default_logreg

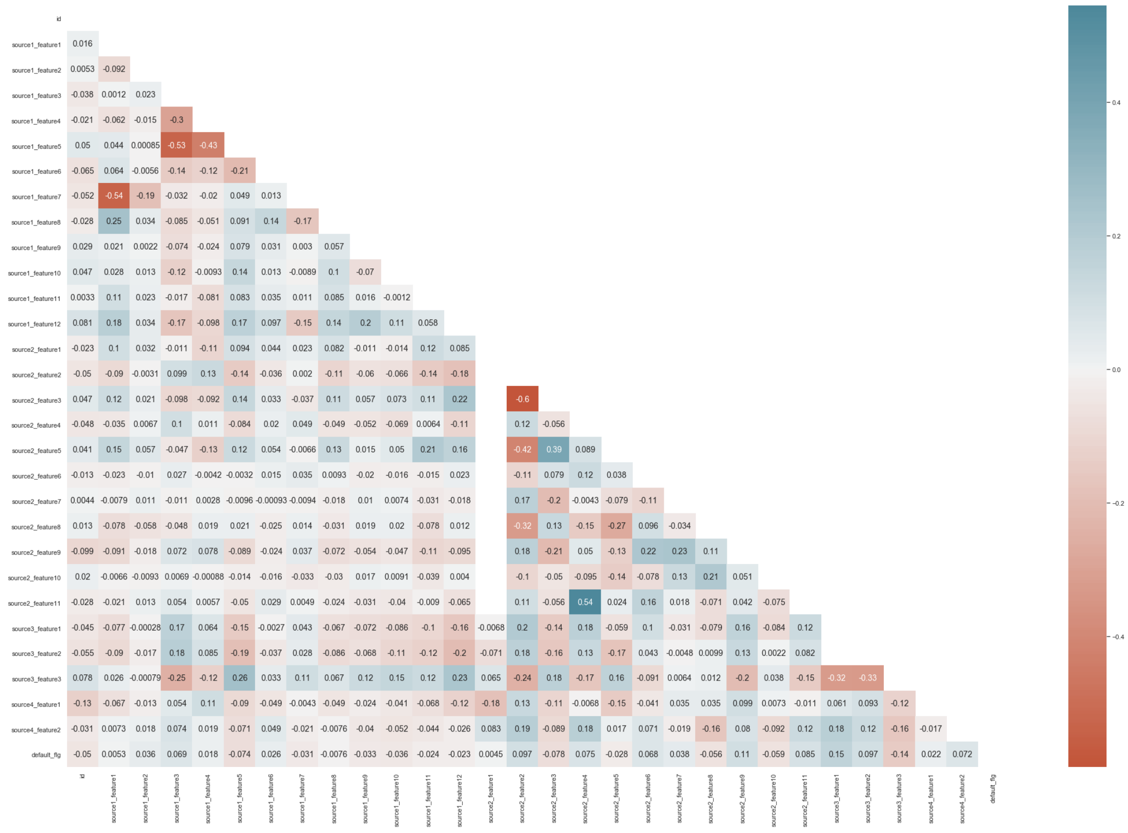
May 25, 2021

[1]:
```
import time
import pandas as pd
import numpy as np

from sklearn.metrics import roc_auc_score
from sklearn.metrics import log_loss
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression

import plotly.express as px

import matplotlib.pyplot as plt
import matplotlib.style as style
%matplotlib inline

import seaborn as sns
sns.set(font_scale=1.0)

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

[2]:
```
df_hw = pd.read_csv('data.csv')
df_hw;
```

[3]:
```
features = list(df_hw.columns)[1: -2]
```

[4]:
```
style.use('ggplot')
sns.set_style('whitegrid')
plt.subplots(figsize = (30,20))

mask = np.zeros_like(df_hw.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

sns_plot = sns.heatmap(df_hw.corr(),
                    cmap=sns.diverging_palette(20, 220, n=200),
                    mask = mask,
                    annot=True,
                    center = 0,
                    annot_kws={"size": 14},
                    cbar="coolwarm",
                   );

fig = sns_plot.get_figure()
plt.tight_layout()
```



[5]:
```
train_df = df_hw[df_hw.sample_part == 'train'].drop(columns=['sample_part'])
test_df = df_hw[df_hw.sample_part == 'test'].drop(columns=['sample_part'])
```

[6]:
```
binary = []

for feature in features:
    if len(train_df[feature].value_counts()) == 2:
        binary.append(feature)

non_binary = list(set(features) - set(binary))
```

[7]:
```
from math import log

def bad_rate(df,feature,target,num_buck = 10):
    return df.assign(bucket = np.ceil(df[feature].rank(pct = True) *␣
↪num_buck),obj_count = 1)\
                .groupby('bucket')\
                .agg({target:'sum','obj_count':sum,feature:'mean'})\
                .rename(columns = {target:'target_sum',feature:'average'})\
                .assign(bad_rate = lambda x:x.target_sum/x.obj_count)

def woe(df,feature,target,num_buck = 10):
    agg = bad_rate(df,feature,target,num_buck).reset_index()
    agg = agg[agg.target_sum != 0]
    return agg.assign(woe = lambda x:(x.bad_rate/(1-x.bad_rate) + 0.00001).
↪apply(log) -
                        log((df[target].sum()/(len(df) - df[target].sum())))).
↪set_index('bucket')

def IV(df,feature,target,num_buck = 10):
    B, G = df[target].sum(), len(df)
    agg = bad_rate(df, feature, target, num_buck).reset_index()
    agg = agg[agg.target_sum != 0]
    data = agg.assign(woe = lambda x:(x.bad_rate/(1-x.bad_rate) + 0.00001).
↪apply(log) -
                        log((B/(G - B)))).set_index('bucket')\
                 .assign(ivi=lambda x: (x.target_sum / B - x.obj_count / G) * x.
↪woe)
    return data.ivi.sum()
```

[8]:
```
filling_values = {}

for feature in binary:
    a, b = train_df[feature].value_counts().index
    iv_0 = IV(train_df.fillna({feature : a}), feature, 'default_flg')
    iv_1 = IV(train_df.fillna({feature : b}), feature, 'default_flg')

    if iv_1 > iv_0:
        filling_values[feature] = b
    else:
        filling_values[feature] = a

    print('feature: {},\t iv_0 : {:.4f},\t iv_1 : {:.4f}'.format(feature, iv_0,␣
↪iv_1))
```

```
feature: source1_feature1,       iv_0 : 0.0008,   iv_1 : 0.0008
feature: source1_feature2,       iv_0 : 0.0092,   iv_1 : 0.0092
feature: source1_feature3,       iv_0 : 0.0392,   iv_1 : 0.0392
feature: source1_feature4,       iv_0 : 0.0035,   iv_1 : 0.0035
feature: source1_feature5,       iv_0 : 0.0643,   iv_1 : 0.0643
feature: source1_feature6,       iv_0 : 0.0119,   iv_1 : 0.0119
```

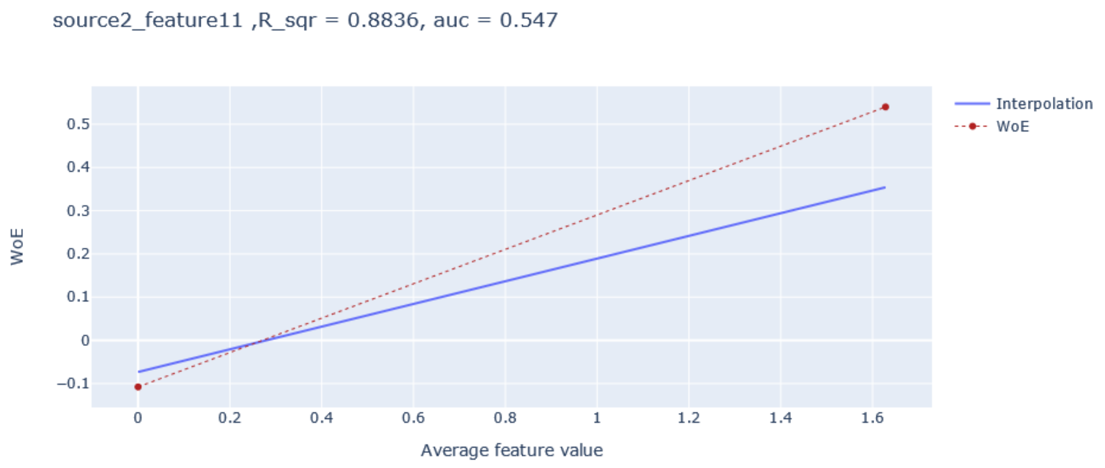
```
feature: source1_feature7,       iv_0 : 0.0119,  iv_1 : 0.0119
feature: source1_feature8,       iv_0 : 0.0002,  iv_1 : 0.0002
feature: source1_feature9,       iv_0 : 0.0161,  iv_1 : 0.0161
feature: source1_feature10,      iv_0 : 0.0151,  iv_1 : 0.0151
feature: source1_feature11,      iv_0 : 0.0101,  iv_1 : 0.0151
feature: source2_feature1,       iv_0 : 0.0001,  iv_1 : 0.0001
feature: source2_feature4,       iv_0 : 0.0430,  iv_1 : 0.0273
feature: source2_feature6,       iv_0 : 0.0404,  iv_1 : 0.0256
feature: source2_feature8,       iv_0 : 0.0291,  iv_1 : 0.0251
```

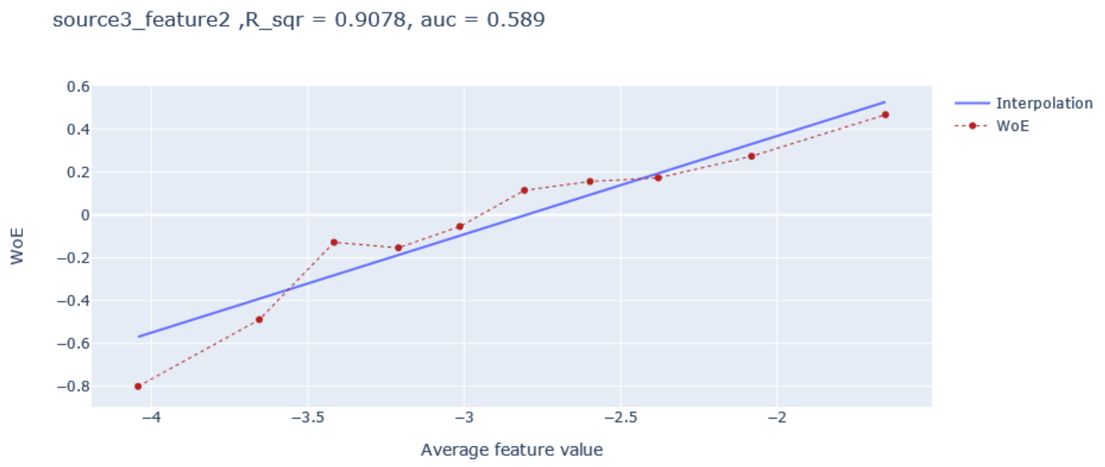
[9]:
```
for feature in sorted(non_binary):
    iv_none = IV(train_df[train_df[feature].notna()], feature, 'default_flg')
    iv_mean = IV(train_df.fillna({feature : train_df[feature].mean()}),␣
 ↪feature, 'default_flg')
    iv_median = IV(train_df.fillna({feature : train_df[feature].median()}),␣
 ↪feature, 'default_flg')
    iv_mode = IV(train_df.fillna({feature : train_df[feature].mode()[0]}),␣
 ↪feature, 'default_flg')
    iv_0 = IV(train_df.fillna({feature : 0.}), feature, 'default_flg')

    ind = np.argmax(np.array([iv_mean, iv_median, iv_mode, iv_0]))
    filling_values[feature] = ['mean', 'median', 'mode', 0][ind]

    print('{},\t iv_none : {:.4f}, iv_mean : {:.4f}, iv_median : {:.4f},␣
 ↪iv_mode : {:.4f}, iv_0 : {:.4f}'\
          .format(feature, iv_none, iv_mean, iv_median, iv_mode, iv_0))
```

```
source1_feature12,       iv_none : 0.0140, iv_mean : 0.0140, iv_median : 0.0140,
iv_mode : 0.0140, iv_0 : 0.0140
source2_feature10,       iv_none : 0.0394, iv_mean : 0.0335, iv_median : 0.0324,
iv_mode : 0.0324, iv_0 : 0.0238
source2_feature11,       iv_none : 0.0525, iv_mean : 0.0447, iv_median : 0.0440,
iv_mode : 0.0440, iv_0 : 0.0440
source2_feature2,        iv_none : 0.1175, iv_mean : 0.0968, iv_median : 0.0965,
iv_mode : 0.0972, iv_0 : 0.0915
source2_feature3,        iv_none : 0.1226, iv_mean : 0.1076, iv_median : 0.1074,
iv_mode : 0.0948, iv_0 : 0.0948
source2_feature5,        iv_none : 0.0103, iv_mean : 0.0078, iv_median : 0.0074,
iv_mode : 0.0053, iv_0 : 0.0078
source2_feature7,        iv_none : 0.0764, iv_mean : 0.0666, iv_median : 0.0654,
iv_mode : 0.0651, iv_0 : 0.0644
source2_feature9,        iv_none : 0.1112, iv_mean : 0.0546, iv_median : 0.0545,
iv_mode : 0.0546, iv_0 : 0.0572
source3_feature1,        iv_none : 0.3209, iv_mean : 0.0280, iv_median : 0.0310,
iv_mode : 0.0348, iv_0 : 0.0284
source3_feature2,        iv_none : 0.0988, iv_mean : 0.0796, iv_median : 0.0794,
iv_mode : 0.0810, iv_0 : 0.0793
source3_feature3,        iv_none : 0.2429, iv_mean : 0.1959, iv_median : 0.1961,
iv_mode : 0.1951, iv_0 : 0.1951
source4_feature1,        iv_none : 0.0039, iv_mean : 0.0039, iv_median : 0.0039,
iv_mode : 0.0039, iv_0 : 0.0039
source4_feature2,        iv_none : 0.0533, iv_mean : 0.0533, iv_median : 0.0533,
iv_mode : 0.0533, iv_0 : 0.0533
```

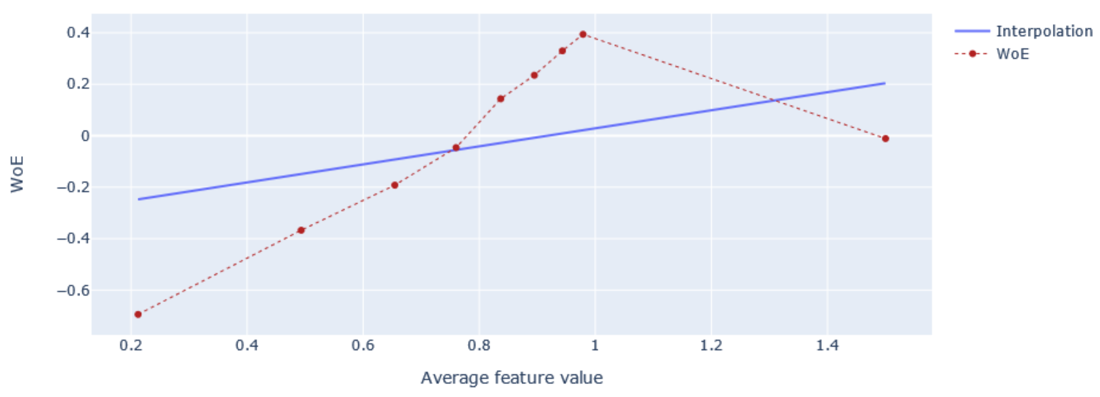
[10]:
```
import plotly.graph_objects as go
from sklearn.preprocessing import StandardScaler
from scipy.special import logit

def simple_reg(df, df_woe, feature, target):
    scaler = StandardScaler()
    scaler.fit(df[[feature]])
    clf = LogisticRegression(penalty='none', solver='lbfgs', max_iter=500)
    clf.fit(scaler.transform(df[[feature]]), df[target])
    df_woe['logreg'] = logit(clf.predict_proba(scaler.
 ↪transform(df_woe[['average']]))[:, 1]) - logit(
            np.clip(np.repeat(df[target].mean(), df_woe.shape[0]), 0.001, 0.
 ↪999))
    return df_woe

def woe_line(df, feature, target, num_buck = 10):
    woe_df = woe(df, feature, target, num_buck)
    simple_reg(df, woe_df, feature, target)

    n_obs = df[target].count()
    bad = df[target].sum()
    good = n_obs - df[target].sum()

    R_sqr = 1 - np.sum(woe_df['obj_count'] * (woe_df['woe'] - woe_df['logreg'])␣
 ↪** 2) / (
                np.sum(woe_df['obj_count'] * (woe_df['woe']) ** 2) - np.sum(
            woe_df['obj_count'] * woe_df['woe']) ** 2 / np.
 ↪sum(woe_df['obj_count']))

    auc_tmp = roc_auc_score(df[target], df[feature])
    auc = max(auc_tmp, 1 - auc_tmp)

    plot_nm = f"{feature} ,R_sqr = {round(R_sqr, 4)}, auc = {round(auc, 3)}"

    fig = go.Figure()
    fig.add_trace(go.Scatter(x=woe_df['average'], y=woe_df['logreg'],
                                 mode='lines',
                                 name='Interpolation'))
    fig.add_trace(go.Scatter(x=woe_df['average'], y=woe_df['woe'],
                     line=dict(color='firebrick', width=1, dash='dot'),
                         error_y=dict(
                                    type='data',
                                    symmetric=False),
                              name='WoE'))
    fig.update_layout(title= plot_nm,
                      width=1000,
                      height=450,
                      xaxis_title='Average feature value',
                      yaxis_title='WoE')

