Тестовое задание.

Определить абонентов (msisdn), которые являются одной и той же Персоной (человеком).

In [1]:

```
import pandas as pd
import os
from os import walk
import datetime
pd.set_option('display.max_columns', 500)

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from datetime import datetime, date

import csv

import folium
from folium import plugins
from folium.plugins import HeatMap
import branca
import branca.colormap as cm
```

Загружаем данные и преобразуем в удобный формат некоторые поля

In [2]:

```
current_directory = r'C:\Users\ekaterina.adischeva\Documents\Scripts\BD_task\Техническ
current_directory = current_directory.replace('\\','/')
os.chdir(current_directory)
```

In [3]:

```
df_data = pd.read_csv('02_Data.csv', sep = ';', dtype = {'imei': str, 'lac' : 'str', '
facts = pd.read_excel('01_Факты.xlsx', header = None)
tac_dict = pd.read_csv('03_устройства.csv', quoting=csv.QUOTE_NONE)
event_type = pd.read_excel('04_event_type.xlsx')
```

In [4]:

```
tac_dict.columns = tac_dict.columns.str.replace('\"','')
for column_name in tac_dict.columns:
    tac_dict[column_name] = tac_dict[column_name].str.replace('\"','')
```

Преобразуем в datetime время регистрации события на базовой станции (количество 1/1000 секунды, прошедших с 01.01.1970 до времени регистрации на БС.)

Выделяем tac из поля imei

In [5]:

```
df_data['datetime'] = pd.to_timedelta(df_data['tstamp']/1000, unit='S') + pd.Timestamp
df_data['datetime'] = pd.to_datetime(df_data['datetime'])

df_data['tac'] = df_data['imei'].str.slice(0,8)
```

Подгружаем значения из справочников в данные о перемещении абонентов

In [6]:

```
df_data = df_data.merge(event_type, left_on = ['event_type'], right_on = 'Номер', how =
df_data = df_data.merge(tac_dict, on = ['tac'], how = 'left')
```

Графики и функции

`show_chart` отображает изменение координат широты и долготы по времени

In [7]:

```
def show_chart(fig, data_a, data_b ):
    """
    The function draws 2 chatrs: msisdn lattitude on time and msisdn longitude on time

"""

    data_a_copy = data_a
    data_b_copy = data_b

    plt.subplot(1, 2, 1)
    feature = 'long'
    plt.scatter(data_a_copy['datetime'], data_a_copy[feature], alpha = .5, c = 'blue')
    plt.plot(data_a_copy['datetime'], data_a_copy[feature+'_max'],'-o',alpha = .3, colo
    plt.plot(data_a_copy['datetime'], data_a_copy[feature+'_min'],'-o',alpha = .3, colo
    plt.scatter(data_b_copy['datetime'], data_b_copy[feature], alpha = .5, c = 'red')
    plt.plot(data_b_copy['datetime'], data_b_copy[feature+'_max'],'-o',alpha = .3, colo
    plt.plot(data_b_copy['datetime'], data_b_copy[feature+'_min'],'-o',alpha = .3, colo
    plt.xlim(pd.concat([data_a,data_b])['datetime'].min(), pd.concat([data_a,data_b])[
    plt.xticks(rotation=70)

    plt.subplot(1, 2, 2)
    feature = 'lat'
    plt.scatter(data_a_copy['datetime'], data_a_copy[feature], alpha = .5, c = 'blue')
    plt.plot(data_a_copy['datetime'], data_a_copy[feature+'_max'],'-o',alpha = .3, colo
    plt.plot(data_a_copy['datetime'], data_a_copy[feature+'_min'],'-o',alpha = .3, colo
    plt.scatter(data_b_copy['datetime'], data_b_copy[feature], alpha = .5, c = 'red')
    plt.plot(data_b_copy['datetime'], data_b_copy[feature+'_max'],'-o',alpha = .3, colo
    plt.plot(data_b_copy['datetime'], data_b_copy[feature+'_min'],'-o',alpha = .3, colo
    plt.xlim(pd.concat([data_a,data_b])['datetime'].min(), pd.concat([data_a,data_b])[
    plt.xticks(rotation=70)

    plt.show()
    return fig
```

`show_circles_on_map` отображает точки в которых был абонент на карте

In [8]:

```
def show_circles_on_map(m, data, latitude_column, longitude_column, color):
    """
    The function draws map with circles on it.
    The center of the map is the mean of coordinates passed in data.

    data: DataFrame that contains columns latitude_column and longitude_column
    latitude_column: string, the name of column for latitude coordinates
    longitude_column: string, the name of column for longitude coordinates
    color: string, the color of circles to be drawn
    """

    location = (data[latitude_column].mean(), data[longitude_column].mean())

    for _, row in data.iterrows():
        folium.Circle(
            radius=100,
            location=(row[latitude_column], row[longitude_column]),
            color=color,
            fill_color=color,
            fill=True,
            fill_opacity = .5
        ).add_to(m)

    return m
```

haversine_array определяет расстояние в метрах по координатам

In [9]:

```
def haversine_array(lat1, lng1, lat2, lng2):
    lat1, lng1, lat2, lng2 = map(np.radians, (lat1, lng1, lat2, lng2))
    AVG_EARTH_RADIUS = 6371  # in km
    lat = lat2 - lat1
    lng = lng2 - lng1
    d = np.sin(lat * 0.5) ** 2 + np.cos(lat1) * np.cos(lat2) * np.sin(lng * 0.5) ** 2
    h = 2 * AVG_EARTH_RADIUS * np.arcsin(np.sqrt(d))
    return h
```

Проверяем данные

Смотрим, где в выборке есть пустые данные

In [10]:

```
df_data.count() - df_data.shape[0], df_data.dtypes
```

Out[10]:

```
(lac               0
 cid               0
 msisdn            0
 imei          -11588
 event_type        0
 tstamp            0
 long              0
 lat               0
 max_dist          0
 cell_type         0
 start_angle       0
 end_angle         0
 datetime          0
 tac           -11588
 Номер             0
 Обозначение       0
 Описание          0
 vendor        -23679
 platform      -23679
 type          -23679
 dtype: int64, lac                    object
 cid                     object
 msisdn                   int64
 imei                    object
 event_type               int64
 tstamp                   int64
 long                   float64
 lat                    float64
 max_dist                 int64
 cell_type               object
 start_angle            float64
 end_angle              float64
 datetime       datetime64[ns]
 tac                     object
 Номер                    int64
 Обозначение             object
 Описание                object
 vendor                  object
 platform                object
 type                    object
 dtype: object)
```

Некоторые номера, принадлижайщие Персонам, не попали в выборку с детальными данными

In [11]:

```
# for 2 persons we don't have second msisdn in our dataset
display(facts[~facts[1].isin(df_data['msisdn'])],
        facts[~facts[0].isin(df_data['msisdn'])])
```

	0	1
1	158510912201	158528852857
17	158528850493	158530004641

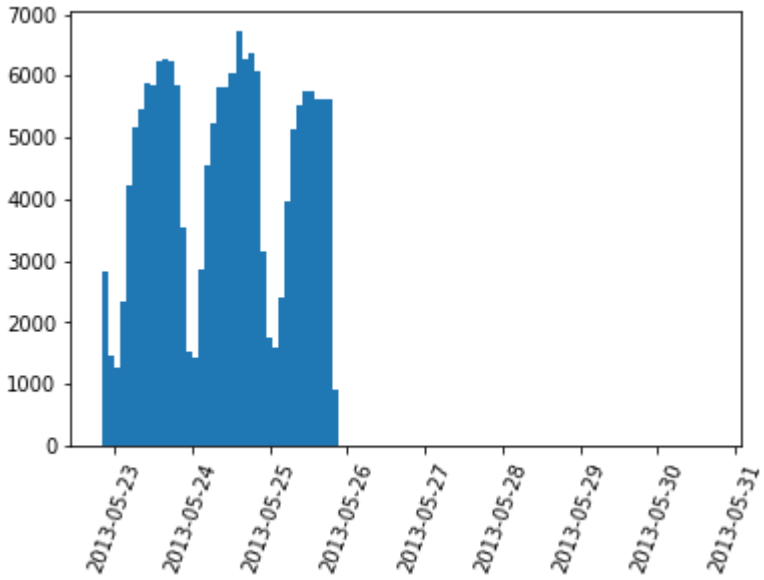
0	1

Смотрим распределение событий по времени. В выборке есть данные, датированные позже 26 мая. Удаляем данные с датой более 26 мая.

In [12]:

```
plt.hist(df_data['datetime'].values, bins=100)
plt.xticks(rotation=70)
plt.show()
```

```
C:\Users\ekaterina.adischeva\AppData\Local\Continuum\anaconda3\lib\site-pack
ages\pandas\plotting\_converter.py:129: FutureWarning: Using an implicitly r
egistered datetime converter for a matplotlib plotting method. The converter
was registered by pandas on import. Future versions of pandas will require y
ou to explicitly register matplotlib converters.

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```



In [13]:

```
df_data.loc[:, 'date'] = df_data['datetime'].dt.date
print(df_data['date'].value_counts())
df_data = df_data[(df_data['datetime']  < datetime.strptime('2013-05-26', '%Y-%m-%d'))
         #         & (df_data['datetime']  >= datetime.strptime('2013-05-23', '%Y-%m-%
                 ]
```

```
2013-05-24    61642
2013-05-23    59463
2013-05-25    48519
2013-05-22     4450
2013-05-29       24
2013-05-28       18
2013-05-30        7
Name: date, dtype: int64
```

Проверяем базовые станции.

- Распоболение базовых станций на карте.
- Распоболение ширины угла.
- Распоболение зоны действия в метрах.

In [14]:

```
CENTER_LAT, CENTER_LONG = 55.7522200, 37.6155600
m = folium.Map(location=(CENTER_LAT, CENTER_LONG))
show_circles_on_map(m, df_data.sample(500), "lat", "long", "blue")
```

Out[14]:

In [15]:

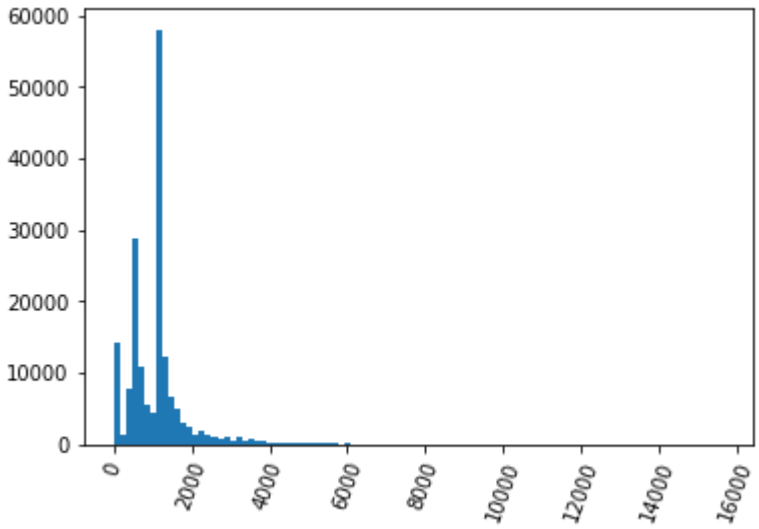
```
df_data['angel'] = np.where(df_data['end_angle'] < df_data['start_angle'], df_data['en
                            df_data['end_angle'] - df_data['start_angle'])

df_data['angel'].value_counts()
```

Out[15]:

```
60.0     156940
47.0       9719
90.0       3104
360.0      2176
65.0       2135
Name: angel, dtype: int64
```

In [16]:

```
plt.hist(df_data['max_dist'].values, bins=100)
plt.xticks(rotation=70)
plt.show()
```



Указываю координаты зоны действия БС для каждого события

По координатам БС, максимальной дистанции приема и направления и угла покрятия определяем координаты прямоугольника, в которых мог находиться абонент при регистрации на БС

In [17]:

```
def BS_rectangle_coordinate(df_):
    df = df_.copy()
    lat_length = haversine_array(CENTER_LAT -.5, CENTER_LONG, CENTER_LAT + .5, CENTER_
    long_length = haversine_array(CENTER_LAT, CENTER_LONG -.5, CENTER_LAT, CENTER_LONG
    max_dist_mean = df['max_dist'].mean()

    df['max_dist_correct'] = np.where(df['max_dist'] < 50, max_dist_mean, df['max_dist
    df['max_dist_correct']

    for feature in ['long', 'lat']:
        if (feature == 'long'):
            coord_length = long_length
        else:
            coord_length = lat_length
        df[feature + '_start'] = np.where(df['angel'] == 360,
                                  df[feature] - df['max_dist_correct']/coord_length/100
                                  df[feature] + df['max_dist_correct'] * np.where(featu









        df[feature + '_end'] = np.where(df['angel'] == 360,
                                  df[feature]+df['max_dist_correct']/coord_length/1000,
                                  df[feature] + df['max_dist_correct'] * np.where(feature









        df[feature + '_min'] = df[[feature + '_start', feature + '_end', feature]].min
        df[feature + '_max'] = df[[feature + '_start', feature + '_end', feature]].max

        df[feature] = (df[feature + '_max'] + df[feature + '_min'])/2
    return df
```

In [18]:

```
df_data_copy = df_data.copy()
df_data = BS_rectangle_coordinate(df_data_copy)
```

In [19]:

```
df_data.head()
```

Out[19]:

	lac	cid	msisdn	imei	event_type	tstamp	long	
0	5029	40798	158529599791	353111050313790	0	1369252800974	37.925474	55.782
1	7782	56870	158520145943	012929002676510	5	1369252801396	37.422847	55.744
2	7794	32226	158521798391	351994049226010	5	1369252802013	37.700660	55.580
3	7758	33528	158537830573	358627016731770	0	1369252802331	37.792199	55.704
4	5060	17568	158510204039	352458051767250	8	1369252803586	37.190576	55.830

Создаю bin по времени для создания ряда событий

In [20]:

```
from sklearn.preprocessing import KBinsDiscretizer
from sklearn.datasets import make_blobs

def CreateTimeBeans(df_, n_bins_ = 10):
    df = df_.copy()

    dt_list = np.reshape(df.datetime.values.tolist(), (len(df.datetime.values.tolist()

    enc = KBinsDiscretizer(n_bins=n_bins_, encode='ordinal', strategy='uniform')
    dt_array_bins = enc.fit_transform(dt_list)

    df['datetime_bin'] = dt_array_bins[:,0]
    df['datetime_bin'] = df['datetime_bin'].astype('int')

    return df, df['datetime_bin'].unique()
```

In [21]:

```
'''
sns.relplot(data=df_data_bin.groupby(['datetime_bin', 'day_of_month'])['lac'].count().r
            y="lac", x="datetime_bin", hue = 'day_of_month')
'''
```

Out[21]:

```
'\nsns.relplot(data=df_data_bin.groupby([\'datetime_bin\', \'day_of_month
\'])[\'lac\'].count().reset_index(),\n            y="lac", x="datetime_bin",
hue = \'day_of_month\')\n'
```

Аггрегирую таблицу с событиями для преобразования в ряд.

При необходимости, добавляю технические события в пустые интервалы времени

In [22]:

```
# декартовое произведение абонентов и интервалов
def CreatePivotBase(df):
    # декартовое произведение абонентов и часов
    df_data_msisdn = pd.DataFrame(df['msisdn'].drop_duplicates()).sort_values('msisdn'
    df_data_datetime = pd.DataFrame(df['datetime_bin'].drop_duplicates()).sort_values(

    df_data_msisdn['K'] = 1
    df_data_datetime['K'] = 1
    df_data_pivot_base = df_data_msisdn.merge(df_data_datetime, on = ['K']).reset_inde

    return df_data_pivot_base
```

In [23]:

```
# аггрегация данных по интервалам
def AggData(df):
    #Создаем датафрейм в котором записаны часовые срезы расположения абонента и нумеруе

    df_data_agg = df.groupby(['msisdn',  'datetime_bin'])['long_min', 'long_max', 'lat_
    df_data_agg['datetime'] = df_data_agg['datetime_bin']
    df_data_agg = df_data_agg.sort_values(['msisdn', 'datetime_bin'])
    df_data_agg['state'] = df_data_agg.groupby(['msisdn'])['msisdn'].cumcount()
    return df_data_agg
```

In [24]:

```
# сдвигаю событие на пол часа и объединяю с оригинальными событиями
from datetime import timedelta
def AddDataDelta(df_):
    df_delta = df_.copy()
    df_delta['datetime'] = df_delta['datetime'] - timedelta(seconds = (60*60))
    df = df_.copy()
    df = df.append(df_delta)
    return df
```

In [25]:

```
# Промежутки времени, в которых не зафиксировано действие абонента, заполняю линейном к
# Концевые точки продолжаю до границы интервала наблюдения константами.

def LinearCombData(df_data_pivot_base, df_data_agg):

    df_data_pivot_base = df_data_pivot_base.merge(df_data_agg[['msisdn', 'datetime_bin
    # указываю состояние, в котором произошло предыдущее событие
    df_data_pivot_base['state_previous'] = 1- df_data_pivot_base['state'].isna()*1
    df_data_pivot_base['state_previous'] = df_data_pivot_base.groupby('msisdn')['state_

    # указываю состояние, в котором произойдет следующее событие
    df_data_pivot_base.drop(columns = ['state'], inplace = True)
    df_data_pivot_base['state_next'] = df_data_pivot_base['state_previous'] + 1
    # подгружаю информацию о предыдущем и следующем событии
    for event in ['previous', 'next']:
        df_data_pivot_base = df_data_pivot_base.merge(df_data_agg[['msisdn', 'long', '
                                                                   'lat', 'lat_max', 'lat_
                                                                        columns =
                                                                                {'long':'lo
                                                                                 'lat':'la
                                                                                 'long_max
                                                                                 'lat_max'
                                                                                 'long_min
                                                                                 'lat_min'
                                                                                  'datetim
                                                                                  'state':
                                                                        on = ['msisdn', 'st
                                                                        how = 'left')
    for feature in ['long', 'long_max', 'long_min', 'lat', 'lat_max', 'lat_min']:
    # расчитываю текущее событие для каждого периода как линейную комбинацию окружающ
        df_data_pivot_base[feature] = np.where(df_data_pivot_base[feature+'_previous']
                                       df_data_pivot_base[feature+'_next'],
                                       np.where(df_data_pivot_base[feature+'_next'].isn
                                               df_data_pivot_base[feature+'_previous']
                                               df_data_pivot_base[feature+'_previous']
                                                    (df_data_pivot_base['datetime_bin'
                                                    (df_data_pivot_base['datetime_bin_
                                                    (df_data_pivot_base[feature+'_next


    return df_data_pivot_base[['msisdn', 'datetime_bin', 'long', 'long_max', 'long_min
```

In [26]:

```
# Промежутки времени, в которых не зафиксировано действие абонента, заполняю линейном к
# Концевые точки продолжаю до границы интервала наблюдения константами.

def CombData(df_data_pivot_base, df_data_agg):

    df_data_pivot_base = df_data_pivot_base.merge(df_data_agg, on = ['msisdn', 'datetim

    return df_data_pivot_base[['msisdn', 'datetime_bin', 'long', 'long_max', 'long_min
```

In [27]:

```
# Создаеь pivot из исходной таблицы
def CreatePivot(df_for_pivot):
    df_pivot = df_for_pivot.pivot(index = 'msisdn', columns = 'datetime_bin', values =
    df_pivot.columns = df_pivot.columns.astype('str') + '_' + 'long'

    for feature in ['long_max', 'long_min', 'lat', 'lat_max', 'lat_min']:
        df_agg_feature = df_for_pivot.pivot(index = 'msisdn', columns = 'datetime_bin'
        df_agg_feature.columns = df_agg_feature.columns.astype('str') + '_' + feature
        df_pivot = pd.concat([df_pivot, df_agg_feature], axis = 1)

    return df_pivot.reset_index()
```

In [28]:

```
# Определяем пары, для которых знаем результаты
# * пары - все, которые указаны в таблице facts
# * не пары: номер, у которого есть пара с ЛЮБЫМ номером не из своей пары (этот номер н
# Упрощение, что нет троек номеров.

def CreateMsisdnPairs(train_pairs, df, frac_):
    msisdn_pairs = pd.concat([train_pairs[[0, 1]].rename(columns = {0:'msisdn_x', 1:'ms
  #                         ,facts[[1, 0]].rename(columns = {1:'msisdn_x', 0:'msisdn_y'}
                          ], axis = 0)
    msisdn_pairs_dubli = pd.concat([train_pairs[[0, 1]].rename(columns = {0:'msisdn_x'}
                          ,train_pairs[[1, 0]].rename(columns = {1:'msisdn_x', 0:'msisdn
                          ], axis = 0)
    msisdn_pairs['is_pair'] = 1
    msisdn_pairs['K'] = 1
    msisdn_pairs_dubli['K'] = 1

    df_data_msisdn = pd.DataFrame(df['msisdn'].drop_duplicates()).sort_values('msisdn'
    df_data_msisdn['K'] = 1

    # любой номер из пары с другими номерами не создает пары
    msisdn_pairs_part1 = df_data_msisdn[~df_data_msisdn['msisdn'].isin(msisdn_pairs_dub
    msisdn_pairs_part1['is_pair'] = 0

    msisdn_pairs_part1 = msisdn_pairs_part1.sample(frac = frac_)

    # объедтняю пары и не пары
    msisdn_pairs = pd.concat([msisdn_pairs, msisdn_pairs_part1
                          # , msisdn_pairs_part2
                          ], axis = 0).drop(columns = ['K'])

    # перемешиваю первый и второй номер
    msisdn_pairs_part1 = msisdn_pairs.sample(frac = .5)
    msisdn_pairs_part2 = msisdn_pairs.append(msisdn_pairs_part1).drop_duplicates(keep =
    msisdn_pairs = msisdn_pairs_part1.append(msisdn_pairs_part2)
    print(msisdn_pairs.shape)

    return msisdn_pairs
```

In [29]:

```
#Создаю таблицу для обучения

def CreateAndProcessPivot(msisdn_pairs, data_agg_pivot, bins_list, identifier,
                          dist_ind = True,
                          coord_ind = True,
                          center_dist_ind = True ):

    df_X = msisdn_pairs.merge(data_agg_pivot.rename(columns = {'msisdn':'msisdn_x'}), (
    # считаю расстояние в каждом часе
    columns_name_dist = list()
    columns_name_intersect = list()
    columns_name_lat_x = list()
    columns_name_long_x = list()
    columns_name_lat_y = list()
    columns_name_long_y = list()
    columns_name_center_dist_x = list()
    columns_name_center_dist_y = list()

    columns_name = list()
    columns_name_identifier = list()

    lat_length = haversine_array(CENTER_LAT -.5, CENTER_LONG, CENTER_LAT + .5, CENTER_
    long_length = haversine_array(CENTER_LAT, CENTER_LONG -.5, CENTER_LAT, CENTER_LONG

    for i in bins_list:
        str_i = str(int(i))
        columns_name_dist.append(str_i+"_dist")
        columns_name_intersect.append(str_i+"_intersect")
        columns_name_lat_x.append(str_i+"_lat_x")
        columns_name_long_x.append(str_i+"_long_x")
        columns_name_lat_y.append(str_i+"_lat_y")
        columns_name_long_y.append(str_i+"_long_y")
        columns_name_center_dist_x.append(str_i+"_center_dist_x")
        columns_name_center_dist_y.append(str_i+"_center_dist_y")

        lat_diff = (df_X[str_i+"_lat_x"].values - df_X[str_i+"_lat_y"].values)*lat_leng
        long_diff = (df_X[str_i+"_long_x"].values - df_X[str_i+"_long_y"].values)*long_
        dist = np.sqrt(lat_diff**2 + long_diff**2)
        df_X[str_i+"_dist"] = dist
        df_X[str_i+'_intersect'] = ((df_X[str_i+'_lat_max_x'] >= df_X[str_i+'_lat_min_
                                (df_X[str_i+'_lat_max_y'] >= df_X[str_i+'_lat_min_x']) &
                                (df_X[str_i+'_long_max_x'] >= df_X[str_i+'_long_min_y'])
                                (df_X[str_i+'_long_max_y'] >= df_X[str_i+'_long_min_x'])
        center_x_lat_diff = (df_X[str_i+"_lat_x"].values - CENTER_LAT)*lat_length
        center_y_lat_diff = (df_X[str_i+"_lat_y"].values - CENTER_LAT)*lat_length

        center_x_long_diff = (df_X[str_i+"_long_x"].values - CENTER_LONG)*long_length
        center_y_long_diff = (df_X[str_i+"_long_y"].values - CENTER_LONG)*long_length

        df_X[str_i+"_center_dist_x"] = np.sqrt(center_x_lat_diff**2 + center_x_long_di
        df_X[str_i+"_center_dist_y"] = np.sqrt(center_y_lat_diff**2 + center_y_long_di

    # считаю среднее расстояние
    df_X['dist_mean'] = df_X[columns_name_dist].mean(axis = 1)
    df_X['intersect_mean'] = df_X[columns_name_intersect].mean(axis = 1)
    df_X['intersect_count'] = df_X[columns_name_dist].count(axis = 1)/len(bins_list)
    df_X['center_dist_x_mean'] = df_X[columns_name_center_dist_x].mean(axis = 1)
    df_X['center_dist_y_mean'] = df_X[columns_name_center_dist_y].mean(axis = 1)

```

```
    if(dist_ind):
        columns_name = [*columns_name, *columns_name_dist]

    if(coord_ind):
        columns_name = [*columns_name, *columns_name_lat_x, *columns_name_lat_y, *colu

    if(center_dist_ind):
        columns_name = [*columns_name, *columns_name_center_dist_x, *columns_name_cent

    columns_name = [*columns_name,
                    *['dist_mean', 'intersect_mean', 'intersect_count', 'center_dist_x

    columns_name = [*columns_name, *['is_pair', 'msisdn_x', 'msisdn_y']]
    df_X = df_X[columns_name].set_index(['is_pair', 'msisdn_x', 'msisdn_y'])
    df_X.columns = df_X.columns + '_' + identifier

    print(df_X.shape)

    return df_X
```

Создаю таблицу

- Линейная комбинация недостающий событий
- Сдвиг по времени событие для создания большего количества общих интервалов

Создаем pivottable

Создаю pivot с координатами в каждый период

Создаем таблицу с данными для обучения

Описание целевого события.

- пары - все, которые указаны в таблице facts
- не пары: номер, у которого есть пара с ЛЮБЫМ номером не из своей пары (этот номер не обязательно из таблицы facts) Упрощение, что нет троек номеров.

Если (a, b) пара, то (b, a) - тоже пара. Если (a, b) - пара, то (a, c) Симметричные пары убираю, для этого выбираю (a > b), а потом перемешиваю.

Пары (a,a) убираем Модифицируем на шум одинаковых абоенентов(?) Смешаем время для сравнения, расширяем диапозоны в обе стороны для второго абонента.

Таблица

Создаю таблицу для обучения

In [30]:

```
#модель
!pip install xgboost

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import average_precision_score

from sklearn.metrics import roc_curve
from sklearn.metrics import plot_roc_curve
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import plot_precision_recall_curve
from sklearn.metrics import average_precision_score
import warnings
warnings.filterwarnings('ignore')
```

```
Requirement already satisfied: xgboost in c:\users\ekaterina.adischeva\appda
ta\local\continuum\anaconda3\lib\site-packages (1.3.3)
Requirement already satisfied: scipy in c:\users\ekaterina.adischeva\appdata
\local\continuum\anaconda3\lib\site-packages (from xgboost) (1.2.1)
Requirement already satisfied: numpy in c:\users\ekaterina.adischeva\appdata
\local\continuum\anaconda3\lib\site-packages (from xgboost) (1.16.2)
```

In [33]:

```
Cls_result_agg = pd.DataFrame()
BINS_QNT = 90
ln_ind = True
shift_inf = False
cls_ind = 'RF'
dist_ind_ln, coord_ind_ln, center_dist_ind_ln = True, False, False
dist_ind_shift, coord_ind_shift, center_dist_ind_shift = True, False, False
clfs = []

for bins_q, ln_ind, shift_ind, clf_ind, dist_ind_ln, coord_ind_ln, center_dist_ind_ln,
    [30, True, True, 'RF', True, True, False, True, False, False],
    [30, True, True, 'GB', True, True, False, True, False, False],
                                                            [30, True, True, 'RF',
                                                            [60, True, False, 'GB'
                                                            [60, True, False, 'RF'
                                                            [60, False, True, 'GB'
                                                            [60, False, True, 'RF'
                                                            [60, True, False, 'GB'
                                                            [60, True, False, 'RF'
                                                            [60, False, True, 'GB'
                                                            [60, False, True, 'RF'
                                                            [90, True, False, 'RF'
                                                            [90, True, False, 'GB'
                                                            [90, True, False, 'RF'
                                                            [90, True, False, 'GB'
                                                            [90, False, True, 'RF'
                                                            [90, False, True, 'GB'
                                                            [90, False, True, 'RF'
                                                            [90, False, True, 'GB'
                                                            [90, False, True, 'RF'
                                                            [90, False, True, 'GB'
                                                            [90, False, True, 'RF'
                                                            [90, False, True, 'GB'
                                                           ]:

    df_data_bin_ln, bins_list_ln = CreateTimeBeans(df_data, BINS_QNT)
    df_data_pivot_base_ln = CreatePivotBase(df_data_bin_ln)
    df_data_bin_agg_ln = AggData(df_data_bin_ln)
    df_data_agg_for_pivot_ln = LinearCombData(df_data_pivot_base_ln, df_data_bin_agg_l

    df_data_shift = AddDataDelta(df_data)
    df_data_bin_shift, bins_list_shift = CreateTimeBeans(df_data_shift, BINS_QNT)
    df_data_pivot_base_shift = CreatePivotBase(df_data_bin_shift)
    df_data_bin_agg_shift = AggData(df_data_bin_shift)
    df_data_agg_for_pivot_shift = CombData(df_data_pivot_base_shift, df_data_bin_agg_s

    df_msisdn_pairs = CreateMsisdnPairs(facts, df_data, 0.25)

    df_data_agg_pivot_ln = CreatePivot(df_data_agg_for_pivot_ln)
    df_data_agg_pivot_shift = CreatePivot(df_data_agg_for_pivot_shift)

    df_X_ln = CreateAndProcessPivot(df_msisdn_pairs, df_data_agg_pivot_ln, bins_list_l
    df_X_shift = CreateAndProcessPivot(df_msisdn_pairs, df_data_agg_pivot_shift, bins_

    df_X = pd.DataFrame()
    if (ln_ind):
        df_X = pd.concat([df_X_ln, df_X], axis = 1)
    if (shift_ind):
        df_X = pd.concat([df_X_shift, df_X], axis = 1)
```

```
60
61      df_X.reset_index(inplace = True)
62
63      X = df_X.drop(columns = ['msisdn_x', 'msisdn_y', 'is_pair']).fillna(0)
64      y = df_X['is_pair']
65
66      X_train, X_test, y_train, y_test = train_test_split(
67          X, y, test_size=0.3, random_state=42)
68      if(clf_ind == 'RF'):
69          clf = RandomForestClassifier(random_state=0)
70      else:
71          clf = XGBClassifier(random_state=42)
72
73      clf.fit(X_train, y_train)
74
75      y_test_predict = clf.predict(X_test)
76      y_train_predict = clf.predict(X_train)
77      y_test_predict_proba = clf.predict_proba(X_test)
78
79      print('[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist
80           [bins_q, ln_ind, shift_inf, cls_ind,
81          dist_ind_ln, coord_ind_ln, center_dist_ind_ln,
82          dist_ind_shift, coord_ind_shift, center_dist_ind_shift])
83
84      print('roc_auc_score:', roc_auc_score(y_train, y_train_predict),
85            'average_precision_score:', average_precision_score(y_train, y_train_predict
86      print("roc_auc_score:", roc_auc_score(y_test, y_test_predict),
87            'average_precision_score:', average_precision_score(y_test, y_test_predict))
88
89
90      disp = plot_precision_recall_curve(clf, X_test, y_test)
91      average_precision = average_precision_score(y_test, y_test_predict)
92      print(average_precision)
93
94      disp = plot_roc_curve(clf, X_test, y_test)
95      roc_auc = roc_auc_score(y_test, y_test_predict)
96      print(roc_auc)
97
98
99      Cls_result = pd.DataFrame(y_test)
100     Cls_result['proba'] = y_test_predict_proba[:,1]
101     Cls_result['predict'] = y_test_predict
102     Cls_result =  Cls_result.join(df_X[['msisdn_x', 'msisdn_y']])
103
104     alpha = .1
105
106     Cls_result['predict_alpha'] = np.where(Cls_result['proba']> alpha, 1, 0)
107     Cls_result_agg = Cls_result_agg.append(Cls_result.groupby(['is_pair', 'predict_alp
108     Cls_result_agg = Cls_result_agg.append(Cls_result.groupby(['is_pair', 'predict']).
109
```

