.001248359910852	0.104712041884817	0.952380952380952
18	0.001273099897900	0.105263157894737	0.952380952380952
19	0.001275029152261	0.105820105820106	0.952380952380952
20	0.001282489048514	0.101063829787234	0.904761904761905
21	0.001310091328871	0.101604278074866	0.904761904761905
22	0.001312560489977	0.102150537634409	0.904761904761905
23	0.001319326222968	0.102702702702703	0.904761904761905
24	0.001319859076936	0.103260869565217	0.904761904761905
25	0.001335931394734	0.103825136612022	0.904761904761905
26	0.001339857278701	0.104395604395604	0.904761904761905
27	0.001344044291431	0.104972375690608	0.904761904761905
28	0.001348382531864	0.105555555555556	0.904761904761905
29	0.001380046540392	0.106145251396648	0.904761904761905
..	...	...	...
176	0.479331261733086	0.406250000000000	0.619047619047619
177	0.604470982062120	0.419354838709677	0.619047619047619
178	0.628715078523703	0.400000000000000	0.571428571428571
179	0.641659549527194	0.413793103448276	0.571428571428571
180	0.710094262708565	0.428571428571429	0.571428571428571
181	0.741428580712281	0.444444444444444	0.571428571428571
182	0.837673830530873	0.423076923076923	0.523809523809524
183	0.849264043167909	0.440000000000000	0.523809523809524
184	0.896950388015558	0.458333333333333	0.523809523809524
185	0.925140643450318	0.478260869565217	0.523809523809524
186	0.929684360238032	0.500000000000000	0.523809523809524
187	0.959657561022827	0.523809523809524	0.523809523809524
188	0.982999915277094	0.500000000000000	0.476190476190476
189	0.993552308852015	0.526315789473684	0.476190476190476
190	0.994374217388783	0.500000000000000	0.428571428571429
191	0.996537257240349	0.529411764705882	0.428571428571429
192	0.997447209111883	0.500000000000000	0.380952380952381
193	0.998042519232706	0.533333333333333	0.380952380952381
194	0.999033841407307	0.571428571428571	0.380952380952381
195	0.999510676153694	0.615384615384615	0.380952380952381
196	0.999592759317198	0.666666666666667	0.380952380952381
197	0.999678994415328	0.727272727272727	0.380952380952381
198	0.999999248239378	0.700000000000000	0.333333333333333
199	0.999999809801445	0.666666666666667	0.285714285714286
200	0.999999886310323	0.750000000000000	0.285714285714286
201	0.999999994319236	0.714285714285714	0.238095238095238
202	0.999999999993502	0.666666666666667	0.190476190476190
203	0.999999999999968	0.600000000000000	0.142857142857143
204	1.000000000000000	0.500000000000000	0.095238095238095
205	1.000000000000000	1.000000000000000	0.000000000000000

	Que
0	0.888888888888889
1	0.884615384615385
2	0.880341880341880
3	0.876068376068376
4	0.871794871794872
5	0.867521367521368
6	0.863247863247863
7	0.858974358974359
8	0.854700854700855
9	0.850427350427350
10	0.846153846153846
11	0.841880341880342
12	0.837606837606838
13	0.833333333333333
14	0.829059829059829

```
14    0.829059829059829
15    0.824786324786325
16    0.820512820512820
17    0.816239316239316
18    0.811965811965812
19    0.807692307692308
20    0.803418803418803
21    0.799145299145299
22    0.794871794871795
23    0.790598290598291
24    0.786324786324786
25    0.782051282051282
26    0.777777777777778
27    0.773504273504274
28    0.769230769230769
29    0.764957264957265
..                  ...
176   0.136752136752137
177   0.132478632478632
178   0.128205128205128
179   0.123931623931624
180   0.119658119658120
181   0.115384615384615
182   0.111111111111111
183   0.106837606837607
184   0.102564102564103
185   0.098290598290598
186   0.094017094017094
187   0.089743589743590
188   0.085470085470085
189   0.081196581196581
190   0.076923076923077
191   0.072649572649573
192   0.068376068376068
193   0.064102564102564
194   0.059829059829060
195   0.055555555555556
196   0.051282051282051
197   0.047008547008547
198   0.042735042735043
199   0.038461538461538
200   0.034188034188034
201   0.029914529914530
202   0.025641025641026
203   0.021367521367521
204   0.017094017094017
205   0.017094017094017

[206 rows x 4 columns]
```