    fig.show()
```

[11]:
```
for feature in non_binary:
    woe_line(train_df[train_df[feature].notna()], feature, 'default_flg')
```

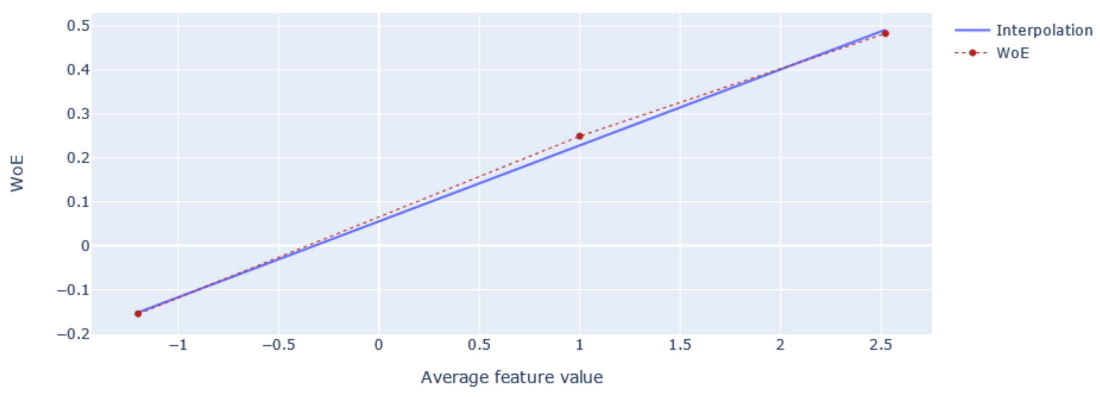


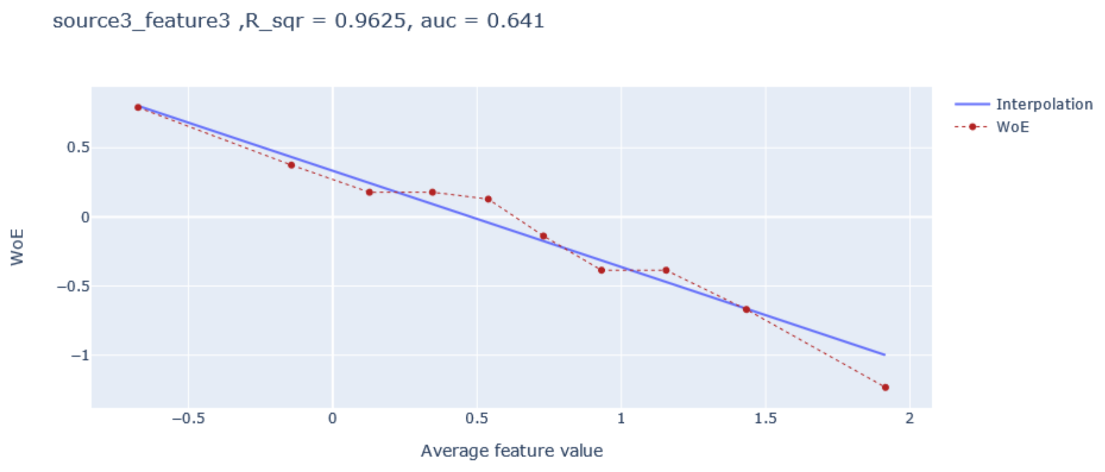


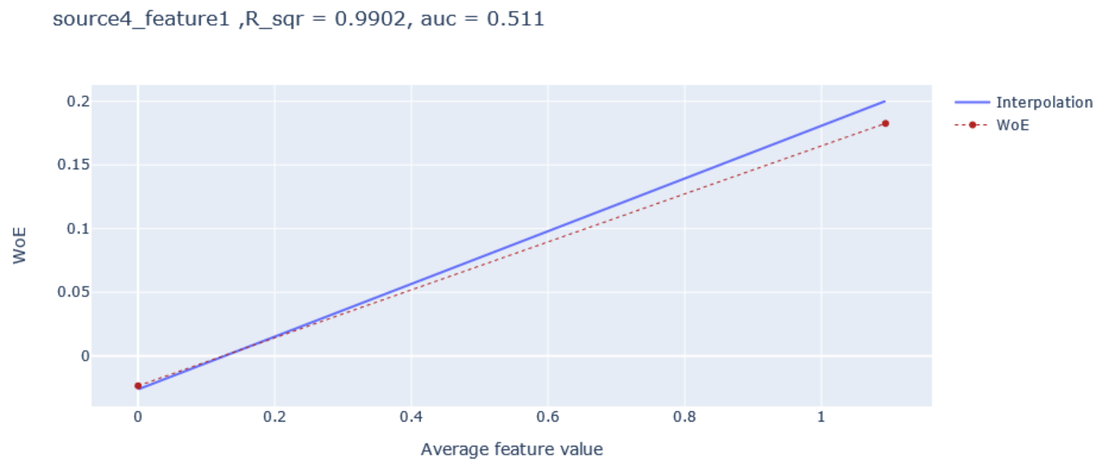
source2_feature7 ,R_sqr = 0.3024, auc = 0.553



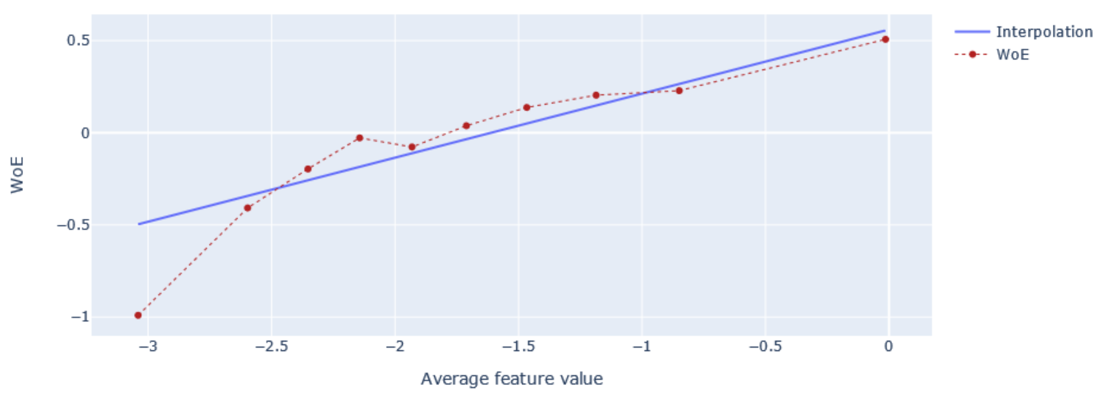
source4_feature2 ,R_sqr = 0.9985, auc = 0.557



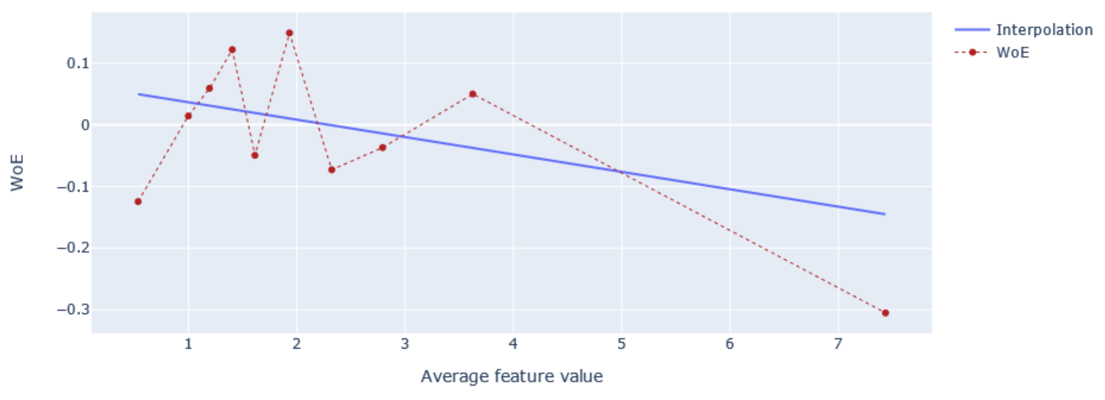

source3_feature3 ,R_sqr = 0.9625, auc = 0.641
Interpolation
WoE
WoE
0.5
0
−0.5
−1
−0.5
0
0.5
1
1.5
2
Average feature value

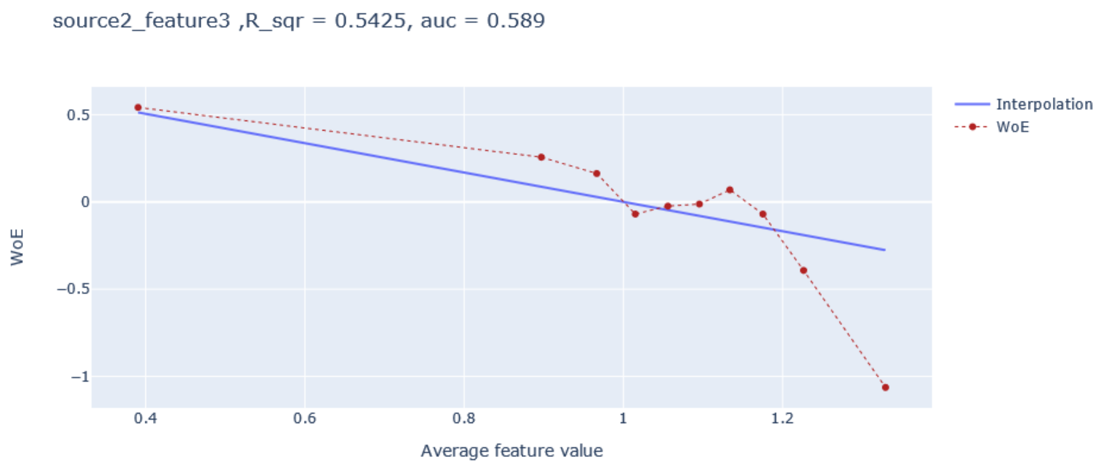

source4_feature1 ,R_sqr = 0.9902, auc = 0.511
Interpolation
WoE
WoE
0.2
0.15
0.1
0.05
0
0
0.2
0.4
0.6
0.8
1
Average feature value

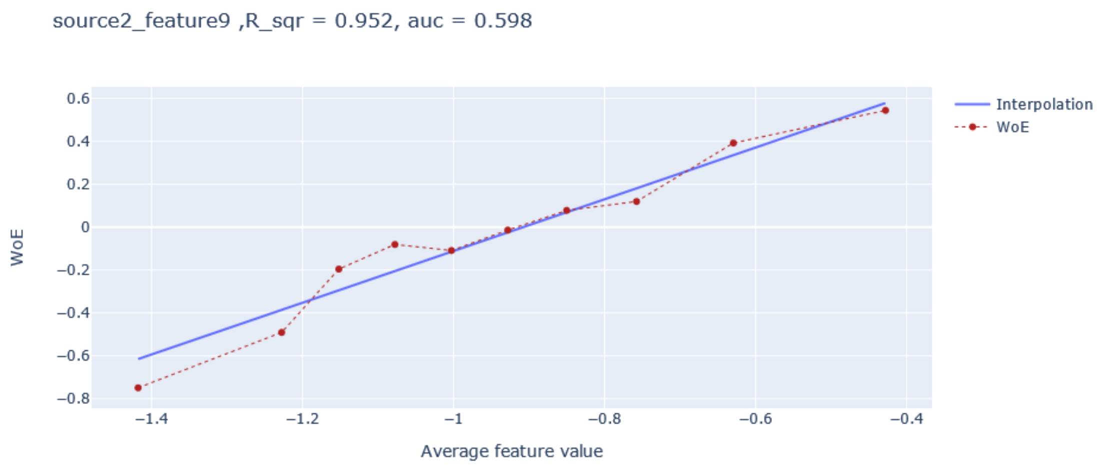
source2_feature2 ,R_sqr = 0.8072, auc = 0.594

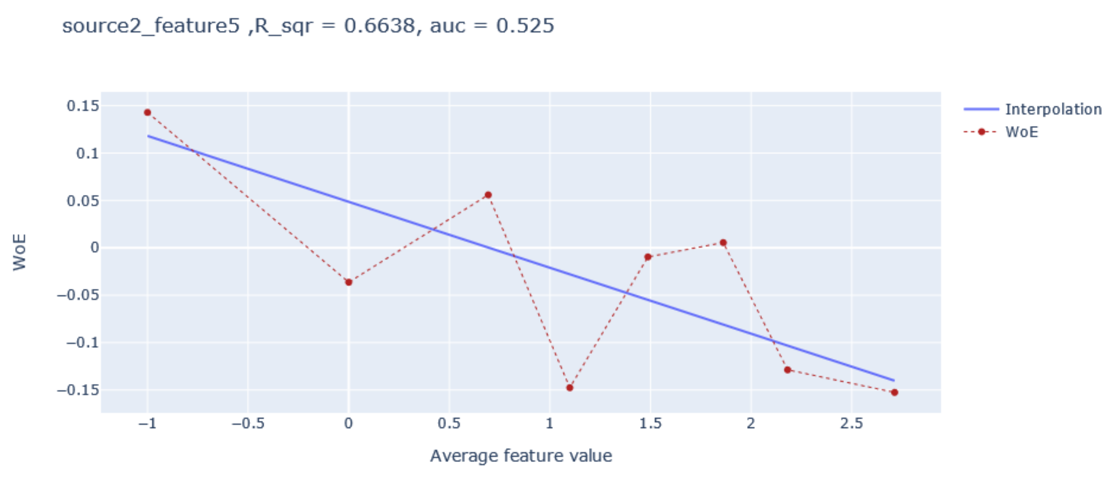


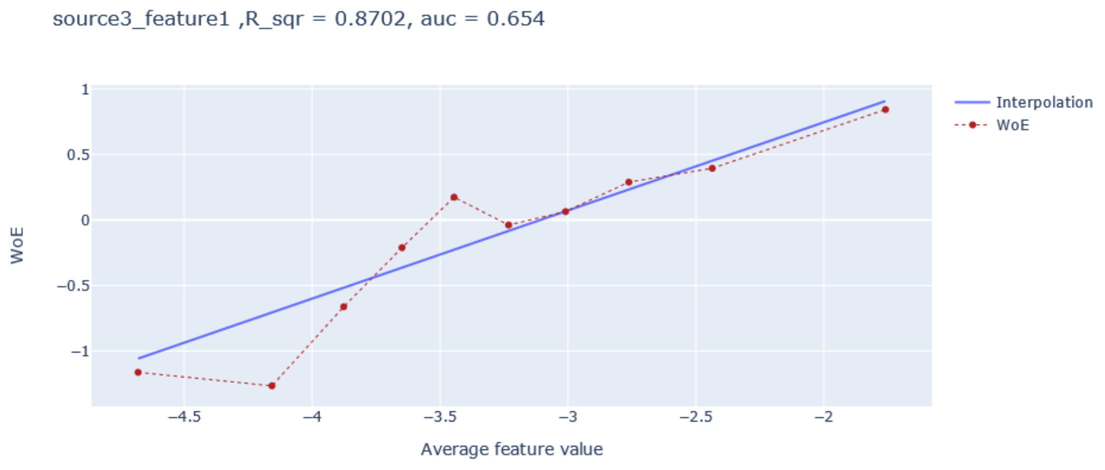
source1_feature12 ,R_sqr = 0.3215, auc = 0.507



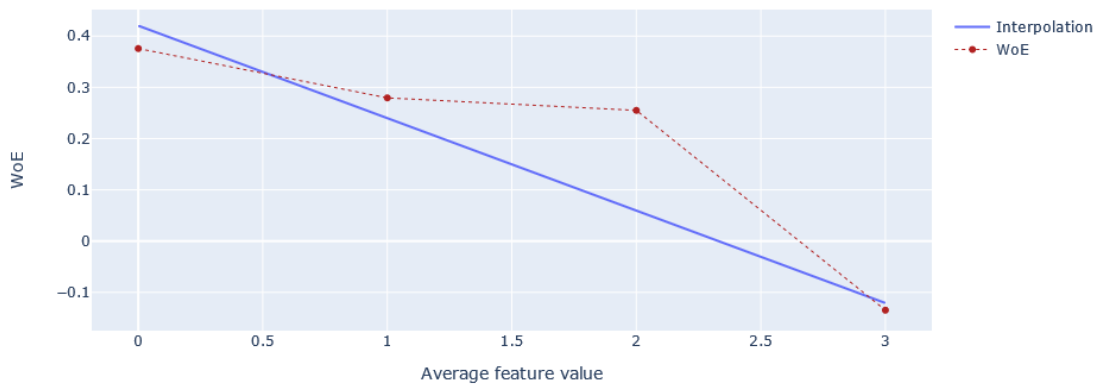

source2_feature3 ,R_sqr = 0.5425, auc = 0.589
Interpolation
WoE
0.5
0
−0.5
−1
0.4
0.6
0.8
1
1.2
WoE
Average feature value


source2_feature9 ,R_sqr = 0.952, auc = 0.598
Interpolation
WoE
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1.4
−1.2
−1
−0.8
−0.6
−0.4
WoE
Average feature value


source2_feature5 ,R_sqr = 0.6638, auc = 0.525
Interpolation
WoE
0.15
0.1
0.05
0
−0.05
−0.1
−0.15
−1
−0.5
0
0.5
1
1.5
2
2.5
Average feature value

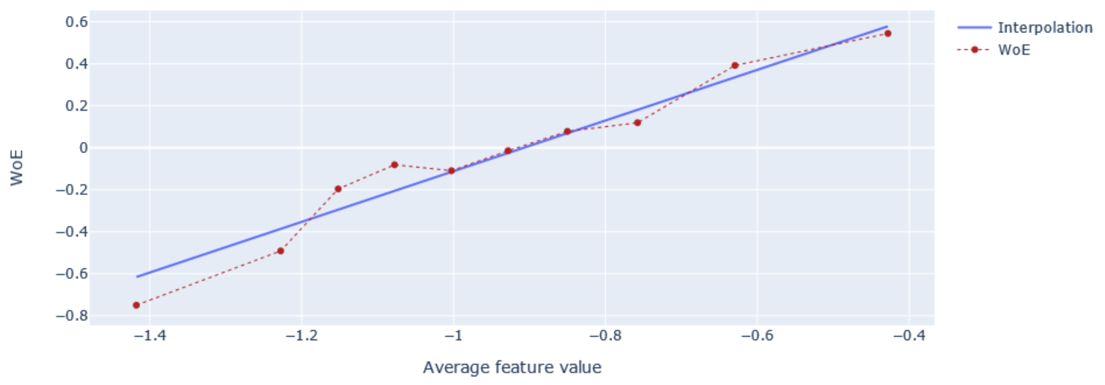

source3_feature1 ,R_sqr = 0.8702, auc = 0.654
Interpolation
WoE
1
0.5
0
−0.5
−1
−4.5
−4
−3.5
−3
−2.5
−2
Average feature value

source2_feature10 ,R_sqr = 0.9372, auc = 0.548



[12]:
```
ftre = 'source2_feature9'
ftre_df = train_df.copy()
ftre_df[ftre] = ftre_df[ftre]
woe_line(ftre_df[ftre_df[ftre].notna()], ftre, 'default_flg')
```

source2_feature9 ,R_sqr = 0.952, auc = 0.598



[13]:
```
new = [
    'flg_source2_feature3',
    'clip_source2_feature3',
    'flg_source2_feature7',
    'clip01_source2_feature7',
    'pow3_source2_feature2',
    'flg_source2_feature5',
    'flg_source1_feature12',
    'flg_source2_feature11',
    'flg_source4_feature2',
]
train_df['flg_source2_feature3'] = (train_df['source2_feature3'] > 0).
 ↪astype(int)
train_df['clip_source2_feature3'] = np.clip(train_df['source2_feature3'], 0, 1.
 ↪09)
train_df['flg_source2_feature7'] = (train_df['source2_feature7'] < 1.5).
 ↪astype(int)
train_df['clip01_source2_feature7'] = np.
 ↪clip(train_df[train_df['source2_feature7'].notna()]['source2_feature7'], 0,␣
 ↪1)
train_df['pow3_source2_feature2'] = np.
 ↪power(train_df[train_df['source2_feature2'].notna()]['source2_feature2'], 3)
train_df['flg_source2_feature5'] = (train_df['source2_feature5'] > -1).
 ↪astype(int)
train_df['flg_source1_feature12'] = (train_df['source1_feature12'] > 14.85).
 ↪astype(int)
train_df['flg_source2_feature11'] = (train_df['source2_feature11'] > 0).
 ↪astype(int)
train_df['flg_source4_feature2'] = (train_df['source4_feature2'] > -1).
 ↪astype(int)
```

[14]:
```
test_df['flg_source2_feature3'] = (test_df['source2_feature3'] > 0).astype(int)
test_df['clip_source2_feature3'] = np.clip(test_df['source2_feature3'], 0, 1.09)
test_df['flg_source2_feature7'] = (test_df['source2_feature7'] < 1.5).
 ↪astype(int)
test_df['clip01_source2_feature7'] = np.
 ↪clip(test_df[test_df['source2_feature7'].notna()]['source2_feature7'], 0, 1)
test_df['pow3_source2_feature2'] = np.power(test_df[test_df['source2_feature2'].
 ↪notna()]['source2_feature2'], 3)
test_df['flg_source2_feature5'] = (test_df['source2_feature5'] > -1).astype(int)
test_df['flg_source1_feature12'] = (test_df['source1_feature12'] > 14.85).
 ↪astype(int)
test_df['flg_source2_feature11'] = (test_df['source2_feature11'] > 0).
 ↪astype(int)
test_df['flg_source4_feature2'] = (test_df['source4_feature2'] > -1).astype(int)
```

[15]:
```
filling_values['flg_source2_feature3'] = 0
filling_values['clip_source2_feature3'] = filling_values['source2_feature3']
filling_values['flg_source2_feature7'] = 0
filling_values['clip01_source2_feature7'] = filling_values['source2_feature7']
filling_values['pow3_source2_feature2'] = filling_values['source2_feature2']
filling_values['flg_source2_feature5'] = 0
```

```
filling_values['flg_source1_feature12'] = 0
filling_values['flg_source2_feature11'] = 0
filling_values['flg_source4_feature2'] = 0
```

[16]:
```
not_worthy = [
    'source2_feature3', 'source2_feature7', 'source2_feature2',␣
 ↪'source2_feature5',
    'source1_feature12', 'source2_feature11', 'source1_feature4',
]
```

[17]:
```
features += new
features = list(set(features) - set(not_worthy))
```

[18]:
```
filling_values
```

[18]:
```
{'source1_feature1': 0,
 'source1_feature2': 0,
 'source1_feature3': 0,
 'source1_feature4': 0,
 'source1_feature5': 0,
 'source1_feature6': 0,
 'source1_feature7': 1,
 'source1_feature8': 0,
 'source1_feature9': 0,
 'source1_feature10': 0,
 'source1_feature11': 1.0,
 'source2_feature1': 1,
 'source2_feature4': 0.0,
 'source2_feature6': 2.0,
 'source2_feature8': 1.0,
 'source1_feature12': 'mean',
 'source2_feature10': 'mean',
 'source2_feature11': 'mean',
 'source2_feature2': 'mode',
 'source2_feature3': 'mean',
 'source2_feature5': 'mean',
 'source2_feature7': 'mean',
 'source2_feature9': 0,
 'source3_feature1': 'mode',
 'source3_feature2': 'mode',
 'source3_feature3': 'median',
 'source4_feature1': 'mean',
 'source4_feature2': 'mean',
 'flg_source2_feature3': 0,
 'clip_source2_feature3': 'mean',
 'flg_source2_feature7': 0,
 'clip01_source2_feature7': 'mean',
 'pow3_source2_feature2': 'mode',
 'flg_source2_feature5': 0,
 'flg_source1_feature12': 0,
 'flg_source2_feature11': 0,
 'flg_source4_feature2': 0}
```

[19]:
```
filling_values['source2_feature9'] = np.nan
filling_values['source3_feature1'] = np.nan
filling_values['source3_feature2'] = np.nan
filling_values['source3_feature3'] = np.nan
```

[20]:
```
def kind(df, feature):
    k = filling_values[feature]
    if k == 'mean':
        return df[feature].mean()
    if k == 'median':
        return df[feature].median()
    if k == 'mode':
        return df[feature].mode()[0]
    else:
        return k

train_df = train_df.fillna({feature : kind(train_df, feature) for feature in␣
 ↪features})
test_df = test_df.fillna({feature : kind(test_df, feature) for feature in␣
 ↪features})
```

[21]:
```
train_df = train_df[['id'] + features + ['default_flg']]
test_df = test_df[['id'] + features + ['default_flg']]
```

[22]:
```
train_df.isna().any()
```

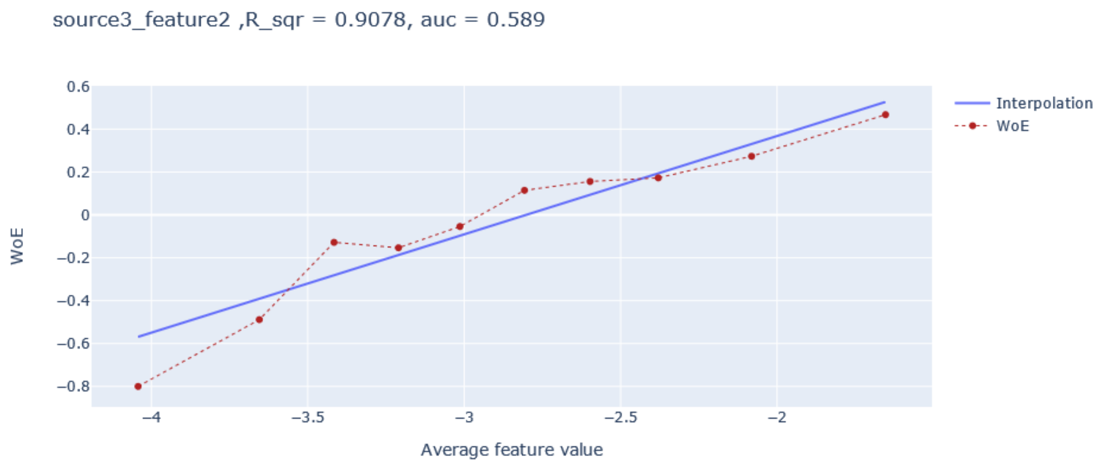
[22]:
```
id                          False
source3_feature2             True
source4_feature2            False
source2_feature6            False
flg_source2_feature7        False
source1_feature1            False
source2_feature8            False
flg_source2_feature3        False
clip_source2_feature3       False
flg_source4_feature2        False
source1_feature11           False
source2_feature1            False
flg_source2_feature11       False
source1_feature3            False
flg_source1_feature12       False
```

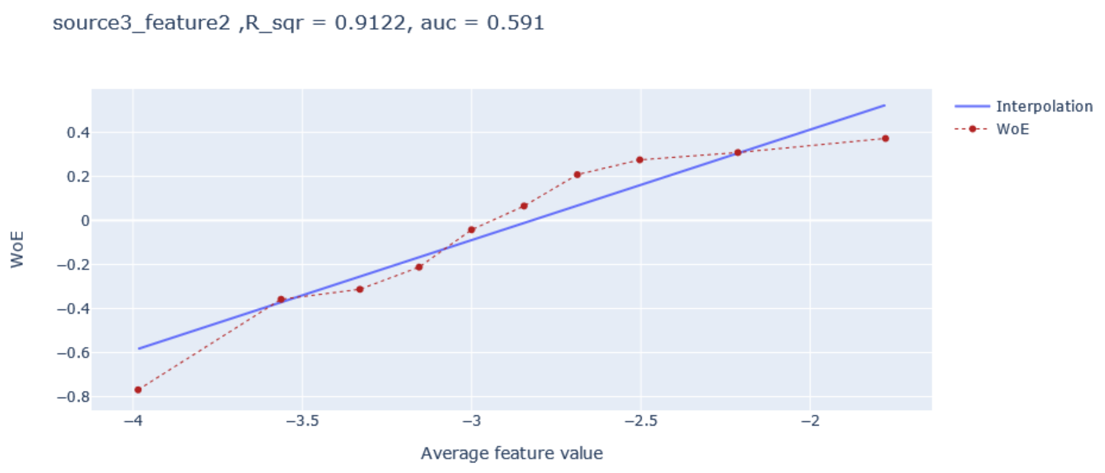
```
source2_feature9           True
source1_feature5          False
source1_feature9          False
flg_source2_feature5      False
source2_feature4          False
source1_feature7          False
pow3_source2_feature2     False
source1_feature8          False
clip01_source2_feature7   False
source2_feature10         False
source1_feature6          False
source3_feature3           True
source4_feature1          False
source1_feature2          False
source1_feature10         False
source3_feature1           True
default_flg               False
dtype: bool
```

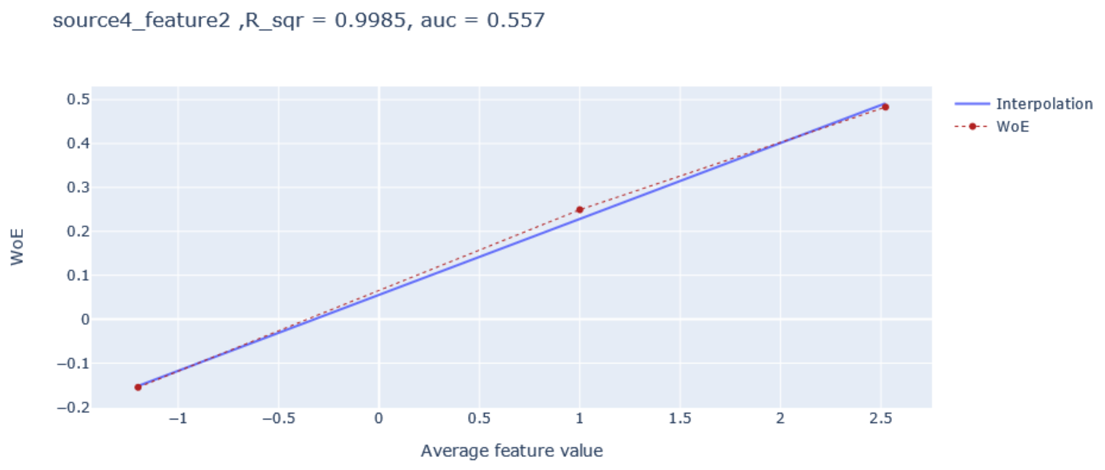
[23]:
```python
from catboost import CatBoostRegressor