```
(129303, 3)
(127227, 435)
(127227, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [30, Tru
e, False, 'RF', True, True, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5483346556393673 average_precision_score: 0.0422082339843
```

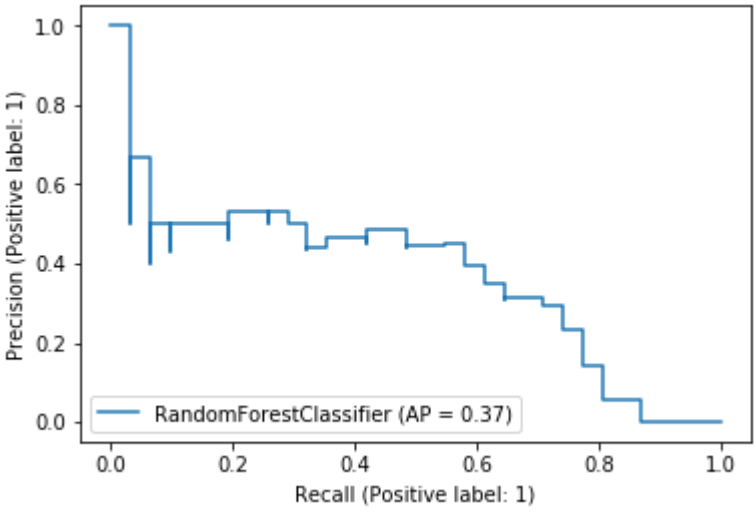
```
6712
0.04220823398436712
0.5483346556393673
(129302, 3)
(127154, 435)
(127154, 91)
[04:11:19] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [30, Tru
e, False, 'RF', True, True, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.634523571703866 average_precision_score: 0.13511345785836
57
0.1351134578583657
0.634523571703866
(129314, 3)
(127190, 263)
(127190, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [30, Tru
e, False, 'RF', True, False, True, True, False, False]
roc_auc_score: 0.9939759036144578 average_precision_score: 0.9879630390193
753
roc_auc_score: 0.5693395154027089 average_precision_score: 0.0542312362620
563
0.0542312362620563
0.5693395154027089
(129305, 3)
(127256, 435)
(127256, 91)
[04:15:37] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [60, Tru
e, False, 'RF', True, True, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5776991036982644 average_precision_score: 0.0847560474613
4122
0.08475604746134122
0.5776991036982644
(129303, 3)
(127142, 435)
(127142, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [60, Tru
e, False, 'RF', True, True, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.560566700857963 average_precision_score: 0.07002436604198
395
0.07002436604198395
0.560566700857963
(129300, 3)
(127196, 263)
(127196, 435)
```

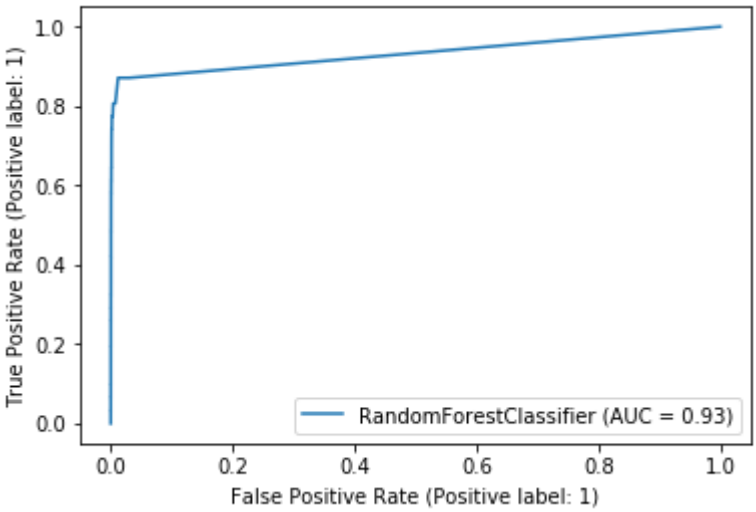
```
[04:19:51] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [60, Fal
se, False, 'RF', True, False, True, True, True, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.513797080794038 average_precision_score: 0.00438943703393
1125
0.004389437033931125
0.513797080794038
(129317, 3)
(127211, 263)
(127211, 435)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [60, Fal
se, False, 'RF', True, False, True, True, True, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5 average_precision_score: 0.0009957027565244733
0.0009957027565244733
0.5
(129286, 3)
(127182, 263)
(127182, 91)
[04:22:08] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [60, Tru
e, False, 'RF', True, False, True, True, False, False]
roc_auc_score: 0.99375 average_precision_score: 0.9875112325474295
roc_auc_score: 0.5768574876692202 average_precision_score: 0.0847809771148
7831
0.08478097711487831
0.5768574876692202
(129317, 3)
(127224, 263)
(127224, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [60, Tru
e, False, 'RF', True, False, True, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5713367705577745 average_precision_score: 0.0668772644249
4835
0.06687726442494835
0.5713367705577745
(129304, 3)
(127170, 263)
(127170, 263)
[04:25:58] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [60, Fal
se, False, 'RF', True, False, True, True, False, True]
```

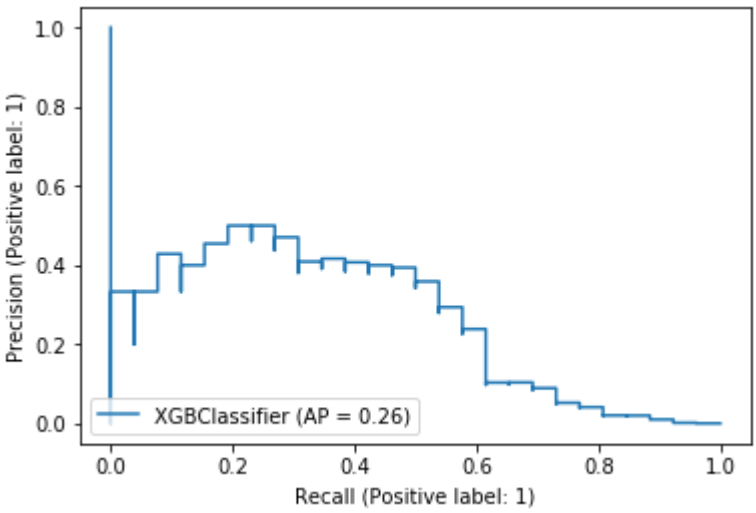
```
roc_auc_score: 0.9943820224719101 average_precision_score: 0.9887752785007
013
roc_auc_score: 0.5166273182760159 average_precision_score: 0.0090934706822
888
0.0090934706822888
0.5166273182760159
(129311, 3)
(127101, 263)
(127101, 263)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [60, Fal
se, False, 'RF', True, False, True, True, False, True]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5 average_precision_score: 0.0011276913797172905
0.0011276913797172905
0.5
(129285, 3)
(127183, 91)
(127183, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, Tru
e, False, 'RF', True, False, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.62156915297164 average_precision_score: 0.169133017173903
06
0.16913301717390306
0.62156915297164
(129297, 3)
(127155, 91)
(127155, 91)
[04:29:51] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, Tru
e, False, 'RF', True, False, False, True, False, False]
roc_auc_score: 0.9941860465116279 average_precision_score: 0.9883833279684
293
roc_auc_score: 0.5605535864495826 average_precision_score: 0.0613662776611
37024
0.061366277661137024
0.5605535864495826
(129307, 3)
(127065, 349)
(127065, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, Tru
e, False, 'RF', False, True, False, True, False, False]
roc_auc_score: 0.99375 average_precision_score: 0.9875112429029176
roc_auc_score: 0.49998687009269716 average_precision_score: 0.001023084994
7534103
0.0010230849947534103
0.49998687009269716
(129297, 3)
(127164, 349)
(127164, 91)
[04:32:14] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
```

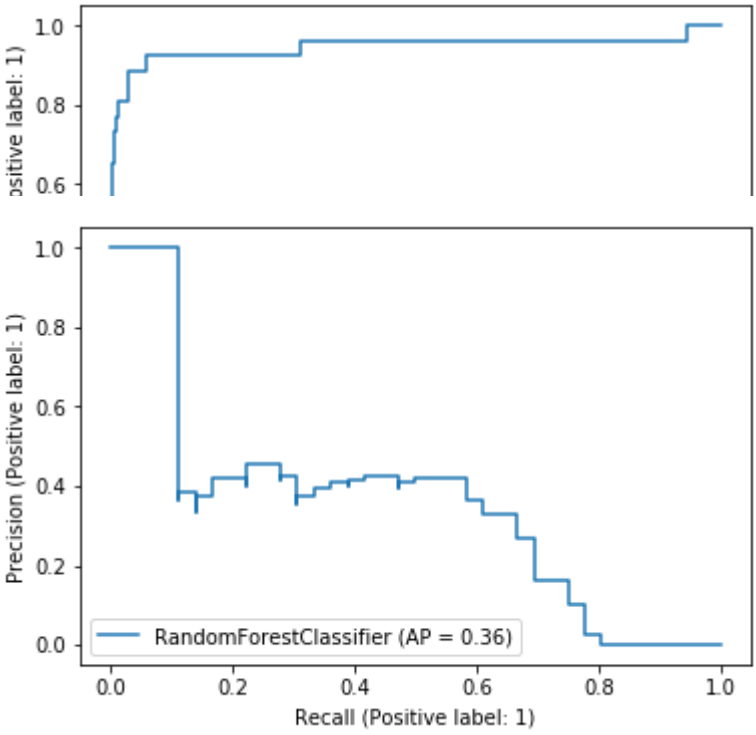
```
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, Tru
e, False, 'RF', False, True, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5254835907796995 average_precision_score: 0.0082958631582
48479