Compare all binary classification algorithms

In [100]:

```
#compare all models
metrics_bn = pd.concat([metrics_lgrb, metrics_dtrb, metrics_rfcb, metrics_svcb, metrics_svlb,
metrics_knnb, metrics_gnbb], axis=1)
metrics_bn
```

Out[100]:

	Logistic Regression B	Decision Tree B	Random Forest B	SVC B	SVC Linear B	
Accuracy	0.931623931623932	0.927350427350427	0.940170940170940	0.914529914529915	0.927350427350427	0.92735042
Precision	0.777777777777778	0.642857142857143	0.888888888888889	1.000000000000000	0.750000000000000	0.83333333
Recall	0.333333333333333	0.428571428571429	0.380952380952381	0.047619047619048	0.285714285714286	0.23809523
F1 Score	0.466666666666667	0.514285714285714	0.533333333333333	0.090909090909091	0.413793103448276	0.37037037
ROC AUC	0.776883523362397	0.832550860719875	0.917057902973396	0.931589537223340	0.769953051643192	0.80952380

Compare the AUC ROC and Precision-Recall curves

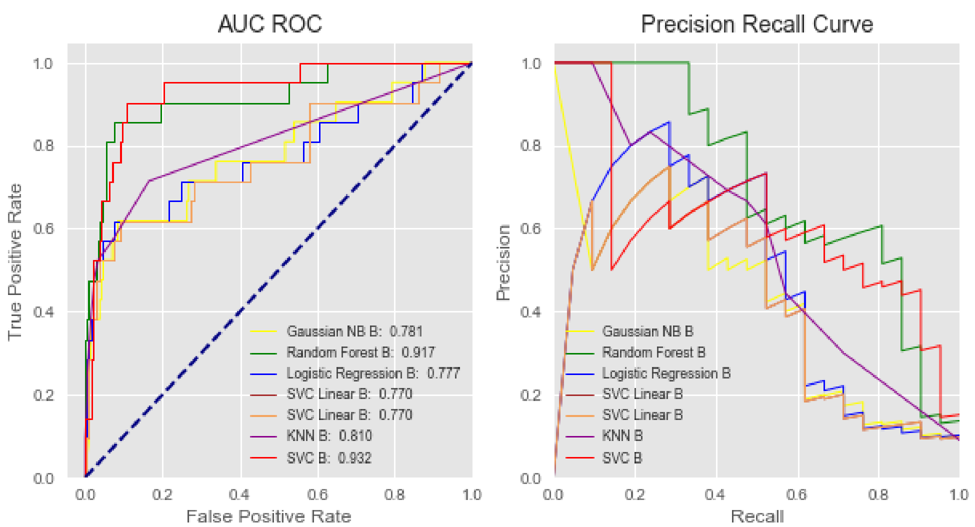
In [105]:

```
# Plot AUC-ROC and precision-recall curves for best models
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, sharex=False, sharey=False )
fig.set_size_inches(10,5)

metrics_rfca = metrics_rfcb
metrics_lgra = metrics_lgrb
metrics_svla = metrics_svlb
metrics_knna = metrics_knnb
metrics_svca = metrics_svcb

prc_rfca = prc_rfcb
prc_lgra = prc_lgrb
prc_svla = prc_svlb
prc_knna = prc_knnb
prc_svca = prc_svcb

ax1.plot(roc_gnbb.FPR, roc_gnbb.TPR, color='yellow', lw=1, label= metrics_gnbb.columns.values.tolis
t()[0] + ':  %.3f' % metrics_gnbb.at['ROC AUC', metrics_gnbb.columns.values.tolist()[0]])
ax1.plot(roc_rfca.FPR, roc_rfca.TPR, color='green', lw=1, label= metrics_rfca.columns.values.tolist
()[0] + ':  %.3f' % metrics_rfca.at['ROC AUC', metrics_rfca.columns.values.tolist()[0]])
ax1.plot(roc_lgra.FPR, roc_lgra.TPR, color='blue', lw=1, label= metrics_lgra.columns.values.tolist(
)[0] + ':  %.3f' % metrics_lgra.at['ROC AUC', metrics_lgra.columns.values.tolist()[0]])
ax1.plot(roc_svla.FPR, roc_svla.TPR, color='brown', lw=1, label= metrics_svla.columns.values.tolist
()[0] + ':  %.3f' % metrics_svla.at['ROC AUC', metrics_svla.columns.values.tolist()[0]])
ax1.plot(roc_svlb.FPR, roc_svlb.TPR, color='sandybrown', lw=1, label= metrics_svlb.columns.values.t
olist()[0] + ':  %.3f' % metrics_svlb.at['ROC AUC', metrics_svlb.columns.values.tolist()[0]])
ax1.plot(roc_knna.FPR, roc_knna.TPR, color='darkmagenta', lw=1, label= metrics_knna.columns.values.
tolist()[0] + ':  %.3f' % metrics_knna.at['ROC AUC', metrics_knna.columns.values.tolist()[0]])
ax1.plot(roc_svca.FPR, roc_svca.TPR, color='red', lw=1, label= metrics_svca.columns.values.tolist()
[0] + ':  %.3f' % metrics_svca.at['ROC AUC', metrics_svca.columns.values.tolist()[0]])
ax1.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
ax1.set_xlim([-0.05, 1.0])
ax1.set_ylim([0.0, 1.05])
```

```
ax1.set_xlabel('False Positive Rate')
ax1.set_ylabel('True Positive Rate')
ax1.legend(loc="lower right", fontsize='small')
ax1.set_title('AUC ROC')

ax2.plot(prc_gnbb.Recall, prc_gnbb.Precision, color='yellow', lw=1, label= metrics_gnbb.columns.val
ues.tolist()[0])
ax2.plot(prc_rfca.Recall, prc_rfca.Precision, color='green', lw=1, label= metrics_rfca.columns.valu
es.tolist()[0])
ax2.plot(prc_lgra.Recall, prc_lgra.Precision, color='blue', lw=1, label= metrics_lgra.columns.value
s.tolist()[0])
ax2.plot(prc_svla.Recall, prc_svla.Precision, color='brown', lw=1, label= metrics_svla.columns.valu
es.tolist()[0])
ax2.plot(prc_svlb.Recall, prc_svlb.Precision, color='sandybrown', lw=1, label= metrics_svlb.columns
.values.tolist()[0])
ax2.plot(prc_knna.Recall, prc_knna.Precision, color='darkmagenta', lw=1, label= metrics_knna.column
s.values.tolist()[0])
ax2.plot(prc_svca.Recall, prc_svca.Precision, color='red', lw=1, label= metrics_svca.columns.values
.tolist()[0])
ax2.set_xlim([0.0, 1.0])
ax2.set_ylim([0.0, 1.05])
ax2.set_xlabel('Recall')
ax2.set_ylabel('Precision')
ax2.legend(loc="lower left", fontsize='small')
ax2.set_title('Precision Recall Curve')
```