#
#
#
#
#       catboost,  . .
#                      ,

filling_model = CatBoostRegressor(
    random_seed=63,
    iterations=1000,
    learning_rate=0.007,
    bagging_temperature=1,
    depth=6,
)

all_data = pd.concat((train_df, test_df), axis=0)[features]

features_to_fill = ['source2_feature9', 'source3_feature2',
                    'source3_feature3', 'source3_feature1']

for feature in features_to_fill:
    start = time.time()

    target = all_data[feature]
    temp_features = list(set(features) - set([feature]))
    mask = target.notna()
    X_train, X_test = all_data[temp_features][mask],␣
↪all_data[temp_features][~mask]
    y_train, _ = target[mask], target[~mask]

    filling_model.fit(X_train, y_train,
#                         cat_features=cat_features,
                      verbose=0)

    y_test = filling_model.predict(X_test)
    length = len(train_df[~mask])
    train_df.loc[~mask, feature] = y_test[:length]
    test_df.loc[~mask, feature] = y_test[length:]

    end = time.time()

    print('{} is done, {:.1f}s'.format(feature, end-start))
```

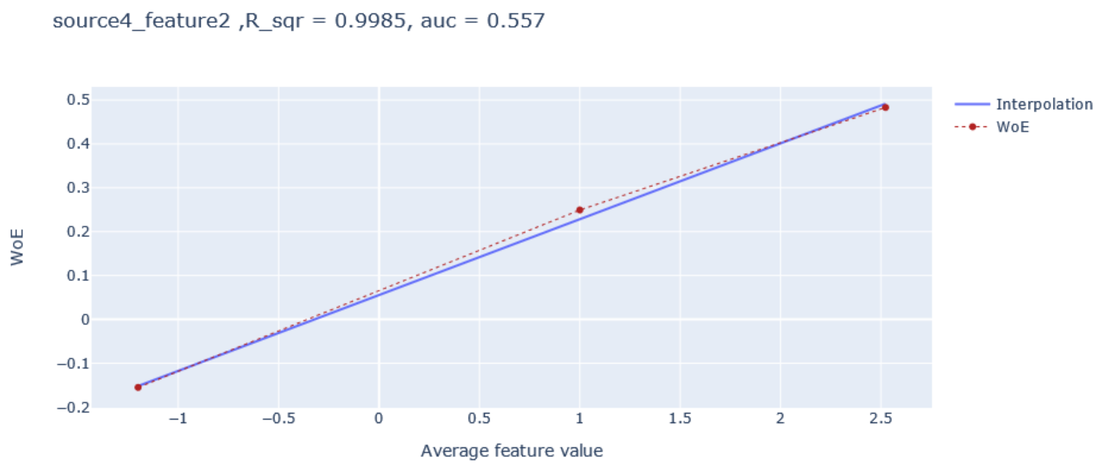
```
source2_feature9 is done, 13.0s
source3_feature2 is done, 10.7s
source3_feature3 is done, 12.4s
source3_feature1 is done, 4.6s
```

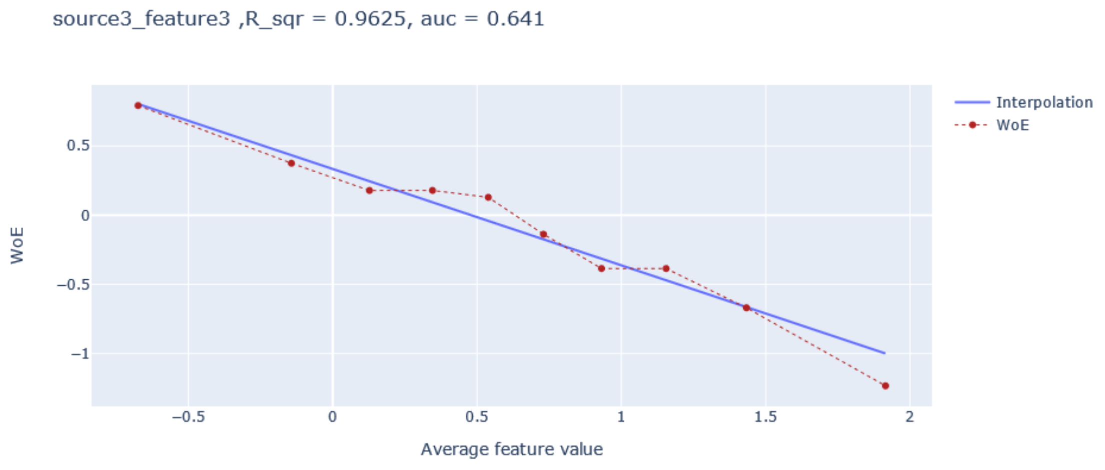
[24]:
```
#   -                  nan
for feature in (list(set(non_binary) - set(not_worthy)) + new):
    f1 = feature
    f2 = feature
    if feature in new:
        f2 = feature[feature.find('_')+1:]
    woe_line(ftre_df[ftre_df[f2].notna()], f2, 'default_flg')
    woe_line(train_df[train_df[f1].notna()], f1, 'default_flg')
```



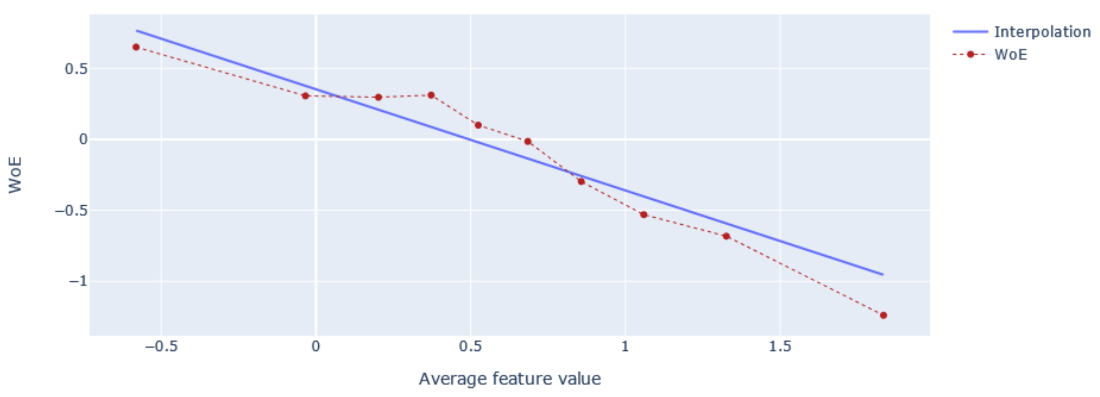

source3_feature2 ,R_sqr = 0.9122, auc = 0.591
Interpolation
WoE
WoE
Average feature value
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−4
−3.5
−3
−2.5
−2


source4_feature2 ,R_sqr = 0.9985, auc = 0.557
Interpolation
WoE
WoE
Average feature value
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−1
−0.5
0
0.5
1
1.5
2
2.5

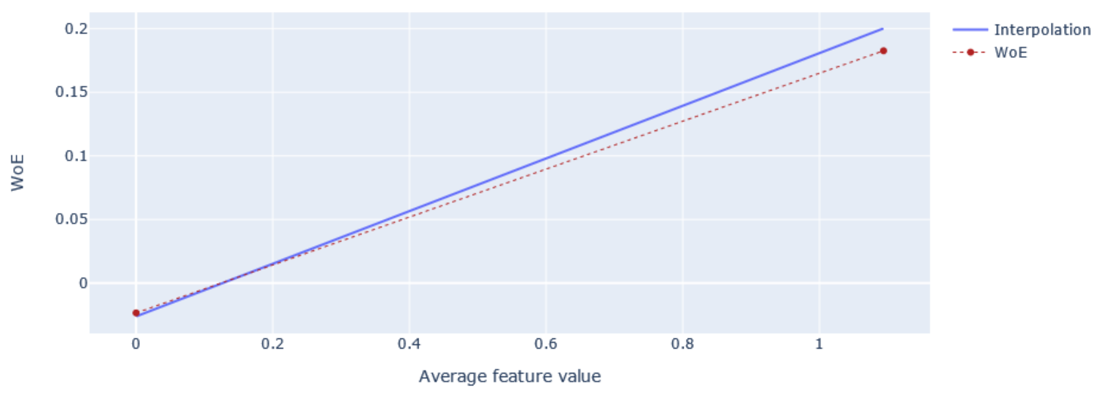

source4_feature2 ,R_sqr = 0.9985, auc = 0.557
WoE
Average feature value
Interpolation
WoE
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−1
−0.5
0
0.5
1
1.5
2
2.5


source3_feature3 ,R_sqr = 0.9625, auc = 0.641
WoE
Average feature value
Interpolation
WoE
0.5
0
−0.5
−1
−0.5
0
0.5
1
1.5
2

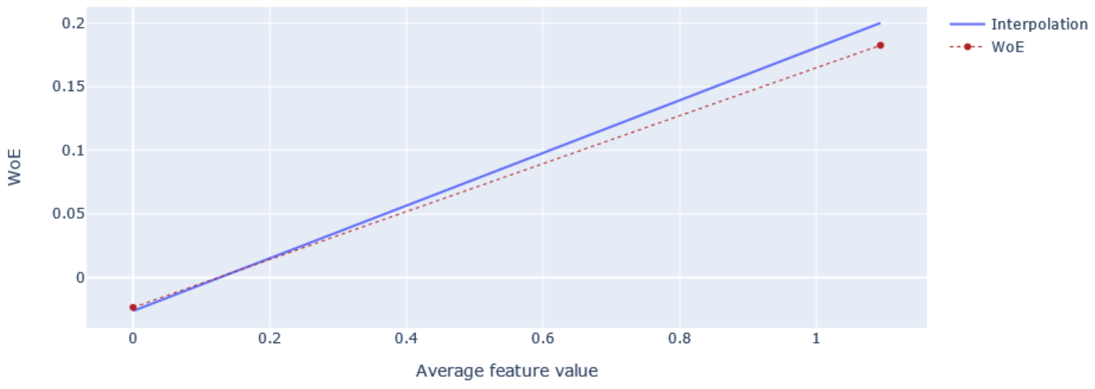
source3_feature3 ,R_sqr = 0.9281, auc = 0.634



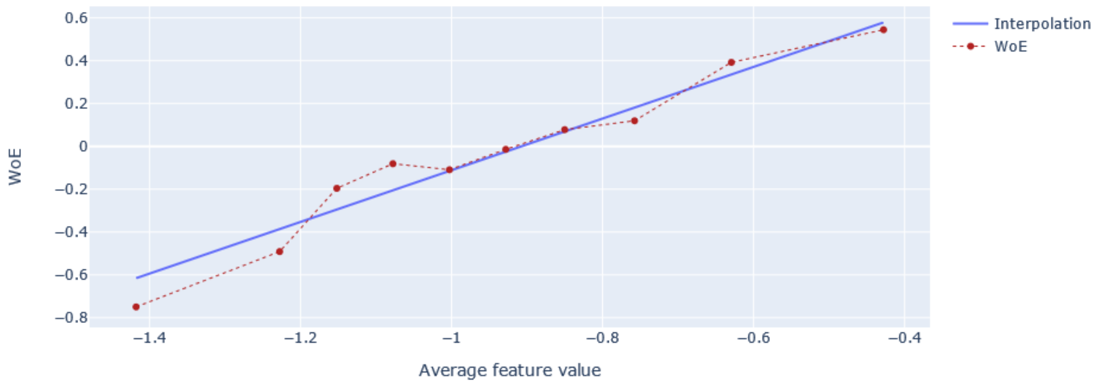
source4_feature1 ,R_sqr = 0.9902, auc = 0.511

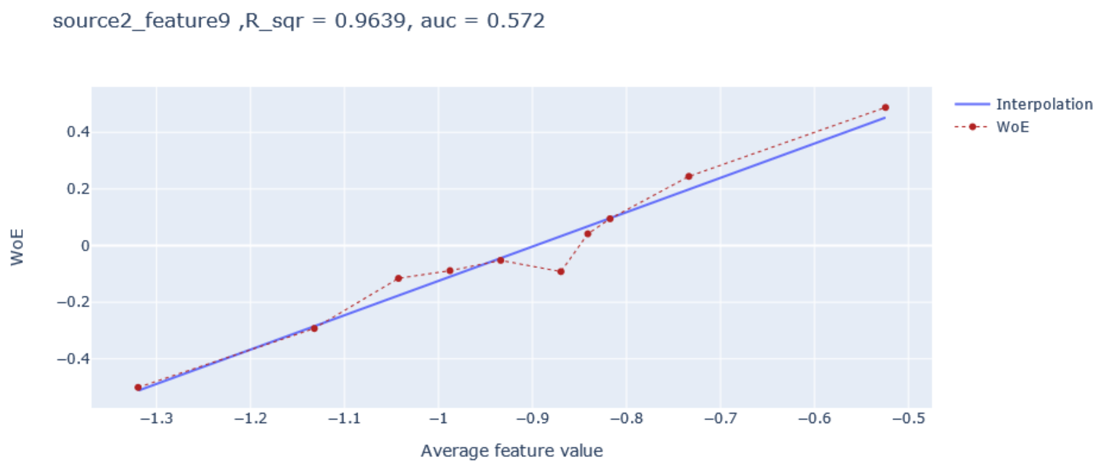


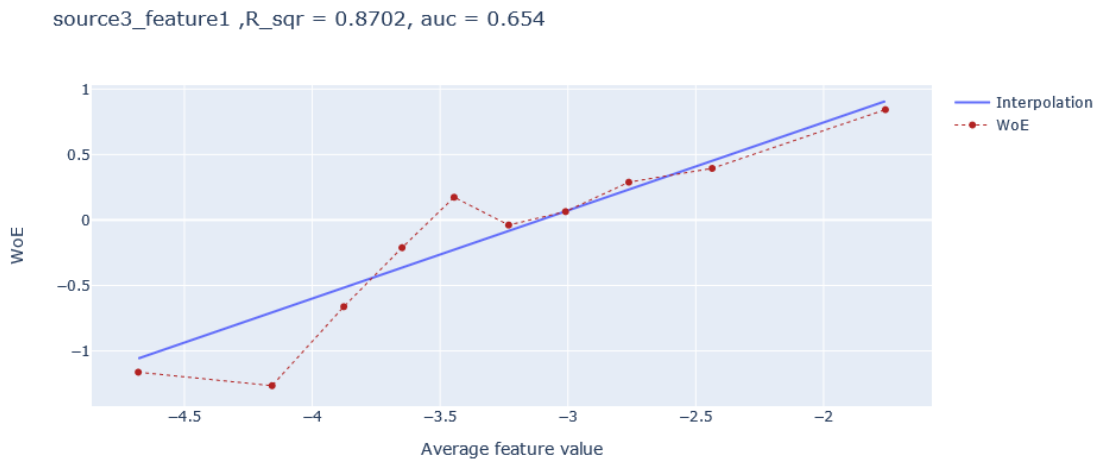
source4_feature1 ,R_sqr = 0.9902, auc = 0.511

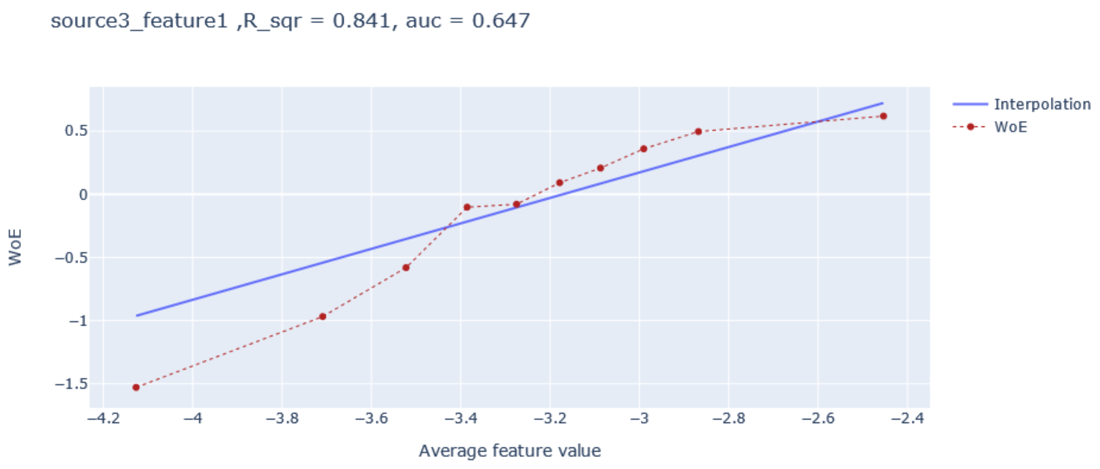


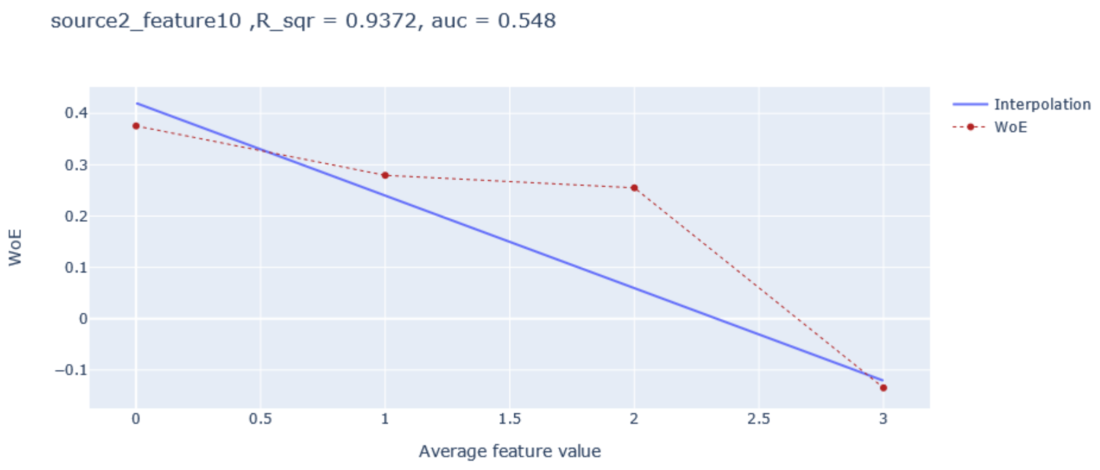
source2_feature9 ,R_sqr = 0.952, auc = 0.598



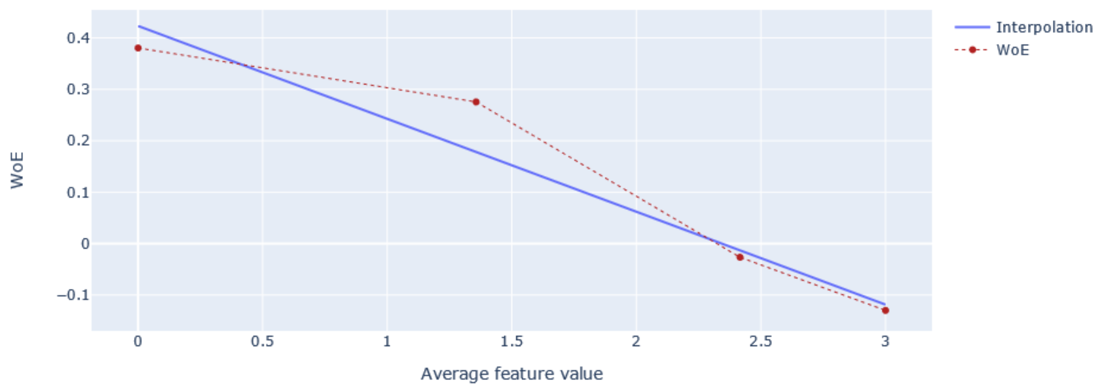

source2_feature9 ,R_sqr = 0.9639, auc = 0.572
Interpolation
WoE
WoE
Average feature value
0.4
0.2
0
−0.2
−0.4
−1.3