0.008295863158248479
0.5254835907796995
(129296, 3)
(127213, 349)
(127213, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, False,
False, 'RF', False, True, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5142594875291773 average_precision_score: 0.010414701463857
735
0.010414701463857735
0.5142594875291773
(129307, 3)
(127176, 349)
(127176, 91)
[04:34:15] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, False,
False, 'RF', False, True, False, True, False, False]
roc_auc_score: 0.9235294117647059 average_precision_score: 0.847204853207135
5
roc_auc_score: 0.5440782966036699 average_precision_score: 0.044930165078376
38
0.04493016507837638
0.5440782966036699
(129315, 3)
(127183, 263)
(127183, 349)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, False,
False, 'RF', True, False, True, False, True, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5 average_precision_score: 0.0013104442405975625
0.0013104442405975625
0.5
(129309, 3)
(127172, 263)
(127172, 349)
[04:35:59] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, False,
False, 'RF', True, False, True, False, True, False]
roc_auc_score: 1.0 average_precision_score: 1.0
```

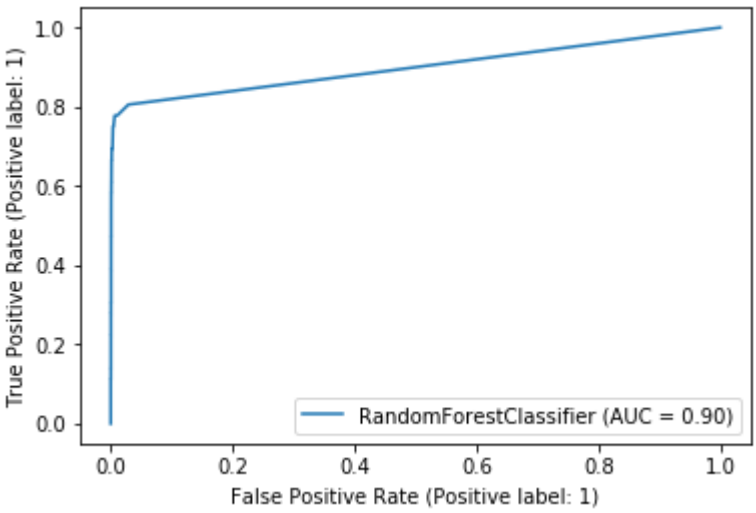
```
roc_auc_score: 0.5127549184039095 average_precision_score: 0.005269520209759
253
0.005269520209759253
0.5127549184039095
(129309, 3)
(127176, 521)
(127176, 263)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, False,
False, 'RF', False, True, True, True, False, True]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5 average_precision_score: 0.0007600975021623464
0.0007600975021623464
0.5
(129311, 3)
(127195, 521)
(127195, 263)
[04:38:04] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, False,
False, 'RF', False, True, True, True, False, True]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5166404399101296 average_precision_score: 0.011871089097955
63
0.01187108909795563
0.5166404399101296
(129315, 3)
(127176, 263)
(127176, 521)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, False,
False, 'RF', True, False, True, False, True, True]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5 average_precision_score: 0.001048410347810133
0.001048410347810133
0.5
(129314, 3)
(127130, 263)
(127130, 521)
[04:40:18] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, False,
False, 'RF', True, False, True, False, True, True]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.4999343745898412 average_precision_score: 0.001153674716169
8
0.0011536747161698
0.4999343745898412
```

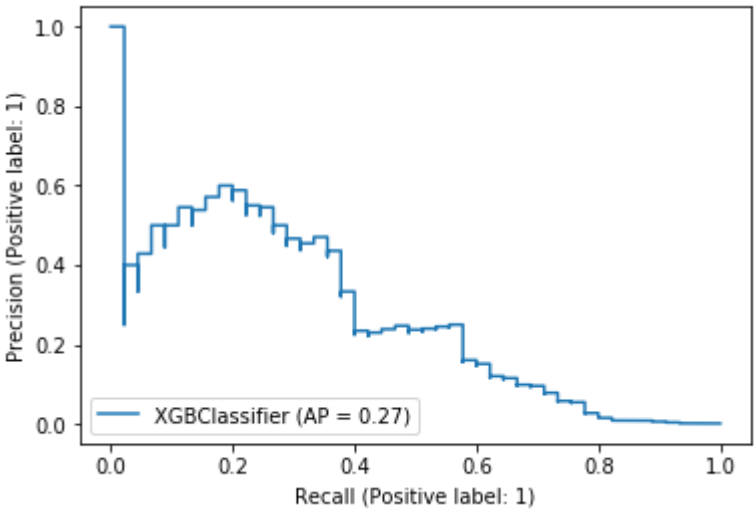

Precision (Positive label: 1)
Recall (Positive label: 1)
RandomForestClassifier (AP = 0.37)
0.0
0.2
0.4
0.6
0.8
1.0

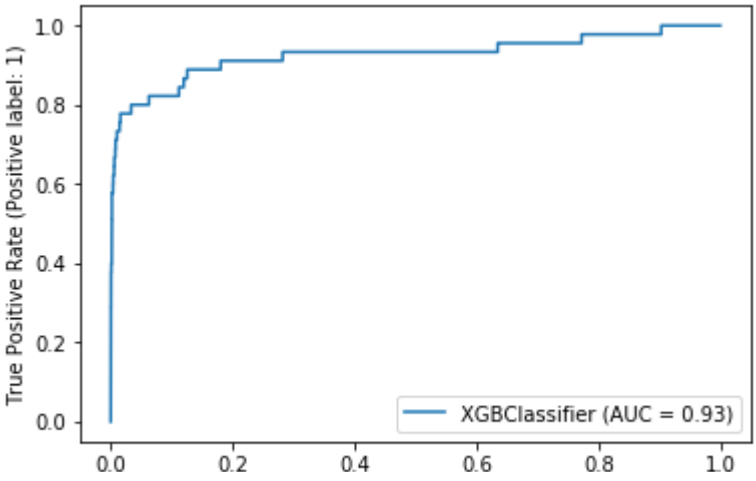

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.93)
0.0
0.2
0.4
0.6
0.8
1.0

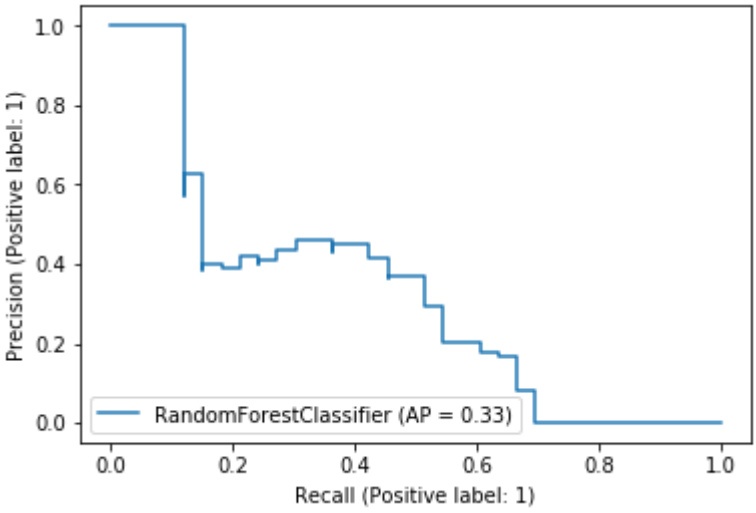

Precision (Positive label: 1)
Recall (Positive label: 1)
XGBClassifier (AP = 0.26)
0.0
0.2
0.4
0.6
0.8
1.0

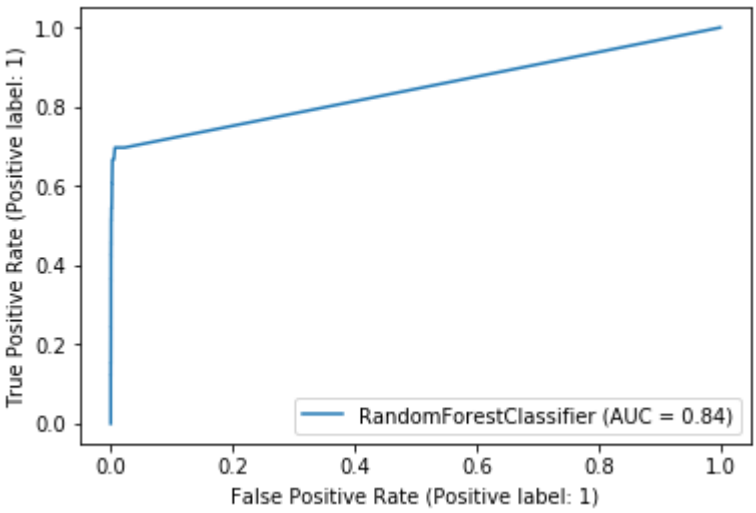

1.0
0.8
0.6
Precision (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
RandomForestClassifier (AP = 0.36)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)

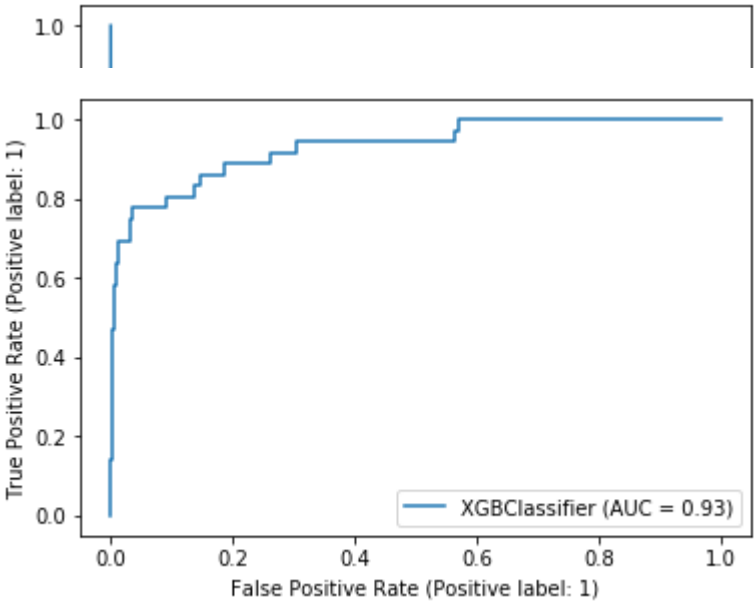

True Positive Rate (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
RandomForestClassifier (AUC = 0.90)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

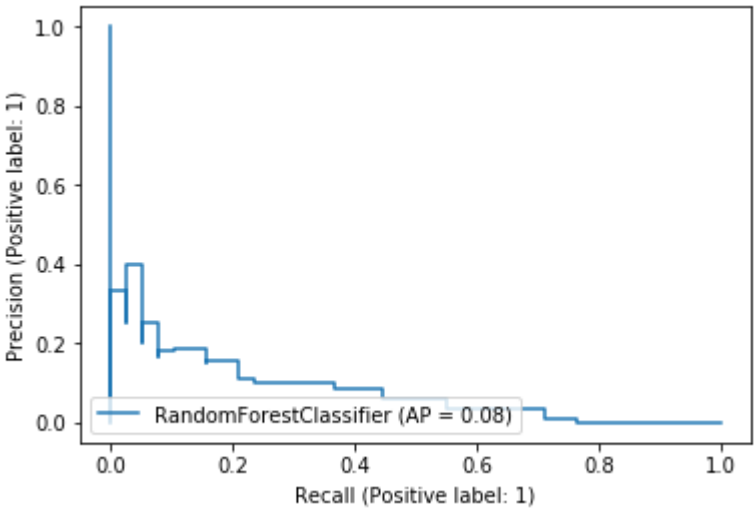

Precision (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
XGBClassifier (AP = 0.27)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)

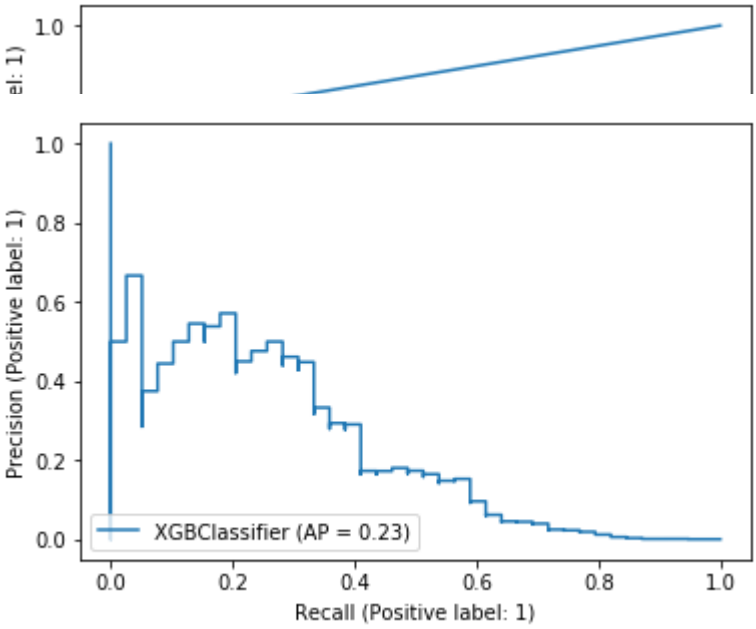

True Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.93)

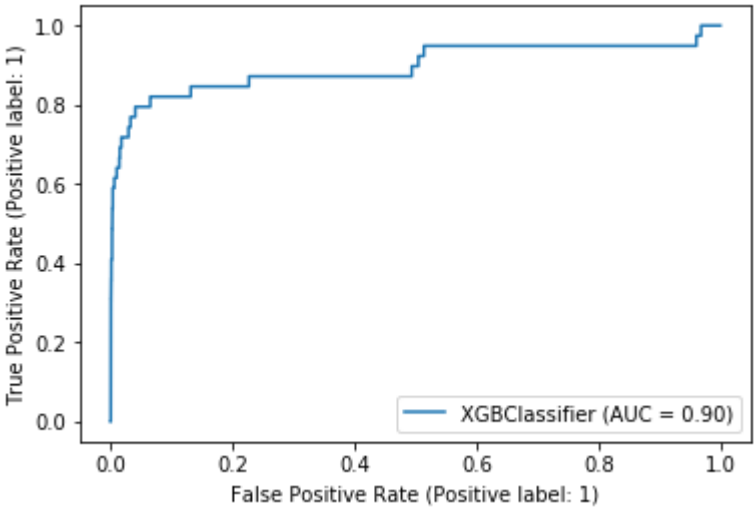

Precision (Positive label: 1)
RandomForestClassifier (AP = 0.33)
Recall (Positive label: 1)

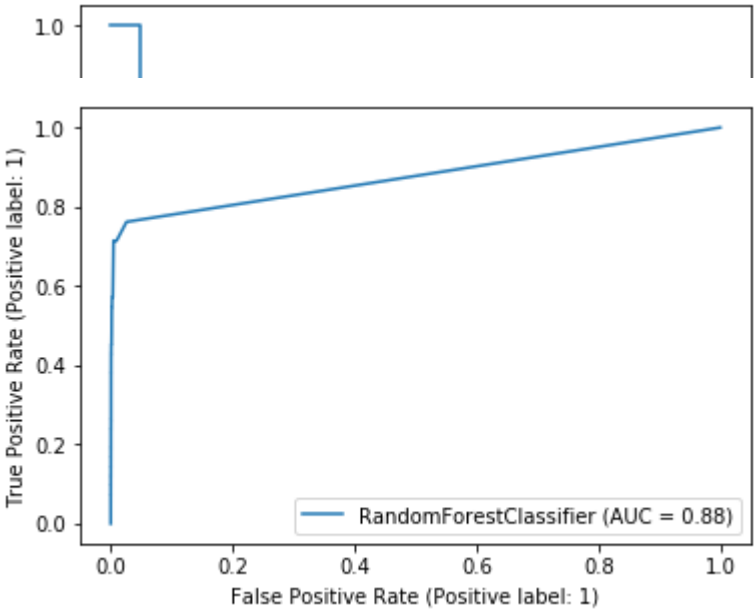

True Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.84)
False Positive Rate (Positive label: 1)

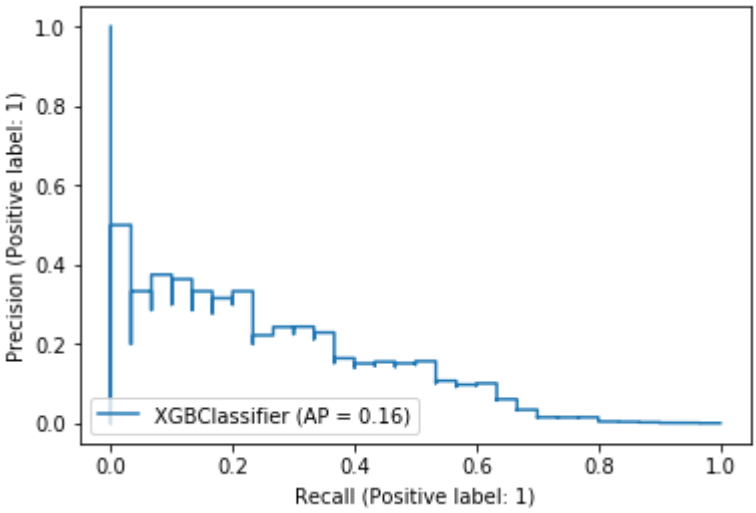

1.0
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.93)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