Out[105]:

```
Text(0.5,1,'Precision Recall Curve')
```



Binary Classification Summary:

- Most of the binary classifiers showed good performance on accuracy, but poor on recall. It is an imbalanced classification problem, it is important to balance the training set and have more addition new features.
- Random Forest showed a good and stable performance than other classifiers.
- The graphs for TPR, FPR should be linked to cost matrix of (TP, FP, TN, FN) to calculate the expected value at different operating points (thresholds) to help optimizing business decisions.
- Need to try and compare different samplings(Oversampling/Undersampling/adaptive) and classification models.
- Need to pay attention to the rows labeled '1' as we are more interested in the approach that yields better prediction of 1's.

Appendix:

Oversampling & Undersampling

1. Classify on Oversampling(SMOTE)

1.1 LinearSVC + SMOTE

In [26]:

```
from sklearn.svm import LinearSVC
from imblearn import over_sampling as os
from imblearn import pipeline as pl
from imblearn.metrics import geometric_mean_score, make_index_balanced_accuracy,
classification_report_imbalanced
# utility libraries
import warnings; warnings.simplefilter('ignore')
#print(__doc__)
```

In [162]:

```
pipeline = pl.make_pipeline(os.SMOTE(random_state=RANDOM_STATE),
LinearSVC(random_state=RANDOM_STATE))

# Train with balancing
pipeline.fit(X_train, y_train)

# Test and get the prediction
y_pred_bal = pipeline.predict(X_test)
pipeline
```