−1.2
−1.1
−1
−0.9
−0.8
−0.7
−0.6
−0.5


source3_feature1 ,R_sqr = 0.8702, auc = 0.654
Interpolation
WoE
WoE
Average feature value
1
0.5
0
−0.5
−1
−4.5
−4
−3.5
−3
−2.5
−2

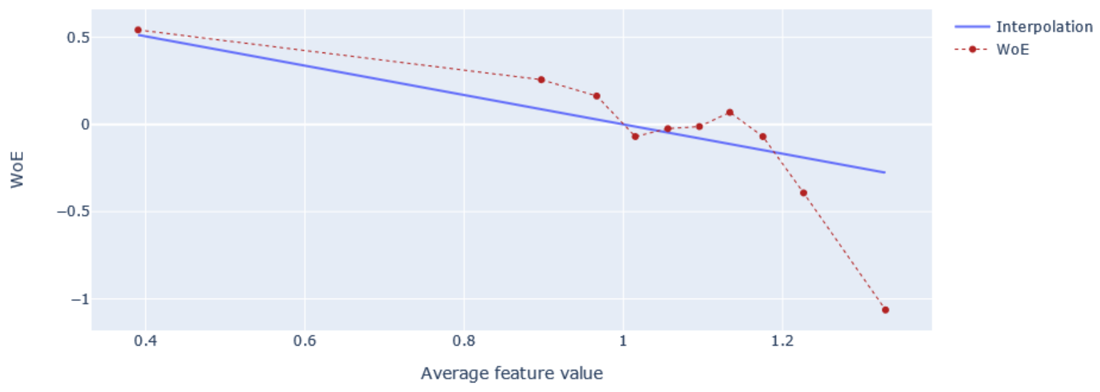

source3_feature1 ,R_sqr = 0.841, auc = 0.647
Interpolation
WoE
0.5
0
−0.5
−1
−1.5
−4.2
−4
−3.8
−3.6
−3.4
−3.2
−3
−2.8
−2.6
−2.4
Average feature value


source2_feature10 ,R_sqr = 0.9372, auc = 0.548
Interpolation
WoE
0.4
0.3
0.2
0.1
0
−0.1
0
0.5
1
1.5
2
2.5
3
Average feature value

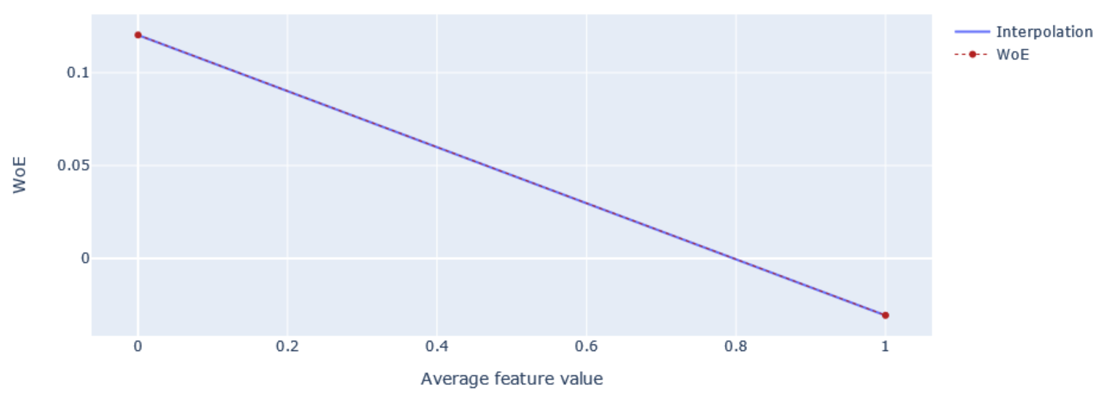
source2_feature10 ,R_sqr = 0.9569, auc = 0.546



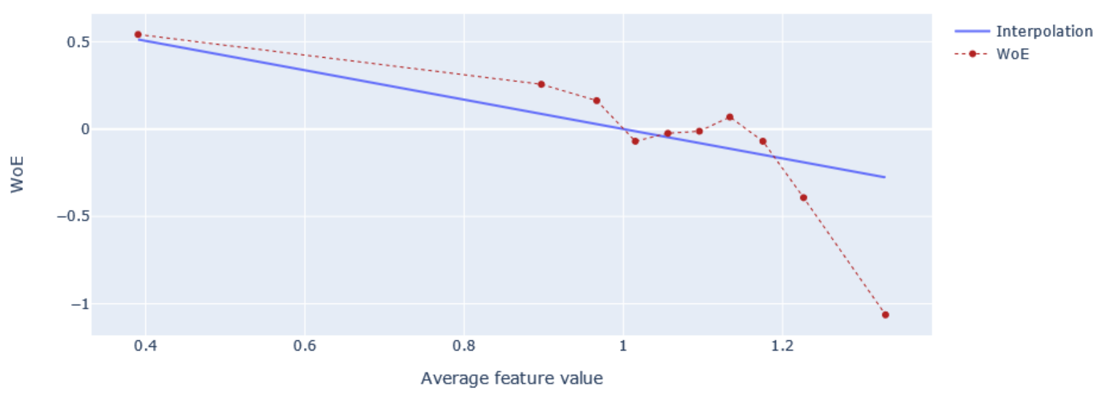
source2_feature3 ,R_sqr = 0.5425, auc = 0.589

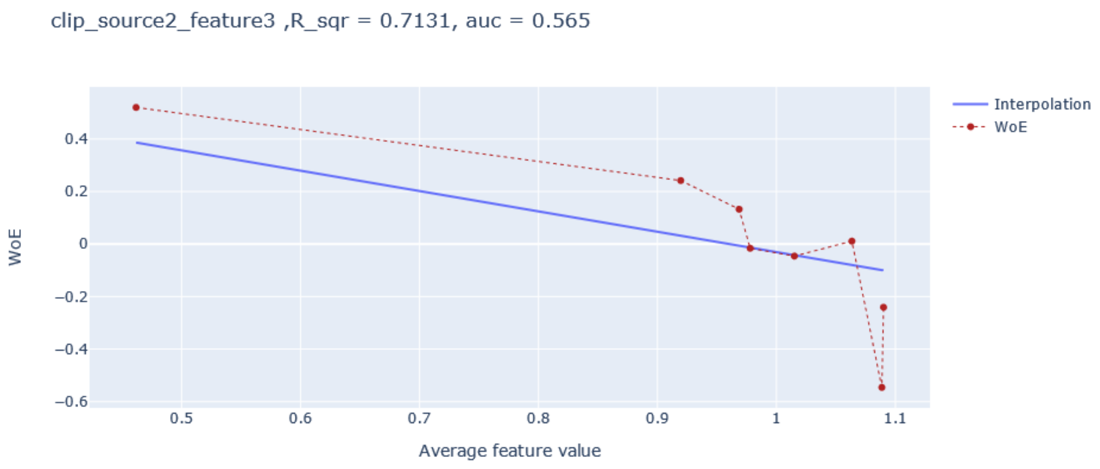


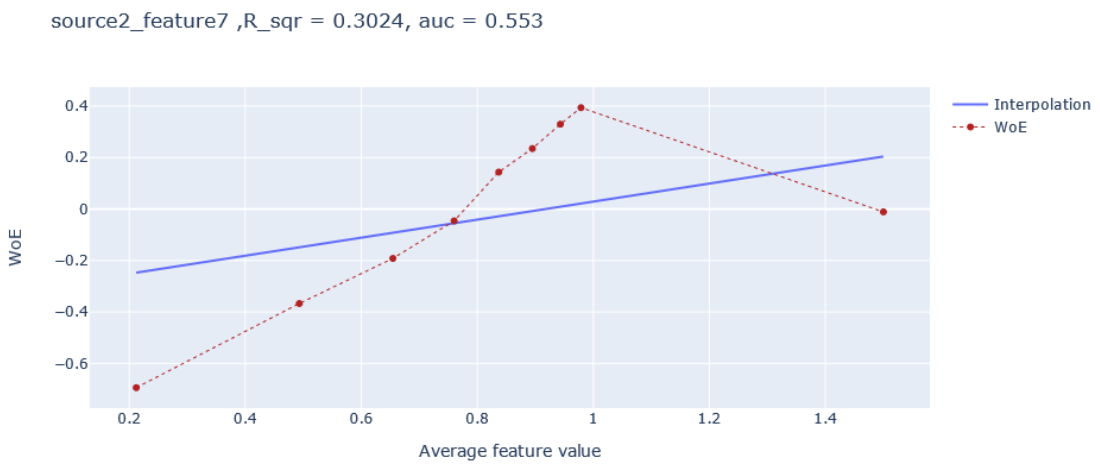
flg_source2_feature3 ,R_sqr = 1.0, auc = 0.512



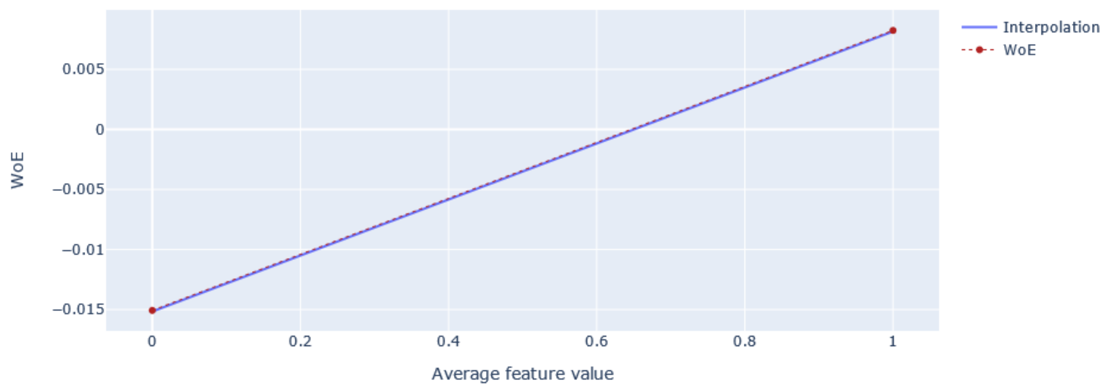
source2_feature3 ,R_sqr = 0.5425, auc = 0.589



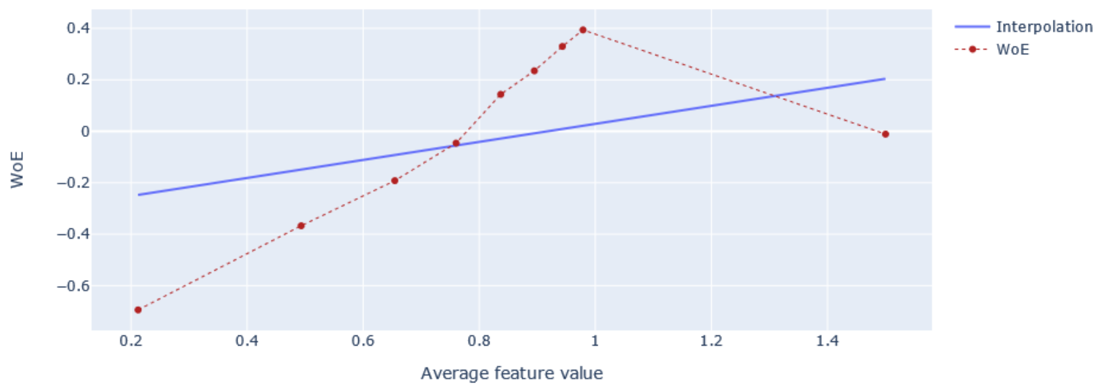

clip_source2_feature3 ,R_sqr = 0.7131, auc = 0.565
Interpolation
WoE
0.4
0.2
0
−0.2
−0.4
−0.6
0.5
0.6
0.7
0.8
0.9
1
1.1
WoE
Average feature value

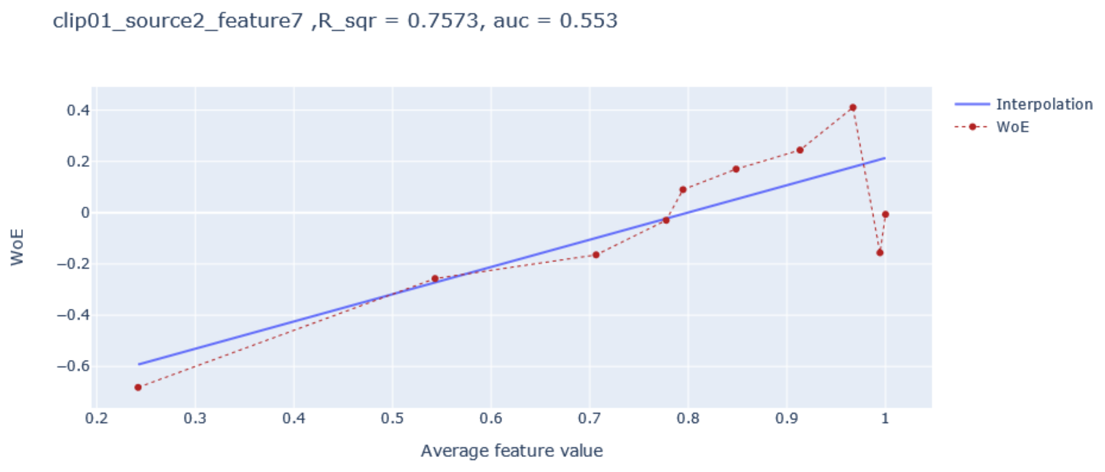

source2_feature7 ,R_sqr = 0.3024, auc = 0.553
Interpolation
WoE
0.4
0.2
0
−0.2
−0.4
−0.6
0.2
0.4
0.6
0.8
1
1.2
1.4
WoE
Average feature value

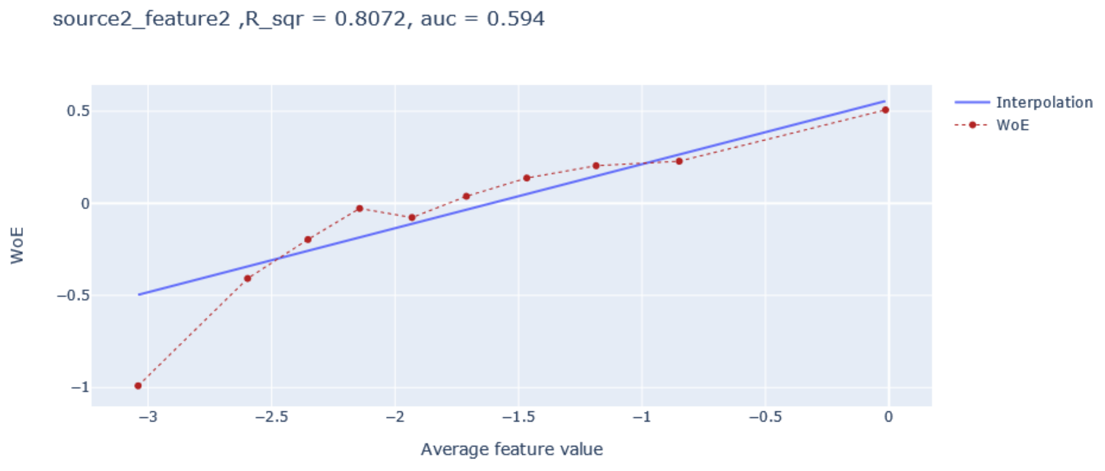
flg_source2_feature7 ,R_sqr = 0.9999, auc = 0.503

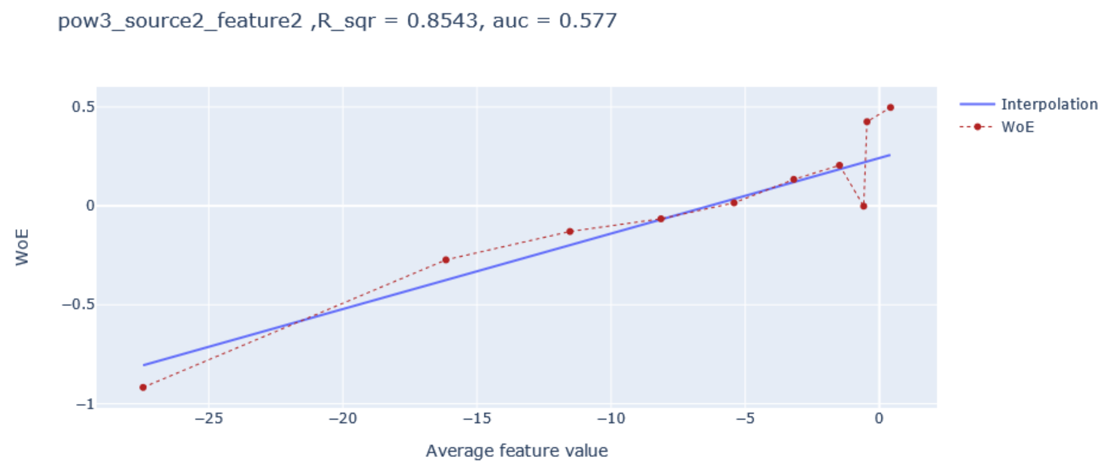


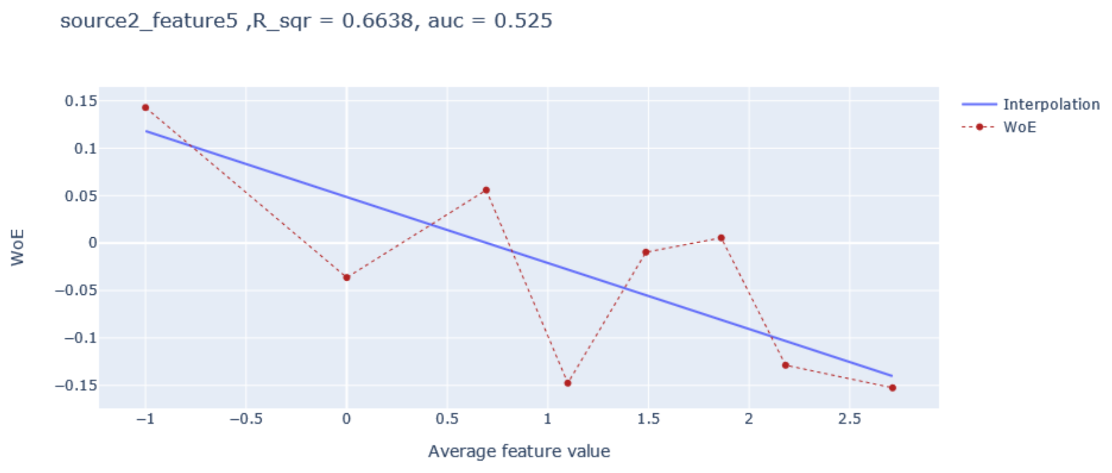
source2_feature7 ,R_sqr = 0.3024, auc = 0.553



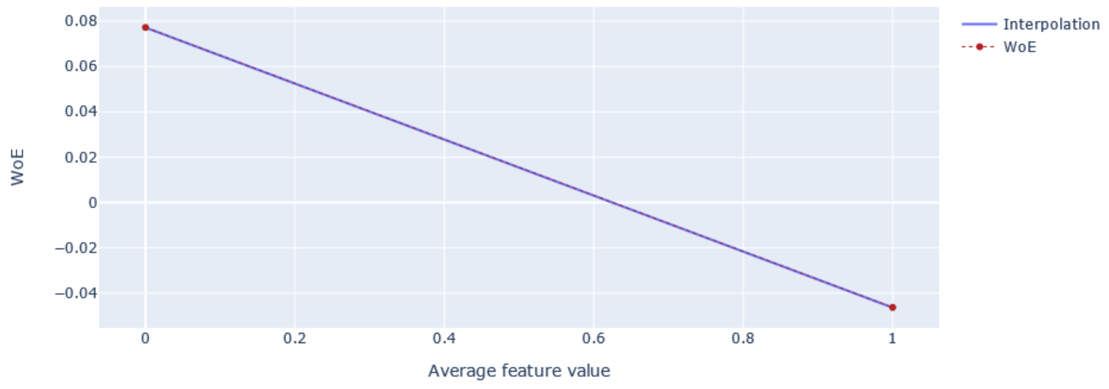

clip01_source2_feature7 ,R_sqr = 0.7573, auc = 0.553
Interpolation
WoE
WoE
Average feature value
0.4
0.2
0
−0.2
−0.4
−0.6
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


source2_feature2 ,R_sqr = 0.8072, auc = 0.594
Interpolation
WoE
WoE
Average feature value
0.5
0
−0.5
−1
−3
−2.5
−2
−1.5
−1
−0.5
0

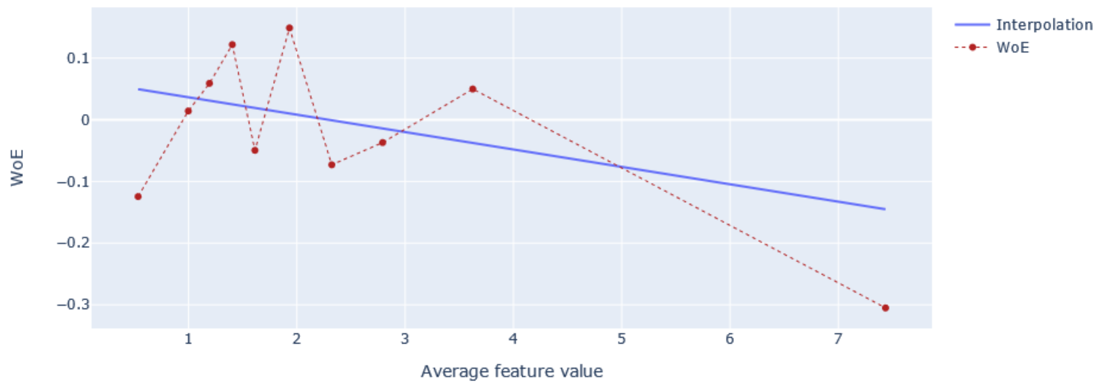

pow3_source2_feature2 ,R_sqr = 0.8543, auc = 0.577
WoE
Average feature value
0.5
0
−0.5
−1
−25
−20
−15
−10
−5
0
Interpolation
WoE

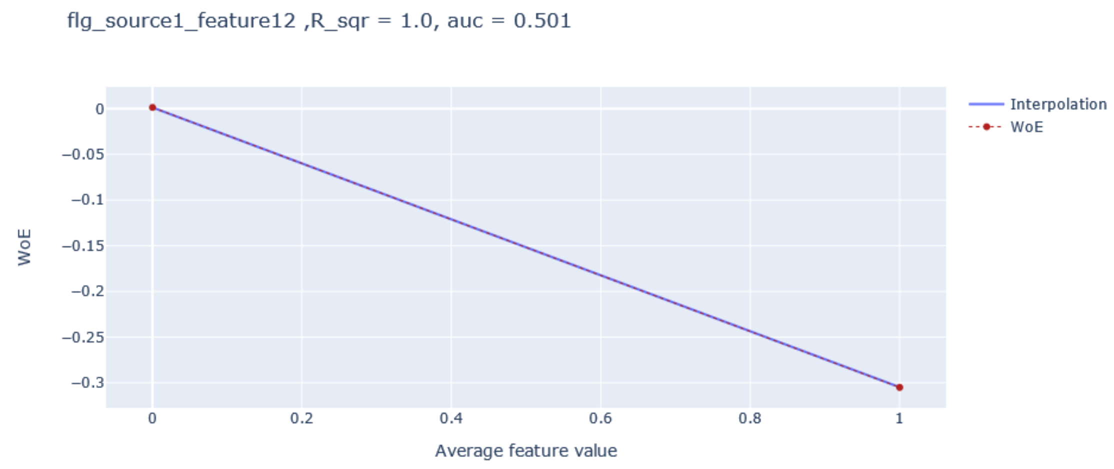

source2_feature5 ,R_sqr = 0.6638, auc = 0.525
WoE
Average feature value
0.15
0.1
0.05
0
−0.05
−0.1
−0.15
−1
−0.5
0
0.5
1
1.5
2
2.5
Interpolation
WoE

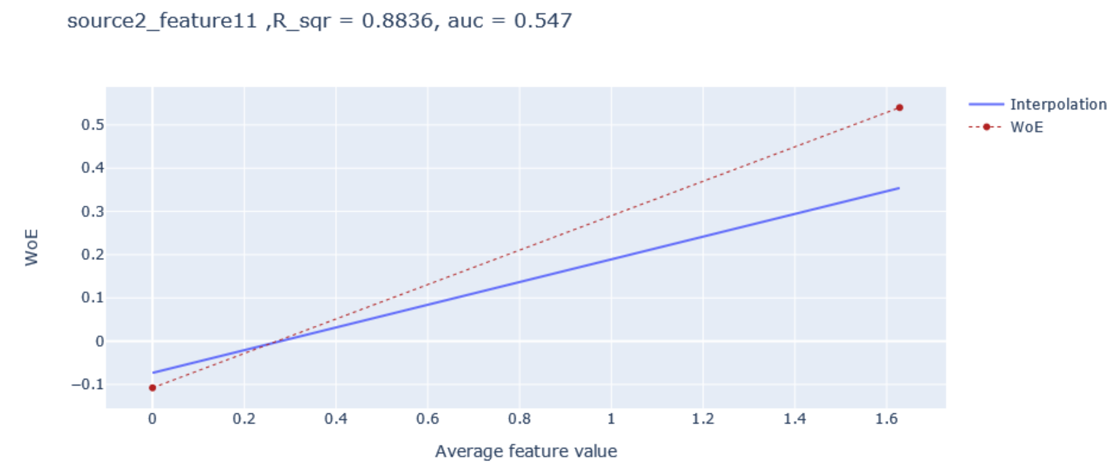
flg_source2_feature5 ,R_sqr = 1.0, auc = 0.514



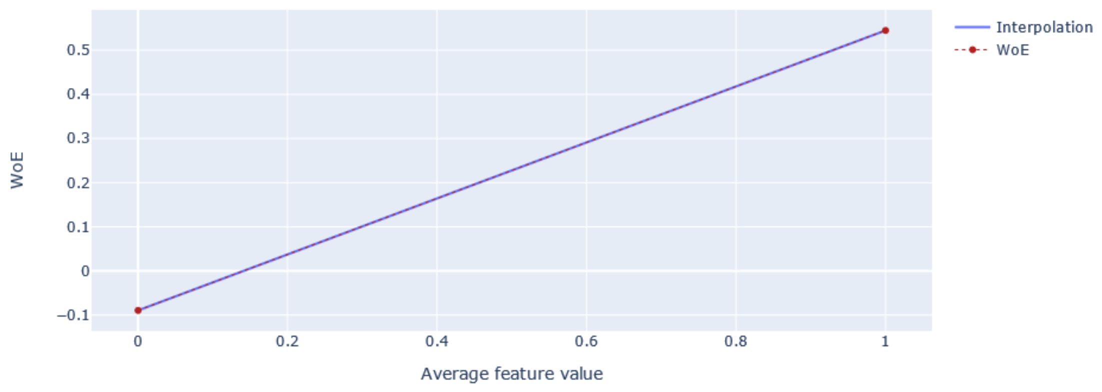
source1_feature12 ,R_sqr = 0.3215, auc = 0.507



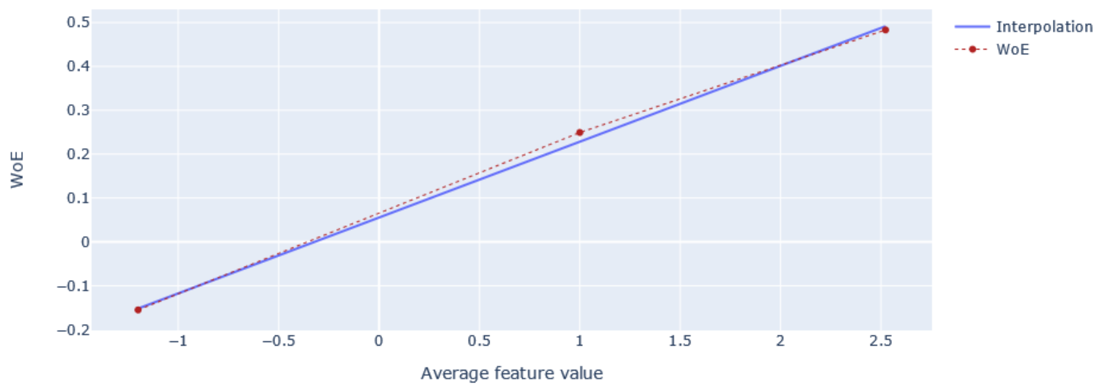

flg_source1_feature12 ,R_sqr = 1.0, auc = 0.501
Interpolation
WoE
WoE
Average feature value
0
−0.05
−0.1
−0.15
−0.2
−0.25
−0.3
0
0.2
0.4
0.6
0.8
1

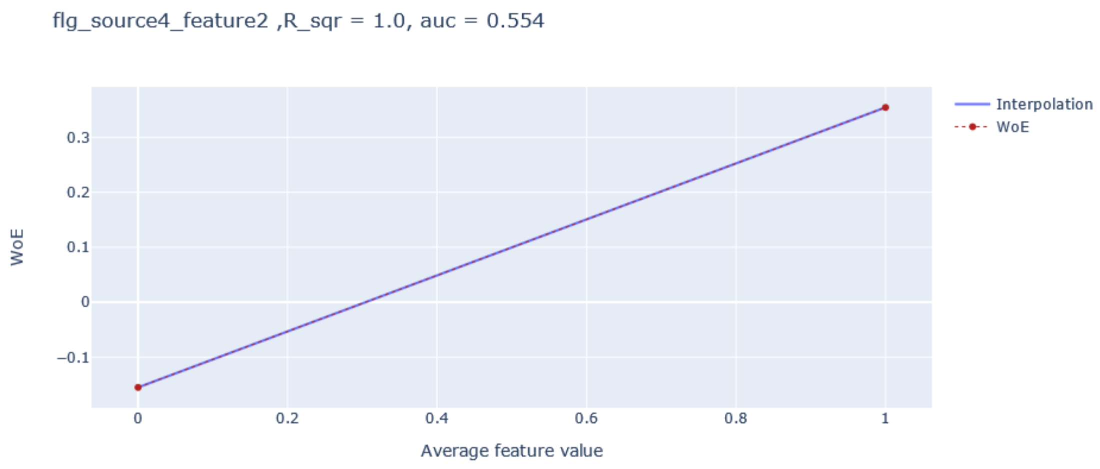

source2_feature11 ,R_sqr = 0.8836, auc = 0.547
Interpolation
WoE
WoE
Average feature value
0.5
0.4
0.3
0.2
0.1
0
−0.1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6

flg_source2_feature11 ,R_sqr = 1.0, auc = 0.538



source4_feature2 ,R_sqr = 0.9985, auc = 0.557



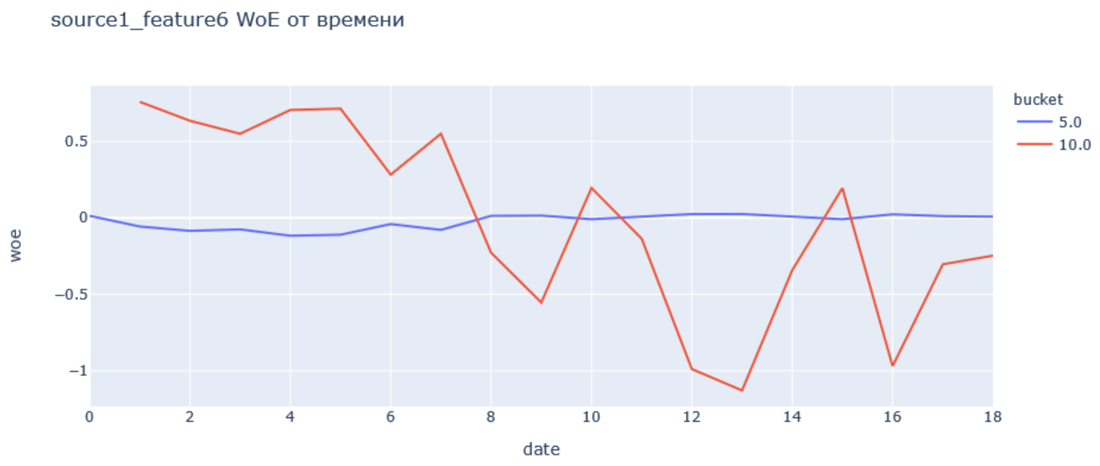