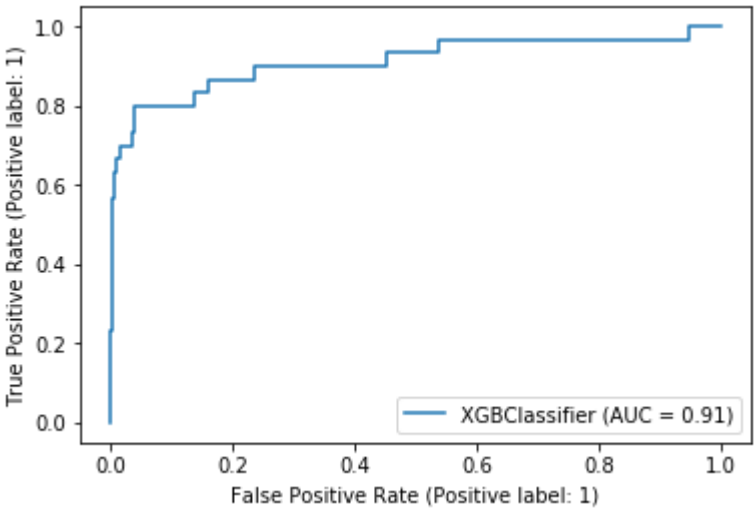

1.0
0.8
0.6
0.4
0.2
0.0
Precision (Positive label: 1)
RandomForestClassifier (AP = 0.08)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)

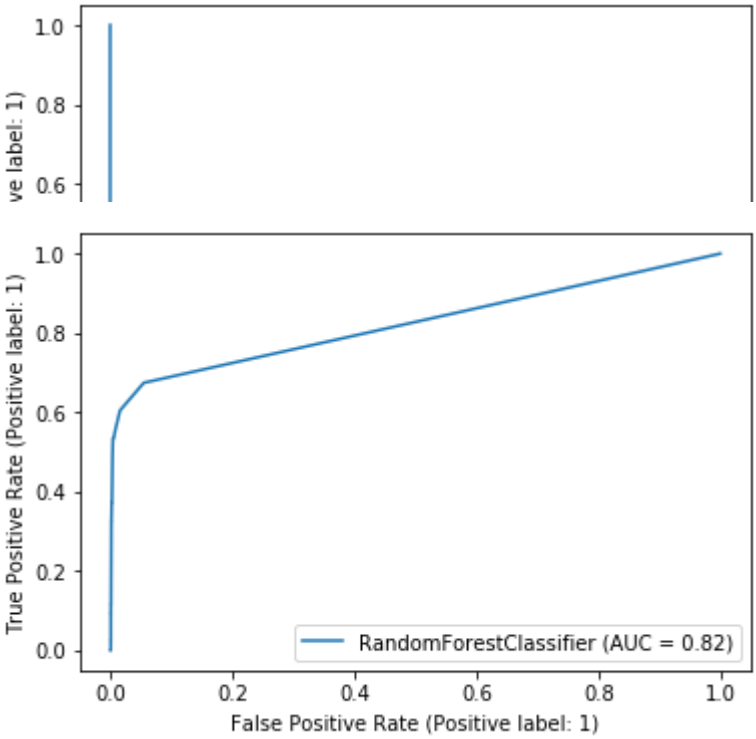

el: 1)
1.0
Precision (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
XGBClassifier (AP = 0.23)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)

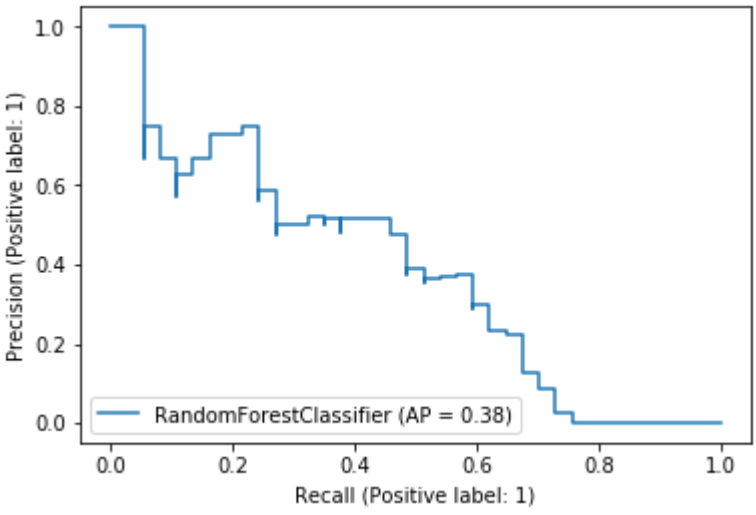

True Positive Rate (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
XGBClassifier (AUC = 0.90)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

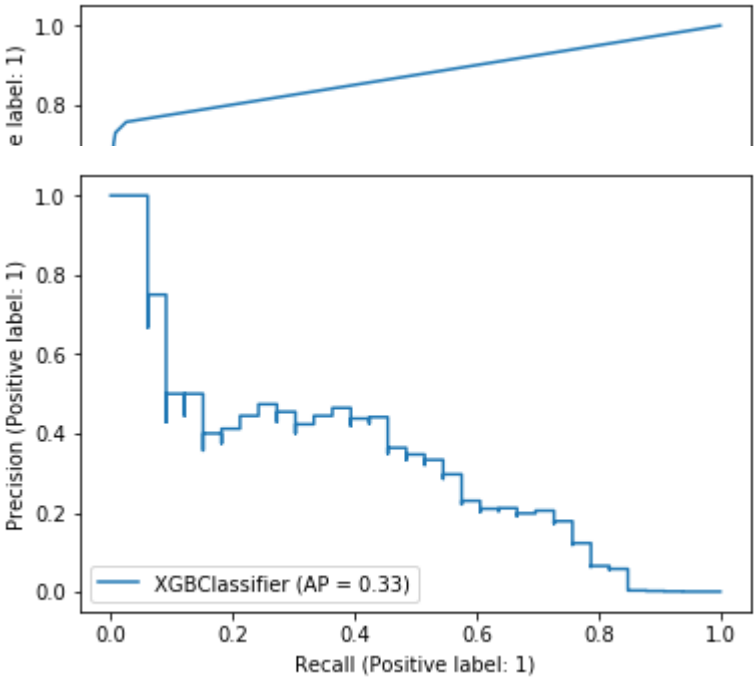

1.0
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.88)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

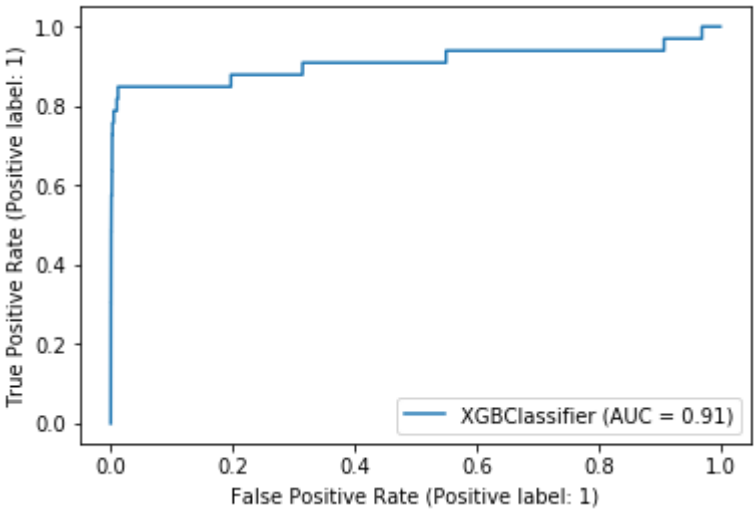

1.0
0.8
0.6
0.4
0.2
0.0
Precision (Positive label: 1)
XGBClassifier (AP = 0.16)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)

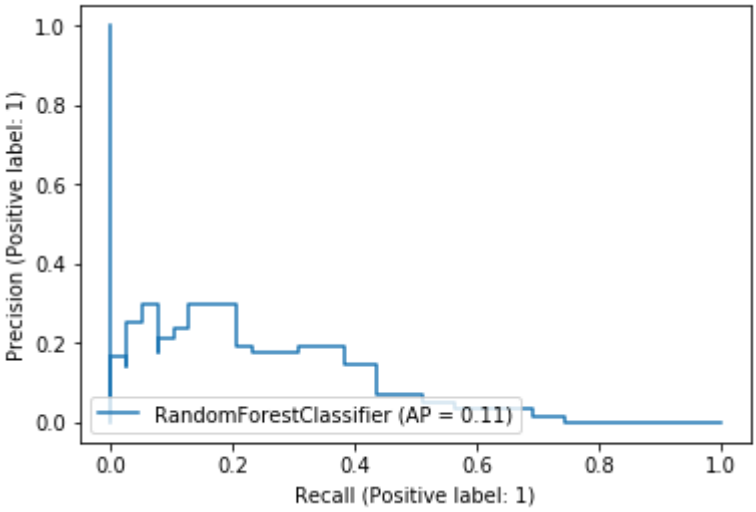

1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.91)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

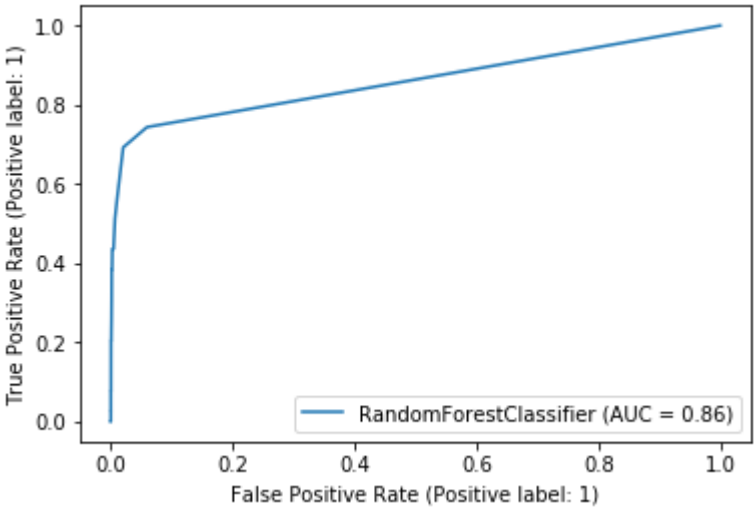

ve label: 1)
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.82)

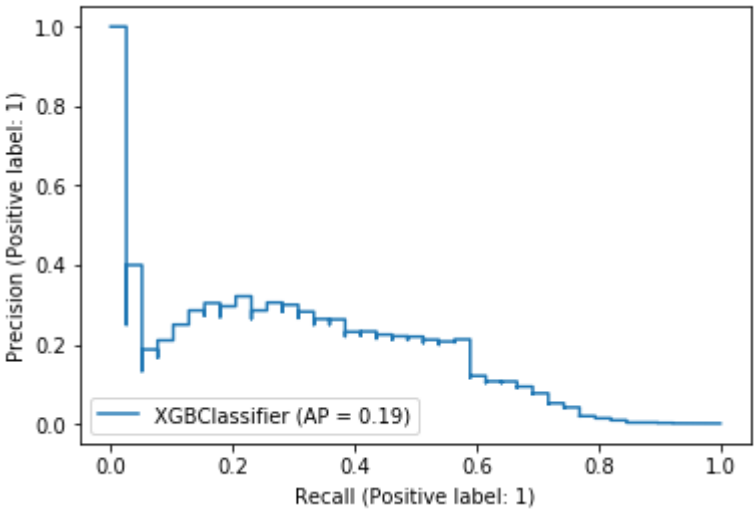

Precision (Positive label: 1)
Recall (Positive label: 1)
RandomForestClassifier (AP = 0.38)

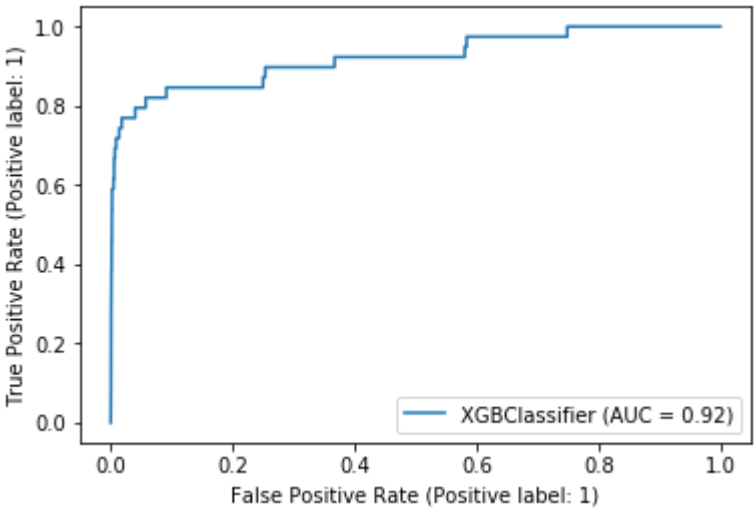

e label: 1)
1.0
0.8
Precision (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
XGBClassifier (AP = 0.33)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)

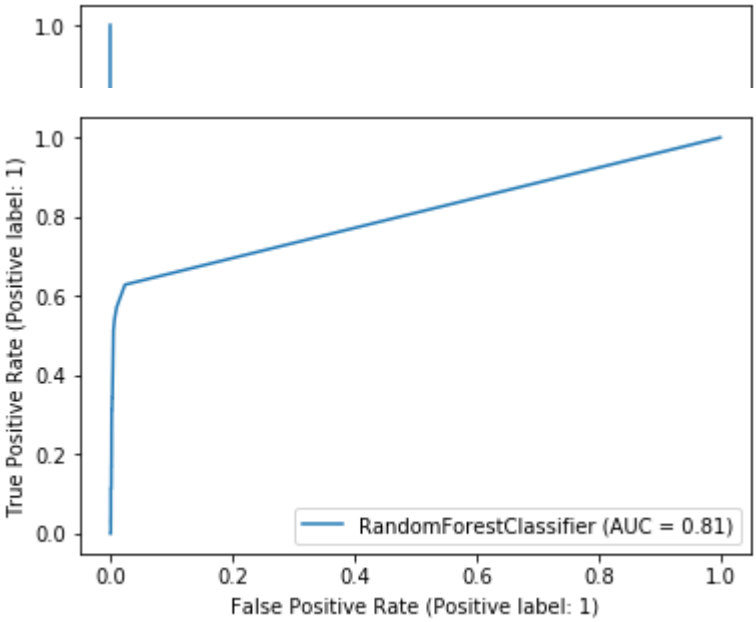

True Positive Rate (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
XGBClassifier (AUC = 0.91)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

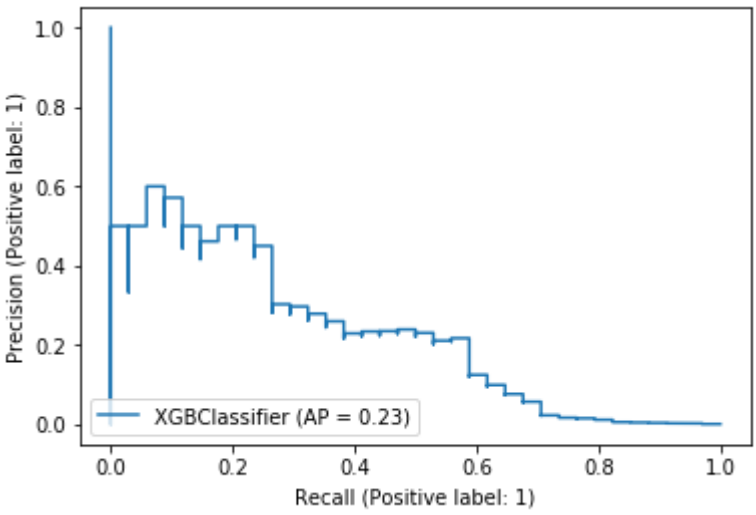

Precision (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
RandomForestClassifier (AP = 0.11)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)

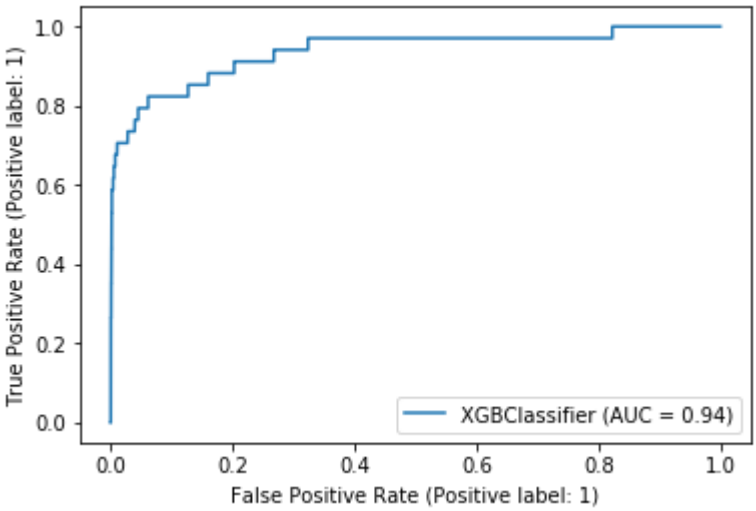

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.86)


Precision (Positive label: 1)
Recall (Positive label: 1)
XGBClassifier (AP = 0.19)

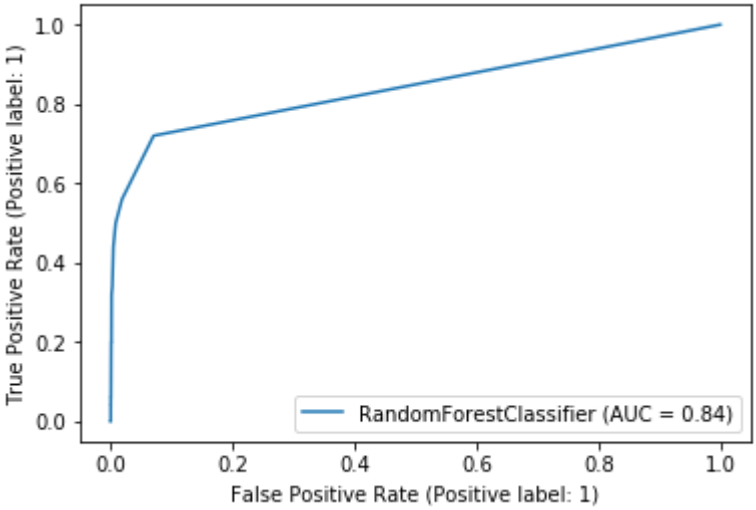

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.92)

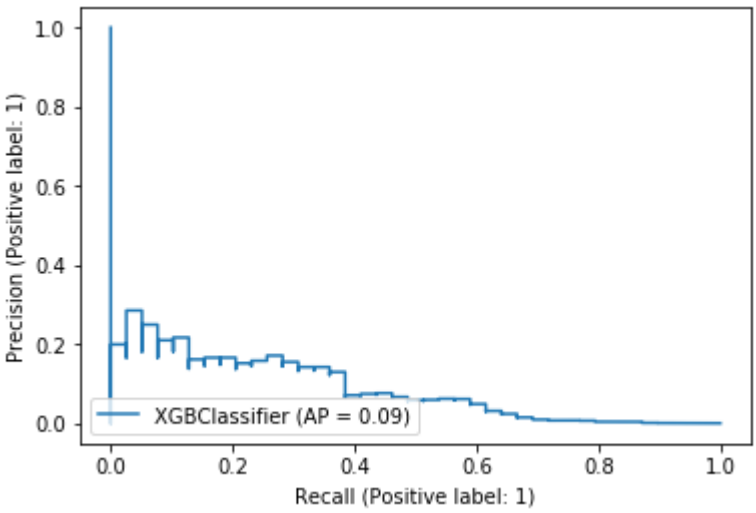

1.0
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.81)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

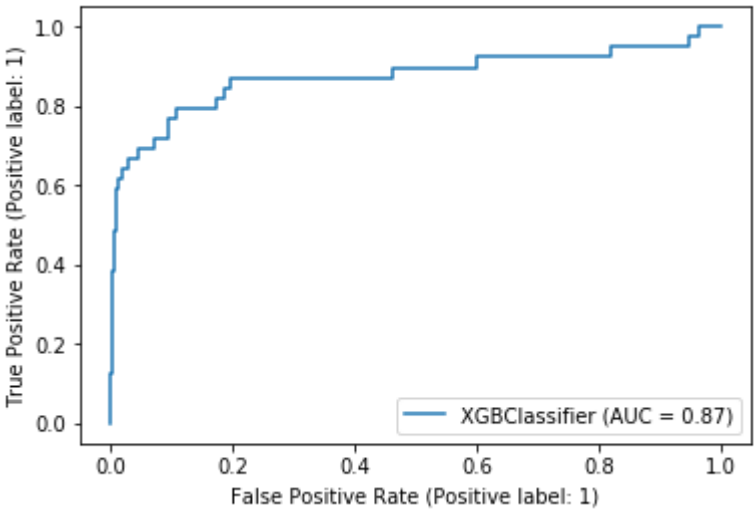

1.0
0.8
0.6
0.4
0.2
0.0
Precision (Positive label: 1)
XGBClassifier (AP = 0.23)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)

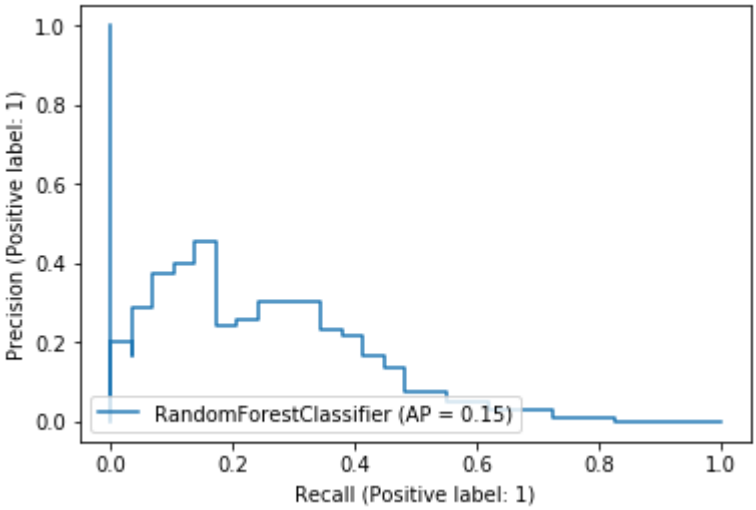

1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.94)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

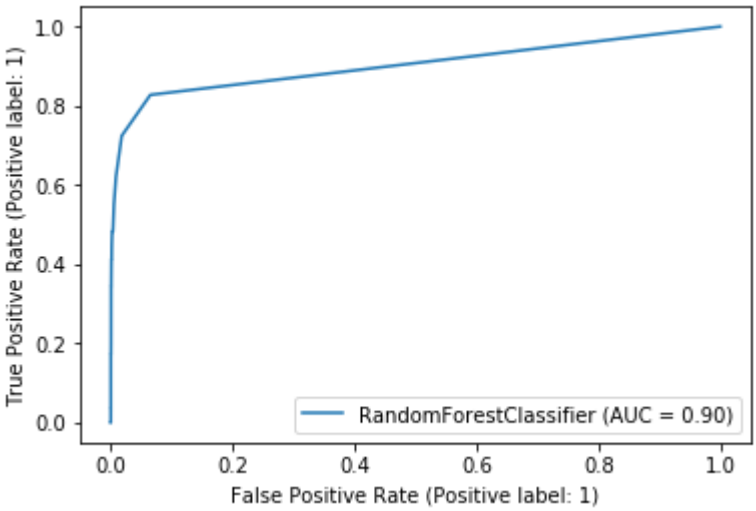

1.0
0.8
0.6

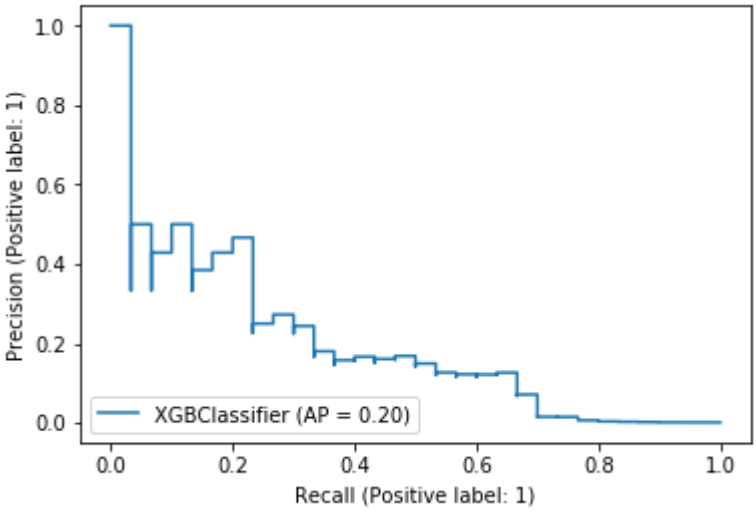

True Positive Rate (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
RandomForestClassifier (AUC = 0.84)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

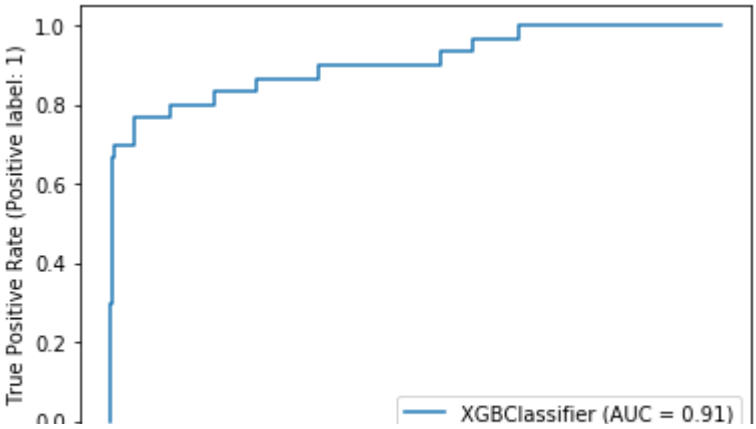

Precision (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
XGBClassifier (AP = 0.09)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)

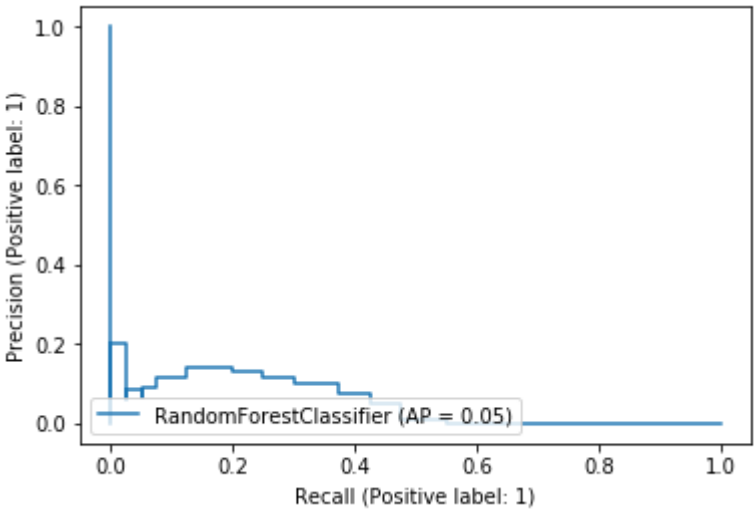

True Positive Rate (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
XGBClassifier (AUC = 0.87)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

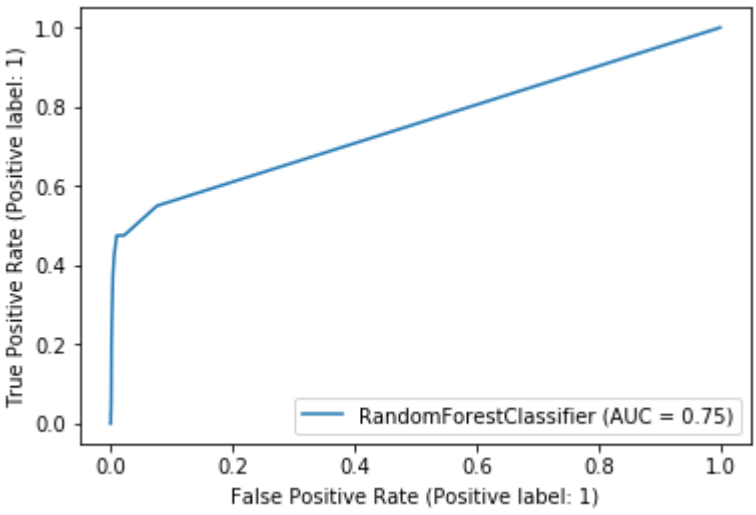

Precision (Positive label: 1)
Recall (Positive label: 1)
RandomForestClassifier (AP = 0.15)

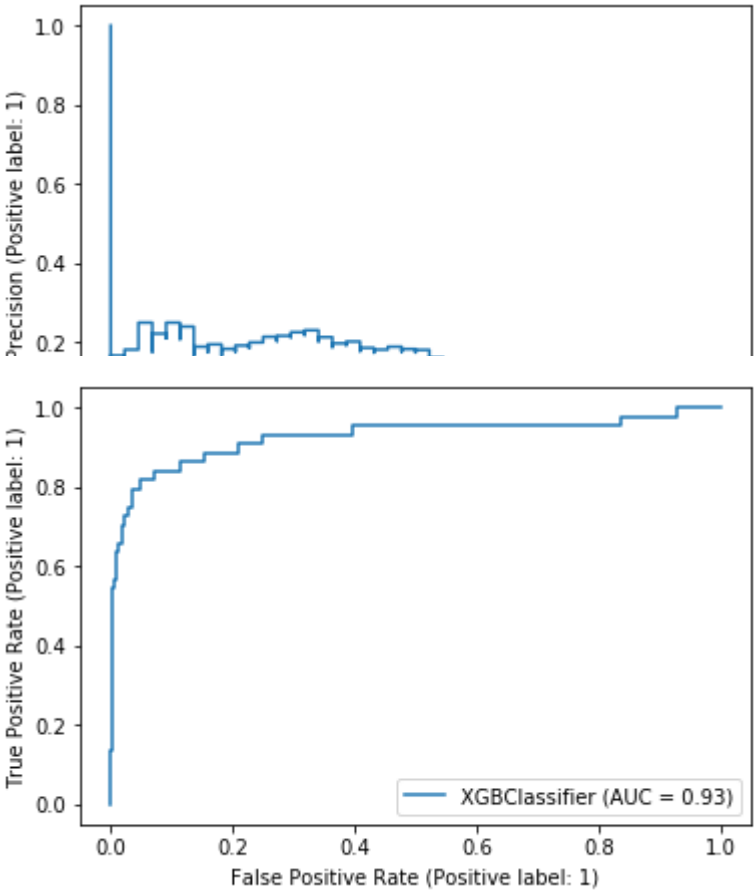

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.90)


Precision (Positive label: 1)
Recall (Positive label: 1)
XGBClassifier (AP = 0.20)


True Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.91)


Precision (Positive label: 1)
RandomForestClassifier (AP = 0.05)
Recall (Positive label: 1)


True Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.75)
False Positive Rate (Positive label: 1)


Precision (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
True Positive Rate (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
XGBClassifier (AUC = 0.93)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