Out[162]:

```
Pipeline(memory=None,
         steps=[('smote',
                 SMOTE(k_neighbors=5, kind='deprecated',
                       m_neighbors='deprecated', n_jobs=1,
                       out_step='deprecated', random_state=110, ratio=None,
                       sampling_strategy='auto', svm_estimator='deprecated')),
                ('linearsvc',
                 LinearSVC(C=1.0, class_weight=None, dual=True,
                           fit_intercept=True, intercept_scaling=1,
                           loss='squared_hinge', max_iter=1000,
                           multi_class='ovr', penalty='l2', random_state=110,
                           tol=0.0001, verbose=0))],
         verbose=False)
```

In [30]:

```
y_train.value_counts()
```

Out[30]:

```
0    849
1     85
Name: failure, dtype: int64
```

In [31]:

```
y_test.value_counts()
```

Out[31]:

```
0    213
1     21
Name: failure, dtype: int64
```

In [32]:

```
#the geometric mean of LinearSVC using SMOTE
print('The geometric mean is {}'.format(geometric_mean_score(y_test,y_pred_bal)))
```

```
The geometric mean is 0.7082914139172094
```

In [163]:

```
#alpha = 0.1 and 0.5 give the same IBA result here
```

```
alpha = 0.1
geo_mean = make_index_balanced_accuracy(alpha=alpha, squared=True)(geometric_mean_score)

print('IBA with alpha = {} and the geometric mean: {}'.format(
    alpha, geo_mean(y_test, y_pred_bal)))

alpha = 0.5
geo_mean = make_index_balanced_accuracy(alpha=alpha, squared=True)(geometric_mean_score)

print('IBA with alpha = {} and the geometric mean: {}'.format(alpha, geo_mean(y_test,
y_pred_bal)))
```

```
IBA with alpha = 0.1 and the geometric mean: 0.5016767270288397
IBA with alpha = 0.5 and the geometric mean: 0.5016767270288397
```

In [34]:

```
test_cm = confusion_matrix(y_test, y_pred_bal)
test_cm
```

Out[34]:

```
array([[204,   9],
       [ 10,  11]])
```

In [35]:

```
accuracy_score(y_test, y_pred_bal)
```

Out[35]:

```
0.9188034188034188
```

In [36]:

```
#summary statistics based on oversampling
print(classification_report_imbalanced(y_test, y_pred_bal))
```

```
                   pre       rec       spe        f1       geo       iba       sup

          0       0.95      0.96      0.52      0.96      0.71      0.52       213
          1       0.55      0.52      0.96      0.54      0.71      0.48        21

avg / total       0.92      0.92      0.56      0.92      0.71      0.52       234
```

1.2 RandomForest Classifier + SMOTE

In [37]:

```
from collections import Counter
from imblearn.over_sampling import SMOTE, ADASYN

sm = SMOTE(random_state=RANDOM_STATE)
X_res, y_res = sm.fit_sample(X_train, y_train)

print('Resampled dataset shape {}'.format(Counter(y_res)))
print(sorted(Counter(y_res).items()))
```

```
Resampled dataset shape Counter({0: 849, 1: 849})
[(0, 849), (1, 849)]
```

In [38]:

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=5000, random_state=RANDOM_STATE)
a = rf.fit(X_res, y_res)
```

In [39]:

```
rf.score(X_res, y_res)
```

Out[39]:

```
1.0
```

In [41]:

```
rf_res_pred=rf.predict(X_res)
rf_cm = confusion_matrix(y_res, rf_res_pred)
rf_cm
```

Out[41]:

```
array([[849,   0],
       [  0, 849]])
```

In [42]:

```
rf_cv_score = cross_val_score(a, X_res, y_res, cv=10, scoring='accuracy')
```

In [45]:

```
rf_cv_score
```

Out[45]:

```
array([0.90588235, 0.97647059, 0.97058824, 0.98235294, 0.98235294,
       0.98235294, 0.94705882, 0.98823529, 0.99411765, 1.        ])
```

In [46]:

```
rf_cv_score.mean()
```

Out[46]:

```
0.9729411764705882
```

In [47]:

```
accuracy_score(y_res, rf_res_pred)
```

Out[47]:

```
1.0
```

In [48]:

```
accuracy_score(y_res, rf_res_pred)
```

Out[48]:

```
1.0
```

In [51]:

```
rf_test_pred=rf.predict(X_test)
rf_test_cm = confusion_matrix(y_test, rf_test_pred)
rf_test_cm
```

Out[51]:

```
array([[210,   3],
       [  9,  12]])
```

In [52]:

```
accuracy_score(y_test, rf_test_pred)
```

Out[52]:

```
0.9487179487179487
```

In [53]:

```
from imblearn.metrics import classification_report_imbalanced
print(classification_report_imbalanced(y_test,rf_test_pred))
```

```
                   pre       rec       spe        f1       geo       iba       sup

          0       0.96      0.99      0.57      0.97      0.75      0.59       213
          1       0.80      0.57      0.99      0.67      0.75      0.54        21

avg / total       0.94      0.95      0.61      0.94      0.75      0.58       234
```

1.3 LinearSVC + Adaptive SMOTE

In [55]:

```
X_resampled, y_resampled = ADASYN().fit_sample(X_train, y_train)
print(sorted(Counter(y_resampled).items()))

clf_adasyn = LinearSVC().fit(X_resampled, y_resampled)
```

```
[(0, 849), (1, 852)]
```

In [56]:

```
clf_adasyn.score(X_resampled, y_resampled)
```

Out[56]:

```
0.7530864197530864
```

In [59]:

```
lsvc_res_pred=clf_adasyn.predict(X_resampled)
lsvc_cm = confusion_matrix(y_resampled, lsvc_res_pred)
lsvc_cm
```

Out[59]:

```
array([[796,  53],
       [367, 485]])
```

In [60]:

```
lsvc_cv_score = cross_val_score(clf_adasyn, X_resampled, y_resampled, cv=10, scoring='accuracy')
lsvc_cv_score
```

Out[60]:

```
array([0.67836257, 0.78362573, 0.61764706, 0.70588235, 0.68235294,
       0.6       , 0.66470588, 0.78235294, 0.67647059, 0.69822485])
```

In [61]:

```
lsvc_cv_score.mean()
```

Out[61]:

```
0.6889624920870456
```

In [62]:

```
accuracy_score(y_resampled, lsvc_res_pred)
```

Out[62]:

```
0.7530864197530864
```

In [64]:

```
lsvc_test_pred=clf_adasyn.predict(X_test)
lsvc_test_cm = confusion_matrix(y_test, lsvc_test_pred)
lsvc_test_cm
```

Out[64]:

```
array([[198,  15],
       [  9,  12]])
```

In [65]:

```
print(classification_report_imbalanced(y_test,lsvc_test_pred))
accuracy_score(y_test, lsvc_test_pred)
```

```
                   pre       rec       spe        f1       geo       iba       sup

          0       0.96      0.93      0.57      0.94      0.73      0.55       213
          1       0.44      0.57      0.93      0.50      0.73      0.51        21

avg / total       0.91      0.90      0.60      0.90      0.73      0.55       234
```

Out[65]:

```
0.8974358974358975
```

2. Classify on Undersampling

2.1 RandomForest Classifier + RandomUnderSampler

In [68]:

```
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(random_state=RANDOM_STATE)
X_und, y_und = rus.fit_sample(X_train, y_train)
print('Resampled dataset shape {}'.format(Counter(y_und)))
```

```
Resampled dataset shape Counter({0: 85, 1: 85})
```

In [71]:

```
und_rf = RandomForestClassifier(n_estimators=5000, random_state=RANDOM_STATE)
u = und_rf.fit(X_und, y_und)
und_rf.score(X_und, y_und)
```

Out[71]:

```
1.0
```

In [72]:

```
und_rf_pred=und_rf.predict(X_und)
und_rf_cm = confusion_matrix(y_und, und_rf_pred)
und_rf_cm
```

Out[72]:

```
array([[85,  0],
       [ 0, 85]])
```

In [73]:

```
und_rf_cv_score = cross_val_score(u, X_und, y_und, cv=10, scoring='accuracy')
und_rf_cv_score
```

Out[73]:

```
array([0.94444444, 0.94444444, 0.77777778, 0.94444444, 0.88888889,
       0.9375    , 0.9375    , 0.9375    , 1.        , 1.        ])
```

In [74]:

```
und_rf_cv_score.mean()
```

Out[74]:

```
0.93125
```

In [75]:

```
accuracy_score(y_und, und_rf_pred)
```

Out[75]:

```
1.0
```

In [76]:

```
und_rf_test_pred=und_rf.predict(X_test)
und_rf_test_cm = confusion_matrix(y_test, und_rf_test_pred)
und_rf_test_cm
```

Out[76]:

```
array([[177,  36],
       [  2,  19]])
```

In [77]:

```
accuracy_score(y_test, und_rf_test_pred)
```

Out[77]:

```
0.8376068376068376
```

In [78]:

```
print("Undersampling & RandomForest Classifier:")
print(classification_report_imbalanced(y_test,und_rf_test_pred))
```

```
Undersampling & RandomForest Classifier:
                   pre       rec       spe        f1       geo       iba       sup

          0       0.99      0.83      0.90      0.90      0.87      0.75       213
          1       0.35      0.90      0.83      0.50      0.87      0.76        21

avg / total       0.93      0.84      0.90      0.87      0.87      0.75       234
```

In [79]:

```
print("Oversampling & RandomForest Classifier:")
print(classification_report_imbalanced(y_test,rf_test_pred))
```

```
Oversampling & RandomForest Classifier:
```

	pre	rec	spe	f1	geo	iba	sup
0	0.96	0.99	0.57	0.97	0.75	0.59	213
1	0.80	0.57	0.99	0.67	0.75	0.54	21
avg / total	0.94	0.95	0.61	0.94	0.75	0.58	234

Note:

- pre is precision;
- rec is recall, which is the same as sensitivity.
- spe is specificity;
- f1 is the harmonic average of the precision and recall;
- geo is the geometric mean of specificity and sensitivity;
- iba is the index of imbalanced accuracy;

Feature Extraction

In [ ]:

```
# Add average and standard deviation for trainingg.
def add_features(df_in, rolling_win_size):
    attribute_cols = ['attribute6', 'attribute1', 'attribute2', 'attribute4', 'attribute5','attribu
te7']

    attr_av_cols = [nm.replace('attribute', 'av') for nm in attribute_cols]
    attr_sd_cols = [nm.replace('attribute', 'sd') for nm in attribute_cols]

    df_out = pd.DataFrame()
    ws = rolling_win_size

    #calculate rolling stats for each device id
    for m_id in pd.unique(df_in.id):

        # get a subset for each device
        df_engine = df_in[df_in['id'] == m_id]
        df_sub = df_engine[attribute_cols]

        # get rolling mean for the subset
        av = df_sub.rolling(ws, min_periods=1).mean()
        av.columns = attr_av_cols

        # get the rolling standard deviation for the subset
        sd = df_sub.rolling(ws, min_periods=1).std().fillna(0)
        sd.columns = attr_sd_cols

        # combine the two new subset dataframes columns
        new_ftrs = pd.concat([df_engine,av,sd], axis=1)

        # add the new features rows to the output dataframe
        df_out = pd.concat([df_out,new_ftrs])

    return df_out
```

Examine Missing Values

In [16]:

```
# Calculate missing values
def missing_values_table(df):
        # Total missing values
        mis_val = df.isnull().sum()

        # Percentage of missing values
        mis_val_percent = 100 * df.isnull().sum() / len(df)

        # Make a table with the results
        mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
```

```
        # Rename the columns
        mis_val_table_ren_columns = mis_val_table.rename(
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})

        # Sort the table by percentage of missing descending
        mis_val_table_ren_columns = mis_val_table_ren_columns[
            mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)

        # Print some summary information
        print ("Selected dataframe has " + str(df.shape[1]) + " columns.\n"
            "There are " + str(mis_val_table_ren_columns.shape[0]) +
              " columns that have missing values.")

        return mis_val_table_ren_columns
```

In [22]:

```
# Missing values statistics
missing_values = missing_values_table(df_all)
missing_values
```

```
Your selected dataframe has 12 columns.
There are 0 columns that have missing values.
```

Out[22]:

	Missing Values	% of Total Values

In [23]:

```
df_all.dtypes.value_counts()
```

Out[23]:

```
int64     11
object     1
dtype: int64
```

In [25]:

```
# one-hot encoding of categorical variables
df_all = pd.get_dummies(df_all)
#app_test = pd.get_dummies(app_test)

print('Training Features shape: ', df_all.shape)
```

```
Training Features shape:  (124494, 1179)
```