```
[25]: train_df['date'] = train_df['id'] // 2000
```

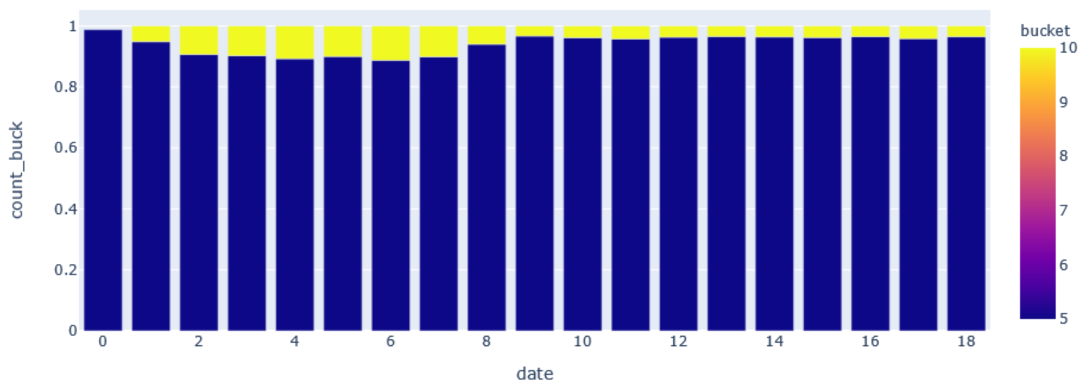
```
[26]: def simple_logreg(feature,target):
          model = make_pipeline(StandardScaler(),LogisticRegression())
          model.fit(np.array(feature).reshape(-1,1),np.array(target))
          return model.predict_proba(np.array(feature).reshape(-1,1))[:,1]

      def woe_stab(df, feature, date, target, num_buck = 10):
          df = df.assign(predict = simple_logreg(df[feature].astype(np.
       ↪float64),df[target]))
          agg = df.assign(bucket = np.ceil(df[feature].rank(pct = True) * num_buck),␣
       ↪obj_count = 1)\
                    .groupby(['bucket',date])\
                    .agg({target:'sum','predict':'mean','obj_count':sum,feature:
       ↪'mean'})\
                    .rename(columns = {target:'target_sum',feature:'av_f'})\
                    .assign(bad_rate = lambda x:x.target_sum/x.obj_count)
          agg = agg.assign(nums = agg.groupby(date)['obj_count'].transform('sum'),
                        bad_nums = agg.groupby(date)['target_sum'].transform('sum'))
          agg = agg.assign(woe = lambda x:((x.bad_rate/(1-x.bad_rate)) + 0.000001).
       ↪apply(log) -
                     (x.bad_nums / (x.nums - x.bad_nums) + 0.000001).apply(log)).
       ↪reset_index()
          agg = agg.assign(count_buck = lambda x: x.obj_count/x.nums)
          agg = agg[agg.target_sum != 0]
          fig_woe = px.line(agg, x=date, y='woe', title=f'{feature} WoE       ',␣
       ↪color = 'bucket')
```