```
1 # Модель
```

In [ ]:

```
#модель
#!pip install xgboost

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier

X = df_X.drop(columns = ['msisdn_x', 'msisdn_y', 'is_pair']).fillna(0)
y = df_X['is_pair']

X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=42)

#clf = RandomForestClassifier(random_state=0)
clf = XGBClassifier(random_state=42)

clf.fit(X_train, y_train)

y_test_predict = clf.predict(X_test)
y_train_predict = clf.predict(X_train)
y_test_predict_proba = clf.predict_proba(X_test)

from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import average_precision_score

print('roc_auc_score:', roc_auc_score(y_train, y_train_predict),
      'average_precision_score:', average_precision_score(y_train, y_train_predict))
print("roc_auc_score:", roc_auc_score(y_test, y_test_predict),
      'average_precision_score:', average_precision_score(y_test, y_test_predict))

from sklearn.metrics import roc_curve
from sklearn.metrics import plot_roc_curve
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import plot_precision_recall_curve

disp = plot_precision_recall_curve(clf, X_test, y_test)
from sklearn.metrics import average_precision_score
average_precision = average_precision_score(y_test, y_test_predict)
print(average_precision)

disp = plot_roc_curve(clf, X_test, y_test)
roc_auc = roc_auc_score(y_test, y_test_predict)
print(roc_auc)
```

Смотрим, где неправильно предсказала модель

In [ ]:

```
Cls_result = pd.DataFrame(y_test)
Cls_result['proba'] = y_test_predict_proba[:,1]
Cls_result['predict'] = y_test_predict
Cls_result =  Cls_result.join(df_X[['msisdn_x', 'msisdn_y']])

alpha = .1

Cls_result['predict_alpha'] = np.where(Cls_result['proba']> alpha, 1, 0)
display(
Cls_result.groupby(['is_pair', 'predict_alpha']).agg({'msisdn_x':'count', 'proba':'mea
Cls_result.groupby(['is_pair', 'predict']).agg({'msisdn_x':'count', 'proba':'mean'}))
```

In [ ]:

```
a, b, is_pair, proba = Cls_result[(Cls_result['is_pair'] == 1) &
                            (Cls_result['predict_alpha'] == 0)].sample(n = 1).iloc[0][[
#a, b, is_pair, proba = 158503476701, 158503536671, 0, 0
print('msisdn_1, msisdn_2', a, b, 'is_pair: ', is_pair, 'proba:', proba )
fig = plt.figure(figsize=(15, 4))
show_chart(fig, df_data[df_data['msisdn'] == a], df_data[df_data['msisdn'] == b])
plt.show()


m = folium.Map(location=(55.7522200, 37.6155600))
show_circles_on_map(m, df_data[df_data['msisdn'] == a ], "lat", "long", "blue")
show_circles_on_map(m, df_data[df_data['msisdn'] == b ], "lat", "long", "red")

```

In [38]:

```
Cls_result_agg_copy = Cls_result_agg.copy()
```

In [43]:

```
Cls_result_agg_2_test[(Cls_result_agg_2_test['is_pair'] == 1) & (Cls_result_agg_2_test
```

Out[43]:

	is_pair	msisdn_x	predict	predict_alpha	proba
2	1	15	NaN	0.0	0.027333
2	1	16	NaN	0.0	0.028750
2	1	10	NaN	0.0	0.041000
2	1	14	NaN	0.0	0.027857
2	1	18	NaN	0.0	0.023889

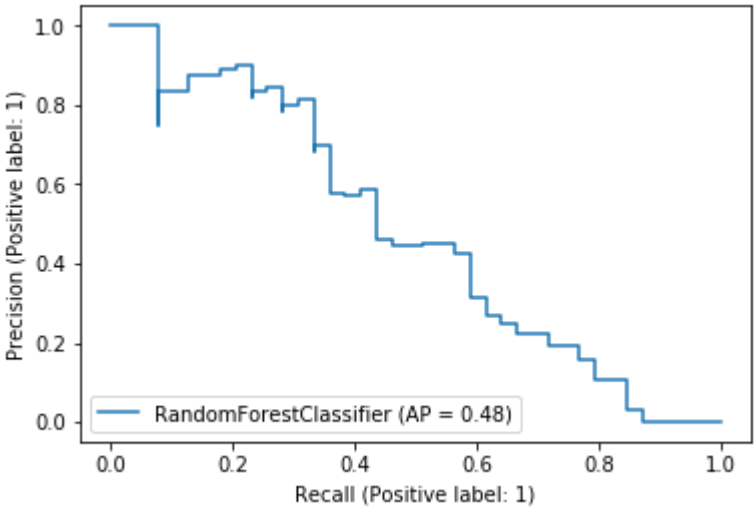
In [39]:

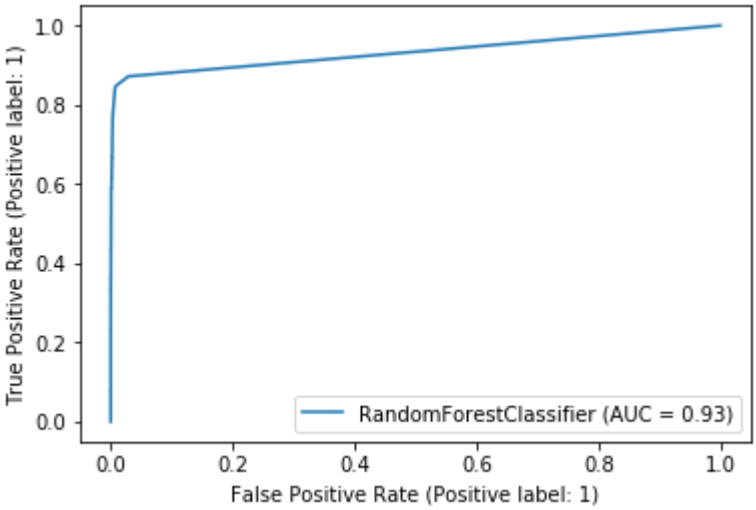
```
Cls_result_agg = pd.DataFrame()
BINS_QNT = 90
ln_ind = True
shift_inf = False
cls_ind = 'RF'
dist_ind_ln, coord_ind_ln, center_dist_ind_ln = True, False, False
dist_ind_shift, coord_ind_shift, center_dist_ind_shift = True, False, False
clfs = []

for bins_q, ln_ind, shift_ind, clf_ind, dist_ind_ln, coord_ind_ln, center_dist_ind_ln,
                            [30, True, False, 'RF', True, False, False, True, False, Fa
                            [60, True, False, 'RF', True, False, False, True, False, Fa
                            [90, True, False, 'RF', True, False, False, True, False, Fa
                            [120, True, False, 'RF', True, False, False, True, False, F
                            [150, True, False, 'RF', True, False, False, True, False, F
                                                                    ]:

    df_data_bin_ln, bins_list_ln = CreateTimeBeans(df_data, BINS_QNT)
    df_data_pivot_base_ln = CreatePivotBase(df_data_bin_ln)
    df_data_bin_agg_ln = AggData(df_data_bin_ln)
    df_data_agg_for_pivot_ln = LinearCombData(df_data_pivot_base_ln, df_data_bin_agg_l

    df_data_shift = AddDataDelta(df_data)
    df_data_bin_shift, bins_list_shift = CreateTimeBeans(df_data_shift, BINS_QNT)
    df_data_pivot_base_shift = CreatePivotBase(df_data_bin_shift)
    df_data_bin_agg_shift = AggData(df_data_bin_shift)
    df_data_agg_for_pivot_shift = CombData(df_data_pivot_base_shift, df_data_bin_agg_s

    df_msisdn_pairs = CreateMsisdnPairs(facts, df_data, 0.25)

    df_data_agg_pivot_ln = CreatePivot(df_data_agg_for_pivot_ln)
    df_data_agg_pivot_shift = CreatePivot(df_data_agg_for_pivot_shift)

    df_X_ln = CreateAndProcessPivot(df_msisdn_pairs, df_data_agg_pivot_ln, bins_list_l
    df_X_shift = CreateAndProcessPivot(df_msisdn_pairs, df_data_agg_pivot_shift, bins_

    df_X = pd.DataFrame()
    if (ln_ind):
        df_X = pd.concat([df_X_ln, df_X], axis = 1)
    if (shift_ind):
        df_X = pd.concat([df_X_shift, df_X], axis = 1)

    df_X.reset_index(inplace = True)