```
    fig_dist = px.bar(agg, x=date, y='count_buck', title=f'{feature}                ␣
↪           ', color = 'bucket')

    fig_woe.update_layout(
                  width=1000,
                  height=450)
    fig_woe.show()

    fig_dist.update_layout(
                  width=1000,
                  height=450)
    fig_dist.show()
```

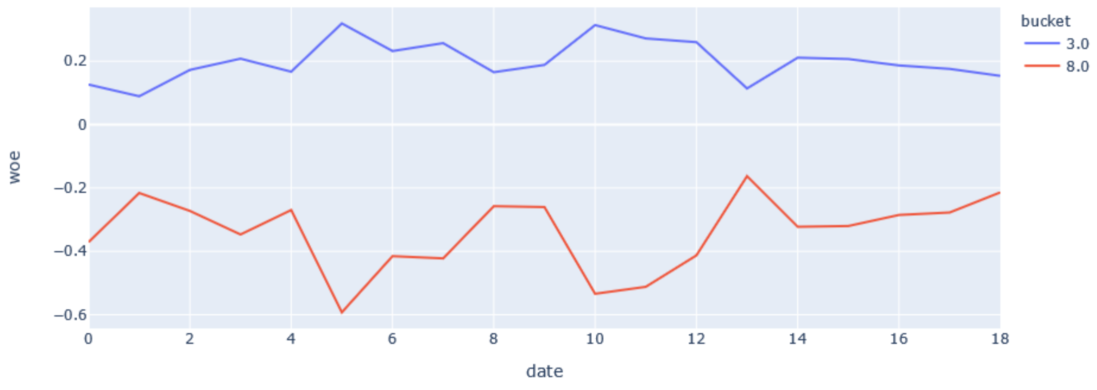
```
[27]: for feature in list(set(binary) - set(not_worthy)):
          woe_stab(train_df, feature, 'date', 'default_flg')
```

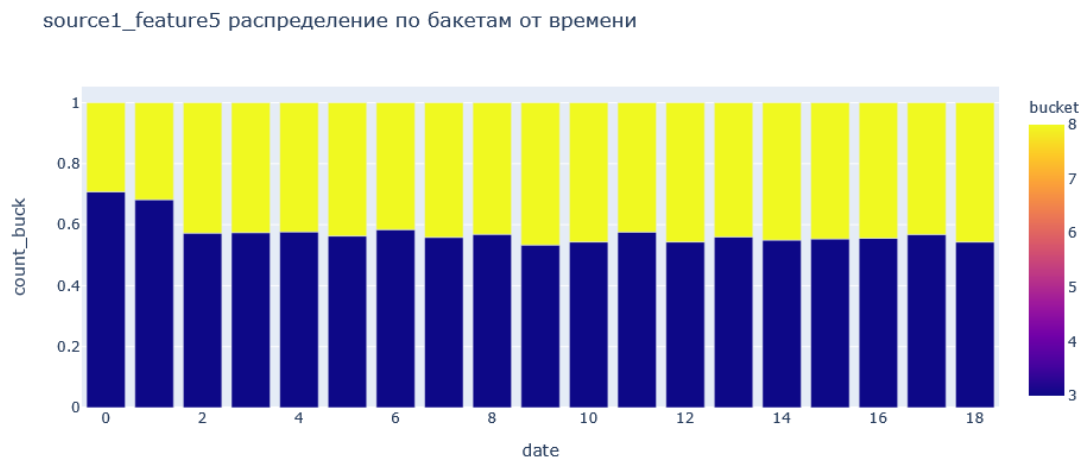


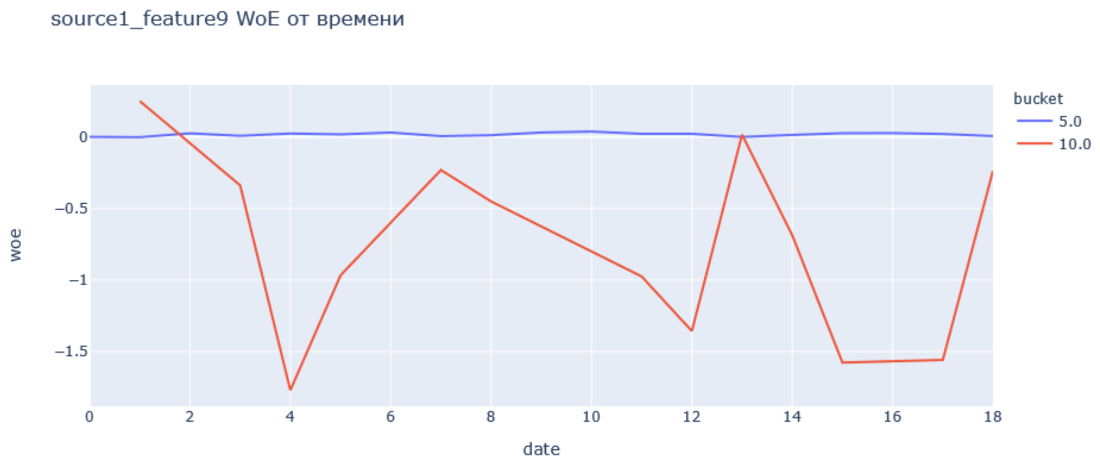
source1_feature6 распределение по бакетам от времени

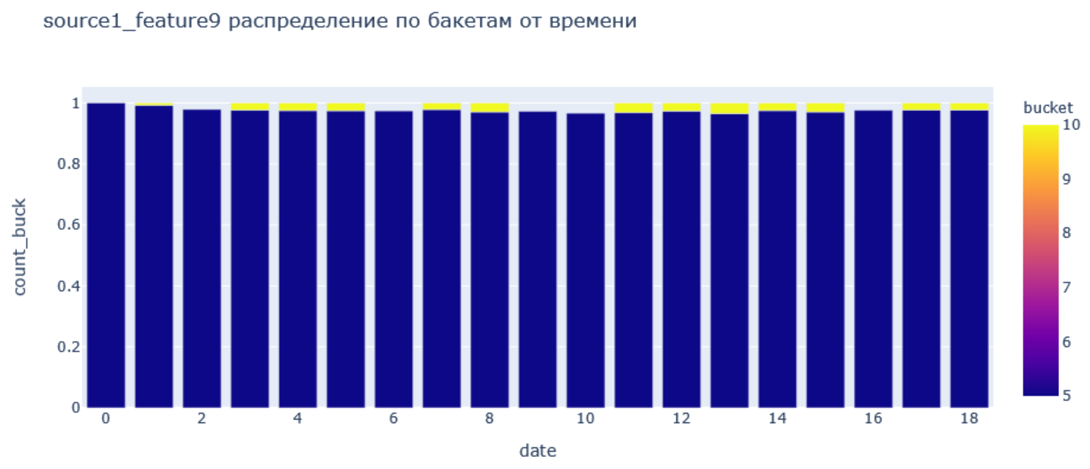


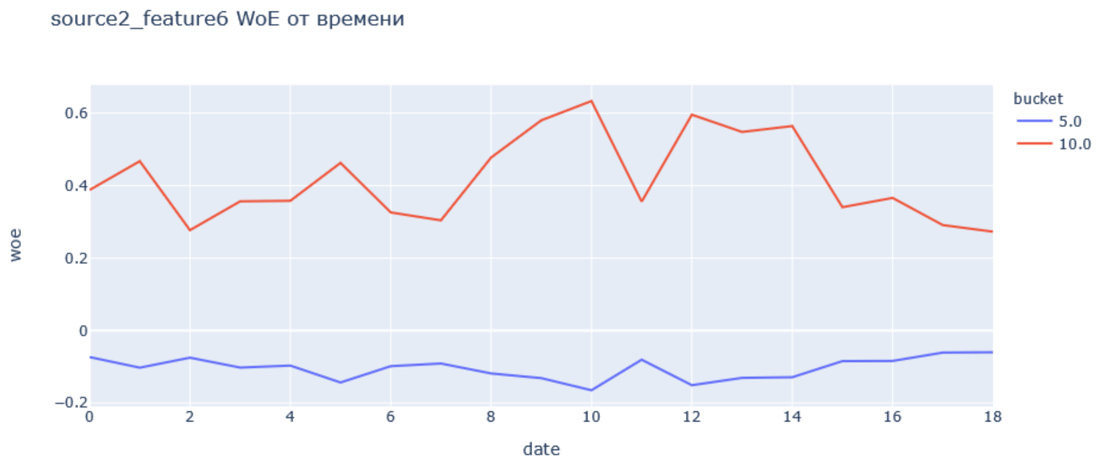
source1_feature5 WoE от времени



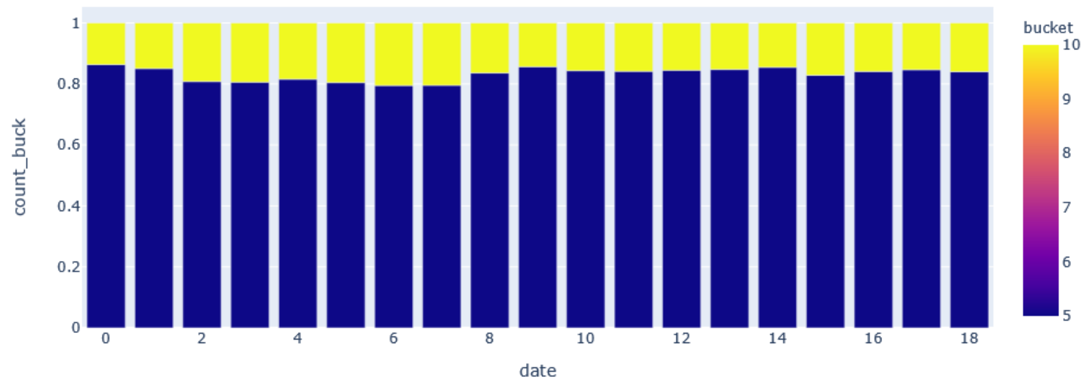

source1_feature5 распределение по бакетам от времени
count_buck
date
bucket
0
0.2
0.4
0.6
0.8
1
2
4
6
8
10
12
14
16
18
3
4
5
6
7
8


source1_feature9 WoE от времени
woe
date
bucket
5.0
10.0
0
−0.5
−1
−1.5
0
2
4
6
8
10
12
14
16
18

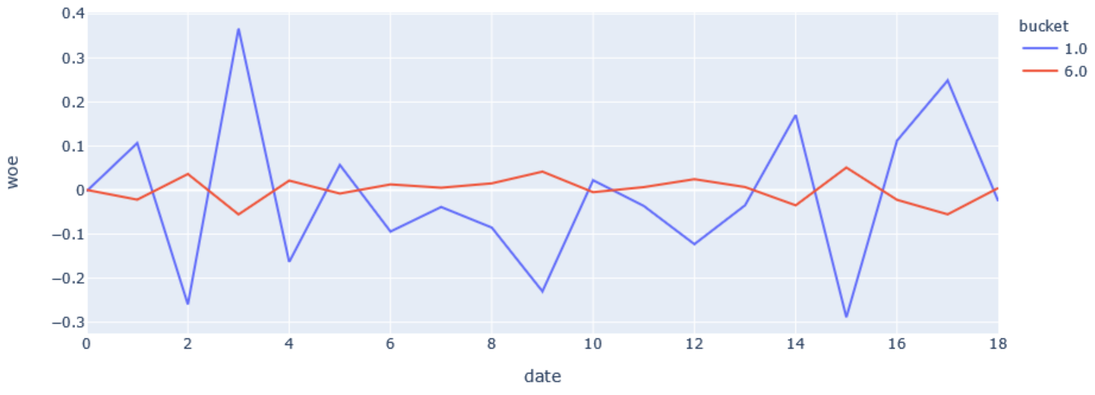

source1_feature9 распределение по бакетам от времени
count_buck
0
0.2
0.4
0.6
0.8
1
0
2
4
6
8
10
12
14
16
18
date
bucket
10
9
8
7
6
5


source2_feature6 WoE от времени
woe
−0.2
0
0.2
0.4
0.6
0
2
4
6
8
10
12
14
16
18
date
bucket
5.0
10.0

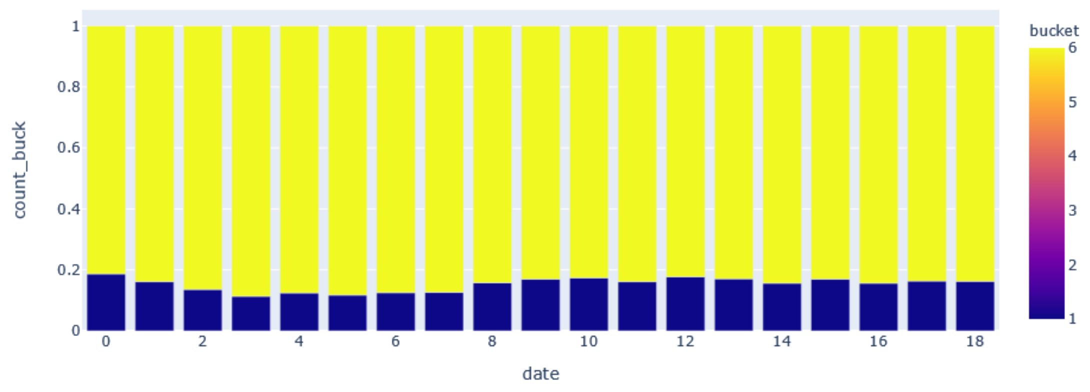
source2_feature6 распределение по бакетам от времени



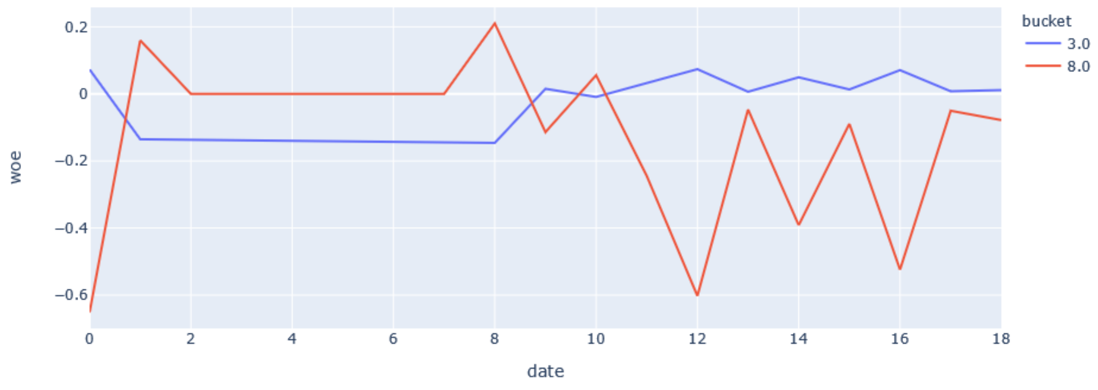
source2_feature1 WoE от времени

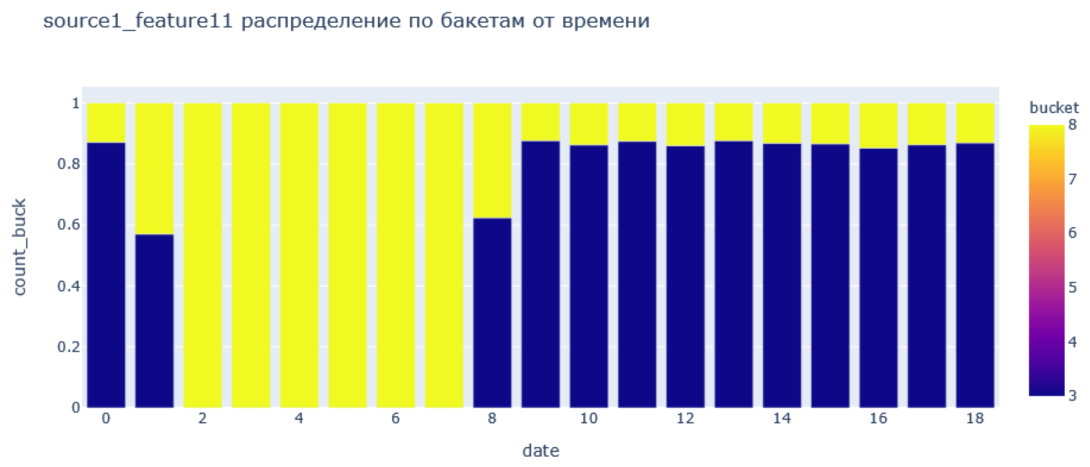


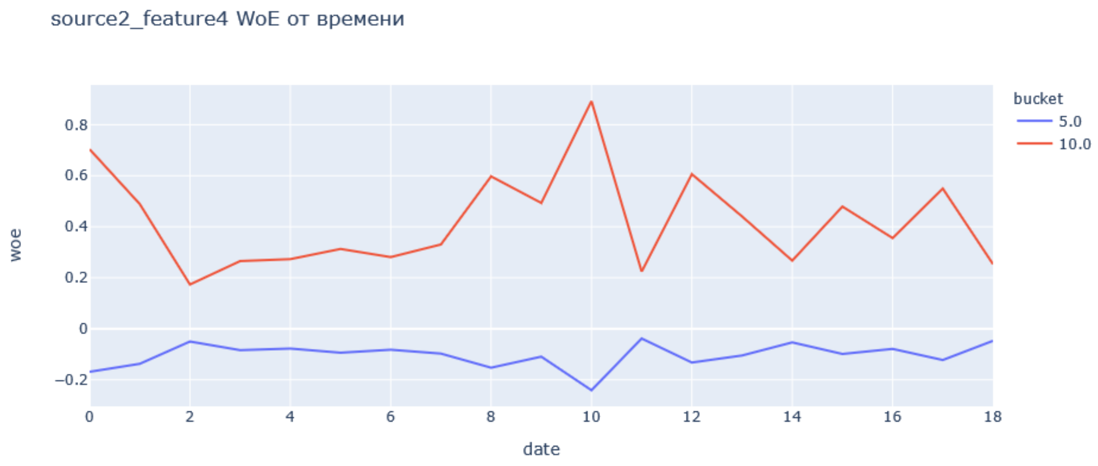
source2_feature1 распределение по бакетам от времени

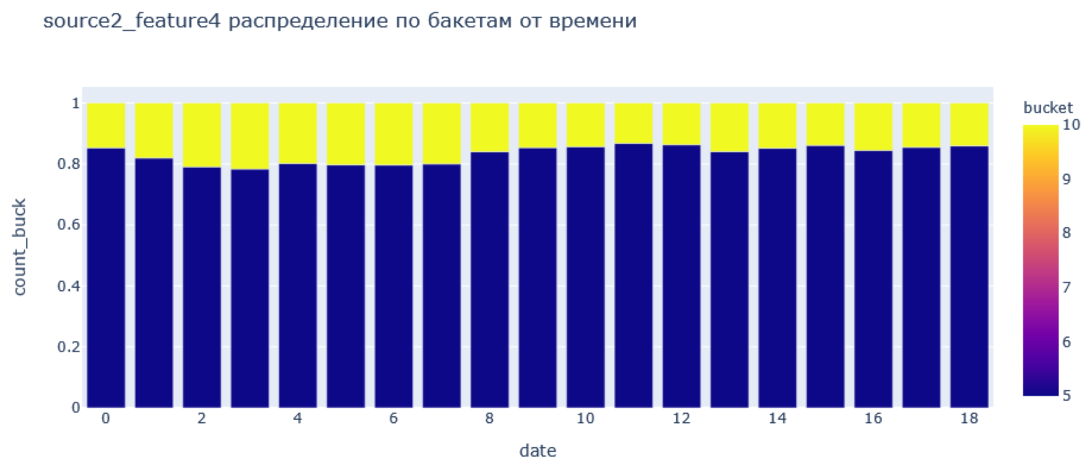


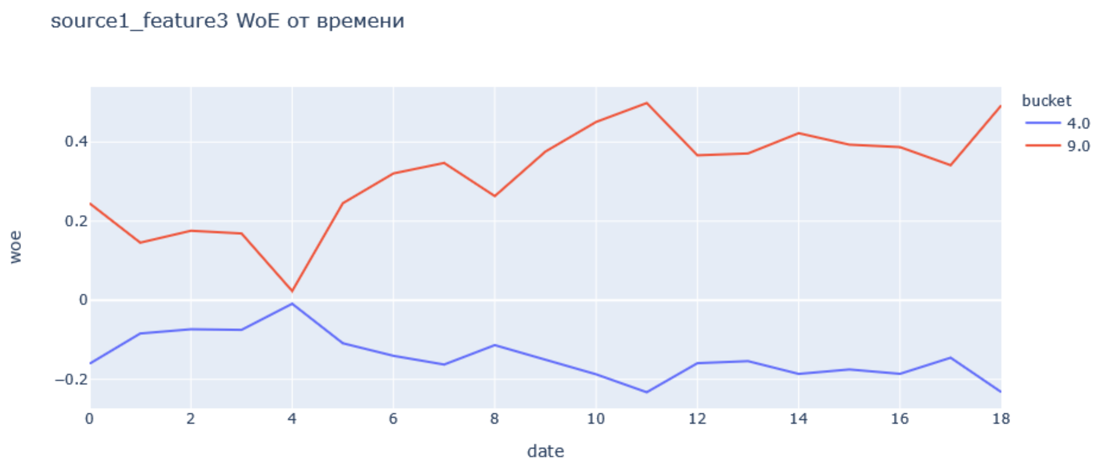
source1_feature11 WoE от времени

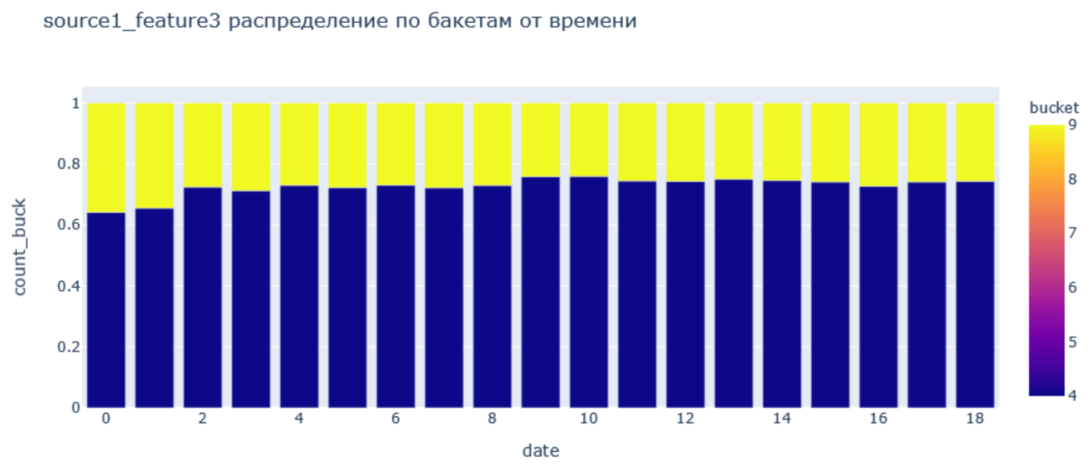


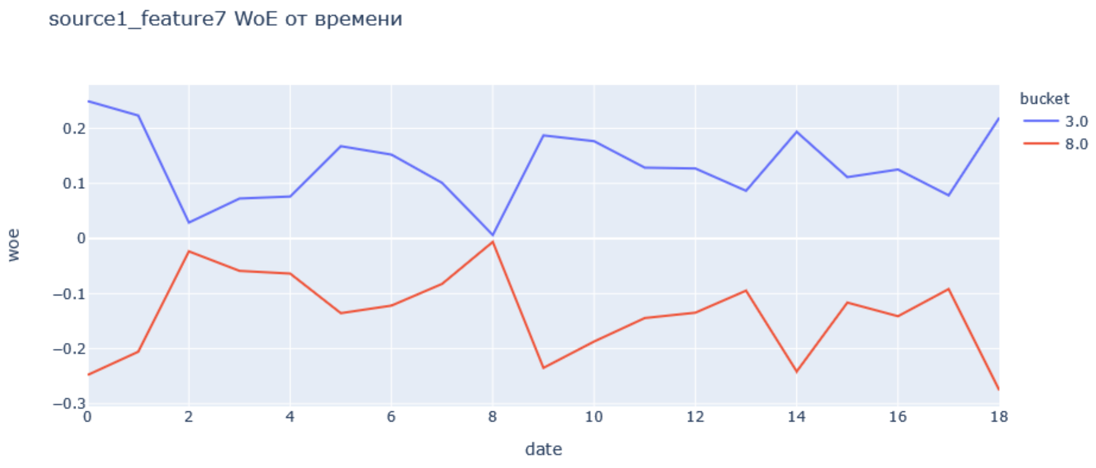

source1_feature11 распределение по бакетам от времени
count_buck
date
bucket
0
0.2
0.4
0.6
0.8
1
0
2
4
6
8
10
12
14
16
18
3
4
5
6
7
8


source2_feature4 WoE от времени
woe
date
bucket
5.0
10.0
−0.2
0
0.2
0.4
0.6
0.8
0
2
4
6
8
10
12
14
16
18

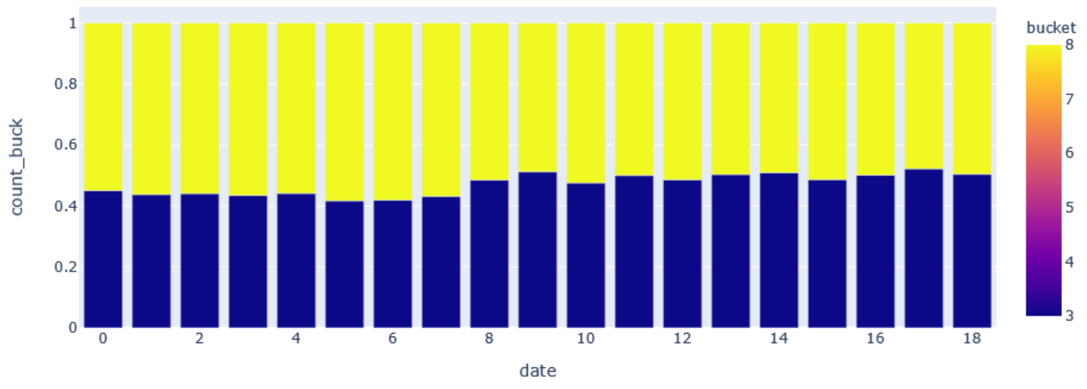

source2_feature4 распределение по бакетам от времени
count_buck
date
bucket
0
0.2
0.4
0.6
0.8
1
0
2
4
6
8
10
12
14
16
18
5
6
7
8
9
10


source1_feature3 WoE от времени
woe
date
bucket
4.0
9.0
−0.2
0
0.2
0.4
0
2
4
6
8
10
12
14
16
18

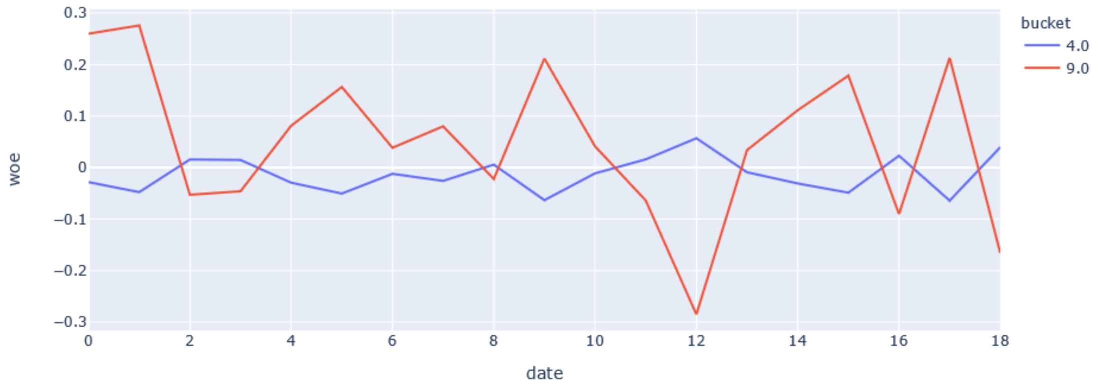

source1_feature3 распределение по бакетам от времени
count_buck
0
0.2
0.4
0.6
0.8
1
0
2
4
6
8
10
12
14
16
18
date
bucket
9
8
7
6
5
4


source1_feature7 WoE от времени
woe
0.2
0.1
0
−0.1
−0.2
−0.3
0
2
4
6
8
10
12
14
16
18
date
bucket
3.0
8.0

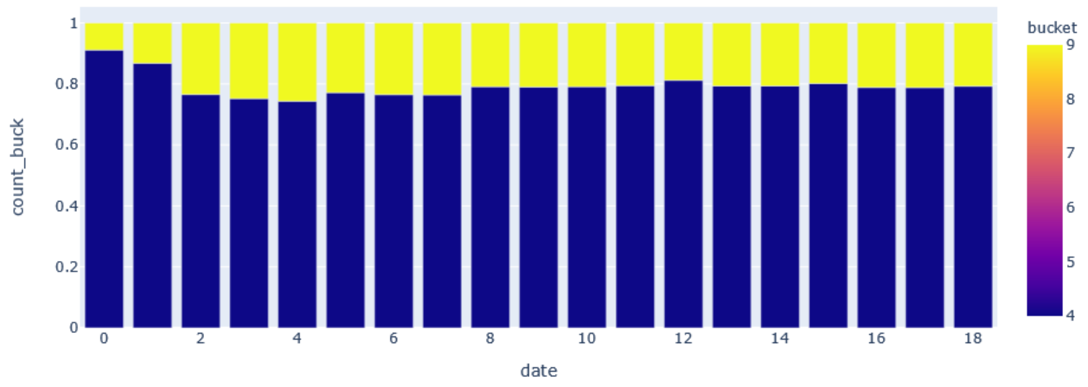
source1_feature7 распределение по бакетам от времени



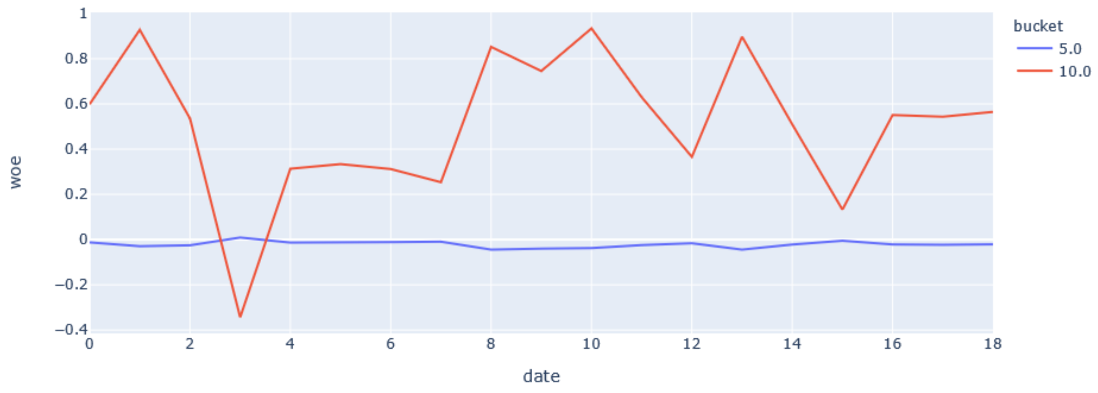
source1_feature1 WoE от времени

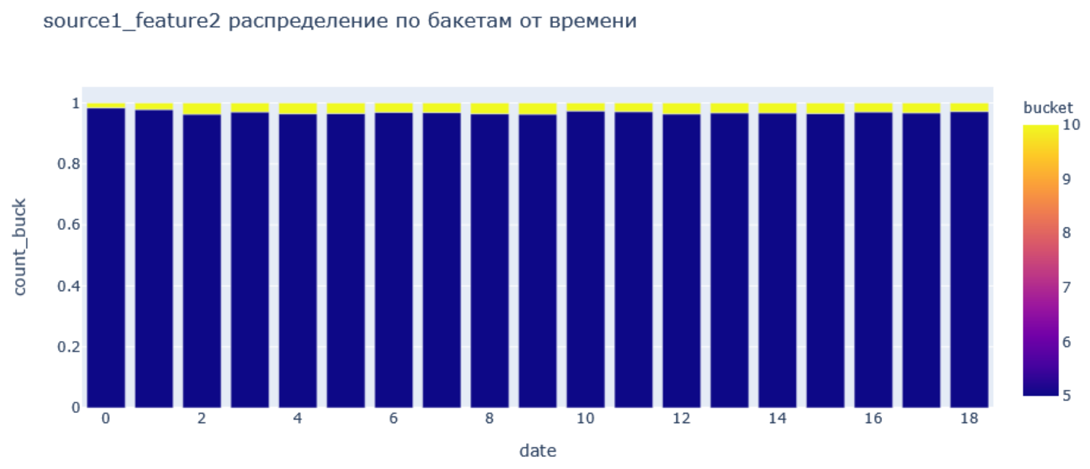


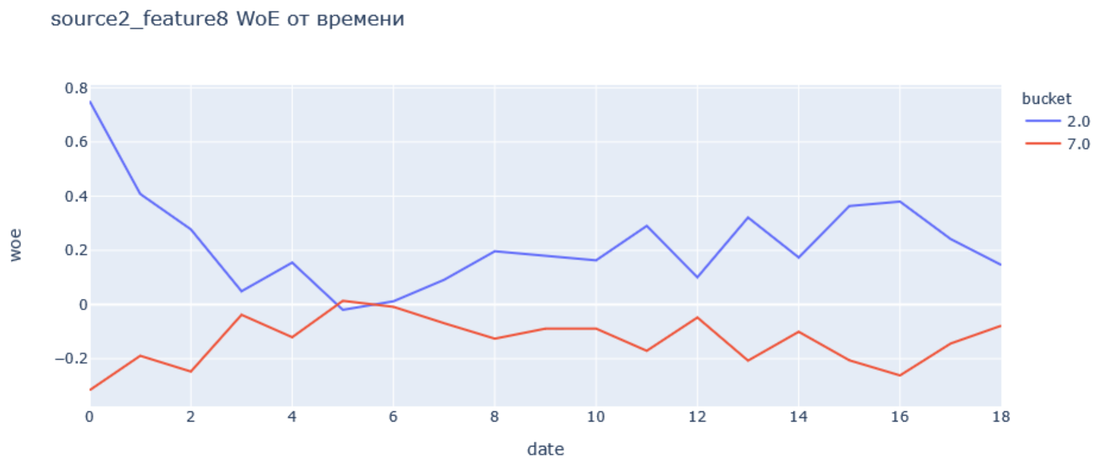
source1_feature1 распределение по бакетам от времени



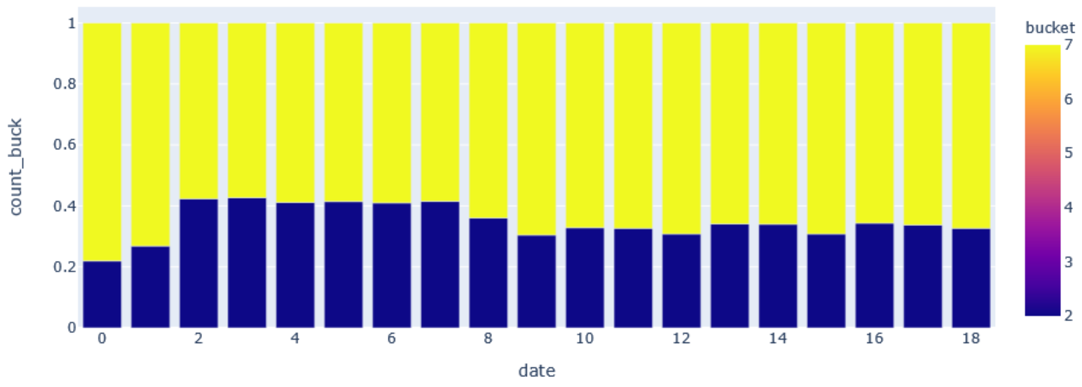
source1_feature2 WoE от времени



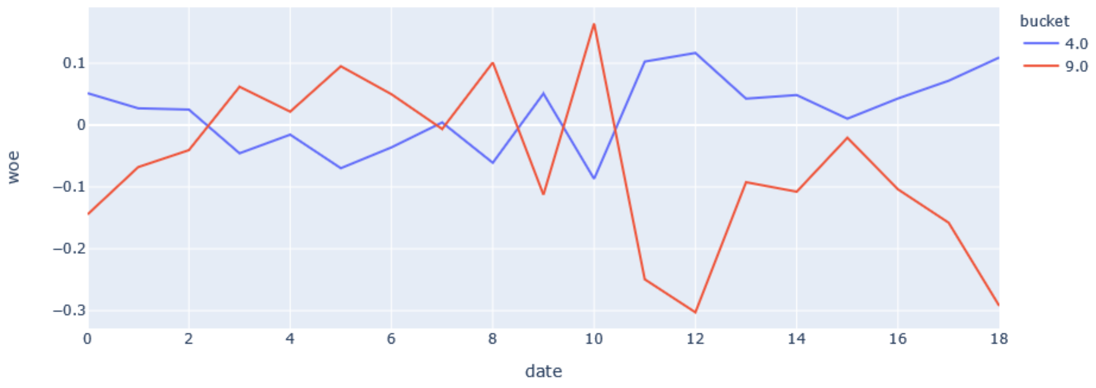

source1_feature2 распределение по бакетам от времени
count_buck
date
bucket
0
2
4
6
8
10
12
14
16
18
0
0.2
0.4
0.6
0.8
1
5
6
7
8
9
10


source2_feature8 WoE от времени
woe
date
bucket
2.0
7.0
0.8
0.6
0.4
0.2
0
−0.2
0
2
4
6
8
10
12
14
16
18

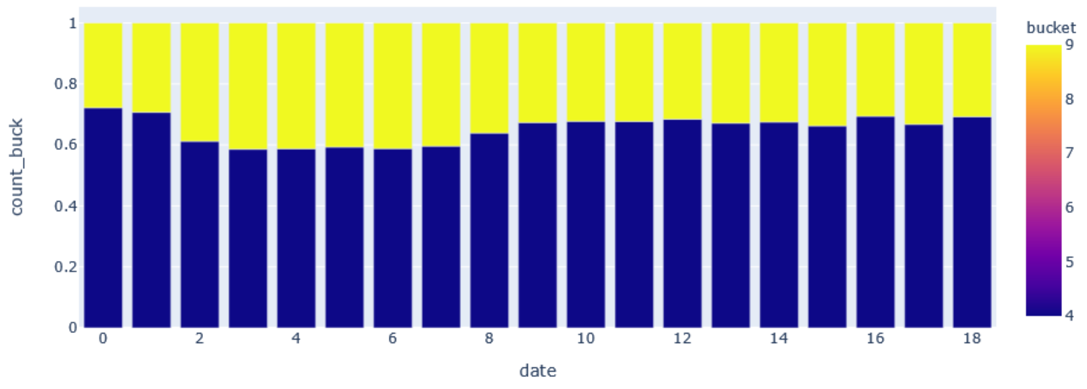
source2_feature8 распределение по бакетам от времени



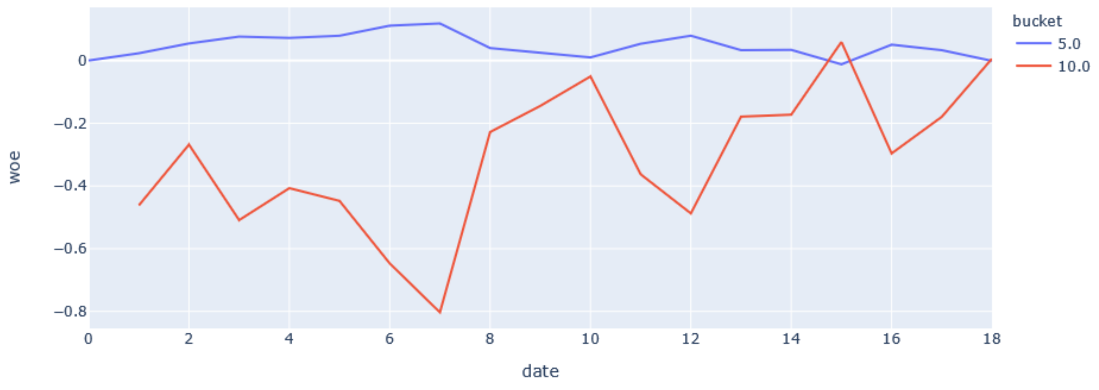
source1_feature8 WoE от времени

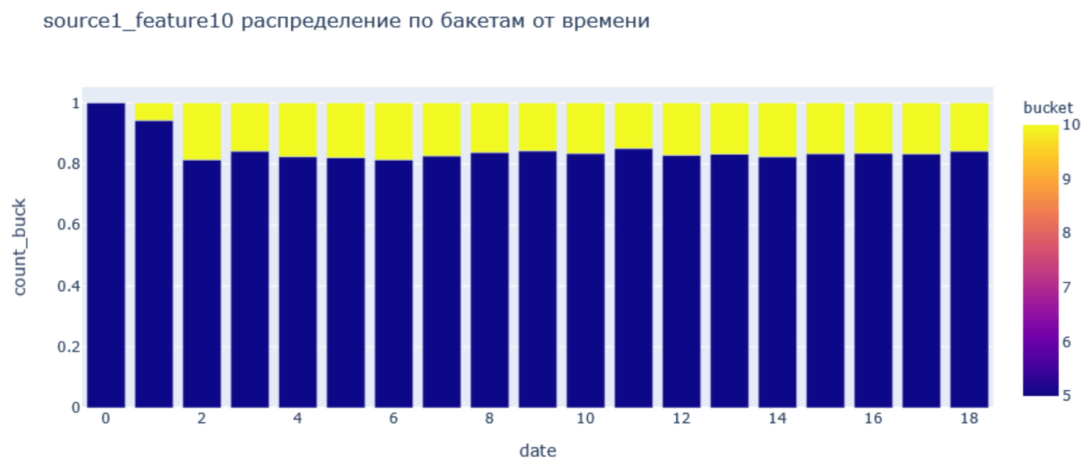


source1_feature8 распределение по бакетам от времени



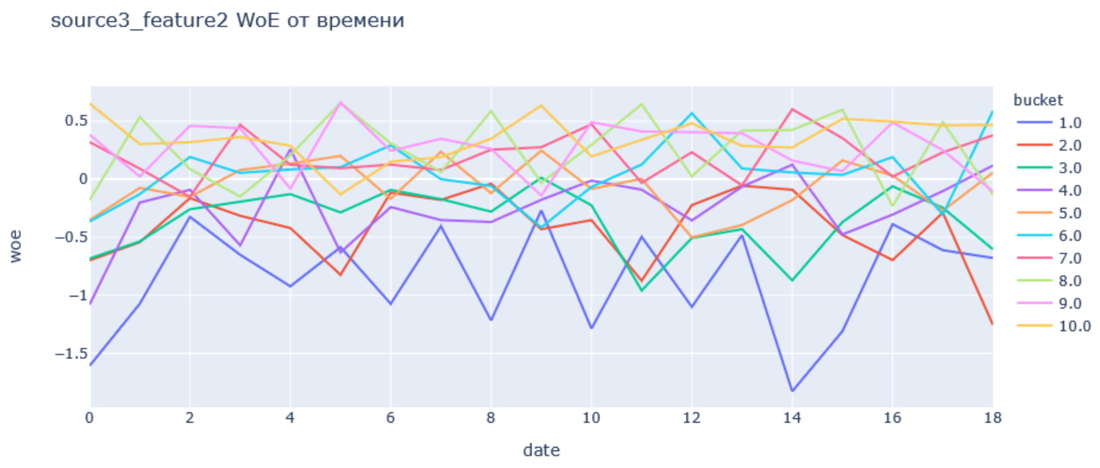
source1_feature10 WoE от времени



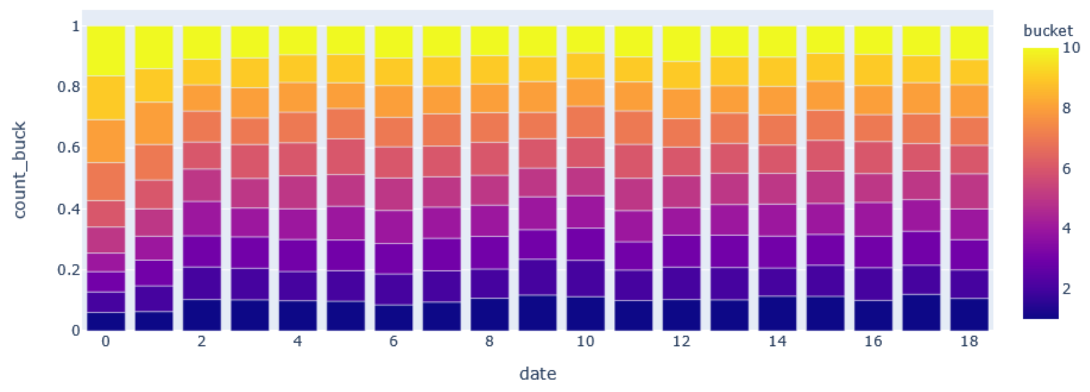


```
[28]: unstable_binary = [
          'source2_feature6', 'flg_source2_feature3'
      ]
```

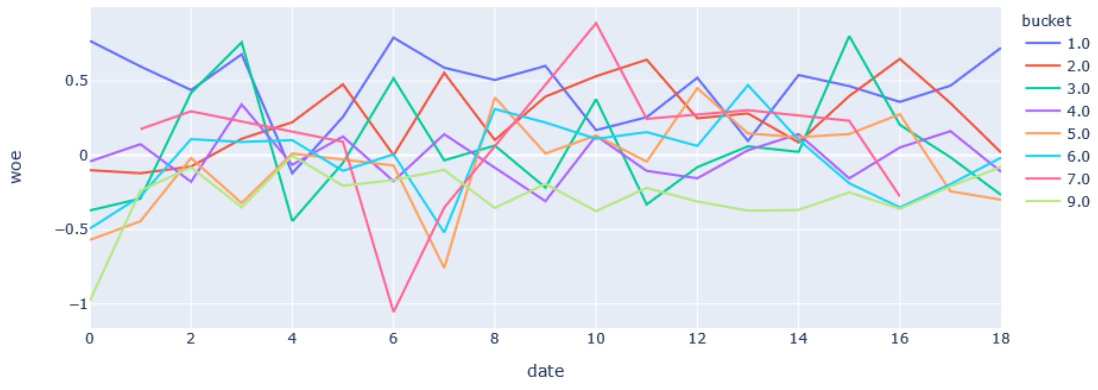
```
[29]: for feature in list(set(features) - set(binary)):
          woe_stab(train_df, feature, 'date', 'default_flg')
```



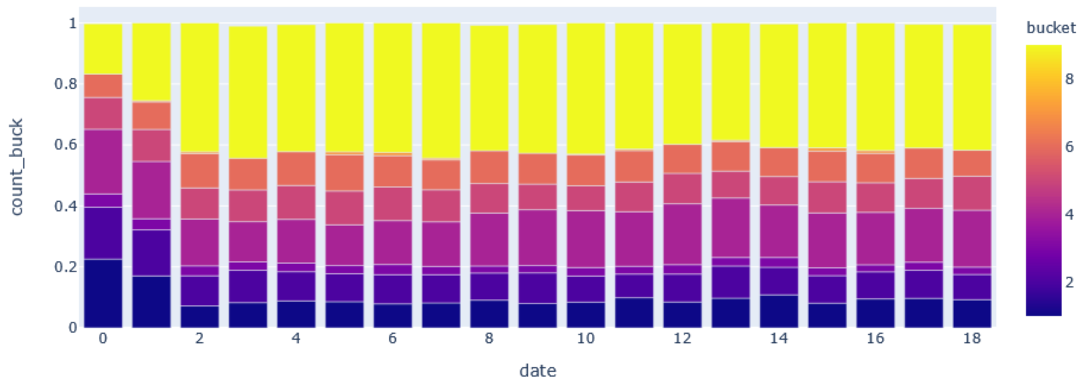
source3_feature2 распределение по бакетам от времени



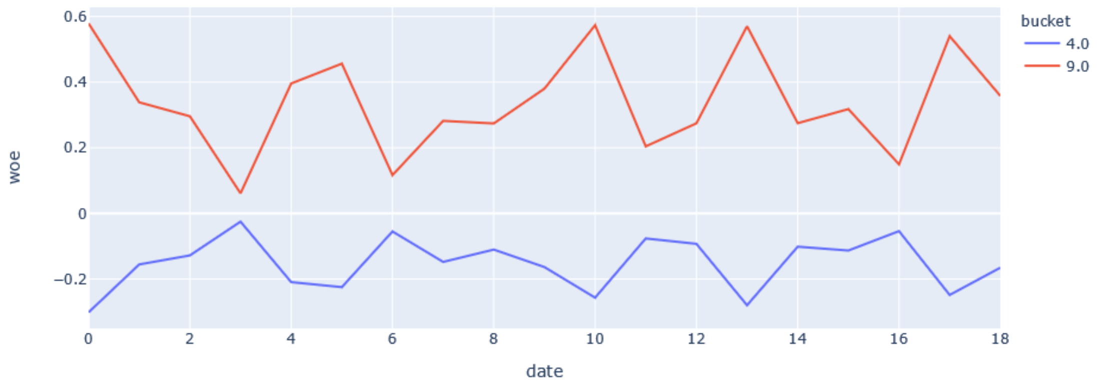
clip_source2_feature3 WoE от времени

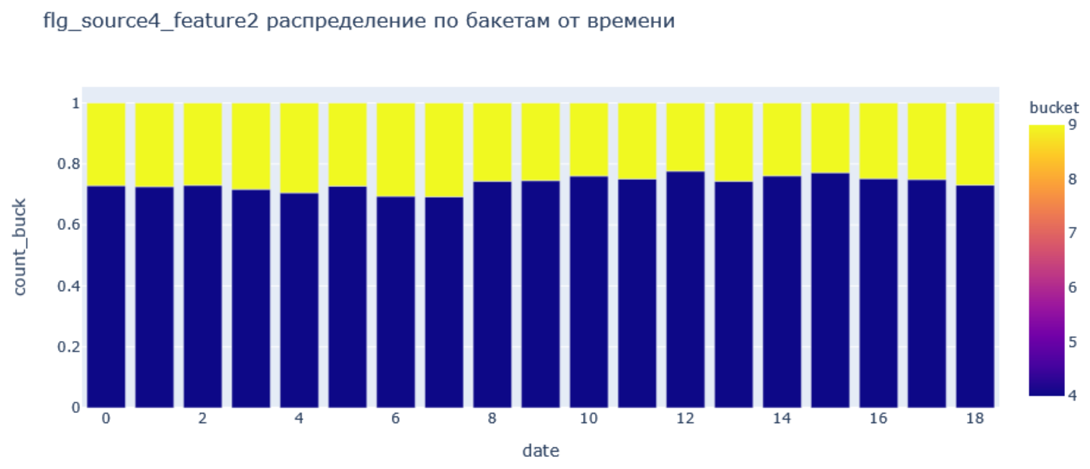


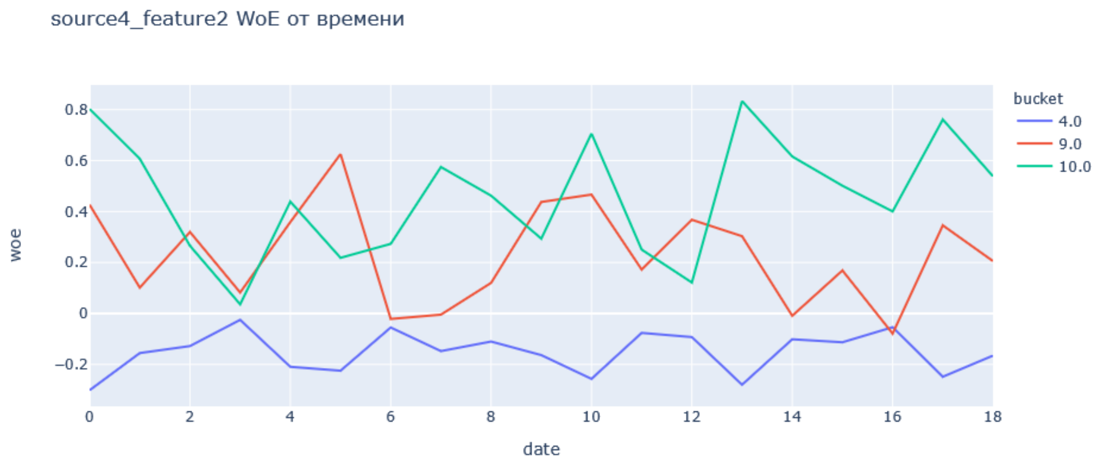
clip_source2_feature3 распределение по бакетам от времени

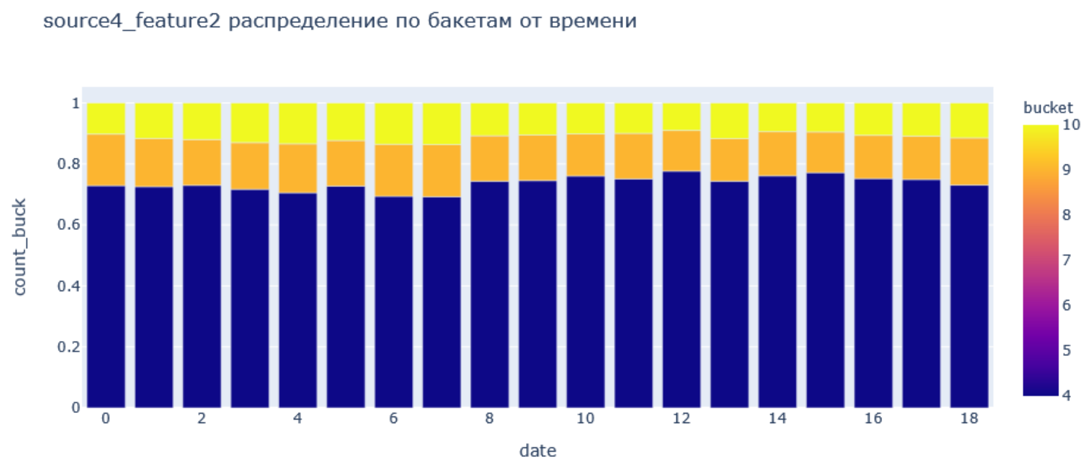


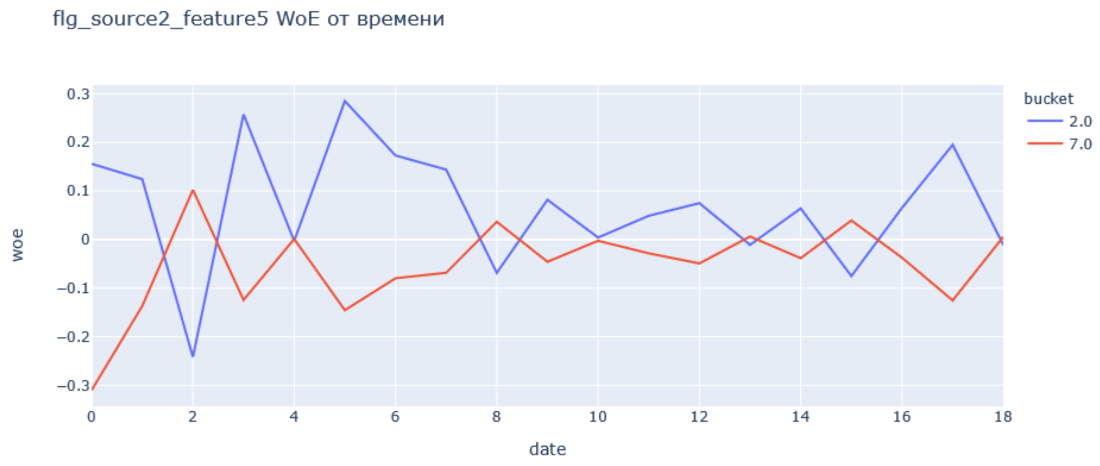
flg_source4_feature2 WoE от времени

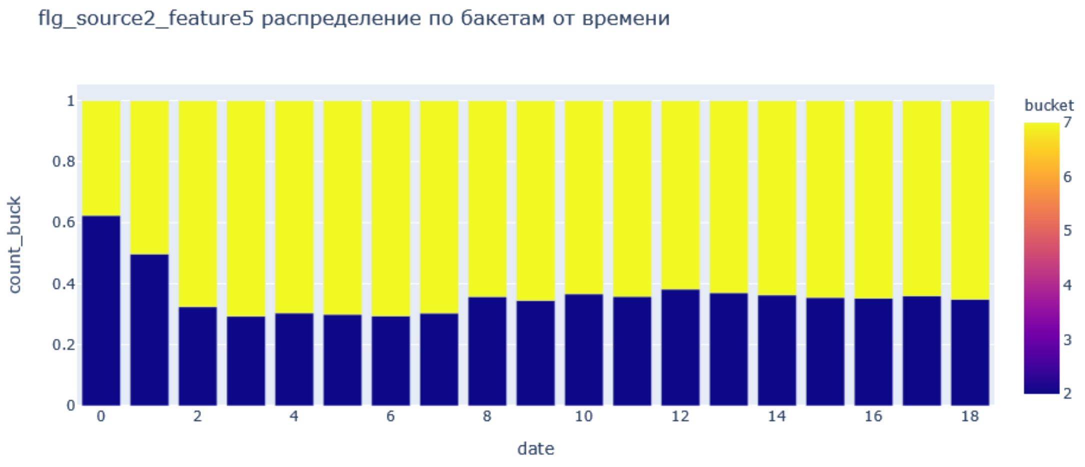


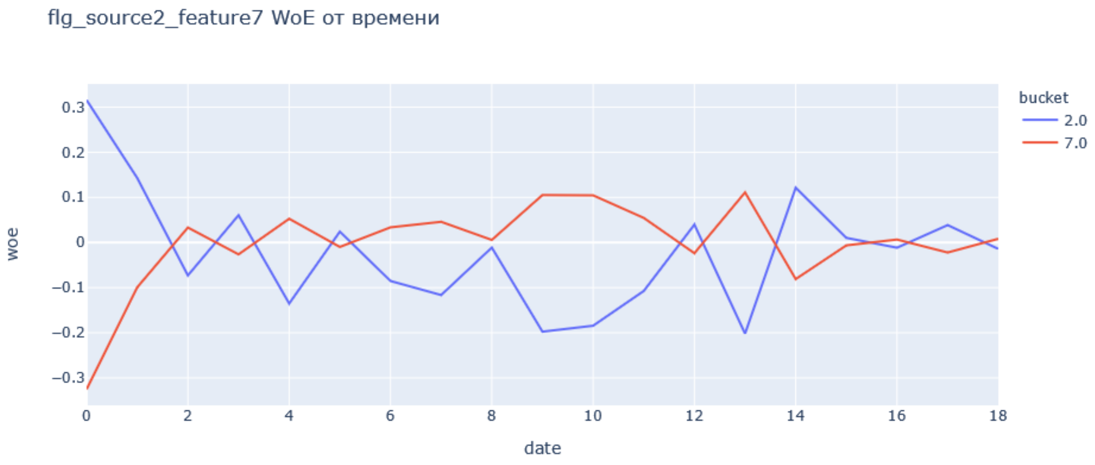

flg_source4_feature2 распределение по бакетам от времени
count_buck
date
bucket
9
8
7
6
5
4
0
0.2
0.4
0.6
0.8
1
0
2
4
6
8
10
12
14
16
18

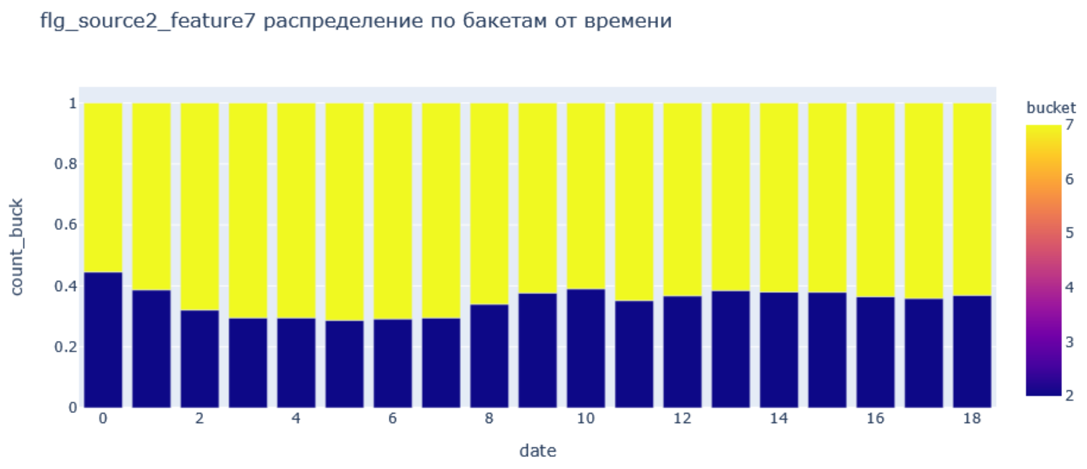

source4_feature2 WoE от времени
woe
date
bucket
4.0
9.0
10.0
0.8
0.6
0.4
0.2
0
−0.2
0
2
4
6
8
10
12
14
16
18

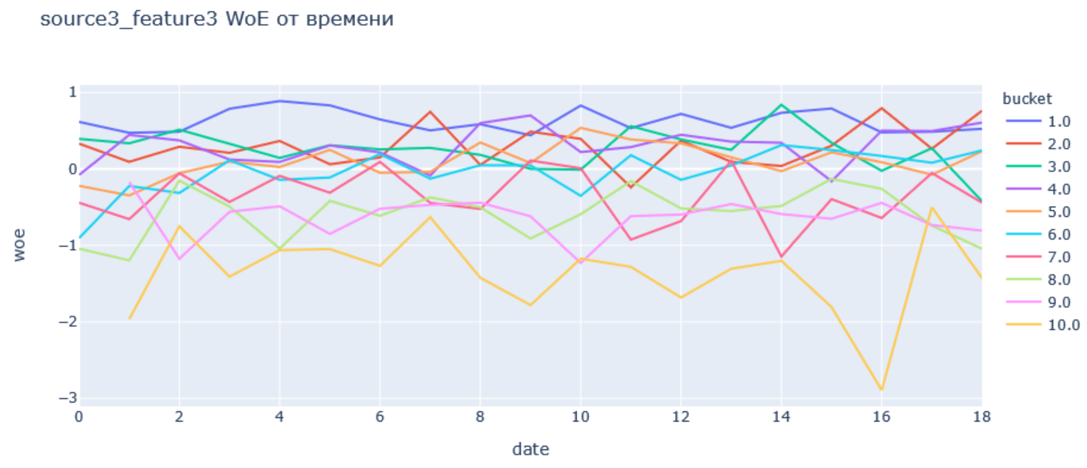

source4_feature2 распределение по бакетам от времени
count_buck
date
bucket
0
0.2
0.4
0.6
0.8
1
0
2
4
6
8
10
12
14
16
18
4
5
6
7
8
9
10


flg_source2_feature5 WoE от времени
woe
date
bucket
2.0
7.0
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
0
2
4
6
8
10
12
14
16
18

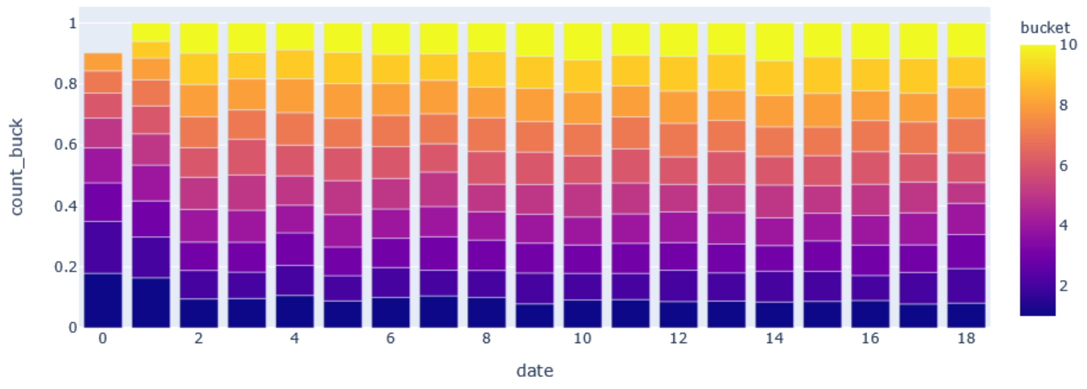

flg_source2_feature5 распределение по бакетам от времени
count_buck
date
bucket
0
0.2
0.4
0.6
0.8
1
2
4
6
8
10
12
14
16
18
3
5
7


flg_source2_feature7 WoE от времени
woe
date
bucket
2.0
7.0
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
2
4
6
8
10
12
14
16
18

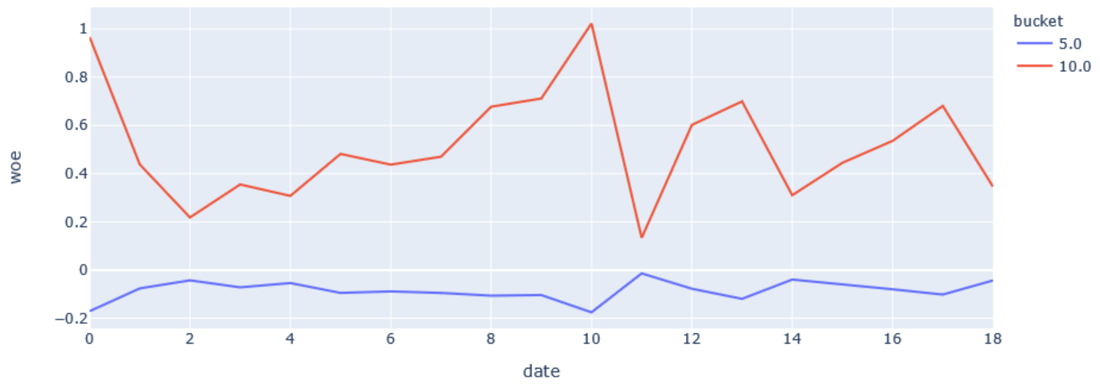

flg_source2_feature7 распределение по бакетам от времени
count_buck
date
bucket
1
0.8
0.6
0.4
0.2
0
2
3
4
5
6
7
0
2
4
6
8
10
12
14
16
18


source3_feature3 WoE от времени
woe
date
bucket
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
1
0
−1
−2
−3
0
2
4
6
8
10
12
14
16
18

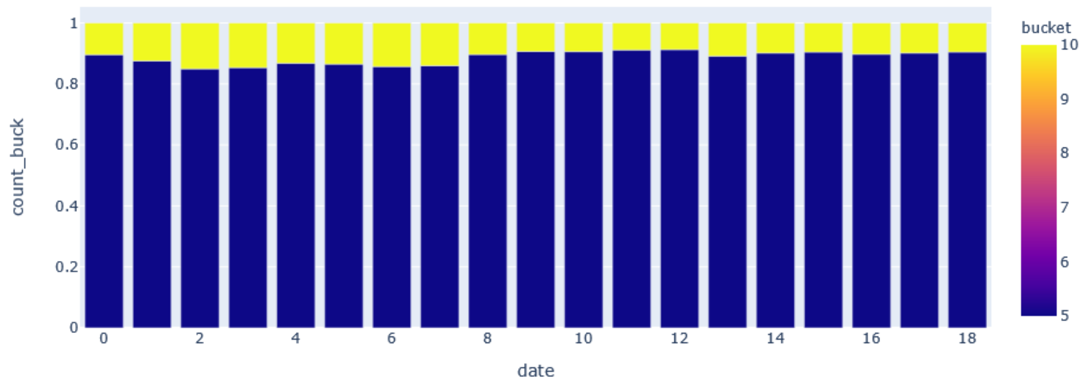
source3_feature3 распределение по бакетам от времени



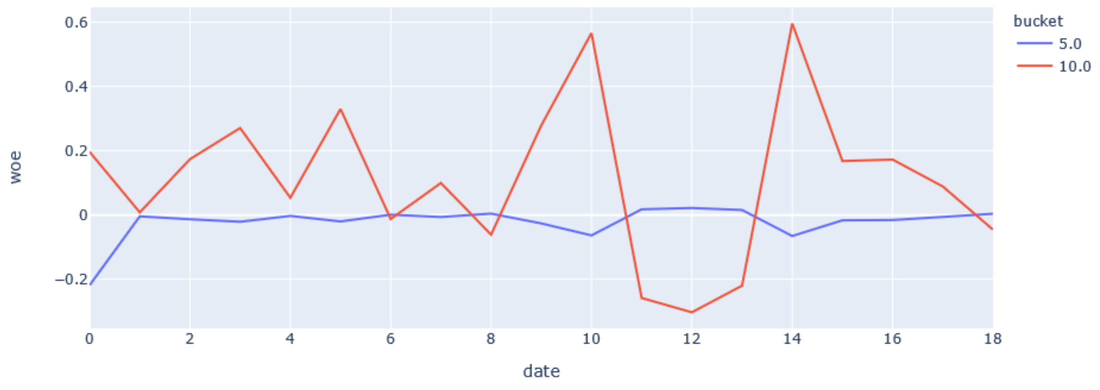
flg_source2_feature11 WoE от времени



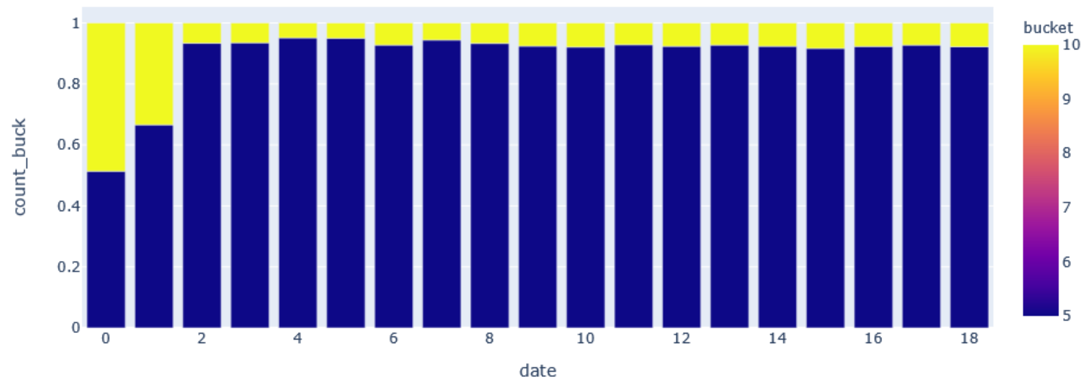
flg_source2_feature11 распределение по бакетам от времени



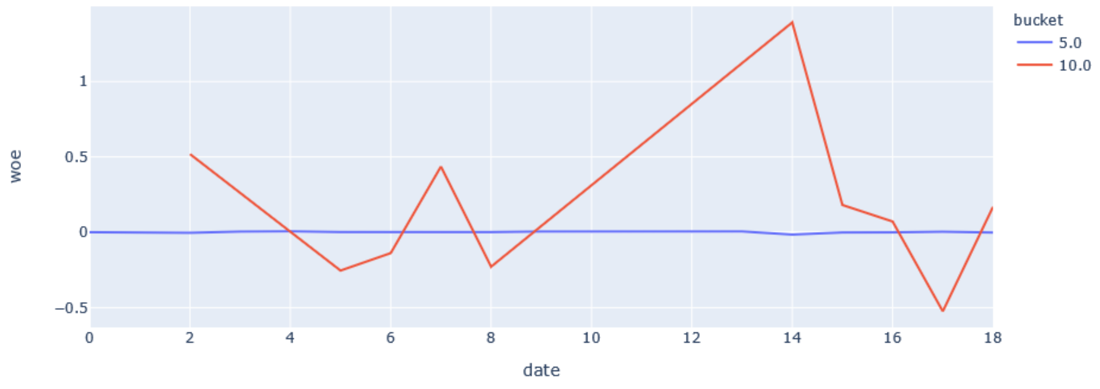
source4_feature1 WoE от времени



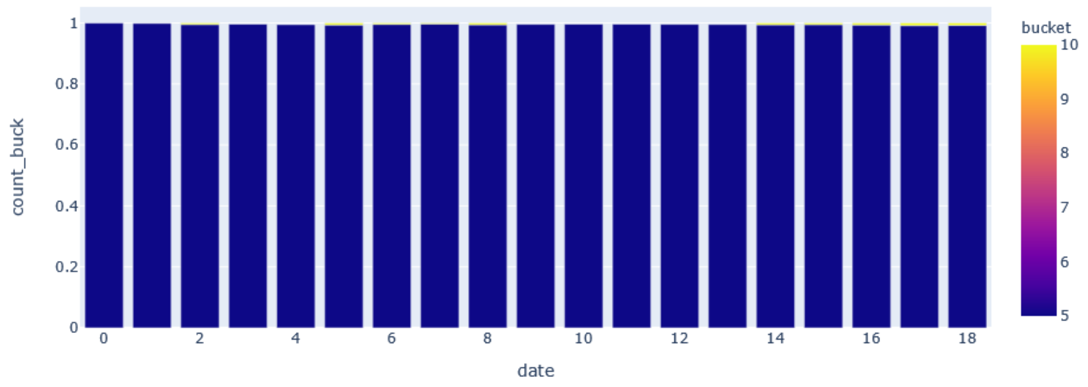
source4_feature1 распределение по бакетам от времени



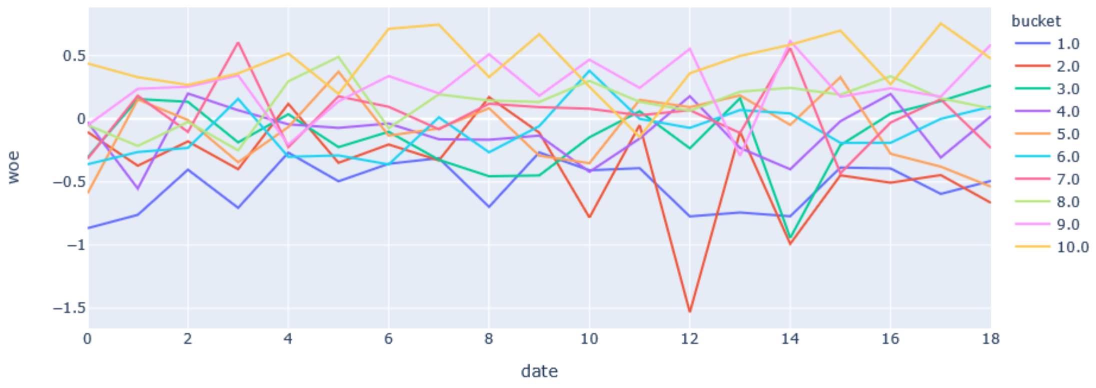
flg_source1_feature12 WoE от времени



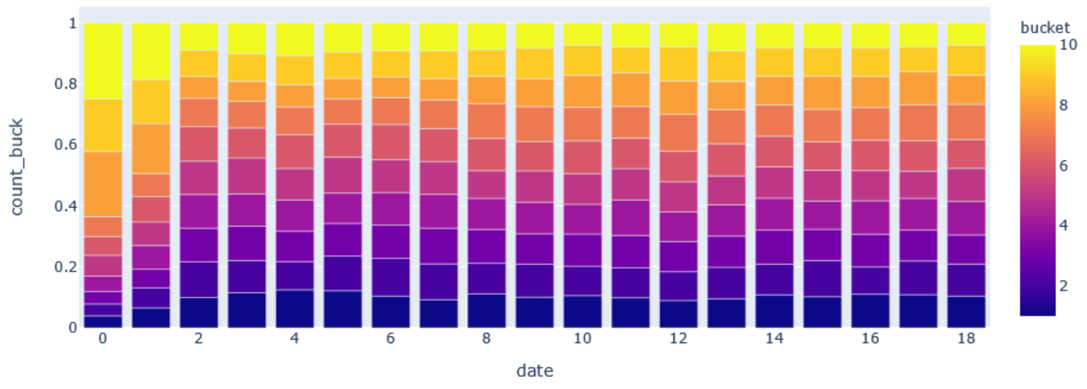
flg_source1_feature12 распределение по бакетам от времени



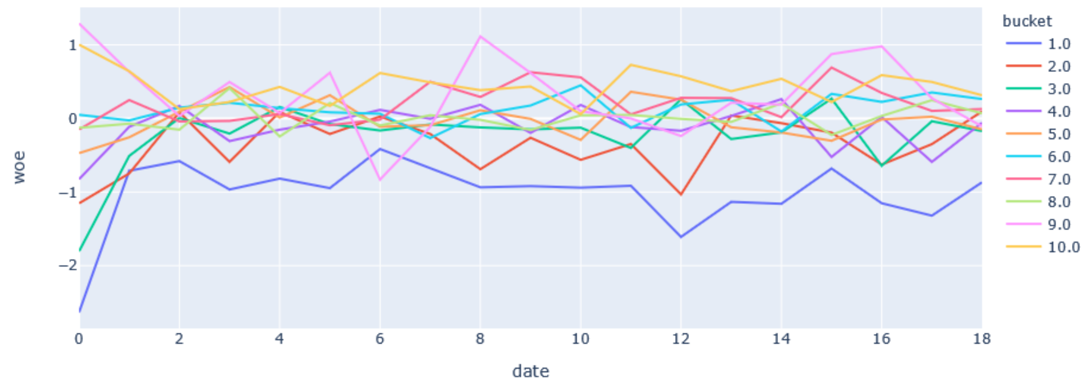
source2_feature9 WoE от времени



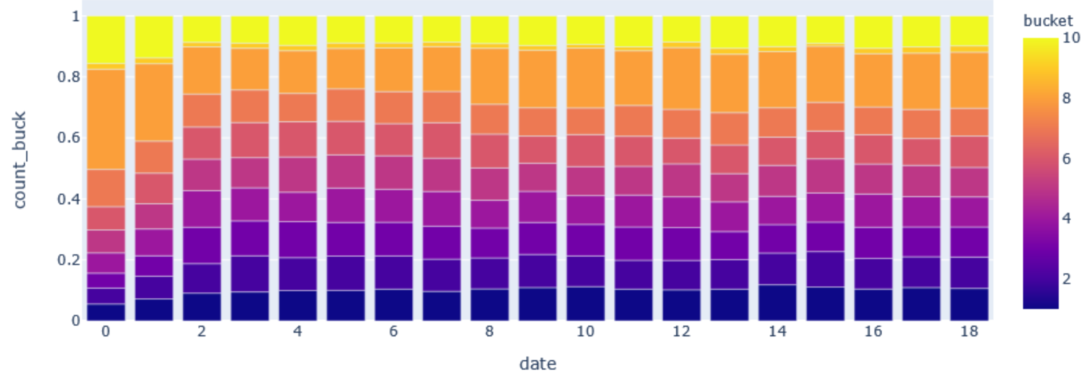
source2_feature9 распределение по бакетам от времени



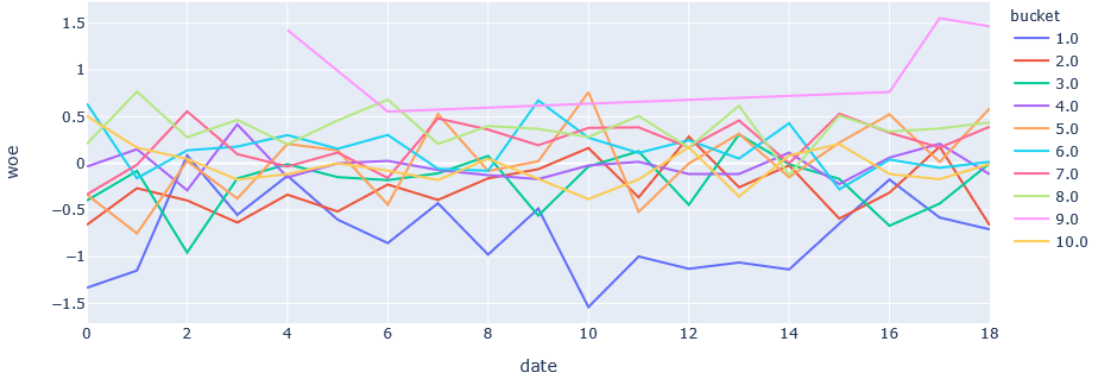
pow3_source2_feature2 WoE от времени



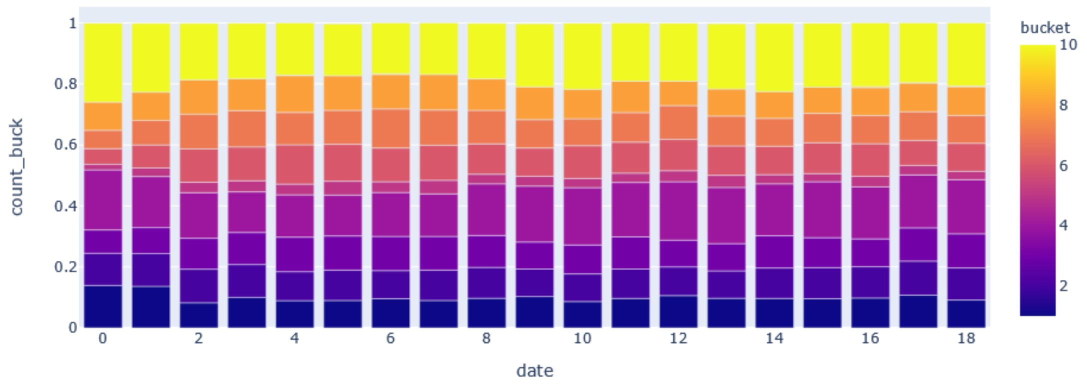
pow3_source2_feature2 распределение по бакетам от времени



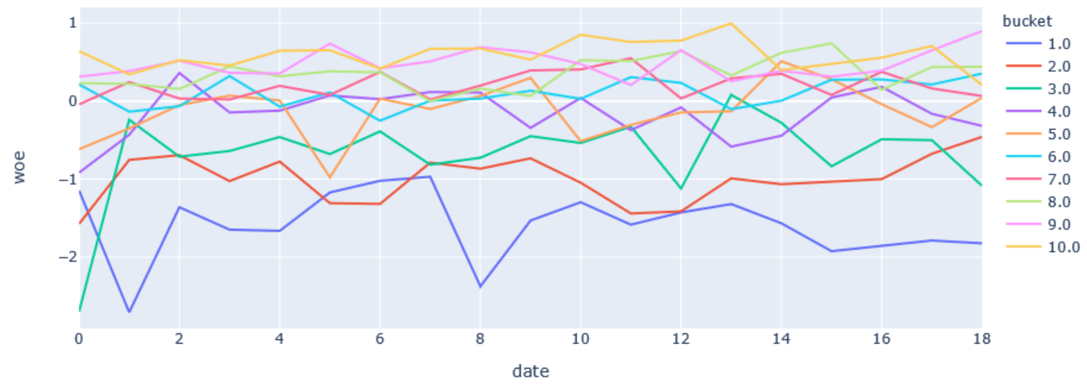
clip01_source2_feature7 WoE от времени



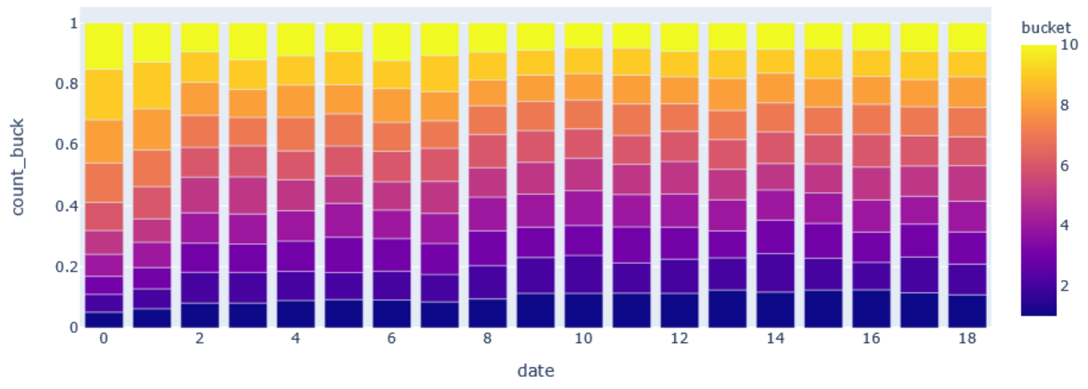
clip01_source2_feature7 распределение по бакетам от времени



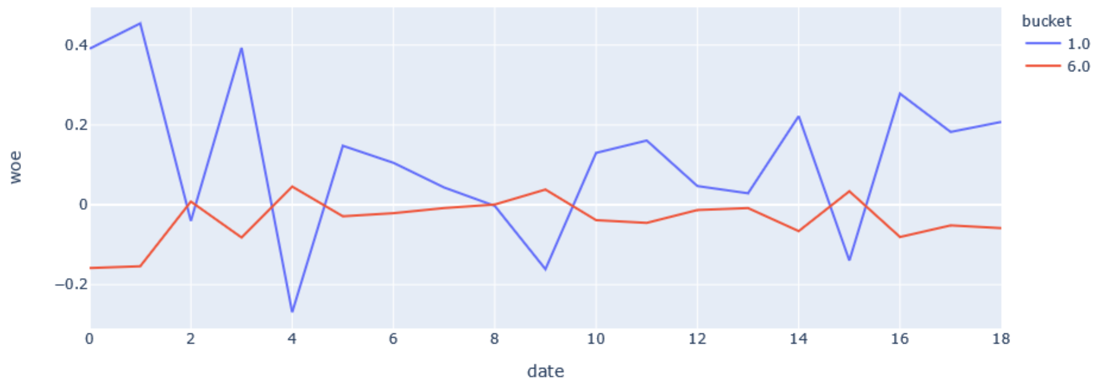
source3_feature1 WoE от времени



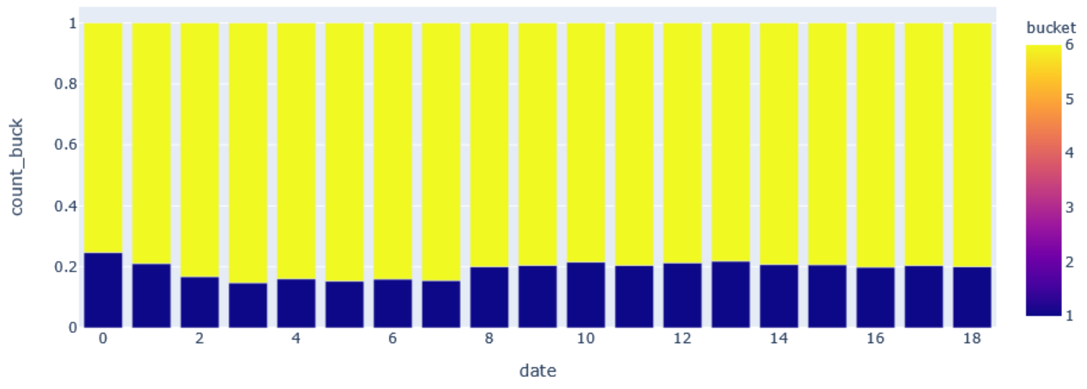
source3_feature1 распределение по бакетам от времени



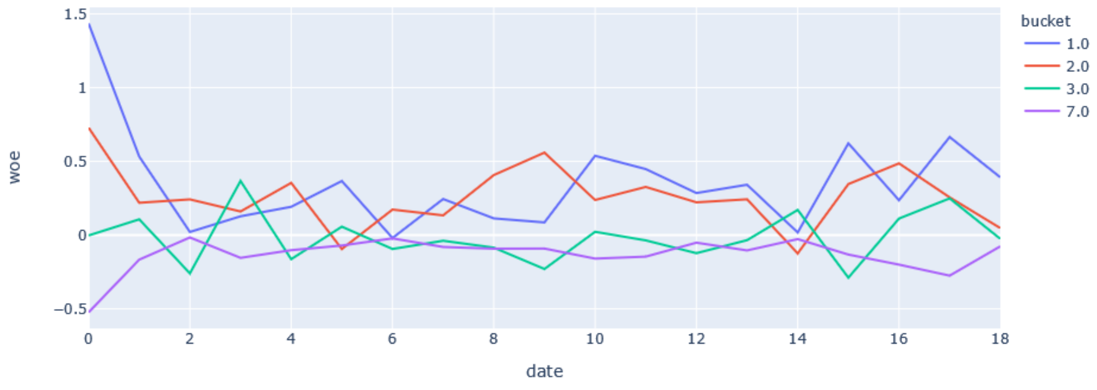
flg_source2_feature3 WoE от времени

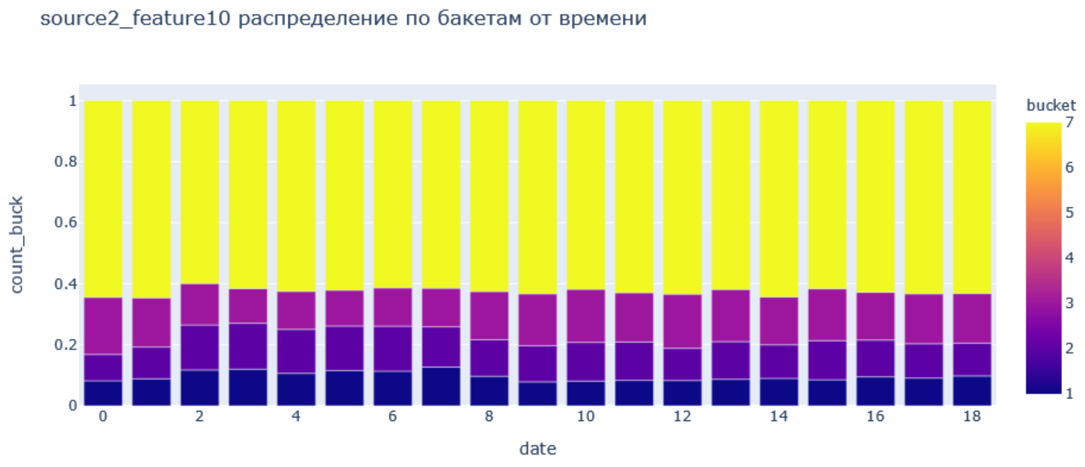


flg_source2_feature3 распределение по бакетам от времени



source2_feature10 WoE от времени

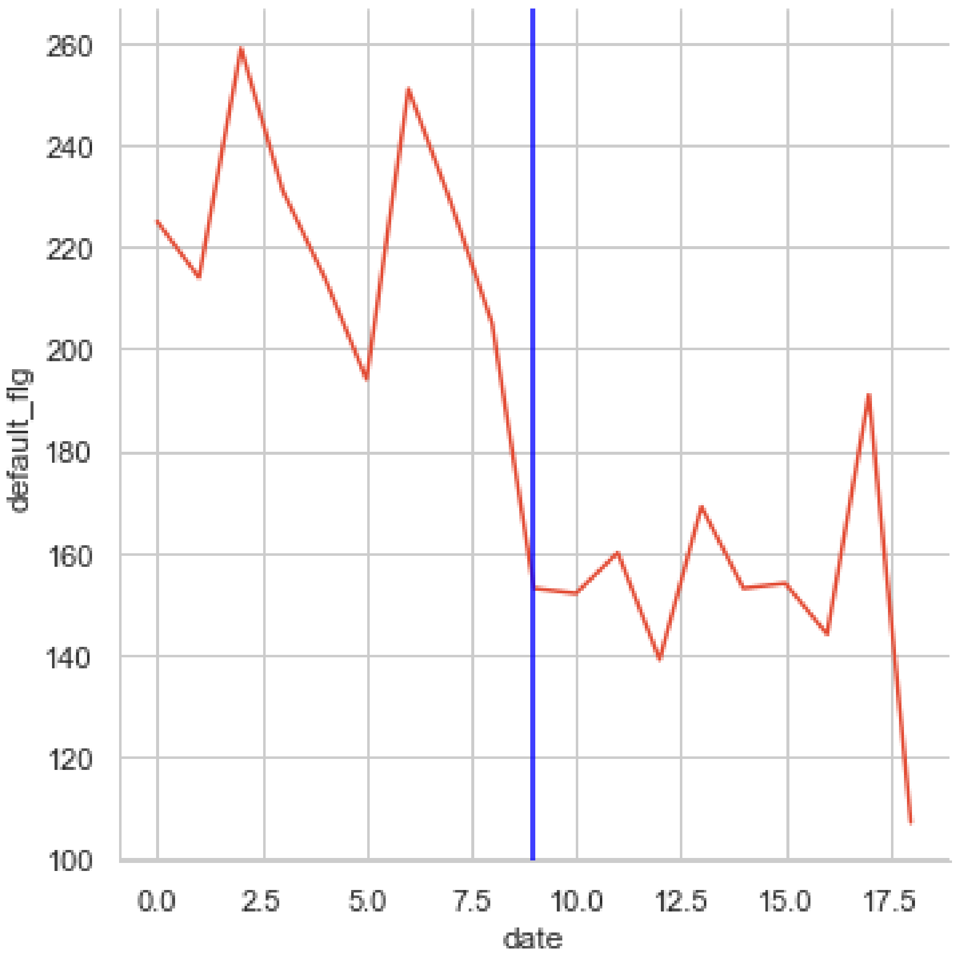




[30]:
```
unstable_non_binary = [