    X = df_X.drop(columns = ['msisdn_x', 'msisdn_y', 'is_pair']).fillna(0)
    y = df_X['is_pair']

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=42)
    if(clf_ind == 'RF'):
        clf = RandomForestClassifier(random_state=0)
    else:
        clf = XGBClassifier(random_state=42)

    clf.fit(X_train, y_train)

    y_test_predict = clf.predict(X_test)
    y_train_predict = clf.predict(X_train)
    y_test_predict_proba = clf.predict_proba(X_test)
```

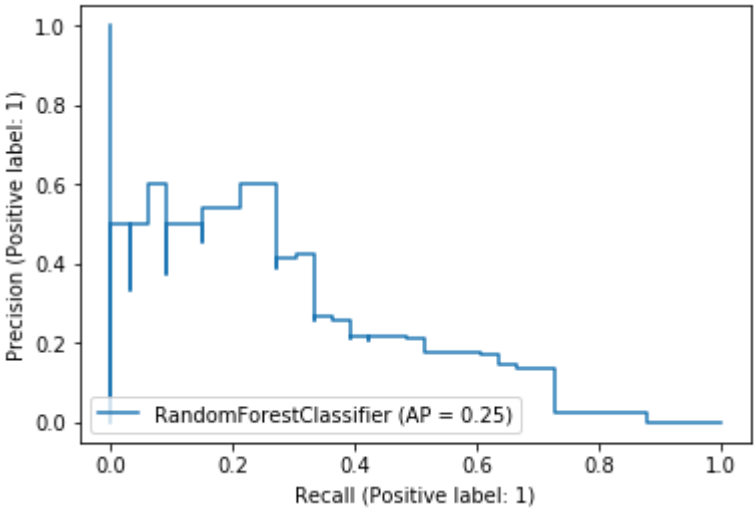
```
60
61      print('[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
62            [bins_q, ln_ind, shift_inf, cls_ind,
63           dist_ind_ln, coord_ind_ln, center_dist_ind_ln,
64           dist_ind_shift, coord_ind_shift, center_dist_ind_shift])
65
66      print('roc_auc_score:', roc_auc_score(y_train, y_train_predict),
67            'average_precision_score:', average_precision_score(y_train, y_train_predict
68      print("roc_auc_score:", roc_auc_score(y_test, y_test_predict),
69            'average_precision_score:', average_precision_score(y_test, y_test_predict))
70
71
72      disp = plot_precision_recall_curve(clf, X_test, y_test)
73      average_precision = average_precision_score(y_test, y_test_predict)
74      print(average_precision)
75
76      disp = plot_roc_curve(clf, X_test, y_test)
77      roc_auc = roc_auc_score(y_test, y_test_predict)
78      print(roc_auc)
79
80
81      Cls_result = pd.DataFrame(y_test)
82      Cls_result['proba'] = y_test_predict_proba[:,1]
83      Cls_result['predict'] = y_test_predict
84      Cls_result =  Cls_result.join(df_X[['msisdn_x', 'msisdn_y']])
85
86      alpha = .1
87
88      Cls_result['predict_alpha'] = np.where(Cls_result['proba']> alpha, 1, 0)
89      Cls_result_agg = Cls_result_agg.append(Cls_result.groupby(['is_pair', 'predict_alp
90      Cls_result_agg = Cls_result_agg.append(Cls_result.groupby(['is_pair', 'predict']).
91
```

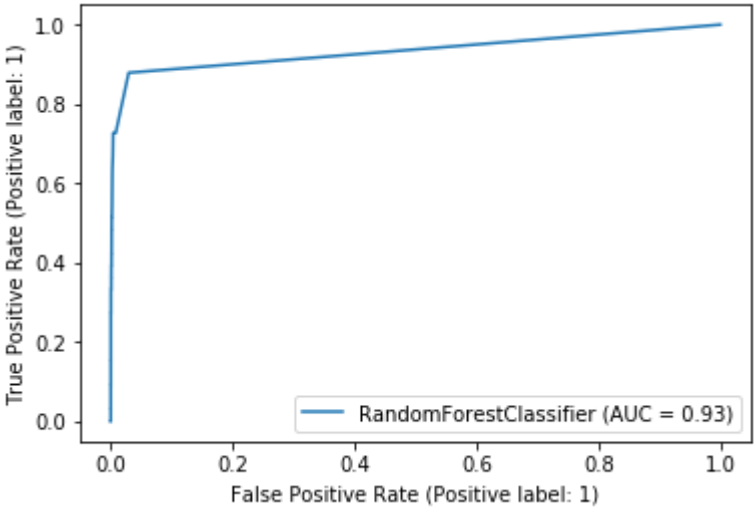
```
(129324, 3)
(127199, 91)
(127199, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [30, True, F
alse, 'RF', True, False, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.6153714992543985 average_precision_score: 0.208478471214320
28
0.20847847121432028
0.6153714992543985
(129326, 3)
(127214, 91)
(127214, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [60, True, F
alse, 'RF', True, False, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.6361800635114387 average_precision_score: 0.107348216010261
25
0.10734821601026125
0.6361800635114387
(129301, 3)
(127169, 91)
(127169, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
```

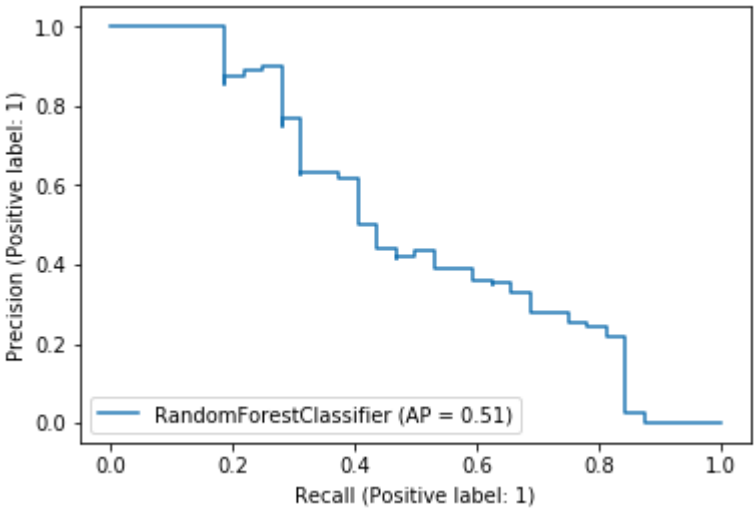
```
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, True, F
alse, 'RF', True, False, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.6249868831816154 average_precision_score: 0.222851301407564
68
0.22285130140756468
0.6249868831816154
(129289, 3)
(127148, 91)
(127148, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [120, True,
False, 'RF', True, False, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.6142069947895191 average_precision_score: 0.131320070301026
4
0.1313200703010264
0.6142069947895191
(129307, 3)
(127165, 91)
(127165, 91)
[bins_q, ln_ind, shift_inf, cls_ind, dist_ind_ln, coord_ind_ln, center_dist_
ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [150, True,
False, 'RF', True, False, False, True, False, False]
roc_auc_score: 1.0 average_precision_score: 1.0
roc_auc_score: 0.5415879554319497 average_precision_score: 0.028642784330857
724
0.028642784330857724
0.5415879554319497
```

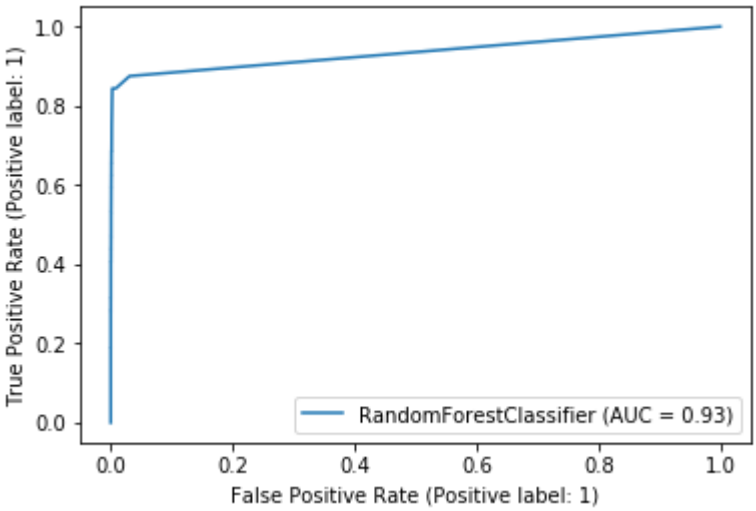


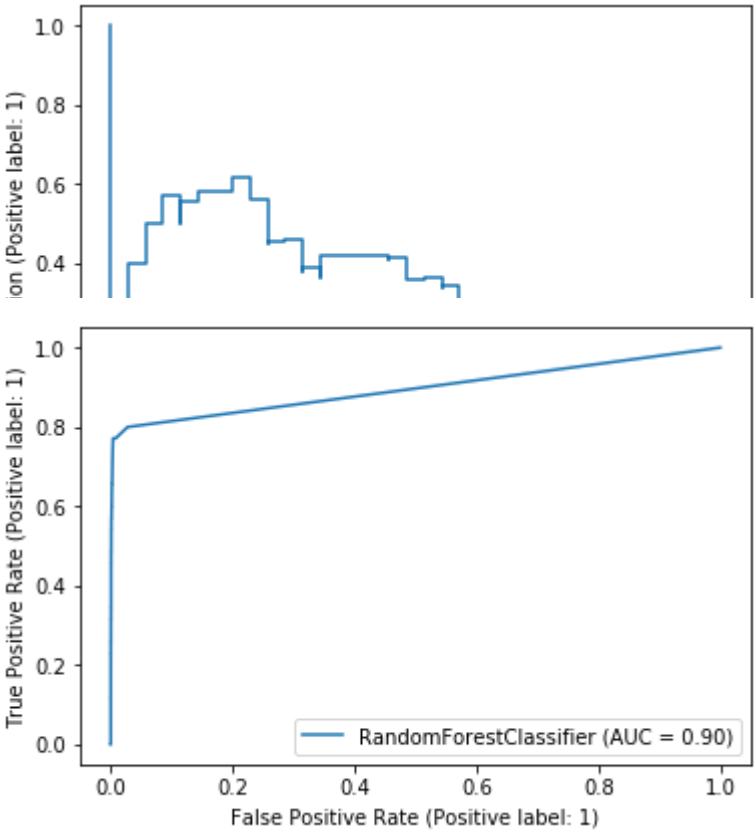

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.93)


Precision (Positive label: 1)
Recall (Positive label: 1)
RandomForestClassifier (AP = 0.25)

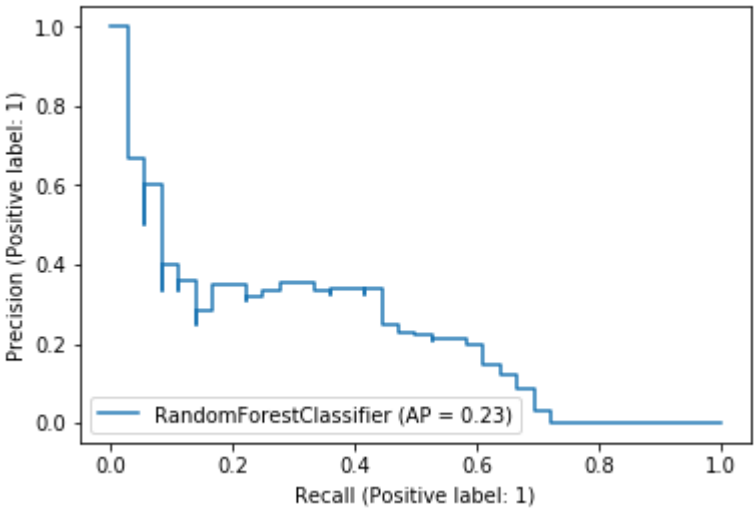

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.93)

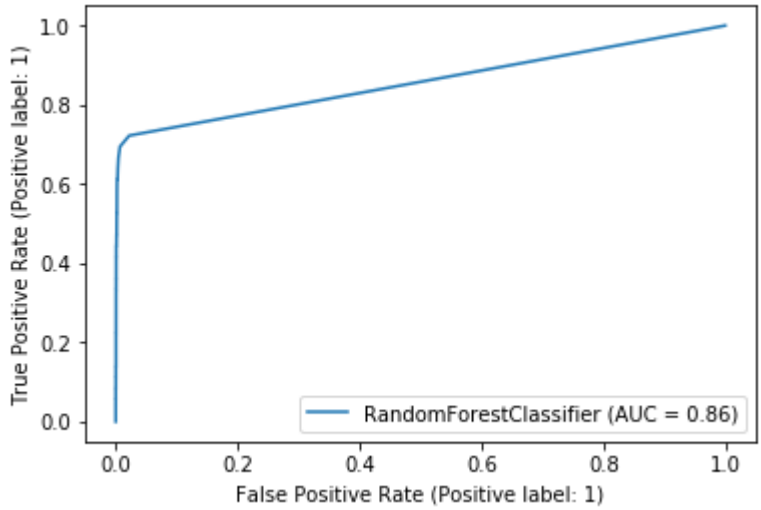

Precision (Positive label: 1)
Recall (Positive label: 1)
RandomForestClassifier (AP = 0.51)
0.0
0.2
0.4
0.6
0.8
1.0


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.93)
0.0
0.2
0.4
0.6
0.8
1.0


1.0
0.8
0.6
0.4
ion (Positive label: 1)

1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.90)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)


1.0
0.8
0.6
0.4
0.2
0.0
Precision (Positive label: 1)
RandomForestClassifier (AP = 0.23)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)



In [40]:

```
1 Cls_result_agg_2_test = Cls_result_agg.copy()
```

Использование модели

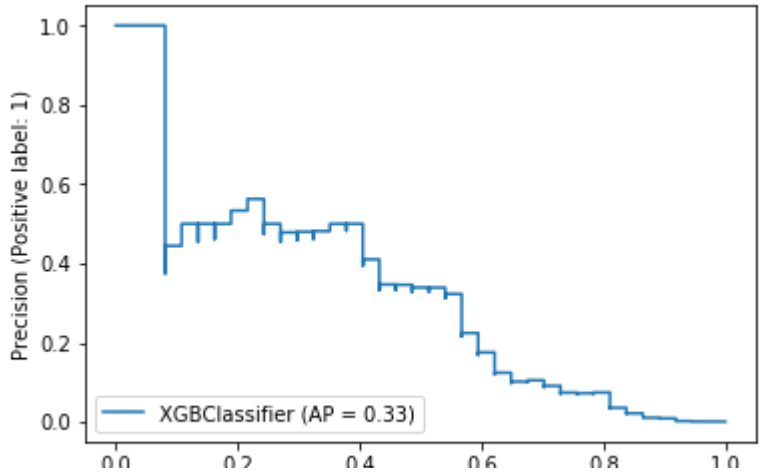
In [41]:

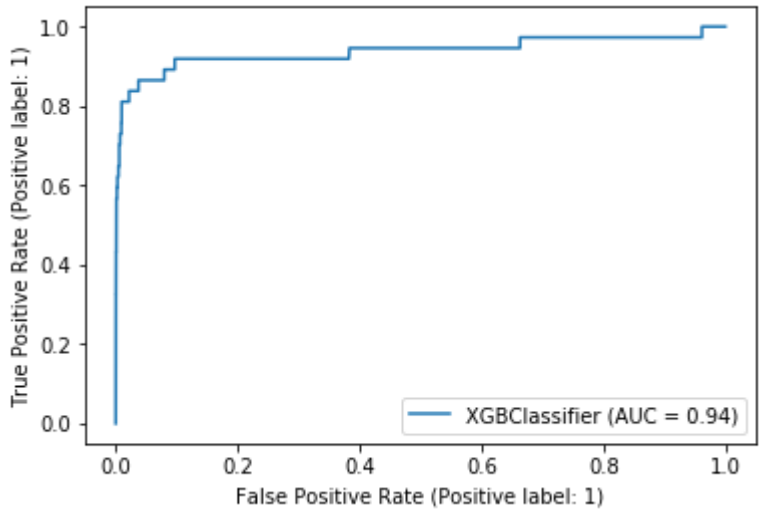
```
Cls_result_agg = pd.DataFrame()
BINS_QNT = 90
ln_ind = True
shift_inf = False
cls_ind = 'RF'
dist_ind_ln, coord_ind_ln, center_dist_ind_ln = True, False, False
dist_ind_shift, coord_ind_shift, center_dist_ind_shift = True, False, False
clfs = []

for bins_q, ln_ind, shift_ind, clf_ind, dist_ind_ln, coord_ind_ln, center_dist_ind_ln,
                            [30, True, False, 'GF', True, False, False, True, False, F
                            [60, True, False, 'GF', True, False, False, True, False, F
                            [90, True, False, 'GF', True, False, False, True, False, F
                            [120, True, False, 'GF', True, False, False, True, False,
                            [150, True, False, 'GF', True, False, False, True, False,
                                                                    ]:

    df_data_bin_ln, bins_list_ln = CreateTimeBeans(df_data, BINS_QNT)
    df_data_pivot_base_ln = CreatePivotBase(df_data_bin_ln)
    df_data_bin_agg_ln = AggData(df_data_bin_ln)
    df_data_agg_for_pivot_ln = LinearCombData(df_data_pivot_base_ln, df_data_bin_agg_l

    df_data_shift = AddDataDelta(df_data)
    df_data_bin_shift, bins_list_shift = CreateTimeBeans(df_data_shift, BINS_QNT)
    df_data_pivot_base_shift = CreatePivotBase(df_data_bin_shift)
    df_data_bin_agg_shift = AggData(df_data_bin_shift)
    df_data_agg_for_pivot_shift = CombData(df_data_pivot_base_shift, df_data_bin_agg_s

    df_msisdn_pairs = CreateMsisdnPairs(facts, df_data, 0.25)

    df_data_agg_pivot_ln = CreatePivot(df_data_agg_for_pivot_ln)
    df_data_agg_pivot_shift = CreatePivot(df_data_agg_for_pivot_shift)

    df_X_ln = CreateAndProcessPivot(df_msisdn_pairs, df_data_agg_pivot_ln, bins_list_l
    df_X_shift = CreateAndProcessPivot(df_msisdn_pairs, df_data_agg_pivot_shift, bins_

    df_X = pd.DataFrame()
    if (ln_ind):
        df_X = pd.concat([df_X_ln, df_X], axis = 1)
    if (shift_ind):
        df_X = pd.concat([df_X_shift, df_X], axis = 1)

    df_X.reset_index(inplace = True)

    X = df_X.drop(columns = ['msisdn_x', 'msisdn_y', 'is_pair']).fillna(0)
    y = df_X['is_pair']

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=42)
    if(clf_ind == 'RF'):
        clf = RandomForestClassifier(random_state=0)
    else:
        clf = XGBClassifier(random_state=42)

    clf.fit(X_train, y_train)

    y_test_predict = clf.predict(X_test)
    y_train_predict = clf.predict(X_train)
    y_test_predict_proba = clf.predict_proba(X_test)
```

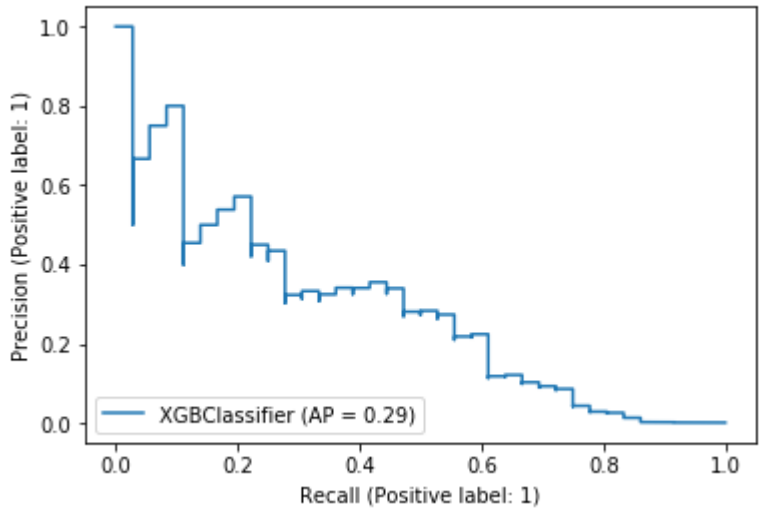
```
60
61      print('[bins_q, ln_ind, shift_inf, clf_ind, dist_ind_ln, coord_ind_ln, center_dist_
62            [bins_q, ln_ind, shift_inf, cls_ind,
63           dist_ind_ln, coord_ind_ln, center_dist_ind_ln,
64           dist_ind_shift, coord_ind_shift, center_dist_ind_shift])
65
66      print('roc_auc_score:', roc_auc_score(y_train, y_train_predict),
67            'average_precision_score:', average_precision_score(y_train, y_train_predict
68      print("roc_auc_score:", roc_auc_score(y_test, y_test_predict),
69            'average_precision_score:', average_precision_score(y_test, y_test_predict))
70
71
72      disp = plot_precision_recall_curve(clf, X_test, y_test)
73      average_precision = average_precision_score(y_test, y_test_predict)
74      print(average_precision)
75
76      disp = plot_roc_curve(clf, X_test, y_test)
77      roc_auc = roc_auc_score(y_test, y_test_predict)
78      print(roc_auc)
79
80
81      Cls_result = pd.DataFrame(y_test)
82      Cls_result['proba'] = y_test_predict_proba[:,1]
83      Cls_result['predict'] = y_test_predict
84      Cls_result =  Cls_result.join(df_X[['msisdn_x', 'msisdn_y']])
85
86      alpha = .1
87
88      Cls_result['predict_alpha'] = np.where(Cls_result['proba']> alpha, 1, 0)
89      Cls_result_agg = Cls_result_agg.append(Cls_result.groupby(['is_pair', 'predict_alp
90      Cls_result_agg = Cls_result_agg.append(Cls_result.groupby(['is_pair', 'predict']).
91
```

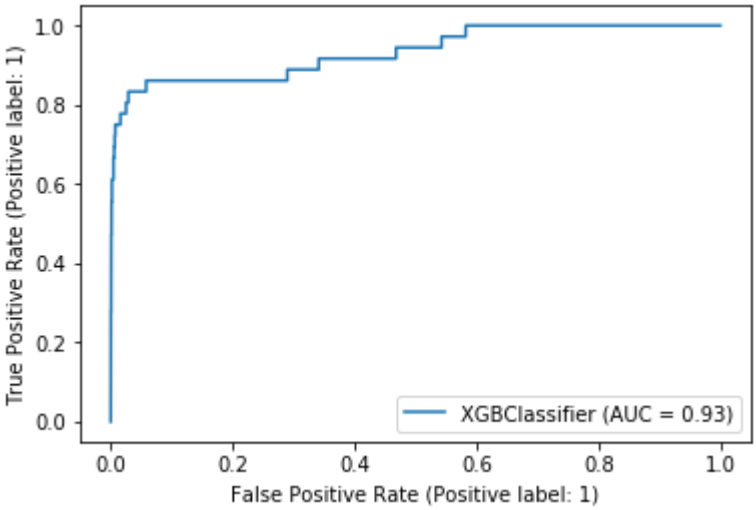
```
(129308, 3)
(127100, 91)
(127100, 91)
[12:26:08] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, clf_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [30, Tru
e, False, 'RF', True, False, False, True, False, False]
roc_auc_score: 0.9878048780487805 average_precision_score: 0.9756322355850
287
roc_auc_score: 0.6215297412236482 average_precision_score: 0.1375586542482
6874
0.13755865424826874
0.6215297412236482
(129315, 3)
(127199, 91)
(127199, 91)
[12:27:12] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, clf_ind, dist_ind_ln, coord_ind_ln, center_dis
```

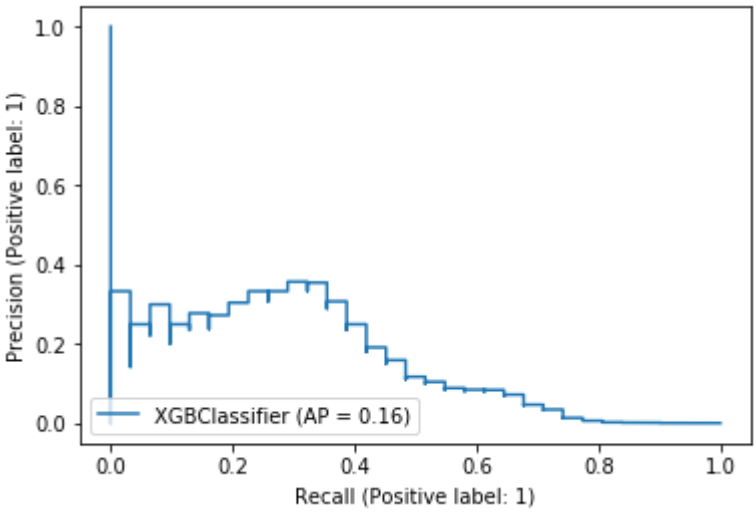
```
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [60, Tru
e, False, 'RF', True, False, False, True, False, False]
roc_auc_score: 0.9879518072289157 average_precision_score: 0.9759260765250
154
roc_auc_score: 0.5554899800650509 average_precision_score: 0.0502212904728
628
0.0502212904728628
0.5554899800650509
(129312, 3)
(127089, 91)
(127089, 91)
[12:28:19] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, clf_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [90, Tru
e, False, 'RF', True, False, False, True, False, False]
roc_auc_score: 0.9886363636363636 average_precision_score: 0.9772952087816
862
roc_auc_score: 0.5482558493991411 average_precision_score: 0.0230668939077
25525
0.023066893907725525
0.5482558493991411
(129314, 3)
(127161, 91)
(127161, 91)
[12:29:28] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, clf_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [120, Tr
ue, False, 'RF', True, False, False, True, False, False]
roc_auc_score: 0.9936708860759493 average_precision_score: 0.9873530065921
989
roc_auc_score: 0.6123950379175523 average_precision_score: 0.1199302502725
3819
0.11993025027253819
0.6123950379175523
(129322, 3)
(127138, 91)
(127138, 91)
[12:30:33] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[bins_q, ln_ind, shift_inf, clf_ind, dist_ind_ln, coord_ind_ln, center_dis
t_ind_ln, dist_ind_shift, coord_ind_shift, center_dist_ind_shift] [150, Tr
ue, False, 'RF', True, False, False, True, False, False]
roc_auc_score: 0.9941176470588236 average_precision_score: 0.9882465305777
127
roc_auc_score: 0.6028362116338085 average_precision_score: 0.0967863124485
2824
0.09678631244852824
0.6028362116338085
```

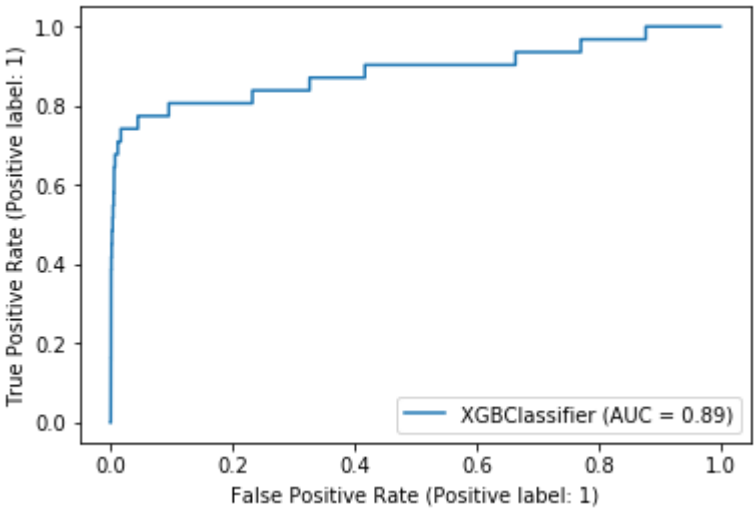

Precision (Positive label: 1)
XGBClassifier (AP = 0.33)

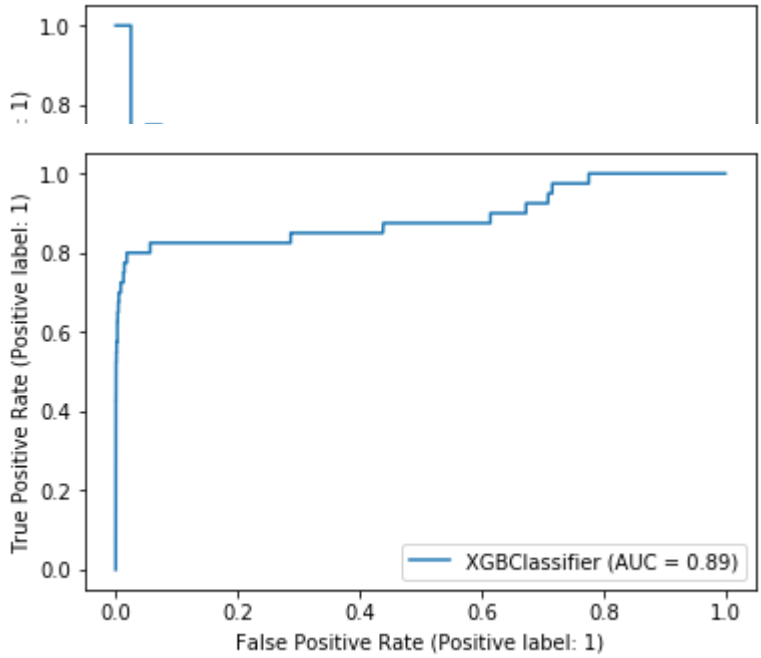

True Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.94)
False Positive Rate (Positive label: 1)

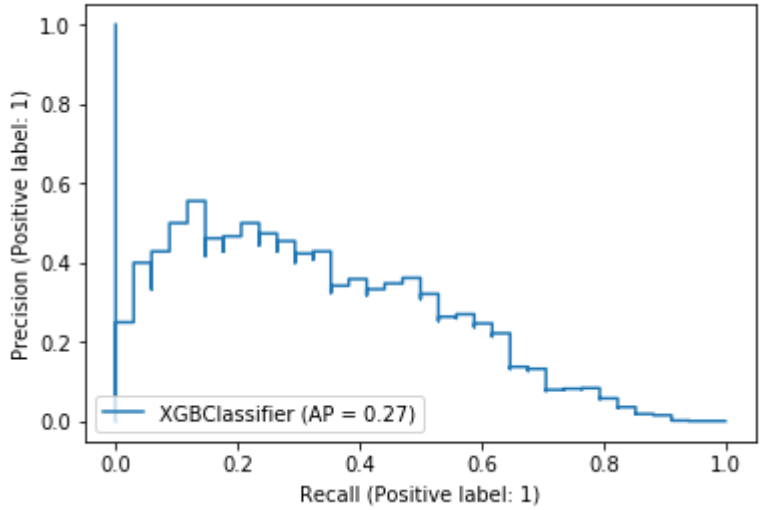

Precision (Positive label: 1)
XGBClassifier (AP = 0.29)
Recall (Positive label: 1)

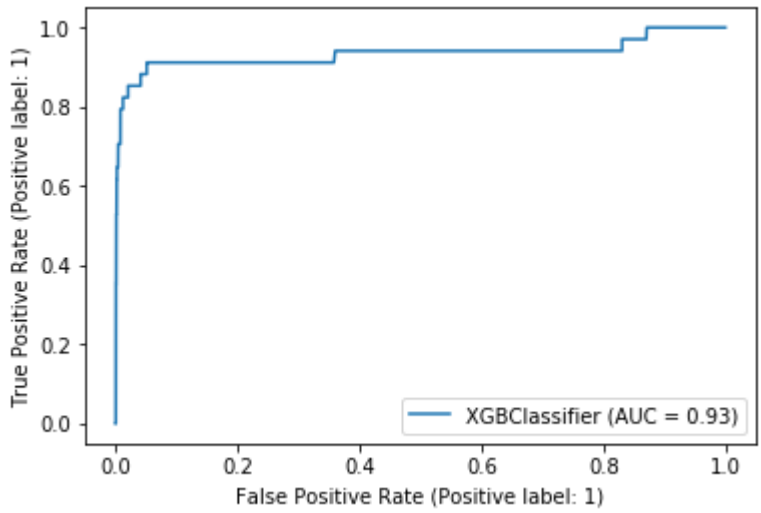

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.93)


Precision (Positive label: 1)
Recall (Positive label: 1)
XGBClassifier (AP = 0.16)


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.89)







Type *Markdown* and LaTeX: α^2