]
```

[31]:
```
unstable = unstable_binary + unstable_non_binary
features = list(set(features) - set(unstable))
```

[32]:
```
sns.relplot(data=train_df.groupby('date')['default_flg'].sum().reset_index(),␣
 ↪x='date', y='default_flg', kind="line")
plt.axvline(9, color='blue')
plt.show()
```



```
[33]: train_df = train_df[['id'] + features + ['default_flg', 'date']]
      test_df = test_df[['id'] + features + ['default_flg']]
```

```
[34]: train_df_early = train_df[train_df['date'] < 9]
      train_df_late = train_df[train_df['date'] >= 9]
```

```
[35]: #                          ,
      #                    train
      print('{:.1%}, {:.1%}'.format(train_df_early.default_flg.sum() /␣
       ↪len(train_df_early),
                                    train_df_late.default_flg.sum() /␣
       ↪len(train_df_late)))
```

```
11.2%, 7.8%
```

```
[36]: X_train = train_df.iloc[:, 1:-2]
      y_train = train_df.iloc[:, -2]
```

```
X_train_early = train_df_early.iloc[:, 1:-2]
y_train_early = train_df_early.iloc[:, -2]
X_train_late = train_df_late.iloc[:, 1:-2]
y_train_late = train_df_late.iloc[:, -2]

X_test = test_df.iloc[:, 1:-1]
y_test = test_df.iloc[:, -1]
```

[37]:
```
# from sklearn.model_selection import GridSearchCV

# param_grid = {
#     'C' : [0.001, 0.01, 1, 10, 100],
#     'penalty' : ['l1','l2'],
#     'solver' : ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
#     'max_iter' : [100],
#     'n_jobs' : [-1],
#     'class_weight' : ['balanced']
# }

# gridsearch = make_pipeline(StandardScaler(),␣
 ↪GridSearchCV(LogisticRegression(),
#                                                                     ␣
 ↪param_grid=param_grid,
#                                                                     scoring='roc_auc',
#                                                                     n_jobs=-1))
# start = time.time()
# gridsearch.fit(pd.concat((X_train, X_test), 0),
#                pd.concat((y_train, y_test), 0));
# end = time.time()
# print('Done in {:.1f}s'.format(end-start))
```

[38]:
```
# gridsearch[1].best_params_
```

[39]:
```
excluded_1 = []

LR = make_pipeline(
    StandardScaler(),
    LogisticRegression(
        random_state=63,
        C=0.001,
        solver='liblinear',
        penalty='l2',
        max_iter=1e3,
        n_jobs=-1,
        class_weight='balanced',
    ),
)
```

```
LR.fit(X_train, y_train)
orig = roc_auc_score(y_test, LR.predict_proba(X_test)[:,1])

LR.fit(X_train_early, y_train_early)
early_orig = roc_auc_score(y_test, LR.predict_proba(X_test)[:,1])

LR.fit(X_train_late, y_train_late)
late_orig = roc_auc_score(y_test, LR.predict_proba(X_test)[:,1])

print('test: {:.6f}, test early: {:.6f}, test late: {:.6f}'.format(orig,␣
 ↪early_orig, late_orig))

features_1 = list(set(features) - set(excluded_1))

for excluded in features_1:
    list_features = list(set(features) - set([excluded]))
    X_train_t, X_train_t_early, X_train_t_late = X_train[list_features],␣
 ↪X_train_early[list_features], X_train_late[list_features]
    X_test_t = X_test[list_features]

    LR.fit(X_train_t, y_train)
    new = roc_auc_score(y_test, LR.predict_proba(X_test_t)[:,1])

    LR.fit(X_train_t_early, y_train_early)
    early_new = roc_auc_score(y_test, LR.predict_proba(X_test_t)[:,1])

    LR.fit(X_train_t_late, y_train_late)
    late_new = roc_auc_score(y_test, LR.predict_proba(X_test_t)[:,1])

    print('{}\t test: {:.4f}, test early: {:.4f}, test late: {:.4f},\t {} - {}␣
 ↪- {}'\
          .format(excluded, new, early_new, late_new, new > orig, early_new >␣
 ↪early_orig, late_new > late_orig))
```

```
test: 0.695775, test early: 0.691366, test late: 0.699988
source3_feature2         test: 0.6888, test early: 0.6844, test late: 0.6940,
False - False - False
source1_feature9         test: 0.6943, test early: 0.6898, test late: 0.6986,
False - False - False
source4_feature2         test: 0.6956, test early: 0.6913, test late: 0.6999,
False - False - False
flg_source2_feature5     test: 0.6958, test early: 0.6920, test late: 0.6996,
False - True - False
flg_source2_feature7     test: 0.6937, test early: 0.6897, test late: 0.6979,
False - False - False
source2_feature4         test: 0.6957, test early: 0.6913, test late: 0.6998,
False - False - False
source1_feature7        test: 0.6955, test early: 0.6910, test late: 0.6999,
False - False - False
source1_feature1        test: 0.6954, test early: 0.6911, test late: 0.6996,
False - False - False
source2_feature8        test: 0.6953, test early: 0.6906, test late: 0.6998,
False - False - False
pow3_source2_feature2   test: 0.6951, test early: 0.6910, test late: 0.6993,
False - False - False
source1_feature8        test: 0.6957, test early: 0.6914, test late: 0.6999,
False - False - False
clip01_source2_feature7 test: 0.6897, test early: 0.6846, test late: 0.6946,
False - False - False
source2_feature10       test: 0.6926, test early: 0.6879, test late: 0.6971,
False - False - False
source1_feature6        test: 0.6955, test early: 0.6908, test late: 0.7001,
False - False - True
clip_source2_feature3   test: 0.6954, test early: 0.6915, test late: 0.6996,
False - True - False
flg_source4_feature2    test: 0.6960, test early: 0.6917, test late: 0.7004,
True - True - True
source1_feature11       test: 0.6987, test early: 0.6955, test late: 0.6997,
True - True - False
source2_feature1        test: 0.6955, test early: 0.6913, test late: 0.6998,
False - False - False
flg_source2_feature11   test: 0.6958, test early: 0.6914, test late: 0.7001,
True - True - True
source3_feature3        test: 0.6899, test early: 0.6853, test late: 0.6946,
False - False - False
source1_feature3        test: 0.6950, test early: 0.6908, test late: 0.6988,
False - False - False
source4_feature1        test: 0.6958, test early: 0.6914, test late: 0.7001,
True - True - True
flg_source1_feature12   test: 0.6958, test early: 0.6914, test late: 0.7000,
False - True - False
source2_feature9        test: 0.6922, test early: 0.6881, test late: 0.6960,
False - False - False
source1_feature2        test: 0.6919, test early: 0.6878, test late: 0.6959,
False - False - False
source1_feature10       test: 0.6957, test early: 0.6917, test late: 0.7005,
False - True - True
source1_feature5        test: 0.6962, test early: 0.6918, test late: 0.7004,
True - True - True
source3_feature1        test: 0.6936, test early: 0.6886, test late: 0.6979,
False - False - False
```

[40]:
```
final_features = list(set(features) - set([]))

X_train_late = X_train_late[final_features]
X_test = X_test[final_features]
```

[41]:
```
X_test = X_test.rename(columns = {
    'clip_source2_feature3'   :  'source2_feature3',
    'pow3_source2_feature2'   :  'source2_feature2',
    'clip01_source2_feature7' :  'source2_feature7',
    'flg_source2_feature11'   :  'source2_feature11',
    'flg_source1_feature12'   :  'source1_feature12',
    'flg_source4_feature2'    :  'source4_feature2',
    'flg_source2_feature5'    :  'source2_feature5',
})
```

[42]:
```
from sklearn.preprocessing import RobustScaler, StandardScaler

LogReg = make_pipeline(
    StandardScaler(),
    LogisticRegression(
        random_state=63,
        C=0.002,
        solver='liblinear',
        penalty='l2',
        max_iter=1e3,
        n_jobs=-1,
        class_weight='balanced',
   ),
)

LogReg.fit(X_train_late, y_train_late)
test_df['default_pred'] = LogReg.predict_proba(X_test)[:,1]
orig = roc_auc_score(y_test, test_df['default_pred'])
print(orig)
```

```
0.7003234047647333
```

[43]:
```
y_pred = test_df['default_pred'].values #

df_model = pd.DataFrame(data = {'    ':['ROC_AUC','log_loss'], '     ':[
    roc_auc_score(np.array(test_df.default_flg), y_pred),
    log_loss(np.array(test_df.default_flg), y_pred)
]}).set_index('    ')

df_model
```

[43]:

```
ROC_AUC   0.700323
log_loss  0.641166
```

[44]:
```
df = pd.DataFrame({'feature' : X_test.columns,
                   'coef_model' : LogReg[1].coef_[0]})
df
```

[44]:
```
                 feature  coef_model
0        source3_feature2    0.154649
1        source1_feature9   -0.079654
2        source4_feature2    0.045935
3        source2_feature5    0.039058
4    flg_source2_feature7    0.137163
5        source2_feature4    0.018585
6        source1_feature7   -0.102282
7        source1_feature1    0.044024
8        source2_feature8   -0.140150
9        source2_feature2    0.086169
10       source1_feature8   -0.032685
11       source2_feature7    0.203512
12      source2_feature10   -0.131327
13       source1_feature6   -0.013287
14       source2_feature3   -0.091921
15       source4_feature2    0.022869
16      source1_feature11   -0.016674
17       source2_feature1   -0.086870
18      source2_feature11    0.037873
19       source3_feature3   -0.216759
20       source1_feature3    0.044928
21       source4_feature1    0.011008
22      source1_feature12    0.013549
23       source2_feature9    0.141897
24       source1_feature2    0.097812
25      source1_feature10    0.043731
26       source1_feature5   -0.062371
27       source3_feature1    0.237317
```

[ ]:
```

```